NASA/CR—2012-215242

Systematic Sensor Selection Strategy (S4) User Guide

T. Shane Sowers
QinetiQ North America, Cleveland, Ohio

February 2012

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to *help@sti.nasa.gov*

- Fax your question to the NASA STI Help Desk at 443–757–5803

- Telephone the NASA STI Help Desk at 443–757–5802

- Write to:
  NASA Center for AeroSpace Information (CASI)
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/CR—2012-215242

Systematic Sensor Selection Strategy (S4) User Guide

T. Shane Sowers
QinetiQ North America, Cleveland, Ohio

Prepared under Contract NNC06BA07B and NNC07E205T

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

February 2012

Acknowledgments

The author would like to thank Kevin Melcher and Donald Simon of the Controls and Dynamics Branch at the NASA Glenn Research Center for their significant patience, helpful guidance and persistent support in the development of this user guide and the associated S4 software package over a period of several years. Additionally, the author would like to thank George Kopasakis of the NASA Glenn Research Center, William Maul of QinetiQ North America, and Michael Santi of the Jacobs ESTS Group for their contributions to preliminary versions of the user guide and for their assistance in the development and advancement of the S4 methodology.

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

Level of Review: This material has been technically reviewed by expert reviewer(s).

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at http://www.sti.nasa.gov

Contents

1.0 Introduction 1
  1.1 Glossary 2
2.0 Key Features 3
  2.1 Knowledge Base 3
    2.1.1 Performance-Related Information 3
    2.1.2 System Simulation 3
  2.2 Iterative Down-Select Process 4
    2.2.1 System Diagnostic Model 4
    2.2.2 Sensor Suite Merit Algorithm 4
    2.2.3 Down-Select Algorithm 4
  2.3 Final Selection Process: Statistical Evaluation Algorithm 4
3.0 S4 Example Application 5
  3.1 Knowledge Acquisition 7
    3.1.1 Benefit Assessment 7
    3.1.2 Target System 7
    3.1.3 Sensor Measurements 8
    3.1.4 Faults 10
    3.1.5 Figures of Merit 11
    3.1.6 Analysis Environment 11
  3.2 Define Sensor Suite Merit Algorithm 11
  3.3 Define System Diagnostic Model 14
    3.3.1 Fault Detection 15
    3.3.2 Fault Isolation 15
    3.3.3 Diagnostic Performance Score 17
  3.4 Choose Down-Select Algorithm 18
  3.5 Compile Data Sets 20
    3.5.1 Inverse Model Reference Data 21
    3.5.2 Test Case Scenario Data 26
  3.6 Develop Software Modules 27
    3.6.1 KnowledgeBase.m 27
    3.6.2 MeritAlgorithm.m 27
    3.6.3 SystemDiagnostic.m 27
    3.6.4 GeneticAlgorithm.m 29
  3.7 Create and Conduct the Sensor Selection Process 29
    3.7.1 Create IterativeDownSelect.m 29
    3.7.2 Execute IterativeDownSelect.m 30
  3.8 Create and Perform Final Selection Process 31
    3.8.1 Knowledge Base 32
    3.8.1 Merit Algorithm 32
    3.8.2 System Diagnostic Model 33
    3.8.3 Down-Select Algorithm 33
    3.8.4 Software Application 33
    3.8.5 Execute Process 33
4.0 Overview of S4 Software Package 34
5.0 Concluding Remarks 35
References 36

Systematic Sensor Selection Strategy (S4) User Guide

T. Shane Sowers
QinetiQ North America
Cleveland, Ohio 44135

1.0 Introduction

This document presents a User Guide for the Systematic Sensor Selection Strategy (S4)—a methodical approach for identifying a suite of sensors that optimally meets engineering requirements for a given application. S4 was developed in an attempt to address the limitations of traditional aerospace sensor selection approaches that struggle to provide optimally effective sensor suite solutions for complex, interdependent system and subsystem requirements with sometimes difficult-to-identify and conflicting constraints.

In aerospace vehicle design, the proper selection of sensors is necessary to ensure that operational, maintenance performance, and more recently, system health assessment requirements are met. Traditional approaches to measurement and sensor selection for aerospace propulsion applications are generally heuristic and are typically directed toward measuring performance characteristics rather than health diagnostics. A typical sensor selection process begins with component teams submitting lists of desired measurements. These measurements are used to support engine development, model verification and/or detection of structural limit violations. As the system design matures, the component teams supply more detailed specifications such as measurement ranges, response requirements, and so forth. This information along with reliability requirements is used to determine the type and number of sensors needed at each system location. The compiled list is then separated into categories associated with measurement use, such as ground test, flight, etc. Measurements are assigned a priority, with the highest priority given to measurements required for system control. The list is often condensed as the design matures due to factors such as accessibility, cable routing, or reduced need. A maximum number of sensors is determined based on storage and transmission capability, cost, and other considerations which may include arbitrary limits. The component teams and chief engineer then negotiate until the final suite is selected. Consequently, sensor selection approaches such as this draw heavily on domain expertise and do not utilize a consistent quantitative method to assess the implications of measurement choices on diagnostic capability.

Research on methods for the systematic selection of sensors has been conducted in a variety of disciplines. Depending on the nature of the problem, these methods are designed to concentrate on various performance criteria such as observability, reliability, fault detectability, fault discrimination, and cost. Reference 1 presents a historical overview of several sensor selection approaches, comparing and contrasting their capabilities with those of S4.

The Systematic Sensor Selection Strategy is a NASA-developed optimization-based approach that can be used to optimally identify a set of sensors that support a specific design objective, like health diagnostics. The methodology is flexible and is meant to be tailored to the specific application needs. Originally developed by NASA to support liquid propellant rocket engine health management (Ref. 2), the S4 architecture offers the flexibility and versatility needed to support a broad range of applications, such as air-breathing engines, power systems, life support systems, ground support equipment, and general integrated vehicle health monitoring.

In developing S4, it was realized that a sensor selection process optimized for health assessment will specify a set of sensors to analyze system function and determine the type, source, and severity of system anomalies. The process should:

- Facilitate fulfillment of diagnostic system requirements for the target system such as fault detection and identification coverage for targeted fault conditions, missed detections, and false alarms
- Utilize a user specified diagnostic model by which sensor suites can be properly, effectively, and efficiently compared
- Access available and pertinent databases and simulations
- Support sensor system decisions impacting health management capability over the design, development, and service life cycle of the target system

Sensor selection processes can generally be categorized as employing either a general approach or a targeted approach. In a general approach, sensors are selected to provide complete fault detection and fault isolation for the system encompassing both known and unknown failures. While this is a desirable goal, it is generally unattainable since it is practically impossible to identify and optimize to every possible fault scenario. A targeted approach selects sensors to optimize detection and isolation capability for specific faults and/or fault sequences relative to a particular system health monitoring philosophy. S4 utilizes a targeted approach.

When applied to complex systems, sensor selection processes tend to produce a set of near-optimal solutions rather than a single comprehensive solution. A collection of satisfactory sensor suite solutions are often the outcome as opposing constraints may cause the optimal sensor suite sets to diversify. In addition, immature or novel systems introduce uncertainty or ambiguity into the process. Therefore, methodical variations in the sensor selection process should be employed to thoroughly explore the solution trade space in a logical fashion leading to a final sensor suite selection.

The purpose of this guide is to familiarize the user with the details of the S4 methodology and assist the user toward applying the S4 procedure. Section 2.0 describes the main features of S4 through an explanation of the general functionality of each S4 element. The process to construct a S4 application is outlined and an example application is presented in Section 3.0. Finally, an overview of the software for the S4 example application is provided in Section 4.0. Source code for software that implements the example application is being made available through the NASA Glenn Research Center Software Catalog (Ref. 3).

1.1 Glossary

Common wording may have subtle but important interpretation differences when applied to detailed descriptions. To help avoid confusion, a list of selected terms and definitions is given to indicate their particular meaning within this guide.

- **Fault**—An abnormal condition or defect that may lead to a system failure. In this guide, a fault is represented by a change in one or more health parameters within a system simulation that simulates off-nominal or fault operation.
- **Fault Detection**—The process of determining that a fault condition exists through analysis of the sensor measurements.
- **Fault Isolation**—The process of identifying the faulty component and characterizing the severity of the fault condition.
- **Fault Signature**—A set of unique measurement values resulting from a fault condition.
- **Target System**—The system to which the sensor selection process is being applied.
- **Health Parameter**—A parameter within a system simulation that represents the performance level of a component of the target system. An example would be pump efficiency.
- **Sensor/Measurement**—A sensed physical property specified by a location and a measurement type. An example would be a pump discharge pressure.
- **Nominal Value**—The normal or baseline value while the target system has no faults or degradations at a particular operating state.

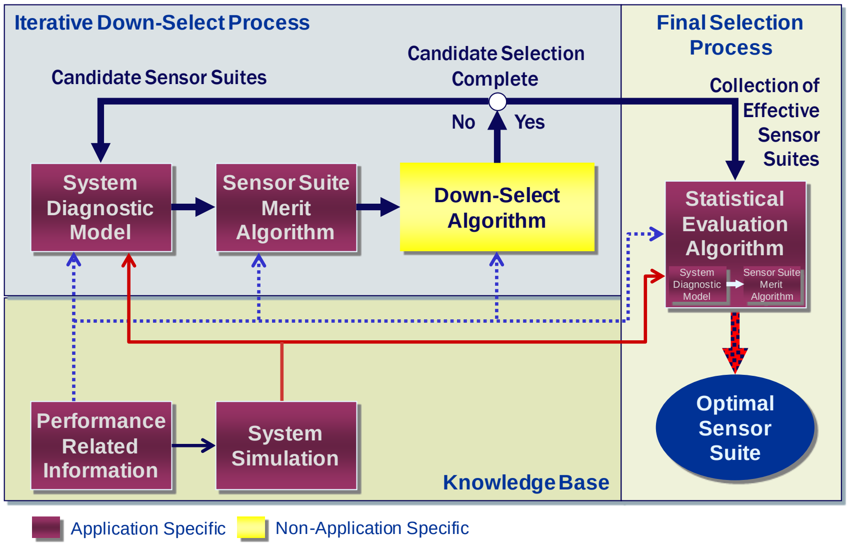

Iterative Down-Select Process
Candidate Sensor Suites
Candidate Selection Complete
No
Yes
System Diagnostic Model
Sensor Suite Merit Algorithm
Down-Select Algorithm
Final Selection Process
Collection of Effective Sensor Suites
Statistical Evaluation Algorithm
System Diagnostic Model
Sensor Suite Merit Algorithm
Optimal Sensor Suite
Performance Related Information
System Simulation
Knowledge Base
Application Specific
Non-Application Specific

Figure 1.—Systematic Sensor Selection Strategy (S4) Framework

2.0 Key Features

The Systematic Sensor Selection Strategy is best described as a general architecture structured to accommodate application-specific components and requirements. Each element is intended to be customized for the target system application, and their general relationships are depicted in Figure 1. A Knowledge Base, Iterative Down-Select Process, and Final Selection Process comprise S4. A functional description of each S4 component is given in the following subsections.

2.1 Knowledge Base

The Knowledge Base is intended to be a repository for information pertaining to the target system and the sensors being considered for use in diagnosis of that system. The repository may contain historical information such as test data, or if historical data is not available, as is often the case during early development and design of the target system, experience from similar systems will provide a basis from which to define and collect pertinent health assessment information. The knowledge base can be partitioned into two modules: the Performance-Related Information and the System Simulation.

2.1.1 Performance-Related Information

Information is obtained from domain experts, manufacturer reports, failure modes and effects analysis (FMEA) studies and hazard analyses. The information includes fault signatures, fault progressions, component/sensor reliabilities, and sensor characteristics (e.g., noise). This data is used as inputs to establish the System Diagnostic Model, and defines key optimization parameters for the Sensor Suite Merit Algorithm and Down-Select Algorithm. In addition, this information is used to define the required system simulations for health assessment system development.

2.1.2 System Simulation

The System Simulation provides input in the form of data sets for the Iterative Down-Select Process and Statistical Evaluation Algorithm. This module may be a collection of simulations of varying fidelities and applicable at various stages of the system operation. The System Simulation for S4 can be developed using a process model, which can be as simple as algebraic relations between the monitored variables and faults, or more complicated dynamic system models that incorporate performance parameters. If available and appropriate, actual test data or flight data could be provided from this module.

2.2 Iterative Down-Select Process

The Iterative Down-Select Process is an iterative procedure used to select a group of near-optimal sensor suites for health assessment. The process involves sequentially generating and evaluating a collection of candidate sensor suites, eventually converging towards an optimum sensor suite. The process can be repeated with systematic permutations to the initialization of the search process, until the results stabilize to a small set of near-optimal suites. The procedure includes a System Diagnostic Model, a Sensor Suite Merit Algorithm, and a Down-Select Algorithm.

2.2.1 System Diagnostic Model

The System Diagnostic Model represents the diagnostic procedure to be applied to the target system. The performance of the diagnostic model is measured for each candidate sensor suite at each relevant operational mode. The measures of performance are provided to the Sensor Suite Merit Algorithm. A key feature of this model is that it must be complete and capable of providing diagnostic analysis for all or any subset of the possible sensors. If the diagnostic system needs to be re-generated with each sensor suite, the sensor selection process will be very time consuming. In addition, the re-generation complicates comparison of diagnostic performance between sensor suites by incorporating diagnostic system development into system diagnostic performance. Developments of applied system diagnostic models to date have used system simulations and health-related information for fault characterization. During the Iterative Down-Select Process, the System Diagnostic Model receives sensor suite configuration information from the Down-Select Algorithm and outputs diagnostic performance information for those evaluated sensor suites to the Sensor Suite Merit Algorithm.

2.2.2 Sensor Suite Merit Algorithm

The Sensor Suite Merit Algorithm generates an evaluation score for each candidate sensor suite. It generates quantified metrics utilizing performance information from the System Diagnostic Model along with other pertinent characteristics of the measurement suite being evaluated. It receives sensor suite diagnostic performance information from the System Diagnostic Model, and it outputs scores to the Down-Select Algorithm. The Sensor Suite Merit Algorithm is normally in the form of an algorithmic function, where performance criteria of interest to the particular system can be combined.

2.2.3 Down-Select Algorithm

The Down-Select Algorithm is a search algorithm that utilizes an optimization technique employed to select sensor suites that will allow progression toward an optimal or near-optimal solution. The sensor selection optimization problem can be termed NP-Complete (Non-deterministic Polynomial time), which refers to the way the problem’s computation time scales as a function of the number of variables. Problems that are NP-Complete are the most difficult class of decision problems to solve. Any solution procedure that guarantees optimality cannot simultaneously guarantee computational time efficiency. Therefore, a trade-off between the level of optimality and the time to solution must be employed. Optimization techniques such as Genetic Algorithms (Ref. 4) produce acceptable results in a reasonable time. However, this particular module in the methodology could use any search algorithm suitable to the user. The Down-Select Algorithm receives system and sensor information from the Knowledge Base to establish any specific guidelines or constraints used in the sensor suite search. Merit value information is used to generate a collection of new candidate sensor suites that will generally progress toward improved performance.

2.3 Final Selection Process: Statistical Evaluation Algorithm

In the Final Selection Process, a collection of the best candidate sensor suites generated from the Iterative Down-Select Process is analyzed in more detail. The user inspects the output and determines how well the optimal sensor suite satisfies the performance criteria and whether some of the performance objectives within the Sensor Suite Merit Algorithm need to be modified. Often times these results yield trends that encourage further trade studies to enhance the selection process and overall final sensor suite selection.

The Statistical Evaluation Algorithm is intended as a final robustness test of each candidate sensor suite. When constructing the Iterative Down-Select Process, often there is a trade between simulation fidelity and diagnostic evaluation speed. Only a small number of fault scenarios can be utilized in the evaluation process or the processing time could be prohibitive. Therefore, a more rigorous evaluation is employed when a finite collection of "optimum" measurement sets has been determined. The Statistical Evaluation Algorithm replicates portions of the Iterative Down-Select Process with the addition of uncertainty effects such as sensor and system noise characteristics, and variations in fault dynamics. Large numbers of faults can be used to evaluate each sensor suite with the same System Diagnostic Model and Sensor Suite Merit Algorithm utilized in the Iterative Down-Select Process. The Statistical Evaluation Algorithm uses input from the Performance-Related Information and the System Simulation data to establish the high fidelity fault scenario data sets. Results from the Statistical Evaluation Algorithm enable a ranking or stratification of the near-optimal sensor suites for final selection.

3.0 S4 Example Application

This section describes an example application of the S4 methodology to an aircraft engine simulation. The intent of this demonstration is to permit a user to initiate and solve sensor selection problems that are specific to their area of interest. As mentioned previously, the S4 methodology is not intended to be rigidly structured. Although detailed steps in formulating the problem and desired objectives are usually application specific by design, there is a great deal of flexibility in establishing an S4 architecture that is particularly suited to the user's needs. Domain expertise and knowledge base information have a large influence on the selection and development of the particulars of the S4 components.

While each application is unique, there is a common approach to applying the S4 methodology with the objective of health assessment improvement. General procedural steps required to construct the sensor selection framework are shown in Figure 2. The steps of this general approach are further described and demonstrated through the presented example application given below.

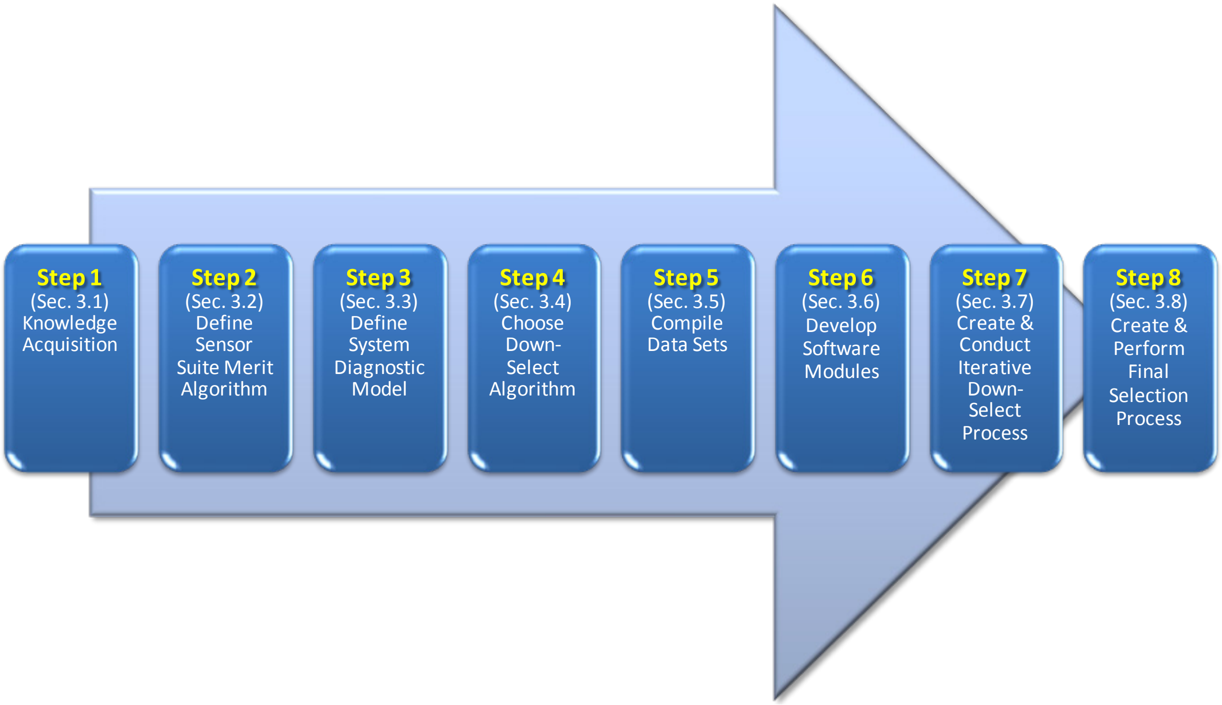


Figure 2.—Procedure for Applying S4 to a Specific System

1. Knowledge Acquisition
   - Identify all potential sensor locations; include sensors that will be available regardless of sensor selection process (e.g., control sensors, actuator position sensors).
   - Define system operational phases and control modes.
   - Identify faults of interest and, if possible, indicate criticality, probability of occurrence, and fault propagation information. If absolute values of fault criticality and probability of occurrence are not available, the relative values of the fault characteristics within the scenarios of interest are still useful.
   - Obtain available domain information, including relevant component design, simulations, and test data.

2. Define Sensor Suite Merit Algorithm.
   - Define metrics of diagnostic performance.
   - Incorporate cost and user-specified design constraints.

3. Define System Diagnostic Model.
   - Configure to at least one operational phase, but being applicable to multiple operational phases will maximize diagnostic coverage and permit sensor selection evaluation across phases.
   - Must be capable of providing fault detection and fault isolation of all failure scenarios of interest with the complete set of potential sensor locations.
   - Must be functional for any subset of the potential sensors.

4. Choose Down-Select Algorithm
   - If feasible, a Brute Force or Exhaustive Search technique is preferred since it will examine every possible sensor suite to determine the true optimal solution.
   - In cases where complete evaluation of all sensor suite combinations is not possible, define the optimization goals and select an optimization algorithm.

5. Compile Data Sets
   - Include the entire list of faults that the diagnostic system is designed to monitor.
   - Obtain any diagnostic information needed by the System Diagnostic Model.

6. Develop Software Modules
   - Select a computational environment consistent with user preferences.
   - Develop software modules.

7. Create and Conduct Iterative Down-Select Process
   - Perform iterative sensor selection process until the optimization goal(s) are realized.
   - While essentially an automated process, expert oversight and possible intervention may be required to ensure realistic performance from the process.

8. Create and Perform Final Selection Process
   - Examine diagnostic performance of selected sensor sets to further stratify the sets and enable final selection.
   - May be performed with higher fidelity data sets and/or data scenarios outside of the constraints used during the initial iterative optimization process.

The implementation of these common steps to the S4 example application is enumerated in the following sections.

3.1 Knowledge Acquisition

Preliminary steps of the sensor selection process involve collecting data on the target system. A proper S4 framework is tailored for the specific application at hand. Information is gathered to ensure the setup and consequential analysis is correct, pertinent, and worthwhile to execute. Therefore, at the onset of the analysis it is helpful to ask some basic questions:

- *What is the system that S4 is being applied to? Can it benefit from a sensor selection procedure?*
- *What sensors are available? What are their characteristics?*
- *What are the sensors monitoring? What should they detect?*
- *How is the information being used? What type of event is being monitored? Is it only fault diagnostics, or something more?*
- *What is the system doing? What does it need to do? What is normal and what is abnormal?*
- *What is the source of data for conducting the sensor selection process? Is actual data available or will a computer simulation be used?*
- *What is the objective of the sensor selection process? What is being optimized?*
- *What assumptions can be made to simplify the S4 procedure?*

3.1.1 Benefit Assessment

An important first step is to determine if the system will benefit from a sensor selection procedure. Before time and effort is invested, it is a good idea to confirm that performing a selection analysis is prudent. In the first benefit assessment, the complexity of the system is evaluated. Sensor selection techniques are generally not applied to simple systems as their solutions tend to be trivial. As a result, there is a focus toward complex systems where conflicting objectives such as performance, cost, and weight need to be satisfied simultaneously along with a need to reduce or minimize the sensor set. In another benefit assessment, the feasibility of altering the sensor suite of the system is evaluated. Measurement sets of systems in production tend to be difficult to adjust. As a result, selection procedures are usually applied to systems in development during which design changes are more feasible and economical. This is also when knowledge of the system is relatively immature causing more dependence on prior experience with similar systems or on engineering judgment of domain experts. A further benefit determination is based on how well the consumers of the sensor data are characterized. A comprehensive understanding of data use is needed to properly select and construct the components of the process. Once its benefit is recognized, the sensor selection procedure is designed and implemented.

3.1.2 Target System

The target system selected for the S4 example application is the Commercial Modular Aeropropulsion System Simulation or C-MAPSS) (Ref. 5). Principle factors in this selection were user availability, freedom to define the problem, and sufficient technical complexity. C-MAPSS is a non-linear aerothermodynamic aircraft engine model developed by the NASA Glenn Research Center for control and diagnostic algorithm research. As such, it incorporates the capability to simulate realistic fault conditions, mainly in terms of rotating component efficiencies and flow capacity shifts from nominal conditions. The model is coded in Matlab/Simulink (The Mathworks, Inc.) and provides a non-proprietary model of a large commercial high-bypass, twin-spool, turbofan engine in the 90,000 $\mathrm{lb_f}$ thrust class. Its operating envelope covers sea-level to 40,000 ft, Mach 0 to 0.90, and sea-level ambient temperatures from –60 to 103 °F.

With C-MAPSS as the target system, assumptions and other restrictions are made about the extent of the S4 procedure to simplify the example application. These assumptions and restrictions include:

- Realistic faults are used to evaluate the candidate sensor suites.
- Only steady-state conditions are considered so that sensor response timing can be ignored.

- There are no sensor or actuator faults. All faults occur within the engine components, and the sensor measurements and actuator positions are assumed to be unbiased.
- All faults are of equal importance with no preference based on fault severity, probability of occurrence or any other factor.
- All of the sensors have similar cost, reliability, weight, etc.
- Data from a single flight operating point is utilized with cruise being selected for its significance in a typical commercial flight profile and because it is consistent with the steady-state assumption of this example.
- The set of control sensors are assumed to always be present. Additional candidate sensors are being selected to supplement the control sensor set.
- The sole objective of the selection process is to improve engine health assessment capabilities for the selected faults. Other potential operational objectives are not considered.
- Sensor noise is assumed to be zero-mean, normally distributed white noise.

With C-MAPSS as the target system, performance data is generated from a system simulation rather than from ground test or flight performance data. This arrangement is typical as actual data are rarely complete enough to evaluate all of the candidate measurements across all relevant operating scenarios. Within C-MAPSS, component faults are simulated by adjusting one or more engine component health parameters. C-MAPSS’s engine component health parameters include flow capacity, efficiency, and pressure ratio modifiers. Complex faults may require correlations to multiple health parameters to more accurately represent the fault condition. Additionally, candidate sensor measurements are related to system simulation output variables. Sensor type, such as pressure, temperature, shaft speed, etc., and location are used to determine the most appropriate output variable. For this demonstration, sensors are simply chosen from the C-MAPSS output variable list. In other applications, a sensor-to-variable correlation may not exist. When this happens, either the system simulation is modified to make the desired output variable available or the sensor is excluded from consideration.

The sensor selection procedure will employ the C-MAPSS simulation at a single operating state. From a typical flight profile, the cruise operating condition was selected with a set of properties assumed. The assumed cruise flight condition is an altitude of 35,000 ft, Mach 0.84, throttle setting of 65 percent under standard atmospheric conditions.

3.1.3 Sensor Measurements

In this example application, there are two classifications of sensor measurements. There are control sensors which are always present, and candidate sensors which will augment the control sensor set. This arrangement is representative of a situation where an existing set of sensors is being augmented with additional measurements. Control sensors typically include a subset of sensors that measure ambient conditions. In this example application, the ambient condition sensors are not included because they measure conditions outside the propulsion system and only vary due to changes in the flight environment. They are insensitive to the fault conditions within the engine and provide no information to improve health assessment. The approximate locations of the control and candidate sensors are shown in Figure 3. Table 1 summarizes the individual sensors assessed in this study along with their corresponding sensor noise levels, nominal values, and assigned sensor classification.

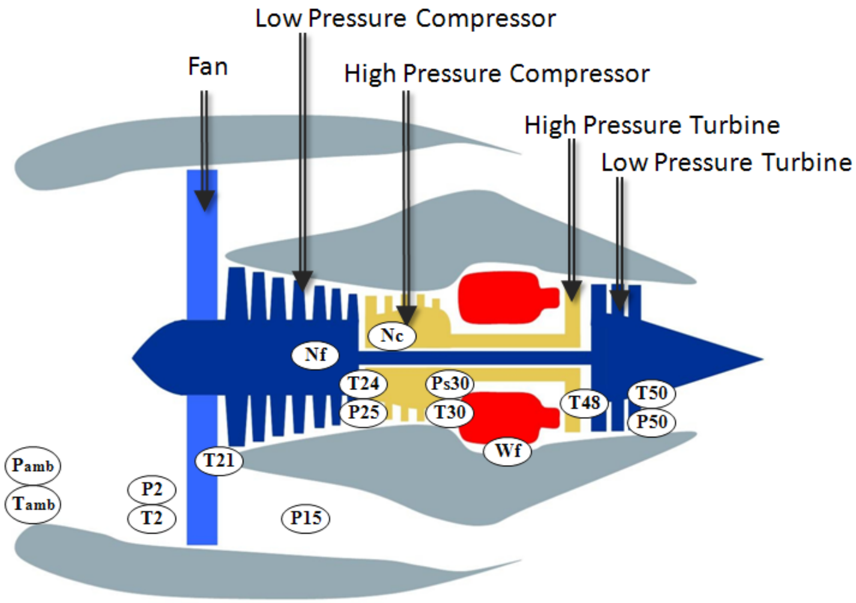


Figure 3.—C-MAPSS Diagram with General Sensor Location and Classification

TABLE 1.—SENSORS USED IN EXAMPLE APPLICATION

Variable	Description	Classification	Baseline	Noise standard deviation
T24	Low pressure compressor exit temperature	Control	526.1 R	2.6307
T30	High pressure compressor exit temperature	Control	1,229.3 R	9.2199
Ps30	High pressure compressor exit static pressure	Control	129.9 psia	0.6494
Wf	Fuel flow rate	Control	1.3 pps	0.0130
Nf	Low pressure shaft speed	Control	1,936.2 rpm	4.8405
Nc	High pressure shaft speed	Control	7,920.8 rpm	19.8020
T48	Low pressure turbine inlet temperature	Control	1,503.3 R	11.2748
P15	Bypass discharge static pressure	Candidate	7.1 psia	0.0355
T21	Fan exit temperature	Candidate	489.5 R	3.6711
P25	High pressure compressor inlet pressure	Candidate	8.8 psia	0.0439
T50	Low pressure turbine exit temperature	Candidate	986.9 psia	7.4021
P50	Low pressure turbine exit pressure	Candidate	4.5 psia	0.0227

Measurement data from the System Simulation are conditioned before use in the S4 procedure. The conditioning process provides numerical stability for data that contain a variety of measurement types and engineering units. It also assists the user by transforming the data into a form that is easier to comprehend from a health monitoring perspective. Measurement values are conditioned according to the following relation,

$$y_i = \frac{(y_{fault,i} - y_{nominal,i})}{\sigma_i} \tag{1}$$

where

y_i	conditioned value for sensor measurement y_i
$y_{fault,i}$	observed value for sensor measurement y_i during a fault
$y_{nominal,i}$	nominal value of sensor measurement y_i
σ_i	standard deviation of sensor measurement noise for y_i

Typically, a Gaussian distribution is assumed for measurement noise about a mean value, and σ_i is set to the standard deviation value of the measurement signal. For each sensor measurement, the normalization is simply the change of the measurement from its nominal value to the current fault level, divided by the measurement noise. The conditioned value, therefore, represents the number of uncertainty intervals by which the signal varies for a given fault scenario. The noise estimates shown in Table 1 are approximations based on engineering judgment. They are expressed as percentages of the nominal operating value of the sensor measurements at a particular flight condition.

3.1.4 Faults

A critical part of the selection process is the definition of faults used in the evaluation of the candidate sensors. Each fault will have a fault signature in the sensor measurement set that is an input to the System Diagnostic Model. Because there may be a very large number of faults for any given target system, care should be given in compiling a list of pertinent faults. The faults should be focused and appropriate. Improper fault selection can misdirect the optimization process and adversely affect selection results. Operational objectives, environmental considerations, and other system requirements define the relative importance of such factors as fault severity, probability of occurrence, timing, and mitigating response availability. These factors are investigated through analysis to determine the faults of interest.

The fault modeling capability of the System Simulation can influence the compiled listing of the faults of interest. If a fault cannot be modeled, the System Simulation is enhanced or the fault is removed from consideration. Accurate fault signatures are important for effective sensor selection.

To produce measurement data for the faults of interest, the fault simulation capability of C-MAPSS is examined. C-MAPSS has an input array of thirteen user adjustable health parameters. For each of the five rotating engine components, there is a flow capacity and efficiency. Additionally, for the fan, low pressure compressor, and high pressure compressor, there is a pressure ratio modifier. Table 2 lists all of the health parameters used in the S4 example application. As can be noted in the table, the low pressure compressor and the pressure ratio modifiers for the fan and the high pressure compressor were not included. Generation of the fault data is described in Section 3.5.

TABLE 2.—C-MAPSS HEALTH PARAMETERS
USED FOR FAULT SIMULATION IN
THE S4 EXAMPLE APPLICATION

Engine component
Fan efficiency
Fan flow capacity
High pressure compressor efficiency
High pressure compressor flow capacity
High pressure turbine efficiency
High pressure turbine flow capacity
Low pressure turbine efficiency
Low pressure turbine flow capacity

As was done for the sensor measurement data, fault data are conditioned before they are input into the S4 procedure. This improves numerical stability and simplifies user interpretation. While C-MAPSS already has fault data in a normalized format, this may not be the case with other system simulations. In those instances, faults should be conditioned using a relation such as the following:

$$x_j = \frac{(x_{fault,j} - x_{nominal,j})}{x_{nominal,j}} \tag{2}$$

where

x_j conditioned value of fault x_j

$x_{fault,j}$ fault value of fault x_j
$x_{nominal,j}$ nominal value of fault x_j

The conditioned fault value represents the fractional shift from the nominal baseline value for a given fault. The values of $x_{fault,j}$ and $x_{nominal,j}$ in Eq. (2) can be directly obtained from the system simulation. If actual hardware is used, fault values usually cannot be directly measured and would need to be estimated using the sensor values.

Not all fault data can be conditioned in the same manner. For instance because leak flows have a baseline value of zero, they require special treatment to avoid division by zero. Leak values may be transformed to the value of the baseline flow through the duct where the leak occurs. For example, given a leak of 2 lb_m/sec in a duct with a baseline flow of 50 lb_m/sec, the fault is conditioned in two steps as follows.

$$x_{duct} = \frac{((50-2)-50)}{50} = -0.04 \quad \Rightarrow \quad \Delta x_{\text{leak}} = -\Delta \text{x}_{\text{duct}} = 0.04$$

Since the conditioned value of –0.04 represents the amount subtracted from the duct flow, but added to the leak flow; the final conditioned value of the leak fault is 0.04.

3.1.5 Figures of Merit

During the knowledge acquisition phase, it is important to determine what characteristics are essential in the final sensor suite. In the S4 example application, health diagnostics in terms of fault detection and isolation are the central items of interest. In a more realistic application, the merit algorithm could be expanded to include other operational considerations such as life cycle cost, reliability, weight, power requirements, communication bandwidth or ease of installation. This information would be used to design the Sensor Suite Merit Algorithm and influence the implementation of the System Diagnostic Model.

3.1.6 Analysis Environment

A final S4 framework design consideration is the analysis environment in which the sensor selection process is performed. Because S4 is flexible, the user has a great deal of freedom in this decision. S4 has been applied using software developed in C/C++, FORTRAN and MATLAB. In each of these applications, the chosen programming environment had advantages and disadvantages. Compiled languages such as C/C++ and FORTRAN offer rapid execution speeds and enable intricate software architectures if required. Conversely, interpreted languages are generally easier to employ when the setup of the selection process is dynamic. Some interpreted programming environments such as MATLAB have mechanisms that facilitate elaborate software architectures. They can incorporate compiled libraries which improve execution speeds and allow incorporation of third party software packages. They may also have built-in functions that support analysis of S4 results. When all of this is considered, the user has a great deal of latitude in selecting the computing environment. Therefore, user preferences along with application considerations will direct this decision. For the S4 example application, MATLAB is selected as the software environment for ease of programming and acceptable execution speed.

3.2 Define Sensor Suite Merit Algorithm

With the initial design decisions made for the S4 process setup, the first module to be constructed is normally the Sensor Suite Merit Algorithm. Of the elements in the Iterative Down-Select Process, the Sensor Suite Merit Algorithm is the most influential. It quantifies the degree to which a sensor suite satisfies the desired characteristics by combining a set of potentially competing metrics into a single merit value. The assignment of merit values will guide the progression of sensor suites output from the Down-Select Algorithm.

When designing the Sensor Suite Merit Algorithm, there are common interface requirements that will be necessary for practically every application. On input, the current collection of candidate sensor suites is needed. This is to identify the sensors included so that relevant Knowledge Base information can be extracted. These data are combined with diagnostic performance information and other relevant properties to compute a sensor suite's merit value. The user may incorporate a wide range of metrics that are limited only by the imagination and programming ability of the user. On output, a single value is computed that represents the worth or goodness of the sensor suite to the application at hand.

NOTE: The Sensor Suite Merit Algorithm presented here is provided for demonstration purposes and is solely intended to support the S4 example application. Alternate approaches may more accurately reflect diagnostic performance. Modifications to the method are encouraged to accommodate application-specific requirements.

The example merit algorithm is focused solely on system health assessment. Its logic flow, shown in Figure 4, is summarized as follows. A sensor suite is selected from a group of candidate sensor suites and is evaluated by the System Diagnostic Model. The System Diagnostic Model will calculate a diagnostic performance score which is then integrated with other factors represented by the penalty term and utility measure to compute the sensor suite merit value. The process repeats until each sensor suite in the collection of candidate sensor suites is evaluated.

To begin the formulation process, a merit function for sensor suite *k* is defined that combines the metrics of interest into a single algebraic equation. For the S4 example application, the diagnostic performance term is integrated with factors that characterize the advantages and disadvantages of each candidate sensor suite. This is done by generating a diagnostic performance score, D_k, that is multiplied by a penalty term, P_k, and a utility measure, U_k, to produce a merit value, $Merit_k$, as shown in Eq. (3). With this merit function formulation, higher merit values reflect better sensor suites. The merit value calculation is given by,

$$Merit_k = D_k \cdot P_k \cdot U_k \tag{3}$$

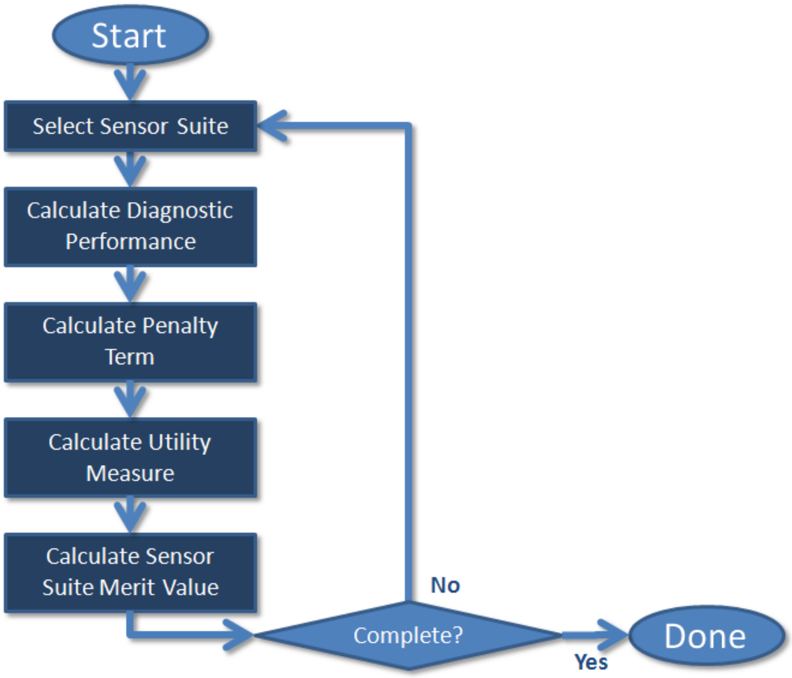

Start
Select Sensor Suite
Calculate Diagnostic Performance
Calculate Penalty Term
Calculate Utility Measure
Calculate Sensor Suite Merit Value
Complete?
No
Yes
Done

Figure 4.—Sensor Suite Merit Algorithm Logic Flow Diagram

As shown in Eq. (4), the utility measure, U_k, is the average of the utility measures, $u_{1\ldots m}$, for each of the m sensors within suite k.

$$U_k = \frac{\sum_{i=1}^{m} u_i}{m} \tag{4}$$

Here, each u_i quantifies the good and bad characteristics of sensor i, while U_k broadly quantifies the characteristics of sensor suite k. The characteristics and quantification methods used to determine each u_i are selected by the user. They may include cost, weight, power requirements, ease of installation, etc. In this formulation, a u_i with value less than one signifies a sensor with a detrimental influence, and a u_i with value greater than one indicates a sensor with a beneficial effect. In the S4 example application, every sensor has a uniform and neutral setting of 1.0.

The next factor in the merit function is the penalty term, Eq. (5), which enforces a preferred sensor suite characteristic. In this implementation, the penalty term includes a normalizing term, a discrepancy term, and a user specified weighting term. The normalizing term is embodied by the “1” in the denominator and is intended to prevent divide-by-zero errors in the function implementation. The discrepancy term measures the sensor suite’s deviation from the desired characteristic. In this case, it is the absolute value of the difference between the actual number of sensors in the sensor suite and the desired number of sensors in the final solution. A weighting term is used to regulate the influence of the penalty term. The penalty term is computed by,

$$P_k = \frac{1}{1+\left(W_{penalty} \cdot \left|N_{desired} - N_{k,actual}\right|\right)} \tag{5}$$

where

$W_{penalty}$ penalty weighting term
$N_{desired}$ desired number of sensors in solution sensor suite
$N_{k,actual}$ actual number of sensors in candidate sensor suite k

Consideration should be given in selecting the value for the penalty weighting term, $W_{penalty}$. When the penalty weighting term is small, there is more exploration of the solution space as it is easier for sensor suites of varying size to be formed. However, the resulting sensor suites may contain an unwanted number of sensors. When the penalty term is large, the resulting sensor suites will tend to have the desired number of sensors, but might not be as optimum. Experimentation is often needed to find the proper penalty weighting term that will balance these potential conflicts. For the S4 example application, a penalty weighting term value of 0.1 is used.

With this merit function formulation, a penalty weighting term value of “1” denotes no penalty. This occurs when $W_{penalty}$ is set to zero by the user or when the number of actual sensors is the same as the number of desired sensors specified by the user. Therefore, performing a selection study with a $W_{penalty}$ of zero with this formulation will disable the penalty term and determine the optimal solution regardless of the desired number of sensors.

The final factor in Eq. (3) is the diagnostic performance score, D_k. The calculation of this factor will be described in detail in Section 3.3.

The merit algorithm process is illustrated by a simple demonstration of the calculations for a single sensor set. Assume that sensor suite k has three sensors while the desired number is two, and that the weight of the penalty term is 0.1. Also assume that suite k is evaluated by the System Diagnostic Algorithm with a resulting diagnostic performance score of 10.0. Further, assign each sensor a utility measure of 1. With these assumed inputs, the merit function, Eq. (3), may be evaluated as follows:

$$
\begin{aligned}
Merit_k &= \left(\frac{\sum_{i=1}^{m} u_i}{m}\right) \cdot \left(\frac{1}{1+\left(W_{penalty}\left|N_{desired}-N_{k,actual}\right|\right)}\right) \cdot D_k \qquad (6)\\
&= \left(\frac{1+1+1}{3}\right) \cdot \left(\frac{1}{1+(0.1\times|2-3|)}\right) \cdot (10)\\
&= (1) \cdot (0.9\overline{09}) \cdot (10)\\
&= 9.\overline{09}
\end{aligned}
$$

The merit value of $9.\overline{09}$ is used by the Down-Select Algorithm to characterize the worth of this sensor suite.

3.3 Define System Diagnostic Model

The next S4 component to be constructed is the System Diagnostic Model. Though characterized as an independent element throughout this guide, the System Diagnostic Model is actually a vital sub-component of the Sensor Suite Merit Algorithm. As such, it tends to be implemented within the Sensor Suite Merit Algorithm. It is shown separately within this document to highlight its importance in the over-all Iterative Down-Selection Process. With its influence on the Sensor Suite Merit Algorithm, the System Diagnostic Module should be identical to, or at the very least representative of, the actual operational diagnostic system.

Even though this module is usually highly customized, there are common interface inputs and outputs that are appropriate for nearly any diagnostic module. To begin, there is a specification of the candidate sensor suite under evaluation so the diagnostic method has an indication as to which measurements are available and which are not. Another prerequisite is for a series of observation data for the faults of interest. Given this information the System Diagnostic Model will attempt to diagnose the occurrence of any faults. Finally, information about the true system condition is to be provided so that a grade or score can be given on how well the diagnostic system performed. On output, the System Diagnostic Module returns the computed diagnostic performance score, D_k. The diagnostic performance score is a scalar value that represents the quality or rating of the given candidate sensor suite to the diagnostic problem. It is subsequently provided as an input to the Sensor Suite Merit Algorithm as shown in Eq. (3).

NOTE: The System Diagnostic Model presented here is provided for demonstration purposes and is solely intended to support the S4 example application. Alternate approaches may more accurately reflect diagnostic performance. Modifications to the method are encouraged to accommodate application-specific requirements.

For the S4 example application, a general purpose System Diagnostic Model is employed that consists of a fault detection element and a fault isolation element. Fault detection is executed on each set of sensor observations to determine if the observed condition deviates significantly enough to indicate a system anomaly. When a system anomaly is detected, the fault isolation component is activated and attempts to identify the faulty system component. The sensor observations used in the diagnostic evaluation are noise-free and contain a single fault magnitude for each fault type. The general system diagnostic process is illustrated in Figure 5.

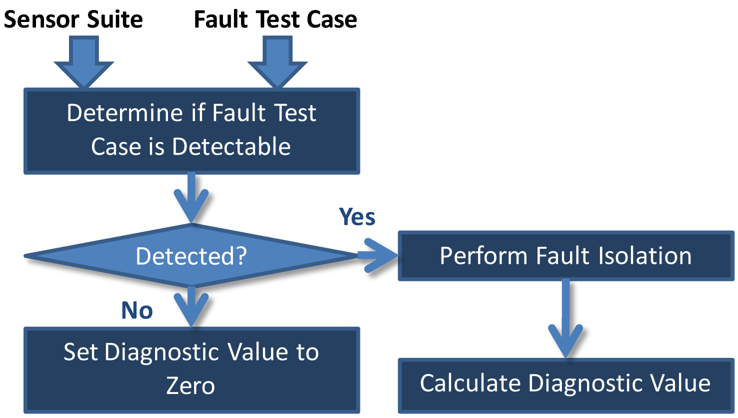

Sensor Suite
Fault Test Case
Determine if Fault Test Case is Detectable
Detected?
Yes
No
Perform Fault Isolation
Set Diagnostic Value to Zero
Calculate Diagnostic Value

Figure 5.—System Diagnostic Model Logic Flow Diagram

3.3.1 Fault Detection

In fault detection, the observed sensor values are examined for the indication of a fault. In this example, two types of fault detection checks are used on the conditioned sensor data. When a threshold is violated in either of these two checks, fault detection is declared. With the first check, each signal in the sensor suite is examined to determine if the absolute value of the sensor signal’s magnitude surpasses a minimum detection threshold of three times the sensor signal’s standard deviation value. In the second check, the combined root-mean-squared (RMS) value of the entire candidate sensor suite is compared to a detection threshold of 0.5. The sensor suite’s RMS value for each observation is computed by,

$$RMS = \sqrt{\frac{1}{m}\sum_{i=1}^{m} y_i^2} \tag{7}$$

where

RMS root-mean-square value of sensor suite
m number of measurements in the sensor suite
y_i conditioned value of sensor i

The threshold limits for the fault detection checks described above were established arbitrarily. In a normal application, the threshold limits are set using design criteria. For instance, threshold limits could be set to prevent specific faults from becoming catastrophic. Alternately, detection thresholds could be set according to missed detection (false positive) and/or false alarm (false negative) goals.

Results of the fault detection test dictate the next action in the diagnostic process. If the fault is not detected, the diagnostic performance score for the fault detection test is set to zero because the measurement set did not sense the fault. If the fault is detected, a fault isolation algorithm is utilized to identify the source of the system anomaly. The fault isolation algorithm utilized in the S4 example application is described in the following section.

3.3.2 Fault Isolation

In the S4 example application, an Inverse Model is employed for fault isolation. An Inverse Model is a parameter optimization technique that performs model inversion to determine the fault type and magnitude that best recovers, or matches, the observed measurement values. In the methodology utilized, each known fault or fault hypothesis is individually evaluated. The fault hypothesis that most closely matches the observed fault signature is assumed to be the correct fault condition. The ability of the Inverse Model to accurately discriminate between correct and incorrect fault hypothesis reflects the suitability of the sensor suite under evaluation for fault isolation.

The Inverse Model used in this study assumes measurement parameters are piecewise linear functions of the fault conditions. For each fault j, an ordered sequence of interpolation points $x_j^1 < x_j^2 < \cdots < x_j^s$ is selected, in part, to ensure the validity of the linearity assumption in each subinterval $\left[x_j^q, x_j^{(q+1)}\right]$. Consequently, interpolation points may not be evenly spaced as regions with increased non-linearity may contain more interpolation points while more linear regions may contain less.

The value of sensor *i* at each interpolation point *q* for fault *j* is provided by the engine simulation model as indicated by the relation below

$$y_{ij}^q = f_i\left(x_j^q\right) \;\; for \;\; i = 1,2, \dots m; \;\; j = 1,2, \dots, n; \;\; q = 1,2, \dots, s; \tag{8}$$

where

n number of faults
m number of measurements
s number of interpolation points for each fault *j*
x_j^q value of fault x_j at interpolation point q
y_{ij}^q predicted value of measurement y_i when $x_j = x_j^q$ and all other faults remain at their nominal values (i.e., equal to 0)
f_i effective functional relation for sensor *i*

To implement the piecewise linear approximation, fault *j* influences on measurement changes over fault *j* subintervals $\left[x_j^q, x_j^{(q+1)}\right]$ are represented by constant influence coefficients A_{ij}^q. For the solution of a single fault, the dimensionless change in any sensor *i* from its nominal value is approximated by the linear relation:

$$\hat{y}_i = f_i(\hat{x}_j) = A_{ij}^q\left(\hat{x}_j - x_j^q\right) + y_{ij}^q \tag{9}$$

where

$\hat{x}_j$ estimated value of fault *j*
$\hat{y}_i$ estimated value of sensor *i* due to estimate fault $\widehat{x_j}$

with the appropriate fault parameter interpolation points x_j^q such that

$$x_j = \in \left[x_j^q, x_j^{(q+1)}\right] \quad \text{for fault } j$$

and with the appropriate subinterval influences defined by the relation

$$A_{ij}^q = \frac{y_{ij}^{(q+1)} - y_{ij}^q}{x_j^{(q+1)} - x_j^q}$$

for fault *j* and for each measurement $i = 1, 2, \dots, m$.

A modified Levenberg-Marquardt nonlinear optimization algorithm (Refs. 6 and 7) is used to solve the nonlinear optimization problem shown in Eq. (10) as a means of predicting the fault value $\hat{x}_j$ for each set of observations. Stated differently, for a given hypothesized fault, j, the algorithm minimizes the Euclidian norm of a vector of measurement residuals between the observed sensor values and the estimated sensor values, from Eq. (9), by adjusting the estimated fault, $\hat{x}_j$,

$$min\left(\sqrt{\sum_{i=1}^{m}\tilde{y}_i^2}\right) \text{ by selection of } \hat{x}_j\text{, where } \tilde{y}_i = y_i - \hat{y}_i \tag{10}$$

Equation (10) can be termed the cost function or the objective function of the optimization process. The Levenberg-Marquardt algorithm will attempt to find the $\hat{x}_j$ that minimizes this equation. In the event that there is more than one $\hat{x}_j$ solution which minimizes Eq. (10), the minimum solution (i.e., the solution nearest the starting point) will be selected.

The Inverse Model diagnostic procedure is briefly summarized as follows. Interpolation points are selected for each targeted fault. At each interpolation point, information that relates fault values to sensor values is obtained and is referred to as Inverse Model reference data. With this data and a fault magnitude approximation, estimates of associated sensor values are calculated using Eq. (9). Measurement residuals are then processed according to Eq. (10) which defines the Levenberg-Marquardt objective function. The Levenberg-Marquardt algorithm uses the Euclidian norm of the measurement residual vector to perturb the estimate of the fault magnitude until the norm in Eq. (10) falls below a user-defined tolerance limit. A number of iterations threshold is also used by the algorithm to stop the iteration process when a non-converging solution occurs—presumably when the fault condition being recovered is not the true cause. When the optimization process terminates, the fault solution is determined.

In this example application, the Inverse Model will be applied to recover each potential fault type, one fault at a time. The hypothesized fault type that best matches the observed fault signature will be classified as the fault type. Depending on the problem setup, it may be possible to solve for all faults simultaneously.

3.3.3 Diagnostic Performance Score

A diagnostic performance score is assigned by the System Diagnostic Model where each candidate sensor suite is evaluated against a prescribed set of faults. The diagnostic performance score is a summation of fault criticality weighting factors multiplied by the fault diagnostic performance metric for all of the test cases. The relation is given by,

$$D_k = \sum_{j=1}^{n} dj = \sum_{j=1}^{n} C_j Z_j \tag{11}$$

where

D_k diagnostic performance score for candidate sensor suite *k*
d_j diagnostic performance score for fault test case *j*
C_j criticality factor of fault test case *j*
Z_j fault diagnostic performance metric for fault test case *j*

The diagnostic performance score of Eq. (11) is comprised of two parameters—a criticality factor and a fault diagnostic performance metric. The criticality factor, C_j, serves as a weighting term that allows the user to prioritize each fault test case by criticality and/or probability of occurrence. In the S4 example application, this weighting term is set to 1.0 for all fault types. Therefore, there is no difference in criticality across all of the fault test cases. For fault test cases that are detectable, the fault diagnostic performance metric, Z_j, is calculated as,

$$Z_j = \sum_{l=1}^{n} aR_l \tag{12}$$

where

$$a \quad \begin{cases} -1 \text{ if fault case hypothesized} = \text{true fault case input (if } l = j) \\ \ \ 1 \text{ if fault case hypothesized} \neq \text{true fault case input (if } l \neq j) \end{cases}$$

R_l measurement residual metric for hypothesized fault *l*

The measurement residual agreement metric, R_l, an output from the Inverse Model, is an indicator of the quality of the recovered solution for hypothesized fault type *l*. During the Inverse Model diagnostic evaluation, each fault is recovered individually along with the set of measurement residuals at the recovered solution. The measurement residual metric is used as an indicator of the quality of the recovered solution by combining the measurement residuals into a root-mean square value using Eq. (13). The measurement residual metric is computed for each fault hypothesis and is combined to construct the fault diagnostic performance metric, Z_j, for the given fault test case. The fault diagnostic performance metric should reflect a preference for small residuals, or a small value of R_l, when the true fault type matches the hypothesized fault type and large residuals, or a large value of R_l, when the fault types do not match. The measurement residual agreement metric is determined by,

$$R_l = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \qquad (13)$$

where

R_l measurement residual agreement metric for fault hypothesis *l*
m number of measurements in the sensor suite
y_i input (true) measurement value for sensor *i*
$\hat{y}_i$ recovered (estimated) measurement value for sensor *i*

3.4 Choose Down-Select Algorithm

The remaining element of the Iterative Down-Select Process is the Down-Select Algorithm. This module selects a manageable collection of sensor suites to be evaluated by the System Diagnostic Model and the Sensor Suite Merit Algorithm. The resulting merit values are then re-evaluated by the Down-Select Algorithm to generate a new set of sensor suites. With each cycle of the Iterative Down-Select Process, the set of sensor suites should be improving and converging toward an effective solution to the sensor selection problem. Details of the sensor selection application such as number of candidate sensors, complexity of the sensor selection problem, and user preference will influence the choice of the proper Down-Select Algorithm.

As implied above, the Down-Select Algorithm has common interface requirements for almost every application. On input, the set of sensor suites along with their merit values are required. This information is used to generate a new set of sensor suites to be returned as output. As previously stated, the output set of sensor suites should, in a general sense, be an improvement on the input set of sensor suites.

If the number of sensor suite combinations is computationally feasible to evaluate in full, an exhaustive search (i.e., brute force) may be applied. Because every possible sensor suite combination is evaluated with this methodology, an exhaustive search has the advantage of finding the true optimum sensor suite as defined by the Sensor Suite Merit Algorithm. For this reason, exhaustive search is the preferred Down-Select Algorithm when possible.

When the number of sensor suite combinations is computationally intractable for complete evaluation, other optimization techniques are to be employed. This typically occurs in situations when a large number of sensors are being considered such as in a test-stand application. Optimization techniques like Genetic Algorithms and Random Search have been applied to these larger problems with success (Refs. 8 and 9). Genetic Algorithms emulate evolutionary biological processes while Random Search will randomly generate and evaluate a large number of sensor suite permutations. Without a complete evaluation of every sensor suite possible, the best performing sensor suite can only be assumed to be near-optimal. While they may not be truly optimal, these solutions are acceptable if they achieve the operational objectives. To increase confidence in sensor selection results, comparisons to the findings of other studies or to expert opinion should be made if possible.

In the S4 example application, a Genetic Algorithm (GA) (Ref. 4) is employed as the Down-Select Algorithm. Even though Exhaustive Search is a feasible option for the given number of candidate sensors, the GA is used to illustrate the cyclic nature of the Iterative Down-Select Process. In an Exhaustive Search, all possible sensor suites are evaluated in a single iteration. Therefore, a GA is presented because it is often employed, and because it provides for a more complete demonstration of the S4 methodology.

As there are variations on GA implementations, the particular features of the S4 example application are described here. Initially, a population of candidate sensor suites and their associated merit values are encoded. Each individual member of the population is referred to as a chromosome and represents a single candidate sensor suite. Each chromosome is composed of bits called genes that are associated with the individual sensors. Binary encoding of the gene (1 or 0) indicates if the particular sensor associated with the gene is present within the sensor suite represented by the chromosome. A chromosome mask is employed to force the selection or omission of particular sensors as desired. In the S4 example application, the chromosome mask allows for the control sensors to always be present. Each chromosome in the input population has an assigned merit value received as input from the merit algorithm. Elitism is used to advance a user-defined number of best sensor suites to the next generation without modification. To populate the remainder of the next generation, parents are chosen using roulette-wheel selection (Ref. 4) whereby sensor suites with higher relative merit values have an increased probability of becoming a parent. Once a pair of parent suites is selected, a random number calculation is made to determine if crossover occurs. If it does, single-point crossover is employed to construct offspring sensor suites. In this process, a crossover point is selected randomly, and genes are copied from the parents to the children as depicted in Figure 6. If crossover does not occur, the parents are simply copied to the next generation. Mutation or random bit-flipping occurs during parent-to-child gene copying for its diversifying effects. As generations progress, the population will increasingly converge to a group of similar sensor suites providing good diagnostic capability. Mutation assists in evolving away from local optima in the solution space. The process repeats for a user-defined number of generations. Experience has found for the given example application that there is little benefit in iterating for more than 20 generations. Beyond this number of generations, the population has converged to such a degree that mutation has little to no effect. The basic steps of the GA employed are outlined below. The actions to determine the next generation are shown in Figure 7.

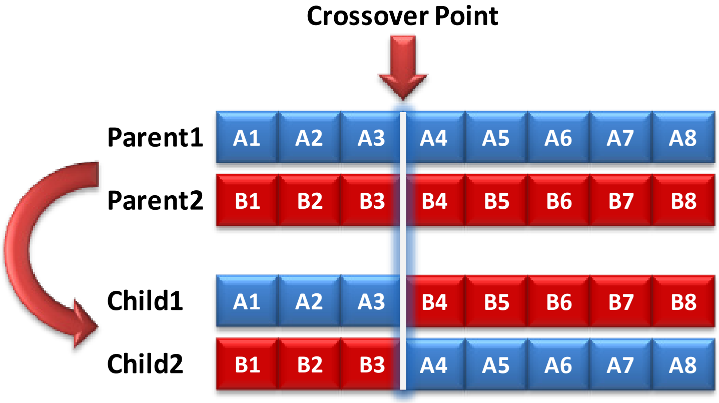


Figure 6.—Single Point Crossover as Used By Genetic Algorithm-Based Down-Select Process

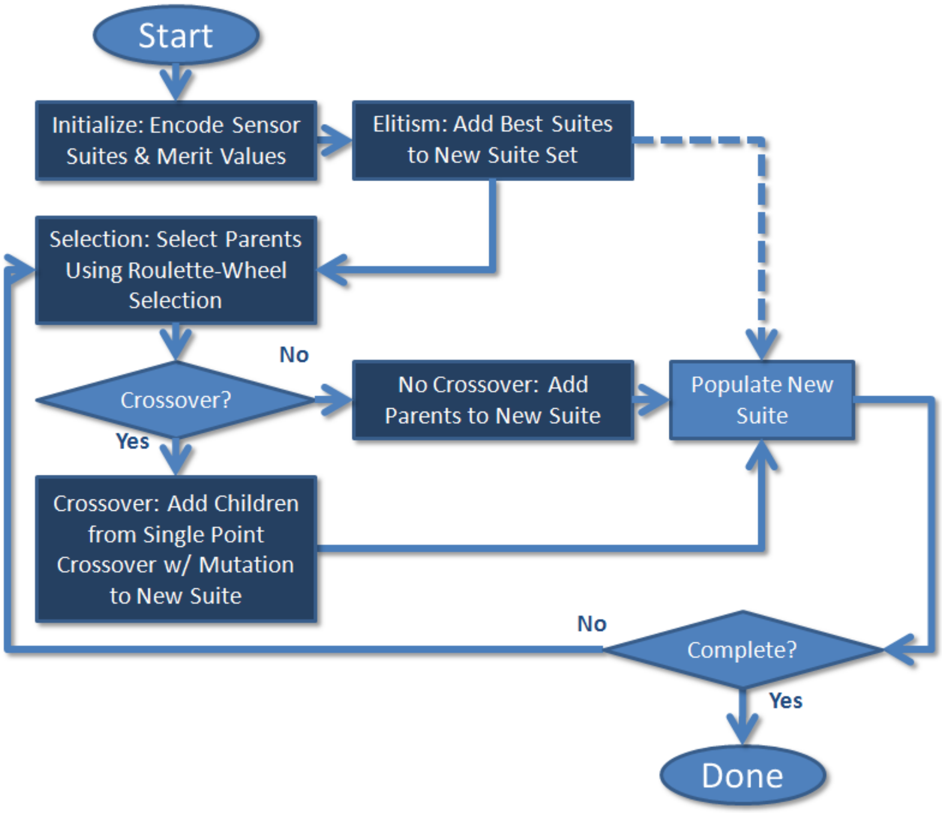

Start
Initialize: Encode Sensor Suites & Merit Values
Elitism: Add Best Suites to New Suite Set
Selection: Select Parents Using Roulette-Wheel Selection
No
Crossover?
No Crossover: Add Parents to New Suite
Populate New Suite
Yes
Crossover: Add Children from Single Point Crossover w/ Mutation to New Suite
No
Complete?
Yes
Done

Figure 7.—Genetic Algorithm Logic Flow Diagram to Populate Next Generation

1. **Initialize**—Input population of sensors (initial population is randomly generated) and their corresponding merit values are calculated by the Sensor Suite Merit Algorithm.
2. **Elitism**—Automatically advance a user-specified number of sensor suites with the highest merit value to the next generation.
3. **Selection**—Select two sensor suites using roulette-wheel selection.
4. **Crossover**—Determine if crossover occurs.
   a. **No Crossover**—Both parents advance to the next generation without modification.
   b. **Crossover**—Apply single point crossover with the possibility of mutation as each gene is copied.
5. **Repeat**—Return to Step 3 until next generation is populated.
6. **Output**—Output the new generation of sensor suites for next initiation of the Iterative Down-Select Process.
7. **Repeat**—Return to Step 1 until target number of generations is reached.

3.5 Compile Data Sets

In this stage, all data sets needed to conduct the sensor selection process are obtained. Data can be gathered from any appropriate source. In a typical application, data is acquired from a computer model that constitutes the System Simulation. While other sources of data are possible, a computer simulation is usually the only means to obtain a data set for all of the candidate sensors for all of the faults. While computer simulations are effective, caution should be exercised. Verification of the simulation is necessary to ensure the results are accurate within some acceptable tolerance. In addition, the boundaries of the simulation's operating envelope are to be recognized. Excursions beyond simulation limits may produce unrealistic results. Care should be taken so that the data is as correct as possible.

In a typical application, two types of fault scenario data are required. In the first, the System Diagnostic Model requires data for training or reference purposes. This is referred to as Inverse Model reference data. In the second, the Sensor Suite Merit Algorithm utilizes test case data to evaluate the candidate sensor suites. These data are typically generated from the System Simulation for the fault scenarios and operating conditions of interest.

In the S4 example application, test case data are selected from the available Inverse Model reference data. For each fault, an Inverse Model reference data interpolation point is used as the fault test case. This is further illustrated in Section 3.5.2.

3.5.1 Inverse Model Reference Data

The System Diagnostic Model uses Inverse Model reference data to classify the fault. For each fault type, nine evenly spaced interpolation points from nominal to the maximum fault magnitude limit are selected that characterize the fault magnitude and associated measurement values.

C-MAPSS allows a user to adjust internal engine health parameters to generate simulated engine measurement data for the component-level faults. Although there are five rotating engine components, only four will have fault condition data generated. Prior C-MAPSS experience has revealed that faults associated with the low pressure compressor are especially difficult to discern. As a result, those component faults scenarios are not included in the S4 example application.

For each engine component fault under consideration, the efficiency and flow capacity health parameters are simultaneously adjusted while all other health parameters remain at their nominal settings as documented by Simon (Ref. 10). Fault ratio and fault magnitude are inputs to Eq. (14) to determine the adjustments to flow capacity and efficiency to characterize the elements of the engine component fault. The fault ratio, $\gamma{:}\eta$, between flow capacity, γ, and efficiency, η, is constant. Because Inverse Model reference data is being generated, nine evenly spaced incremental fault magnitudes, $F_{Magnitude}$, are computed from 0 percent to the maximum fault magnitude. The maximum fault magnitude and fault ratio for each engine component fault is shown in Table 3. The resulting efficiency and flow capacity adjustments for each fault magnitude are shown in Table 4. Both of these adjustments are inputs to C-MAPSS and define a single component fault. In the following descriptions, the component fault magnitudes, not the individual adjustments to efficiency and flow capacity, are used to reference the fault magnitude.

$$\eta_{adjustment} = \frac{F_{magnitude}}{\sqrt{1+(\gamma:\eta)^2}}$$

$$\gamma_{adjustment} = \eta_{adjustment} \times (\gamma:\eta) \tag{14}$$

TABLE 3.—ENGINE COMPONENT FAULT CONDITIONS

Engine component fault	Maximum fault magnitude	Fault ratio ($\gamma{:}\eta$)
FAN	0.07	1.50
High pressure compressor	0.07	1.50
High pressure turbine	0.07	–0.75
Low pressure turbine	0.07	–0.75

With the health parameter inputs in Table 4, C-MAPSS is used to generate the data sets for the S4 example application. In the subsequent data tables, the raw model output is given followed by the conditioned values that are used in the S4 procedure. The conditioned data are calculated using Eq. (1) and the baseline and noise values shown in Table 1. The resulting fan fault scenario data are shown in Table 5 and Table 6. The high pressure compressor fault data are displayed in Table 7 and Table 8. Likewise, the high pressure turbine fault data are located in Table 9 and Table 10 and the low pressure turbine fault data in Table 11 and Table 12.

In the S4 example application, all C-MAPSS data is generated under closed-loop control operating conditions. C-MAPSS has a fan speed controller, so no deviation in N_f occurs under any of the aforementioned fault scenarios.

TABLE 4.—C-MAPSS HEALTH PARAMETER ADJUSTMENT FOR ENGINE COMPONENT FAULTS

Point	FAN			Point	High pressure compressor		
	Magnitude	$\eta_{adjustment}$	$\gamma_{adjustment}$		Magnitude	$\eta_{adjustment}$	$\gamma_{adjustment}$
1	0.0000	0.0000	0.0000	1	0.0000	0.0000	0.0000
2	0.0088	–0.0049	–0.0073	2	0.0088	–0.0049	–0.0073
3	0.0175	–0.0097	–0.0146	3	0.0175	–0.0097	–0.0146
4	0.0263	–0.0146	–0.0218	4	0.0263	–0.0146	–0.0218
5	0.0350	–0.0194	–0.0291	5	0.0350	–0.0194	–0.0291
6	0.0436	–0.0243	–0.0364	6	0.0436	–0.0243	–0.0364
7	0.0525	–0.0291	–0.0437	7	0.0525	–0.0291	–0.0437
8	0.0613	–0.0340	–0.0510	8	0.0613	–0.0340	–0.0510
9	0.0700	–0.0388	–0.0582	9	0.0700	–0.0388	–0.0582
Point	High pressure turbine			Point	Low pressure turbine		
	Magnitude	$\eta_{adjustment}$	$\gamma_{adjustment}$		Magnitude	$\eta_{adjustment}$	$\gamma_{adjustment}$
1	0.00000	0.0000	0.0000	1	0.00000	0.0000	0.0000
2	0.00875	–0.0070	0.0053	2	0.00875	–0.0070	0.0053
3	0.01750	–0.0140	0.0105	3	0.01750	–0.0140	0.0105
4	0.02625	–0.0210	0.0158	4	0.02625	–0.0210	0.0158
5	0.03500	–0.0280	0.0210	5	0.03500	–0.0280	0.0210
6	0.04357	–0.0350	0.0263	6	0.04357	–0.0350	0.0263
7	0.05250	–0.0420	0.0315	7	0.05250	–0.0420	0.0315
8	0.06125	–0.0490	0.0368	8	0.06125	–0.0490	0.0368
9	0.07000	–0.0560	0.0420	9	0.07000	–0.0560	0.0420

TABLE 5.—UNCONDITIONED (RAW) FAN FAULT DATA

	Fan fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	526.14	526.04	525.93	525.81	525.68	525.55	525.42	525.28	525.14
T30	1229.32	1227.88	1226.42	1224.87	1223.25	1221.56	1219.80	1217.96	1216.03
PS30	129.88	129.12	128.34	127.41	126.45	125.47	124.46	123.44	122.39
Wf	1.30	1.29	1.28	1.27	1.26	1.24	1.23	1.22	1.20
Nf	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21
Nc	7920.79	7915.22	7909.51	7903.54	7897.32	7890.86	7884.13	7877.12	7869.83
T48	1503.30	1500.55	1497.74	1494.82	1491.77	1488.59	1485.26	1481.77	1478.13
P15	7.11	7.09	7.07	7.05	7.02	7.00	6.97	6.94	6.91
T21	489.48	489.24	489.00	488.74	488.46	488.18	487.88	487.58	487.26
P25	8.77	8.76	8.74	8.71	8.69	8.66	8.63	8.60	8.57
T50	986.95	985.53	984.10	982.73	981.30	979.81	978.24	976.59	974.86
P50	4.55	4.54	4.52	4.51	4.49	4.48	4.46	4.45	4.43

TABLE 6.—CONDITIONED FAN FAULT DATA

	Fan fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	0.0000	–0.0401	–0.0807	–0.1269	–0.1750	–0.2244	–0.2753	–0.3274	–0.3810
T30	0.0000	–0.1563	–0.3152	–0.4833	–0.6591	–0.8422	–1.0334	–1.2329	–1.4414
PS30	0.0000	–1.1749	–2.3691	–3.8023	–5.2855	–6.7988	–8.3454	–9.9262	–11.5425
Wf	0.0000	–0.7768	–1.5641	–2.4741	–3.4151	–4.3771	–5.3624	–6.3714	–7.4050
Nf	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Nc	0.0000	–0.2809	–0.5693	–0.8709	–1.1851	–1.5115	–1.8514	–2.2051	–2.5735
T48	0.0000	–0.2444	–0.4928	–0.7519	–1.0226	–1.3050	–1.6006	–1.9094	–2.2325
P15	0.0000	–0.4986	–1.0019	–1.7227	–2.4683	–3.2196	–3.9765	–4.7391	–5.5076
T21	0.0000	–0.0643	–0.1302	–0.2015	–0.2759	–0.3531	–0.4334	–0.5166	–0.6031
P25	0.0000	–0.3921	–0.7863	–1.3938	–2.0220	–2.6517	–3.2826	–3.9146	–4.5474
T50	0.0000	–0.1913	–0.3844	–0.5699	–0.7629	–0.9647	–1.1764	–1.3994	–1.6335
P50	0.0000	–0.5355	–1.0761	–1.7183	–2.3775	–3.0444	–3.7203	–4.4054	–5.0998

TABLE 7.—UNCONDITIONED (RAW) HIGH PRESSURE COMPRESSOR FAULT DATA

	High Pressure Compressor Fault Magnitude Increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	526.14	526.34	526.54	526.74	526.94	527.15	527.36	527.57	527.79
T30	1229.32	1232.95	1236.61	1240.30	1244.03	1247.77	1251.51	1255.28	1259.09
PS30	129.88	129.72	129.56	129.40	129.24	129.07	128.89	128.70	128.52
Wf	1.30	1.31	1.31	1.32	1.32	1.33	1.34	1.34	1.35
Nf	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21
Nc	7920.79	7927.19	7933.66	7940.22	7946.87	7953.51	7960.07	7966.70	7973.42
T48	1503.30	1510.92	1518.63	1526.40	1534.25	1542.28	1550.62	1559.06	1567.58
P15	7.11	7.11	7.11	7.11	7.11	7.11	7.11	7.12	7.12
T21	489.48	489.50	489.52	489.54	489.56	489.58	489.60	489.62	489.64
P25	8.77	8.78	8.79	8.79	8.80	8.80	8.81	8.82	8.82
T50	986.95	992.76	998.65	1004.59	1010.60	1016.74	1023.12	1029.59	1036.12
P50	4.55	4.55	4.54	4.54	4.54	4.54	4.54	4.53	4.53

TABLE 8.—CONDITIONED HIGH PRESSURE COMPRESSOR FAULT DATA

	High pressure compressor fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	0.0000	0.0752	0.1510	0.2271	0.3036	0.3817	0.4624	0.5438	0.6256
T30	0.0000	0.3932	0.7906	1.1908	1.5949	2.0011	2.4059	2.8151	3.2280
PS30	0.0000	–0.2446	–0.4914	–0.7398	–0.9887	–1.2500	–1.5320	–1.8165	–2.1033
Wf	0.0000	0.4374	0.8778	1.3196	1.7648	2.2157	2.6814	3.1501	3.6211
Nf	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Nc	0.0000	0.3233	0.6501	0.9813	1.3171	1.6524	1.9835	2.3184	2.6579
T48	0.0000	0.6757	1.3592	2.0485	2.7451	3.4571	4.1967	4.9451	5.7012
P15	0.0000	0.0236	0.0472	0.0709	0.0946	0.1188	0.1436	0.1686	0.1936
T21	0.0000	0.0055	0.0111	0.0166	0.0222	0.0279	0.0337	0.0396	0.0454
P25	0.0000	0.1368	0.2741	0.4115	0.5490	0.6891	0.8332	0.9779	1.1229
T50	0.0000	0.7855	1.5807	2.3832	3.1955	4.0251	4.8872	5.7602	6.6430
P50	0.0000	–0.0758	–0.1523	–0.2293	–0.3060	–0.3861	–0.4692	–0.5532	–0.6380

TABLE 9.—UNCONDITIONED (RAW) HIGH PRESSURE TURBINE FAULT DATA

	High pressure turbine fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	526.14	526.55	526.96	527.38	527.81	528.23	528.66	529.05	529.43
T30	1229.32	1226.70	1224.06	1221.35	1218.65	1215.95	1213.26	1210.57	1207.88
PS30	129.88	128.59	127.29	125.99	124.70	123.43	122.19	120.95	119.73
Wf	1.30	1.31	1.32	1.33	1.35	1.36	1.37	1.38	1.39
Nf	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21
Nc	7920.79	7909.47	7898.08	7886.35	7874.78	7863.36	7852.09	7840.98	7830.02
T48	1503.30	1518.98	1535.10	1552.02	1569.06	1586.21	1603.50	1620.98	1638.62
P15	7.11	7.11	7.11	7.12	7.12	7.12	7.12	7.12	7.12
T21	489.48	489.52	489.56	489.60	489.64	489.69	489.73	489.77	489.81
P25	8.77	8.79	8.80	8.81	8.82	8.84	8.85	8.86	8.87
T50	986.95	998.76	1010.92	1023.71	1036.59	1049.60	1062.71	1076.00	1089.44
P50	4.55	4.54	4.54	4.54	4.53	4.53	4.52	4.52	4.52

TABLE 10.—CONDITIONED HIGH PRESSURE TURBINE FAULT DATA

	High pressure turbine fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	0.0000	0.1529	0.3086	0.4710	0.6331	0.7950	0.9567	1.1051	1.2475
T30	0.0000	–0.2848	–0.5707	–0.8647	–1.1580	–1.4506	–1.7425	–2.0344	–2.3260
PS30	0.0000	–1.9967	–3.9847	–5.9978	–7.9796	–9.9293	–11.8485	–13.7515	–15.6317
Wf	0.0000	0.8701	1.7561	2.6746	3.5894	4.5011	5.4097	6.3158	7.2195
Nf	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Nc	0.0000	–0.5716	–1.1466	–1.7393	–2.3236	–2.9001	–3.4694	–4.0301	–4.5836
T48	0.0000	1.3910	2.8205	4.3213	5.8321	7.3537	8.8870	10.4370	12.0017
P15	0.0000	0.0478	0.0962	0.1463	0.1959	0.2451	0.2939	0.3427	0.3914
T21	0.0000	0.0112	0.0226	0.0343	0.0460	0.0575	0.0690	0.0804	0.0919
P25	0.0000	0.2775	0.5580	0.8486	1.1363	1.4215	1.7042	1.9252	2.1177
T50	0.0000	1.5959	3.2392	4.9659	6.7070	8.4634	10.2358	12.0309	13.8461
P50	0.0000	–0.1560	–0.3147	–0.4835	–0.6528	–0.8230	–0.9944	–1.1724	–1.3538

TABLE 11.—UNCONDITIONED (RAW) LOW PRESSURE TURBINE FAULT DATA

	Low pressure turbine fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	526.14	525.79	525.42	525.02	524.63	524.22	523.82	523.42	523.01
T30	1229.32	1232.10	1234.90	1237.74	1240.61	1243.49	1246.40	1249.33	1252.28
PS30	129.88	130.88	131.90	132.91	133.93	134.97	136.02	137.09	138.17
Wf	1.30	1.31	1.32	1.33	1.34	1.36	1.37	1.38	1.39
Nf	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21	1936.21
Nc	7920.79	7931.05	7941.47	7952.10	7962.86	7973.76	7984.80	7996.00	8007.34
T48	1503.30	1503.98	1504.76	1505.71	1506.72	1507.81	1508.97	1510.18	1511.46
P15	7.11	7.11	7.11	7.11	7.10	7.10	7.10	7.10	7.10
T21	489.48	489.44	489.40	489.37	489.33	489.29	489.26	489.22	489.18
P25	8.77	8.76	8.75	8.74	8.72	8.71	8.69	8.68	8.66
T50	986.95	991.42	995.96	1000.65	1005.38	1010.16	1014.99	1019.87	1024.80
P50	4.55	4.57	4.59	4.61	4.63	4.65	4.67	4.70	4.72

TABLE 12.—CONDITIONED LOW PRESSURE TURBINE FAULT DATA

	Low pressure turbine fault magnitude increment								
	Point 1	Point 2	Point 3	Point 4	Point 5	Point 6	Point 7	Point 8	Point 9
Fault magnitude	0.0000	0.0086	0.0175	0.0263	0.0350	0.0436	0.0525	0.0613	0.0700
T24	0.0000	–0.1346	–0.2738	–0.4251	–0.5771	–0.7296	–0.8827	–1.0365	–1.1911
T30	0.0000	0.3008	0.6047	0.9133	1.2240	1.5370	1.8522	2.1698	2.4897
PS30	0.0000	1.5427	3.0987	4.6591	6.2385	7.8378	9.4569	11.0961	12.7560
Wf	0.0000	0.8098	1.6358	2.4809	3.3414	4.2184	5.1114	6.0207	6.9469
Nf	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Nc	0.0000	0.5183	1.0445	1.5812	2.1249	2.6752	3.2328	3.7981	4.3711
T48	0.0000	0.0600	0.1290	0.2135	0.3035	0.4003	0.5026	0.6103	0.7235
P15	0.0000	–0.0424	–0.0849	–0.1271	–0.1696	–0.2125	–0.2557	–0.2994	–0.3434
T21	0.0000	–0.0099	–0.0199	–0.0298	–0.0398	–0.0498	–0.0600	–0.0702	–0.0805
P25	0.0000	–0.2459	–0.5172	–0.8536	–1.1930	–1.5350	–1.8802	–2.2286	–2.5803
T50	0.0000	0.6038	1.2181	1.8506	2.4895	3.1360	3.7888	4.4480	5.1137
P50	0.0000	0.8889	1.7957	2.7166	3.6585	4.6218	5.6074	6.6160	7.6483

3.5.2 Test Case Scenario Data

Test case scenario data sets are constructed to serve as inputs to the Sensor Suite Merit Algorithm and the System Diagnostic Model. As such, the required size and format of the test case scenario data set is dependent on the design of these two elements.

In the S4 example application, the test case data are simply copied from the mid-point fault magnitude case (interpolation point 5) of each fault type in the Inverse Model reference data. This was done to simplify presentation of the example application. The fault test case data are shown in Table 13.

TABLE 13.—FAULT TEST CASES

	Fan	HPC	HPT	LPT
Fault magnitude	0.0350	0.0350	0.0350	0.0350
T24	–0.1750	0.3036	0.6331	–0.5771
T30	–0.6591	1.5949	–1.1580	1.2240
PS30	–5.2855	–0.9887	–7.9796	6.2385
Wf	–3.4151	1.7648	3.5894	3.3414
Nf	0.0000	0.0000	0.0000	0.0000
Nc	–1.1851	1.3171	–2.3236	2.1249
T48	–1.0226	2.7451	5.8321	0.3035
P15	–2.4683	0.0946	0.1959	–0.1696
T21	–0.2759	0.0222	0.0460	–0.0398
P25	–2.0220	0.5490	1.1363	–1.1930
T50	–0.7629	3.1955	6.7070	2.4895
P50	–2.3775	–0.3060	–0.6528	3.6585

3.6 Develop Software Modules

The remaining step before the sensor selection process can begin is to implement the elements of S4 as functioning software. The sophistication of the software can extend from one of high automation to one requiring constant user intervention and adjustment. The available computational resources, the demands of the sensor selection application, and the programming ability of the user dictate how complicated the software should be.

While S4 is intended to be tailored for each specific application, software developed for other S4 applications can provide a basis from which a customized framework is built. Furthermore, software from previous S4 implementations is normally used to begin the development of new frameworks. Appropriate software elements are selected and modified for incorporation into the new application. Therefore, standardizing software interfaces helps promote modularity and may reduce development efforts.

For the S4 example application, MATLAB is selected as the software programming environment. Key reasons include ease of programming and acceptable execution speed. Additionally, MATLAB allows for the incorporation of compiled code through MEX-files. MEX is a shorthand word for “MATLAB executable.” MEX-files are library functions written in FORTRAN, C, or C++ that can be evoked from the MATLAB environment. Since these files are compiled and not interpreted, their use greatly increases the speed of execution of the MATLAB routine. They are ideal for computationally intensive functions that are not modified often. Another benefit of MEX-files is that they permit the inclusion of software packages not originally intended for MATLAB. In this way, legacy software packages may be integrated with MATLAB.

MATLAB software modules utilized in the S4 example application are described in the following subsections.

3.6.1 KnowledgeBase.m

`KnowledgeBase.m` contains general information utilized by multiple software modules in the performance of the Iterative Down-Select Process. Examples of data it includes are Inverse Model reference data and fault test case data. To help promote modularity, `KnowledgeBase.m` does not include data which is only pertinent to a single software module, that data is instead located in the specific module that utilizes it.

3.6.2 MeritAlgorithm.m

The Sensor Suite Merit Algorithm described in Section 3.2 is implemented in `MeritAlgorithm.m`. In the file, there is an initialization section, a merit algorithm section, and a parameter clean-up or termination section. The initialization section will set the penalty term parameter values and allocate memory for vectors used by the merit algorithm. In the merit algorithm section, the evaluation procedure is performed for each candidate sensor suite. In the examination, the diagnostic term is first reset to zero and the System Diagnostic Model is called for each fault test case. The returned value of the diagnostic performance of the candidate sensor suite for the fault is multiplied by the fault criticality and is added to the diagnostic term. After the diagnostic term is computed, the penalty term and utility measure are calculated. The product of these three parameters forms the merit value of the candidate sensor suite. When the processing is complete, memory is freed in the parameter clean-up section. Because of the association with the System Diagnostic Model, the Sensor Suite Merit Algorithm file will also perform variable clean-up on the diagnostic parameters. The process flow is depicted in Figure 8.

3.6.3 SystemDiagnostic.m

The System Diagnostic Model described in Section 3.3 is implemented in `SystemDiagnostic.m`. The file has an initialization section and two diagnostic sections. Because the variables are still being used at process termination, the variable clean-up processing occurs within `MeritAlgorithm.m`. In the initialization section, the threshold limits are set and memory is allocated for the measurement residuals vector. The first diagnostic section performs fault detection. If the fault is detected, fault isolation is performed in the second diagnostic section. If the fault is not detected, the diagnostic performance value is set to zero and the diagnostic process ends for that fault. The diagnostic process flow is shown in Figure 9.

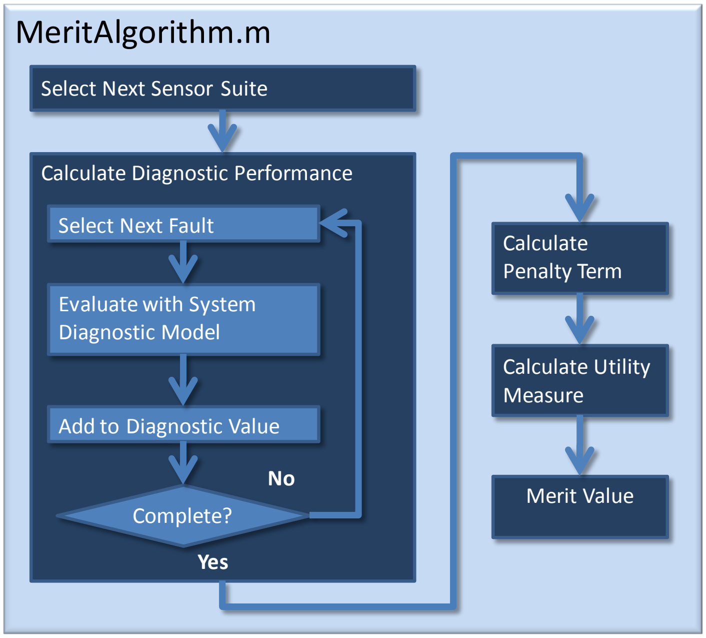


Figure 8.—Process Flow of Sensor Suite Merit Algorithm File

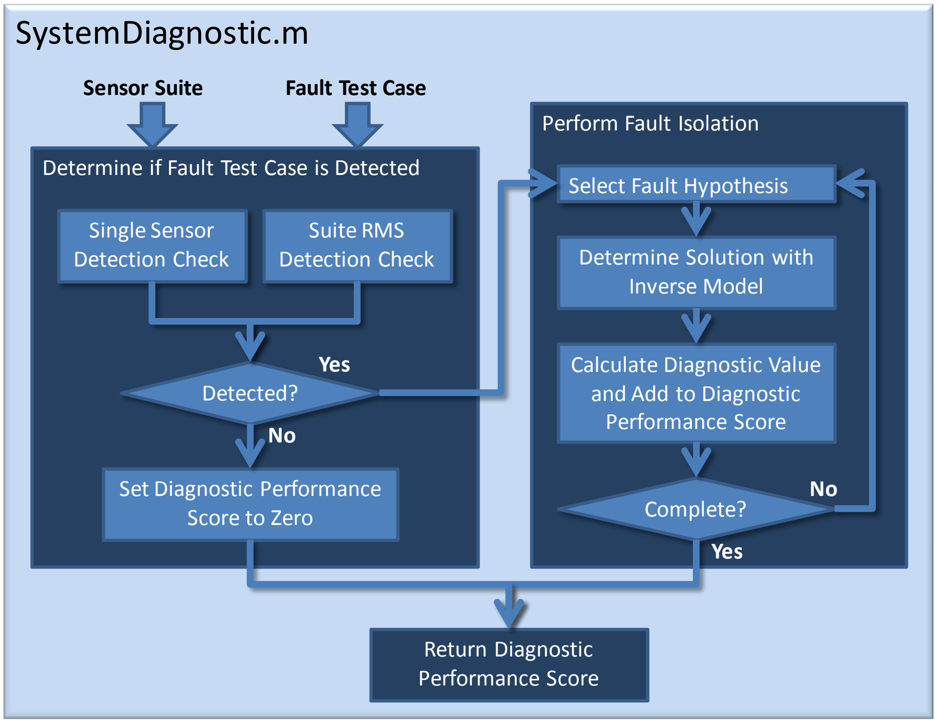

SystemDiagnostic.m
Sensor Suite
Fault Test Case
Determine if Fault Test Case is Detected
Single Sensor Detection Check
Suite RMS Detection Check
Yes
Detected?
No
Set Diagnostic Performance Score to Zero
Perform Fault Isolation
Select Fault Hypothesis
Determine Solution with Inverse Model
Calculate Diagnostic Value and Add to Diagnostic Performance Score
No
Complete?
Yes
Return Diagnostic Performance Score

Figure 9.—Process Flow of System Diagnostic Model File

3.6.4 GeneticAlgorithm.m

The Genetic Algorithm described in Section 3.4 is implemented in `GeneticAlgorithm.m` and `mexGA.mexw32`. Most of the functionality of the Genetic Algorithm is contained within `mexGA.mexw32`. The `GeneticAlgorithm.m` file has an initialization section, an execution section, and a clean-up section. The initialization section sets the values of the Genetic Algorithm parameters. On the first call, only the initialization section is executed because the Iterative Down-Select Process is still initializing and merit values have not been assigned to the sensor suites. On subsequent calls, the execution section is entered and the Genetic Algorithm MEX-File is evoked. Parameter clean-up is performed during the last iteration of the process.

3.7 Create and Conduct the Sensor Selection Process

Here, the Iterative Down-Selection Process is created and executed. In the S4 example application, a higher-level Matlab script, or m-file, is developed that implements the Iterative Down-Select Process using the m-files and MEX-files developed in Section 3.6.

3.7.1 Create IterativeDownSelect.m

The subroutine that begins the sensor selection process is `IterativeDownSelect.m`. The first action of this file is to load Knowledge Base information. Next, an iterative loop is entered where the subroutines *`MeritAlgorithm`* and *`GeneticAlgorithm`* are called a prescribed number of times. Merit calculations are done within *`MeritAlgorithm`* which calls *`SystemDiagnostic`* for the diagnostic performance analysis. The subroutine *`SystemDiagnostic`* subsequently calls the MEX-File `mexInverseModel.mexw32` to execute the Inverse Model. For *`GeneticAlgorithm`*, the MEX-File `mexGA.mexw32` performs the intensive Genetic Algorithm computing. When the iterations are complete, the subroutine *`PostAnalysis`* is evoked to do post analysis and any function termination processing. The calling structure is shown in Figure 10.

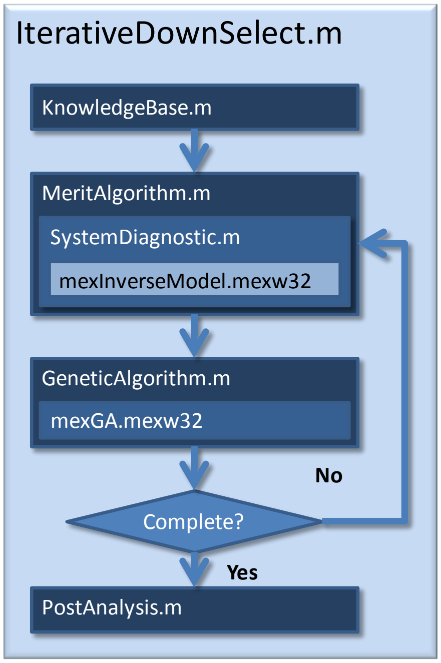


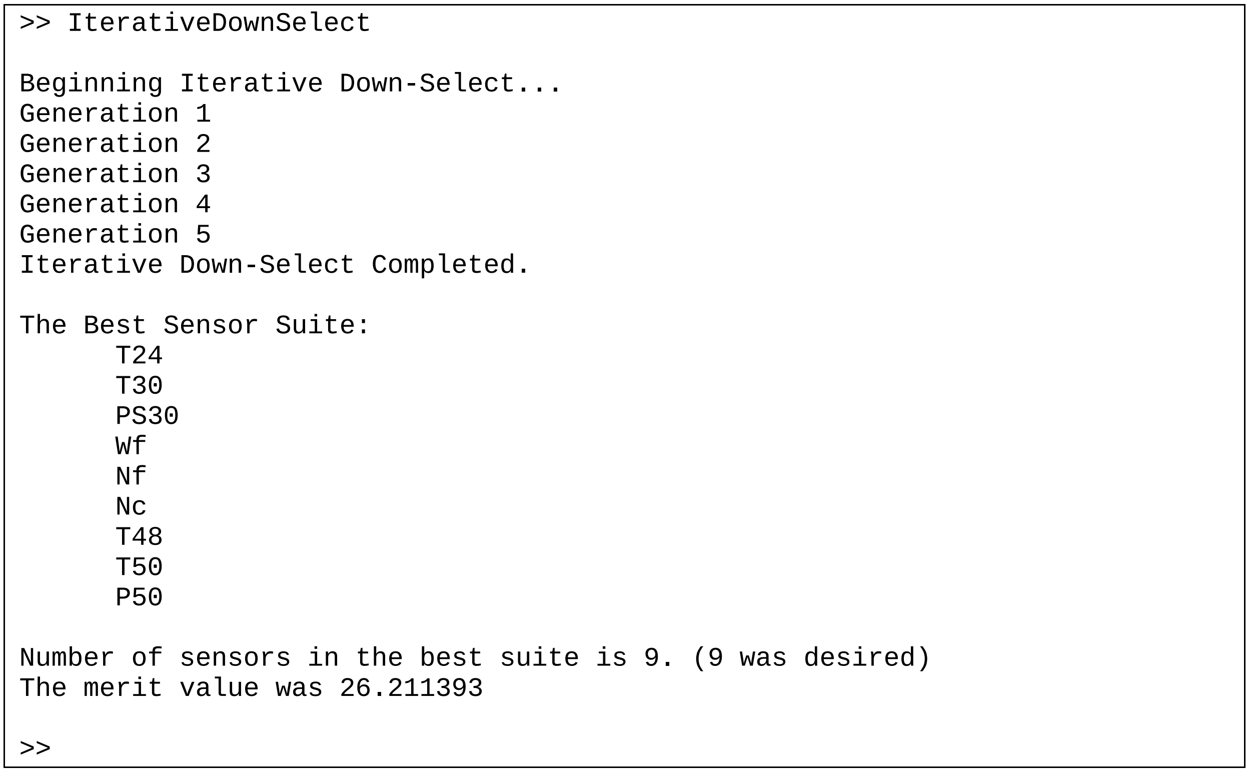
Figure 10.—Process Flow of Iterative Down-Select Process Files

3.7.2 Execute IterativeDownSelect.m

In the S4 example application, the procedure is executed by typing “`IterativeDownSelect`” in the MATLAB Command Window. As the procedure executes, the status is displayed on the screen as shown in Figure 11.

In this evaluation, the best sensor suite was found to include the control sensor suite plus T50 and P50 with a merit value of 26.211393. Although not shown in Figure 11, the two next best sensor suites were found to include the set of control sensors augmented by P25 and P50, and the set of control sensors with P25 and T50. Both of these suites had the same merit value of 24.279624.

The S4 example application can be modified to exercise the process. While the default settings of the parameters are sufficient, adjustments can be made to explore through experimentation. Interesting results can be obtained by changing the desired number of sensors. By default, this number of sensors is set to 9. This translates to two candidate sensors being selected to augment the 7 control sensors. For practical purposes, this number can vary from 8 (for 1 additional sensor) to 11 (for all sensors except one). Another area of experimentation is the number of generations. Because the number of possible sensor suite combinations with 5 candidate sensors is limited, the number of generations is set to five. Larger pools of candidate sensors require a greater number of iterations. With the setup described in this guide, it has been observed that there is never a benefit in more than 20 generations regardless of the number of additional candidate sensors. After 20 generations, the collection of candidate sensor suites has insufficient diversity to generate unique sensor suites. In effect, the process converges to an optimum. Better results are obtained by restarting the process with an initial set of sensor suites that includes a few of the better performing sensor suites.



```
>> IterativeDownSelect

Beginning Iterative Down-Select...
Generation 1
Generation 2
Generation 3
Generation 4
Generation 5
Iterative Down-Select Completed.

The Best Sensor Suite:
      T24
      T30
      PS30
      Wf
      Nf
      Nc
      T48
      T50
      P50

Number of sensors in the best suite is 9. (9 was desired)
The merit value was 26.211393

>>
```

Figure 11.—Example of Iterative Down-Select Process Software Executing

When non-deterministic Down-Select Algorithms like Genetic Algorithms are employed, multiple runs of the Iterative Down-Selection Process are usually necessary. The randomness of the Down-Select Algorithm can produce different results each time the process is executed. This is especially true with larger sensor selection problems. When this occurs, the analysis should be repeated with the best results being recorded and compared. By repeating the study and examining the consistency of the results, confidence that the better performing sensor suites are being found is gained. (Please note that the S4 example application software has been configured to be repeatable in order to produce results that correspond to this guide by resetting the random number generators during application initialization.) After a set of well-performing sensor suites have been identified, the Final Selection Process is performed to select the final sensor suite.

3.8 Create and Perform Final Selection Process

Once a set of sensor suites have been identified from the Iterative Down-Select Process, the Statistical Evaluation Algorithm within the Final Selection Process (Figure 12) can begin. As the main element of the Final Selection Process, the Statistical Evaluation Algorithm, analyzes these sensor suites with a rigorous evaluation. Factors that are otherwise too computationally expensive or simply too complex to include in the Iterate Down-Select Process may now be incorporated. As with other components of S4, the user has the flexibility to incorporate capabilities tailored to their specific application

The implementation of the Statistical Evaluation Algorithm in the S4 example application is described in the following subsections. The modules developed in Section 3.6 are used as the primary building blocks with slight modifications as noted to facilitate the Final Selection Process.

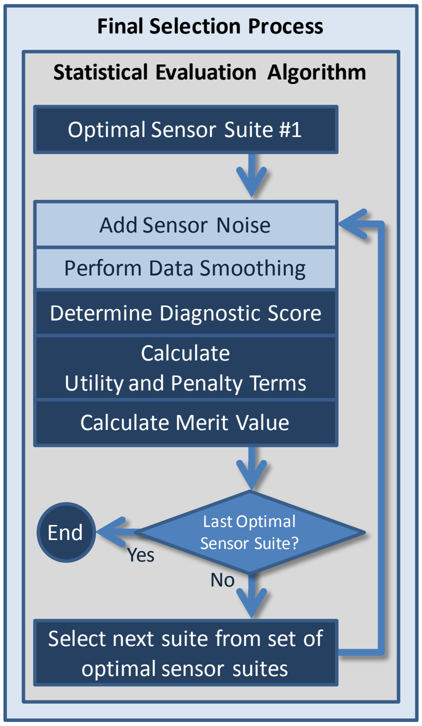
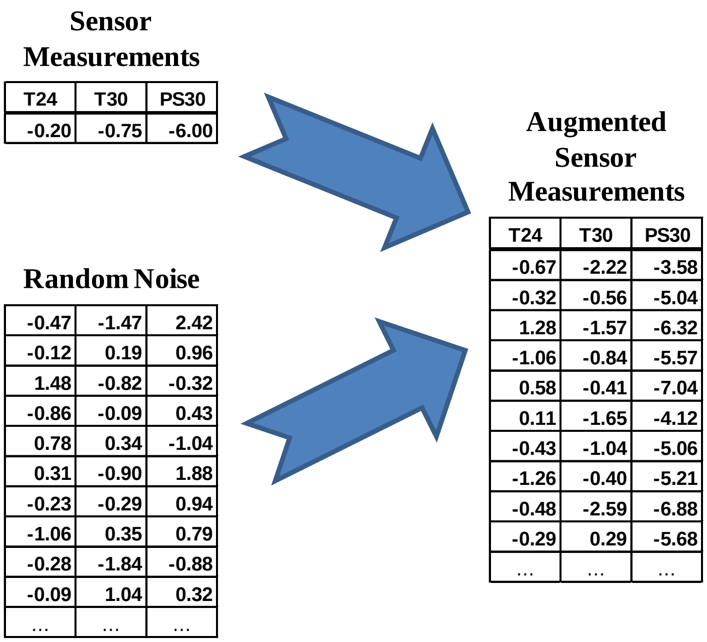

Final Selection Process
Statistical Evaluation Algorithm
Optimal Sensor Suite #1
Add Sensor Noise
Perform Data Smoothing
Determine Diagnostic Score
Calculate Utility and Penalty Terms
Calculate Merit Value
Last Optimal Sensor Suite?
Yes
End
No
Select next suite from set of optimal sensor suites

Figure 12.—Process Flow of Final Selection Process



Sensor Measurements

T24	T30	PS30
-0.20	-0.75	-6.00

Random Noise

-0.47	-1.47	2.42
-0.12	0.19	0.96
1.48	-0.82	-0.32
-0.86	-0.09	0.43
0.78	0.34	-1.04
0.31	-0.90	1.88
-0.23	-0.29	0.94
-1.06	0.35	0.79
-0.28	-1.84	-0.88
-0.09	1.04	0.32
…	…	…

Augmented Sensor Measurements

T24	T30	PS30
-0.67	-2.22	-3.58
-0.32	-0.56	-5.04
1.28	-1.57	-6.32
-1.06	-0.84	-5.57
0.58	-0.41	-7.04
0.11	-1.65	-4.12
-0.43	-1.04	-5.06
-1.26	-0.40	-5.21
-0.48	-2.59	-6.88
-0.29	0.29	-5.68
…	…	…

Figure 13.—Adding Random Noise to Sensor Measurement Values

3.8.1 Knowledge Base

The Performance Related Information and System Simulation contained in the Knowledge Base are used to institute a sensor suite evaluation environment that is as authentic in all regards as possible. During operation the target system can experience various sources of measurement uncertainty. For example, sensor data are subject to signal noise, calibration and measurement resolution issues, and potential limits in the range of operation among others factors. Target system performance characteristics are influenced by factors such as hardware build discrepancies, modes of operation and variations in environmental conditions. This information may be used to produce new data sets.

In the S4 example application, the fault data are the same as that used in the Iterative Down-Select Process, but a larger sample set is generated with random noise superimposed on each measurement. To accomplish this, a noise array sized for a sequence of 100 time intervals for each sensor measurement is created. A random number generator with a normal distribution is employed to populate the noise array. The vector of sensor measurements for the fault test case is added to each row of the noise array to create the augmented sensor measurement array. Figure 13 illustrates the process with 3 sensor measurements and random noise for 10 time samples.

Special care should be exercised when using noisy data. To insure that consistent evaluations are made from the data, the noise values superimposed on the fault test case data should be the same for each sensor for each fault test case. In other words, every sensor should have the same sequence of noise added to it in every evaluation. Otherwise, the results could be skewed by the differing noise values.

3.8.1 Merit Algorithm

Typically, the Sensor Suite Merit Algorithm is the same as the one employed in the Iterative Down-Select Process though it may be modified to account for changes in the evaluation setting. The penalty term may be removed if undesired characteristics have been filtered from the subset of well performing sensor suites. Conversely, the penalty term may be reconstructed to penalize for other features relevant to the statistical nature of the Final Selection Process. In a similar manner, the utility measure can be expanded to incorporate detailed performance specifications of each particular sensor type.

In the S4 example application, the penalty term and the utility measure were removed to simplify the process presentation. Hence, the merit value for each sensor suite is equivalent to the summation of the diagnostic terms for each fault test case from the System Diagnostic Model, see Eq. (11).

3.8.2 System Diagnostic Model

The enhanced data employed in the Statistical Evaluation Algorithm usually require the System Diagnostic Model to be improved. A preprocessor may be inserted to adjust the data to be in a form that is compatible with the System Diagnostic Model. This may include utilizing a data smoothing technique for noisy data, combining redundant sensor measurements or correcting bias effects in the data. Additional modifications may be necessary to tune the diagnostic algorithm for features present within the data sets.

In the S4 example application, data smoothing is used to reduce the effect of noise in the fault test case data so that the same fault detection and isolation algorithms as described in Section 3.3 may be employed. The data are smoothed by averaging over a three sample window. The same fault detection thresholds are used with 3.0 for the single measurement value checks and 0.5 for the sensor suite root-mean-square check. When a fault is detected, the smoothed averaged data is used by the Inverse Model (see Section 3.3.2) for fault isolation. The diagnostic score is calculated as before (see Section 3.3.3).

3.8.3 Down-Select Algorithm

The Down-Select Algorithm is not present because the Statistical Evaluation Algorithm is not iterative. Each candidate sensor suite is evaluated once and results are given.

3.8.4 Software Application

The Final Selection Process software was developed in MATLAB. The files and directory structure used in the S4 example application are given in Section 4.0.

3.8.5 Execute Process

The Final Selection Process is executed by typing "`FinalSelection`" at the MATLAB command line prompt. The output should resemble Figure 14. In this setup, three of the better performing sensor suites from the Iterative-Down Select Process are examined. Once the process is evoked, no other user input is necessary.

In the output, the results of the Iterative Down-Select Process are confirmed. The performances of the sensor suites are consistent. The first sensor suite remains the best with a merit value of 20.415497. The remaining two sensor suites have identical merit values of 19.962710.

```
>> FinalSelection

Beginning Final Selection…
Final Selection Completed.

Sensor suite 1 has a merit value of 20.415497.
Sensor suite 2 has a merit value of 19.962710.
Sensor suite 3 has a merit value of 19.962710.

>>
```

Figure 14.—Example of Final Selection Process Software Executing

In many applications, it is beneficial to repeat the Final Selection analysis with different formulations of the Sensor Suite Merit Algorithm while keeping the set of sensor suites under evaluation the same.

Typically, the setup of the Sensor Suite Merit Algorithm is subjective with the merit factors and weighting terms established using user estimates. Modifying the Sensor Suite Merit Algorithm assists the user in assessing the sensitivity of the sensor suites to evolving or conflicting requirements. With the variations in the figures of merit, results are compared to illustrate how the sensor suites perform when importance is shifted. This helps to reveal how robust the sensor suites may be to conflicting requirements. After all of the results are analyzed, the final decision is made by the user for the best sensor suite for the specific application. It is not unusual for several sensor suites to be considered as a viable solution.

4.0 Overview of S4 Software Package

The S4 software package is classified as Class E software under NASA Procedural Requirement 7150.2A. In its current form, the S4 software package and example problem are designed to “perform minor desktop analysis of science or experimental data.” They are not intended for use in operational flight or ground systems, or for use in supporting technical decisions relevant to those systems. However, the application of the S4 technology to a specific flight or ground system could result in application software that is appropriate for use in flight or ground systems.

The S4 example application software was constructed using MATLAB version R2010a running under the Windows XP Professional (32-bit) operating system. The software was designed with a separate module for each S4 component to make the software arrangement easier to understand and to present to the user. In this manner, each element of the Iterative Down-Select Process and Final Selection Process is contained within a single source code file, with files organized to follow the general flow of the S4 process as described in Sections 2.0 and 3.0 of this guide. For portability and for compatibility with the LCC compiler (provided with MATLAB), MEX-files included with the S4 software package were written in C. All files required to execute and verify results for the S4 example application are bundled in the S4 software package.

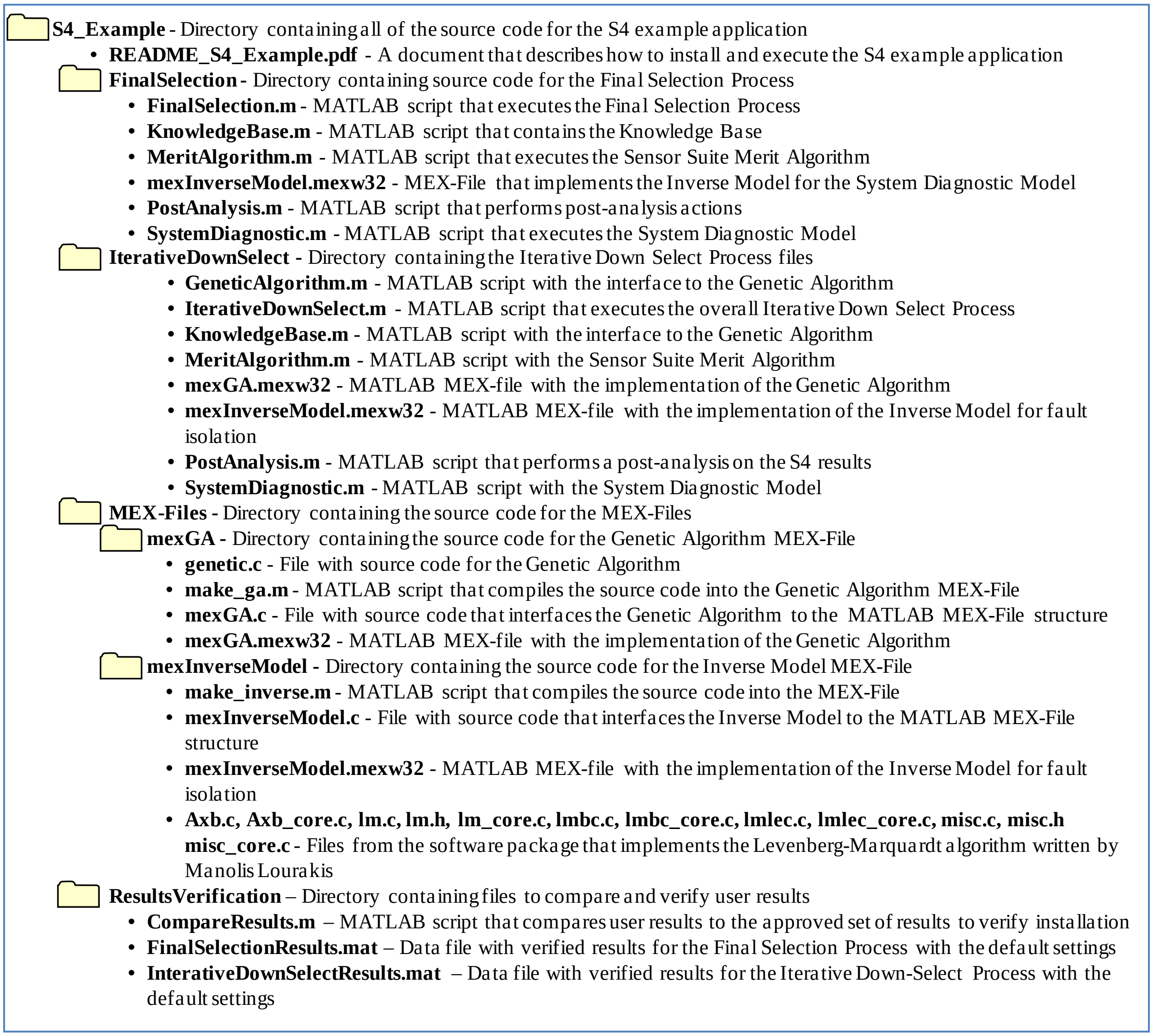
The S4 example application is available to non-NASA parties via the NASA Glenn Research Center online software catalog (Ref. 3). The software is packaged in a compressed file called `S4_Example.zip`. The file can be placed in any folder or directory on the user’s computer as there are no location dependencies. Files should be extracted to the user’s computer from the zip-file using the directory structure embedded in the zip-file. Once the file is uncompressed (e.g., “unzipped”), the file structure shown in Figure 15 will exist with four top-level directories. Additionally at the top-level, the file `README_S4_Example.pdf` contains detailed instructions for installing and executing the S4 example application software.

A Knowledge Base containing all of the data required to run the S4 example application is provided in the form of a Matlab m-file, `KnowledgeBase.m`. Separate versions of the Knowledge Base are provided for the Iterative Down-Select Process and the Final Selection Process. These m-files are located in the `IterativeDownSelect` and `FinalSelection` directories, respectively.

The `IterativeDownSelect` directory contains all of the Iterative Down-Select Process files. Likewise, all of the Final Selection Process files are located in the `FinalSelection` directory. Source code for MATLAB MEX-files used by the Iterative Down-Select Process and the Final Selection Process is located in the `MEX-Files` subdirectory with lower-level subdirectories containing the Genetic Algorithm and Inverse Model functions.

The software has been configured with default settings. To insure proper installation, the user should follow the steps outlined in `README_S4_Example.pdf`. This includes running the Matlab script `CompareResults.m` in the `ResultsVerification` directory. This Matlab script runs the Iterative Down-Select Process and the Final Selection Process. It then compares the results with baseline data provided in the S4 example application software. The user should also note that, prior to verifying the installation, it may be necessary to re-compile the MEX-files included in the S4 example application software. This is likely to be the case if the user is not running the software on a 32-bit Windows operating system. The `README_S4_Example.pdf` file contains instructions for compiling the MEX-files. After the software installation has been verified, the seed to the random number generator should be modified as described in the `README_S4_Example.pdf` file.

With the software properly installed, the S4 m-files can be executed to explore the S4 example problem. To execute the Iterative Down-Select Process, the user should navigate to the `IterativeDownSelect` directory in MATLAB and run the Matlab script `IterativeDownSelect.m`. In a similar fashion, executing the Final Selection Process requires that the user navigate to directory `FinalSelection`, and run the Matlab script `FinalSelection.m`.



S4_Example - Directory containing all of the source code for the S4 example application

- **README_S4_Example.pdf** - A document that describes how to install and execute the S4 example application

FinalSelection - Directory containing source code for the Final Selection Process

- **FinalSelection.m** - MATLAB script that executes the Final Selection Process
- **KnowledgeBase.m** - MATLAB script that contains the Knowledge Base
- **MeritAlgorithm.m** - MATLAB script that executes the Sensor Suite Merit Algorithm
- **mexInverseModel.mexw32** - MEX-File that implements the Inverse Model for the System Diagnostic Model
- **PostAnalysis.m** - MATLAB script that performs post-analysis actions
- **SystemDiagnostic.m** - MATLAB script that executes the System Diagnostic Model

IterativeDownSelect - Directory containing the Iterative Down Select Process files

- **GeneticAlgorithm.m** - MATLAB script with the interface to the Genetic Algorithm
- **IterativeDownSelect.m** - MATLAB script that executes the overall Iterative Down Select Process
- **KnowledgeBase.m** - MATLAB script with the interface to the Genetic Algorithm
- **MeritAlgorithm.m** - MATLAB script with the Sensor Suite Merit Algorithm
- **mexGA.mexw32** - MATLAB MEX-file with the implementation of the Genetic Algorithm
- **mexInverseModel.mexw32** - MATLAB MEX-file with the implementation of the Inverse Model for fault isolation
- **PostAnalysis.m** - MATLAB script that performs a post-analysis on the S4 results
- **SystemDiagnostic.m** - MATLAB script with the System Diagnostic Model

MEX-Files - Directory containing the source code for the MEX-Files

mexGA - Directory containing the source code for the Genetic Algorithm MEX-File

- **genetic.c** - File with source code for the Genetic Algorithm
- **make_ga.m** - MATLAB script that compiles the source code into the Genetic Algorithm MEX-File
- **mexGA.c** - File with source code that interfaces the Genetic Algorithm to the MATLAB MEX-File structure
- **mexGA.mexw32** - MATLAB MEX-file with the implementation of the Genetic Algorithm

mexInverseModel - Directory containing the source code for the Inverse Model MEX-File

- **make_inverse.m** - MATLAB script that compiles the source code into the MEX-File
- **mexInverseModel.c** - File with source code that interfaces the Inverse Model to the MATLAB MEX-File structure
- **mexInverseModel.mexw32** - MATLAB MEX-file with the implementation of the Inverse Model for fault isolation
- **Axb.c, Axb_core.c, lm.c, lm.h, lm_core.c, lmbc.c, lmbc_core.c, lmlec.c, lmlec_core.c, misc.c, misc.h misc_core.c** - Files from the software package that implements the Levenberg-Marquardt algorithm written by Manolis Lourakis

ResultsVerification – Directory containing files to compare and verify user results

- **CompareResults.m** – MATLAB script that compares user results to the approved set of results to verify installation
- **FinalSelectionResults.mat** – Data file with verified results for the Final Selection Process with the default settings
- **InterativeDownSelectResults.mat** – Data file with verified results for the Iterative Down-Select Process with the default settings

Figure 15.—Directory and File Structure of Software Release Package

5.0 Concluding Remarks

The Systematic Sensor Selection Strategy is a procedure that selects a set of sensors to support an objective such as improved health assessment. The methodology is flexible and is meant to be tailored to specific application needs which traditionally have been focused on health diagnostics. The procedure utilizes an optimization algorithm to select a set of sensors based on the system fault diagnostic approach while taking conflicting objectives such as cost, weight, and reliability into consideration. With the opposing constraints, several satisfactory sensor suite solutions are often the outcome. Experimentation is often necessary to explore the solution trade space to determine a final sensor suite solution.

This guide has been provided to assist in formulating a meaningful sensor selection framework. A simple S4 example application using the Commercial Modular Aeropropulsion Simulation System is described and provides a foundation from which more complex procedures can be developed. Corresponding software is provided that allows a user to exercise the procedure and to modify it for their needs.

References

1. Maul, W.A, Kopasakis, G., Santi, L.M., Sowers, T.S., Chicatelli, A.K., “Sensor Selection and Optimization for Health Assessment of Aerospace Systems,” AIAA 2007-2849, May 2007.
2. Santi, L.M., Sowers, T.S., Aguilar, R.B., “Optimal Sensor Selection for Health Monitoring Systems,” NASA/TM—2005-213955, AIAA-2005-4485, June 2005.
3. NASA Glenn Research Center Software Catalog, http://sr.grc.nasa.gov/, Last accessed June 2011.
4. Goldberg, D., “Genetic Algorithms in Search, Machine Learning and Optimization,” New York, Addison Wesley, 1989.
5. Frederick, D.K., DeCastro, J.A., Litt, J.S., “User’s Guide for the Commercial Modular Aero-Propulsion System Simulation,” NASA/TM—2007-215026, 2007.
6. More, J., “The Levenberg-Marquardt Algorithm, Implementation and Theory” in Numerical Analysis, G. A. Watson, editor, Lecture Notes in Mathematics 630, Springer-Verlag, 1977.
7. levmar: Levenberg-Marquardt Non-linear Least Squares Algorithms in C++, http://www.ics.forth.gr/~lourakis/levmar/, Last accessed August 2008.
8. Sowers, T.S., Kopasakis, G., Simon, D.L., “Application of the Systematic Sensor Selection Strategy for Turbofan Engine Diagnostics,” ASME Turbo Expo 2008, GT2008-50525, June 2008.
9. Sowers, T.S., Fittje, J.E., Kopasakis, G., Simon, D. L., “Expanded Application of the Systematic Sensor Selection Strategy for Turbofan Engine Diagnostics,” ASME Turbo Expo 2009, GT2009-59251, June 2009.
10. Simon, D.L., Bird, J., Davison, C., Volponi, A.J., Iverson, R.E., “Benchmarking Gas Path Diagnostic Methods: A Public Approach,” NASA/TM—2008-215271; ASME Turbo Expo 2008, GT2008-51360, June 2008.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01-02-2012	**2. REPORT TYPE** Final Contractor Report	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Systematic Sensor Selection Strategy (S4) User Guide	**5a. CONTRACT NUMBER** NNC06BA07B; NNC07E205T
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Sowers, T., Shane	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER WBS 645846.02.07.03.03.01
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) QinetiQ North America 21000 Brookpark Road, MS 500-AOS Cleveland, Ohio 44135	**8. PERFORMING ORGANIZATION REPORT NUMBER** E-16533
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSORING/MONITOR'S ACRONYM(S)** NASA
	11. SPONSORING/MONITORING REPORT NUMBER NASA/CR-2012-215242

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Categories: 01 and 12
Available electronically at http://www.sti.nasa.gov
This publication is available from the NASA Center for AeroSpace Information, 443-757-5802

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper describes a User Guide for the Systematic Sensor Selection Strategy (S4). S4 was developed to optimally select a sensor suite from a larger pool of candidate sensors based on their performance in a diagnostic system. For aerospace systems, selecting the proper sensors is important for ensuring adequate measurement coverage to satisfy operational, maintenance, performance, and system diagnostic criteria. S4 optimizes the selection of sensors based on the system fault diagnostic approach while taking conflicting objectives such as cost, weight and reliability into consideration. S4 can be described as a general architecture structured to accommodate application-specific components and requirements. It performs combinational optimization with a user defined merit or cost function to identify optimum or near-optimum sensor suite solutions. The S4 User Guide describes the sensor selection procedure and presents an example problem using an open source turbofan engine simulation to demonstrate its application.

15. SUBJECT TERMS
Optimization; Design optimization; Genetic algorithms

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** STI Help Desk (email:help@sti.nasa.gov)
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	46	**19b. TELEPHONE NUMBER** ***(include area code)*** 443-757-5802