Constructing an Efficient Self-Tuning Aircraft Engine Model for Control and Health Management Applications

Jeffrey B. Armstrong[1] and Donald L. Simon[2]

[1]*Vantage Partners, LLC, Cleveland, Ohio, 44135, U. S. A.*
jeffrey.b.armstrong@nasa.gov

[2]*NASA Glenn Research Center, Cleveland, Ohio, 44135, U. S. A.*
donald.l.simon@nasa.gov

ABSTRACT

Self-tuning aircraft engine models can be applied for control and health management applications. The self-tuning feature of these models minimizes the mismatch between any given engine and the underlying engineering model describing an engine family. This paper provides details of the construction of a self-tuning engine model centered on a piecewise linear Kalman filter design. Starting from a nonlinear transient aerothermal model, a piecewise linear representation is first extracted. The linearization procedure creates a database of trim vectors and state-space matrices that are subsequently scheduled for interpolation based on engine operating point. A series of steady-state Kalman gains can next be constructed from a reduced-order form of the piecewise linear model. Reduction of the piecewise linear model to an observable dimension with respect to available sensed engine measurements can be achieved using either a subset or an optimal linear combination of "health" parameters, which describe engine performance. The resulting piecewise linear Kalman filter is then implemented for faster-than-real-time processing of sensed engine measurements, generating outputs appropriate for trending engine performance, estimating both measured and unmeasured parameters for control purposes, and performing on-board gas-path fault diagnostics. Computational efficiency is achieved by designing multidimensional interpolation algorithms that exploit the shared scheduling of multiple trim vectors and system matrices. An example application illustrates the accuracy of a self-tuning piecewise linear Kalman filter model when applied to a nonlinear turbofan engine simulation. Additional discussions focus on the issue of transient response accuracy and the advantages of a piecewise linear Kalman filter in the context of validation and verification. The techniques described provide a framework for constructing efficient self-tuning aircraft engine models from complex nonlinear simulations.

1. INTRODUCTION

An emerging technology in the field of aircraft engine controls and health management is the inclusion of real-time on-board self-tuning models for the in-flight estimation of engine performance variations (Luppold, Roman, Gallops, & Kerr, 1989). Self-tuning engine models are comprised of an engine model and an associated tracking filter as shown in Figure 1. Here, the Aircraft Engine block also includes an on-board engine control computer (not shown). Real-time sensor and actuator information available within this control computer is used as inputs to the engine model. The engine model reflects engine aero-thermodynamic performance at both steady-state and transient conditions. Analytical modeling approaches, such as piecewise linear models and nonlinear component level models, are commonly applied. Recently, Volponi (2008) also demonstrated the merits of a hybrid modeling approach combining analytical (physics-based) and empirical (neural network) elements. The tracking filter, typically based on Kalman filter estimation concepts, is designed to automatically adjust tuning parameters within the engine model to enable the model to match the observed performance of the physical engine. The discrepancy between the physical engine parameters and the model variables is referred to as model mismatch. The tuning parameter adjustment is necessary to enable the model to account for the gas turbine engine performance variations caused by deterioration, wear, and fouling that turbomachinery will incur over time with usage.

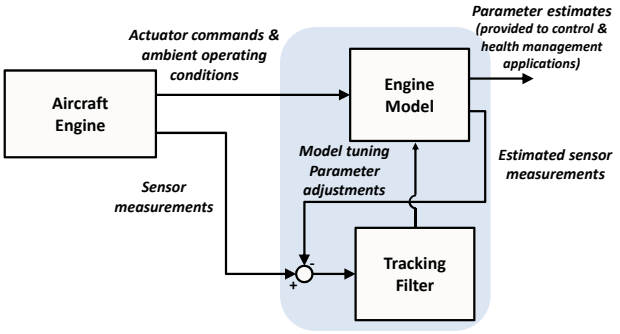

Figure 1. Self-tuning engine model architecture including an aircraft engine, an engine model, and a tracking filter to tune the model

One of the earliest investigations to consider the use of self-tuning engine model technology was under the NASA and Department of Defense led Performance-Seeking Control (PSC) program of the 1980's and 1990's (Shaw, Foxgrover, Berg, Swan, Adibhatla, & Skira, 1986), (Nobbs, Jacobs, & Donahue, 1992). The PSC program demonstrated multiple performance benefits achievable through in-flight propulsion control optimization including reduced fuel-burn, increased thrust, and increased component life (Gilyard & Orme, 1993). Central to the PSC design is the inclusion of a self-tuning engine model that provides the control system with real-time estimates of unmeasured engine performance parameters. Follow-on research efforts have continued to mature and advance aircraft engine model-based control technology (Dwyer, 1990), (Klaus & Kreiner, 2001), (Brunell, Bitmead, & Connolly, 2002).

In addition to control applications, self-tuning engine model technology also holds benefits for propulsion system health management. This includes estimating, trending, and forecasting the level of performance deterioration within the major rotating modules of the engine, and diagnosing faults that impact engine gas path performance (Gallops, Gass, & Kennedy, 1992), (Bushman & Gallops, 1992), (Armstrong & Simon, 2011). It is expected that the application of on-board self-tuning engine model technology within the aircraft engine industry will continue to increase. A future vision put forth by Behbahani, Adibhatla, and Rauche (2009) is to develop an integrated on-board self-tuning model-based engine controller architecture with control, diagnostic, and prognostic functionalities.

This paper is intended to serve as a guide for the development and implementation of self-tuning engine models for turbomachinery diagnostics, prognostics, and health management. It will cover the required information, design considerations, and design steps necessary to construct and implement such models. The focus of this work is specifically on piecewise linear self-tuning engine models with a Kalman filter as the tracking filter.

The remainder of this paper guides the reader through the design and implementation details of a self-tuning engine model. First, the overall design is discussed briefly to introduce the various requirements and components of such a model. Next, the generation of linear state space engine models and the design and setup of the Kalman filter is explained. Then, the implementation is explicated to aid the reader in constructing an efficient adaptation of this design. This is followed by the presentation of example results from the application of the technique to a turbofan engine simulation. Finally, a discussion of some design choices and model advantages is presented.

2. MODEL COMPONENTS

The self-tuning engine model outlined in this paper is composed of a number of modular components. Two major components comprise this design: a piecewise linear state-space engine model and an associated piecewise linear steady-state Kalman filter.

The input signals available to a self-tuning engine model are restricted to the following signals, which are also provided to the engine controller: engine actuator commands and sensor measurements from the engine. Additional inputs may consist of airframe sensed measurements, such as free stream conditions, that affect gas-path analyses.

The outputs of the self-tuning engine model will normally consist of three components: sensor estimates, tuning parameters, and unmeasured parameter estimates. In this architecture, the sensor estimates should be nearly identical to the actual sensed measurements since the model is "tuned" to match these measurements. Tuning parameters generated by the model are available as outputs. The parameters selected and applied as model tuning parameters depend on design decisions, but they often act as proxies for engine performance parameters, as is discussed later. Finally, the self-tuning engine model can produce estimates of unmeasured engine parameters. Since the engine model is physics-based, the estimates of unmeasured parameters are expected to approach the actual values. The estimates often include thrust, stall margins, and gas-path pressures and temperatures where physical measurement is impractical.

A piecewise linear state-space model forms the basis for modeling theoretical engine dynamics. This model is comprised of state space matrices and associated trim points, both of which are interpolated based on operating point. The piecewise linear model is created from a nonlinear aerothermal model, but offers some advantages over the more complex physics-based simulation. A piecewise linear model is usually less computationally intensive than its nonlinear equivalent, and the simpler structure allows for straightforward design of tuning solutions, a Kalman filter in this case.

Accompanying the piecewise linear model in this approach is a piecewise steady-state Kalman filter. This implementation uses gain matrices that are pre-computed and held constant as opposed to updating the gain matrices online while processing the input data. Similar to the state-space matrices, the Kalman gain matrices are also interpolated based on operating point. The Kalman filter produces tuning parameters that drive the difference between the actual sensor measurements and their respective estimates to zero.

When employing piecewise solutions, design considerations related to interpolation mechanisms, parameter scaling, degrees of freedom, and processing time are important. For accuracy reasons, it is often advantageous to consider interpolation based on multiple dimensions, as will be described. Efficient algorithms should be employed to exploit multidimensional data alignment and shared scheduling. Additionally, correction techniques applied internally within the model, as discussed later, require some supporting code infrastructure.

3. MODEL DATA COMPUTATION

Constructing this self-tuning model requires translating a nonlinear aerothermal model to an equivalent piecewise linear model and piecewise linear Kalman filter. The initial step is to linearize the nonlinear model at multiple operating points to generate a piecewise linear state-space model and an associated set of trim points. Once linearization is complete, the Kalman gain matrices are computed. The resultant trim points, state-space system matrices, and Kalman gains can then be used in the self-tuning engine model implementation.

3.1. Linearization

The nonlinear model of an aircraft engine can be represented by the following equations

$$\begin{aligned} \dot{x} &= f(x,u,h) \\ y &= g(x,u,h) \\ z &= g_z(x,u,h) \end{aligned} \tag{1}$$

where x and u represent the vectors of engine state variables and control command inputs, respectively. The vector h represents health parameters, such as efficiency or flow capacity, reflective of performance deterioration within the major modules of the engine. For given input values, the nonlinear functions f, g, and g_z generate the vectors of state derivatives $\dot{x}$, sensed engine outputs y, and unmeasured engine outputs z, respectively. By linearizing the engine model at a given operating point, the following state-space equations are obtained:

$$\begin{aligned} \dot{x} &= A\underbrace{(x-x_{trim})}_{\Delta x} + B\underbrace{(u-u_{trim})}_{\Delta u} + L\underbrace{(h-h_{ref})}_{\Delta h} \\ \dot{x} &= A\Delta x + B\Delta u + L\Delta h \end{aligned}$$

$$\begin{aligned} \underbrace{y-y_{trim}}_{\Delta y} &= C\underbrace{(x-x_{trim})}_{\Delta x} + D\underbrace{(u-u_{trim})}_{\Delta u} + M\underbrace{(h-h_{ref})}_{\Delta h} \\ \Delta y &= C\Delta x + D\Delta u + M\Delta h \end{aligned} \tag{2}$$

$$\begin{aligned} \underbrace{z-z_{trim}}_{\Delta z} &= F\underbrace{(x-x_{trim})}_{\Delta x} + G\underbrace{(u-u_{trim})}_{\Delta u} + N\underbrace{(h-h_{ref})}_{\Delta h} \\ \Delta z &= F\Delta x + G\Delta u + N\Delta h \end{aligned}$$

Here, A, B, C, D, F, G, L, M, and N are the state-space matrices reflecting system dynamics. The trim vectors, denoted by the subscript "*trim*," reflect the values of the state variables, commands, and measured and unmeasured outputs when the model is at steady-state (i.e., $\dot{x} = 0$) at the given operating point. Collectively, the trim vectors define what is referred to as a "trim point." The vector h_{ref} represents a reference health condition specified by the system designer. In Equation 2, parameter deviations relative to trim or reference conditions are denoted by the delta symbol (Δ).

The initial step in creating this self-tuning engine model is the computation of linear state-space models from the nonlinear model at multiple operating points. These operating points serve as the interpolation scheduling parameters in the piecewise linear model. Figure 2 shows a notional three-dimensional example of operating point specification using altitude, Mach number, and power setting as the scheduling parameters. The number of operating points and spacing between operating points, which does not have to be uniform, are design decisions left to the end user. In general, a denser grid of operating points will allow the piecewise linear model to more closely approximate the nonlinear model. However, that will increase memory storage requirements required for implementation. While the operating points generally reside within a standard flight envelope, it may be advantageous to select some operating points beyond the standard envelope (assuming the nonlinear engine model is operable and valid at these points). This expanded operating envelope functionality is necessary to enable the piecewise linear model to account for scenarios where the actual aircraft engine operates beyond normal expected operating conditions.

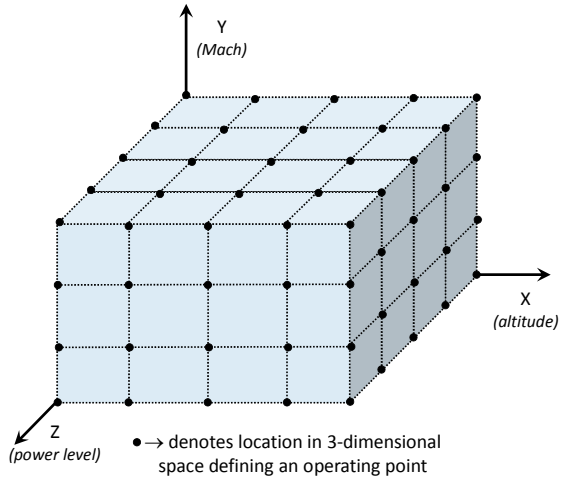

Figure 2. Example of three-dimensional piecewise linear model operating point scheduling

As described above, the piecewise linear model will use two related datasets: the trim points for the piecewise linear model and the state-space system matrices that model engine dynamics. Scheduling of the model and the generation of these datasets are discussed below.

3.1.1. Selection of Operating Points and Scheduling

The authors suggest selecting engine power level and flight conditions as scheduling parameters to enhance simulation accuracy. The rotational speed of the engine, specifically the fan speed for a two-spool turbofan, is considered a suitable proxy for power level. Alternatively, engine command parameters, such as power lever angle, may be used. However, since these variables do not necessarily reflect actual engine conditions, their usage may introduce inaccuracies during interpolation of trim points.

When scheduling on flight condition, Mach number and pressure altitude are suggested. While neither value is directly measurable, both can be easily computed from free stream and inlet pressure and temperature measurements. The use of Mach number and altitude will hide the nonlinearities that may be present in pressure and temperature changes over the flight envelope, allowing for more uniform steps in scheduling parameters.

The number of dimensions chosen for scheduling is a tradeoff between computational complexity and accuracy. The added accuracy that higher dimensional scheduling provides comes with both data storage and computational requirement penalties; significantly more mathematical operations are required as interpolation dimensions increase. As is discussed later, parameter correction may be used to minimize altitude effects. One may choose, therefore, to eliminate altitude as a scheduling parameter. However, parameter correction is imperfect, and there may be a resultant loss in accuracy.

Because two discrete data sets are generated (i.e., trim points and state-space system matrices) the interpolation scheduling may be separated if desired to decrease data storage requirements. Trim points require nearly an order of magnitude less storage space per operating point compared to state-space matrices. Furthermore, the dynamics of the system are not expected to change drastically; scheduling of the state-space matrices may not require the same grid density as the steady-state trim points. It may be advantageous to generate a denser or higher dimensional data set of trim points, while keeping the dynamics data set sparser or of a lower scheduling dimension (Brotherton, Volponi, Luppold, & Simon, 2003).

3.1.2. Trim Points Calculation

Generation of trim point data can be achieved using a steady-state engine model, as the trim points represent only the engine inputs and outputs without dynamics. After choosing operating conditions at which to compute trim points, the data can be generated in a hierarchical manner. For example, to generate the trim points in the three recommended dimensions discussed above, one would implement the algorithm below:

for each X in selected altitudes:

for each Y in selected Mach numbers:

for each Z in selected power levels:

Compute engine inputs, sensed outputs, and unmeasured outputs

Append trim point data sets

The trim points generally should be computed around the traditional flight envelope of the engine. Many models should be capable of computing conditions beyond the typical operating limits of the engine. Having data in regions beyond the expected operating envelope will help to protect against unforeseen flight conditions. Furthermore, the algorithm described will inherently generate unrealistic conditions. If, for example, the Mach numbers of interest vary from 0 to 0.8 and the altitude varies from sea level to a high cruise point, the algorithm will attempt to compute a condition of 0.8 Mach at sea level, which is likely an unrealistic condition. If generating the trim points from a physics-based model, this condition can most likely be computed with reasonable accuracy as altitude and Mach number translate into free stream and engine inlet conditions.

In situations where the algorithm cannot compute the desired steady-state condition, the operating point should be logged as a failure. After attempting to compute all conditions using the outlined algorithm, the failed conditions can be interpolated from those that were successfully calculated. It is suggested that the interpolation of failed conditions be performed linearly from nearest

successful computations along the power level axis. Extrapolation can be performed when a failed steady-state condition is not bounded by successful calculations. Extrapolated parameters, however, should not be relied on heavily for accuracy.

If the self-tuning model is to be applied to a variety of actual engines, it may be advantageous to carefully choose an engine performance level at which to compute the trim points. A fleet of engines will exhibit a statistical distribution of degradation of the rotating modules, which is manifested in changes to each module's efficiency and flow capacity (Sallee, 1978). The self-tuning engine model would benefit from being designed at a mean or median degradation condition with regards to the specific fleet to minimize the possible difference in performance variations across all engines.

3.1.3. Dynamics Calculation

Each of the matrices previously presented in Equation 2 must be calculated by perturbing their respective driving parameters via the nonlinear model. In this self-tuning implementation, the dynamics are captured by perturbing a given parameter within a modified version of the steady-state model that balances at a point described with non-zero state derivative conditions. When the perturbation is applied, the modified steady-state solver will attempt to calculate a balanced engine state, where the state variables are held constant, and the state derivatives are permitted to assume non-zero values. This condition is the instantaneous dynamic response of the engine after a perturbation is applied. The difference between the perturbed and unperturbed engine model outputs and state derivatives represent the dynamic behavior of the system for each given perturbation.

As stated earlier, it may be advantageous to compute the state-space matrices at different operating conditions than the trim points to address storage space concerns. If we again assume that the recommended three dimensions are used for scheduling state-space matrix interpolation, the algorithm will be:

for each X in selected altitudes:

for each Y in selected Mach numbers:

for each Z in selected power levels:

Compute state-space matrices

Append matrix data sets

The procedure for computing the matrices via perturbation of the inputs flows as follows:

Compute steady-state x_{trim}, y_{trim}, z_{trim}, and u_{rim} for current operating point

for each state in x:

Compute x, y, z for $(x_i+\delta x_i)$, where δx_i is the perturbation size for the i^{th} state, as x_p, y_p, z_p using "unbalanced" steady-state model

Compute state derivatives from the "unbalanced" state

Set column i of A to be $\dot{x}_p / \delta x_i$

Set column i of C and F to be $(y_p - y_{trim}) / \delta x_i$, $(z_p - z_{trim}) / \delta x_i$ *respectively*

for each input in u:

Compute x, y, z for $(u_i+\delta u_i)$, where δu_i is the perturbation size for the i^{th} actuator, as x_p, y_p, z_p using "unbalanced" steady-state model

Compute state derivatives from the "unbalanced" state

Set column i of B to be $\dot{x}_p / \delta u_i$

Set column i of D and G to be $(y_p - y_{trim}) / \delta u_i$, $(z_p - z_{trim}) / \delta u_i$ *,respectively*

for each health parameter in h:

Compute x, y, z for $(h_i+\delta h_i)$, where δh_i is the perturbation size for the i^{th} health parameter, as x_p, y_p, z_p using "unbalanced" steady-state model

Compute state derivatives from the "unbalanced" state

Set column i of L to be $\dot{x}_p / \delta h_i$

Set column i of M and N to be $(y_p - y_{trim}) / \delta h_i$, $(z_p - z_{trim}) / \delta h_i$ *respectively*

The health parameters are set nominally to the desired design point that describes a mean or median engine deterioration level. Note that the procedure above generates state-space systems in continuous form. Conversion to discrete-time state-space equivalents can be performed using an appropriate technique such as zero-order hold after generating the continuous time matrices.

The scale of the applied perturbations depends on the variables to be perturbed. The authors suggest using perturbations at least an order of magnitude less than a particular variable's steady-state value. Some trial and error experimentation is necessary with each parameter to determine an acceptable perturbation scale.

The algorithm outlined suggests that the small changes are performed in a single direction. However, improved accuracy might be gained by applying perturbations in two directions and computing an average effect from these two perturbations. Moreover, some perturbations in a given direction may not be possible if, for example, they exceed the capabilities of a given engine actuator. The implementer should take care to consider these special cases when performing linearization.

Similar to the trim point calculation procedure, one may encounter problematic operating conditions due to the algorithm's simplicity with respect to cycling through desired points. However, unlike the solution for failed convergence of trim points during calculations, interpolation and extrapolation is not recommended when dealing with matrices that cannot be reliably computed. Extrapolation can quickly lead to unrealistic dynamic behavior if great care is not taken, and the system dynamics are not expected to vary drastically across neighboring operating points. Instead, it is suggested that a nearby (in terms of scheduling parameters) successfully computed state-space matrix set should be used for the given failed calculation point.

3.2. Kalman Filter Design

The piecewise linear Kalman filter is the core of this self-tuning engine model. For each state-space system of the piecewise linear engine model, a corresponding Kalman gain matrix must also be computed. In this implementation, steady-state Kalman filtering is applied. This means that the Kalman gain matrix corresponding to each state-space system is invariant—it is pre-computed off-line, which helps to reduce computational requirements at runtime that would accompany the online calculation of the Kalman gain.

The system must be observable with respect to the number of available sensed engine measurements to construct this steady-state Kalman filter. The goal of tuning is to eliminate model mismatch due to the unknown performance characteristics of the engine. It is assumed that the model itself is theoretically correct, but the actual engine may exhibit behavior that differs from the theoretical model due to performance degradation, manufacturing variations, or other unknown variables. The health parameters, which theoretically quantify these performance differences, can be selected as engine tuning parameters. Additionally, since these parameters remain relatively constant in the short-term, they are usually measured over the course of a single flight.

The observability issue may prove problematic when dealing with aircraft gas turbine engines. Often times the number of sensors available for use with the self-tuning engine model is less than the number of health parameters present in the model. To overcome this underdetermined estimation problem, two techniques are suggested to transform the state-space matrices appropriately. If the health parameters are shifted to become states in our model in Equation 2, the system becomes:

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{h}\end{bmatrix} &= \begin{bmatrix}A & L\\ 0 & 0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta h\end{bmatrix} + B\Delta u\\ \Delta y &= \begin{bmatrix}C & M\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta h\end{bmatrix} + D\Delta u\\ \Delta z &= \begin{bmatrix}F & N\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta h\end{bmatrix} + G\Delta u\end{aligned} \tag{3}$$

Since engine performance deterioration evolves slowly in time, the health parameter states in Equation 3 are modeled without dynamics. Once the health parameters are augmented with the state variables, they can be estimated by applying a Kalman filter as long as the system is observable. However, a necessary condition for observability given the Equation 3 formulation is that there are at least as many measurements as health parameters (España, 1994). To construct a reduced-order state space system of appropriate dimension to enable Kalman filter formulation, consider a transformation matrix, V^*, that maps the health parameter vector, h, to a tuning vector of lower dimension, q, such that:

$$q = V^* h \tag{4}$$

An approximation for h based on q can be calculated using the pseudoinverse of V^*:

$$h = V^{*\dagger} q \tag{5}$$

Then, substituting Equation 5 into Equation 3 produces the following reduced-order state space system:

$$\begin{aligned}\begin{bmatrix}\dot{x}\\ \dot{q}\end{bmatrix} &= \begin{bmatrix}A & LV^{*\dagger}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta q\end{bmatrix} + B\Delta u\\ \Delta y &= \begin{bmatrix}C & MV^{*\dagger}\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta q\end{bmatrix} + D\Delta u\\ \Delta z &= \begin{bmatrix}F & NV^{*\dagger}\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta q\end{bmatrix} + G\Delta u\end{aligned} \tag{6}$$

The choice of the transformation matrix is a design decision to be made prior to constructing the Kalman gains. To allow for piecewise interpolation of the Kalman gain, the value of the transformation matrix must remain constant regardless of operating condition so that the definition of the tuning vector does not change based on operating point (Simon, Armstrong, & Garg, 2011).

The first technique for dimensional reduction of the health parameters is to select a subset of health parameters to use as tuning parameters, effectively assuming the excluded parameters remain constant. In this scenario, the elements

of the transformation matrix will be comprised of ones and zeros appropriately selected to map the selected subset of health parameters properly. Defining this subset of parameters is a design decision. Based on a theoretical error analysis, the optimal subset can be algorithmically selected (Simon, Armstrong, & Garg, 2011). While this technique preserves the definition of the selected health parameters, the excluded health parameters cannot be estimated and "smearing" effects may cause inaccuracies in the estimation of the selected subset of parameters (Simon, Armstrong, & Garg, 2011).

A second technique, referred to as "optimal tuner selection," can be employed to produce a transformation matrix that is a linear combination of all health parameters (Simon & Garg, 2010). This method involves optimizing the transformation matrix and the resultant definition of the tuning vector to minimize a desired estimation error. The error to be minimized is normally either the theoretical estimation error in a selection of unmeasured outputs (z), health parameters (h), or a combination of both. The selection of which errors to minimize is tailored to the intended usage of the self-tuning engine model. Because the value of the transformation matrix must not change with operating condition, a global optimization algorithm should be employed across the expected flight envelope (Simon, Armstrong, & Garg, 2011).

Once the dimensions of the tuning vector are reduced (if necessary) to make the estimation problem observable and the process and measurement noise covariance matrices are specified, the Kalman gain matrices are constructed at every operating point where a state-space system exists (Simon, Armstrong, & Garg, 2011). The general algorithm proceeds as follows, again, assuming three-dimensional interpolation:

for each X in selected altitudes:

for each Y in selected Mach numbers:

for each Z in selected power levels:

Transform or reduce the state-space system per Equation 6

Calculate the associated Kalman gain

Append transformed state-space and Kalman gain matrix data sets

Because of the mathematics inherent in computing the Kalman gain, most notably the algebraic Riccati equation, the gain matrix may not be calculable at some operating points (Zarchan & Musoff, 2005). In these cases, it is suggested that the entire state-space system be disposed of and replaced with the nearest (in terms of scheduling points) state-space system where a Kalman gain can be reliably computed. The failure to construct the Kalman gain may imply some mathematical stability issues with the state-space system used as a basis for the computation.

3.3. Data Storage

The algorithms presented in the previous section do not address data storage. The issue of storing the data generated is somewhat architecture and platform dependent, but some general guidelines are suggested to improve efficiency when using the resultant data.

For efficient interpolation, the authors found that it was beneficial to align the data in memory such that any given single vector or matrix in a data set containing multiple vectors or matrices exist in a single, congruent memory location. Placing each data set in a continuous memory block allowed for fast pointer arithmetic during interpolation. Access to vectors or matrices in the data set "in place" via pointer arithmetic avoided unnecessary penalties resulting from copying data to temporary storage during interpolation procedures.

The suggested memory layout of data described above should be considered when designing long-term storage of the data sets. If the data is initially generated in the proper format and saved to a permanent storage medium, the process of loading the data at runtime should result in an advantageous memory layout automatically. One suggestion is to store high-dimensional (3+ dimensions) data as concatenated two-dimensional arrays with scheduling as shown in Figure 3 for the three dimensional case. The advantage of this layout will become apparent when the interpolation implementation is discussed.

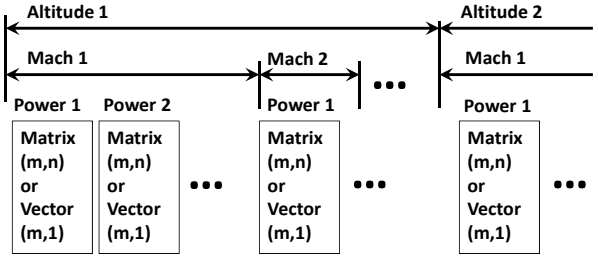

Figure 3. Suggested hierarchical data storage

4. Model Implementation

The implementation details surrounding the self-tuning engine model are application-specific. This section outlines an implementation that emphasizes efficiency and accuracy. The resultant design is appropriate for online, real-time applications and ground-based data analyses. The discussion will focus on a discrete-time implementation.

The overall self-tuning engine model design is illustrated in the block diagram in Figure 4. The self-tuning engine model requires sensed engine measurements (y) and actuator

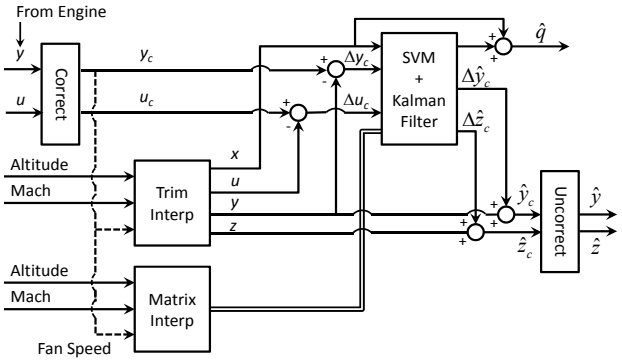

Figure 4. Self-tuning engine model overview

inputs (u). The model produces sensor estimates ($\hat{y}$), unmeasured engine parameter estimates ($\hat{z}$), and tuning parameters ($\hat{q}$). The sensor estimates should match the sensed measurements. The tuning parameters will be transformable back into estimates of engine performance parameters (Simon & Garg, 2010).

The individual components of this self-tuning engine model are explained below. The details of the implementation are independent of computing language and hardware platform.

4.1. Parameter Correction

For aircraft engine applications, the use of corrected parameters within the self-tuning model is encouraged to improve accuracy and to reduce the number of operating points included in the piecewise linear design. Parameter correction is used to minimize the effects of atmospheric variations due to temperature and pressure (Volponi, 1998). The inlet total pressure and temperature sensors are employed to normalize parameters with respect to standard day sea level static conditions. In a gas turbine engine, it is likely that the only actuator input requiring correction will be a fuel flow command. The sensor measurements must also be corrected prior to use. The tuning parameters, which are considered to be proxies for the health parameters and are assumed to be independent of altitude and Mach number, will not require correction. The correction of trim values and system matrices is performed during linearization. Within the block shown in Figure 4, all values remain in corrected form, including inputs, to improve accuracy.

4.2. Kalman Filter Implementation

The filter in this self-tuning engine model is a steady-state Kalman filter implementation. The Kalman gain, state-space matrices, and trim vectors are delivered to this module via the interpolation routines, as will be discussed. This implementation uses a discrete-time form of the Kalman filter. A block-diagram in Figure 5 outlines the structure of this Kalman filter. Here, k represents the discrete time

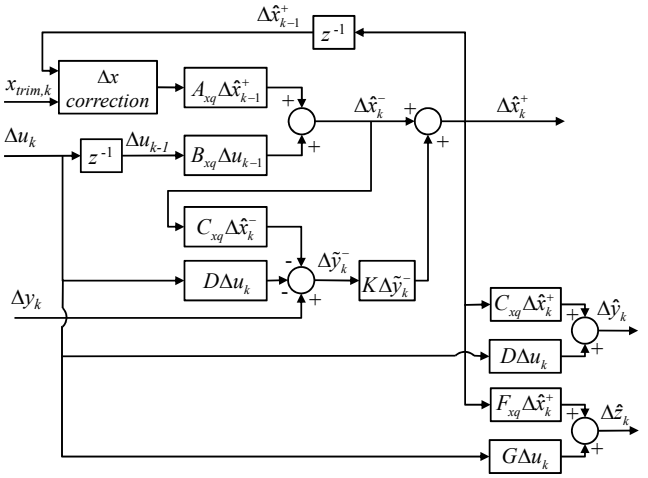

Figure 5. Block diagram of Kalman filter

index and the matrices and vectors have been augmented per Simon and Garg (2010) to include tuning parameters in the state vector, Δx, accompanied by proper state-space matrix modifications. Here, the "+" and "-" superscripts denote Kalman filter *a posteriori* and *a priori* estimates, respectively, $\Delta \tilde{y}^-_k$ is the residual between the sensed and estimated measurement vector, and z^{-1} is the unit sample delay.

The implementation of a piecewise linear Kalman filter does pose unique implementation requirements. The Kalman filter, which is a recursive estimator, relies on state estimates calculated at the previous time step. However, on each time step in the piecewise linear implementation, the trim vectors, state-space matrices, and Kalman gain matrix are interpolated, and all are likely to shift from the previous time point. Therefore, the *a posteriori* state estimate calculated the previous time step will reflect a deviation relative to the state trim vector applied during the previous time step. Prior to use on the current time step, the *a posteriori* state estimate must be updated to reflect the change in the trim values, as shown below:

$$\Delta x'_{k-1} = \Delta x_{k-1} + (x_k - x_{k-1}) \tag{7}$$

In the above equation, the expression inside the parenthesis reflects the change in trim values from one time step to the next. Applying this adjustment ensures that deviations from trim are relative to the trim point applied at time step k, as opposed to the trim point previously applied at time step k-1. For additional details on the formulation of the Kalman filter, readers are referred to Simon & Garg, 2010.

4.3. Interpolation Technique

Self-tuning engine model computational efficiency is highly dependent on the interpolation technique employed within the model. Therefore, to lessen overall model computational requirements the interpolation procedure

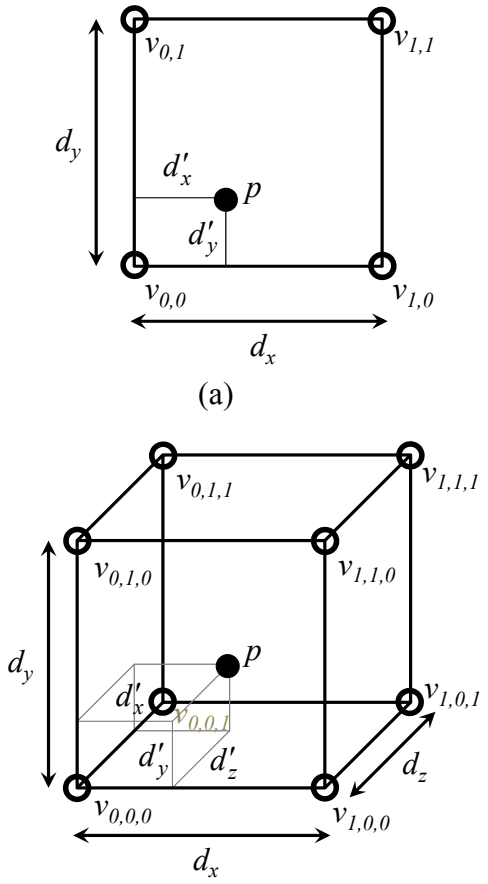

Figure 6. (a) Two- and (b) Three-dimensional interpolation

should employ rapid and efficient techniques. The earlier section, focusing on the storage of the trim vectors, state-space matrices, and Kalman gain matrix data sets, discussed the proper layout of data in memory for efficient interpolation. Once this data is stored in memory, efficient interpolation methods can be used across the multiple dimensions.

When working in multiple dimensions, linear interpolation requires an expansion. Consider the two simple interpolation cases in Figure 6. The point, p, at which to interpolate can be projected onto each of the interpolation axes, as the figure shows, and the point is bounded by either four or eight schedule points for the two- and three-dimensional cases, respectively. For each axis (i.e., dimension), weights are computed that represent the normalized distances from each bounding point on that axis. For a single axis, assuming d is the distance between two schedule operating points and d' is the distance between the projection of p onto the axis and the schedule point that precedes p, two weights can be calculated as:

$$\begin{aligned} w_0 &= 1 - (d'/d) \\ w_1 &= d'/d \end{aligned} \tag{8}$$

The weight w_0 represents the contribution of the schedule point v_0 that precedes the projection of p, while the weight w_1 is the contribution of the schedule point v_1 that follows the projection of p. The weight pairs are computed for each axis. This procedure will yield two pairs of weights for the two-dimensional interpolation case and three weighting pairs for the three-dimensional case. The interpolated value for the two-dimensional case (bi-linear interpolation) is then defined as:

$$\begin{aligned} x_p = {} & w_{x,0} w_{y,0} v_{0,0} + w_{x,1} w_{y,0} v_{1,0} + \\ & w_{x,0} w_{y,1} v_{0,1} + w_{x,1} w_{y,1} v_{1,1} \end{aligned} \tag{9}$$

Likewise, the interpolated value for three dimensions (tri-linear interpolation) would be:

$$\begin{aligned} x_p = {} & w_{x,0} w_{y,0} w_{z,0} v_{0,0,0} + w_{x,1} w_{y,0} w_{z,0} v_{1,0,0} + \\ & w_{x,0} w_{y,1} w_{z,0} v_{0,1,0} + w_{x,1} w_{y,1} w_{z,0} v_{1,1,0} + \\ & w_{x,0} w_{y,0} w_{z,1} v_{0,0,1} + w_{x,1} w_{y,0} w_{z,1} v_{1,0,1} + \\ & w_{x,0} w_{y,1} w_{z,1} v_{0,1,1} + w_{x,1} w_{y,1} w_{z,1} v_{1,1,1} \end{aligned} \tag{10}$$

The calculations described above require determining the bounding points along each axis from the schedules. A simple search for the desired index is usually sufficient, but some efficiency gains can be realized by using a bracketed, binary search with memory of the last successful search between requests. A binary search technique can rapidly search through a sorted array, such as our scheduling axes, with a worst case $O(\log n)$* performance (Knuth, 1997). An additional improvement can be gained by storing the index of the lower bounding point for each axis at each time step. Because the change in operating point on each time step is likely to be relatively small, the previously used index on each axis can be rapidly checked to see if it is still applicable rather than performing a binary search on every time step. The performance on the majority of time steps would remain at $O(1)$, and only the applied weights would require recalculation.

Another way to improve efficiency is to limit the number of schedule searches and subsequent weight calculations based on the sharing of scheduling axes. The trim point vectors would all share one set of scheduling axes, while the state-space matrices and Kalman gains would use a less dense set. Therefore, only two passes of weight calculations would be necessary, one for trim vectors and another for matrices. The weights could then be shared between trim point vector interpolations when applying the data set for each parameter

* Shown in Big O notation indicating that worst case computational time grows proportional to $\log n$, where n is the number of grid points on the scheduling axis.

to either equation 9 or 10. Similarly, the matrix weights could be shared for all the state-space matrix data sets and the Kalman gain data set. This approach decreased the number of schedule searches by a factor of five for this three-dimensional model design.

As explained earlier, the memory layout of the data can lead to dramatic improvements in performance. Matrix interpolation may seem costly, and this implementation is suggesting a minimum of five matrix interpolations per time step, with each involving a considerable number of products to be computed. To minimize the impact of the large number of products necessary, one method may be to exploit "single instruction, multiple data," or SIMD, instructions that are conveniently available on many modern central processing units, including modern embedded processors (ARM, 2010-2011), (Intel Corporation, 1997-2012), (International Business Machines Corporation, 2006). Rather than focus on processor-specific capabilities, the use of the Basic Linear Algebra Subprograms, or BLAS, library is suggested (Lawson, Hanson, Kincaid, & Krogh, 1979). Some modern interpreted languages will use these procedures internally, and modern optimizing compilers can often detect and use these procedures in a manner transparent to the designer. Examining equations 9 and 10, one may notice that the BLAS routines "**AXPY*," which multiplies a vector by a scalar (our weights) and adds the product to another vector, can be applied multiple times to calculate the desired interpolated value (BLAS, 2011). Using such routines eliminates the element-by-element multiplication that might be used naively, allowing the SIMD capabilities of the processor to be used.

The continuous memory locations, which had been suggested earlier, allow for further efficiency improvement. If each data set is held in a single memory block, each matrix or vector can be accessed "in-place" rather than copying or extracting the matrix or vector elements to an appropriately sized array prior to weighting each individual point. In a lower level language, pointer arithmetic can be exploited to specify the location in memory of an individual matrix or vector within a data set. By employing these efficiency gains, the computational costs of performing multidimensional interpolation of matrices and vectors are minimized without sacrificing accuracy.

5. EXAMPLE RESULTS

For evaluation purposes, this self-tuning engine model is compared against the nonlinear engine model upon which it is based. To illustrate the capabilities of the suggested design, an appropriate self-tuning engine model has been derived from the Commercial Modular Aero-Propulsion System Simulation 40k, or C-MAPSS40k, a nonlinear aerothermal model that simulates a 40,000lbf-class turbofan engine (May, Csank, Lavelle, Litt, & Guo, 2010). The self-tuning engine model was designed to reflect an engine at

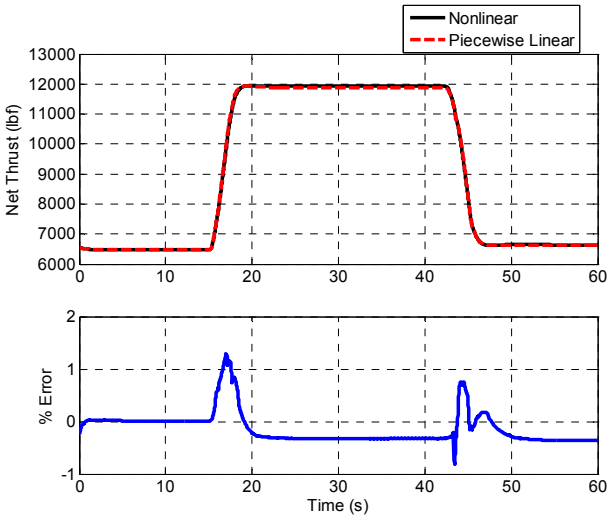

Figure 7. Thrust calculations using nonlinear and piecewise linear models at 25,000 ft and 0.55 Mach

50% of useful life remaining. The piecewise linear state-space model and Kalman filter have been designed as discrete-time models using the same time step size as the nonlinear model serving as the comparison basis.

First, the standalone piecewise linear model (without tuning) is compared with the nonlinear model to determine the accuracy of the linearization and effectiveness of the interpolation algorithms. Figure 7 shows the thrust estimate for the piecewise linear model as compared to the nonlinear C-MAPSS40k model running at the mean degradation design point for a rapid power increase followed by a subsequent decrease to the original power level. The altitude and Mach number conditions tested lie between schedule points, meaning the interpolation algorithm is being exercised in this example. The percent error graph, which examines the point-by-point difference in thrust between the piecewise linear model versus the nonlinear model, shows a noticeable increase in the residuals during transients.

The self-tuning model provides much of its advantage for engines that operate away from the model degradation design point. Under the same transient situation with tuning enabled, the self-tuning model should be able to maintain accuracy when applied to engines that are not represented by the mean performance level. Figure 8 illustrates the accuracy of a self-tuning engine model when estimating the unmeasured combustor exit temperature for an ideal (new) engine and an end-of-life engine. The combustor exit temperature has been chosen for comparison because it experiences significant shift as engine performance degrades. In this example, the nonlinear engine model serves as the "truth" model and its sensed outputs and actuator commands are provided as inputs to the self-tuning engine model. The figure shows the outputs of the nonlinear

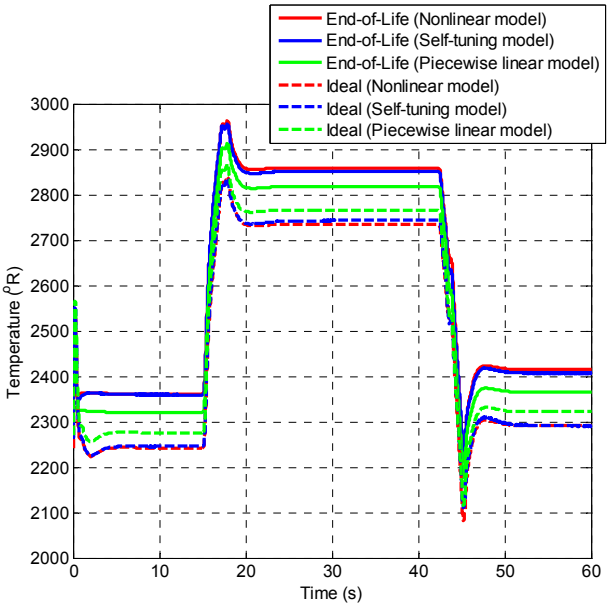

Figure 8. Unmeasurable combustor exit temperature estimates for an off-design engine

engine model (red), the self-tuning engine model (blue), and the piecewise linear model without tuning (green). Here, the estimate of combustor exit temperature produced by the self-tuning engine model exhibits good matching with the nonlinear model at these two extreme performance ranges. Conversely, the piecewise linear model, which does not have self-tuning functionality, is unable to match the nonlinear model as well.

The self-tuning engine model is considerably simpler than the full nonlinear model. While C-MAPSS40k is capable of running faster than real-time on modern consumer-grade computing hardware, the self-tuning engine model exhibits better computational performance due to its efficient interpolation algorithm and the inherent simplicity of its linear design. Compared to the nonlinear model, the self-tuning engine model runs approximately an order of magnitude faster than the nonlinear model on the same computer.

6. Discussion

The main advantage of the self-tuning engine model is its ability to eliminate the mismatch between the theoretical model and the actual engine. Because of this feature, on-board implementations of such models may be desirable for a variety of reasons, including control system integration, on-board model-based engine diagnostics, and simple informational purposes. However, this model loses much of its utility if it is unable to accurately estimate parameters during transient operation.

The accuracy during transients is related to a number of design choices made during the generation of the self-tuning model's data. The schedule density of the piecewise linear model must always be considered during the design phase. Comparison of the piecewise linear model itself against its nonlinear basis model during simple transients represents a "best case" accuracy that can be achieved by the subsequent self-tuning engine model. Often times, adjustments to the perturbations during linearization must be made to better improve dynamic accuracy. Additionally, some highly nonlinear parameters, such as stall margins, may not lend themselves to linearization inherently; these parameters appear to result in large mismatches even when great care is taken to improve accuracy.

Transient behavior of the tuning parameters within the Kalman filter is normally adjusted via modification of the process noise. Scaling of the noise has been shown to accelerate or decelerate the response of tuning parameters to engine transients (Simon, Armstrong, & Garg, 2011). Furthermore, ongoing research suggests that numerical factors related to the globally optimal tuner selection strategy may produce tuner transformation matrices that have undesirable transient properties. The current optimization algorithm does not consider the transient behavior of resultant Kalman filters, only the steady-state errors. Depending on the models involved, this issue may or may not be encountered during implementation of a self-tuning engine model.

The advantage of this self-tuning engine model is the relative simplicity of its design, among other benefits. The structure of the Kalman filter, although this model uses a piecewise version, lends itself to well-known verification and validation, or V&V, procedures (Schumann & Liu, 2007). For online implementations or control system integration, the ability to perform V&V on this self-tuning engine model using accepted processes is advantageous when comparing against alternatives such as a full nonlinear, physics-based online model.

7. Conclusion

A piecewise linear Kalman filter has been proposed as a self-tuning engine model solution. The well-understood Kalman filter algorithm combined with an efficient implementation make this piecewise solution an attractive candidate for resolving differences between theoretical, physics-based models and actual engine hardware.

The two outputs of the self-tuning engine model, tuning parameters and unmeasured parameter estimates, can be exploited for a variety of purposes. Estimates of these parameters, which could be employed in either ground-based or on-board solutions, could be used for performance trending and assisting in current engine health management programs. Such trending information is also useful for engine diagnostic algorithms by allowing these conceptual algorithms to discern between normal engine degradation and possible faults. The estimated parameters, while useful

for informational purposes alone, introduce the prospect of advanced parameter synthesis and control algorithms, including controlling directly on thrust. Possible efficiency gains may be realized through the accurate estimation of engine-specific operational limits, providing the opportunity to relax generally conservative stall margin, temperature, and pressure limits.

ACKNOWLEDGEMENT

The research associated with this work was performed under the Vehicle Systems Safety Technologies project as part of the National Aeronautics and Space Administration's Aviation Safety Program.

NOMENCLATURE

A, B, C, D, F, G, L, M, N	Linear state-space system matrices
d, d'	Distance in interpolation calculations
h	Health parameter vector
q	Engine tuning parameter vector
U	Engine control input vector
*V**	Transformation matrix mapping health parameters to engine tuning parameters
X,Y,Z	Interpolation scheduling axes
f	Nonlinear function of engine state derivatives
g	Nonlinear function of engine outputs
u	Control inputs
v	Value placeholder for interpolation
w	Weighting for interpolation
x	Engine state vector
y	Engine sensed measurement vector
z	Engine unmeasured parameter vector
z^{-1}	Unit sample delay
Δ *prefix*	Deviation from trim value
δ *prefix*	Perturbation value

Subscripts

c	Corrected value
k	Discrete time index
ref	Health parameter reference vector
trim	Trim vector

Superscripts

†	Pseudoinverse
-	*a priori* Kalman filter estimate
+	*a posteriori* Kalman filter estimate

Diacritical Marks

ˆ	Estimated value
~	Residual between estimated and sensed measurement vector

REFERENCES

ARM. (2010-2011). ARMv6 SIMD Instruction Intrinsics. In *ARM Compiler Toolchain Version 4.1: Compiler Reference* (pp. A-1 - A-67). ARM.

Armstrong, J. B., & Simon, D. L. (2011). Implementation of an Integrated On-Board Aircraft Engine Diagnostic Architecture. *47th AIAA Joint Propulsion Conference.* San Diego, CA.

Behbahani, A., Adibhatla, S., & Rauche, C. (2009). Integrated Model-Based Controls and PHM for Improving Turbine Engine Performance, Reliability, and Cost. *45th AIAA Joint Propulsion Conference.* Denver, CO.

BLAS. (2011). Retrieved April 30, 2012, from http://www.netlib.org/blas/

Brotherton, T., Volponi, A., Luppold, R., & Simon, D. (2003). eSTORM: Enhanced Self Tuning On-board Real-time Engine Model. *Proceedings of the 2003 IEEE Aerospace Conference.* Big Sky, MT.

Brunell, B., Bitmead, R., & Connolly, A. (2002). Nonlinear Model Predicive Control of an Aircraft Gas Turbine Engine. *Proceedings of the IEEE Conference on Decision and Control*, *4*, pp. 4649-4651. Las Vegas, NV.

Bushman, M. A., & Gallops, G. A. (1992). In-Flight Performance Diagnostic Capability of an Adaptive Engine Model. *28th AIAA Joint Propulsion Conference.* Nashville, TN.

Dwyer, W. J. (1990). *Adaptive Model-Based Control Applied to a Turbofan Aircraft Engine.* Cambridge, MA: Massachusetts Institute of Technology.

España, M. D. (1994). Sensor Biases Effect on the Estimation Algorithm for Performance-Seeking Controllers. *ASME Journal of Propulsion and Power , 10*, 527-532.

Gallops, G. W., Gass, F. D., & Kennedy, M. H. (1992). On-Board Condition Management for Aircraft Gas Turbines. *37th ASME International Gas Turbine and Aeroengine Congress and Exposition.* Cologne, Germany.

Gilyard, G. B., & Orme, J. S. (1993). *Performance Seeking Control: Program Overview and Future Directions*. NASA.

Intel Corporation. (1997-2012). *Intel Architecture Instruction Set Extensions Programming Reference.*

International Business Machines Corporation. (2006). *PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Enviornments Manual.* Hopewell Junction, NY.

Klaus, L., & Kreiner, A. (2001). Model Based Control Concepts for Jet Engines. *ASME Turbo Expo 2001*. New Orleans, LA.

Knuth, D. (1997). The Art of Computer Programming, Volume 3: Sorting and Searching. Reading, Massachusetts: Addison-Wesley.

Lawson, C. L., Hanson, R. J., Kincaid, D., & Krogh, F. T. (1979). Basic Linear Algebra Subprograms for

FORTRAN Usage. *ACM Transactions on Mathematical Software* , 308-323.

Luppold, R. H., Roman, J. R., Gallops, G. W., & Kerr, L. J. (1989). Estimating In-Flight Engine Performance Variations Using Kalman Filter Concepts. *25th AIAA Joint Propulsion Conference.* Monterey, CA.

May, R. D., Csank, J., Lavelle, T. M., Litt, J. S., & Guo, T. H. (2010). A High-Fidelity Simulation of a Generic Commercial Aircraft Engine and Controller. *46th AIAA Joint Propulsion Conference.* Nashville, TN.

Nobbs, S. G., Jacobs, S. W., & Donahue, D. J. (1992). Development of the Full-Envelope Performance Seeking Control Algorithm. *28th AIAA Joint Propulsion Conference.* Nashville, TN.

Sallee, G. (1978). *Performance Deterioration Based on Existing (Historical) Data – JT9D Jet Engine Diagnostics Program.*

Schumann, J., & Liu, Y. (2007). Tools and Methods for the Verification and Validation of Adaptive Aircraft Control Systems. *2007 IEEE Aerospace Conference.* Big Sky, MT.

Shaw, P., Foxgrover, J., Berg, D. F., Swan, J., Adibhatla, S., & Skira, C. A. (1986). A Design Approach to Performance Seeking Control. *22nd AIAA Joint Propulsion Conference.* Huntsville, AL.

Simon, D. L., & Garg, S. (2010, March). Optimal Tuner Selection for Kalman Filter-Based Aircraft Engine Performance Estimation. *Journal of Engineering for Gas Turbines and Power , 132*.

Simon, D. L., Armstrong, J. B., & Garg, S. (2011). Application of an Optimal Tuner Selection Approach for On-Board Self-Tuning Engine Models. *Proceedings of the ASME Turbo Expo 2011* .

Volponi, A. (2008). *Enhanced Self Tuning On-Board Real-Time Model (eSTORM) for Aircraft Engine Performance Health Tracking.* National Aeronautics and Space Administration.

Volponi, A. J. (1998). Gas Turbine Parameter Corrections. *ASME International Gas Turbine and Aeroengine Congress and Exposition.* Stockholm, Sweden.

Zarchan, P., & Musoff, H. (2005). Fundamentals of Kalman Filtering: A Practical Approach. AIAA.

BIOGRAPHIES

Jeffrey B. Armstrong is a Controls Engineer working as a contractor at the NASA Glenn Research Center in Cleveland, Ohio. He holds bachelor (2000) and master of science (2002) degrees in aerospace and mechanical engineering from Case Western Reserve University and has years of experience designing and implementing numerical simulations. In the past he has worked in microgravity research, rocket trajectory validation and verification, and enterprise medical software fields. Currently Jeffrey participates in air-breathing propulsion diagnostics research.

Donald L. Simon is a Controls Engineer at the NASA Glenn Research Center. He holds a B.S. degree from Youngstown State University (1987) and a M.S. degree from Cleveland State University (1990), both in electrical engineering. During his career he has focused on the development of advanced control and health management technologies for current and future aerospace propulsion systems. Mr. Simon's specific research interests are in aircraft gas turbine engine performance diagnostics and performance estimation. He currently leads the propulsion gas path health management research effort ongoing under the NASA Aviation Safety Program, Vehicle Systems Safety Technologies Project.