NASA/TM—2005-213588 ARL–MR–0611

Foreign Object Damage Identification in Turbine Engines

William Strack
N&R Engineering and Management Services Corporation
Parma Heights, Ohio

Desheng Zhang
AYT Corporation, Brook Park, Ohio

James Turso
QSS Group, Inc., Cleveland, Ohio

William Pavlik
RS Information Systems, Inc., Cleveland, Ohio

Isaac Lopez
U.S. Army Research Laboratory, Glenn Research Center, Cleveland, Ohio

March 2005

NASA/TM—2005-213588 ARL–MR–0611

Foreign Object Damage Identification in Turbine Engines

William Strack
N&R Engineering and Management Services Corporation
Parma Heights, Ohio

Desheng Zhang
AYT Corporation, Brook Park, Ohio

James Turso
QSS Group, Inc., Cleveland, Ohio

William Pavlik
RS Information Systems, Inc., Cleveland, Ohio

Isaac Lopez
U.S. Army Research Laboratory, Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center

March 2005

Trade names or manufacturers' names are used in this report for identification only. This usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

Foreign Object Damage Identification in Turbine Engines

William Strack
N&R Engineering & Management Services Corporation
Parma Heights, Ohio 44130

Desheng Zhang
AYT Research Corporation
Cleveland, Ohio 44135

James Turso
QSS Group, Inc.
Cleveland, Ohio 44135

William Pavlik
RS Information Systems, Inc.
Cleveland, Ohio 44135

Isaac Lopez
U.S. Army Research Laboratory
Glenn Research Center
Cleveland, Ohio 44135

Summary

This report summarizes the collective work of a 5-person team from different organizations examining the problem of detecting foreign object damage (FOD) events in turbofan engines from gas path thermodynamic and bearing accelerometer sensors, and determining the severity of damage to each component (diagnosis). Several detection and diagnostic approaches were investigated and a software tool (FODID) was developed to assist researchers detect/diagnose FOD events. These approaches include: (1) fan efficiency deviation computed from upstream and downstream temperature/pressure measurements, (2) gas path weighted least squares estimation of component health parameter deficiencies, (3) Kalman filter estimation of component health parameters, and (4) use of structural vibration signal processing to detect both large and small FOD events. The last three of these approaches require a significant amount of computation in conjunction with a physics-based analytic model of the underlying phenomenon -- the NPSS thermodynamic cycle code for approaches 1-3 and the DyRoBeS reduced-order rotor dynamics code for approach 4.

The direct fan efficiency approach is appealing because it is simple to apply. However, it inherently assumes zero measurement errors and therefore its usefulness is limited to FOD events where the magnitude of the fan efficiency change is large enough to overcome sensor noise. This limitation does not apply to the extended weighted least squares and extended Kalman filtering methods that are more general but also more complex mathematically. The vibration analysis offers a completely different approach than the other 3 and has the potential to detect small events that do not cause detectible permanent damage. Data fusion of the gas path and vibration results provides an opportunity to obtain a more reliable diagnosis that either approach used separately.

A potential application of the FODID software tool, in addition to its detection/diagnosis role, is using its sensitivity results to help identify the best types of sensors and their optimum locations within the gas path, and similarly for bearing accelerometers.

Introduction

The ingestion of foreign objects into aircraft engines has caused damage that ranges from negligible to catastrophic. It is rather difficult to determine the severity of damage by visual inspection (except in catastrophic events) and even identifying which components have been damaged without an engine removal/tear-down is challenging. Consequently, airline operators are forced to make difficult and economically sensitive decisions unless they apply some form of FOD detection and diagnostic procedures. The major engine manufacturers have been active in this field for several decades (refs. 1-10). Their investigations have revealed that solving the diagnostic problem accurately is more complex than it might seem at first glance and many valuable lessons have been learned in this regard (see Appendix A). However, the details of their current methodologies and implementation tools are competition sensitive and therefore generally unavailable. NASA has had some ongoing effort in recent years – mostly theoretical research to develop better diagnostic methodologies (refs. 11-14).

The effort reported herein extends the NASA-sponsored effort in several new directions. Firstly, it examines the possibility of simply using upstream/downstream pairs of temperature and pressure sensors around the most likely FOD-damaged component (the fan) to determine the degree of damage. Secondly, it performs a comparative investigation of the weighted least-squares and Kalman filter gas path methodologies. Thirdly, an integrated NPSS/MATLAB system was established in which the NPSS thermodynamic cycle code provides input data to MATLAB/SIMULINK to perform a state-space control analysis for FOD event detection. The MATLAB/SIMULINK output is then retrieved by NPSS for simulation adjustment and further analysis.

Fourthly, an investigation of using dynamic structural response measurements to detect FOD events was pursued. This led to the use of wavelet signal processing to unmask the FOD features hidden by vibration sensor noise for small-sized events that produce only temporary (i.e., millisecond) behavior aberrations. Such events may not cause immediate failures or performance degradation because the geometry is not altered, but may cause premature parts failure later on due to internal damage.

Fifthly, a fuzzy logic framework to fuse the results of gas path and vibration analyses together into a composite FOD diagnosis was established. The initial investigations of the wavelet analysis of the structural response and the fuzzy logic data fusion technique were carried out using MATLAB/SIMULINK, but were later implemented in the FODID software tool.

Lastly, a software tool (FODID) was developed that can be used to detect and diagnose FOD events using a variety of methodologies (gas path and vibration analyses). Specifically, it addresses the issue of determining how severely each component is damaged given a set of thermodynamic sensor measurements and associated uncertainties. Accounting for system and sensor uncertainties is important since their magnitude can be of the same order as the FOD-induced effects. The tool is intended to be used initially in a research role to explore alternative detection/diagnostic strategies, but ultimately it could morph into a prototype production tool for commercial use.

Ostensibly, it should be easy to detect an FOD event during flight and determine the degree of damage incurred. Deviations in component health parameters such as efficiency and flow capacity from their nominal values are all that is required. Unfortunately, there are no means to directly measure such health parameters (i.e., efficiency and flow capacity sensors do not exist). Instead, they must be inferred from measurements of other parameters such as shaft speed, fuel flow rate, and temperatures and pressures at various gas path flow stations. The inference process to accomplish this is known as a “gas path analysis” and is summarized in Figure 1. The inference methodology generally utilizes a thermodynamic model of the engine which is not perfectly accurate, nor are the sensor measurements due to sensor errors and noise. The modeling and sensor errors can seriously compromise the accuracy of the diagnostic solution since they may be of the same magnitude as the sensed deviations. Hence, some means to cope with these uncertainties is usually needed to obtain a reasonably confident diagnosis.

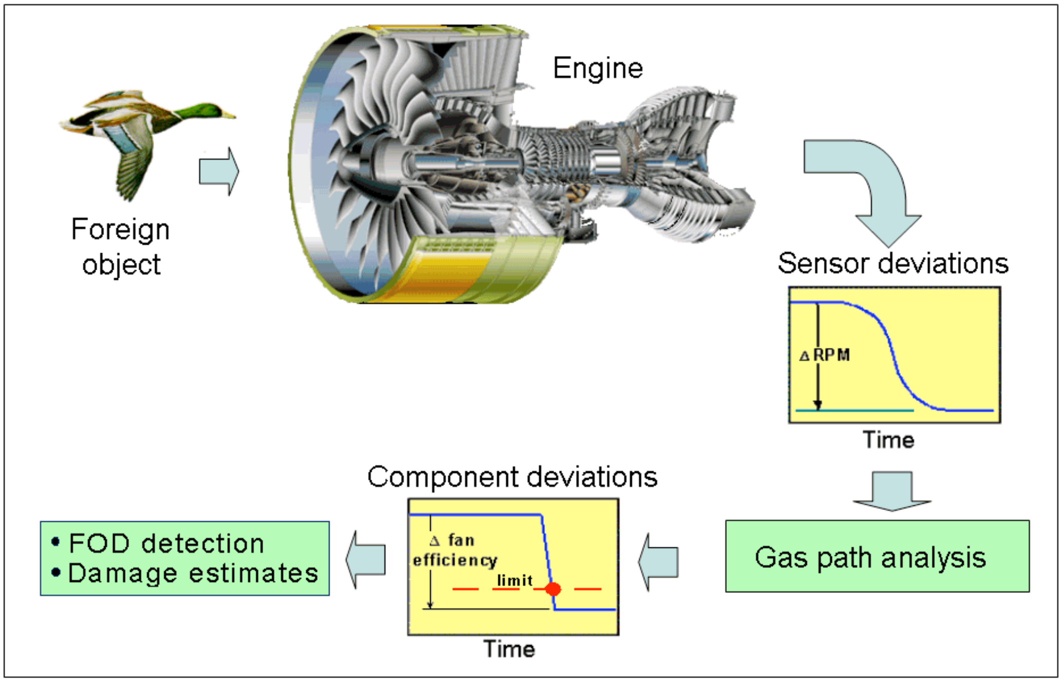


Figure 1 – The overall gas path analysis process to detect/diagnose FOD events.

A direct approach to detecting an FOD event

A direct approach to detecting an FOD event that damages the fan compares the actual fan adiabatic efficiency to a reference fan efficiency using known fan inlet and exit average temperatures and pressures. Adiabatic fan efficiency η_{adib} is defined as the ratio of the ideal enthalpy rise to the actual enthalpy rise through the fan:

$$\eta_{adib} = (H_{ideal,out} - H_{in}) / (H_{out} - H_{in}) \tag{1}$$

where,

$$\begin{aligned} H_{in} &= \text{entrance enthalpy} &&= \text{fct}\,(T_{in},\ P_{in}) \\ H_{out} &= \text{actual exit enthalpy} &&= \text{fct}\,(T_{out},\ P_{out}) \\ H_{ideal,out} &= \text{ideal exit enthalpy} &&= \text{fct}\,(H_{in},\ P_{out}) \end{aligned}$$

where T_{in} is the measured entrance temperature, P_{in} is the measured entrance pressure, T_{out} is the measured exit temperature, and P_{out} is the measured exit pressure. If the constant-pressure specific heat is invariant (which is nearly the case for a high bypass turbofan), then equation 1 becomes even simpler since it does not require enthalpy tables:

$$\eta_{adib} = (T_{ideal,out} - T_{in}) / (T_{out} - T_{in}) \tag{2}$$

Or, in terms of the actual bulk-average pressures and temperatures, and average specific heat ratio, γ:

$$\eta_{adib} = \frac{\left(\frac{P_{out}}{P_{in}}\right)^{\frac{\gamma-1}{\gamma}} - 1}{\frac{T_{out}}{T_{in}} - 1} \tag{3}$$

So using either equation 1 in conjunction with enthalpy tables to obtain an exact value, or using equation 3 without enthalpy tables to obtain a very good approximation, one can readily determine the actual adiabatic efficiency. An important caveat is that these relationships require the use of a bulk-average temperatures and pressures. Since both temperature and pressure vary significantly radially from hub to tip, one would need to locate the sensors carefully to obtain values that represent bulk-averages.

All that remains is to compare the actual adiabatic efficiency with a reference adiabatic efficiency. If the engine is operating in a steady-state condition such as at cruise, one could simply monitor a time trace of η_{adib} to detect significant deviations. That is, the chosen reference value $\eta_{adib,ref}$ could be a time-averaged prior value. But FOD events would most likely occur at a transient condition such as during takeoff in which case a previous time average is not meaningful. Instead, $\eta_{adib,ref}$ ought to be the expected value as calculated from a calibrated thermodynamic cycle model such as provided by NPSS. "Calibrated" here implies that a standard engine model is adjusted slightly for each specific engine based on performance qualification testing data. Then, one could monitor the ratio r of the actual to reference adiabatic efficiency even if the operating conditions are continuously changing:

$$\text{Fan efficiency factor} \equiv \; r = \frac{\eta_{adib}}{\eta_{adib,ref}} \qquad (4)$$

Figure 2 illustrates what such a 100-point averaged time trace might look like for a FOD event that occurs at time step 2500.

The same methodology can possibly be extended to the other turbomachinery components. However, there is a complicating factor in applying this concept to these components; namely, downstream component input conditions are affected by upstream components. As an example, the input temperature and input pressure for the LPC will be affected by any fan efficiency changes. Hence, η_{adib} will not equal $\eta_{adib,ref}$ even if the LPC is operating normally (undamaged) if the fan is damaged.

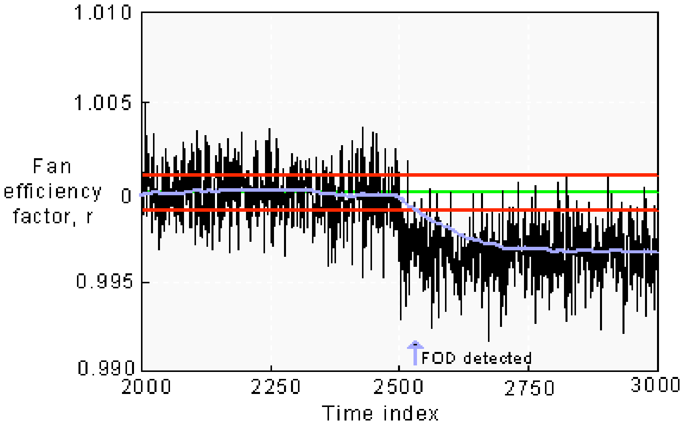


Figure 2 – Fan efficiency factor time trace with an FOD event at time 2500.

To cope with more general situations that involve possible damage to multiple components, and that account for more realistic effects (i.e., system and sensor noise), we pursued more comprehensive methodologies as discussed in the following sections.

Least-squares approximation (LSA)

Our pursuit of a more comprehensive methodology begins by casting our FOD problem as a problem in linear algebra. By perturbing the values of the health parameters (independents) in a thermodynamic engine model and recording the corresponding output values of the sensor measurements (dependents), we can use a least-squares approximation to establish the relationships (response curves) between the independent and dependent variables. We can use these response curves to fit observed measurements and estimate the values of sought-after health parameters. The NPSS thermodynamic cycle code was used to perform the numerical calculations cited below.

Fitting curves to given sets of data points is an important practical application of linear algebra. Assume that we have *k* independent variables $x_i,\ i = 1,\cdots,k$ and *m* sets of data points $x_{ij},\ i = 1,\cdots,\ k,\ j = 1,\cdots,m$ where x_{ij} is the value of the dependent variable x_i at the *j*'th measurement. We would like to determine a function $F(x_1,\cdots,x_k)$ such that

$$y_i = F(x_{1i},...,x_{ki}) + \varepsilon_i,\ i = 1,...,m \tag{5}$$

where the approximation errors ε_i are small. We assume that $F(x_1,\cdots,x_k)$ is a linear sum of *n* sub-functions $f_i(x_1,\cdots,x_k)$ with the form

$$F(x_1,\cdots,x_k) = c_0 + c_1 f_1(x_1,\cdots,x_k) + \cdots + c_n f_n(x_1,\cdots,x_k) \tag{6}$$

We wish to determine the set of coefficients c_i so that $F(x_1,\cdots,x_k)$ best fits the *m* measured data points.

If we define

$$A = \begin{pmatrix} 1 & f_1(x_{11},...,x_{x1}) & \cdots & f_n(x_{11},...,x_{x1}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f_1(x_{1m},...,x_{xm}) & \cdots & f_n(x_{1m},...,x_{xm}) \end{pmatrix},\ c = (c_i) \text{ and } y = (y_i),\ i = 1,\cdots,m \tag{7}$$

then the predicted value of *y* is:

$$y_{pre} = Ac \tag{8}$$

and the *m*-vector ε of approximation errors is:

$$\mathring{a} = y_{pre} - y = Ac - y \tag{9}$$

The least-squares solution for ε is:

$$A^T Ac = A^T y \tag{10}$$

where A^TA is a symmetric positive-definite matrix that is not singular. The coefficient vector c can be uniquely solved by LUP decomposition. If the curve crosses the origin, then c_0 is zero and the left-most column should be deleted from matrix A.

We investigated the application of this solution method to a mid-size separate flow, 6:1 bypass ratio turbofan engine similar to a PW2037 whose control system maintains constant thrust by adjusting the combustor fuel/air ratio. Six health parameters and five sensor parameters were selected. The six health parameters and their baseline values at the sea level static/takeoff thrust operating point were:

• Fan efficiency, η	0.890
• Fan flow capacity, W_c	1345 lb_m/sec
• LPC efficiency, η	0.900
• LPC flow capacity, W_c	192 lb_m/sec
• HPC efficiency, η	0.865
• HPC flow capacity, W_c	192 lb_m/sec

The five sensor parameters and their baseline values were:

• LP rotor speed, N_1	4550 RPM
• HP rotor speed, N_2	12360 RPM
• LPC exit temperature, $T_{2.5}$	718 ºR
• HPT inlet temperature, T_{41}	2932 ºR
• Exhaust temperature, T_{49}	1353 ºR

To reveal whether the model is linear about the operating point, we perturbed each health parameter independently and recorded the changes in the sensor parameters. While the sensor parameters show good linear relationship with all three efficiency parameters (Fig. 3), they are in at least a quadratic relationship with the three flow capacity parameters (Fig. 4).

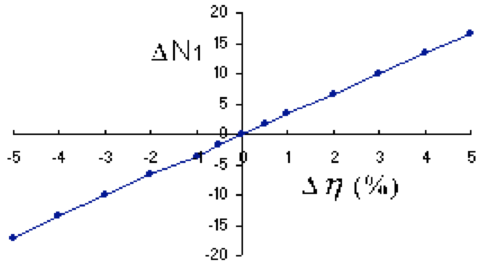


Figure 3 - N_1 change versus fan efficiency change

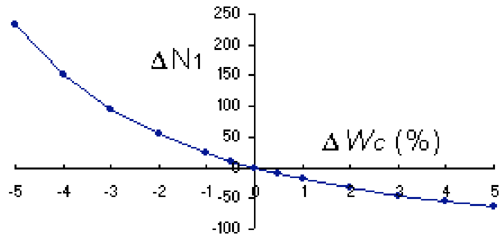


Figure 4 - N_1 change versus fan flow capacity change

We next studied the interactions between health parameters. By perturbing two health parameters together it was observed that there exist interactions between some pairs. For example, there is a 13 percent interaction for N_2 between fan flow capacity and HPC efficiency when they are both perturbed 5 percent simultaneously. (Interaction here refers to the difference in sensor values obtained by perturbing multiple health parameters simultaneously compared to perturbing them sequentially.) These observations (non-linear flow capacity response and significant interactions) suggest that the response curves obtained via least-squares approximation should contain quadratic terms including cross-terms. Also the response curves must cross the origin (zero health parameter perturbation causes no change in measurements). Hence, equation 6 can be replaced with a more general non-linear form for our specific case of 6 health and 5 sensor parameters:

$$g_k = \sum_{i=1}^{6} c_{ki} x_i + \sum_{i=1}^{6} c_{k(i+6)} x_i^2 + \sum_{i=1}^{5} \sum_{j=i+1}^{6} c_{k(6+j+i(11-i)/2)} x_i x_j \qquad k = 1,2\ldots5 \qquad (11)$$

where $x_1, x_2, \cdots, x_6$ are percentage changes of the six health parameters and g_k are the sensor parameter changes.

The levels of $x_1, x_2, \cdots, x_6$ used to perturb the health parameters are –5, -3.5, -2.0, -0.5, 0.5, 2.0, 3.5, 5. If full-factorial experiments were designed, there would be 8^6 = 262,144 cases to run. The number would become prohibitively high if more health parameters were included. We would prefer to reduce the number of experiments without sacrificing too much fidelity compared to using a full-factorial experiment. One way to dramatically reduce the number of cases while preserving reasonable fidelity is to randomly select a representative subset of the full-factorial experiments. The second way is to determine which health parameter pairs cause very weak interactions so that their interactions can be excluded from the set of experiments. The third way is to reduce the number of levels for some health parameters. For example, the levels for the three efficiency parameters can all be reduced due to their linear effect on the sensor parameters (Fig. 3).

In this study, we adopted the first approach. Approximately 2000 health parameter combinations were randomly selected and the response curves for the five sensor parameters were generated. To check the accuracy of the resulting response curves, their predictions can be compared to the values generated by running the NPSS code. For instance, based on –3 percent change in fan efficiency, the following predictions and NPSS solutions are obtained for the 5 sensor parameters:

	g1	*g2*	*g3*	*g4*	*g5*
Equation 3 predictions:	–12.4835	57.7034	41.4279	-0.75890	20.6195
NPSS calculations:	–10.1274	51.1341	41.3033	-0.65334	20.6559

While not perfect, these predictions are judged to be sufficiently accurate for most purposes. To further illustrate this method and check its accuracy, we now consider a more challenging problem; namely, estimate the efficiency changes in the fan, LPC, and HPC due to various sets of deviations in N_1, N_2, and T_{49}. First, we solve for the response curve coefficients using the least-squares approximation technique described above using NPSS and efficiency perturbations ranging from -5 to +5 percent. This yields the coefficient values shown in Table 1.

Table 1 – Example response curves generated with equation 11.

$$g(x,y,z) = c_1 x + c_2 y + c_3 z + c_4 x^2 + c_5 y^2 + c_6 z^2 + c_7 xy + c_8 xz + c_9 yz$$

where x = fan efficiency, y = LPC efficiency , z = HPC efficiency

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
N_1	3.35718	0.52324	0.89534	-0.00967	-0.00605	-0.00111	0.00922	0.02066	0.00050
N_2	-16.82756	-10.89637	16.83345	0.30459	0.03298	0.11904	-0.08686	-0.66856	0.04896
T_{49}	-6.29113	-2.58067	-11.40829	0.13568	0.06123	0.35265	-0.00834	0.01526	0.05570

Next, an NPSS FOD detection model was created[1] in order to calculate the actual changes (NPSS determined) and the predicted changes (response curves) in the 3 health parameters for 12 sets of perturbation values. Table 2 shows that the actual changes (left panel) are predicted well by the least-squares approximation model (right panel) over the ±10 percent efficiency change range.

Table 2: Least-squares prediction accuracy comparison.

NPSS actual changes			Least-squares predicted changes		
Fan □	LPC □	HPC □	Fan □	LPC □	HPC □
0	0	0	0	0	0
2	0	0	2.05	-0.19	-0.05
0	1	0	-0.03	1.05	0.00
0	0	5	-0.05	0.52	4.94
-2	-1	0	-1.98	-1.00	-0.05
-5	-0.5	-2	-5.04	-0.30	-2.04
-5	-5	-5	-4.99	-5.16	-4.85
5	5	5	4.82	5.40	5.35
-7	-10	-2	-7.21	-9.47	-1.73
-10	0	0	-10.23	0.23	0.16
0	-10	0	-0.02	-10.22	0.29
0	0	-10	0.32	-0.53	-9.72

Although the least-squares approximation method works well for efficiency deviations as depicted above, it is less successful for more general cases involving mass flow capacities as a health parameters due to their non-linear behavior (Fig. 4). In such cases, we have experienced frequent solver problems -- convergence to an erroneous solution or non-convergence. Also, we have ignored another important issue; namely, sensor measurements with noise. In reality, sensor measurements always contain some noise from the sensors directly and from the subsequent signal processing. If typical noise levels were introduced, the pure least-squares approximation method would yield inaccurate predictions. Although a moving average could be used to alleviate this problem, an alternative approach is to filter out the noise and generate more accurate results.

[1] In NPSS parlance, each compressor efficiency parameter was wrapped as an interpreted element which contains the efficiency as an independent object and a "calculate() method" that computes the estimated values of dependent variables based on response curves as the solver changes the independent efficiency variable. N_1 and N_2 are wrapped as dependent objects with constraint conditions (i.e., equal measurement and estimated values). A third dependent object was defined for minimizing the squares of the differences between the estimated values and measurement values of the dependents.

FOD detection/diagnosis noise filtering methods

These problems (non-convergence and inability to handle random measurement errors) can be better handled using stochastic estimation to filter the noise. One widely used filter is called Kalman filtering and it relies on three assumptions: (1) the system is linear, (2) the random errors are comprised of white noise, and (3) the white noise is distributed normally. While the latter two are often satisfied, many real-world problems such as our diagnostic problem are non-linear. If the system is non-linear, the filter can be extended by linearization about the current state. This works adequately on some problems where the nonlinear effects are small, but can fail if the non-linearity is strong. First we briefly review the fundamentals of Kalman filtering, then apply it to our diagnostic problem.

Kalman filter - The Kalman filter addresses the general problem of estimating the state $x \in \Re^n$ of a linear discrete-time process; that is:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{12}$$

with a measurement $y \in \Re^m$:

$$y_k = Mx_k + v_k \tag{13}$$

where *k* is the discrete time step index, *A* is an *n* x *n* matrix that relates the previous state to the present state, *B* is an *n* x *l* matrix that relates the optional control input $u \in \Re^l$ to the state, *M* is an *m* x *n* matrix that relates the state to the measurement *y*, w_k is random system noise, and v_k is random measurement noise. Both w_k and v_k are assumed to be white and normal with mean value 0 and with covariance matrices *Q* and *R,* respectively; that is, their probability distributions *p* are:

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

The Kalman filter estimates the process state *x* by using a recursive predictor-corrector algorithm. The predictor has two equations for calculating the priori estimate state $\hat{x}_k^-$ and the priori estimate error covariance P_k^-:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{14}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{15}$$

The corrector has three equations to calculate the gain matrix K_k, the posteriori estimate state $\hat{x}_k$ and the posteriori estimate error covariance P_k:

$$K_k = P_k^- M^T (MP_k^- M^T + R)^{-1} \tag{16}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - M\hat{x}_k^-) \tag{17}$$

$$P_k = (I - K_k M)P_k^- (I - K_k M)^T + K_k R K_k^T \tag{18}$$

K_k obtained above minimizes P_k and thus $\hat{x}$ is called the minimum variance estimate of *x*.

Extended Kalman Filter (EKF) - If the governing process and measurement equations are not linear, then a Kalman filter that linearizes about the current state is called an extended Kalman filter or EKF. The process and measurement are described by:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{19}$$

$$y_k = g(x_k) + v_k \tag{20}$$

Let A_k be the partial derivative matrix of $f(x_{k-1}, u_{k-1})$ at the previous posteriori estimate $\hat{x}_{k-1}$. Linearizing around $\hat{x}_{k-1}$ gives

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}) \tag{21}$$

$$P_k^- = A_k P_{k-1} A_k^T + Q \tag{22}$$

Let M_k be the partial derivative matrix of $g(x_k)$ at the present priori estimate $\hat{x}_k^-$. Linearizing around $\hat{x}_k^-$ gives

$$K_k = P_k^- M_k^T (M_k P_k^- M_k^T + R)^{-1} \tag{23}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (y_k - g(\hat{x}_k^-)) \tag{24}$$

$$P_k = (I - K_k M_k) P_k^- (I - K_k M_k)^T + K_k R K_k^T \tag{25}$$

Application of EKF to fault detection - We can apply the EKF method to our previously defined example FOD diagnostic problem rather simply. The state variables are the percentage changes in the six health parameters $x_1, x_2, \cdots, x_6$ and the measurement variables are the changes in the five sensor parameters $y_1, y_2, \cdots, y_5$ as defined above. We create an EKF about the operating point as follows:

The process and measurement estimation equations are:

$$x_k = x_{k-1} + w_{k-1}, \quad p(w) \sim N(0, Q) \tag{26}$$

$$y_k = g(x_k) + v_k, \quad p(v) \sim N(0, R) \tag{27}$$

where $g(x) = (g_k)$ as described above in equation 11.

The predictor equations are:

$$\hat{x}_k^- = \hat{x}_{k-1} \tag{28}$$

$$P_k^- = P_{k-1} + Q \tag{29}$$

The corrector equations are the same as equations 23, 24 and 25. The terms of M_k are obtained by partially differentiating *g(x)* with respect to *x* and updating it at each time step.

We assumed that all state variables have a standard deviation $\sigma = 0.05$ so the diagonal terms of covariance matrix *Q* are $\sigma^2 = 0.0025$. The off-diagonal terms were arbitrarily set to 0.0005. To set the terms of *R*, we assumed that all sensor measurements have a σ of 0.005 times their base values (that is, the signal/noise ratio, SNR = 200) and that there are no correlations among the measurements, so all off-diagonal terms are zero. The filter needs an initial guess to start the process and x_0 was set to zero, while P_0 was set to *Q*.

The measurements were synthesized using a random number generator, standard deviations of 0.005 times the sensor base values, and normal probability distributions (Fig 5). When synthesizing the measurements, we used a different seed for each measurement variable so that the measurements among the variables are not correlated.[2]

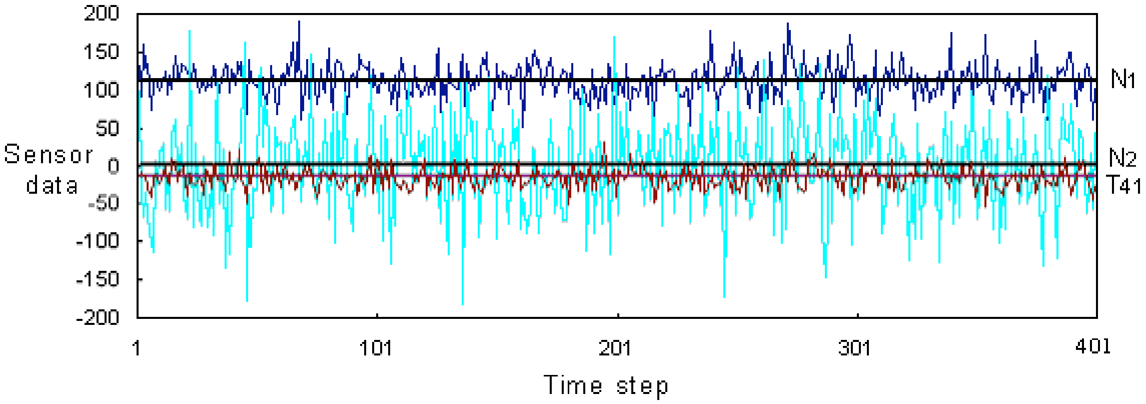


Figure 5 - Example synthesized noisy sensor data resulting from a -3 percent change in fan flow capacity. The black horizontal lines represent the true measurement changes while the curves are the synthesized noisy data with SNR = 200.

To emulate single component fault events, we made -3 percent changes in the 6 health parameters (state parameters in EKF terminology) and computed the corresponding changes in the 5 sensor measurement parameters using two different methods: (a) modeling the cycle behavior with the NPSS code, and (b) using the previously generated response curves from equation 3. Method b requires considerably less execution time than method a, but is not as accurate. In both methods, we added white noise as illustrated in Figure 5. Then, using the calculated noisy sensor changes, we applied the EKF method to predict the health parameter changes.

Ten sets of the health parameter changes were analyzed as shown in Table 3 for both methods of generating the measurement changes. The results indicate that both methods yield comparable results; that is, the results tabulated under the "Predicted changes using LSA y_{pre} -values " heading (method b) are only slightly less accurate than using NPSS to generate the measurement changes.

Figure 6 displays typical EKF solution convergence behavior for the first of these sets. Starting the iteration with zero for the initial condition ($x_0 = 0$), the EKF method yields acceptably accurate solutions in all cases, both single fault cases and multiple fault cases, except the fourth case where only the LPC efficiency was changed. In this case the Kalman filter incorrectly predicts a combination of positive LPC flow capacity change and negative HPC flow capacity change. However, if x_0 was set to the true state [0,0,0,-3,0,0] instead of 0, the filter yields the correct prediction (case 5 in Table 3). This indicates that multiple health parameter states may generate the same sensor measurements, and that the predicted set of health parameters depends on the assumed initial state. This is an inherent byproduct of non-linear systems. The same anomaly was also observed using noiseless measurements (not displayed). Because the EKF method cannot differentiate multiple solutions from each other based solely on the measurements, other means must be used. In the specific case encountered here, the pattern of positive predicted LPC flow capacity change and negative HPC flow capacity change may be used to discard this solution. The problem of multiple solutions may be mitigated somewhat by choosing measurement variables that are strongly dependent on the health parameter changes.

[2] If the same seed is used, a singular matrix will be generated which renders the filter inoperable.

It should be noted that the least-squares approximation solution method is also subject to this issue of finding inaccurate solutions. In addition, both the EKF method and the least-squares method tend to smear the allocation of predicted faults among all of the health parameters instead of concentrating it totally on one (or several) true faults.

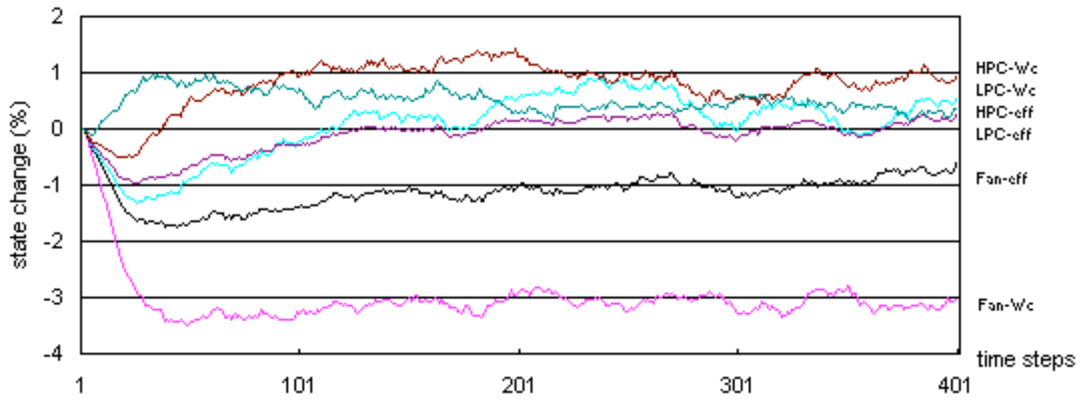


Figure 6 – Example EKF iteration history of case 1 of Table 3 with SNR = 200. The true health parameter fault is a -3% change in fan flow capacity at step 1, all other changes are zero.

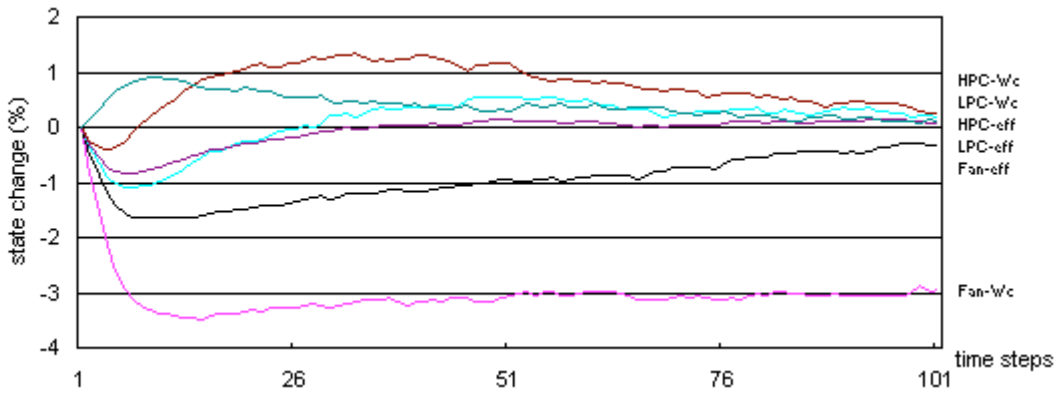


Figure 7 - Example EKF iteration history of case 1 of Table 3 with SNR = 1000. The true health parameter fault is a -3% change in fan flow capacity at step 1, all other changes are zero.

Table 3: EKF solution results for ten fault cases with SNR = 200.

The six column headings are the health parameters: fan W_c, fan η, LPC W_c, LPC η, HPC W_c, HPC η.

Case	True changes, %	Predicted changes using NPSS y-values	Predicted changes using LSA y_{pre} -values
1	-3 0 0 0 0 0	-2.68 -0.26 0.46 0.26 0.58 0.12	-3.09 -0.73 0.49 0.19 0.86 0.29
2	0 -3 0 0 0 0	0.25 -2.03 -1.09 -1.04 -0.43 -0.26	0.34 -2.08 -0.99 -1.02 -0.52 -0.18
3	0 0 -3 0 0 0	-0.07 0.06 -3.47 -1.27 0.73 0.22	-0.04 0.06 -3.39 -1.25 0.63 0.25
4	0 0 0 -3 0 0	-0.12 -0.22 1.72 0.33 -1.03 -0.53	-0.10 -0.22 1.78 0.35 -1.05 -0.54
5	0 0 0 -3 0 0	-0.09 -0.05 0.06 -2.94 0.16 0.02	-0.07 -0.05 0.11 -2.93 0.14 0.01
6	0 0 0 0 -3 0	-0.12 -0.26 -0.08 -0.45 -2.24 0.09	-0.07 -0.06 0.10 0.06 -2.89 0.00
7	0 0 0 0 0 -3	-0.22 -0.20 -0.24 -0.51 0.25 -3.01	-0.09 -0.05 0.08 0.06 0.14 -2.97
8	-3 -3 0 0 0 0	-2.62 -2.14 -0.56 -0.86 -0.04 -0.04	-2.99 -2.46 -0.62 -0.96 0.03 -0.01
9	-3 0 -3 0 0 0	-2.66 -0.19 -3.06 -1.13 1.21 0.44	-3.06 -0.63 -3.05 -1.19 1.40 0.56
10	0 0 0 0 -3 -3	-0.28 -0.51 -0.41 -1.04 -2.28 -2.83	-0.13 -0.49 -0.22 -0.99 -2.05 -2.68

Notes: 1. The predicted values are the average values of the last 10 of 400 iterations.
2. Iterations were initialized with $x_0 = 0$, except case 5 where $x_0 = [0\ 0\ 0\ 0\ -3\ 0\ 0]$.

The measurement covariance matrix R affects the convergence rate of the EKF method. The smaller the R, the larger the gain matrix K_k, thus more weight is given to the measurements and the filter takes less time steps to converge. Figure 7 shows that the increase in SNR not only speeds the convergence but also decreases the fluctuation significantly. To effectively detect short duration FOD events that do not cause permanent health parameter changes, there should be enough time steps for the filter to show positive convergence. This can be achieved by either increasing the frequency of measurements or raising the SNR. Figure 6 shows a positive detection at about time step 50. If the event lasts 1 second, then the recording rate must be at least 50 Hz. If the SNR is raised from 200 to 1000 (Fig. 7), positive detection takes place at time step 10 and the recording rate can be lowered to 10 Hz.

The selection of the covariance matrix Q must be fine tuned. Setting Q too small has the effect of putting too much trust on the estimate and the filter takes more time steps to converge. Setting Q too high, however, causes the filter to "drift" -- the predicted health parameters may contain values outside those that were used to generate the response curves so the results cannot be trusted.

It is desirable that Kalman filter should be able to operate continuously without divergence after an FOD event. Figure 8 shows the continuous EKF operation with 3 fault events of reduced fan, LPC, and HPC weight flow taking place at different times. The filter detects the events quickly and correctly, and returns to the baseline values rapidly after the faults are removed.

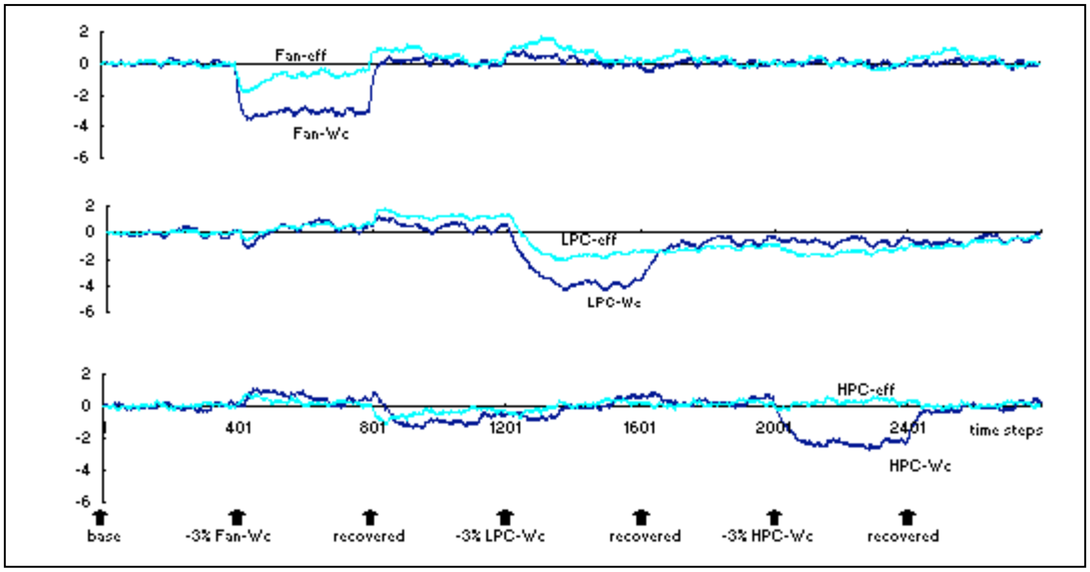


Figure 8 - Continuous operation of EKF with SNR of 200 and three FOD events. The events occurred at time steps 400, 1200 and 2000, each with a 400 time step duration, and no event caused sustained damage.

In summary, we have shown that the response curves obtained by perturbing analytic engine models and a least-squares approximation can be used to establish an EKF that successfully detects FOD fault events. Setting some of the filter configuration parameters is important to achieving good results; specifically, the initial state x_0 and measurement signal/noise ratio SNR.

State-space open-loop system

A MATLAB/SIMULINK state-space open-loop system can be used to represent the behavior of a linear system:

$$\begin{aligned} \dot{x} &= \mathrm{A}x + \mathrm{B}u \\ y &= \mathrm{C}x + \mathrm{D}u \end{aligned} \tag{30}$$

where the vectors *x*, *u*, and *y* represent the state, control, and output variables, respectively. The A, B, C and D matrices can be conveniently generated with the help of a NPSS utility called the Linear Model Generator (LMG) that automatically perturbs a baseline case multiple times.

First, we verify that such an open loop system behaves similarly in both a NPSS transient simulation and a MATLAB/SIMULINK simulation. The engine model used was a high-bypass turbofan wherein the single control variable was fuel flow deviation dW_{fuel} and the health parameter was fan efficiency deviation *dFanEff*. Both state and the output variables were the low-spool and high-spool shaft speeds, dN_1 and dN_2. Here, *d* represents the deviation from nominal value at the steady-state condition.

Figure 9 shows the system behavior in SIMULINK and Figure 10 shows the transient simulation of the same system in NPSS. Step inputs of dW_{fuel} and *dFanEff,* separately and together, were introduced at time 1 second. The shapes and magnitudes of the responses are similar in both simulations, although there is about a 10 percent steady-state difference caused by the non-linearity of the model.

Because the engine health parameters can only be observed instead of serving as controls, we augment the state-space representation to include health parameters as state variables so that they can be estimated. Also an observer was added so that the states can be estimated from measured sensor outputs.

Figure 11 shows such a complete state observer system. The augmented state vector contains dN_1, dN_2, and compression system efficiency deviations (*dFanEff, dLpcEff* and *dHpcEff*). The control is dW_{fuel}. The measurements are dN_1, dN_2, burner inlet total temperature (*dITt*), burner exit total pressure (*dBurnOPt*), and burner exit total temperature (*dBurnOTt*). Note that the Kalman gain in the state observer is calculated using the continuous-time algebraic Riccati equation. Figure 10 shows the responses. A step change of dW_{fuel} was introduced in the time interval 5-15 seconds (panel A) and FOD-caused efficiency deviations of all three compression system components occur at time 0 (panel B). The noisy measurements are shown in panel D. Panel E shows that the estimated efficiency deviations agree with the actual values shown in panel B. The estimated outputs (panel F) are also close to the actual outputs (panel C) and the results obtained from NPSS transient simulation (panel G).

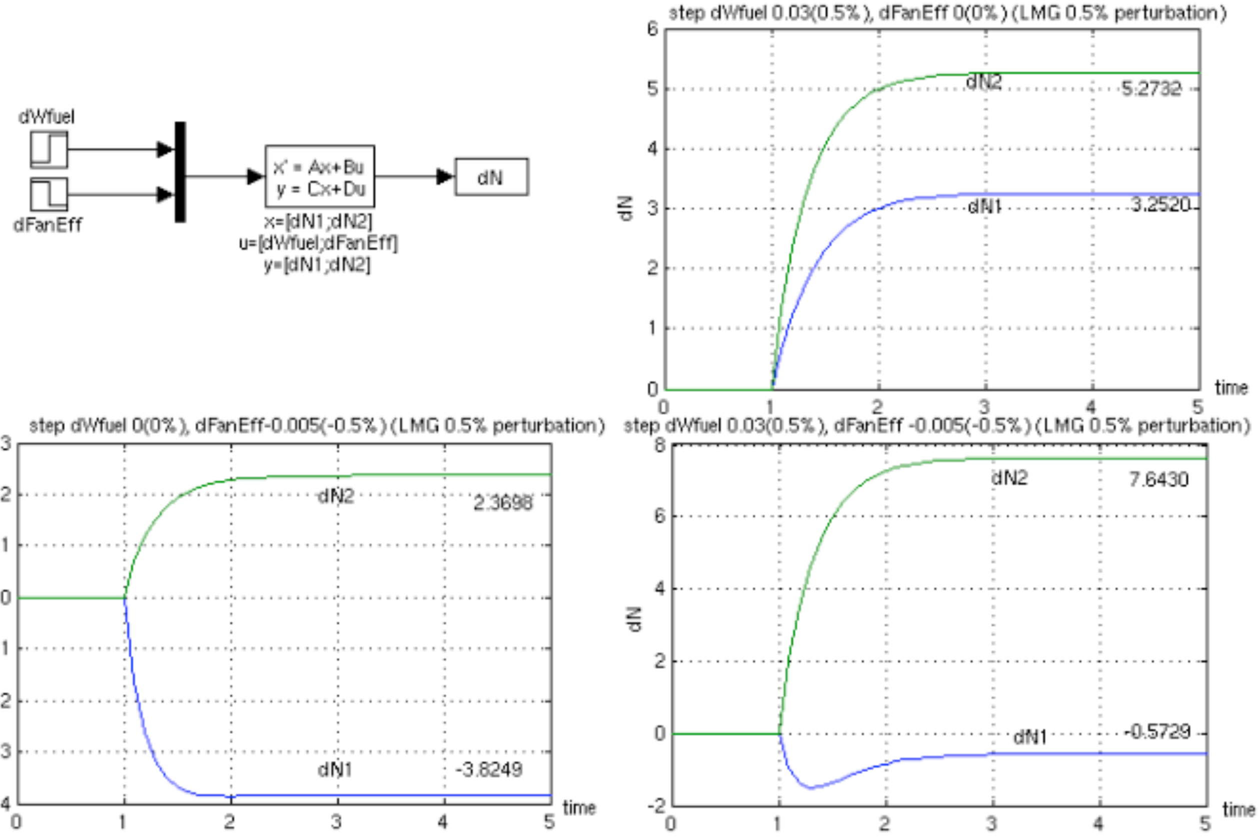


Figure 9 – A simple state-space MATLAB/SIMULINK simulation of a FOD event at 1 second.

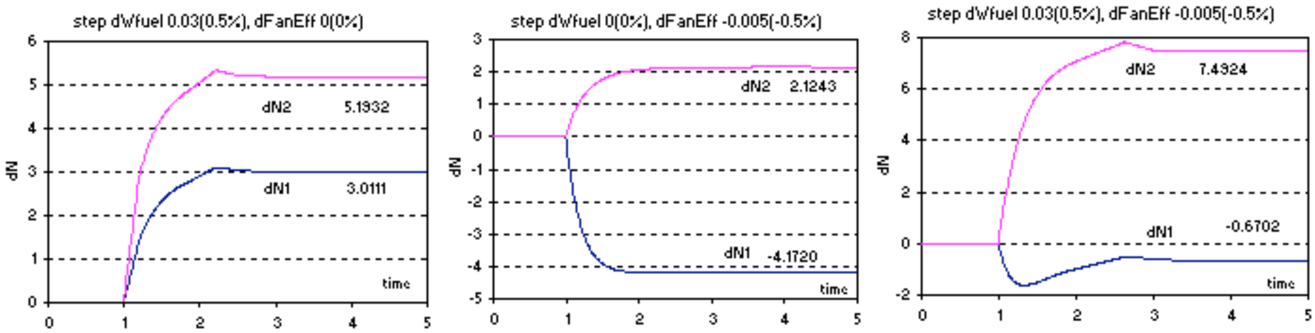


Figure 10 – A NPSS transient simulation of the same FOD event as used in the SIMULINK simulation.

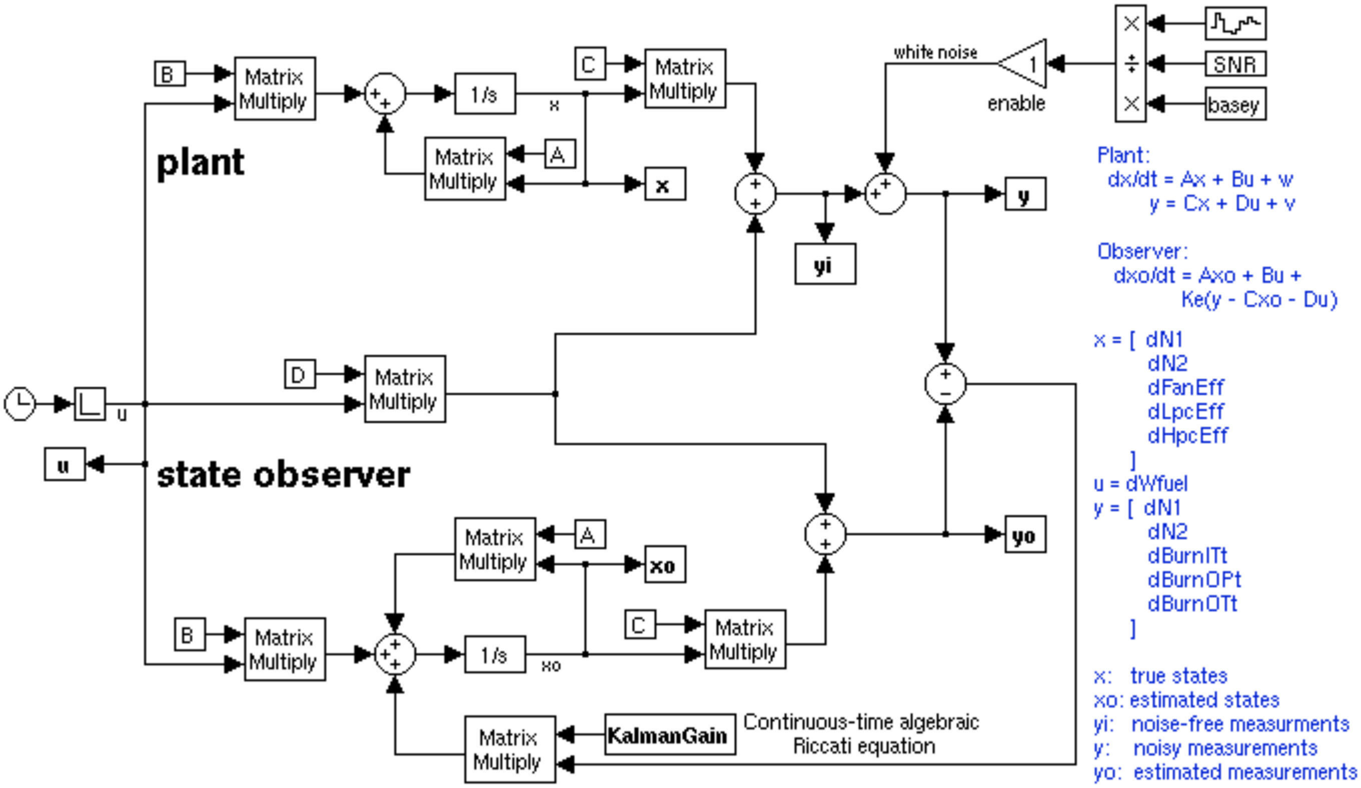


Figure 11 - A state observer for FOD detection.

The state-space open loop state observer system as described in Figure 11 is capable of detecting FOD events. Note that the control variable *u* and noisy measurements *y* are input to the system, and the estimated state *xo* and estimated measurements *yo* are output from the system. An NPSS simulation can provide the inputs, but we need a way to link NPSS to the MATLAB/SIMULINK model to do so and to retrieve the outputs from the MATLAB/SIMULINK simulation.

NPSS does provide a Generic Controller dynamically loadable module (DLM) which enables NPSS to drive MATLAB. A definition file is required to specify the SIMULINK model file name, input variables, and output variables. At run time, NPSS parses the definition file and communicates with SIMULINK. The model shown in Figure 11 needs to be modified so that *u* and *y* are inputs. Such an integrated system was implemented and generated essentially the same results as shown in Figure 12.

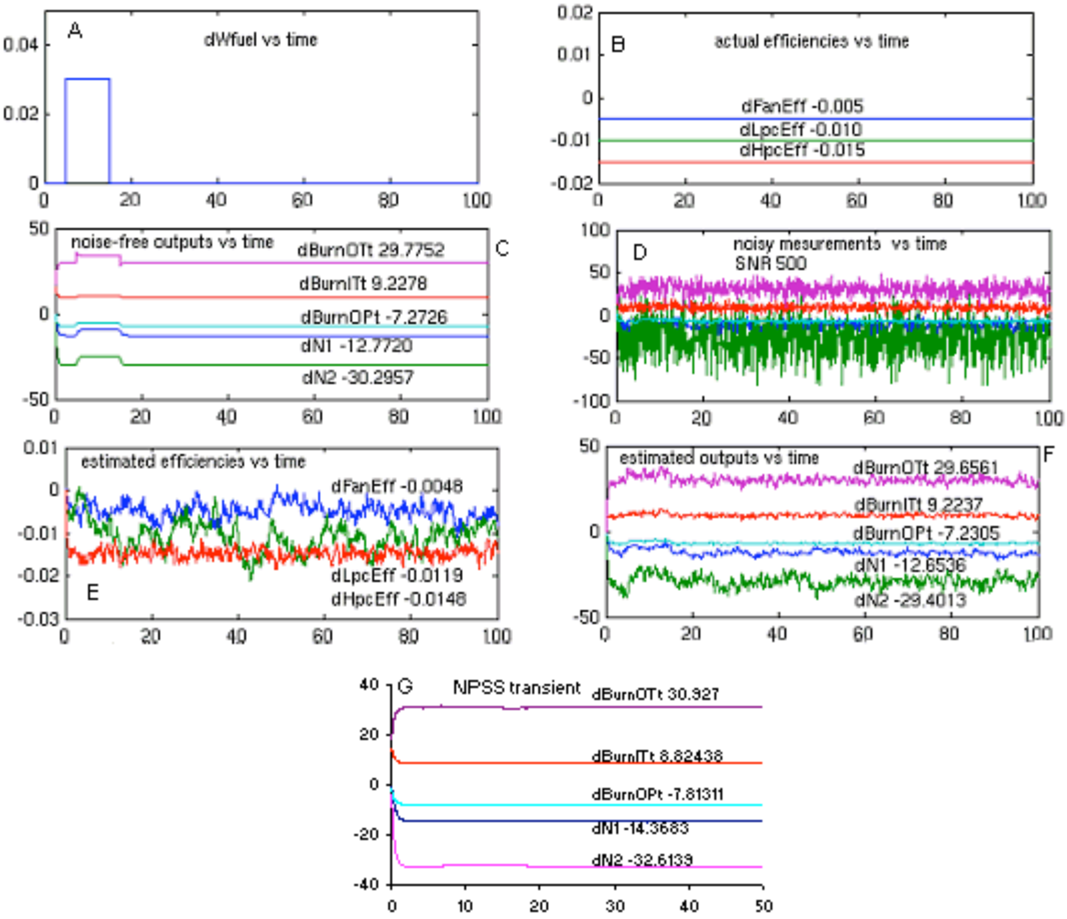


Figure 12 - The responses of the state observer for FOD detection.

FOD event detection via vibration signals

Up to this point we have only discussed the use of traditional gas path analyses to detect and diagnose FOD events. These analyses infer damage to specific components based on indirect measurements. That is, they estimate damage based on the thermodynamic consequences of geometry changes sustained during the event. But what if the event causes damage without changing the geometry? For instance, a relatively small impact may only initiate a crack rather than break or deform blades. The crack may sooner or later precipitate a serious failure, but a gas path analysis will not detect the event when it occurs – only if and when the crack enlarges and causes either deformation or failure.

To cope with this possibility as well as provide confirming FOD detection evidence for large impact events, it was hypothesized that structural dynamics offers the possibility of detecting both minor and major FOD events. The basic concept is to continuously monitor bearing accelerometer signals (or other dynamic responses) and search for FOD-caused anomalies. FOD events that cause permanent damage are likely to cause an increase in rotor imbalance that can be readily detected as a step change in an acceleration moving average (similar to Fig. 2). Just as in the gas path analysis, a moving average can be used to handle noisy acceleration signals. On the other hand, the FOD event may not cause a permanent increase in imbalance and in such cases only a very short transient signal (<< 1 second) aberration occurs. In this case we must invoke a more sophisticated signal processing methodology to isolate such events, particularly when they are masked by system and sensor noise.

The original approach used to pursue this concept was to create a reduced-order structural response model of a typical turbofan engine that accurately captures all first-order effects as illustrated in Figure 13.

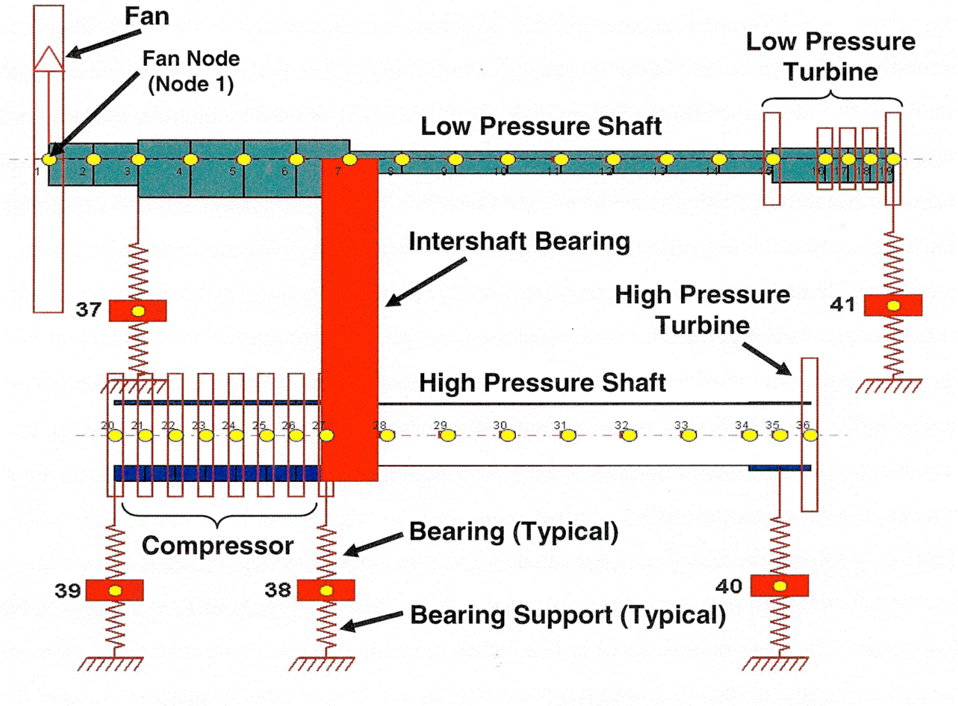
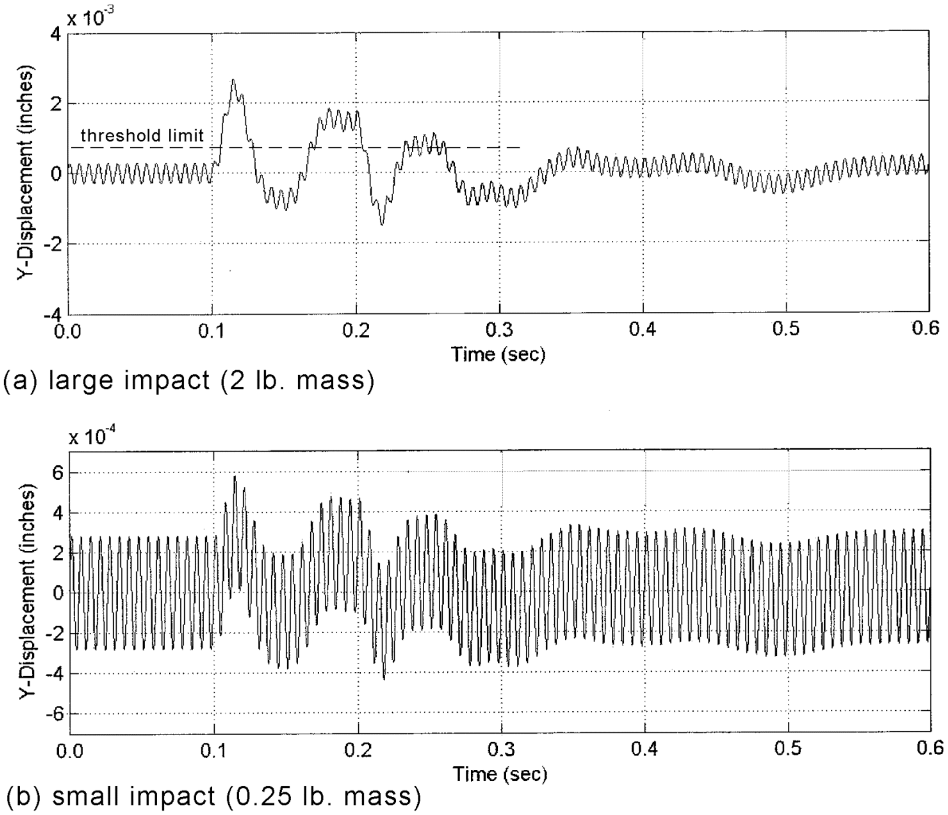


Figure 13 - FEM representation of a turbofan rotor/bearing system (ref. zzz).

This model was used in conjunction with MATLAB/SIMULINK to develop a continuous-time state-space vibration model which can be used to generate rotor bearing vibration signals in response to given FOD events (i.e., mass of object, fan impact radius, impact velocity and duration, permanent increase in imbalance). Reference 13 contains a detailed description of this methodology.

In an ideal situation (neither process nor sensor noise), strategically located bearing vibration sensors would transmit a constant amplitude oscillating signal due to residual rotor imbalance as displayed in Figure 14a in the time span 0 to 0.1 seconds. If a relatively large FOD event occurs, the signal response would change significantly as illustrated and simple threshold-exceedance detection logic would be adequate. If the FOD impact is small as in Figure 14b, discerning the event is more challenging but still tenable.

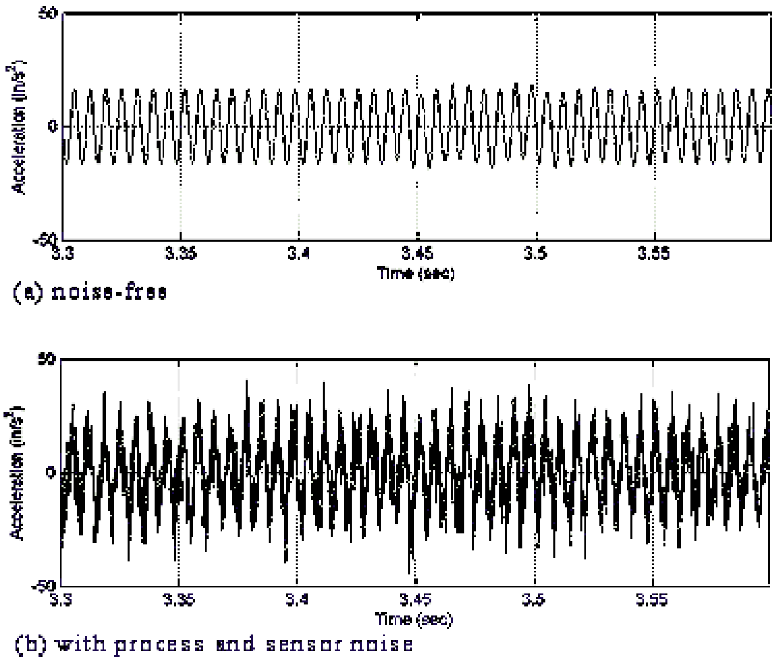
However, the situation becomes much more challenging when noise is included and the FOD impact is small. In these cases, the short duration event will cause no permanent deformation and the event signature will be completely masked by noise as illustrated in Figure 15. Part a of this figure displays the weak HPT bearing signal resulting from a ½ pound - 300 feet/sec foreign object impact at the 20 inch fan radius location assuming a fan eccentricity of 0.001 inches. The event is modeled as a 0.04 second square-wave pulse that occurs at 3.44 seconds and is barely noticeable. When process and sensor noise (assuming a signal/noise ratio of 3.5) is added to this ideal signal (Fig. 15b), the FOD event is completely masked.



(a) large impact (2 lb. mass)

(b) small impact (0.25 lb. mass)

Figure 14 – Noise-free bearing displacement response signals from FOD impacts (ref. 13)



(a) noise-free

(b) with process and sensor noise

Figure 15 – Bearing accelerometer signals in response to a small FOD event at 3.44 sec. (ref. 13)

To unmask such events we must resort to advanced signal processing techniques. One powerful approach is to use a discrete time wavelet transform to perform this function. Our analytical investigations indicated that such events can be successfully isolated using a 3-step Daubechies 8 wavelet smoothing filter followed by a third order differentiation of the wavelet transform as depicted in Figure 16. The acceleration signal is passed through an approximation filter three times and subsequently through a detail filter in order to focus on the subband of interest in the original signal.

The net result of these signal processing manipulations is the ability to clearly detect an FOD-induced vibration event. Figure 17a below illustrates the same raw accelerometer signal displayed previously in Figure 15b but at a coarser time scale. The impact pattern is lost in the noise. Figure 17b displays the processed signal using the wavelet analysis. The impulse feature is quite clear with spikes both at the start and end of the event.

Encouraged by the results of the continuous-time state space MATLAB/SIMULINK simulation, we proceeded to implement the same vibration analysis methodology using NPSS. This required extending NPSS to perform such structural analyses. However, NPSS does not provide an approach to do continuous-time state space model simulation, so we first created several discrete-time state space models at different sampling frequencies to determine which discrete-time frequency model adequately represents the continuous-time model.

Using the c2d() MATLAB function, the discrete-time A, B, C, D matrices were obtained at sampling frequencies of 100, 1000, and 10,000 Hz. Then an NPSS interpreted element entitled Vibration was created. Using input such as sampling frequency, time, engine speed, engine speed deviation, FOD residual imbalance, impact mass, and impact duration, this element generates bearing displacement and acceleration signals at the locations previously defined in Figure 13. Figure 18 shows that the NPSS Vibration element with 1000 Hz and 10,000 Hz sampling frequencies generated results similar to the continuous SIMULINK model, while sampling at 100 Hz gave poor results. Given that the engine speed is around 7000 rpm, 1000 Hz sampling yields approximately 8 points per cycle, which is greater than the required Nyquist rate[3]. Therefore, sampling at 1000 Hz or greater is valid for our discrete-time implementation.

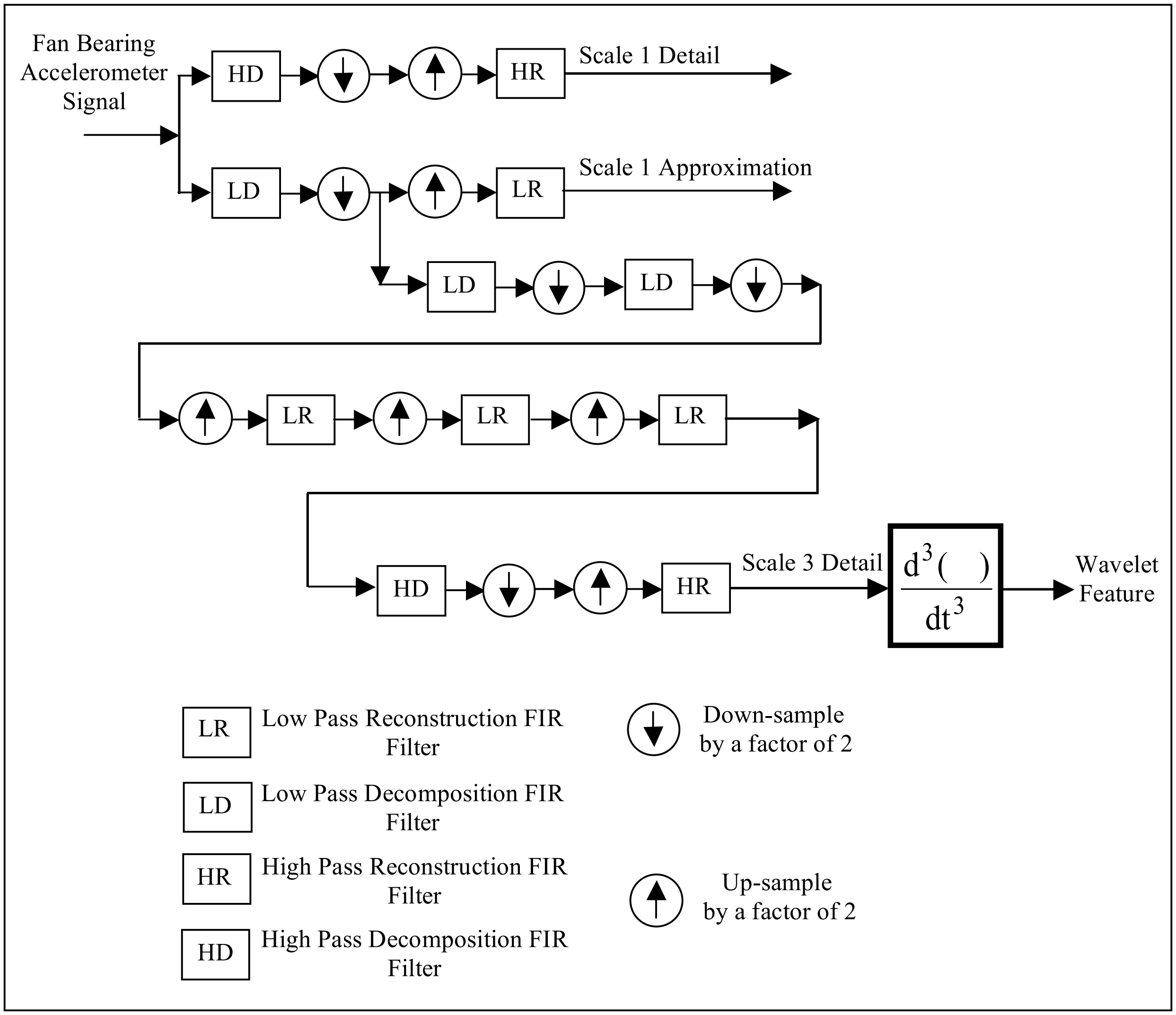

Fan Bearing Accelerometer Signal
HD
HR
Scale 1 Detail
LD
LR
Scale 1 Approximation
LD
LD
LR
LR
LR
HD
HR
Scale 3 Detail
$\frac{d^3(\ )}{dt^3}$
Wavelet Feature
LR Low Pass Reconstruction FIR Filter
LD Low Pass Decomposition FIR Filter
HR High Pass Reconstruction FIR Filter
HD High Pass Decomposition FIR Filter
Down-sample by a factor of 2
Up-sample by a factor of 2

Figure 16. Wavelet transform-based vibration feature extraction implemented using an analysis and synthesis finite impulse response (FIR) filter bank (from ref. 13).

The platform used for these tests was a Windows 2000 Advanced Server with four x86 Intel 278 processors and 3.9 GB of physical memory. To simulate 1 second of real time data, it took approximately 2 seconds of execution time for the 100 Hz model, 18 seconds for the 1000 Hz model, and 180 seconds for the 10,000 Hz model. None of these models is fast enough for real time simulations. Because NPSS interpreted elements execute several times slower than dynamically loadable module (DLM) elements, we subsequently created a corresponding DLM version of the Vibration element. The corresponding execution times were 0.2 seconds for the 100 Hz model, 1 second for the 1000 Hz model, and 10 seconds for the 10,000 Hz model. Thus, at 1000 Hz the DLM-version of the Vibration element can achieve real-time performance.

[3] The Nyquist rate is lowest sampling rate that permits accurate reconstruction of the sampled analog signal.

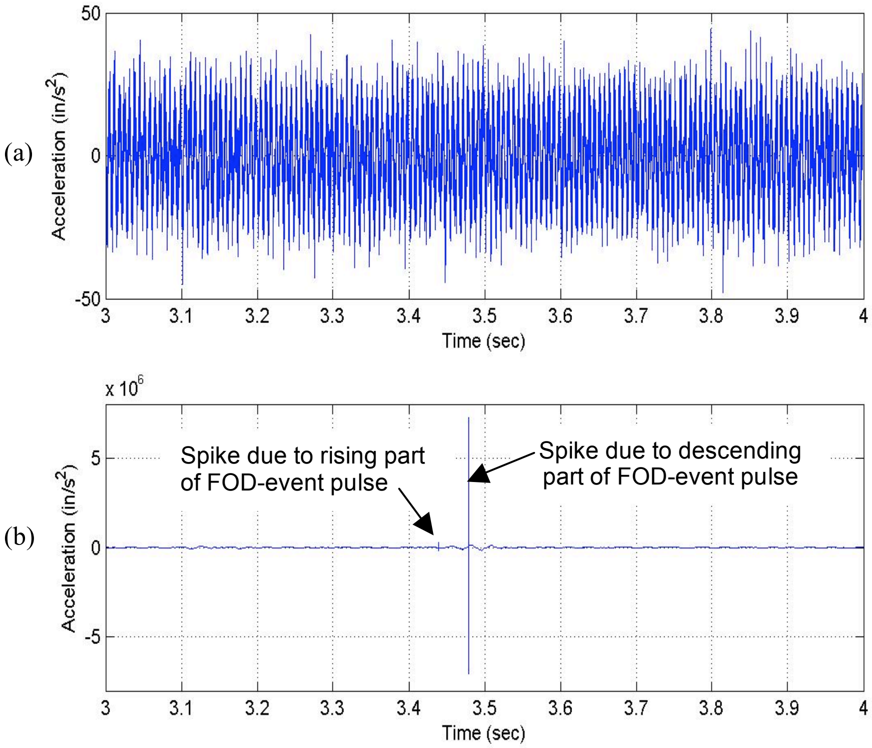


Figure 17. Example structural vibration signal containing noise and a small FOD event at 3.42 seconds lasting 0.02 seconds: (a) raw accelerometer signal, (b) wavelet-processed signal that exposes the hidden FOD event.

Wavelet analysis was performed in NPSS using various levels of lowpass decomposition[4] followed lowpass reconstruction as illustrated in Figure 19 for 2 levels. If a FOD event is embedded in the noisy vibration signals, it should show up more distinctly in the smoothed signal. Because the signals generated by the Vibration element do not contain noise, a Gaussian element was developed that generates random Gaussian numbers with specified mean and standard deviation using the Box-Mulder method[5]. The wavelet analysis is coded in another NPSS element (Daub) that implements the Daubechies 8 filter and contains methods for lowpass decomposition, lowpass reconstruction, highpass decomposition, highpass reconstruction, down-sample, up-sample, differentiation, and recombination of two signals. The filter coefficients were obtained from the prior MATLAB implementation.

First we verified that the filter works properly. We synthesized sine wave signals containing Gaussian noises (Fig. 20, left). After 1 level of filtering, we obtained the “approximation signal” (Fig. 20, middle top) and the “detailed signal” (Fig. 20, middle bottom). The approximation signal and detailed signal were then recombined to accurately form the original signal (Fig. 20, right). We then applied the Daub element to the analysis of vibration data obtained using the Vibration and Gaussian elements.

[4] Decomposition separates the signal into 2 fundamental constituents: an average signal called the “approximation” and the remaining portion of the signal called “detail.”

[5] Random numbers are generated using the linear conguential method because NPSS does not contain a random number generator.

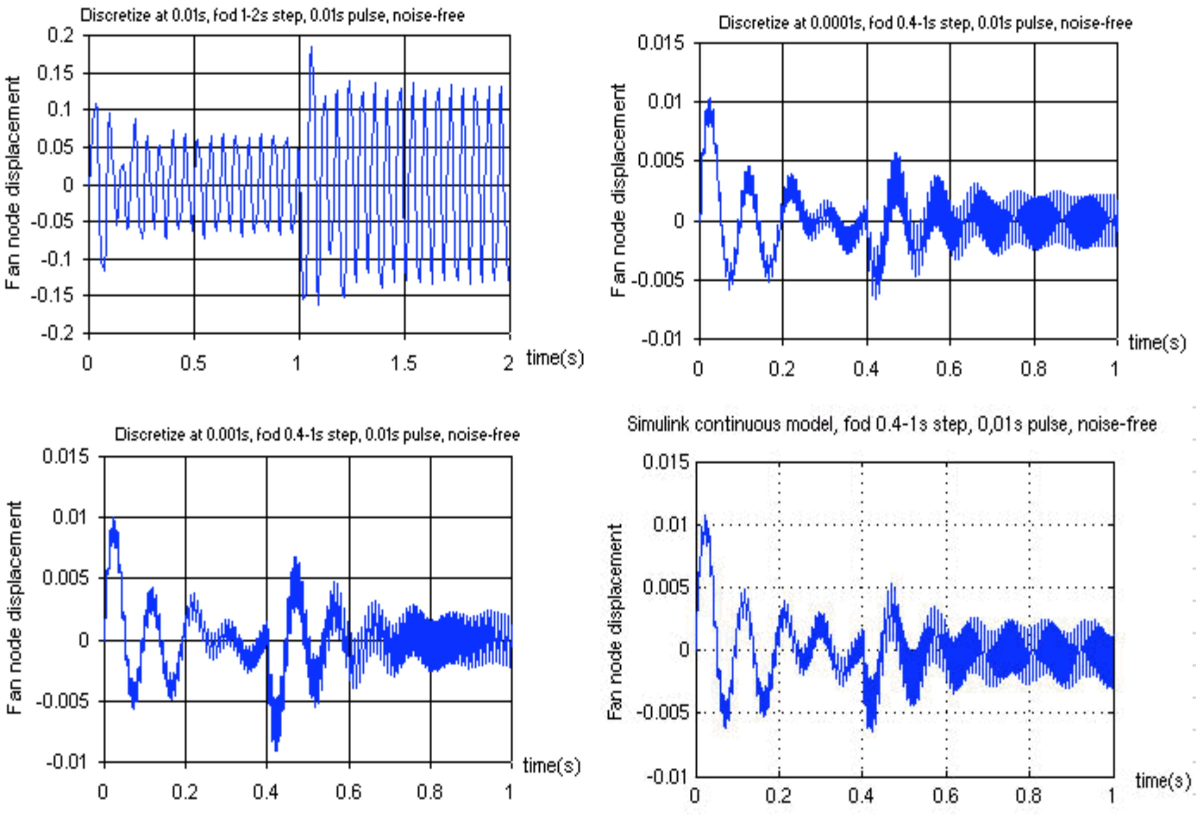


Figure 18 – Discretized NPSS vibration results for100, 1000, 10,000 Hz sampling frequencies and baseline SIMULINK results .

Here we used a 1000 Hz sampling rate and an FOD event was introduced at 2 seconds with an impact duration of 0.01 seconds and an FOD-imbalance duration of 0.2 seconds. Figure 21 shows the results with the fan node displacement acceleration signal. After 2 levels of low-pass filtering, the signals were smoothed to reveal the incidence of the FOD event (Fig. 21B).

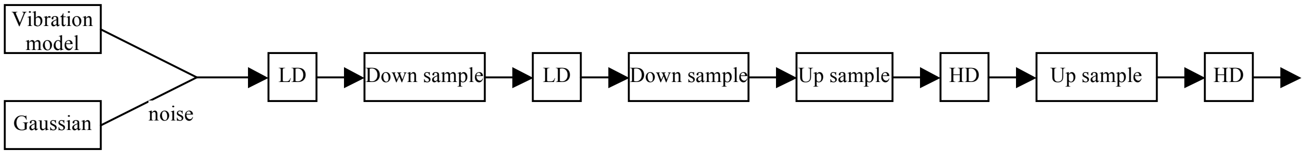


Figure 19- Wavelet analysis of vibration signals using 2 levels of lowpass filtering.

FOD events can also be detected in vibration signals by observing the signal "magnitude" and rate of change of magnitude. Magnitude is calculated as the root mean square (RMS) of a group of samples. Rate of change of magnitude is just the difference between two neighboring magnitudes. Figure 21 shows an example magnitude analysis for a group size of 100 samples. Parts C and D show the results without filtering and parts E and F show the results after 2 levels of lowpass wavelet filtering. In all cases the FOD event was identified, but the results without filtering appear superior.

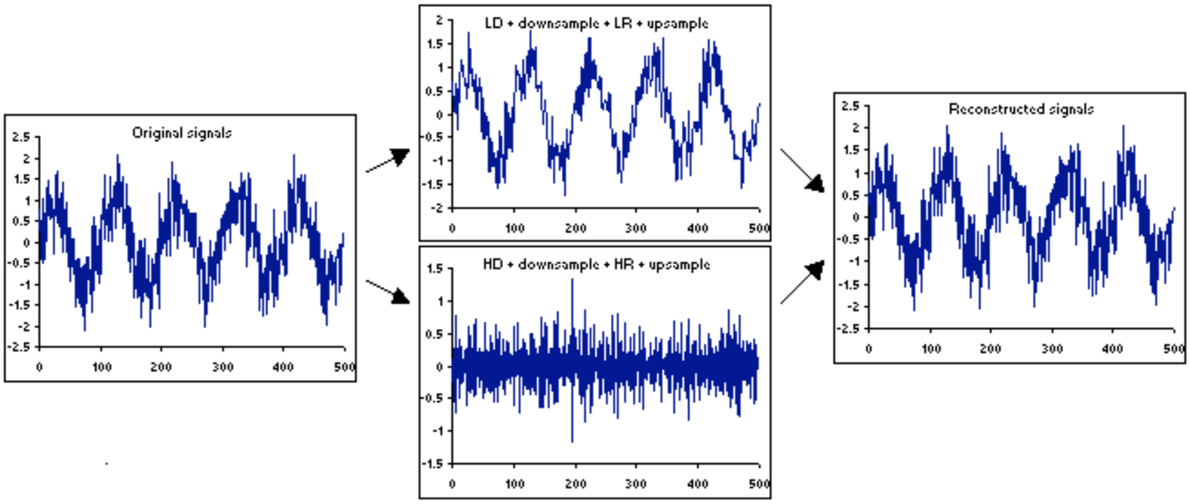


Figure 20 - Verification of wavelet Daub NPSS element.

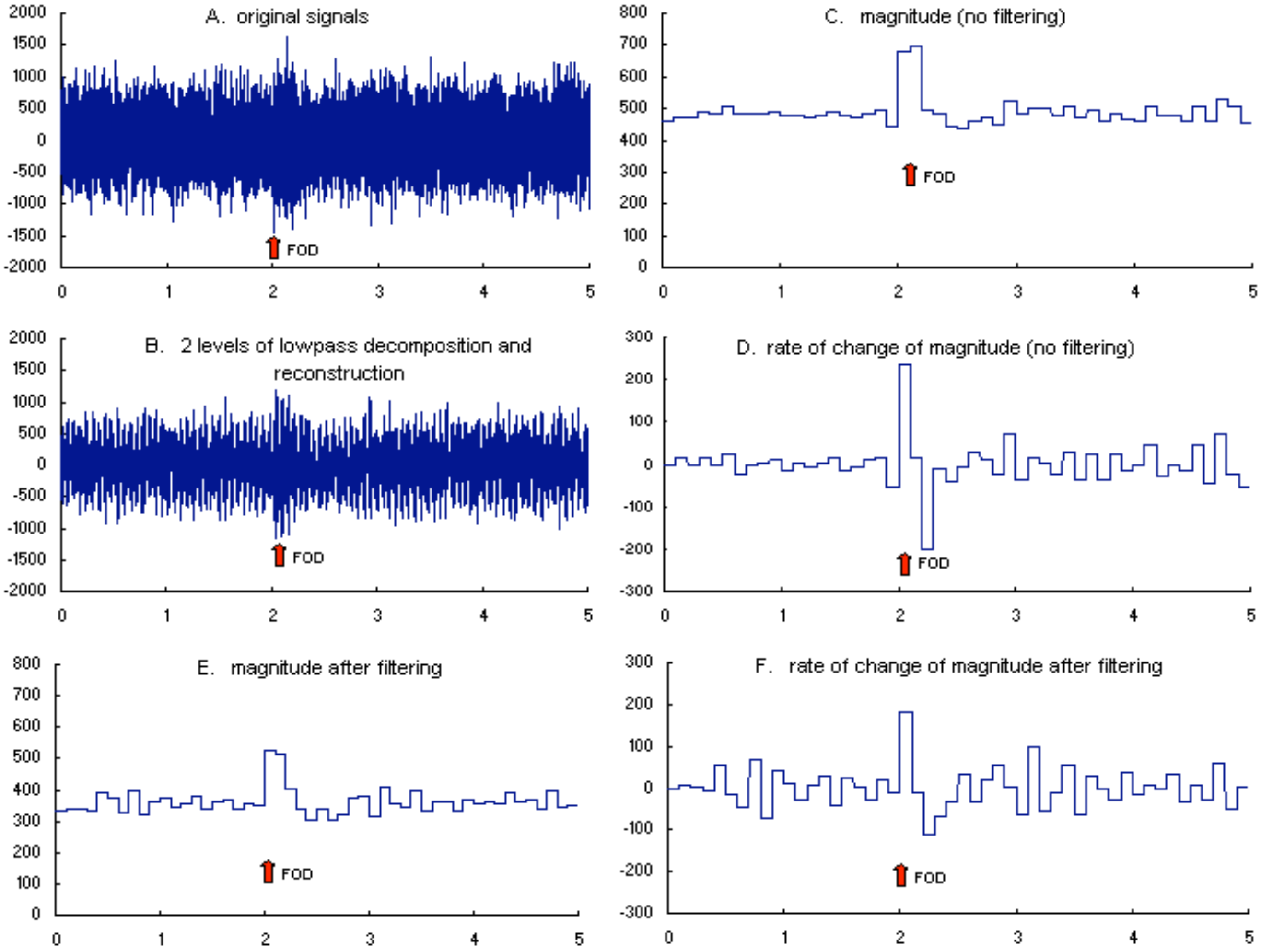


Figure 21 - Wavelet and magnitude analysis of fan node displacement acceleration signals.

Fusing gas path results with vibration results

Since neither the gas path analysis approach nor the structural dynamics approach guarantee accurate solutions (they both involve significant uncertainties), a data fusion framework using a fuzzy logic inference system was formulated that merges the results from both analytic approaches into a single detection/diagnostic result possessing greater confidence than either approach provides separately. Raw data from different phenomenological sensors (gas path and vibration) observing the same event can be reduced to a set of compact results using “membership functions” that categorize the results into weighted beliefs. That is, the gas path results yield evidence that either an FOD event occurred, it did not occur, or it is uncertain (and similarly for the vibration analysis). The strength of the belief for each category is quantified by calculating a “probability mass” value as illustrated in the left hand side of Figure 22. The gas path and vibration “expert” results can then be fused together into a composite picture that weighs the evidence of both experts. A full discussion of this effort is reported in reference 12.

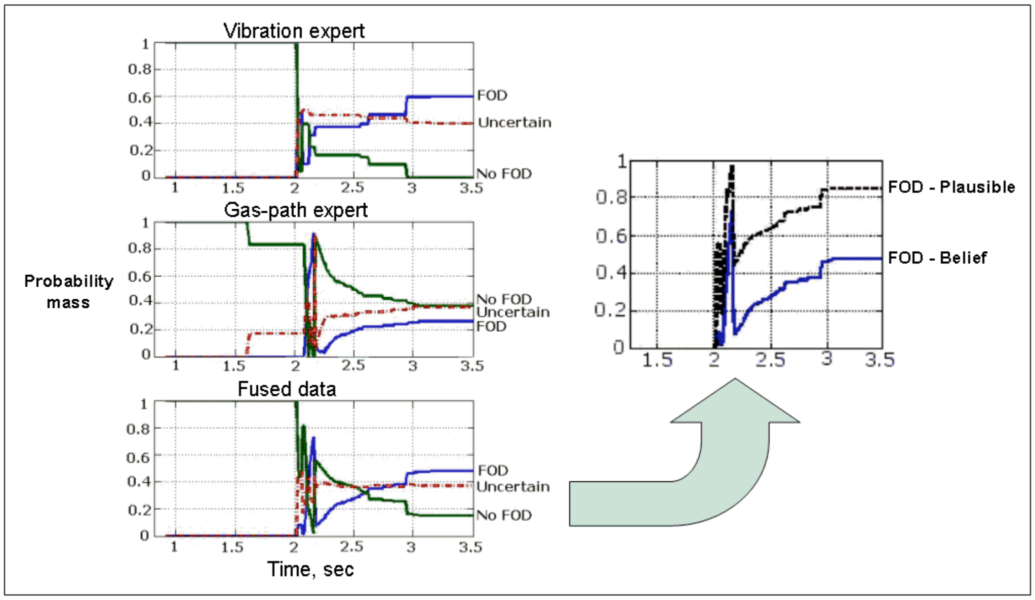


Figure 22 – Example fuzzy logic inference data fusion with marginal evidence of an FOD event.

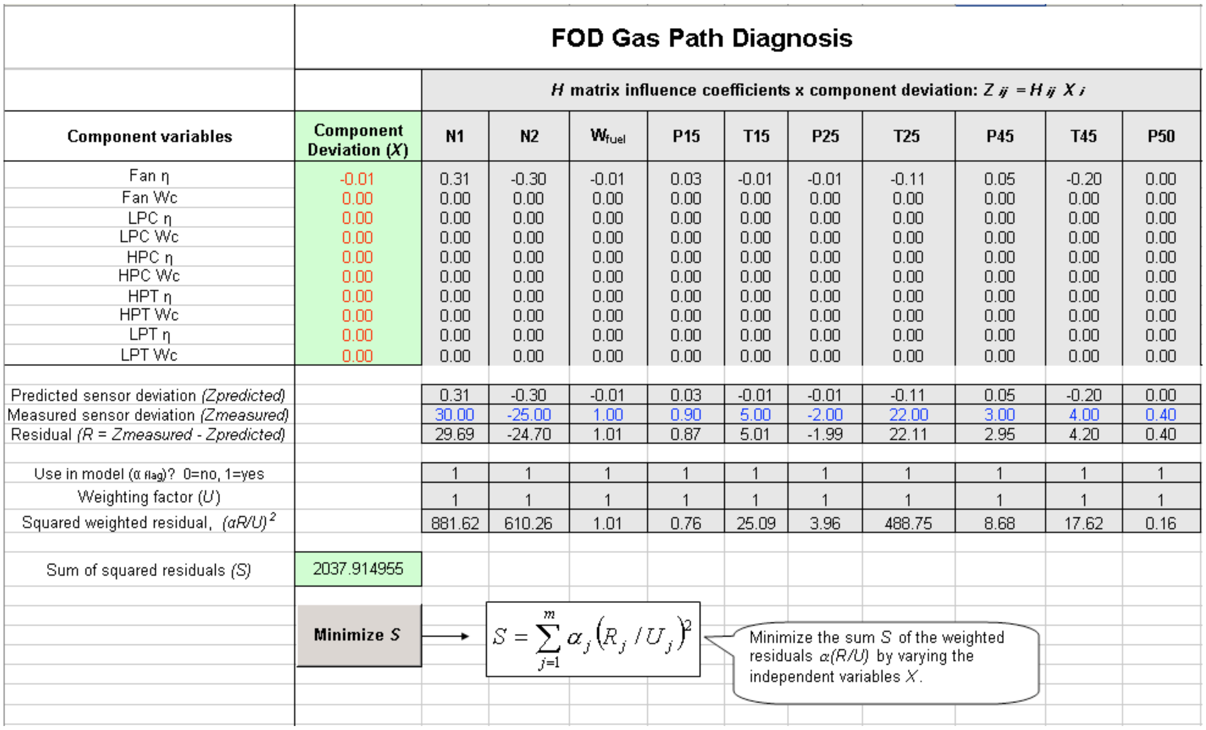
FOD detection/diagnosis software tool

The foregoing investigations laid the foundation for the development of a FOD detection/diagnosis software tool. The objective was to implement the methodologies within a flexible, generic architecture on a commonly available computer platform (Windows 2000/XP) with as little reliance as possible on commercial software. The initial approach to this requirement was to create an Excel spreadsheet as illustrated in Figure 23. The component (health parameter) deviations X are listed vertically in green and the corresponding contributions to the sensor measurement deviations in the gray area. Each cell in the gray area contains a simple equation of the form

$$Z_{ij} = H_{ij}\, X_i \tag{31}$$

where,

Z_{ij} = contribution to the jth sensor deviation from the ith component deviation
H_{ij} = sensitivity coefficients input by the user (elements of the H matrix)
X_i = guessed value of the deviation of the ith component



FOD Gas Path Diagnosis

		H matrix influence coefficients x component deviation: $Z_{ij} = H_{ij} X_i$									
Component variables	**Component Deviation (*X*)**	**N1**	**N2**	**W_{fuel}**	**P15**	**T15**	**P25**	**T25**	**P45**	**T45**	**P50**
Fan η	-0.01	0.31	-0.30	-0.01	0.03	-0.01	-0.01	-0.11	0.05	-0.20	0.00
Fan Wc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LPC η	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LPC Wc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HPC η	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HPC Wc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HPT η	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
HPT Wc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LPT η	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
LPT Wc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Predicted sensor deviation *(Zpredicted)*		0.31	-0.30	-0.01	0.03	-0.01	-0.01	-0.11	0.05	-0.20	0.00
Measured sensor deviation *(Zmeasured)*		30.00	-25.00	1.00	0.90	5.00	-2.00	22.00	3.00	4.00	0.40
Residual *(R = Zmeasured - Zpredicted)*		29.69	-24.70	1.01	0.87	5.01	-1.99	22.11	2.95	4.20	0.40
Use in model (α flag)? 0=no, 1=yes		1	1	1	1	1	1	1	1	1	1
Weighting factor *(U)*		1	1	1	1	1	1	1	1	1	1
Squared weighted residual, $(\alpha R/U)^2$		881.62	610.26	1.01	0.76	25.09	3.96	488.75	8.68	17.62	0.16
Sum of squared residuals *(S)*	2037.914955										

$$S = \sum_{j=1}^{m} \alpha_j \left(R_j / U_j\right)^2$$

Minimize the sum *S* of the weighted residuals $\alpha(R/U)$ by varying the independent variables *X*.

Figure 23 – Spreadsheet engine FOD diagnostic tool

The total predicted deviation of the *j*th sensor (Z_j) is the sum of the column values (ΣZ_{ij}). These values are compared to the measured deviations (Z_{measured}) to form a set of residuals *R*. The goal is to minimize the sum *S* of the squared weighted residuals as indicated and is accomplished with Excel's minimization function. Ideally, *S = 0*, which would indicate that the set of calculated component deviations produced a set of predicted measurement deviations that exactly match the measured sensor deviations. Hence, the magnitude of S is an indicator of the methodologies accuracy. An obvious shortcoming of the spreadsheet implementation is the large amount of manual labor required to fetch the H_{ij} coefficient values from $n + 1$ cycle code output cases and transcribe these values into the spreadsheet. This error-prone, tedious process must be repeated for each operating point of interest and for each engine of interest. Moreover, this particular formulation does not account for system and sensor noise – a serious shortcoming as discussed previously.

A much improved implementation was subsequently developed that avoids these shortcomings while also adding the vibration analysis method described above; namely, a C++ GUI-driven code entitled FODID (Foreign Object Damage Identification). FODID interfaces its GUI with NPSS to automatically generate the required H influence matrix and predict component deviations given a set of sensor deviations. The primary purpose of FODID is to serve as a prototype computational tool for diagnosing damage caused by FOD events.

The actual gas path mathematical method utilized is known as maximum a-posteriori which is essentially a generalization of the weighted least squares approach. In form, it closely resembles the extended Kalman filter approach discussed above. We let X_e be a n_e-vector of component deviations (i.e., the health parameter deviations), and augment it by an m-vector of sensor errors $X_{s.}$ That is, the total independent deviation vector $X = X_e \mid X_s$ and is of dimension $n = n_e + m.$ The solution is:

$$X = \left[P_0^{-1} + H^T R^{-1} H\right]^{-1} H^T R^{-1} Z \qquad (32)$$

where Z is the m-vector of sensor deviations, P_0 is the n x n state covariance matrix, and R is the m x m measurement covariance matrix that captures the effect of sensor measurement uncertainties. If we delete P_0 we have a weighted least squares formulation. R is computed from:

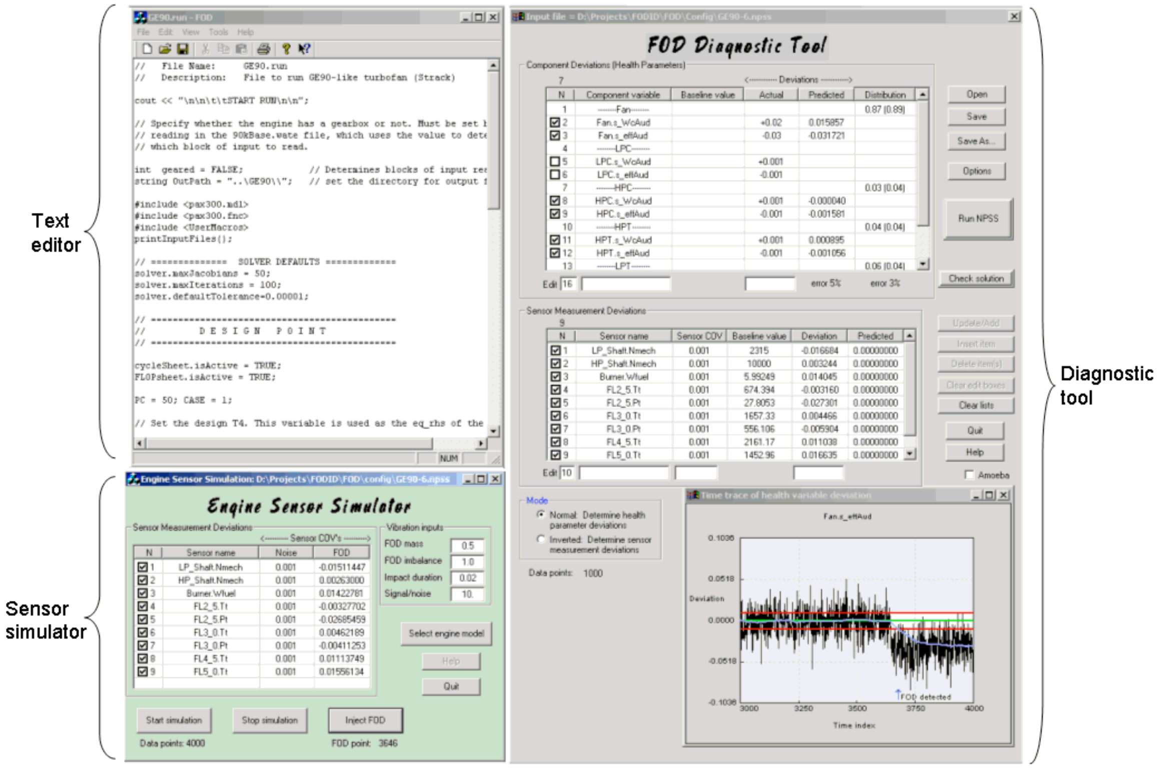
$$R = H_s \Sigma H_s^T \qquad (33)$$

where Σ is an m x m diagonal matrix of squared sensor measurement standard deviations:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \searrow & \\ & & & \sigma_m^2 \end{pmatrix} \qquad (34)$$

In general, the sensor influence coefficient matrix H_s relates changes in corrected measurement deviations that account for non-standard conditions (inlet temperature and pressure variations, and engine pressure ratio variation) to changes in raw measurement deviations and therefore includes 3 additional rows and columns (ref. 2). But in the current formulation we presume that such corrections are not required because the baseline values also include their effects. Hence, H_s is simply an identity matrix. For the same reason, Σ would also be expanded to include 3 more terms in a more general formulation.

A typical FODID user interface is illustrated in Figure 24 and consists of several interrelated windows: (1) a text editor that normally displays the baseline NPSS primary input file, (2) the main diagnostic tool that defines the problem, sets options, edits deviation values, invokes NPSS, and displays output, and (3) an optional engine sensor simulator that supplies a continuous stream of noisy sensor data for the diagnostic tool to analyze.



```
//   File Name:     GE90.run
//   Description:   File to run GE90-like turbofan (Strack)

cout << "\n\n\t\tSTART RUN\n\n";

// Specify whether the engine has a gearbox or not. Must be set b
// reading in the 90kBase.wate file, which uses the value to dete
// which block of input to read.

int  geared = FALSE;            // Determines blocks of input rea
string OutPath = "..\GE90\\";   // set the directory for output f

#include <pax300.mdl>
#include <pax300.fnc>
#include <UserMacros>
printInputFiles();

// ==============  SOLVER DEFAULTS =============
solver.maxJacobians = 50;
solver.maxIterations = 100;
solver.defaultTolerance=0.00001;

// ==============================================
//          D E S I G N   P O I N T
// ==============================================

cycleSheet.isActive = TRUE;
FLOPsheet.isActive = TRUE;

PC = 50; CASE = 1;

// Set the design T4. This variable is used as the eq_rhs of the
```

Component Deviations (Health Parameters)

7

N	Component variable	Baseline value	Actual	Predicted	Distribution
1	-------Fan-------				0.87 (0.89)
☑ 2	Fan.s_WcAud		+0.02	0.015857	
☑ 3	Fan.s_effAud		-0.03	-0.031721	
4	-------LPC-------				
□ 5	LPC.s_WcAud		+0.001		
□ 6	LPC.s_effAud		-0.001		
7	-------HPC-------				0.03 (0.04)
☑ 8	HPC.s_WcAud		+0.001	-0.000040	
☑ 9	HPC.s_effAud		-0.001	-0.001581	
10	-------HPT-------				0.04 (0.04)
☑ 11	HPT.s_WcAud		+0.001	0.000895	
☑ 12	HPT.s_effAud		-0.001	-0.001056	
13	-------LPT-------				0.06 (0.04)

Edit 16 error 5% error 3%

Sensor Measurement Deviations

9

N	Sensor name	Sensor COV	Baseline value	Deviation	Predicted
☑ 1	LP_Shaft.Nmech	0.001	2315	-0.016684	0.00000000
☑ 2	HP_Shaft.Nmech	0.001	10000	0.003244	0.00000000
☑ 3	Burner.Wfuel	0.001	5.99249	0.014045	0.00000000
☑ 4	FL2_5.Tt	0.001	674.394	-0.003160	0.00000000
☑ 5	FL2_5.Pt	0.001	27.8053	-0.027301	0.00000000
☑ 6	FL3_0.Tt	0.001	1657.33	0.004466	0.00000000
☑ 7	FL3_0.Pt	0.001	556.106	-0.005904	0.00000000
☑ 8	FL4_5.Tt	0.001	2161.17	0.011038	0.00000000
☑ 9	FL5_0.Tt	0.001	1452.96	0.016635	0.00000000

Edit 10

Mode
● Normal: Determine health parameter deviations
○ Inverted: Determine sensor measurement deviations

Data points: 1000

Sensor Measurement Deviations

N	Sensor name	Noise	FOD
☑ 1	LP_Shaft.Nmech	0.001	-0.01511447
☑ 2	HP_Shaft.Nmech	0.001	0.00263000
☑ 3	Burner.Wfuel	0.001	0.01422781
☑ 4	FL2_5.Tt	0.001	-0.00327702
☑ 5	FL2_5.Pt	0.001	-0.02685459
☑ 6	FL3_0.Tt	0.001	0.00462189
☑ 7	FL3_0.Pt	0.001	-0.00411253
☑ 8	FL4_5.Tt	0.001	0.01113749
☑ 9	FL5_0.Tt	0.001	0.01556134

Vibration inputs

FOD mass	0.5
FOD imbalance	1.0
Impact duration	0.02
Signal/noise	10.

Data points: 4000

FOD point: 3646

Figure 24 - Typical FODID screen display to users.

FODID can be operated in either of two modes – static mode or dynamic mode as indicated in Figure 25. In the static mode, a single diagnosis is normally performed on user-defined sensor deviations. However, it is also possible request FODID to invert this process and determine sensor deviations given a set of component deviations. In the dynamic mode, a continuous sequence of diagnoses is performed on the input sensory data stream supplied by the independently running Engine Sensor Simulator code.

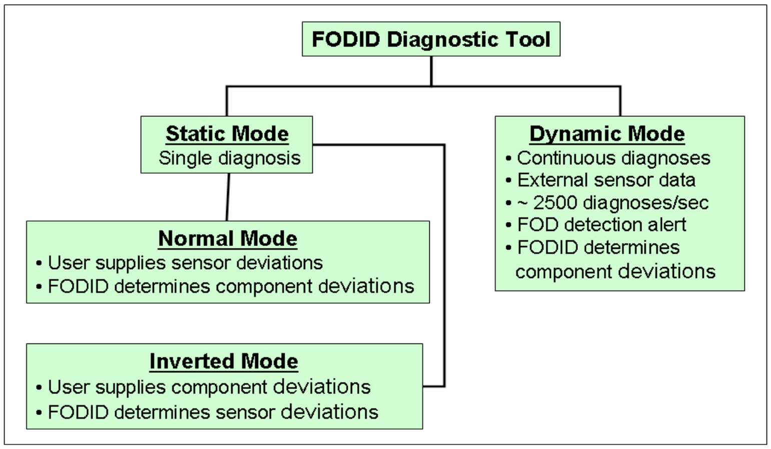
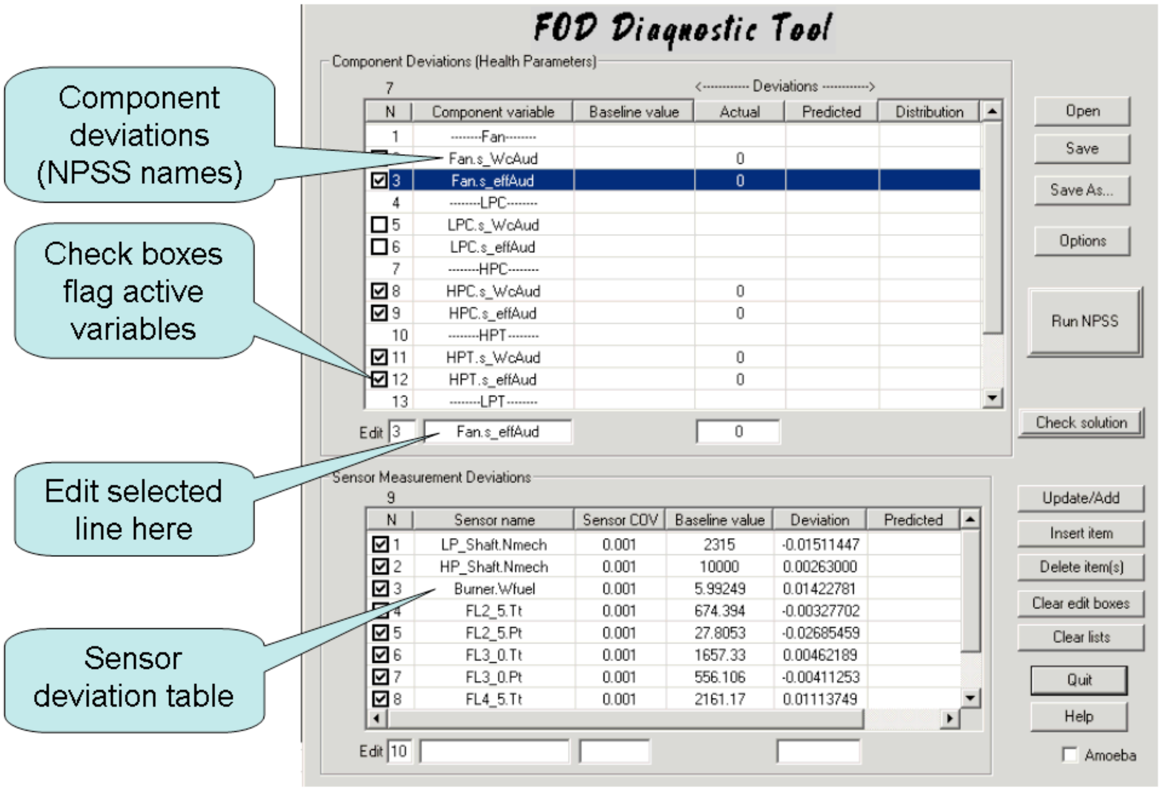

FODID Diagnostic Tool
Static Mode
Single diagnosis
Dynamic Mode
• Continuous diagnoses
• External sensor data
• ~ 2500 diagnoses/sec
• FOD detection alert
• FODID determines component deviations
Normal Mode
• User supplies sensor deviations
• FODID determines component deviations
Inverted Mode
• User supplies component deviations
• FODID determines sensor deviations

Figure 25 – FODOD mode choices

In all modes, the user is presented with the main dialog shown in Figure 26. The upper portion controls the input and output for the component deviations, and the lower portion controls the sensor inputs and outputs. In the upper portion, a list of user-defined component health parameters appears in column 2 grouped according to user-defined component names (i.e., the entries without checkboxes in column 1), and a similar sensor name list appears in the lower table. These are NPSS-specific names that must be defined within the NPSS engine model. Column 1 enables users to turn these deviations on or off by clicking the appropriate check boxes.

Both tables also include a *Baseline value* column that is filled in automatically after invoking NPSS. The *Sensor COV* column contains a list of sensor Coefficients of Variation (standard deviations normalized by baseline values) to define typical sensor signal noise levels (actual sensor noise plus any associated signal processing noise). The *Deviation* column in the Sensor Measurement Deviation table must also be input by the user to specify the FOD-induced sensor measurement deviations. All of these deviation values are normalized by the corresponding baseline values to give them equal weights.

The tool permits users to add, delete, or modify some of these entries using the edit boxes underneath each table and the editing buttons on the far right hand side. Clicking anywhere within a table row automatically fills the edit boxes with the appropriate column values. The information in these 2 tables constitutes a “problem configuration” and may be saved and recalled using the *Open, Save,* and *Save As*…buttons.



FOD Diagnostic Tool

Component Deviations (Health Parameters)

7

<----------- Deviations ----------->

N	Component variable	Baseline value	Actual	Predicted	Distribution
1	--------Fan--------				
	Fan.s_WcAud		0		
☑ 3	Fan.s_effAud		0		
4	--------LPC--------				
□ 5	LPC.s_WcAud				
□ 6	LPC.s_effAud				
7	--------HPC--------				
☑ 8	HPC.s_WcAud		0		
☑ 9	HPC.s_effAud		0		
10	--------HPT--------				
☑ 11	HPT.s_WcAud		0		
☑ 12	HPT.s_effAud		0		
13	--------LPT--------				

Edit 3 Fan.s_effAud 0

Sensor Measurement Deviations

9

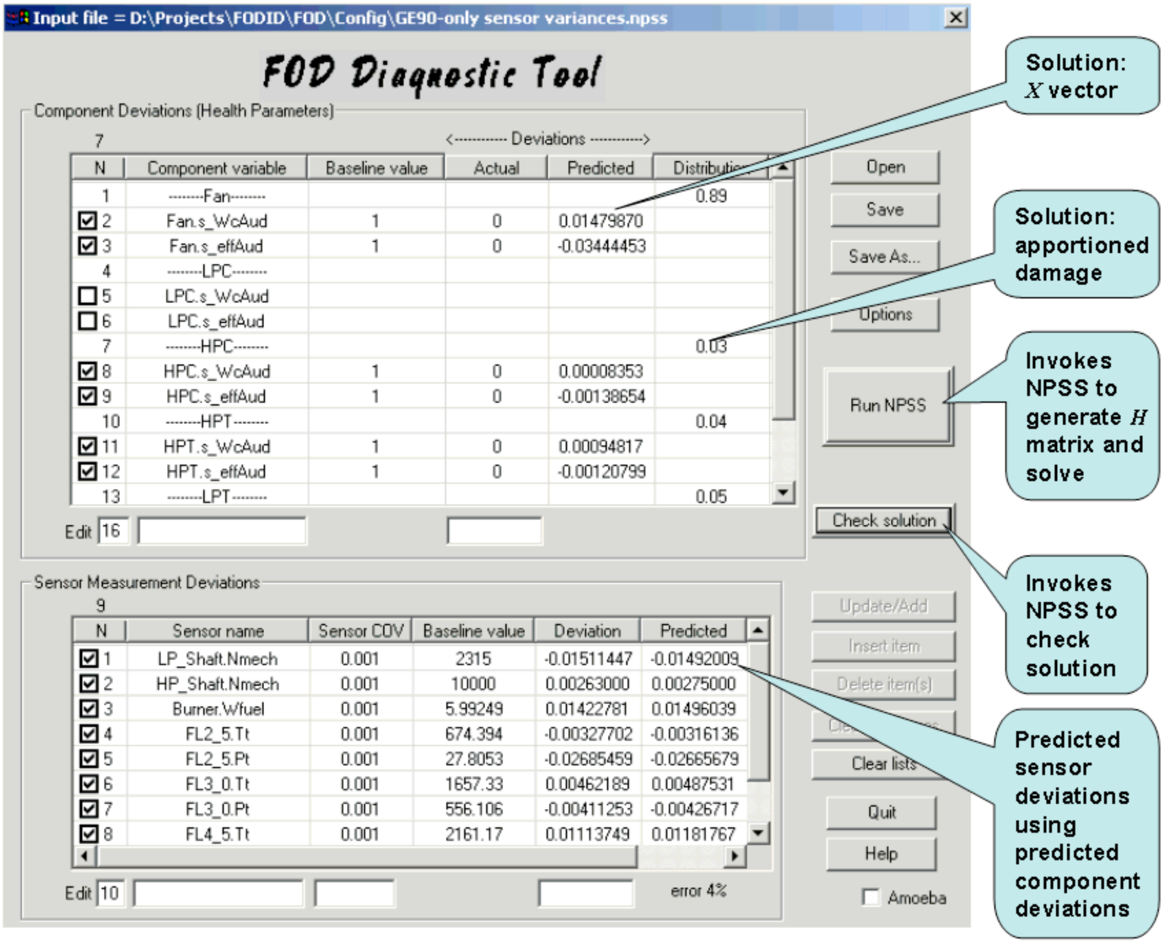
N	Sensor name	Sensor COV	Baseline value	Deviation	Predicted
☑ 1	LP_Shaft.Nmech	0.001	2315	-0.01511447	
☑ 2	HP_Shaft.Nmech	0.001	10000	0.00263000	
☑ 3	Burner.Wfuel	0.001	5.99249	0.01422781	
☑ 4	FL2_5.Tt	0.001	674.394	-0.00327702	
☑ 5	FL2_5.Pt	0.001	27.8053	-0.02685459	
☑ 6	FL3_0.Tt	0.001	1657.33	0.00462189	
☑ 7	FL3_0.Pt	0.001	556.106	-0.00411253	
☑ 8	FL4_5.Tt	0.001	2161.17	0.01113749	

Edit 10

Figure 26 – Defining the problem.

Normal static mode – This steady state mode requires users to input a set of sensor variables and supply their deviations. Figure 26 illustrates the appearance of the dialog after this step and before a diagnosis is performed. The sensor deviations are defined, the component actual deviations are zeroed, and the component predicted deviation column is blank. Once these inputs are defined, and the user clicks the *Run NPSS* button, the code prepares an NPSS input file and invokes the NPSS simulation code multiple times to generate baseline values of the sensor variables as well as n sets of perturbed values (obtained by sequentially perturbing the n component deviations). FODID then harvests the n+1 NPSS output files to obtain the response variable values it needs to solve equation 7. After solving this equation it presents the results to the user on-screen as a column of predicted component deviations as illustrated in Figure 27 It also predicts how much of the total deviation is due to each component as a rough indicator of which components are most severely damaged. These normalized damage allocations are enumerated in the *Distribution* column.

The diagnostic solution may be checked by clicking the *Check solution* button. Doing so invokes NPSS with the solution X-vector to calculate a predicted sensor deviation vector $Z_{predicted}$. Comparing the predicted sensor deviation vector to the measured sensor deviation vector yields an error estimate that appears underneath the sensor predicted deviation column.



Component Deviations (Health Parameters)

7

<------------ Deviations ------------>

N	Component variable	Baseline value	Actual	Predicted	Distributi
1	--------Fan--------				0.89
☑ 2	Fan.s_WcAud	1	0	0.01479870	
☑ 3	Fan.s_effAud	1	0	-0.03444453	
4	--------LPC--------				
□ 5	LPC.s_WcAud				
□ 6	LPC.s_effAud				
7	--------HPC--------				0.03
☑ 8	HPC.s_WcAud	1	0	0.00008353	
☑ 9	HPC.s_effAud	1	0	-0.00138654	
10	--------HPT--------				0.04
☑ 11	HPT.s_WcAud	1	0	0.00094817	
☑ 12	HPT.s_effAud	1	0	-0.00120799	
13	--------LPT--------				0.05

Edit 16

Sensor Measurement Deviations

9

N	Sensor name	Sensor COV	Baseline value	Deviation	Predicted
☑ 1	LP_Shaft.Nmech	0.001	2315	-0.01511447	-0.01492009
☑ 2	HP_Shaft.Nmech	0.001	10000	0.00263000	0.00275000
☑ 3	Burner.Wfuel	0.001	5.99249	0.01422781	0.01496039
☑ 4	FL2_5.Tt	0.001	674.394	-0.00327702	-0.00316136
☑ 5	FL2_5.Pt	0.001	27.8053	-0.02685459	-0.02665679
☑ 6	FL3_0.Tt	0.001	1657.33	0.00462189	0.00487531
☑ 7	FL3_0.Pt	0.001	556.106	-0.00411253	-0.00426717
☑ 8	FL4_5.Tt	0.001	2161.17	0.01113749	0.01181767

Edit 10

error 4%

Solution: X vector

Solution: apportioned damage

Invokes NPSS to generate H matrix and solve

Invokes NPSS to check solution

Predicted sensor deviations using predicted component deviations

Figure 27 – FODID dialog appearance after diagnosis

Inverted static mode – FODID can also generate a set of simulated sensor deviations given a set of component deviations. This is the inverse of the normal static problem and it utilizes NPSS to predict the sensor deviations. If users run this mode first and then run the normal static mode, there will be two columns of component deviations – actual values and predicted values. Using this sequence, it is possible to determine an accuracy metric for the diagnostic solution by comparing the differences between the predicted and actual component deviations. FODID displays an overall accuracy metric labeled *Error* underneath the component *Predicted Deviation* column. It is calculated as the weighted RMS average component error; that is, the errors for each component are averaged and the contribution from each component to the total is weighted according to its magnitude:

$$error = \sqrt{\sum_{i=1}^{N}\left[w_i\left(\frac{\overline{X}_{i,pre}}{\overline{X}_{i,act}} - 1\right)\right]^2} \qquad (35)$$

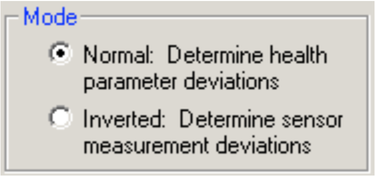
where w_i are the weighting factors, $\overline{X}_{i,pre}$ is the average predicted deviation for component i, $\overline{X}_{i,act}$ is the average actual deviation, and N is the number of components:

$$w_i = \frac{\overline{X}_{i,act}}{\sum_{j=1}^{k}\overline{X}_{j,act}} \qquad \overline{X}_{i,pre} = \frac{\sum_{j=1}^{k} X_{j,pre}}{k} \qquad \overline{X}_{i,act} = \frac{\sum_{j=1}^{k} X_{j,act}}{k} \qquad (36)$$

Here, k is the number of health parameters associated with a single component (typically, $k = 2$), $X_{j,pre}$ is the predicted deviation of parameter j, and $X_{j,act}$ is the actual deviation of parameter j.

A similar error metric is displayed underneath the *Distribution* column and represents the average discrepancy between the actual and predicted damaged allotments listed.

Options – Switching from normal static mode to inverted static mode (or vice versa) is accomplished by setting the Mode selector switch (Figure 28) on the Diagnostic Dialog.



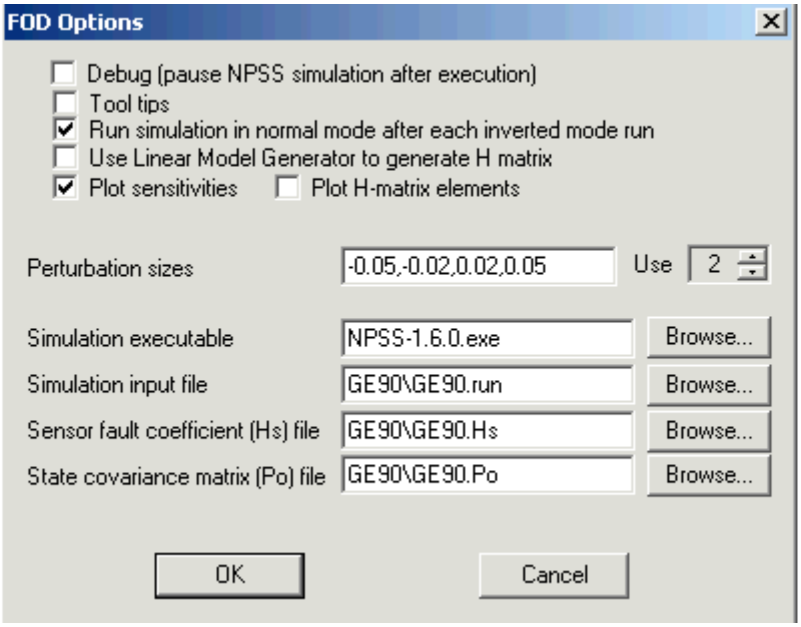
Mode

● Normal: Determine health parameter deviations

○ Inverted: Determine sensor measurement deviations

Figure 28 – Mode selection switch

Here, the Normal static mode is labeled "Normal: Determine health parameter deviations" and the inverted static mode labeled "Inverted: Determine sensor measurement deviations." Other options may be set by clicking the *Options* button which brings up the Options Dialog (Figure 29).



□ Debug (pause NPSS simulation after execution)
□ Tool tips
☑ Run simulation in normal mode after each inverted mode run
□ Use Linear Model Generator to generate H matrix
☑ Plot sensitivities □ Plot H-matrix elements

Perturbation sizes: -0.05,-0.02,0.02,0.05 Use 2

Simulation executable: NPSS-1.6.0.exe
Simulation input file: GE90\GE90.run
Sensor fault coefficient (Hs) file: GE90\GE90.Hs
State covariance matrix (Po) file: GE90\GE90.Po

Figure 29 – Options dialog

Normally, when NPSS is invoked an output window is automatically opened to enable users to monitor the progress of the NPSS run. This console window is also automatically closed upon completion of the NPSS run. This console window is created and destroyed so quickly, however, that users cannot spot any trouble that might arise. The "*Debug"* option provides this capability by pausing the NPSS output window after each run before destroying it.

The "*Tool tips*" option turns on the feature that displays a brief help message whenever the mouse cursor hovers over a control in the dialog.

The "*Run simulation in normal mode after each inverted mode run*" option causes FODID to perform a normal static mode diagnosis immediately after each inverted static mode case.

The "*Use Linear Model Generator to generate the H matrix*" option causes NPSS to use its LMG feature to generate the *H* matrix. However, this is not recommended since this feature is not as reliable as the default method in producing accurate *H* matrices.

The "*Plot sensitivities*" option causes FODID to display a sensitivity chart (Figure 30) each time an active component row is clicked (upper table). The chart appears in the lower right side of the diagnostic dialog and shows the sensitivity of each sensor deviation to the selected component deviation and the degree of linearity. This information is valuable because it enables users to quickly verify that there is at least one sensor deviation that is strongly dependent on the selected component deviation. If this condition is not satisfied, the diagnostic solution is likely to incur substantial error. Four perturbation sizes are used to generate this chart and they are listed in the Perturbation sizes edit box. Users may alter these values if desired. Also, they may select any one of the four perturbation sizes to use in the solution method (the second size is the default value). An alternative chart that plots the relevant *H*-matrix elements is available by selecting the "*Plot H-matrix elements*" box. These plots may be printed or saved to the clipboard by right clicking the mouse to bring up a context menu.

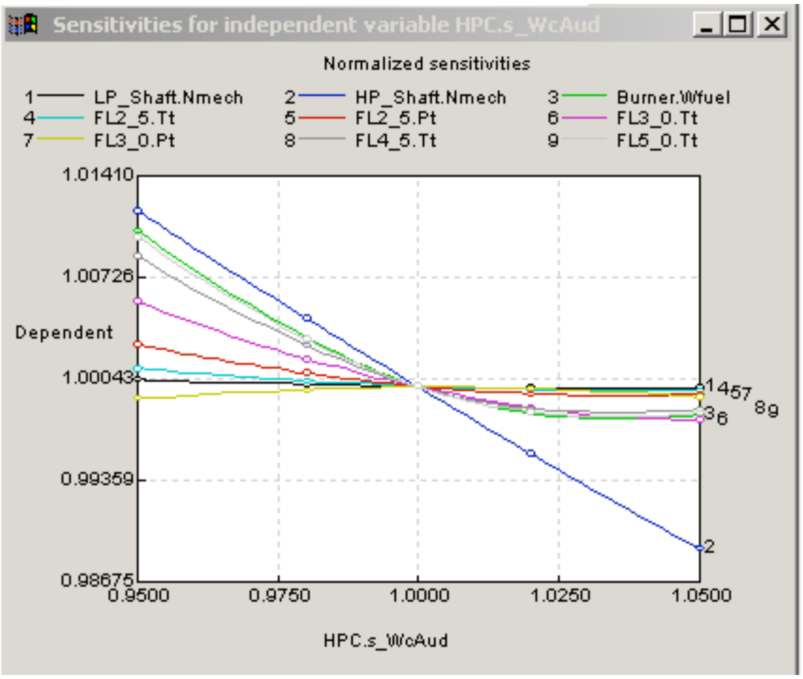


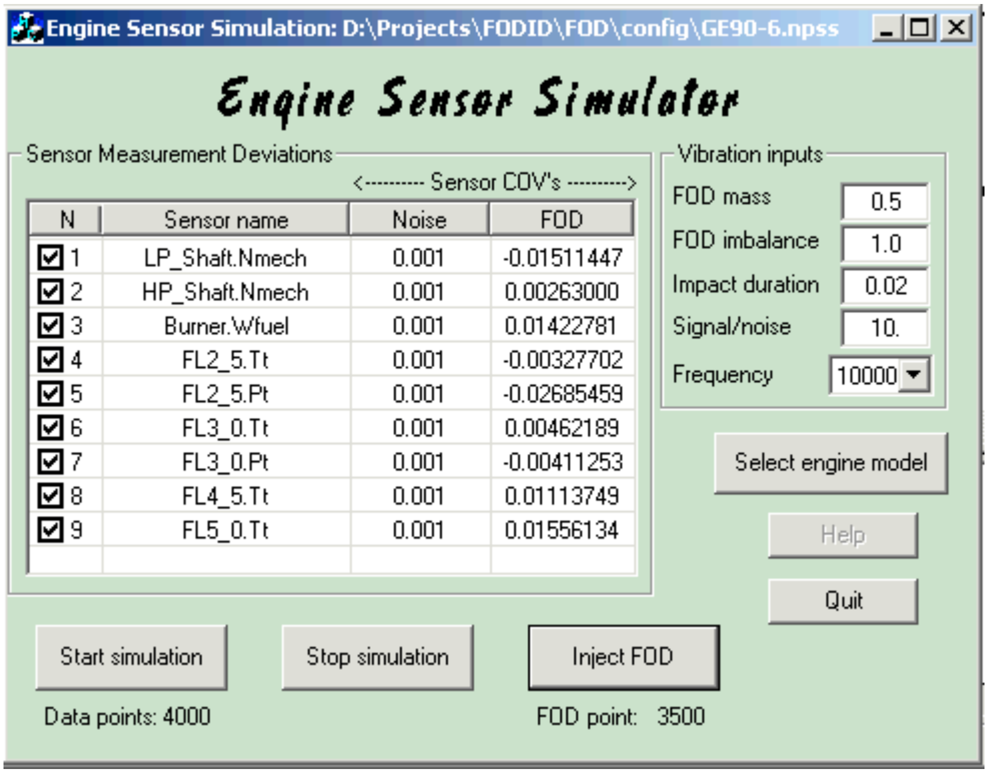
Figure 30 – Representative sensitivity chart

The remaining 4 option items are all file names that inform FODID where to find:

- The NPSS executable file
- The NPSS primary input file
- The sensor fault matrix (H_s) file
- The state covariance matrix (P_0) file

<u>Dynamic mode</u> – In this mode a separate code called the Engine Sensor Simulator, running in parallel with FODID, generates a continuous stream of noisy sensor data that it transmits to FODID in groups of 1000 or 10,000 datasets at a time via a file transfer mechanism. Each dataset consists of a time index value and a vector of sensor measurement values ($Z_{measured}$). Each element of $Z_{measured}$ is generated randomly using a Gaussian distribution and the user-defined standard deviation vector that appears in the lower table of the FODID diagnostic dialog. At any time after the Engine Sensor Simulator is started, a user may click the *"Inject FOD"* button to initiate an FOD event (see Figure 31). Then every subsequent dataset will contain both random noise and the sensor deviations defined in the lower table of the FODID diagnostic table. For convenience, the list of sensor names, the noise COV vector, and the FOD deviation vector are all listed in the Engine Sensor Simulator dialog. By default these values are identical to the values used by FODID, but users may select another set if they desire by clicking the *"Select engine model"* button.

The current time index value is continuously updated after the *"Data points:"* header on the Engine Sensor Simulator dialog and a similar update takes place on the FODID dialog. The updates are a visual clue to the user that data transfer is actually occurring. The data transfer bandwidth is on the order of 2500 datasets/second.


Engine Sensor Simulation: D:\Projects\FODID\FOD\config\GE90-6.npss

Engine Sensor Simulator

Sensor Measurement Deviations

<--------- Sensor COV's --------->

N	Sensor name	Noise	FOD
☑ 1	LP_Shaft.Nmech	0.001	-0.01511447
☑ 2	HP_Shaft.Nmech	0.001	0.00263000
☑ 3	Burner.Wfuel	0.001	0.01422781
☑ 4	FL2_5.Tt	0.001	-0.00327702
☑ 5	FL2_5.Pt	0.001	-0.02685459
☑ 6	FL3_0.Tt	0.001	0.00462189
☑ 7	FL3_0.Pt	0.001	-0.00411253
☑ 8	FL4_5.Tt	0.001	0.01113749
☑ 9	FL5_0.Tt	0.001	0.01556134

Vibration inputs

FOD mass 0.5

FOD imbalance 1.0

Impact duration 0.02

Signal/noise 10.

Frequency 10000

Data points: 4000

FOD point: 3500

Figure 31 – The Engine Sensor Simulator dialog

The Engine Sensor Simulator (ESS) may be invoked either separately or via the *Tools* menu on the text editor window. ESS synthesizes both gas path sensory information and vibration sensory information. The gas path sensor signals contain just normally distributed noise until the user requests an FOD event Then the noise and the FOD deviations are added before being transmitted to FODID. The 8 vibration signals are synthesized using results from a specific reduced-order structural analysis model (ref. 11) and the user-selected FOD parameters listed on the dialog.

Once ESS is started, FODID will process each dataset it receives looking for an FOD event. It does this by using each transmitted dataset in conjunction with equation 32 to generate a continuous stream of gas path diagnoses, and examining each resulting component deviation vector (X_e) to see if any preset threshold violations have occurred. To prevent random noise from triggering a false detection, only 100-point moving averages are used to detect threshold violations. Once an FOD event is detected, FODID displays the event in a time history chart as illustrated in Figure 32 and updates the solution information in the deviation table.

Similarly, FODID will examine the raw vibration data looking for and flagging threshold violations. In addition, it will perform a continuous stream of wavelet analyses using the process defined in figure 16 to isolate small FOD events masked by noise. Whenever an FOD event is detected via either type of vibration analysis, FODID will display a time history chart and update the deviation table.

Triggering a simulated FOD event within the ESS also causes ESS to cease generating and transmitting data after the set of data containing the FOD event is transmitted.

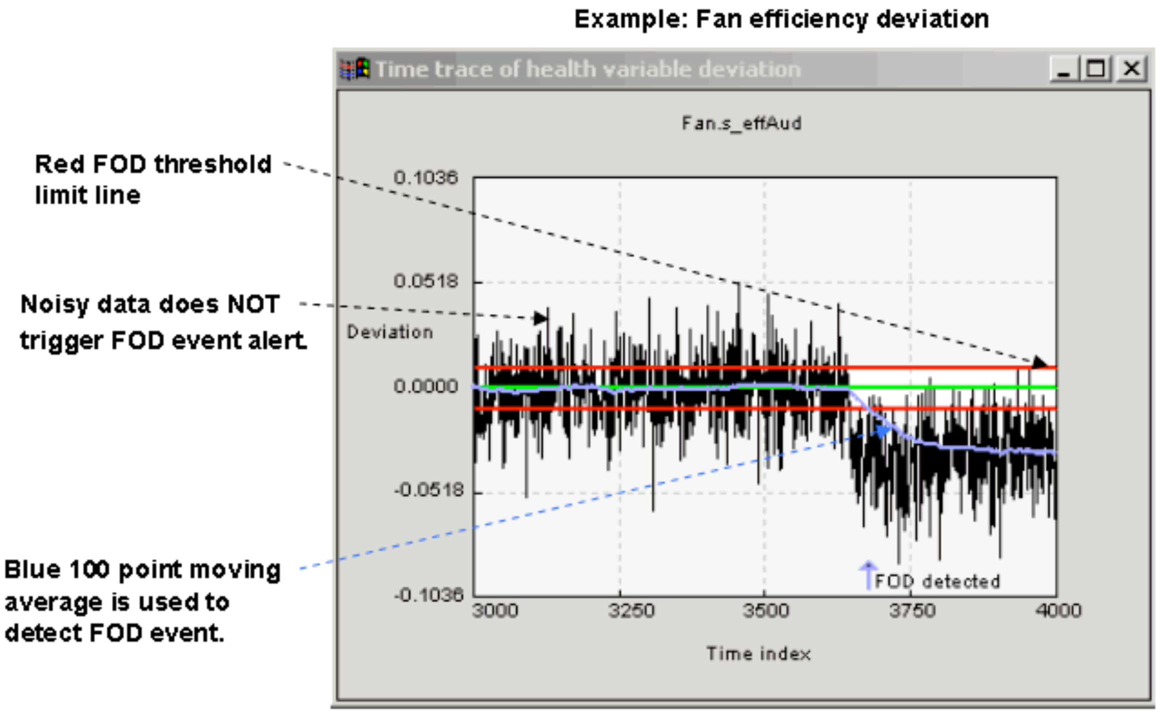


Figure 32 – Example FOD detection chart with 100-point moving average violating red threshold line.

Users may select whichever health parameter deviation or sensor deviation time trace they wish to view by using the right-button context menu option as displayed in Figure 33.

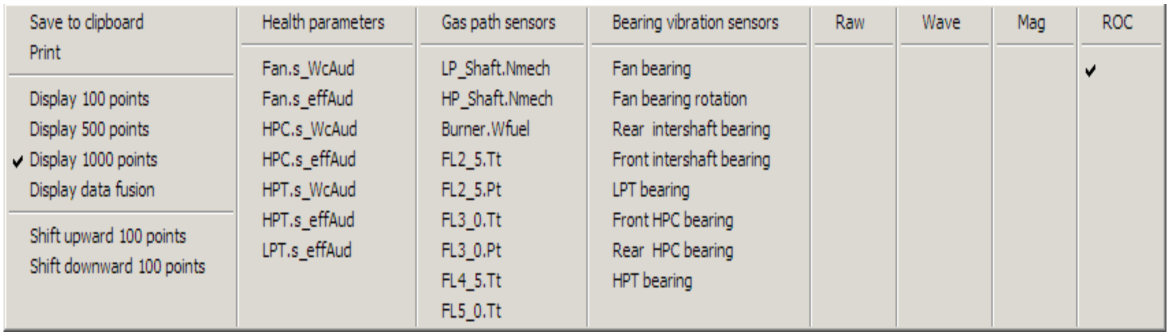


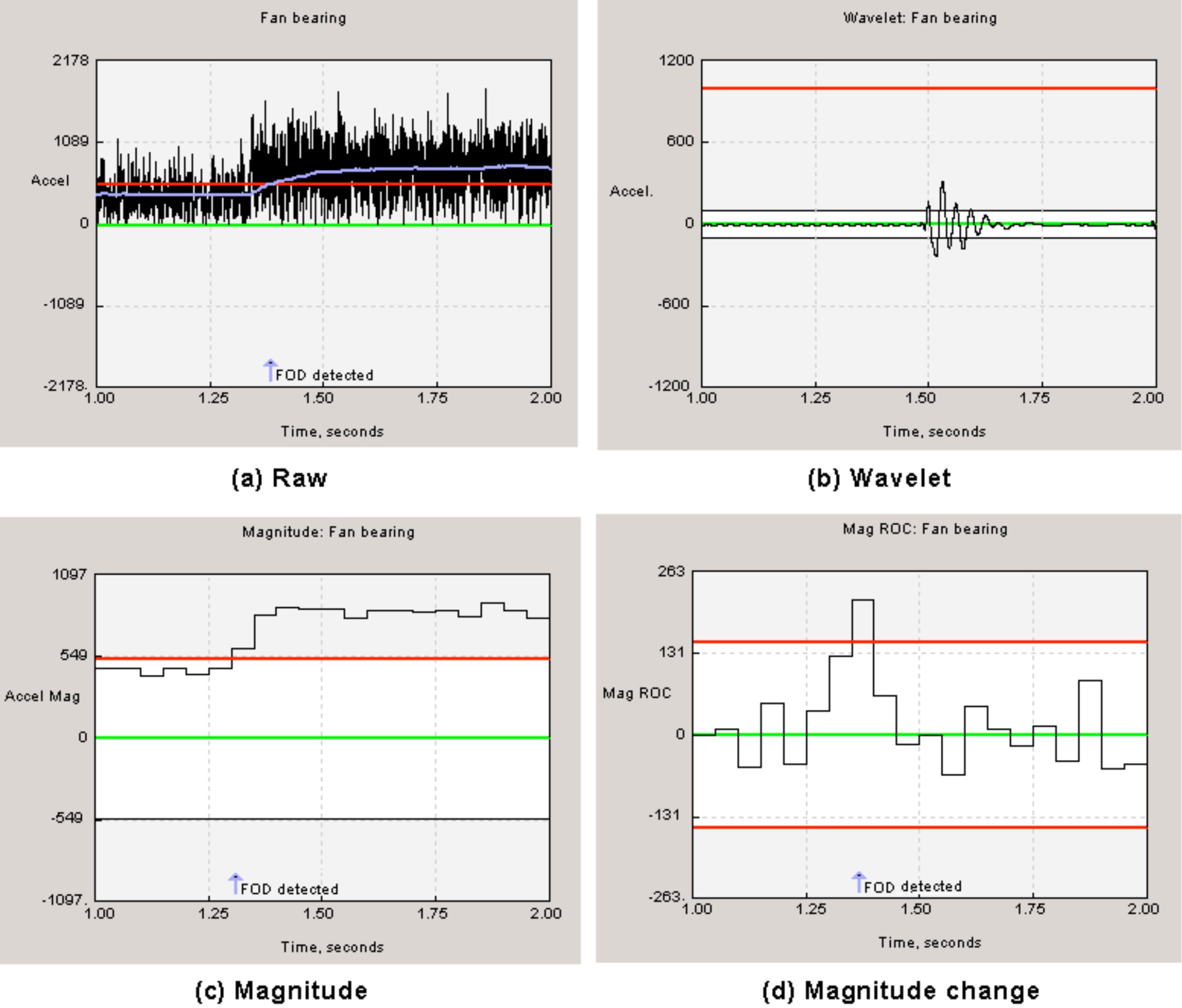
Figure 33 – Context menu for FOD detection charts.

Vibration sensor data may be plotted four different ways according to the 4 rightmost column headings:

- Raw: the vibration data is plotted without any signal processing
- Wave: the vibration data is processed using the Daubechies wavelet analysis
- Mag: the vibration data is grouped into RMS-averaged blocks
- ROC: the changes from block-to-block are plotted.

Figure 34 illustrates these 4 choices for the fan bearing data and 6 levels of wavelet filtering.

A checkmark in the context menu (Fig. 33) indicates the currently selected plot. Users may print or save these plots, and can adjust the plot resolution using the same menu.



(a) Raw

(b) Wavelet

(c) Magnitude

(d) Magnitude change

Figure 34 – Four different signal processing plots for bearing vibration data.

<u>Data fusion</u> – Since the detection of an FOD event is via the monitoring of many different parameters, some thermodynamic from the gas path analysis while others are vibration related, any one of which might trigger an alert, some means of consolidating all of this information is needed. Doing so with fuzzy logic yields a composite detection indicator that weighs the evidence provided by both the gas path and vibration analyses.

In fact, even within the gas path domain the need for data fusion arises because we typically have multiple parameters for each component (e.g., efficiency and flow capacity) that may yield dissimilar diagnoses. Thus, we first fuse the gas path results for each component as well as the multiple vibration analysis results before fusing the gas path data with the vibration data (Figure 35). We chose to simply fuse two of the vibration measurements: the fan and LPT moving average bearing accelerations normalized by their respective threshold values.

Finally, the gas path data fusion results can be fused to the vibration fusion results using the fused fan result.

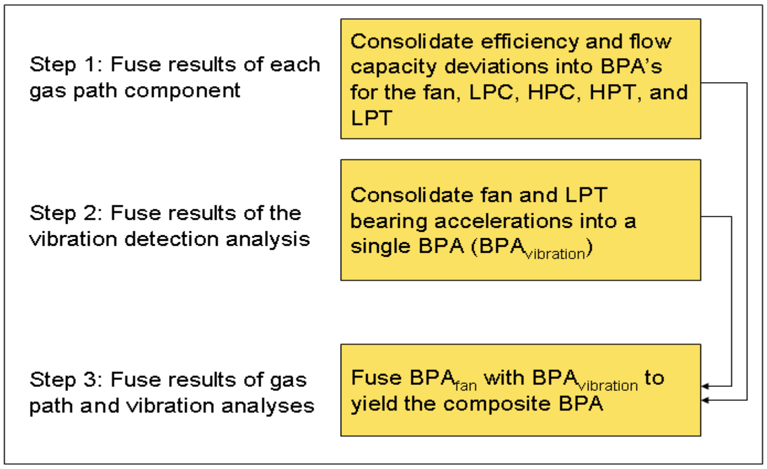

Step 1: Fuse results of each gas path component
Consolidate efficiency and flow capacity deviations into BPA's for the fan, LPC, HPC, HPT, and LPT
Step 2: Fuse results of the vibration detection analysis
Consolidate fan and LPT bearing accelerations into a single BPA (BPA$_{vibration}$)
Step 3: Fuse results of gas path and vibration analyses
Fuse BPA$_{fan}$ with BPA$_{vibration}$ to yield the composite BPA

Figure 35 – Steps to fuse the gas path and vibration analysis results into a composite result.

The fuzzy logic process to perform these steps requires defining a set of membership functions and combination rules that relate input values such as fan efficiency deviation to outputs (basic probability assignment, BPA) that ultimately determine a numerical score that represents the degree of belief that an FOD event has occurred (ref. 12). For example, we design the gas path membership functions and rules to yield an output value near zero for normal engine operation and near 1 if we are quite certain that an FOD event has occurred, accounting for all the evidence. Intermediate values represent lesser degrees of certainty that a damaging FOD event has occurred. Figure 36 displays the particular membership functions used by FODID. There are 3 fuzzy sets: normal, uncertain, and FOD. "Normal" represents normal operation with little suspicion that an FOD event has occurred. "Uncertain" means that the evidence is too weak to conclude that an FOD event occurred, but neither does the evidence strongly support a non-FOD event conclusion. "FOD" represents the cases where the evidence strongly supports that an FOD event occurred.

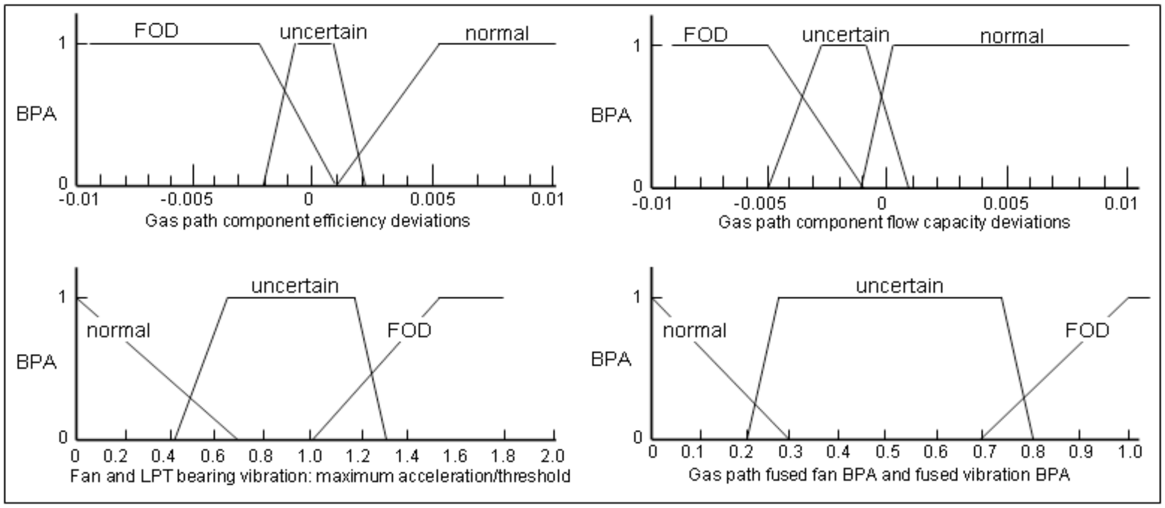


Figure 36 – Fuzzy logic membership functions.

The 9 rules that combine 2 input values into a single output value are:

1. If input 1 and input 2 are both normal, then the output is "Improbable"
2. If input 1 and input 2 are both FOD, then the output is "Probable"
3. If input 1 and input 2 are both uncertain, then the output is "Uncertain"
4. If input 1 is normal and input 2 is FOD, then the output is "Uncertain"
5. If input 2 is normal and input 1 is FOD, then the output is "Uncertain"
6. If input 1 is normal and input 2 is uncertain, then output is "Unlikely"
7. If input 2 is normal and input 1 is uncertain, then output is "Unlikely"
8. If input 1 is FOD and input 2 is uncertain, then output is "Likely"
9. If input 2 is FOD and input 1 is uncertain, then output is "Likely"

To obtain a numerical BPA value from the 5 output sets, we assign them values as follows:

probable	=	1.00
likely	=	0.75
uncertain	=	0.20
unlikely	=	0.10
improbable	=	0.05

We have used the same defuzzification rules for all 3 fusion processes, although this is quite arbitrary. The entire set of data fusion results may be displayed on a single chart using the context menu as illustrated in Figure 37. Here a 0.5 lb. object strikes the fan at 300 ft/s and 10 inches from the axis of rotation at time step 1500. The fused fan gas path results are strongly signaling fan damage, while the HPC and HPT are probably only slightly damaged, and the LPT undamaged. Both the fused vibration result (solid red line) and the overall fusion result (dashed red line) are signaling a very high probability of an FOD event shortly after the actual event occurred.

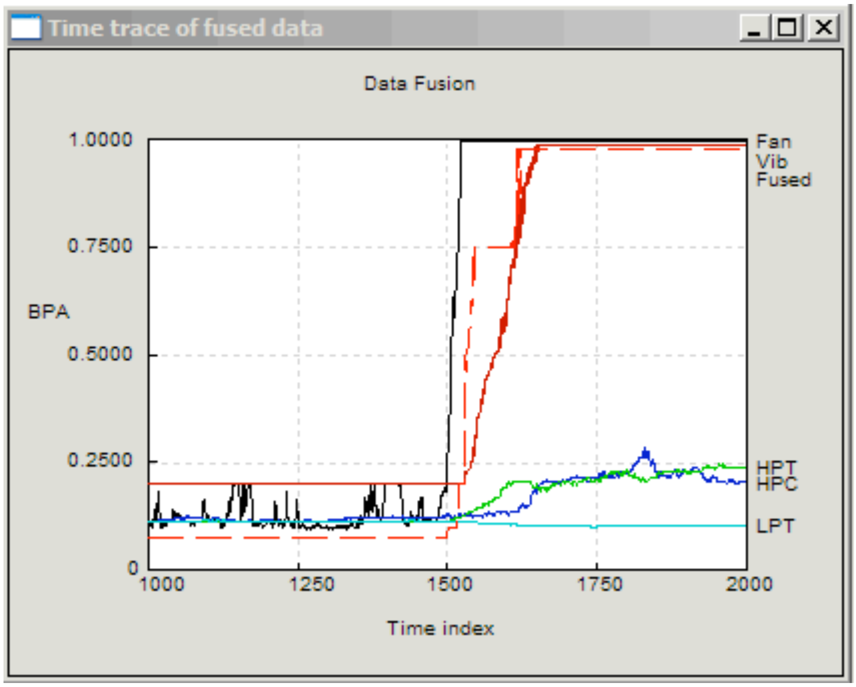


Figure 37 – Typical data fusion results. FOD occurred at time 1500 for a 0.5 lb. object impacting the fan at 10 inch radius and 300 ft/s. The fused vibration result is solid red, the overall fusion result is dashed red.

Concluding remarks

Although the FODID tool is useful in its current state of development, it should be viewed as a prototype tool that could be strengthened by the addition of several key features. For example, the diagnostic accuracy could probably be enhanced if multiple gas path diagnoses were performed at different operating conditions. And it would be more convenient if users could specify the operating point via a few button clicks rather than the current manual procedure of editing the baseline NPSS input file. Perhaps most importantly, a more realistic control system model that included variable geometry schedules and limits, bypass valve schedules, temperature limits, acceleration schedules, and other control logic elements would be required to represent real engines. Adding such control system modeling is a significant challenge for many potential users since this information is rarely disclosed in the open literature.

Another useful feature that could be added is a FOD detection threshold manager. Currently FODID uses fixed threshold values (or ratios) for the many different parameters being monitored. Users cannot modify these limits, but it would be preferable to provide users the ability to set these limits separately for each monitored parameter.

It would be advantageous to strengthen the data fusion technique in FODID. The fuzzy logic technique involves a host of membership function parameters and rules that currently require source code modification to change. A membership function manager would be a welcome addition. Also, alternative fusion strategies to the single representative one provided could be offered.

Most of the tool development effort was devoted to implementing the detection/diagnostic methodology including alternative techniques for presenting data, calculating moving averages efficiently, and so forth. Proportionately less time was spent in detailed explorations of the conditions under which the methodology performs well. Hence, further effort in validating the FODID code and discovering its practical limits of applicability would be valuable.

The thermodynamic engine model used in the charts represents a modern high bypass turbofan engine similar to a GE90, but FODID is not limited to this type of gas turbine engine. Any gas turbine that can be modeled with NPSS can be handled equally well. This basic flexibility was in fact one of the motivating factors in initiating this effort.

On the other hand, the vibration analysis implementation is not generic – it represents a specific high-bypass engine configuration. Structural dynamic model changes would require regenerating the dynamic response matrices externally (e.g., using the reduced-order DyRoBeS code or equivalent). Interfacing FODID with a structural dynamics code such as DyRoBeS would add considerable flexibility to the overall simulation.

References

1. Louis A. Urban, Gas Path Analysis Applied to Turbine Engine Condition Monitoring, AIAA paper No.72-1082, November, 1972.

2. A.J. Volponi, et al, Gas Turbine Condition Monitoring & Fault Diagnosis, von Karman Institute for Fluid Dynamics, Lecture Series 2003-01, January 13-17, 2003.

3. A.J. Volponi, et al, The Use of Kalman Filter and Neural Network Methodologies in Gas Turbine Performance Diagnostics: A Comparative Study, ASME 2000-GT-4, May, 2000.

4. Louis A. Urban and Allen Volponi, Mathematical Methods of Relative Engine Performance Diagnostics, SAE Trans., 101 Aerospace Technical Paper 922048, 1992.

5. D.L. Doel, An Assessment of Weighted-Least-Squares-Based Gas Path Analysis, ASME Trans., J. Eng. For Gas Turbines and Power, vol. 116, p.366-373, April, 1994.

6. R.H. Luppold, et al., Estimating In-Flight Engine Performance Variations Using Karman Filter Concepts, AIAA-89-2584, July 1989.

7. A. Stamatis, et al, Jet Engine Fault Detection with Differential Gas Path Analysis at Discrete Operating Points, ISABE 89-7133, 1989.

8. A.J. Volponi, et al, Development of an Information Fusion System for Engine Diagnostics and Health Management, NASA/TM—2004-212924, ARL-TR-3127, February, 2004.

9. D.L. Doel, TEMPER – A Gas Path Analysis Tool for Commercial Jet Engines, ASME J. Eng. For Gas Turbines and Power, Vol. 116, p.82-89, Jan. 1994.

10. L.J. Kerr, et al, Real-Time Estimation of Gas Turbine Engine Damage Using a Control Based Kalman Filter Algorithm, ASME 91-GT-216, June 1991.

11. Dan Simon and Donald Simon, Kalman Filtering with Inequality Constraints for Turbofan Engine Health Estimation, NASA/TM-2003-212111, ARL-TR-2899, Feb. 2003.

12. Takahisa Kobayashi and Donald Simon, Application of a Bank of Kalman Filters for Aircraft Engine Fault Diagnostics, ASME GT2003-38550, June, 2003.

13. James Turso, Charles Lawrence, Jonathan Litt, Reduced-Order Modeling and Wavelet Analysis of Turbofan Engine Structural Response Due to Foreign Object Damage (FOD) Events, NASA/TM-2004-213118, also ARL-TR-3256, August, 2004

14. James Turso, Jonathan Litt, A Foreign Object Damage Event Detector Data Fusion System for Turbofan Engines, AIAA 2004-4047, 2004.

15. William Strack, Gas Turbine Engine Foreign Object Damage Diagnostic Tool, FODID Users Manual, October, 2004.

Appendix A – Some Lessons Learned from Previous FOD Diagnostic Research

Researchers have been pursuing the development of diagnostic methodologies of jet engine faults for several decades. A review of the pertinent literature has revealed several important "lessons learned" as enumerated below.

Lesson 1 – The pure weighted least squares method is only adequate in an ideal situation – where there are no significant modeling or measurement errors. Attaining the ideal situation is nearly impossible, hence more sophisticated algorithms must be used to cope with such errors. By itself, the weighted least squares underestimates the actual deviations – typically attributing half or more of the real faults to other faults and measurement errors. Researchers have used "augmentation methods" such as fault logic to improve accuracy.

Lesson 2 – Defining an appropriate engine performance baseline to measure deviations is not nearly as simple as one might expect. Engine performance varies from one engine to another as well as versus time for any given engine. Consequently, it is critical to utilize baselines that characterize the specific engine in question. Figure A1 illustrates this point using six P&W FT4 turboshaft engines (ref. 1). Therefore, it is not good enough to use an average fleet value or a new-engine thermodynamic cycle model for a baseline. The variations amongst the operational samples is of the same magnitude as the actual deviations for a specific sample. Each specific engine ought to have its own baseline – and preferably a baseline from recent measured performance since performance values deteriorate with operational time due to wear and tear.

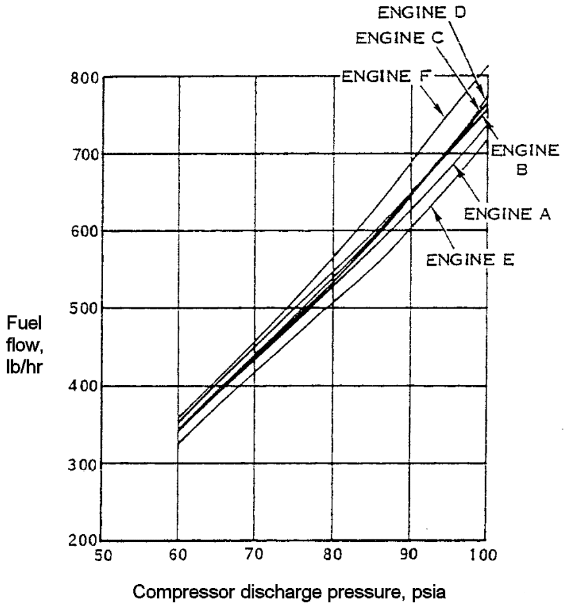


Figure A1 – Engine-to-engine deviation in fuel

Lesson 3 – Typical measurement errors are of the same magnitude as the sought deviations. Hence, it is important to include these deviations as well as the obvious engine health parameter deviations. Doing so, however, increases the set of unknowns to a greater number than the set of measurements – hence, we will have an underdetermined set of linear equations and therefore increased mathematical complexity. Kalman filters can be used to help estimate sensor errors. Nevertheless, sensor inaccuracies and simply the lack of sufficient sensors seriously hinders this method. Temperature and pressure sensors only yield local values whereas the flowfields are really 2- or 3-dimensional. It would be beneficial if these sensors gave a more accurate representation of the bulk average values at a given station. More sensors would improve the identification accuracy of defective components because multiple combinations of possible component faults yield similar system-level response deviations.

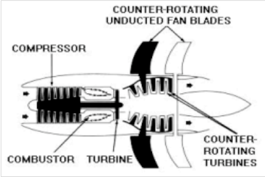
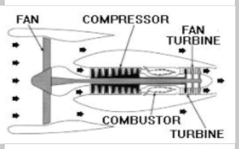
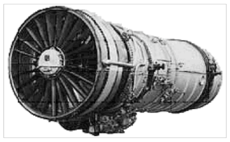
Lesson 4 – One approach to diminish the uncertainty of the fault estimation results is to invoke the weighted least squares analysis at several discrete points instead of just a single operating point. A previous investigation of this approach reduced the overall error index by about 50 percent using just 2 discrete points, and about 70 percent using 4 points. This does not significantly increase the mathematical complexity, it just utilizes more of the available sensor input data.

Lesson 5 – It is important to accurately represent the control system architecture, not just the basic flowpath components, in a diagnostic situation involving real engines. The reason is that the mathematical simulation must accurately represent the engine responses to component performance deviations and these can depend strongly on the control system behavior. For example, if the actual engine utilizes a variable compressor vane angle schedule with minimum/maximum angle limits, the engine response will differ from a hypothetical engine simulation that ignores such limits. Notwithstanding this point, developing and testing diagnostic methodologies can proceed without such control system fidelity. It is only needed during the application of the methodologies to actual engines.

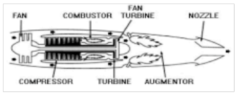
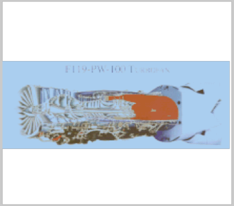
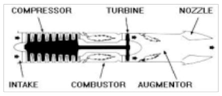
Lesson 6 – End users cannot be expected to interpret the raw results from the weighted least squares algorithm. A post-processing step must be included that interprets the output into an engine diagnosis that helps the user identify the appropriate corrective action.

Lesson 7 – The engine industry has developed rather mature and sophisticated diagnostic software tools to cope with the multitude of uncertainties involved. General Electric's *TEMPER* code (ref. 9) is an example. According to the literature, the airlines are already using these diagnostic tools to detect and isolate faults – not just in engine manufacturer's test cells, but also in flight service.

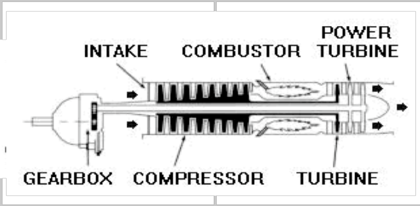
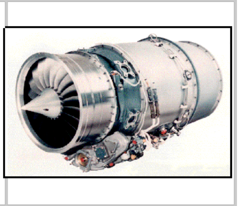
Appendix B – Aircraft classes with representative summary characteristics

Engine class	Ultra-High Bypass Engine	Large High Bypass Turbofan	Medium Bypass Turbofan	Low Bypass Turbofan
Typical engine cross-section or isometric	 COUNTER-ROTATING UNDUCTED FAN BLADES COMPRESSOR COMBUSTOR TURBINE COUNTER-ROTATING TURBINES			
Typical application	Commercial	Long-range transport	Regional transport	Commercial
Representative examples	GE-36 UDF	GE90-115B* PW4000 GP7270	PW2037 CFM56-7B27*	PW JT8D-219*
Physical properties				
Fan/compressor stages		1/3/10	1/3/9	
HPT/LPT stages		2/7	1/4	
Maximum diameter, inches		135	61	54
Length, inches		204	98.7	168.6
Dry weight, pounds		18,260	5216	4430
Performance				
Maximum SLSS power thrust, lbs.	24,000	115,300	27,300	21,000
Cruise TSFC (or BSFC)	~20% less	~0.55	0.64	0.82
Thermodynamic cycle				
Bypass ratio	~20	9+	5.1	1.74
Overall pressure ratio at max. power		42	32.8**	18.2-19.4
Fan pressure ratio	~1.15			1.91
T41, deg. F				~2050
Mixed or separate flow exhaust	separate	separate	separate	mixed
* Used to set properties listed below				
** at max. power cruise				

Appendix B – Aircraft classes with representative summary characteristics (continued)

Engine class	Low Bypass Afterburning Turbofan	Low Bypass Supercruise Thrust Vectoring Turbofan	Afterburning Turbojet
Typical engine cross-section or isometric	 FAN COMBUSTOR FAN TURBINE NOZZLE COMPRESSOR TURBINE AUGMENTOR		 COMPRESSOR TURBINE NOZZLE INTAKE COMBUSTOR AUGMENTOR
Typical application	Fighter	Fighter	Fighter
Representative examples	F100-PW-100 F110-GE-400*	PW F119* GE F120	J79-GE-10/17*
Physical properties			
Fan/compressor stages	3/9	3/6	17
HPT/LPT stages	1/2	1/1	3
Maximum diameter, inches	46.5		39.1
Length, inches	232.3		208.5
Dry weight, pounds	4400		3855
Performance			
Maximum SLSS power thrust, lbs.	27,000	~35,000	17,820
Cruise TSFC (or BSFC)	2		1.93
Thermodynamic cycle			
Bypass ratio	0.76		0
Overall pressure ratio at max. power	29.9		13.4
Fan pressure ratio			N.A.
T41, deg. F	~2500		
Mixed or separate flow exhaust	mixed	mixed	N.A.
* Used to set properties listed below			
** at max. power cruise			

Appendix B – Aircraft classes with representative summary characteristics (concluded)

Engine class	Turboprop (with gearbox)	Turboshaft (w/o gearbox)	Small Engine
Typical engine cross-section or isometric	 INTAKE COMBUSTOR POWER TURBINE GEARBOX COMPRESSOR TURBINE		
Typical application	Commuter	Helicopter	Cruise missile
Representative examples	PW2000 CT7-9D*	PT6 CT7-8*	F107-WR-101* FJ44
Physical properties			
Fan/compressor stages	6	6	
HPT/LPT stages	2/2	2/2	
Maximum diameter, inches	29	26	12
Length, inches	96	48.8	40
Dry weight, pounds	805	537	146
Performance			
Maximum SLSS power thrust, lbs.	1940 shp	2520 shp	600
Cruise TSFC (or BSFC)	0.455	0.450	
Thermodynamic cycle			
Bypass ratio	0	0	
Overall pressure ratio at max. power	18	21	
Fan pressure ratio	N.A.	N.A.	
T41, deg. F			
Mixed or separate flow exhaust	N.A.	N.A.	mixed
* Used to set properties listed below			
** at max. power cruise			

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2005	Technical Memorandum

4. TITLE AND SUBTITLE

Foreign Object Damage Identification in Turbine Engines

5. FUNDING NUMBERS

WBS–22–728–10–01
1L161102AF20

6. AUTHOR(S)

William Strack, Desheng Zhang, James Turso, William Pavlik, and Isaac Lopez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135 – 3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E – 15056

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001
and
U.S. Army Research Laboratory
Adelphi, Maryland 20783–1145

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM—2005-213588
ARL–MR–0611

11. SUPPLEMENTARY NOTES

William Strack, N&R Engineering and Management Services Corporation, 6659 Pearl Road, Suite 400, Parma Heights, Ohio 44130; Desheng Zhang, AYT Corporation, 2001 Aerospace Parkway, Brook Park, Ohio 44142; James Turso, QSS Group, Inc., 21000 Brookpark Road, Cleveland, Ohio 44135; William Pavlik, RS Information Systems, Inc., 21000 Brookpark Road, Cleveland, Ohio 44135; and Isaac Lopez, U.S. Army Research Laboratory, NASA Glenn Research Center. Responsible person, Isaac Lopez, organization code PRV, 216–433–5893.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 03 and 01

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report summarizes the collective work of a five-person team from different organizations examining the problem of detecting foreign object damage (FOD) events in turbofan engines from gas path thermodynamic and bearing accelerometer sensors, and determining the severity of damage to each component (diagnosis). Several detection and diagnostic approaches were investigated and a software tool (FODID) was developed to assist researchers detect/diagnose FOD events. These approaches include (1) fan efficiency deviation computed from upstream and downstream temperature/ pressure measurements, (2) gas path weighted least squares estimation of component health parameter deficiencies, (3) Kalman filter estimation of component health parameters, and (4) use of structural vibration signal processing to detect both large and small FOD events. The last three of these approaches require a significant amount of computation in conjunction with a physics-based analytic model of the underlying phenomenon—the NPSS thermodynamic cycle code for approaches 1 to 3 and the DyRoBeS reduced-order rotor dynamics code for approach 4. A potential application of the FODID software tool, in addition to its detection/diagnosis role, is using its sensitivity results to help identify the best types of sensors and their optimum locations within the gas path, and similarly for bearing accelerometers.

14. SUBJECT TERMS

FOD; Aviation safety; Fuzzy logic

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	