NASA Technical Memorandum 112198

Direct Adaptive Aircraft Control Using Dynamic Cell Structure Neural Networks

Charles C. Jorgensen, Ames Research Center, Moffett Field, California

May 1997

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

DIRECT ADAPTIVE AIRCRAFT CONTROL USING DYNAMIC CELL STRUCTURE NEURAL NETWORKS

Charles C. Jorgensen
Computational Sciences Division
Ames Research Center

SUMMARY

A Dynamic Cell Structure (DCS) Neural Network was developed which learns topology representing networks (TRNs) of F-15 aircraft aerodynamic stability and control derivatives. The network is integrated into a direct adaptive tracking controller. The combination produces a robust adaptive architecture capable of handling multiple accident and off-nominal flight scenarios. This paper describes the DCS network and modifications to the parameter estimation procedure. The work represents one step towards an integrated real-time reconfiguration control architecture for rapid prototyping of new aircraft designs. Performance was evaluated using three off-line benchmarks and on-line nonlinear virtual reality simulation. Flight control was evaluated under scenarios including differential stabilator lock, soft sensor failure, control and stability derivative variations, and air turbulence.

BACKGROUND

In 1994, the NASA Advanced Programs Office began research on real-time neural adaptive flight control of damaged aircraft. In a joint NASA/industry initiative, NASA Ames Research Center and McDonnell Douglas Aircraft Corporation initiated a four-year program with a goal of flight demonstrating a concept for identification of aircraft stability and control derivatives using neural networks. The technology development was focused on optimization of aircraft performance under nominal, off-nominal, and simulated accident scenarios, as well as speeding up of controller software development for new designs.

In 1995, Leavenberg-Marquardt perceptrons (ref. 1) were trained on large aerodynamic data sets covering the full flight envelope of an F-15. The neural networks were incorporated in the third flight channel of a specially modified test aircraft (fig. 1). This was to provide the existing flight controller with the best available estimates of aircraft aerodynamics across the flight envelope and at the same time produce a dramatically more compact way to store the stability and control derivatives. The second phase of this program, begun in 1996, emphasized real-time, on-line learning networks capable of handling changes in aircraft flight properties as a result of off nominal conditions such as the loss of a wing, ice accretion, mid-air collision, or actuator/control surface failure. Attainment of this goal requires an integrated treatment of both system identification and control techniques. In this paper we emphasize the control method.

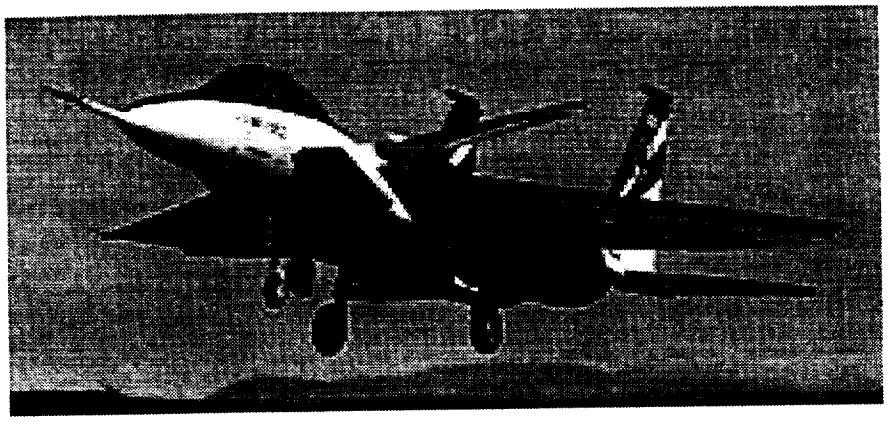


Figure 1. F-15 ACTIVE aircraft.

In 1994, Kim and Calise (ref. 2) proposed a method for F-18 flight control where neural networks were employed to represent the nonlinear inverse transformations needed for feedback linearization. Totah (ref. 3) used this approach to control a simulated nonlinear F-15 model developed. His model represented a modified F-15 aircraft stationed at NASA Dryden Flight Research Facility called the Advanced Control Technology for Integrated Vehicles (ACTIVE) aircraft. This aircraft has specially added canards and vectored thrust capability to permit simulation of abnormal flight conditions, such as, partial loss of wing surface or yaw moments caused by side panel damage. The aircraft was also changed to permit a third flight computer channel in which neural adaptive flight controllers could be embedded.

In 1996, ACTIVE demonstrated the capability of a neural network to successfully learn aircraft aerodynamic coefficients with sufficient accuracy to drive the existing robust controller. However, real-time, on-line learning of dramatic changes to the aircraft model or accidents exceeding design robustness limits required additional developments in neural learning algorithms. This paper discusses recent results for one on-line neural network candidate that appears fast, compact, and accurate enough to serve that function.

Perfect Typology Preserving Networks (TPN)

A number of computer science and mathematics problems can be efficiently solved by reducing descriptions to a geometric representation of clusters of points and their proximity neighborhood in a geometric space. Examples are nearest neighbor or k-nearest neighbor searches, data interpolation, generation of Euclidean minimum spanning trees, or the solution of finite element problems. The general nature of proximity problems are discussed in greater detail in Preparata and Shamos (ref. 4) and Knuth (ref. 5).

In 1994, Martinetz and Schulten (ref. 6) showed that a competitive Hebbian adaptation rule can be used to learn a Voronoi diagram and its dual, the Delaunay triangulation. The Delaunay triangulation is defined as the straight line dual of a Voronoi diagram cast in an embedding space $\Re^D$ of arbitrary dimension D.

The Voronoi diagram V_S of a set $S=\{w_1, \ldots w_n\}$ of points, $w_i \in \Re^D$ is produced by N, D-dimensional polyhedra V_i composed the set of points $\mathbf{v} \in \Re^D$ closer to w_i than to any other point $w_j \in S, j \neq i$.

Specifically, a Voronoi region for point i is defined as:

$$V_i = \{ v \in \Re^D \mid \| \mathbf{v} - w_i \| \leq \| \mathbf{v} - w_j \| \; \forall \; i,j = 1,\ldots N \}$$

where $\| ( ) \|$ is the Euclidean norm.

In neural network literature V_i is often referred to as the receptive field of neuron i where w_i is defined as a weight vector for the neuron. For all input vectors (i.e., patterns) $\mathbf{v} \in V_i$, neuron i is defined as the best matching unit (bmu) of that field and serves as the encoding location to represent that region of points.

In $\Re^2$ space, the Delaunay triangulation in turn is generated by connecting all pairs $w_i, w_j \in S$ for which the corresponding Voronoi polygons V_i and V_j share an edge. The Delaunay triangulation D_S of $S=\{w_1, \ldots w_n\}$, $w_i \in \Re^D$ is defined as a graph whose vertices are the w_i and an associated adjacency matrix **A** whose elements are

$$A_{ij} \in \{ 0 , 1\} \; \forall \; i,j = 1,.. N \text{ such that } A_{ij} = 1 \text{ iff } V_i \cap V_j \neq \varnothing.$$

Two vertices w_i, w_j are therefore connected by an edge if their Voronoi polyhedra V_i, V_j are adjacent. **A** then represents the connectivity status of arbitrary nodes w_i and w_j.

Finding the Delaunay triangulation is important for neural networks, because it can be added to a learning rule to provide an efficient representation for solving many of the above proximity problems. Among possible triangulations of a point set, the Delaunay triangulation is special because it has been shown to be optimal for function approximation (ref. 7).

Importantly, computation time can also be reduced. For example, time required to find a Euclidean minimum spanning tree can be reduced from O(N log N) to O(N) because the edges of the tree are a subset of the edges of the Delaunay triangulation. Nearest neighbor and k- nearest neighbor search times, a major limiting factor of many SOM and radial basis neural network programs, can be reduced from O(N) to O(log N) time (ref. 5). Martinetz refers to a network having this property as a TPN.

Growing Networks

Demonstrating that learning a perfect TPN of a function facilitates recall accuracy, Fritzke (ref. 8) proposed a modification to self-organizing maps (SOFM), (ref. 9) using typology preservation called a Growing Cell Structure (GCS) Network. GCS, along with computationally related networks (e.g., incremental self-organizing networks (GSOM), (ref. 10), Growing Grid Nets (ref. 11) and Growing Neural Gas Networks (ref. 12)), all share certain common characteristics (ref. 13) which can be categorized by:

Network Architecture. Usually a graph (directed or otherwise) consisting of nodes (neurons), links (connection strengths), and topology. A neuron is usually indexed (e.g., w_i above) as a point in R^D.

Point Movement. Where learning effects both the closest neuron relative to a training point and its nearest neighbors in w space. A common method to change w_i is:

$$\Delta w_{bmu} = \xi_{bmu}(\tau - w_{bmu}) \text{ for the closest point, i.e., best matching unit}$$
$$\Delta w_i = \xi_{other}(\tau - w_i) \; (\forall i \in Nw_{bmu})$$

where:

Nw_{bmu} are the topological neighbors of w_{bmu}
ξ are adaptation gains with $\xi_{bmu} >> \xi_{other}$
ξ may also be a function $f(x) = \xi$ of some other variable x such as a time step index

Local error. Where Ew_{bmu} is a local error magnitude accumulated by winning nodes (w_{bmu}). Its value depends upon the metric used. This is often the Euclidean norm

$$\Delta Ew_{bmu} = || w_{bmu} - \tau ||^2$$

where τ is an arbitrary member of a training set. Alternate metrics such as information, entropy, and subjective value can also be used. In contrast to Backpropagation, differentiability of this error metric is not required, permitting measures to be used at a researcher's discretion.

Network growth. New nodes are added based on the amount of reduction of accumulated error within a topological region of w_{bmu}. The different variations deal differently with error reduction during learning and temporal record keeping. Some variations discard error values after insertion, others maintain a decaying history of error along the lines of temporal difference methods. In the current paper we chose to discard error information prior to the immediate learning step.

Topological constraints. Constraints can be placed on neighborhood shapes to reduce combinatorial load complexity as the dimension of problems increase. GCS for example, uses k-dimensional hyper-tetrahedrons (e.g., lines, triangles, or tetrahedrons) as approximating regions. Growing Grids use hyper-rectangles or k-dimensional hypercubes. In this paper shape is constrained by the local connection rule.

Recall and estimation. When a topology has been learned for a function, a reconstruction rule is used to estimate values for untrained points or to determine where new nodes are added. Nearest neighbor, or radial basis approaches are often chosen. However, there is opportunity to exploit the topology only if the network can be shown to produce an accurate topological representations. In the SOFM a mapping is usually from $R^n \rightarrow R^2$. If $R^n \rightarrow R^m$ a perfect topology preserving mapping is usually not possible since $R^n \rightarrow R^m$ may be a non-orthogonal projection onto a two-dimensional manifold. Hence, recall accuracy for new points suffers as the dimension mismatch increases.

Dynamic Cell Structures

Bruske and Sommer (ref. 14) developed a synergistic combination of the above principles they called Dynamic Cell Structures. They used insertion of nodes based on accumulated error proposed in (ref. 15) and Martinetz's use of competitive Hebbian learning to adjust connection strengths while at the same time preserving topological mapping constraints.

Their network chose a radial basis function representation, a subset of Martinetz's TPR network, to concurrently learn and use perfectly topology preserving feature maps. The network applied Hebbian learning to adjust topological connections and a Kohonen-like learning rule to adjust node positions during training. It was capable of both supervised and unsupervised learning.

Neighborhood topology is used to determine which nodes are activated during recall or new value estimation. Activation levels are determined by radial basis proximity. The network grows, starting with two nodes and adds new nodes sequentially in areas in which ΔEw_{bmu} is maximum. Over time, the effect is that error becomes homogeneously distributed over an entire point density. Training continues until ΔEw_{bmu} reaches a predetermined criteria. Formally, the DCS network is defined as follows:

Given an input manifold $I \subset \Re^n$ and output manifold $O \subset \Re^m$ n usually > m, a DCS network is a collection of points:

$n = (c,w,R,y) \in I \times O \times [0,1] \times \Re_{\geq 0}$ where:
- c the center of a point in $\Re^n$
- w a weight in I associated with the point
- R a function weighting the influence of n as a function of distance
- y an error value associated with the estimation value of n as a graph

G = (N,L, S) where:
- $N \subset I \times O \times [0,1] \times \Re_{\geq 0}$ is a set of nodes
- $L \subset \{\{a,b\} \mid a,b \in N \text{ and } a \neq b\}$ are links between nodes
- $S : L \rightarrow \Re_{\geq 0}$ is a lateral connection strength function of an adjacency
- Matrix C, $C \in \Re^{|N| \times |N|}$ for which:
  - $C_{ii} = 0 \; \forall \; i \in \{1 \ldots |N|\}$
  - $C_{ij} = 0 \; \forall$ i,j not connected
  - $0 \leq C_{ij} \leq 1$ if i,j connected with strength C_{ij}, and

A connection strength function S is defined using a Hebbian rule:

$$C_{ij}(t+1) = \begin{bmatrix} \max(y_i \bullet y_j, C_{ij} : & \geq y_k \bullet y_l & \forall (1 \leq k,l \leq |N|) \\ 0 : & C_{ij}(t) < \theta & \forall (1 \leq k,l \leq |N|) \\ \alpha C_{ij} & : \quad otherwise & \end{bmatrix}$$

where α, $0 < \alpha < 1$, is a forgetting constant

θ, $0 < \theta < 1$, is a deletion threshold for weak lateral connections

$y_i = \frac{\|\tau - w_i\|}{\sum_u \|\tau - w_i\|}$ i.e., for this paper's network y_i is a normalized distance to τ

w_i is the center of a neuron's receptive field and

τ is a training pattern's coordinate location in $I \subset \Re^n$.

DCS Modifications for F-15 Learning

In the present work, R in (ref. 15) was replaced with a context sensitive linear interpolation Γ calculated between topological neighbors. Γ was found to be more accurate for the estimation of F-15 aerodynamic coefficients (~62%). Point locality tests were needed to avoid excessive errors if neighborhoods were incompletely learned under real-time constraints. If we let Ω be the network's output given τ then:

if $||\tau - w_{bmu}|| < ||\tau - w_{2cd\ bmu}|| > ||w_{bmu} - w_{2cd\ bmu}||$

$\Omega = \Psi(w_{bmu})$

else $\Omega = \Gamma(w_{bmu}, w_{2cd\ bmu})$ where $\Gamma = \gamma_1\Psi(w_{bmu}) + \gamma_2\Psi(w_{2cd\ bmu})$.

γ_i interpolated for each dimension of w_{bmu} and $w_{2cd\ bmu}$. $\Psi(\ )$ is the output value associated with point n at coordinates w. $\Gamma(w_{bmu}, w_{bmu})$ is an interpolation between $\Psi(\ )$ values associated with the neurons in the topological neighborhood of τ chosen based on the magnitude of their cumulative node errors ΔEw_{bmu} and $\Delta Ew_{2cd\ bmu}$.

This operation is summarized as follows. Given a new input value τ_{input} the output Ω associated with τ_{input} is estimated by finding the closest DCS node w_{bmu} and its nearest topological neighbor $w_{2cd\ bmu}$ to τ_{input}. If the distance between τ_{input} and $w_{2cd\ bmu}$ is not less than the distance between w_{bmu} and w_{2cdbmu} (the second closest node to τ_{input} within the connected topological neighborhood of w_{bmu}) a case is present in which the network does not have a connected neighbor on the "other side" of τ_{input}. This can occur during incomplete function learning or excessively high values of α. Linear interpolation (or nonlinear estimates such as B splines) using local topology would otherwise return an artificially inflated estimate for Ω. To avoid this situation, Γ ignores $w_{2cd\ bmu}$ and defaults to a nearest neighbor estimate.

Weight Change

Learning uses a standard Kohonen-like rule in which w_{bmu} and its topological neighbors are adjusted according to:

$$\Delta w_{bmu} = \varepsilon_{bmu}(\tau - w_{bmu})$$
$$\Delta w_{N(j)} = \varepsilon_{N(j)}(\tau - w_{n(j)})$$

Where N(j) of unit j is defined as

$$N(j) = \{i \mid (C_{ji} \neq 0,\ 1 \leq I \leq N)\}$$

Only the best matching unit and only its positively connected topological neighbors are adjusted during each learning cycle.

Adding Nodes

Nodes are added incrementally to the network during training. They are selected from areas with maximum estimation error. New nodes are placed between a node having the highest error and its topological neighbor having the second highest error. The location of the center of the receptive field for a new node w_{new} is calculated according to a ratio of error values ΔEw_{bmu} and $\Delta Ew_{2cd\ bmu}$ where ΔEw_{bmu} and $\Delta Ew_{2cd\ bmu}$ are the nodes with the highest and second highest error in the topological neighborhood respectively. The ΔE of units w_{bmu} and $w_{2cd\ bmu}$ are redistributed among w_{new}, $w_{2cd\ bmu}$ and w_{bmu}. The equations defining this distribution are as follows:

define: $n1 = Ew_{bmu}$
define: $n2 = Ew_{2cd\ bmu}$ then

$\gamma = n1 / (n1 + n2)$
$\Delta n1 = .5\ (1 - \gamma)n1$
$\Delta n2 = .5\ \gamma n2$
$w_{new} = w_{bmu} + \gamma\ (w_{2cd\ bmu} - w_{bmu})$
$Ew_{new} = \Delta n1 + \Delta n2$
$Ew_{bmu} = Ew_{bmu} - \Delta n1$

These equations redistribute error equally among the three nodes and reduce overall error level in the region of maximum deviation from the true function distribution. The connection strengths of A are then adjusted. This is accomplished by setting:

$C\ w_{new} \rightarrow C\ w_{2cd\ bmu} = 1$
$C\ w_{new} \rightarrow C\ w_{bmu} = 1$
$C\ w_{bmu} \rightarrow C\ w_{2cd\ bmu} = 0$

where $\rightarrow$ represents the connection between nodes.

Feedback Linearized Control

Feedback linearization is a method used to design stable, nonlinear control systems. It operates by transforming a nonlinear plant in such a way that the transformed system exhibits linear dynamics about an operating condition. After this is accomplished, controller design methods such as PID, LQR, Eigen structure assignment, Mu synthesis, or H_∞ can be used to create a robust controller for the linearized system. The key to usability is being able to invert transformed controller outputs back into the original coordinate reference system during execution. This is often difficult because complete understanding of aircraft dynamics is needed to perform the inversion. Such information is often not available, particularly if the operational environment causes sudden changes to aircraft flight characteristics. This could occur due to unexpected events such as ice accretion or more drastic effects caused by severe damage such as mid-air collision or explosion.

In this paper, we are concerned with developing an efficient neural network method for on-line learning of a plant inverse that could be used in control augmentations such as feedback linearization. There are two main situations which one might encounter. First, a control paradigm may be so robust that even with changes in physical characteristics of the plane, the output commands are still within a stable control envelope. In this case, on-line learning would not be necessary. In the second, extreme changes such as loss of a wing would completely invalidate an *apriori* plant model. In this case, an entirely new model would have to be learned on-line in real-time. The latter case will be addressed in greater detail in a follow-on to this paper where the focus is system identification.

In the first case, although the plant model after an accident might not correspond well to a pretrained model, the robustness of neural network methods are still sufficient to deal with some severe accident scenarios. Indeed, it has been shown that incorporating a neural plant model into a feedback linearization design is very effective for stabilizing and controlling aircraft. Sanner and Slotine (ref. 16) proposed a method for direct adaptive control using Gaussian Networks. They showed that with some assumptions about nonlinearities, Lyapunov methods could be used to prove convergent tracking errors for the neural weight adjustment algorithm.

Calvet (ref. 17) and Kim (ref. 18) proposed a method by which MLP neural networks can be integrated into a direct adaptive tracking controller using feedback linearization. Their controller approach, and the one used in this work, is composed of a command augmentation system (CAS) with two parts, an inner loop responsible for airframe stabilization given commanded rates, and an outer loop that tracks pilot commands. A multi-layer perceptron is used to perform the model inversion needed by the CAS. Kim demonstrated that a stability augmentation system based on feedback linearization could be developed for the equations of motion of an AH-64 Helicopter and a 6DOF model of the F/A-18. He also showed a neural network can be used to perform inverse transforms required to calculate the control signals, and a second on-line neural network can be used to correct for inversion errors in the first network resulting from a non exact inversion of the coordinate transformation matrix.

Our lab extended this work in the following ways. In 1995, Eshaghi demonstrated that for a full nonlinear 6DOF F-15 aircraft simulation, neural learning accuracy could be markedly improved for the off-line model by using a Leavenberg-Marquardt perceptron (ref. 1) to achieve lower RMS error rates for the aircraft derivatives. Totah (ref. 3) showed in 1996 that the Calise/Kim method could maintain F-15 system stability for a 6DOF nonlinear model under demonstrated failures of differential canard lock and soft sensor failures.

The drawback of Leavenberg-Marquardt (LM) learning was the speed and memory requirements required since off-line LM training, although accurate, proved to be very resource intensive due to the data matrix inversions required. The F-15 ACTIVE onboard flight processor (OFP) is actually very limited in memory, thus compactness becomes vital for real world flight test demonstration. This was attained by LM methods in 1995, but on-line learning required too many resources to use that method on-line.

We realized early on that the on-line part of the program would probably require some form of basis function learning technique to be fast enough. In 1996, Jorgensen, Kim, and Fuji (internal white paper) began development of a modified DCS network to take advantage of the perfect topology

representation properties and increased compactness of the DCS network. With the addition of an improved output estimation method, this network showed extremely fast learning with a markedly more compact data structure than CMAC architectures tested earlier. Under conditions of heavy turbulence introduced into the training data, a combined DCS/Feedback Linearized controller exhibited impressive stability. We now discuss the network's performance in more detail.

EXPERIMENTS

Duffing Equation

The modified DCS network was tested on a set of benchmarks to verify learning performance prior to training on the F-15 data sets. Several functions were evaluated to check the response of the network to conditions which might be encountered in control system learning. To test chaotic sequences, the Duffing equation, representative of many control problems was used. We trained a particularly messy parameterization (fig. 2) defined as:

$$y = x'' + bx' + k_1 x + k_2 x^3 = a\cos(\omega t)$$

where

$$k_1 = 0$$

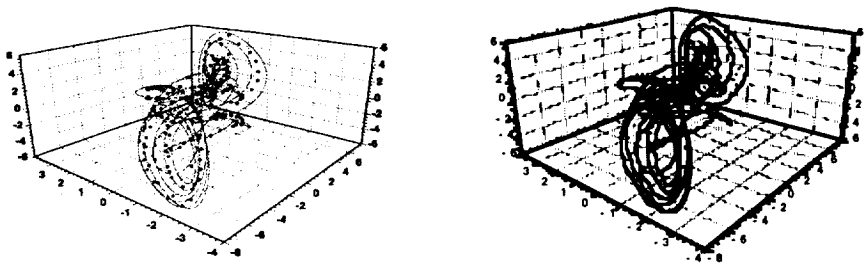
$$k_2 = 1$$

$$b = 0.05$$

$$a = 7.5$$

$$\omega = 1$$

chosen to induce chaotic behavior in the phase trajectory.



2(a). Duffing Equation 5% RMS Error, 200 Nodes. 2(b). Duffing Equation 1% RMS Error, 400 Nodes.

Figure 2. Phase plot of DCS learned Duffing equation.

The net was allowed to grow to 200 points (equivalent to about 2.5 seconds of aircraft sensor recording time at 80 Hz). This resulted in about 5% normalized RMS learning error. Point estimation for new untrained values used the Γ function and are shown as dots in figure 2(a). As can be seen in figure 2(b) using 400 points, the complex manifold is being captured by the DCS network. The

greatest error shows up in the most recent (outer) portions of the phase trajectory. This is because there were less Kohonen training iterations on the average for the most recent points as the stopping criteria was reached.

CMU Two Spirals Benchmark

The CMU two spirals benchmark tested by Bruske and Sommer (fig. 3) was trained using 1000 of 1200 normalized values. We did not find their suggested rate equation ($\alpha = \sqrt[n]{\theta}$ where n is the number of training samples) effective because it resulted in insufficient severing of extraneous connections between units. We were able to replicate results using fixed parameter values at:

$\alpha = .7$
$\theta = .8$
$\varepsilon_{bmu} = .99$
$\varepsilon_{N(u)} = .006.$

Bruske, et. al., did not detail their error measure for the two spirals benchmark. We used mean point error as a percentage of range. We achieved values close to zero, i.e.,<.0001 generalization error for the randomly chosen test set of 200 untrained spiral points. The results appear comparable to their paper.

Table 1. DCS parameters for two spirals benchmark.

DCS Rule	Bmu constant	# nodes for error <.0001
Bruske	$\alpha = \sqrt[n]{\theta}$	196
Ames α	0.1	213
Ames α	0.99	276

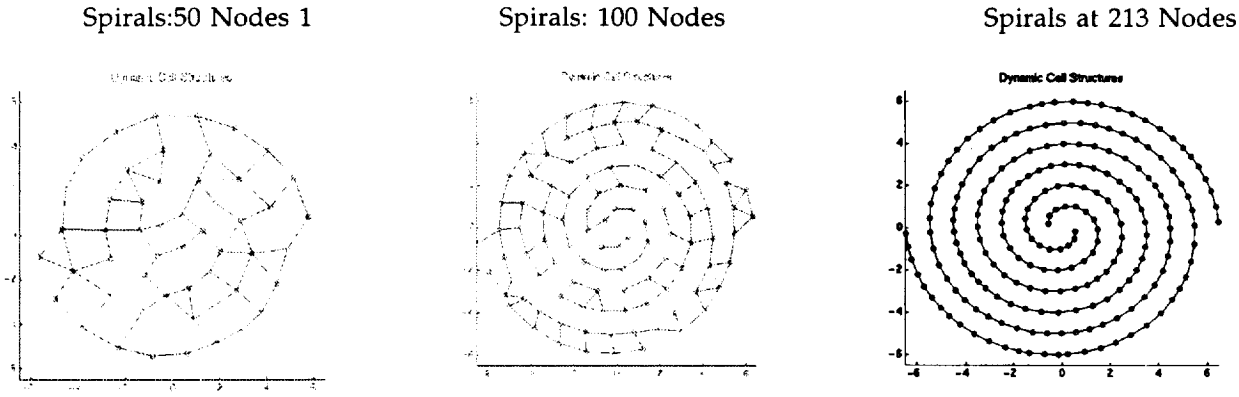


Figure 3. Two spirals benchmark.

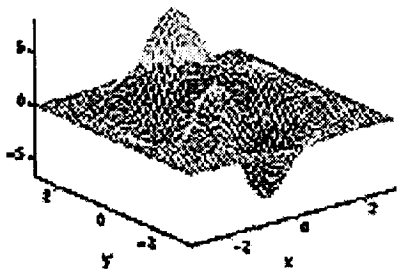
Matlab Peaks Equation

The Matlab peaks equation (figure 4)

$$z = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(x/5 - x^3 - y^5)e^{-x^2-y^2} - 1/3e^{-(x+1)^2-y^2}$$

was trained to 3% error as a two input one output (MISO) mapping test. This equation was chosen to evaluate recall with both multiple minima and exponential functions. Results were satisfactory with 613 nodes required to achieve 3% (our required learning level) with an original training set size of 900 nodes. The task also illustrated a nice property of the DCS net, that is the ability to pre-select accuracy levels.

Peaks Distribution



DCS Net Learning at 3% Error Level

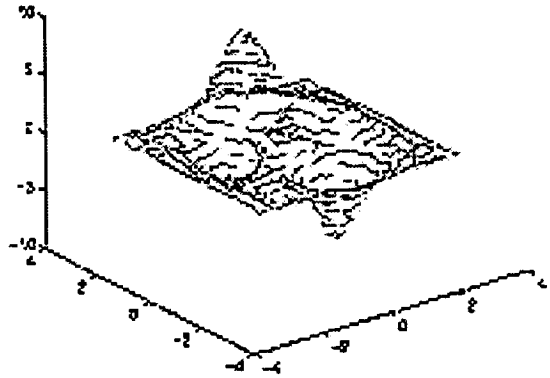


Figure 4. DCS peaks learning.

F-15 ACTIVE Stability and Control Coefficients

Single Maneuver Tests. To test generalization of this result to aircraft data, the DCS net was trained on 32 F-15 ACTIVE stability and control derivatives for a single 30 second full power banked turn maneuver from Mach .6 to Mach 1.1 and 9,000 to 11,000 ft. altitude (fig. 3). As can be seen from the graph showing effects of the change in angle of attack α, the network was able to effectively learn the $C_{m\alpha}$ values using only 34 nodes. The same network predicted the other 31 derivatives equally well. A full test of the network required demonstration of coefficient learning across the entire flight envelope. In addition, real world conditions needed to include sensor noise as well as atmospheric turbulence during periods of on-line learning. In our current implementation, the DCS network receives on-line training data through the intermediary of a ring buffer set to filter redundant sensor information from the training data set. This is required to avoid a stability plasticity dilemma of over training during periods of slow parametric changes.

Full Flight Envelope Tests. The single maneuver performance above demonstrated adequate multi-input single output learning. The full F-15 ACTIVE flight test envelope composed of 2752 points (i.e., altitude and Mach vs. 32 aerodynamic coefficients (ref. 3) was trained to demonstrate multi-input multi-output behavior. Three percent accuracy had been previously determined off-line to be sufficiently accuracy for stable ACTIVE control using the feedback linearized control architecture.

The training envelope ranged from Mach .3 to Mach 2.0 with altitude from sea level to 50,000 ft. The network achieved 3% error using 63 nodes, 179 seconds on a SG Indy work station and the Γ recall metric. In contrast, 85 nodes and 268 seconds were required to achieve the same level of accuracy using a nearest neighbor recall rule during DCS training. The results demonstrated a clear advantage in both compactness and speed utilizing Γ. It is anticipated that given minimum computational penalties, a b-spline or higher order nonlinear interpolation scheme may provide still better results.

Overall network control performance is shown in figures 5(a)–5(d) for the 30 second power banked turn using the full envelope trained network. In figure 5(a), the maneuver is plotted in terms of lateral and longitudinal stick movement in inches over a time interval of 30 seconds. From 1 to 9 seconds, .5 inches of lateral stick are applied resulting in a right bank at a roll rate of about 10 degrees/second with speed accelerating from Mach .6 to about Mach .8. At 10 seconds 2.1 inches of positive longitudinal stick are applied for 20 seconds resulting in an accelerating descending turn with altitude dropping to 9.2 thousand feet and increased lift resulting in recovery back to 9.8 thousand feet. The maneuver pulls approximately positive 4 Gs. The lower graph shows the behavior of the DCS network versus a perfect plant model of the aircraft. As can be seen, the DCS and plant overlap for $C_{m\alpha}$. Similar accuracies were obtained for the other 31 derivatives.

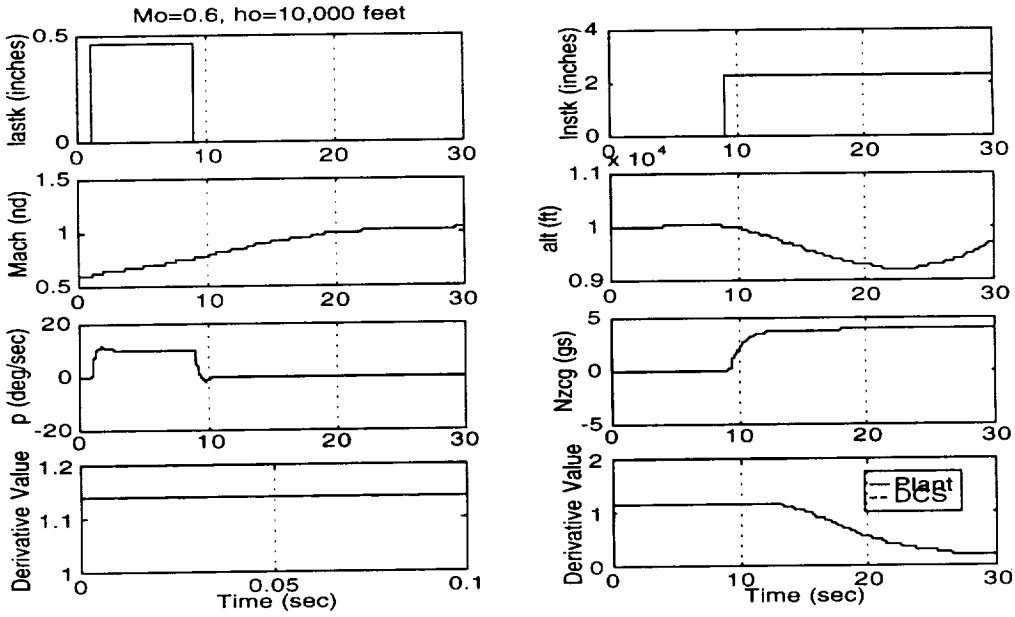


Figure 5(a). Learning a maneuver.

In figure 5(b) the model of the aircraft is abruptly changed by 20% uniformly across all 32 derivatives. The DCS network is permitted to learn on-line using as a basis the trained network of figure 5(a). The result is seen on the $C_{m\alpha}$ time plot. After initial convergence, the DCS net rapidly damps and within .05 seconds tracks the new changed plant model exactly. This perturbation corresponds to an extreme change in the aircraft model as would have occurred in a significant accident.

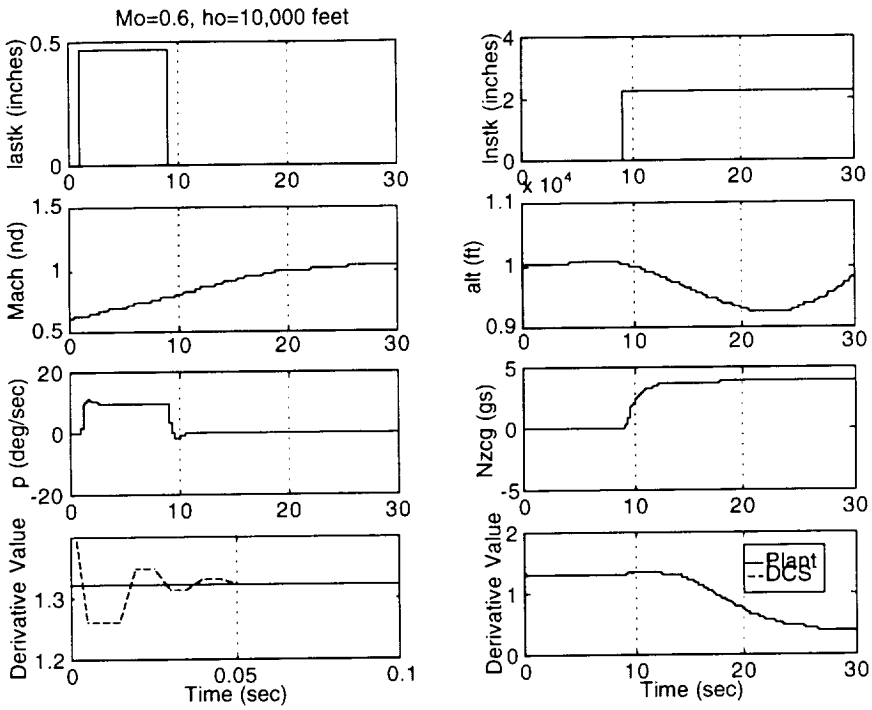


Figure 5(b). Learning parameter changes.

Figure 5(c) repeats the experiment except heavy turbulence as defined by the Dryden turbulence model is added to the training data. The model quickly damps and again converges within .5 seconds. Line jitter reflects system turbulence not learning error of the DCS controller.

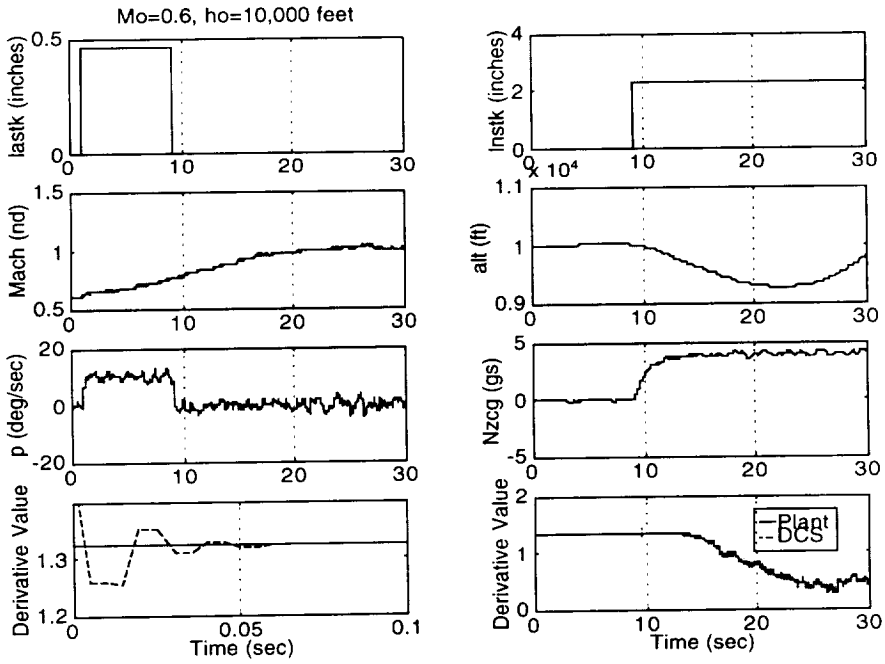


Figure 5(c). Learning with turbulence.

Figure 5(d) illustrates tracking behavior more clearly by an extreme blow up of a half second time interval. As can be seen DCS again tracks well.

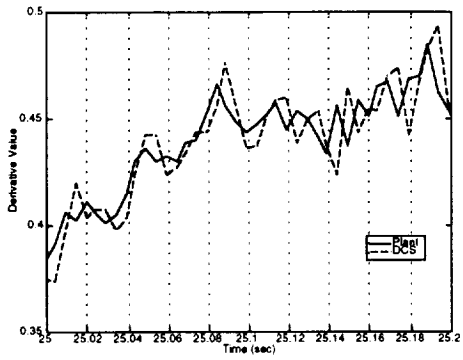


Figure 5(d). Enlarged DCS learning.

VR Simulation

Adequate numeric performance does not guarantee adequate pilot feel for an on-line controller. To facilitate pilot testing prior to flight tests, the Ames Neuro Engineering Lab designed a virtual reality simulator (fig. 6). Neural networks coded in C++ were integrated with aircraft plant models, a physical environment including turbulence and dynamic graphics rendered on a SG Reality Engine II workstation. This mixture permits comparative evaluation of controller performance, as well as a platform to test new visualization tools. In the near future, the DCS simulation will be incorporated into a VME-based Neuro controller board designed for hypersonic flight control. This will allow additional speed trade-off studies between software and hardware network implementations.

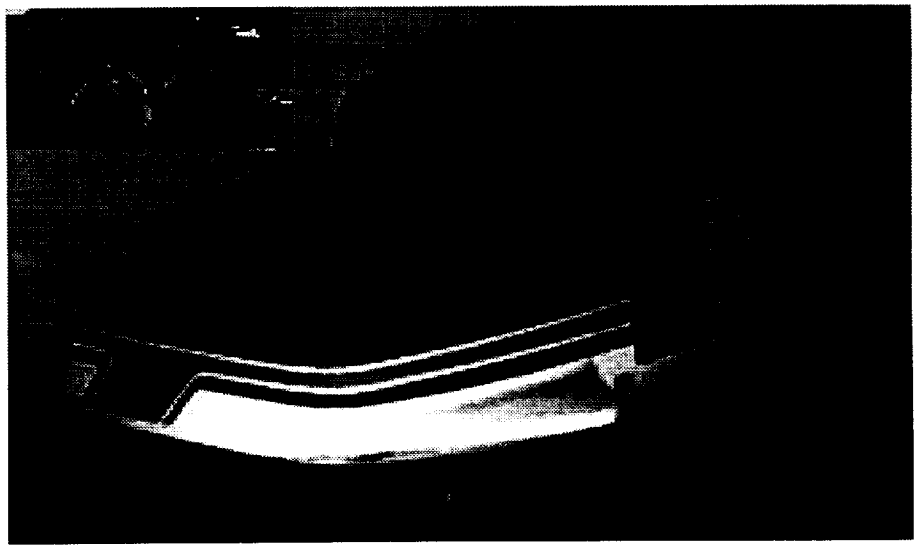


Figure 6. Ames virtual reality simulator.

CONCLUSIONS

This paper has demonstrated some advantages of combining dynamic cell structure networks with real-time flight controllers. A modified DCS network was incorporated into a feedback linearized control augmentation system and demonstrated using a piloted VR F-15 simulation. Gains from perfect topology preserving networks were shown and the network performance was tested through a full flight envelope. The network appears promising and will be further evaluated through flight tests on a modified F15 at Dryden Flight Research Center in 1997. A combination of typology preserving networks and nonlinear control system architectures appears to be a productive direction for development of stable, rapidly adapting nonlinear controllers. Future work will continue to develop feedback linearization as well as generalized predictive control and adaptive critic architectures.

REFERENCES

1. Nørgaard, M.; Jorgensen, C.; and Ross, J.: "Neural Network Prediction of New Aircraft Design Coefficients." NASA TM-112197, May 1997.

2. Kim, B.S.; and Calise, A.J.: "Nonlinear Flight Control Using Neural Networks," AIAA Paper 94-3646-CP, 1994.

3. Totah, J.: "Simulation of a Neural Based Flight Controller," AIAA Paper, AIAA-96-3503, 1996.

4. Preparata, F.P.; and Shamos M.I.: *Computational Geometry: An Introduction*, New York: Springer-Verlag, 1988.

5. Knuth, D.E.: *The art of computer programming. Volume III Sorting and Searching*, Addison-Wesley, 1993.

6. Martinetz, M.; and Schulten, K.: "Topology Representing Networks," Neural Networks, Vol. 7. No. 3, pp. 507-522, 1994.

7. Omohundro, S.M.: "The Delaunay triangulation and function learning," TR-90-001, International Computer Science Institute, Berkeley, 1990.

8. Fritzke, B.: "Growing cell structures–a self-organizing network for unsupervised and supervised learning," *Neural Networks*, 7(9):1441-1460, 1994.

9. Kohonen, T.: "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, 43:59-69, 1982.

10. Bauer, H.U.; and Villman, Th.: "Growing a hypercubical output space in a self-organizing feature map," TR-95-030, International Computer Science Institute, Berkeley, 1995.

11. Fritzke, B.: "Growing grid–a self-organizing network with constant neighborhood range and adaptation strength," Neural Processing Letters, 2(5):9-13, 1995.

12. Fritzke, B.: "Let it grow–self-organizing feature maps with problem dependent cell structure." *Artificial Neural Networks,* pp 403-408. North-Holland, 1991.

13. Fritzke, B.: "Growing Self-organizing Networks - Why?" In M. Verleysen, ed. ESANN96, D-Facto Publishers, Brussels, pp. 61-72, 1995.

14. Bruske, J;. and Sommer, G.: "Dynamic Cell Structures," NIPS, 1996, p. 497-504.

15. Martinetz, T.M.: "Competitive Hebbian learning rule forms perfectly topology preserving maps," ICANN93, pp. 427-434, Amsterdam, 1993. Springer.

16. Sanner, R.M.; and Slotine, J.J.E.: "Gaussian Networks for Direct Adaptive Control Transactions on Neural Networks," IEEE Vol. E, No. 6, 1992, pp. 837-863.

17. Calvet, J.P.: "A Differential Geometric Approach for the Nominal and Robust Control of Nonlinear Chemical Processes." Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, 1989.

18. Kim, B.S.: "Nonlinear Flight Control Using Neural Networks," Ph.D. Thesis, Georgia Institute of Technology, School of Aerospace Engineering, 1993

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1997	Technical Memorandum

4. TITLE AND SUBTITLE

Direct Adaptive Aircraft Control Using Dynamic Cell Structure Neural Networks

5. FUNDING NUMBERS

519-30-12

6. AUTHOR(S)

Charles C. Jorgensen

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

Ames Research Center, Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

A-976719A

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

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-112198

11. SUPPLEMENTARY NOTES

Point of Contact: Charles C. Jorgensen, Ames Research Center, MS 269-1, Moffett Field, CA 94035-1000 (415) 604-6725

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified-Unlimited
Subject Category–01, 03

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

A Dynamic Cell Structure (DCS) Neural Network was developed which learns topology representing networks (TRNs) of F-15 aircraft aerodynamic stability and control derivatives. The network is integrated into a direct adaptive tracking controller. The combination produces a robust adaptive architecture capable of handling multiple accident and off-nominal flight scenarios. This paper describes the DCS network and modifications to the parameter estimation procedure. The work represents one step towards an integrated real-time reconfiguration control architecture for rapid prototyping of new aircraft designs. Performance was evaluated using three off-line benchmarks and on-line nonlinear Virtual Reality simulation. Flight control was evaluated under scenarios including differential stabilator lock, soft sensor failure, control and stability derivative variations, and air turbulence.

14. SUBJECT TERMS

Aerodynamic stability, On-line learning, Neural networks, Linearization

15. NUMBER OF PAGES

20

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		