NASA-TM-107110
19960008932

NASA
Technical Memorandum 107110

Army Research Laboratory
Technical Report ARL–TR–765

Adaptive Optimization of Aircraft Engine Performance Using Neural Networks

Donald L. Simon
Vehicle Propulsion Directorate
U.S. Army Research Laboratory
Lewis Research Center
Cleveland, Ohio

and

Theresa W. Long
NeuroDyne, Inc.
Williamsburg, Virginia

Prepared for the
86th Symposium on Advanced Aero Engines Concepts and Controls
sponsored by the Advisory Group for Aerospace Research
and Development's Propulsion and Energetics Panel
Bellevue, Washington, September 25–29, 1995

National Aeronautics and
Space Administration

U.S. ARMY
RESEARCH LABORATORY

Adaptive Optimization of Aircraft Engine Performance Using Neural Networks

Donald L. Simon
Vehicle Propulsion Directorate
U.S. Army Research Laboratory
NASA Lewis Research Center
Cleveland, Ohio 44135

Theresa W. Long
NeuroDyne, Inc.
123 Hunting Cove
Williamsburg, VA 23185

ABSTRACT

Preliminary results are presented on the development of an adaptive neural network based control algorithm to enhance aircraft engine performance. This work builds upon a previous National Aeronautics and Space Administration (NASA) effort known as Performance Seeking Control (PSC). PSC is an adaptive control algorithm which contains a model of the aircraft's propulsion system which is updated on-line to match the operation of the aircraft's actual propulsion system. Information from the on-line model is used to adapt the control system during flight to allow optimal operation of the aircraft's propulsion system (inlet, engine, and nozzle) to improve aircraft engine performance without compromising reliability or operability. Performance Seeking Control has been shown to yield reductions in fuel flow, increases in thrust, and reductions in engine fan turbine inlet temperature. The neural network based adaptive control, like PSC, will contain a model of the propulsion system which will be used to calculate optimal control commands on-line. Hopes are that it will be able to provide some additional benefits above and beyond those of PSC. The PSC algorithm is computationally intensive, it is valid only at near steady-state flight conditions, and it has no way to adapt or learn on-line. These issues are being addressed in the development of the optimal neural controller. Specialized neural network processing hardware is being developed to run the software, the algorithm will be valid at steady-state and transient conditions, and will take advantage of the on-line learning capability of neural networks. Future plans include testing the neural network software and hardware prototype against an aircraft engine simulation. In this paper the proposed neural network software and hardware is described and preliminary neural network training results are presented.

INTRODUCTION

Performance Seeking Control (PSC) is an adaptive model-based control algorithm which optimizes aircraft propulsion system performance in flight. The adaptive nature of the control system enables it to account for engine to engine manufacturing variations, off nominal engine component performance, or deterioration which may occur to the engine over time. This technology was a joint NASA, McDonnell Douglas, and Pratt & Whitney effort. The PSC algorithm has been flight tested on a NASA research aircraft at the NASA Ames/Dryden Flight Research Facility [1, 2, 3].

A NASA Small Business and Innovative Research (SBIR) contract which builds upon the previous NASA PSC effort has been established with NeuroDyne Inc. The objective of this effort is to investigate the use of neural networks for the implementation of a model-based adaptive control algorithm. Preliminary progress under this effort is presented.

Neural networks are computational representations of biological neurons in the human brain. Consisting of several layers of nodes connected by weighted synaptical connections, neural networks can be trained to recognize a pattern of inputs and provide desired outputs. They lend themselves very well to pattern recognition problems, or for this application, estimation of system parameters.

Neural networks have the promise of being faster and requiring less memory than traditional computer algorithms when implemented in specialized hardware. The original performance seeking control algorithm is only valid at near-steady-state conditions due to the computational burden of modeling transient maneuvers. A neural network implementation may be able to overcome this limitation. Neural networks also have the added benefit of being able to learn on line. Thus they may be able to adapt to new or unexpected conditions that the traditional PSC implementation could not account for.

PERFORMANCE SEEKING CONTROL

The objective of Performance Seeking Control is to adaptively optimize the near steady-state performance of an aircraft propulsion system in real-time by calculating engine control trims which are applied to the nominal engine schedules. Performance Seeking Control can select one of three modes for optimization. The modes are minimizing fuel consumption while maintaining nominal thrust, minimizing fan turbine inlet temperature (FTIT) while maintaining nominal thrust, or maximizing thrust while maintaining engine nominal FTIT. A block diagram of the PSC architecture is shown in Figure 1. It includes an estimator, a model of the propulsion system, and an optimizer. The estimator consists of a Kalman Filter using flight measurements to estimate five component deviation parameters. These component deviation parameters are those of the low pressure turbine efficiency (DELPT), the high pressure turbine efficiency (DEHPT), the fan airflow (DWFAN), the high pressure compressor airflow (DWHPC), and the high pressure turbine area (AAHT). The propulsion system model consists of linear and nonlinear models used to estimate unmeasured engine parameter based on flight measurements and the five component deviation parameters estimated by the Kalman filter. The model is continuously updated to adaptively model the dynamics of the actual propulsion system. The optimizer uses linear programming techniques to optimize control trim settings based upon the present operating condition of the propulsion system and the optimization mode selected. The performance seeking control algorithm has been flight tested at the NASA Ames/Dryden Flight Research Facility on a NASA F-15 research vehicle. This aircraft is equipped with two Pratt & Whitney 1128 afterburning turbofan engines. The results from this testing has shown that Performance Seeking Control does indeed yield significant improvements over traditional control techniques. Thrust increases up to 15% at military power, turbine temperature decreases up to 120°F at military power, and Specific Fuel Consumption (SFC) improvements up to 2.0% at cruise have been achieved [4].

A limitation with the PSC algorithm is the speed at which it can be executed. The algorithm is rather computationally complex and is not able to achieve real-time performance when implemented in conventional computer processors where the calculations take place in a serial fashion. Therefore it is valid only at near steady state conditions. Also the accuracy of the algorithm is dependent on the accuracy of the adaptive model. Neural networks, because of their parallel nature and ability to learn on-line may help to overcome such limitations.

OPTIMAL NEURAL CONTROLLER

The proposed optimal neural controller is being developed to control a Pratt & Whitney 1128 engine simulation. The proposed architecture is shown in Figure

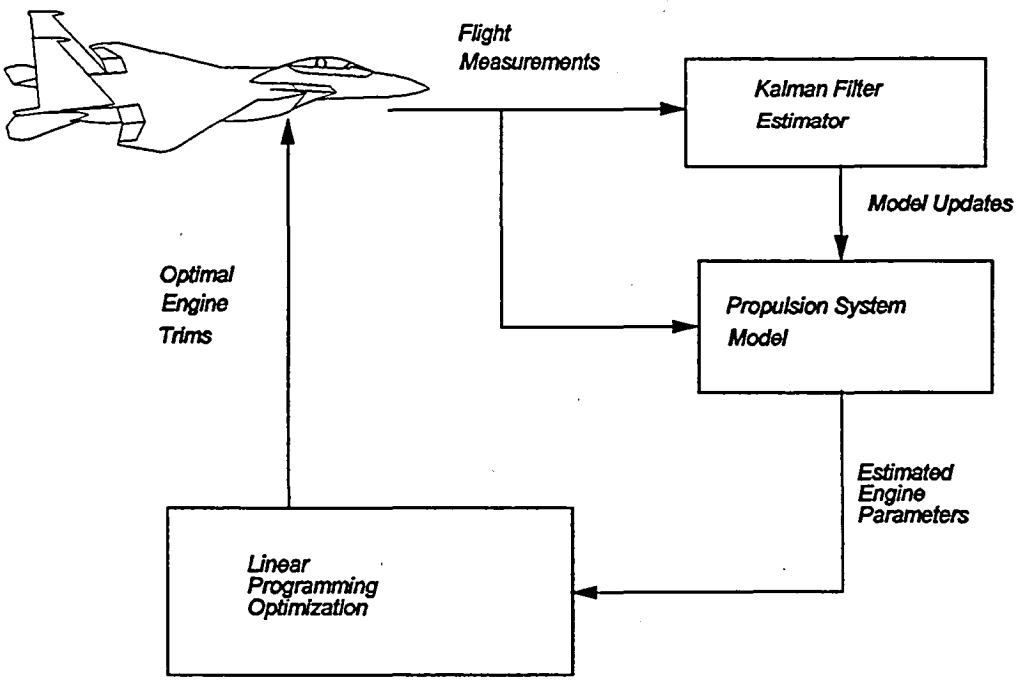


Figure 1. Performance Seeking Control Architecture

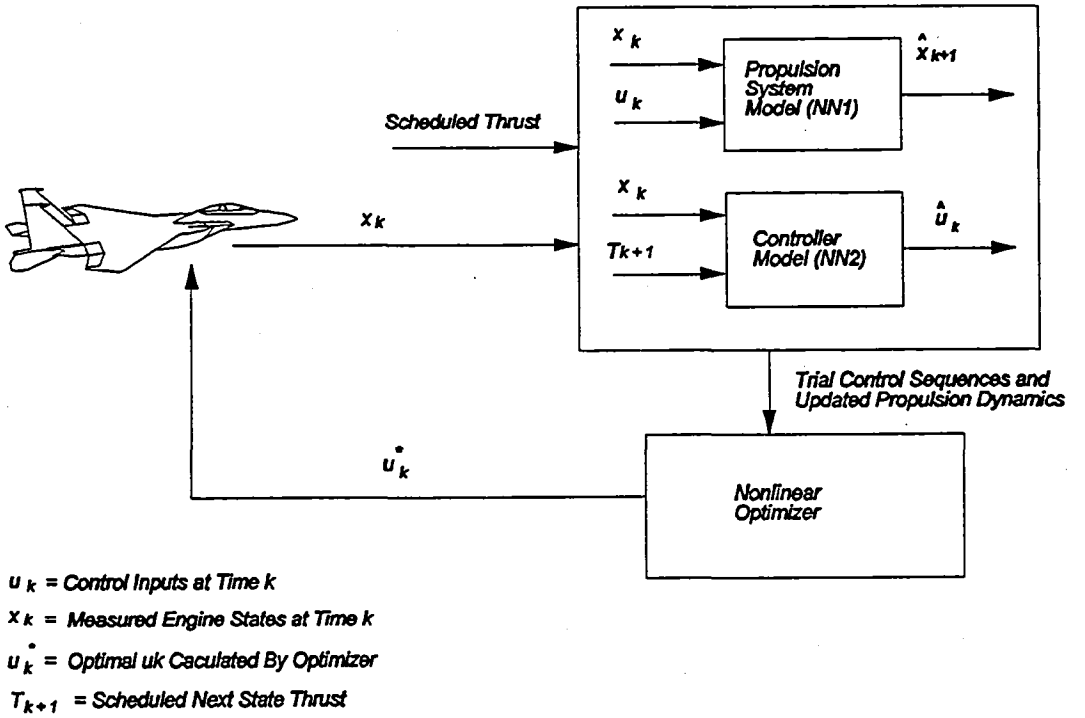


Figure 2. Optimal Neural Controller

2. It consists of a propulsion system model, a controller model, and a nonlinear optimizer. Plans call for implementing each of these components as neural networks. The propulsion system and controller models are used to generate trial control sequences n time steps into the future spanning the range of admissible control based on the present operating conditions and the scheduled thrust. These trial control sequences along with the dynamics of the propulsion system model are used by the optimization algorithm to perform an iterative optimal gradient search. After convergence, the optimizer produces the optimal control according the desired cost function. Logic is included to insure that the control and dynamic constraints are not exceeded. Prior to on-line operation the propulsion system model and the controller model neural networks are trained using simulation data. The controller model is further refined off-line by training on the optimizer outputs. On-line, the error between the propulsion system model predicted outputs and the measured outputs of the actual engine are monitored. If the error exceeds a certain threshold, the propulsion system model will be updated to maintain an accurate representation of the engine dynamics. A change in the propulsion system model will of course impact the response of the controller model as it's outputs adjust to allow the propulsion system to continue to provide the scheduled thrust. The optimizer then calculates a new optimal control for this condition using knowledge of the updated propulsion system dynamics and the new trial control sequences. The new optimal control is fed back to the inner loop control to update control trims. It is also used to update the controller model neural net on-line.

Preliminary neural network versions of the propulsion system model and the controller model have been developed. The optimization routine is currently being coded in a higher level language, however future plans call for implementing this routine in a neural network also. Special purpose neural processing hardware is being developed for future implementation of the entire algorithm. By doing so we hope to demonstrate real-time operation. The development of the elements of the optimal neural controller are further discussed in the following sections.

The Propulsion System and the Controller Models

Pratt & Whitney has provided NeuroDyne data from an 1128 engine simulation for development of the propulsion system and controller neural network models. The data sets provided were collected from the engine simulation by perturbing the inputs around steady state operating points throughout the state space. This data consists of 8 measurable engine states and 6 control inputs. The engine state vector, x, is defined as

$x = [PS20\ T20\ PB\ T45\ P60\ N1\ N2\ PAM]^T$

PS20 : Engine Inlet Pressure
T20 : Engine Inlet Temperature
PB : Burner Pressure
T45 : Low Pressure Turbine Inlet Temperature
P60 : Nozzle Inlet Pressure
N1 : Low Rotor Speed
N2 : High Rotor Speed
PAM : Ambient Pressure

The control vector, u, consists of 6 elements

$u = [WF\ AJ\ FVV\ CVV\ HPX\ BLD]^T$

WF : Main Burner Fuel Flow
AJ : Exhaust Nozzle Area
FVV : Fan Variable Vanes
CVV : Compressor Variable Vanes
HPX : Horsepower Extraction
BLD : Bleed Flow

The Propulsion System Model

The propulsion system model (NN1) estimates the state at time k+1 from the states and control inputs at time k. There are 14 inputs to NN1 (8 states, 6 controls), and 8 outputs from NN1 (8 states). An initial propulsion system model has been developed by training a 1-hidden layer feedforward neural net consisting of 50 hidden nodes. The performance of this neural network in matching the response of the Pratt & Whitney engine simulation is very good. The average generalization error is 1.00% where the average generalization error is defined as:

$$e = \sqrt{\frac{\sum_{i=1}^{n} [x_i^* - x_i]^T [x_i^* - x_i]}{\sum_{i=1}^{n} [x_i^*]^T [x_i^*]}}$$

where

x_i^* is the target output for the ith data point of the training or testing set.

x_i is the neural net estimate of the ith data point of the training or testing set.

n is the total number of points in the training or testing set

Table 1 shows the generalization error for each output of NN1 as a percentage of the outputs operating range. Also shown in the table is the approximate accuracy of each sensor as provided by Pratt & Whitney. On-line, the propulsion system model accuracy will be limited by the accuracy of the sensor measurements which are provided as inputs.

Table 1 Propulsion System Model (NN1) Estimation Error

state	Estimation error (%)	Sensor error (%)
PS20	±0.34	±0.86
T20	±0.21	±0.41
PB	±0.61	±0.56
T45	±1.5	±1.59
P60	±0.55	±0.45
N1	±0.13	±0.13
N2	±0.10	±0.10
PAM	±0.20	±2.00

Figure 3 shows the P&W simulator output of T_{45} and NN1 estimated T45 for a steady state flight at mil power, 1.05 operating line, 60,000 ft altitude and Mach 2.0. As expected the neural net closely tracked the response of the simulator. Although the neural net was trained on steady state data it was desirable to check the networks ability to handle transient conditions. Figure 4 shows the simulator and NN1 estimated T45 for a transient condition of acceleration at 30,000 ft altitude and Mach 0.9. Once again the neural network was able to closely match the outputs of the simulation.

The Controller Model

The controller model (NN2) estimates the required control (u_k) based on the present state (x_k) and scheduled thrust (T). NN2 has 9 inputs (8 states, and 1 thrust), and 6 outputs (6 controls). Once again a 1-hidden layer neural net having 50 hidden nodes was selected for initial development. The controller model (NN2) will eventually be trained on the optimal control commands generated by the optimizer, but for an initial starting guess of the model weights the network was trained on the Pratt & Whitney engine simulation data. The average generalization error for this network is 7.54%. It should be noted that the accuracy requirement for the initial controller model is not as stringent as the propulsion system model NN1 because the controller model weights will be further refined by the optimizer. Table 2 shows the generalization error for each output of NN2 as a percentage of the outputs operating range. Measurement accuracy information is unavailable.

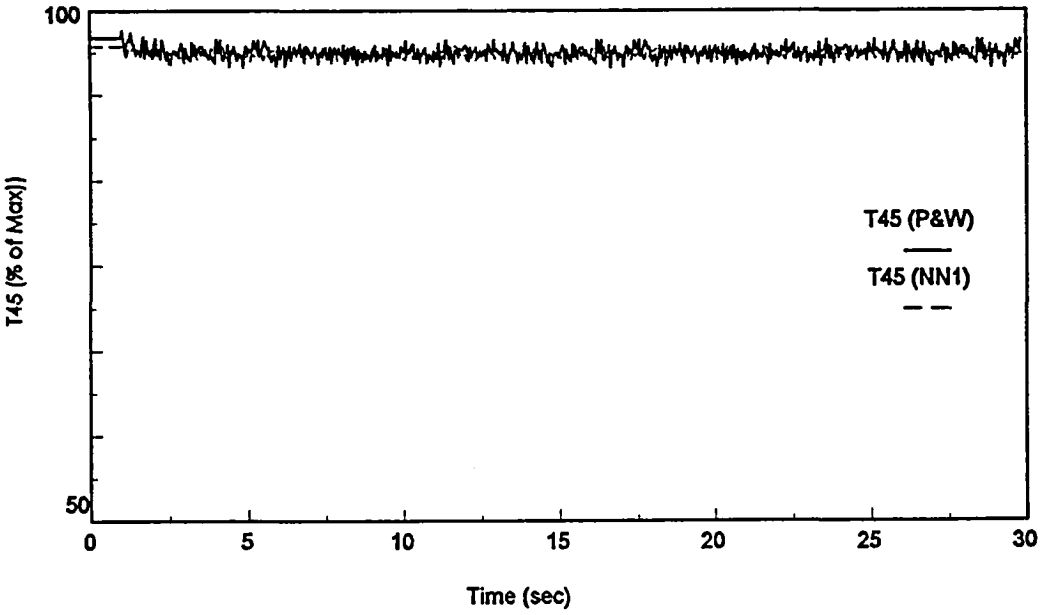


Figure 3. Simulator vs. NN1 Estimated T45
at Steady State Condition

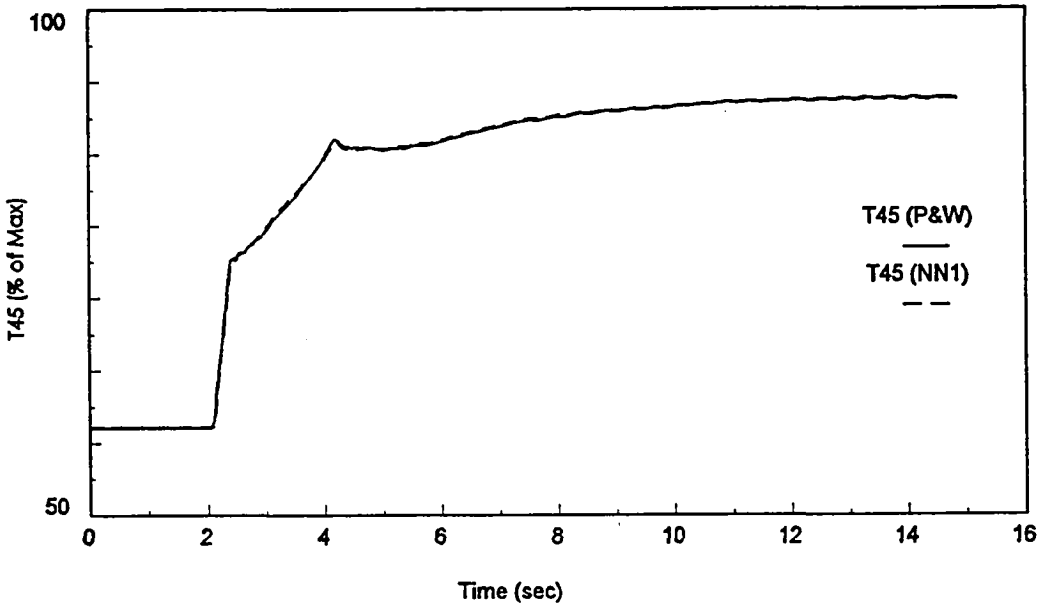


Figure 4. Simulator vs. NN1 Estimated T45
at Transient Condition

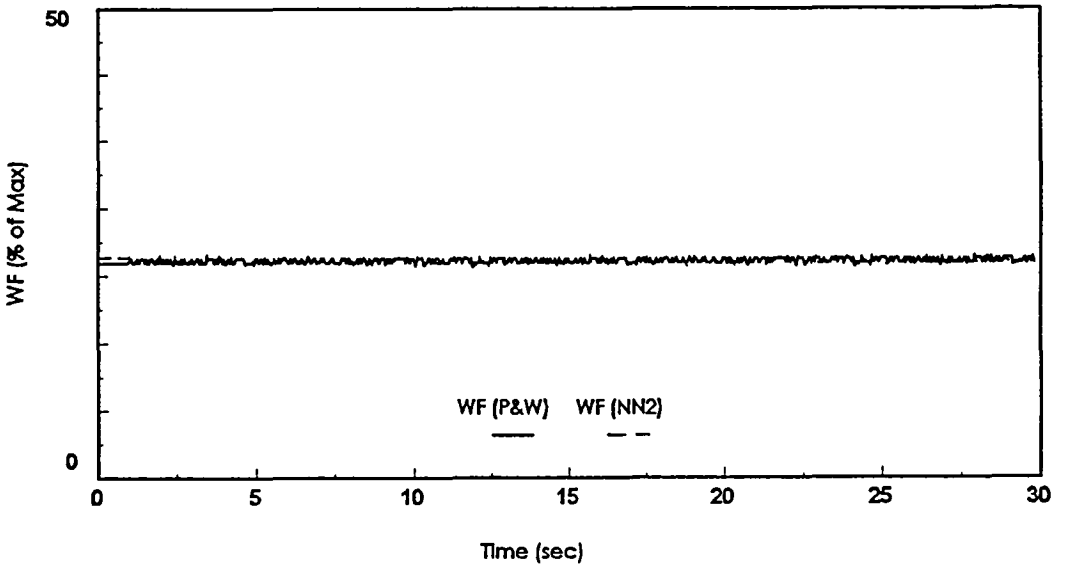


Figure 5. Simulator vs. NN2 Estimated
WF at Steady State Condition

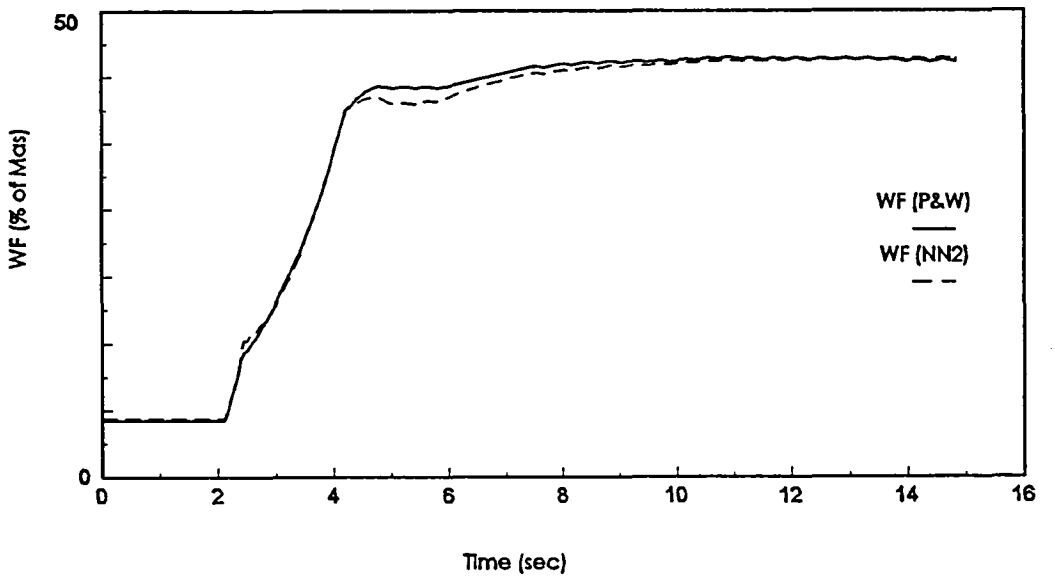


Figure 6. Simulator vs. NN2 Estimated
WF at Transient Condition

Table 2 Controller Model (NN2) Estimation Error

Control Input	Estimation error (%)
WF	1.14
AJ	1.22
FVV	6.19
CVV	3.51
HPX	4.86
BLD	8.00

Figure 5 shows the P&W simulator output of WF (fuel flow rate) and NN2 estimated WF for a steady state flight at mil power, 1.05 operating line, 60,000 ft altitude and Mach 2.0. Figure 6 shows the simulator and NN2 estimated WF for the transient acceleration condition at 30,000 ft altitude and Mach 0.9. Both figures show that NN2 did a fair job of tracking the simulator output.

On-Line Learning

Figure 7 illustrates the on-line learning of these two neural networks. The propulsion system model will be updated based on the error between its output and the actual measured propulsion system output. This will allow it to adapt to account for any off nominal engine behavior or any deterioration which may occur over time. The controller model will be updated based on the error between its output and that of nonlinear optimizer allowing it to update to account for any changes in the optimal control. The one hidden layer feed forward neural network used to obtain the initial results is probably not the best neural net architecture for on-line learning. Later in the paper alternative neural network architectures which are undergoing evaluation will be discussed.

Generating Trial Control and State Sequences

The nonlinear optimization routine requires an initial starting guess for the sequence of states and control inputs to meet the required constraints. This can be accomplished by cascading together the propulsion system model (NN1) and the controller model (NN2) as shown in Figure 8. For illustration purposes a constant net thrust constraint is used. The constraint can be altered to be an acceleration or deceleration schedule for thrust. Figure 8 shows that at time stage k given a next stage scheduled thrust, T_{k+1}, and the current measured state, x_k, the necessary control, u_k, can be computed from the controller model. Therefore, having selected a constant thrust value T, and an acceptable variation for the thrust (ΔT), m increments of thrust between $T-\Delta T$ and $T+\Delta T$ can be created.

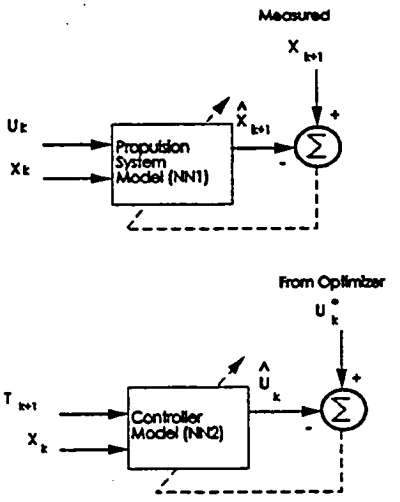


Figure 7. Neural Networks On-Line Update

With each of the m thrusts, a starting value of u_k can be computed using NN2. All the u_k's are checked against constraints such as bounds and rate limit, and corrections are made to stay within the constraints. We will follow through the calculations for one of the m u's. The rest would be similar. At time stage k, given the starting value of u_k (generated by NN2), the estimated next state $\hat{x}_{k+1}$ can be predicted using the propulsion system model (NN1). Moving onto time stage k+1 using the NN1 estimated $\hat{x}_{k+1}$ and the next stage scheduled thrust (T_{k+2}) as the inputs, the next stage estimate control input $\hat{u}_{k+1}$ can be computed from NN2. This continues until the end of the horizon is reached, thus obtaining a sequence of states and control inputs as a starting guess.

Nonlinear Optimizer

The nonlinear optimization routine is currently being developed in the "C" language on a workstation computer. The calculations involved are highly parallel in nature and are dependent on the architecture of the propulsion system model (NN1) and the outputs of both the propulsion system model (NN1) and the controller model (NN1). NeuroDyne plans to investigate the implementation of the optimization routine in a neural network in the future.

The optimization routine uses a receding horizon cost function which is a function of engine states, x, and control inputs, u, over a finite number of time stages, n, into the future. The cost function has the following form at the present time stage k

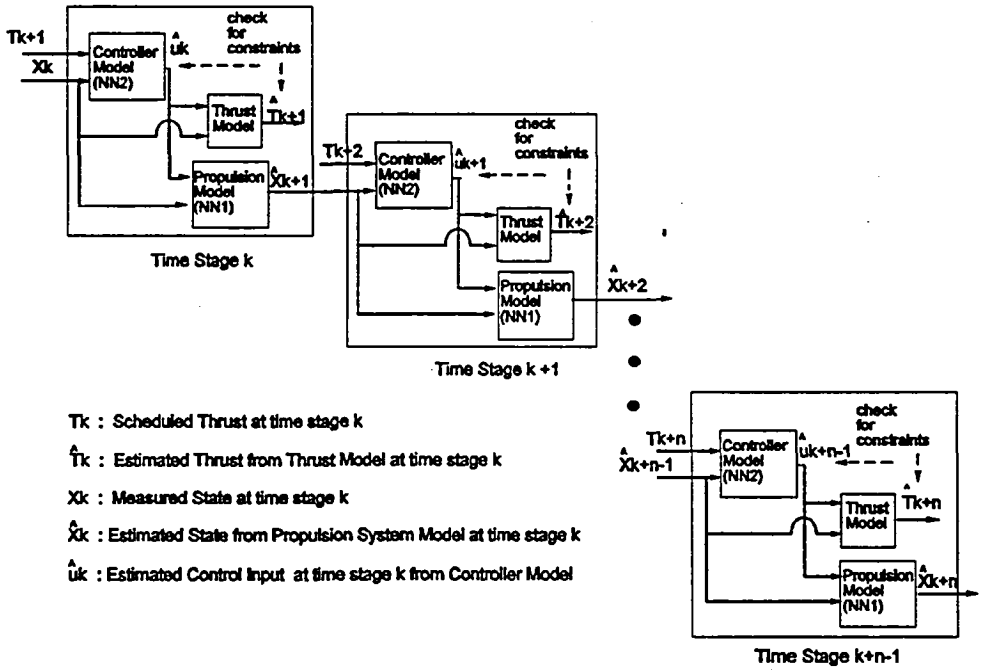


Figure 8. Generating Trial Control Sequences

$$J_k = \sum_{i=k}^{n+k-1} f(x_i, u_i) \qquad (1)$$

where

x_i is the state at time i
u_i is the control input at time i
n is the size of the look ahead horizon.

One of the design issues of this optimization approach is selection of the horizon size used in the objective function. Theoretically, only an infinite horizon produces a global optimum. For a nonlinear system, it is necessary to use a receding (or limited) horizon. The size of the horizon depends heavily on the accuracy of the model. Therefore it is essential that the neural nets used for implementation provide good accuracy.

After NN1 and NN2 have be used to obtain an estimate of the propulsion system dynamics and an initial estimated sequence of state and control over the horizon of n, the sequence of estimated control inputs is corrected iteratively according to

$$U_{NEW} = U - \alpha \frac{\partial J}{\partial U} \qquad (2)$$

where

U is a sequence of $[\hat{u}_k, \hat{u}_{k+1}, \ldots\ldots \hat{u}_{k+n-1}]$

$\partial J/\partial U$ is dependent on the propulsion system model neural network architecture.

α is a step scaling factor between 0 and 1

Each time U_{NEW} is computed it is applied to NN1 to generate a new estimated sequence of states. These estimated states and U_{NEW} are then used as inputs to the thrust model. If the resulting net thrust at each time stage is still within $T \pm \Delta T$, U_{NEW} is saved as U, and the algorithm proceeds to the next iteration. Otherwise, the offending individual u_i is corrected, or the step size α in equation (2) is reduced and U_{NEW} is recomputed.

After convergence, the final J is computed and saved for this sequence of U. We then proceed to the next initial U. At each time stage k, there will exist m sequences of U and m J's. The optimal control sequence will be the sequence which minimizes the cost function J. The optimal control, u^*_k, at the current time stage is the first control vector of that optimal control sequence. This is analogous to playing chess. At each point of the game, the player has an optimal sequence computed but only plays the first move of that sequence.

An example is provided below to illustrate operation of the optimization routine.

Optimizer Example

The optimization algorithm is set up so that various forms of the cost function can be used. For their preliminary work NeuroDyne has been working with a cost function suggested by Pratt & Whitney which penalizes high fuel usage and high low pressure turbine temperatures.

Equation (3) shows the instantaneous (1 stage) cost function $J_{instant}$. The receding horizon cost function of horizon n would be a sum of n of such functions.

$$J_{instant}=C_1 WF+\frac{C_2}{e^{[-C_3(T45-T_{ref})]}-1} \qquad (3)$$

where
C_1, C2, and C3 are constants suggested by Pratt
WF: fuel flow
T45 : LPT inlet temperature
T_{ref} = Reference temperature provided by Pratt

By dividing by C_1 the coefficient of WF becomes one. Then by defining a = C2/C1, b = C3, and c = C3•T_{ref} you obtain:

$$J_{instant}=WF+\frac{a}{e^{[-b*T45+c]}-1} \qquad (3a)$$

Let us assume for multiple look ahead, a horizon size of 3 is used so all the terms can be shown in expanded form.

define

$$g_k=\frac{1}{e^{[-b*T45_k+c]}-1} \qquad (4)$$

where the index k indicates time stage k.

The cost function at time stage k is:

$$J_k=WF_k+WF_{k+1}+WF_{k+2}+a(g_{k+1}+g_{k+2}+g_{k+3}) \qquad (5)$$

Since T45 is the fourth element of x, and WF is the first element of U we obtain the following:

$$\frac{\partial g_k}{\partial x_k}=[0\ 0\ 0\ \frac{b*e^{[-bT45_k+c]}}{(e^{[-bT45_k+c]}-1)^2}\ 0\ 0\ 0\ 0]^T \qquad (6)$$

$$\frac{\partial J_k}{\partial u_k}=[1\ 0\ 0\ 0\ 0\ 0]^T+a\left[\left(\frac{\partial g_{k+1}}{\partial x_{k+1}}\right)^T\left(\frac{\partial x_{k+1}}{\partial u_k}\right)^T+\left(\frac{\partial g_{k+2}}{\partial x_{k+2}}\right)^T\left(\frac{\partial x_{k+2}}{\partial x_{k+1}}\right)^T\left(\frac{\partial x_{k+1}}{\partial u_k}\right)^T+\left(\frac{\partial g_{k+3}}{\partial x_{k+3}}\right)^T\left(\frac{\partial x_{k+3}}{\partial x_{k+2}}\right)^T\left(\frac{\partial x_{k+2}}{\partial x_{k+1}}\right)^T\left(\frac{\partial x_{k+1}}{\partial u_k}\right)^T\right]^T$$

$$\frac{\partial J_k}{\partial u_{k+1}}=[1\ 0\ 0\ 0\ 0\ 0]^T+a\left[\left(\frac{\partial g_{k+2}}{\partial x_{k+2}}\right)^T\left(\frac{\partial x_{k+2}}{\partial u_{k+1}}\right)^T+\left(\frac{\partial g_{k+3}}{\partial x_{k+3}}\right)^T\left(\frac{\partial x_{k+3}}{\partial x_{k+2}}\right)^T\left(\frac{\partial x_{k+2}}{\partial u_{k+1}}\right)^T\right]^T$$

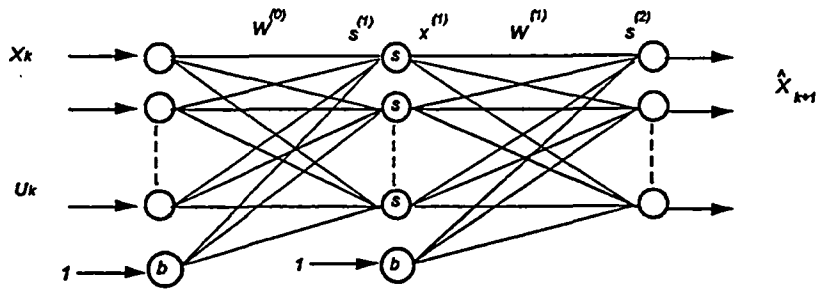
$$\frac{\partial J_k}{\partial u_{k+2}}=[1\ 0\ 0\ 0\ 0\ 0]^T+a\left[\left(\frac{\partial g_{k+3}}{\partial x_{k+3}}\right)^T\left(\frac{\partial x_{k+3}}{\partial u_{k+2}}\right)^T\right]^T \qquad (7)$$

For a horizon of n, the number of multiplications for equation set (7) would be $O(n^3)$ x (matrix multiplication for 1 time stage). The matrix multiplication for 1 time stage is a function of the size of the neural net, the state vector, and the control vector. But a simple approximation for 1 time stage cost for a 1 hidden layer net would be $O(L^3)$, where L = max(N_u, R, P), with
N_u : size of the control vector u
R : size of the state vector x
P : # of hidden nodes for a 1 hidden layer net.

Recursive calculation methods are being investigated to help reduce the computing cost of the gradients in equation set (7).

The architecture of the propulsion system model (NN1) determines the calculation of the Jacobians used in $\partial J/\partial U$. Assuming a 1 hidden layer net as shown in Figure 9 the Jacobians used in equation set (7) can be calculated as shown in equations (8) and (9).

$$\frac{\partial x_{i+1}}{\partial x_i}=W^{(1)}\frac{\partial x_i^{(1)}}{\partial s_i^{(1)}}W_x^{(0)} \qquad (8)$$



N : Number of input nodes
N = R + Nu
R : The dimension of the state x, and the number of ouput nodes
Nu : The dimension of the control u
P : Number of hidden nodes
b : Bias node
s : Sigmoid node, which is a sigmoid variant of the form : $\frac{x}{1+|x|}$

Figure 9. One Hidden Layer Net

$$\frac{\partial x_{i+1}}{\partial u_i} = W^{(1)} \frac{\partial x_i^{(1)}}{\partial s_i^{(1)}} W_u^{(0)} \qquad (9)$$

where

$x_i^{(1)}$: hidden layer output at time stage i

$s_i^{(1)}$: input to the hidden layer at time stage i

$W^{(1)}$: weight matrix coming out of the hidden layer nodes to the output layer nodes, but the weights from the bias node is not included. dim. R x P

$W_x^{(0)}$: weight matrix coming out of the state input (x_k) nodes to the hidden layer nodes, note that the weights from the bias node are not included. dim. P x R
note also that $R + N_u = N$ at the input

$W_u^{(0)}$: weight matrix coming out of the control input (u_k) nodes to the hidden layer nodes, note that the weights from the bias node are not included. dim. P x N_u with N_u the dimension of the control u vector.

The sigmoid nodes of the neural net used in this project will use an Elliott nodal function of the form x/(1 + |x|). This implementation computes faster than the frequently used hyperbolic tangent, and achieves similar accuracy. Therefore,

$\left[\frac{\partial x_i^{(1)}}{\partial s_i^{(1)}}\right]_{jj}$ for the jth hidden node is : $(1 + |x_i^{(1)}(\text{jth node})|)^{-2}$. They form the diagonal of a PxP matrix.

NEURAL NETWORK ARCHITECTURES

Three important considerations for the selection of the neural network architectures used for the implementation of the optimal neural controller are :

1. Accuracy , specifically the generalization ability of the neural networks.
2. Speed of Computation : This often means the size of the network should be as small as possible.
3. Redundancy : The redundancy helps to increase the stability or memory retention of the neural network.

The accuracy of the propulsion system model affects the accuracy of the control command, as well as the size of the look ahead horizon. As an example, if the network has a 98% accuracy in predicting the outcome at time stage k+1, based on information gathered up to time stage k (the current stage), then to predict the outcome at time stage k+10, we can repeat the same process 10 times, each time using the predicted outcome as the input for further forward prediction. The accuracy of the prediction at time stage k+10 would be proportional to 0.98^{10}, which is about 80% accuracy. On the other hand, if the one stage prediction accuracy is about 90% accurate then the 10th stage prediction is only about 35% accurate. So the accuracy of the neural network limits how far the controller can look ahead in selecting an optimal control policy. It

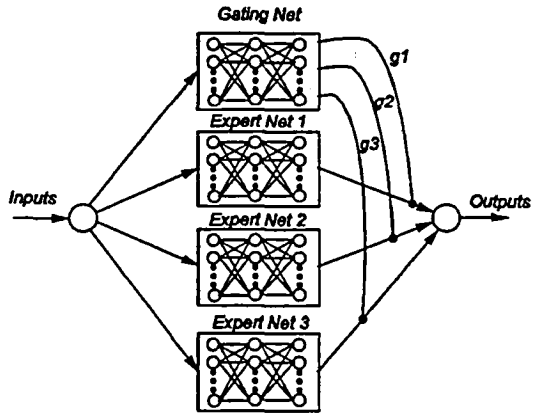


Figure 10. Competitive Net Architecture

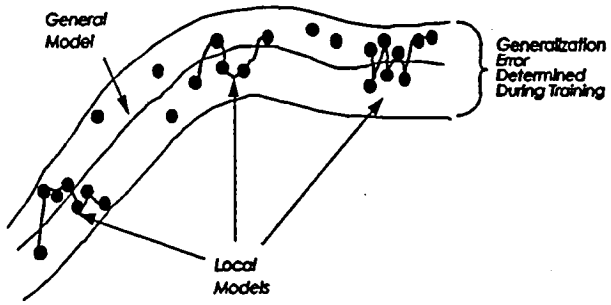


Figure 11. Global and Local Models

should be emphasized that the controller of interest here is for a nonlinear system whose dynamics is approximated by the neural net, and there is no accurate analytical model to provide an infinite horizon.

The computation speed of a neural net is an important factor in a neurocontroller which requires on-line learning and real-time optimization. The speed is very much related to the size of the network. It is necessary to choose a network as small as possible while achieving desirable accuracy.

A network with some redundancy built in is generally more robust in the face of a changing environment. Redundancy helps to increase the stability or the memory retention of the network. During on-line learning it is essential the network be able to learn new dynamics in one region of the state space without compromising accuracy at another region.

To address the issues of accuracy, speed, and redundancy NeuroDyne is considering alternative neural network architectures. Two of these architectures, a hierarchical mixture expert (HME) network [5] or competitive net , and the use of global and local neural network models in tandem are discussed below.

Hierarchical Mixture Expert (HME) Network

Figure 10 shows an HME network or a competitive network. It consists of a gating network and multiple expert networks. The input space is made up of several local regions separated by soft boundaries where data points may lie in multiple regions simultaneously. Individual expert nets provide accurate modeling at particular local regions throughout the input space. The gating net, whose output is a function of the input that goes into the individual expert nets, determines the contribution of each of the expert nets to the overall output. Initial evaluation by NeuroDyne has shown that this architecture exhibits increased accuracy and converges (learns) about an order of magnitude faster than a conventional feedforward net. When undergoing on-line learning at a particular region in the state space, one of the expert networks can be trained while the other networks remain relatively unmodified. This will allow the network to maintain accuracy in a local region without compromising accuracy throughout the rest of the state space. Another advantage is the parallel nature of the architecture. Each of the expert nets and gating nets can be assigned to a separate processor and work in parallel. A disadvantage of this architecture is that it increases the complexity of the mathematics necessary in computing the Jacobians used by the optimization routine that were shown in equations (8) and (9).

Global and Local Models

Another approach being considered is maintaining both global and local versions of the neural networks on-line. The use of a global model and a local model in tandem is designed to maintain both good generalization and model accuracy [6]. The global model and local model are identical in terms of architecture but are updated and maintained differently. The global model is valid throughout the state space while the local model can depict localized phenomena more accurately than the global model, but can not be used anywhere else except in a local region. Figure 11 illustrates the difference between the general model and its many local models.

The local models are used by the neural controller to produce accurate control commands on-line. Since the local models can not be applied to input-output pairs in different parts of the state space they must be able to

learn on-line very quickly to track the system dynamics through varying local regions. If a new trajectory is to be tracked, the localized model representing the end point of a previous trajectory is most likely a poor starting guess for the first point of the new trajectory. The search for the network weight parameters generally does not converge if we start with this local model. Therefore, when a task is changed or when a trajectory discontinuity occurs, the general model must be used to start the evolution of the local model. Because of the fast on-line learning and accuracy requirements, the local model neural networks have a faster learning rate and a lower generalization error threshold.

The global models are initially trained off-line using simulation data collected from operating points throughout the state-space. On-line the global model will still be allowed to update but at a much slower learning rate and a higher generalization error threshold than that of the local model.

NEURAL NETWORK HARDWARE

To take full advantage of the inherently parallel nature of neural networks, they need to be implemented in hardware designed specifically for their implementation. Doing so will yield significant increases in execution speed over traditional sequential computing techniques. NeuroDyne has established a subcontract to develop the neural network hardware which will run the neural network algorithm. This task will be accomplished through the use of a commercially available digital neural network processor. Although both analog and digital neural network hardware is commercially available, it was determined that a digital solution would best meet the needs of this program. Analog implementations tend to be more susceptible to temperature variations and also tend to have lower resolution than digital implementations. Although analog neural network chips are faster than their digital counterparts, they suffer from the need to communicate with the digital world. Because of the additional delays induced by the D/A and A/D convertors required for I/O, an analog implementation only presents a speed advantage when a large network of 100 to 1000 nodes is used. A digital implementation also has the advantage of being similar to current approved flight hardware. Verification therefore would be more straightforward. The digital neural network chip which has been selected is the CNAPS chip from Adaptive Solutions. This hardware contains 64 nodes on a single chip and has 16 bit resolution. The developed architecture has been purposely designed with this limit in mind to insure implementation on a single chip would be possible. For a neural network with about 60 nodes a feedforward pass requires approximately 1 microsecond. The chip is provided on a processor board which is PC bus compatible. The development environment for the CNAPS processor is very similar to the C language which should allow for easy algorithm conversion.

Future plans call for implementing the neural network algorithm in the CNAPS hardware housed in a PC chassis. This hardware/software prototype, running the optimal neural control algorithm, will be interfaced to a real time implementation of the PW 1128 state variable model. The performance of the neural network implementation in controlling the 1128 simulation will be evaluated.

SUMMARY

The Performance Seeking Control (PSC) program has demonstrated the benefits of model-based adaptive control algorithms. Implementing such an algorithm using neural networks offers the advantage of a faster implementation and the ability to learn on-line. Preliminary work has begun on the use of neural networks for adaptive optimization of aircraft engine performance. Future plans call for the refinement of the optimization algorithm and for the implementation of the software algorithm in a hardware prototype consisting of specialized neural network hardware. Capabilities which need to be demonstrated include the algorithm's ability to adaptively optimize the control for expected conditions of deterioration or off-nominal behavior. The speed at which the algorithm can accommodate these conditions also needs to be adequately demonstrated.

REFERENCES

[1] Orme, J. and Gilyard, G.,"Subsonic Flight Test Evaluation of a Propulsion System Parameter Estimation Process for the F100 Engine", AIAA 92-3745, 1992.

[2] Nobbs, S.G., Jacobs, S.W., and Donahue, D.J., "Development of the Full-Envelope Performance Seeking Control Algorithm", AIAA 92-3748, 1992.

[3] Mueller, F.D., Nobbs, S.G., Stewart, J.F., "Dual Engine Application of the Performance Seeking Control Algorithm", AIAA 93-1822, 1993.

[4] Chisholm, J.D., "In-Flight Optimization of the Total Propulsion System," AIAA 93-3744, 1992.

[5] Jordan, M.I., Jacobs, R.A., "Hierarchical Mixtures of Experts and the EM Algorithm", Neurocomputation, vol. 2, issue 2, March 1994.

[6] Long, T.W., "A Learning Controller for Decentralized Nonlinear Systems", American Control Conference, June 2-4, 1993, San Francisco.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1995	Technical Memorandum

4. TITLE AND SUBTITLE

Adaptive Optimization of Aircraft Engine Performance Using Neural Networks

5. FUNDING NUMBERS

WU-244-02-01

6. AUTHOR(S)

Donald L. Simon and Theresa W. Long

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

NASA Lewis Research Center
Cleveland, Ohio 44135-3191
and
Vehicle Propulsion Directorate
U.S. Army Research Laboratory
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-10015

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

National Aeronautics and Space Administration
Washington, D.C. 20546-0001
and
U.S. Army Research Laboratory
Adelphi, Maryland 20783-1145

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-107110
ARL-TR-765

11. SUPPLEMENTARY NOTES

Prepared for the 86th Symposium on Advanced Aero Engines Concepts and Controls sponsored by the Advisory Group for Aerospace Research and Development's Propulsion and Energetics Panel, Bellevue, Washington, September 25-29, 1995. Donald L. Simon, Vehicle Propulsion Directorate, U.S. Army Research Laboratory, and Theresa W. Long, NeuroDyne, Inc., 123 Hunting Cove, Williamsburg, Virginia 23185 (work funded by NASA Contract NAS3-27250). Responsible person, Donald L. Simon, organization code 2550, (216) 433-3740.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 08

This publication is available from the NASA Center for Aerospace Information, (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Preliminary results are presented on the development of an adaptive neural network based control algorithm to enhance aircraft engine performance. This work builds upon a previous National Aeronautics and Space Administration (NASA) effort known as Performance Seeking Control (PSC). PSC is an adaptive control algorithm which contains a model of the aircraft's propulsion system which is updated on-line to match the operation of the aircraft's actual propulsion system. Information from the on-line model is used to adapt the control system during flight to allow optimal operation of the aircraft's propulsion system (inlet, engine, and nozzle) to improve aircraft engine performance without compromising reliability or operability. Performance Seeking Control has been shown to yield reductions in fuel flow, increases in thrust, and reductions in engine fan turbine inlet temperature. The neural network based adaptive control, like PSC, will contain a model of the propulsion system which will be used to calculate optimal control commands on-line. Hopes are that it will be able to provide some additional benefits above and beyond those of PSC. The PSC algorithm is computationally intensive, it is valid only at near steady-state flight conditions, and it has no way to adapt or learn on-line. These issues are being addressed in the development of the optimal neural controller. Specialized neural network processing hardware is being developed to run the software, the algorithm will be valid at steady-state and transient conditions, and will take advantage of the on-line learning capability of neural networks. Future plans include testing the neural network software and hardware prototype against an aircraft engine simulation. In this paper the proposed neural network software and hardware is described and preliminary neural network training results are presented.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Adaptive control; Optimal control; Neural nets; Engine control; Aircraft engines	14
	16. PRICE CODE: A03

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