Proceedings of ASME Turbo Expo 2011
GT2011
June 6-10, 2011, Vancouver, British Columbia, Canada

GT2011-45565

A GAS TURBINE ENGINE MODEL OF TRANSIENT OPERATION ACROSS THE FLIGHT ENVELOPE

Reza Rezvani, Metin Ozcan, Brian Kestner, Jimmy Tai, Dimitri N. Mavris
Aerospace Systems Design Laboratory (ASDL)
Georgia Institute of Technology
Atlanta, GA

Richard Meisner, Steven Sirica
Pratt & Whitney
East Hartford, CT

ABSTRACT

This paper introduces a method to create engine transient models that retain the fidelity and non-linearity of complex models as well as simplicity and speed of lower fidelity linearized models. The method is based on the design of experiments (DOE) and neural network methodology to create an analytic non-linear model of engine transient operation which has the potential to be used in on-board and off-board applications. The feed forward neural net models were created for a high fidelity model of high bypass turbofan engine (truth model). The performance of the neural net models was verified against the truth model. The verification results showed good agreement between the output of the neural net models and the truth model. Initial investigations also showed a significant reduction in the model execution time.

INTRODUCTION

Engine transient models have many applications in design, development, and operation of gas-turbine engines. Since some of the important engine performance parameters such as thrust, airflow, and stall margin cannot be measured directly, an engine model is required to provide knowledge of these performance parameters based on the available data. To that end, an observer is required to estimate the engine performance parameters from known (measured or sensed) data. One of the requirements in parameters estimation is the need for the model execution speed being comparable to the rate at which the parameters are changing in the engine [1]. Reference [2] provides a detailed description of the dynamic engine models and their on-board and off-board applications in different fields such as the control development bench, integrated flight/propulsion control evaluation, embedded software for flight systems, system model within model based control and engine/control model in flight simulation. Numerous observation techniques are provided in the literature to provide an accurate estimation of the unmeasured parameters. The most well-known approach is based on the use of Kalman filter and its extensions. Most of Kalman filters used in the parameter estimation are linear; therefore, linearization of the engine model around a single or multi operating points is needed [3].

Engine transient operation is highly non-linear and to accurately model it, main and secondary effects such as torque balance, rotor inertia, flow dynamics, acoustics, heat soakage effects and blade tip clearance are considered in the engine aerothermal models. Sanghi et al. provided an extensive overview of the engine thermodynamic simulation from 1950 to 2000 [4]. Fawke and Saravanamuttoo created an aerothermal engine model for a turbofan engine with inclusion of the rotor inertia and components heat capacity for the steady-state and transient phases [5]. Stamatis et al. [6] used the methodology provided in reference [2] to model a turbofan engine and compare the execution time for various parameters variation (such as tolerance, implicit or explicit implementation and number of fixed updates per time step).

In general, high fidelity aerothermal models are computationally intensive, which may preclude the direct application of such models in applications where execution time is an important factor. To reduce the execution time and increase the speed of dynamics engine models, various simplified models have been proposed. Transfer functions and state-space models are among these simplified models where they are linearized around a single or multiple operating points (base-points). These models have good accuracy around the base-points, but as the operating condition deviates from these points, the linear interpolation between the neighboring points starts to deteriorate the accuracy. The accuracy of these models as well as their execution time is a function of the number of the base-points. Lichtsinder [7] proposed a method to retain the non-linearity of the transient engine model using a quasi-linear approach to create a transfer function. A fast model for a micro-jet turbine engine was developed using the Novel Generalized Describing Function (NGDF) to decouple the coupled non-linear ordinary differential equations from the algebraic equations using characteristic times and constant coefficients. The results showed good agreement between the closed-loop simulation of the non-linear model and NGDF model when the input signal is of the form of generalized quasi-polynomial.

Another promising method is the use of surrogate models. They can represent the complex nature of the engine transient operation with relatively simple mathematical models. The simple structure of surrogate models makes them potentially fast and accurate. They can also provide a differentiable analytical form of the engine transient model which is the advantage of the surrogate models over the linearized models which are not differentiable at base-points.

Chiras et al. [8] used the data obtained from Rolls Royce Spey MK202 engine using inverse repeat maximum length binary sequences (IRMLBS) and multi-sine input signals with the amplitude of the 10% of the fuel flow to show that a linear frequency-domain model is adequate for small-signal dynamics of the engine. They also showed the ability of a second order polynomial-based model (NNARMAX model) to capture the non-linearity of the engine's large-signal dynamics. In another work, Chiras et al. [9] used recurrent feed-forward neural net in NNARMAX concept to model the large-signal nonlinear behavior of the same engine mentioned above The data were acquired by perturbation of the fuel input (multi-sine or IRMLBS) around a number of steady-state points. The results showed the superiority of the NNARMAX model to the linear model for large input signal variations.

Luppold et al. [10] offered engine diagnosis methods (STORM and e_STORM) that uses actual engine data to reduce the model uncertainty and increase the parameter estimation accuracy and fault diagnosis capability. They used a linear state variable model (SVM) and a Kalman filter to estimate the engine health parameters though STORM method and to eliminate engine model mismatches they used data coming from the engine during the fligh to sequentially build an on-line neural network model to compensate for those errors. The new modified method is called e-STORM. To make the model robust to flight condition variations, they used flight segments and flight envelope sections to train the neural nets. Shankar and Yedavalli [11] used a radial basis function neural network (RBFN) to capture the engine non-linearity and degradation of a two-spool turbofan engine. They created a hybrid model with combinations of Kalman filter and an RFBN model to compensate for the shortcomings of the Kalman filter. The RFBN is trained on-line with a growing-pruning strategy to keep its structure optimized. They showed the estimation improvement using the hybrid model against using the Kalman filter alone for new and degraded engines. The importance of training data for RBFN to improve the estimation accuracy was also emphasized by them.

Combination of fuzzy logic and neural network, results in elimination of the weaknesses of each approach and taking advantages of both of them. The neural network approach has good learning capability while fuzzy logic can handle uncertainties. Using the neural network can help to find the optimized fuzzy set membership function. A fuzzy relation-based neural network (FRNN) [12] along with the genetic algorithm was used to find the near optimal point for fuzzy neural network (FNN) membership function optimization.

An area that has not been addressed properly in the literature is a methodology to generate a good set of training data for the whole operating range and the flight envelope. A combination of such a methodology with a method to create and train neural network can provide more accurate estimation of the observed parameters.

This paper describes a methodology to create a set of feed-forward neural network models for the transient operation of a high-bypass ratio turbofan engine for the maximum range of the fuel flow variation across the flight envelope.

The rest of the paper is organized as follows: First a description of the gas turbine engine transient model and application of neural network in their modeling is provided. After that, the process to create a data set to train the neural network and estimate the outputs of interest from the neural network models will be described and the results will be discussed. The paper ends with conclusion and suggested future work.

GAS TURBINE NEURAL NETWORK MODEL

The usage, structure, and training of a generic neural network to represent a nonlinear function are described in detail in Appendix A. The specific application of this methodology to gas turbine engine modeling is described below.

The structure of the gas turbine neural network is analogous to a state variable model where the state derivatives and system outputs are nonlinear functions of the initial state, control inputs, and environmental parameters. The steady-state neural network captures the nonlinear relationship between system outputs and the current state, control inputs, and environmental parameters. The transient neural network is a combination of two neural networks: state derivative and system output. The first network in the transient neural network is used to determine the states derivative as shown in Eq. (1). The first neural network maps the current states, control inputs, engine health parameters, and environmental parameters to the states derivative. The states potentially may include shaft speeds, metal temperatures, or blade tip clearances. Engine control inputs could include fuel flow, exhaust nozzle area, or guide vane position. Engine health parameters are scalars (component flows and efficiencies) used to either match the raw data or indicate the health of a component. The environmental parameters could include altitude, Mach number, and ambient temperature. As shown in Eq. (2), the derivative of each state is then explicitly integrated over a specified time-step. These new state values as well as the current control inputs and environmental parameters are inputs to a neural network mapping state values, control inputs, and environmental parameters to the system responses as shown in Eq. (3).

$$\dot{x}_t = f(x_t, u_t, p_t, z_t) \tag{1}$$

$$x_{t+1} = x_t + \int_t^{t+1} \dot{x}_t \cdot dt \cong x_t + \dot{x}_t \cdot \Delta t \tag{2}$$

$$y_{t+1} = g(x_{t+1}, u_{t+1}, p_{t+1}, z_{t+1}) \tag{3}$$

Health parameters, although generally considered a state, are modeled as steady-state and therefore no derivatives for these parameters have to be tracked. From a neural network simulation perspective, health parameters are analogous to a control input. The inclusion of health parameters increases the number of inputs to a neural network model as well as adds another layer of nonlinearity. An accurate transient model must be able to account for this nonlinearity since all engines at some point in time have degraded performance. As a general rule of thumb for neural networks, as the number of inputs to the model increases, the number of training points increases too.

GENERATING TRAINING DATA FOR NEURAL NETWORK

For any type of neural network, both training and verification data are needed to train the network. The training data is required to ensure minimal model-fit error (MFE) while verification data is needed to ensure minimal model-representation error (MRE). When generating training and verification data points, it is important that all the points sufficiently cover the entire design space.

Design Space Sampling

Design space sampling can be regarded as the most important step of the neural network methodology because accuracy of the neural networks depends on how many points are selected for neural network training and how these points are distributed throughout the design space. The process of selecting these points and creating a training set from them is called design of experiments (DOE). In this section design space sampling methods are explained for both steady-state and transient operations.

Steady-State Training

The DOE parameters used here are inputs to the truth model which are altitude, Mach number, ambient temperature, and power setting. Limits on altitude, Mach number, and ambient temperature are determined based on the representative flight envelope given in Fig. 1 and Fig. 2.

The type of design space sampling method used for steady-state operation is Latin hypercube sampling [13]. The Latin hypercube can adequately sample the interior of the design space; however, it has the inherent disadvantage of poor sampling of the extremes of the design space. In this study, the design space is formed by the combination of the flight envelope parameters and fuel flow. The latin hypercube samples the interior of this design space to create a DOE. The value of the fuel flow is bounded between the minimum and maximum level of achievable fuel flow (engine power) at the given flight conditions. The points created by the Latin hypercube are used for training the neural networks.

Moreover, additional random points are needed to test the prediction capability of the neural networks at points other than the training points. Eventually, Latin hypercube points and random points are fed into the model to track the responses.

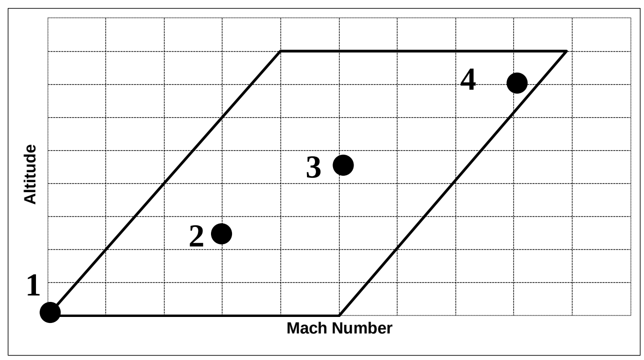


Figure 1. REPRESENTATIVE FLIGHT ENVELOPE - ALTITUDE VS. MACH NUMBER

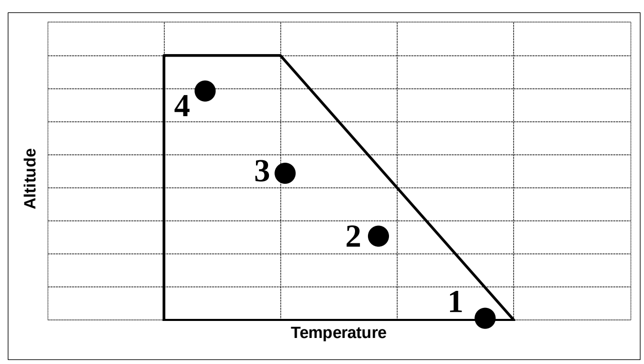


Figure 2. REPRESENTATIVE FLIGHT ENVELOPE - ALTITUDE VS. AMBIENT TEMPERATURE

Transient Training

The DOE parameters for the transient operation are altitude, Mach number, ambient temperature and fuel flow profile. The Latin hypercube samples the interior of the design space for altitude, Mach number and ambient temperature. In contrast to steady-state operation, the fuel flow in the transient operation is not a single value. It is a control input profile that represents the variation of the fuel flow as a function of time.

Relative to steady-state operation, generating training data for transient operation is more challenging because the engine state is often correlated with the control input which is the fuel flow in this case. If these inputs are independently generated, there will be many unrealistic combinations which result in unacceptable results; therefore, an additional step in generating training data is required to assure that these points are not included in the training set. To address this issue, before running the transient operation, a steady-state case is run at the maximum power setting at each point to find the maximum achievable fuel flow rate at the given flight conditions. Then, the maximum fuel flow rate value and flight conditions are fed into the transient model with a random ramp input with the upper and lower bounds equal to zero and one. The fuel flow rate profile is obtained by multiplying each point in the ramp input to the obtained maximum fuel flow. An example of the random ramp inputs used in generating the input is shown in Fig. 3. The flight conditions and the fuel flow rate input are fed into the transient model to acquire the responses at these conditions. These runs yield the training points for the neural networks obtained at each time step through the transient operation. Initially the data of all the time steps are recorded. The result is a very large data set which significantly increases the training time of the neural networks; therefore, a data reduction scheme is performed to eliminate some of the data points to reduce the size of the data set to a manageable number of points.

As it is the case in the sampling of steady-state operation, random points are needed to test the prediction capability of the neural networks in transient operation.

Altitude, Mach number and ambient temperature are chosen randomly and the random ramp input is generated in the same way that is described before and fed into the transient model with the flight conditions. The neural network results are compared with the truth model results to assess the accuracy of the model at random ramp inputs.

RESULTS

Model Description and Automated Environment

The engine model used to create DOE and random points for steady-state and transient operations is an educational aerothermal model. It is a twin spool separate flow high-bypass ratio turbofan. The model is constructed using the NASA numerical propulsion system simulation (NPSS) code. In this research, only shaft dynamics is included in the transient model.

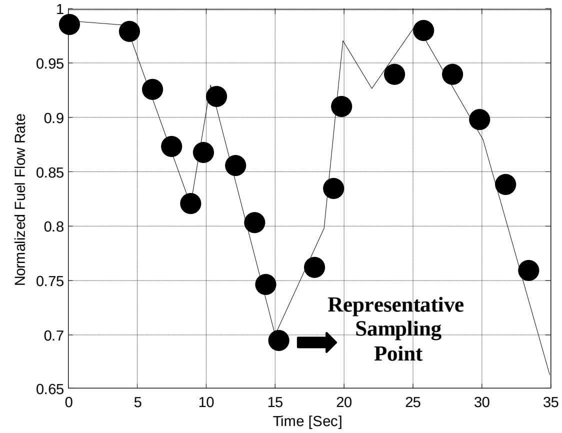


Figure 3. RANDOM RAMP INPUT

Sampling the design space and executing these points would be very intensive if they were done manually; therefore, the process is automated using Model Center software. Model Center is developed by Phoenix Integration Inc. for process integration and design optimization purposes. The sampling steps explained in previous section are implemented in the integration environment so the entire process can be executed numerous times with minimal effort.

The data points are used to train the neural networks for steady-state and transient operations. The static feed-forward neural net (FFNN) with one hidden layer is selected and trained for single time step. Details about the selected type and structure of the neural net, its activation function and the rationales behind this selection are provided in Appendix A.

Neural Network Training Results

The steady-state network without health parameters was trained on 10500 data points with an additional 2625 points used for verification. To assess the prediction accuracy of the neural nets at the training points and random points, the histogram of the model prediction error at the training points and random points are investigated. The former is called model fit error (MFE) and the latter is called model representation error (MRE). For all the parameters simulated, the MFE and MRE demonstrate the normal distribution of errors with the standard deviation of the MFE and MRE are of the same order of magnitude, with the MRE typically being slightly larger. In general, the standard deviations of the MFE or MRE are not greater than 0.3% with thrust having the largest error. Pressures and temperatures often have errors less the 0.1%. Figure 4 shows the MFE and MRE of net thrust for the steady-state neural network.

For transient operation, the neural networks without health parameters were trained with 37769 single time step training points and 7540 single time step verification points, which is a few times greater than the steady-state training data size. This is because of the inclusion of the transient data, which greatly expand the required design space to be sampled.

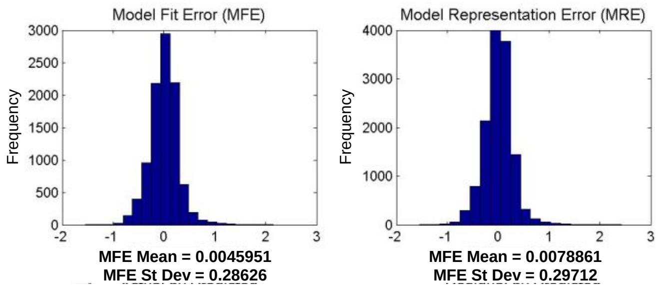


Figure 4. STEADY-STATE MFE AND MRE FOR NET THRUST

Figure 6 shows the MFE and MRE of the net thrust for the transient operation. The standard deviation of the MFE and MRE are of the same order of magnitude. There is also an increase in the standard deviation of the MRE to 0.42% relative to 0.3% standard deviation of the MRE of the steady-state case.

Similar to the steady-state results, temperature and pressure neural networks had the smallest magnitude of standard deviation of MFE and MRE.

In addition to the single time-step MFE and MRE analysis, the transient neural networks require an open loop signal analysis where the single time steps neural nets are integrated over the whole transient time to determine the engine transient outputs as a function of time. Such a step is required to capture the error of the model in transient operation. This additional analysis is required because error can propagate at each time-step resulting in an increased error at the end of the simulation time. Additionally, it is important to ensure that none of the models diverge over time. To study this error, open loop verification runs were generated at different operating points in the flight envelope. These points are highlighted in Fig. 1 and Fig. 2.

The errors between high fidelity aerothermal model and neural networks prediction for a few selected operating conditions for a given step fuel input signal are shown in Figs. 7, 8, and 9 for selected model outputs. It can be seen that the neural network predictions are within 2% of the high fidelity model predictions. An exception can be seen in the net thrust (F_N) results which show a short period of time when the absolute error value exceeds 2%. Looking at the other responses, it can be seen that at the initial point of the transient operation, the neural network model has the largest deviation from the high fidelity model. This time corresponds to a time when the rate of change of fuel flow is maximum.

There is also a steady-state error for W_{25R} and F_N; however, these errors are small and less than one percent. It is noted that as the operating points approach the edges of the flight envelope the accuracy of the model starts to deteriorate. The reason is the poor Latin hypercube sampling capability of training points near the design space edges.

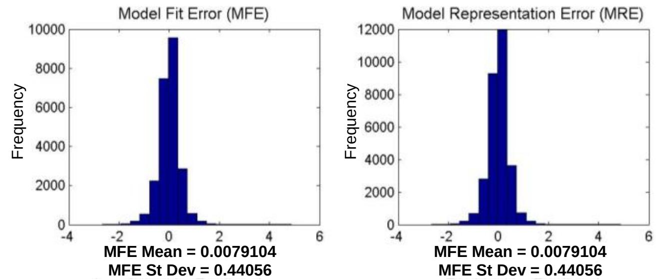


Figure 5. TRANSIENT MFE AND MRE FOR NET THRUST

NEURAL NETWORK OPEN LOOP RESULTS

CONCLUSIONS

Static feed-forward neural networks have been trained using a Latin hypercube sampling to create a transient engine model across the flight envelope. The model consists of neural net models of derivative of the state variables (shaft dynamics) and engine outputs of interest. The results show good prediction accuracy of the models for the interior points across the flight envelope when they are compared to the results of the higher fidelity aerothermal model. Initial investigations showed that replacing the high fidelity aerothermal model with neural net models reduced the execution time. The model accuracy is strongly dependent on the training data quality and the way they have been created. The model can be potentially used for on-board and off-board applications.

The advantage of the method is the simplicity of structure of its neural networks when they are compared to more complex ones such as recurrent (Hopfield) neural networks. Such complex structure can also increase the execution time. The cost of the simplicity of the models used in this study is the reduction of the time horizon to the single time step which potentially affects the accuracy.

The accuracy of the model can be improved by improving the quality of the training data, especially by sampling more points at the extremes of the flight envelope where the Latin hypercube sampling has poor performance. Combination of different sampling methodologies can address this issue.

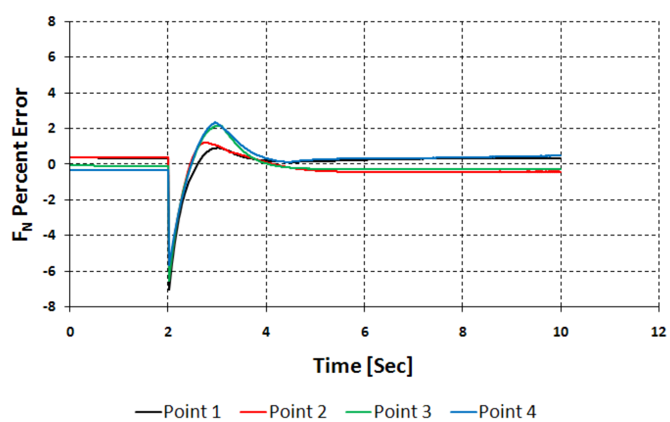


Figure 6. F_N TRANSIENT PERCENT ERROR VS. TIME

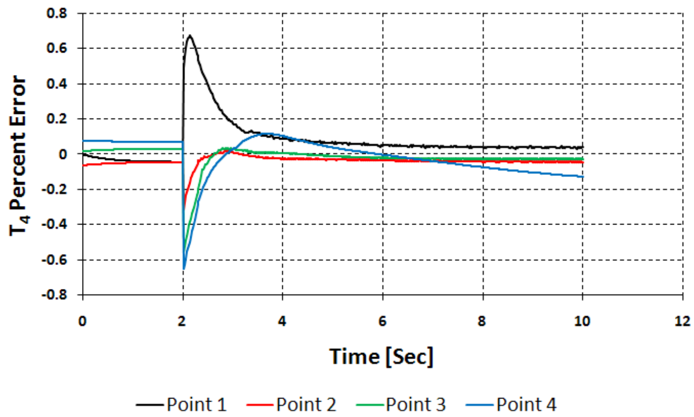


Figure 7. T_4 TRANSIENT PERCENT ERROR VS. TIME

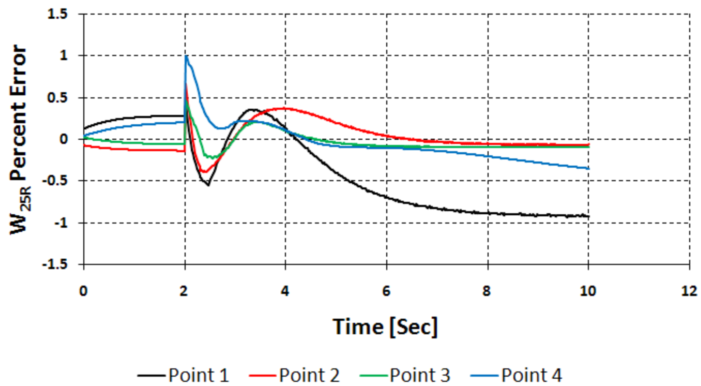


Figure 8: W_{25R} TRANSIENT PERCENT ERROR VS. TIME

NOMENCLATURE

x	System state variable
u	System input
P	Health parameter
z	Environmental parameter
y	Actual system output
$\hat{y}$	Predicted system output
e	Error between actual and predicted outputs
X_i	Neural network input
w_i	Hidden node wieighting factor
b_i,d	Neural network intercept terms
c_i	Hidden node coefficient

ACKNOWLEDGEMENT

This work was conducted with the support and supervision of the Pratt & Whitney experts.

REFERENCES

[1] Viassolo, D. E., Adibhatla, S., Brunell, B. J., Down, J. H., Gibson, N. S., Kumar, A., Mathews, H. K., and Holcomb, L. D., 2007, "Advanced Estimation for Aircraft Engines," Proc. American Control Conference, New York City, pp. 2807-2821.

[2] Aerospace Information Report, 1995, "Real-Time Modeling Methods for Gas Turbine Engine Performance," SAE Standard AIR4548.

[3] Mihaloew, J. R., Roth, S., and Creekmore, R., 1984, "Real-Time Pegasus Propulsion System Model V/Stol-Piloted Simulation Evaluation," Journal of Guidance, Control, and Dynamics.

[4] Sanghi, V., Lakshamanan, B. K., and Sundararajan, V., 2000, "Survey of Advancements in Jet-Engine Thermodynamic Simulation," Journal of Propulsion and Power, 16 (5).

[5] Fawke, A. J., and Saravanamuttoo, H. I. H., 1973, "Digital Computer Simulation of the Dynamic Response of a Twin-Spool Turbofan with Mixed Exhausts," Aeronautical Journal, 77, pp. 471-478.

[6] Stamatis, A., Mathioudakis, K., Ruiz, J., and Curnock, B., 2001, "Real Time Engine Model Implementation for Adaptive Control & Performance Monitoring of Large Civil Turbofans," Proc. Proceeding of ASME TURBO EXPO Power for Land, Sea and Air, New Orleans, Louisiana, 1, pp. 25.

[7] Lichtsinder, M., and Levy, Y., 2006, "Jet Engine Model for Control and Real-Time Simulations," Journal of Engineering for Gas Turbines and Power, 128 (4), pp. 745-753.

[8] Chiras, N., Evans, C., and Rees, D., 2002, "Global Nonlinear Modeling of Gas Turbine Dynamics Using Narmax Structures," Journal of Engineering for Gas Turbines and Power, 124 (4), pp. 817-826.

[9] Chiras, N., Evans, C., and Rees, D., 2002, "Nonlinear Gas Turbine Modeling Using Feedforward Neural Networks," Proc. Proceeding of ASME TURBO EXPO Power for Land, Sea and Air, Amsterdam, Netherlands, pp. 2002-30035.

[10] Luppold, R., Brotherton, T., and Volponi, A., 2007, "Adaptive on-Wing Gas Turbine Engine Performance Estimation," Proc. Aerospace Conference, pp. 1-12.

[11] Shankar, P., and Yedavalli, R. K., 2009, "Neural-Network-Based Observer for Turbine Engine Parameter Estimation," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 223 (6), pp. 821-832.

[12] Oh, S. K., Pedrycz, W., and Park, H. S., 2007, "Fuzzy Relation-Based Neural Networks and Their Hybrid Identification," IEEE Transactions on Instrumentation and Measurement, 56 (6), pp. 2522-2537.

[13] Mckay, M. D., Beckman, R. J., and Conover, W. J., 1979, "A Comparison of Three Methods for Selecting Values of

Input Variables in the Analysis of Output from a Computer Code," Technometrics, 21 (2), pp. 239-245.
[14] Norgaard, M., Ravn, O., Poulsen, N. K., and Hansen, L. K., 2000, Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook, Springer Verlag.
[15] Cybenco, G., 1989, "Approximation by Superpositions of a Sigmoidal Function," Mathematics of Control, Signals, and Systems, 2 (4), pp. 303-314.

APPENDIX A.

NEURAL NETWORK METHODOLOGY TO MODEL DYNAMIC SYSTEMS

A neural network is a network of simple units which together can model very complex behaviors [14]. The neural network methodology is based on mimicking the function of the brain of biological systems as it is described below.

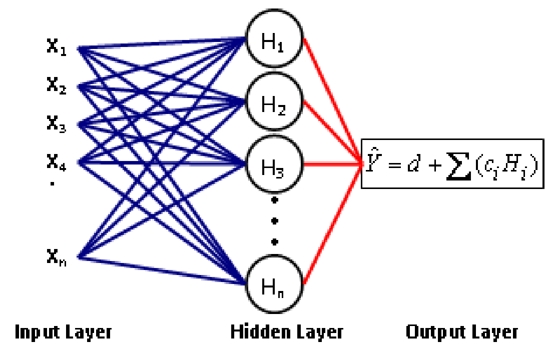
Neural Network Anatomy

In a simplified description, the brain of a biological system consists of connection of million brain cells called neuron in the form of a network. Each cell receives signals from others through a numerous branches called "dendrite" and can send signal to other cell through a single branch called "axon". Neurons can be excited by receiving an electrical signal that is above the threshold from the neighboring neurons through the dendrites and excite the next neighbor using electrical signals through their single axon. The interconnections of these cells results in a complex behavior of the brain. Similar to the biological brain, a mathematical model of neurons and their connection is developed to model a complex and non-linear behaviors. The building block of a neural network is a node or neuron which can receive multiple inputs. Input values are sent to an activation function and the activation function provides an output value to the next node or the output layer.

The connection of inputs and outputs of nodes can create a network called neural network. While it is not necessary for the connections to be in any specific order, for the sake of simplicity, the neural networks are commonly structured in a way to have a layer of inputs, single or multiple layers of nodes which are called hidden layers and an output layer. The input layer receives the inputs and sends them to the nodes in the first hidden layer. In simple neural nets there is no connections between nodes in each hidden layer and connections are only from the outputs of one hidden layer to the nodes in the next hidden layer. The outputs of the nodes of the last hidden layer are sent to output layer to combine to a single output value (Fig. 9).

Different type of activation functions are defined for the neural nets. In feed-forward neural networks (FFNN), the activation function is a deterministic function. The most commonly used functions for non-discrete sets are sigmoid or tangent hyperbolic functions as they are provided in Eq. (4) and Eq. (5).

$$f_{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{4}$$



$$f_{\tanh}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{5}$$

Figure 9. NEURAL NETWORK STRUCTURE

In FFNN, the inputs are multiplied to weighting factors and their sum is sent to an activation function as shown in Eq. (6).

$$H_j = f_{activation}\left(\sum_{i=1}^{n_{inputs}} w_i X_i + b_j\right) \tag{6}$$

In radial basis function neural networks (RBFNN), the activation function is defined as a probability function which provides a probability value based on the distance of the input values to pre-defined values from a data-set. These pre-defined values are considered to be the nodes in an RBFNN. In both FFNN and RBFNN models, the output of the activation functions of all nodes are sent to the output layer to combine to a single value as demonstrated in Eq. (7).

$$\hat{Y} = d + \sum_{j=1}^{n_{node}} c_j H_j \tag{7}$$

RBFNN can handle the larger range of variability and may be used for online training, but compared to feed-forward neural networks, they need a large number of nodes to provide an accurate prediction. A neural network with a single hidden layer using sigmoid or hyperbolic activation function can model any continuous function [15].

Training Neural Network

To determine the optimum values of the inputs weighting factors, hidden node coefficients and intercept terms, a data set (training set) with enough data points is required. An optimization scheme determines the value of weighting factors that minimize the sum square of errors between the predicted and actual values as shown in Eq. (8). With enough training points, the model can be used to predict the output of any input combination inside the training range with good accuracy. Increasing the number of neurons can also increase the prediction accuracy of the model; however, for higher number of the neurons a larger training data set is required. Attention must be paid to avoid over-fitting the weighting factors in which case they provide accurate results for the training set, but starts to deteriorate the prediction results of inputs other than the training set.

$$e = \sum_{k=1}^{n_{sample}} (\hat{Y}_k - Y_k)^2 \tag{8}$$

The mathematical process of training the neural net is straightforward but creating an appropriate data set to train the parameters has significant effects on the accuracy of the outputs. Identifying the valid range of inputs, selecting a correct method of sampling the data and filtering unrealistic combinations that can contaminate the training set play a pivotal role to have a good training set.