IN-13

206662

21P

NASA Technical Memorandum 106482

Optimal Space Station Solar Array Gimbal Angle Determination Via Radial Basis Function Neural Networks

Daniel J. Clancy and Ümit Özgüner
Ohio State University
Columbus, Ohio

and

Ronald E. Graham
Lewis Research Center
Cleveland, Ohio

(NASA-TM-106482) OPTIMAL SPACE STATION SOLAR ARRAY GIMBAL ANGLE DETERMINATION VIA RADIAL BASIS FUNCTION NEURAL NETWORKS (NASA) 21 p

N94-24482

Unclas

G3/13 0206662

February 1994

Optimal Space Station Solar Array Gimbal Angle Determination Via Radial Basis Function Neural Networks

Daniel J. Clancy and Ümit Özgüner
Ohio State University
Columbus, Ohio 43210

and

Ronald E. Graham
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

The potential for excessive plume impingement loads on Space Station Freedom solar arrays, caused by jet firings from an approaching Space Shuttle, is addressed. An artificial neural network is designed to determine commanded solar array beta gimbal angle for minimum plume loads. The commanded angle would be determined dynamically. The network design proposed here involves radial basis functions as activation functions. Design, development and simulation of this network design are discussed.

NOMENCLATURE

Name	Description
i	index of exemplar
j	index of kernel node (RBF)
k	index of output
m	number of exemplars
p	number of inputs
q	number of outputs
y_{ik}	k^{th} output for i^{th} exemplar
$\mathbf{z}$	$\mathbf{R}\,\Lambda$, solution to least-squares problem
J	cost function for optimization
$\mathbf{C}_j$	centroid vector for jth kernel RBF
$\mathbf{E}$	network error vector (length m)
M	number of kernel neurons
$\mathbf{Q}$	matrix used by least-squares, m x (M+1)
$\mathbf{R}$	matrix used by least-squares, (M+1) x (M+1)
$\mathbf{X}_i$	input vector for i^{th} exemplar
$\mathbf{Y}_i$	output vector for i^{th} exemplar
ϵ_i	network residual error for i^{th} exemplar
η_j	most significant regressor for j^{th} kernel
λ_{0k}	bias on k^{th} output
λ_{jk}	weight between j^{th} RBF and k^{th} output
ϕ_{ij}	$\phi(\lVert \mathbf{X}_i - \mathbf{C}_j \rVert)$, the RBF
ρ	tolerance for regressor selection
σ	smoothing factor for RBFs
Λ_0	row vector of output biases
Λ_j	weight row vector from jth RBF to output
Υ_i	transpose of $\mathbf{Y}_i$
Θ	matrix of radial basis functions (w/biases)

INTRODUCTION

Two photovoltaic solar arrays, such as that shown in Figure 1, provide power for early stage configurations of the Space Station Freedom. These arrays are about 100 feet long and 40 feet wide, but are very thin, thus displaying typical characteristics of large flexible space structures.

The flexibility of these solar arrays makes suppression of dynamic loads difficult. A relatively small force, applied with a large enough angle of attack relative to the array plane and to a sensitive location on the array, could cause the mast to fail [1]. One such force comes from the plume (as shown conceptually in Figure 2) of a Space Shuttle reaction control system (RCS) jet. These jets are fired throughout the Shuttle's approach to Freedom, shown in Figure 3.

The firing of the RCS jets is governed by the following rules of thumb for proximity operations:

(1) The Shuttle stays within an approach cone of given half-angle with apex at the berthing point on the Station [2].

(2) The closure rate is confined to around 0.1% of the closure distance -- the so-called 0.1% Rule [3].

The position of the solar arrays relative to the rest of the Station is controlled by an alpha gimbal (which rotates about the truss axis) and two beta gimbals, one for each array, as shown in Figure 4. Since the alpha gimbal is rotating a significant amount of inertia, and since it has a large amount of static friction to overcome, the beta gimbals are the only practical means for controlling plume angle of attack.

The baseline solution to this problem is the locking of the beta gimbals prior to approach, in a feathered position determined a priori from structural dynamics analysis. A plume impingement may still result if the arrays are not feathered optimally.

The difficulties with casting this problem as one of active beta gimbal control are as follows:

(1) Each RCS firing has some "randomness" associated with it, as they are under human control. The Shuttle pilot chooses not only which jets fire and when, but also for how long.

(2) Thruster plumes and their interaction with solid surfaces are highly nonlinear and geometry-dependent, making their modeling unwieldy for dynamic analysis.

For these reasons, the control problem is difficult to cast into a standard framework -- but the problem lends itself nicely to a neural network solution. Such a solution is desirable because the approach is slow enough that the network could command the arrays with a well-behaved continuous function, avoiding potential problems with residual vibrations [4].

The approach chosen here involves using a neural network, such as shown in Figure 5, to act as an open-loop controller (or, predictor), to determine an optimal commanded gimbal angle (one which should minimize plume angle of attack) for a wide range of approach scenarios. In particular, neurons are chosen that apply radial basis functions as activation functions, because they are well-suited for irregularly positioned data, and because they enable the network to learn faster than if it were trained using backpropagation. The conceptual system design, including placement of the neural network, is shown in Figure 6.

NEURAL NETWORK SOLUTION

Design. The radial basis function (RBF) methodology is useful for dealing with irregularly-spaced data. In this methodology, a linear function space is created which depends on a distance measure between known data points. This design is developed for a single output by Powell [5].

The RBF network depends on the selection of (a) centers for the activation functions and (b) a smoothing factor, which determines the degree of interpolation between known points. If the number of centers selected equals the number of data points, the data will be fit exactly, although this is not a good design from an implementation perspective.

Several methods are commonly used for selecting centers. Lloyd [6] and MacQueen [7] use the standard k-means clustering algorithm as an iterative process over the entire set of training data. MacQueen uses the same algorithm as an adaptive process in real time. A commonly-used method for selecting smoothing factors is the "P nearest-neighbor" method given by Moody and Darken [8].

In this study, the Gram-Schmidt least-squares procedure suggested by Chen et. al. [9] is used to determine the most significant centers from all available candidates, given the type of center chosen (in this case multivariate Gaussian) and a user-selected tolerance used to trade off accuracy and design complexity. The goal of this study was to minimize the number of nodes, while maintaining normalized error below five percent.

A single smoothing factor was assumed, with its initial value determined using a nearest-neighbor criterion. The design then depended only on three parameters: the number of centers desired, the tolerance, and the smoothing factor.

The design procedure is:

(1) Select the smoothing factor (σ) and regressor selection tolerance (ρ).

(2) For the given training data, use one of the centroid selection methods mentioned above, along with either classical or modified Gram-Schmidt orthogonalization.

(3) Compare the RBF response with the desired response, determining error and normalized error.

(4) Repeat steps (1)-(3) as needed until the design goal is achieved.

The centroids of the kernel nodes were selected to correspond with the training data. The initial smoothing factor was chosen to be unity -- actually a good approximation of nearest-neighbor data, shown in Figure 6. This procedure is described in more detail, and the equations describing the network design are given, in the Appendix.

Training. Moody and Darken [8] have demonstrated that "networks of locally tuned units" learn substantially faster than backpropagation neural networks because they have linear learning rules such as least-squares, as indicated above for RBFs, while backpropagation alone requires nonlinear learning rules, such as gradient descent. Having settled on a design configuration for the network, learning rules are simple. In this case, backpropagation was used, and the learning rule was linear.

The training data used for this solution encompassed 500 seconds of simulated Shuttle approach to Freedom, starting from a closure distance of 25 feet. The training data was selected to exploit a priori knowledge of the 0.1% Rule and an assumed 10-degree approach cone, as well as the likelihood of multiple RCS firings in that range. This resulted in 5000 input-output combinations provided by each simulation. The inputs chosen for the network were as follows:

- o closure distance
- o rate of change of closure distance
- o angular position within approach cone (two inputs)
- o rate of change of position within approach cone (two inputs)

Six such simulations were used to generate the total set of training data -- a total of 30000 points. The initial conditions of the simulations were a guess at what conditions would effectively partition the input space:

(1) edge of approach cone, +x direction
(2) edge of approach cone, -x direction
(3) edge of approach cone, +y direction
(4) edge of approach cone, -y direction
(5) approach faster than indicated by 0.1% Rule
(6) approach slower than indicated by 0.1% Rule

More comprehensive sets of training data have since been developed [10], and more work could still be done in this area. This is necessary because there are segments of the approach where there is a significant amount of "switching" between jets that could induce a plume load, and this behavior must be enveloped as much as possible by the training data for a valid design.

If the training data is persistently exciting over the entire input space, retraining of the RBF network is not necessary. Further, the network must only be designed for one of the two beta gimbals, with a modification to output weights for a second network required to command the other.

TEST RESULTS

Testing was performed by implementing the neural network in the simulation used for training, using initial conditions other than those used to generate the training data. Before implementation, more rigorous testing would be expected. The performance of the network is found in terms of the sum of squares of error between network output and known optimal gimbal angle.

For a tolerance of 0.0001, the design process resulted in 260 kernel nodes. That's quite a few, but the test result of this design is very good, as seen in Figure 6, especially in the neighborhood of discontinuities in the input data. The only problem with the response is oscillation near the end of each run. This could result from narrow activation functions employed by the kernel nodes. A design with fewer nodes will of course be less narrow, and dampens oscillation somewhat.

The design process was repeated with smoothing factors of 5, 7.5, 10, 15 and 25; yielding 61, 45, 44, 53 and 41 kernel nodes, respectively, each with a bias. In general, performance near discontinuities degrades as network complexity decreases -- but the oscillation seen in the initial design also goes away.

The results indicate that the "best" smoothing factor (in terms of this study's goal) is somewhere between 5 and 15. The design obtained with a smoothing factor of 10 (44 kernel nodes) was selected as the best for this study, since it had the least complexity and maintained normalized error below five percent except in a small neighborhood of some discontinuities.

The following table shows a comparison of sum of squared error for the series of designs examined:

#RBFs	Sum Squared Error						
	1	2	3	4	5	6	Avg.
260	.81	.70	.40	.58	.26	.32	.51
61	.45	.52	.59	.67	.70	.12	.51
45	.47	.59	.55	.55	.58	.25	.50
44	.70	.62	.57	.61	.38	.16	.51
53	.71	.73	.51	.58	.32	.13	.50
41	.82	.99	.82	.78	.39	.06	.64
45	.48	1.17	.55	1.07	.58	.25	.68
44	.71	1.24	.57	1.11	.38	.16	.69

The final two rows of this table show the results of testing designs with 45 and 44 kernel nodes with data other than the six sets of initial conditions used for training data. The table shows a compromise between performance and design complexity for these two design cases.

CONCLUSIONS AND SUGGESTED FURTHER WORK

This paper describes a radial basis function neural network, applied to the problem of Space Station Freedom solar array feathering for plume impingement load relief during Shuttle approach. Such a neural network was designed to minimize the angle of attack of expected plumes.

The design goal was to minimize network complexity while maintaining acceptable dynamic performance. The network was trained with a set of 30000 data points, then tested with additional data not contained by the training set. The network showed very good performance.

Before such a network can be used in actual operation, it must be trained with a larger and more persistently exciting set of data, and tested with an exhaustive set, possibly leading to the need for additional training.

The network could also have a more robust structure if it takes advantage of position feedback from the beta gimbals. The stability and performance of this control structure should be determined and enveloped.

REFERENCES

[1] Wagner, M. N. "SSF PV Array Plume Loads Transient Analysis." Rockwell International, Rocketdyne Division memo 91-EPS-8E-057, dated 16 September 1991.

[2] Meshcheryakov, I. P. and S. A. Mineav. "Construction and Investigation of an Information Model of the Process of Approach of Piloted Spacecraft." Kosmicheskie Issledovaniya, Vol. 15, No. 6, pp. 937-940, Nov.-Dec. 1977.

[3] Brody, A. R. "Evaluation of the '0.1% Rule for Docking Maneuvers." Journal of Spacecraft and Rockets, Vol. 27, No. 1, Jan.-Feb. 1990, pp. 7-8.

[4] Greene, F. "The Dampening of Modal Vibrations on a Flexible Beam Using a Neural Network Transformation." Master's Thesis, the Ohio State University, 1993.

[5] Powell, M. Radial Basis Functions for Multivariable Interpolation: A Review." Proceedings of the IMA Conference on Algorithms for the Approximation of Functions and Data, Royal Military College of Science, Shrivenham, July 1985.

[6] Lloyd, S. P. "Least Squares Quantization in PCM." IEEE Transactions on Information Theory, 28:281-294, February 1982.

[7] MacQueen, J. "Some Methods for Classification and Analysis of Multivariate Observations." Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Univ. of California, Berkeley, June 1965.

[8] Moody, J. and C. J. Darken. "Fast Learning in Networks of Locally-Tuned Processing Units." Neural Computation, 1:281-294, 1989.

[9] Chen, S., C. F. Cowan and P. M. Grant. "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks." IEEE Transactions on Neural Networks, 2:281-294, March 1991.

[10] Graham, R. E. "Dynamic Feathering of Space Station

Freedom Solar Arrays via a Neural Network." Doctoral dissertation, Cleveland State University, 1994.

[11] Broomhead, D. S. and D. Lowe. "Multivariable Functional Interpolation and Adaptive Networks." Complex Systems, 2:321-355, 1988.

[12] Micchelli, C. A. "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions." Constructive Approximation, 2:11-22, 1986.

[13] Park, J. and I. W. Sandberg. "Universal Approximation Using Radial Basis Function Networks." Neural Computation, 3:246-257, 1991.

[14] Powell, M. Radial Basis Function Approximations to Polynomials. Proceedings of the 12th Dundee Biennial Conference on Numerical Analysis, University of Dundee, June 1987.

[15] Schagen, I. P. "Sequential Exploration of Unknown Multidimensional Functions as an Aid to Optimization." IMA Journal of Numerical Analysis, 4:337-347, 1984.

[16] Rumelhart, D. E., G. E. Hinton and R. J. Williams. "Learning Internal Representations by Error Propagation." Parallel Distributed Processing, Vol. 1: Foundations. Cambridge, MA: MIT Press, 1986.

Appendix. Radial Basis Function Neural Networks

Broomhead and Lowe [11] represent RBF networks by the mapping f_{ik}: $\mathbb{R}^p \rightarrow \mathbb{R}^q$, such that

$$y_{ik} = \lambda_{0k} + \sum_{j=1}^{M} \lambda_{jk} \, \phi \, (\| \boldsymbol{X}_i - \boldsymbol{C}_j \|) \tag{1}$$

for $1 \leq k \leq q$ and $1 \leq i \leq m$, and, referring to Figure 4,

y_{ik}	is the kth output for the ith exemplar;
λ_{0k}	is the kth output bias;
M	is the number of kernel neurons;
λ_{jk}	is the synaptic weight between the jth kernel function and kth output;
$\phi(\| \mathbf{X}_i - \mathbf{C}_j \|)$	is the RBF $\phi(\mathbb{R}) = \{ \phi(r): r \in \mathbb{R}^+ \}$;
$\mathbf{X}_i$	is the vector of inputs, length p;
$\mathbf{C}_j$	is the vector of centroids for the jth kernel RBF.

Alternatively, the RBF can be represented by a matrix mapping F_i: $\mathbb{R}^p \rightarrow \mathbb{R}^q$ such that

$$\boldsymbol{Y}_i = \boldsymbol{\Lambda}_0 + \sum_{j=1}^{M} \boldsymbol{\Lambda}_j \, \phi \, (\| \boldsymbol{X}_i - \boldsymbol{C}_j \|) \tag{2}$$

where the output bias and synaptic weights from (1) have been lumped into a row vector.

Powell [5] formulates the multivariable interpolation problem for real numbers in the same manner as was presented above, for a single output. For this case, we can extend the solution to a set of linear equations for the synaptic weights, as follows:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix} = \begin{bmatrix} 1 & \phi(\|X_1 - C_1\|) & \cdots & \phi(\|X_1 - C_M\|) \\ 1 & \phi(\|X_2 - C_1\|) & \cdots & \phi(\|X_2 - C_M\|) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi(\|X_m - C_1\|) & \cdots & \phi(\|X_m - C_M\|) \end{bmatrix} \begin{bmatrix} \lambda_{01} & \cdots & \lambda_{0q} \\ \lambda_{11} & \cdots & \lambda_{1q} \\ \vdots & \ddots & \vdots \\ \lambda_{M1} & \cdots & \lambda_{Mq} \end{bmatrix} \tag{3}$$

where $\mathbf{Y}_i = \mathrm{F}_i$. We may make the following definitions:

$$[\,\mathbf{Y}_1\ \mathbf{Y}_2\ \cdots\ \mathbf{Y}_m\,] = [\,Y_1\ Y_2\ \cdots\ Y_m\,]^T$$

$$\mathbf{\Theta} = \begin{bmatrix} 1 & \phi(\|X_1 - C_1\|) & \cdots & \phi(\|X_1 - C_M\|) \\ 1 & \phi(\|X_2 - C_1\|) & \cdots & \phi(\|X_2 - C_M\|) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi(\|X_m - C_1\|) & \cdots & \phi(\|X_m - C_M\|) \end{bmatrix} \tag{4}$$

$$[\,\mathbf{\Lambda}_1\ \cdots\ \mathbf{\Lambda}_q\,] = \begin{bmatrix} \lambda_{01} & \cdots & \lambda_{0q} \\ \lambda_{11} & \cdots & \lambda_{1q} \\ \vdots & \ddots & \vdots \\ \lambda_{M1} & \cdots & \lambda_{Mq} \end{bmatrix}$$

from which we may see that the vectors of synaptic weights can be solved for independently, decoupling (3) and reducing it to

$$\mathbf{Y} = \mathbf{\Theta}\,\mathbf{\Lambda} \tag{5}$$

For the special case of a square matrix of RBFs containing no bias terms, Micchelli [12] proved that the matrix is always nonsingular if the input data points are all unique. Equation (5) is then solved simply, by inverting the RBF matrix. This is an exact interpolation solution, and requires that the number of RBF centroids equal the number of exemplars.

This is impractical to realize for many applications, particularly those in signal processing, and may even lead to redundancy being used to fit noisy data. The general solution to (5), however, requires minimizing a linear least-squares cost function,

$$J = \| \mathbf{\Theta\Lambda} - \mathbf{Y} \|^2 \tag{6}$$

which is minimized with the application of the pseudoinverse of the RBF matrix,

$$\hat{\mathbf{\Lambda}} = (\mathbf{\Theta}^T \mathbf{\Theta})^{-1} \mathbf{\Theta}^T \mathbf{Y} \tag{7}$$

The RBF network can be uniquely determined by considering it as q multiple-input, single-output (MISO) systems, each of which is a special case of the linear regression model

$$\hat{\mathbf{Y}} = \mathbf{\Theta}\hat{\mathbf{\Lambda}} + \boldsymbol{E} \tag{8}$$

where **E**, a vector of length m, is the residual error introduced by using the estimated weights of equation (7) as opposed to the exact weights (and all the extra centroids that come with them).

Park and Sandberg [13] have shown that the RBF network is capable of universal approximations for activation functions of the form

$$\phi\left(\frac{\|\boldsymbol{x}_i - \boldsymbol{c}_j\|}{\sigma}\right) \tag{9}$$

which uses the same smoothing factor or width, σ, for each kernel node j. Typically-used RBFs, representing research by Powell [5, 14] and Schagen [15] are shown in the following table:

$\phi(r)$	RBF Classification
r	linear
r^3	cubic
$r^2 \log r$	thin plate spline
$(r^2 + c^2)^{1/2}$	multiquadratics
$(r^2 + c^2)^{-1/2}$	inverse multiquadratics
$e^{-(r/c)^2}$	multivariate Gaussian

Gaussian functions were selected here.

RBF learning can be considered a hybrid process. The first step is finding the centroids, $\mathbf{c}_j$, and the smoothing factor(s), σ_j. The second step is finding the network weights between the kernel and output nodes by solving equations (6) and (7). These selections are critical in determining network performance.

If we make the definition for the RBF matrix from (4), then the orthogonal least-squares procedure transforms the vectors θ_i into orthogonal basis vectors. The RBF matrix can then be factored into

$$\boldsymbol{\Theta} = \boldsymbol{Q}\,\boldsymbol{R} \tag{10}$$

where **Q** is an orthogonal (m x M+1) matrix, subject to

$$\boldsymbol{Q}^T\boldsymbol{Q} = \begin{bmatrix} q_1{}^Tq_1 & 0 & 0 & \cdots & 0 \\ 0 & q_2{}^Tq_2 & 0 & \cdots & 0 \\ 0 & 0 & q_3{}^Tq_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & q_{M+1}{}^Tq_{M+1} \end{bmatrix} \tag{11}$$

and **R** is an upper triangular (M+1 x M+1) matrix, given by

$$\boldsymbol{R} = \begin{bmatrix} 1 & r_{1,2} & r_{1,3} & \cdots & r_{1,M} & r_{1,M+1} \\ 0 & 1 & r_{2,3} & \cdots & r_{2,M} & r_{2,M+1} \\ 0 & 0 & 1 & \cdots & r_{3,M} & r_{3,M+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & r_{M,M+1} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{12}$$

Now, if we make the following definition,

$$\boldsymbol{z} = \boldsymbol{R}\,\boldsymbol{\Lambda} \tag{13}$$

then the solution to the linear least-squares problem given in equation (7) is

$$\hat{\boldsymbol{z}} = \boldsymbol{R}\,\hat{\boldsymbol{\Lambda}} = (\boldsymbol{Q}^T\boldsymbol{Q})^{-1}\boldsymbol{Q}^T\,\hat{\boldsymbol{\Upsilon}} \tag{14}$$

or equivalently, in terms of individual components,

$$\hat{z}_i = \frac{q_i^{\,T}\,\Upsilon}{q_i^{\,T} q_i} \tag{15}$$

Equation (15) can be considered a linear measurement model, and for zero residual error the mean of the output is

$$\overline{\boldsymbol{\Upsilon}} = \boldsymbol{\Theta}\,\overline{\boldsymbol{\Lambda}} \tag{16}$$

If the residual error is uncorrelated with Λ, then the mean square error is given by the trace of the matrix formed by multiplying ($\boldsymbol{\Upsilon}$ - expected value of $\boldsymbol{\Upsilon}$) by its transpose.

For the scalar case, (16) eventually can be reduced to

$$\frac{1}{m}\mathbf{Y}^T\mathbf{Y} = \frac{1}{m}\sum_{j=1}^{M+1} z_j^2 q_j^T q_j + \frac{1}{m}\boldsymbol{E}^T\boldsymbol{E} \tag{17}$$

The first term on the right-hand side of (17) is output variance due to regressors z_j. The other term is output variance. The contribution of a particular regressor to the output variance can be defined as

$$\eta_j = \frac{z_j^2 q_j^T q_j}{\mathbf{Y}^T\mathbf{Y}} \tag{18}$$

for $1 \le j \le M+1$. The definition of (18) allows for regressors to be determined in a forward regression manner.

The Gram-Schmidt orthogonalization procedure is as follows:

(1) <u>Initialization</u>:

- o Set k to 1.
- o Set Y_t to Y.
- o For $1 \le j \le M+1$, solve (15) for z_j and (18) for η_j.
- o Let η_1 be the maximum value of η_j and l be the j at which η_1 was achieved.

(2) <u>Gram-Schmidt Orthogonalization</u>:

- o Select q_1 equal to θ_l and use this to solve (15) for z_1.
- o Reorganize RBF matrix so that the first column of the matrix is q_1.
- o For this column, and for $2 \le j \le M+1$, do the following:

$$\begin{aligned} r_{1j} &= q_1^T\,\theta_j / q_1^T q_1 \\ \theta_j &= \theta_j - r_{1j}\,q_1 \\ Y_i &= Y_i - z_1\,q_1 \end{aligned} \tag{19}$$

(3) Regressor selection: the number of regressors to be selected is based on the chosen tolerance ρ ($0 < \rho < 1$). Regressor selection ends when

$$\rho > 1 - \sum_{k=1}^{M_t} \eta_k \qquad (20)$$

where M becomes the number of significant centroids selected by the procedure, or kernel neurons.

(4) Procedure until termination:

- o Provided equation (20) is true and $k < M+1$, then for $k \leq j \leq M+1$, solve (15) for z_j and (18) for η_j.
- o Let η_k equal the maximum value of η_j and l be the j at which η_k was achieved.
- o Set q_k to θl and solve (15) for z_k.
- o Reorganize the RBF matrix so that the first column q_k.
- o Perform Gram-Schmidt orthogonalization with respect to this column, as in (21):

$$\begin{aligned} r_{kj} &= q_k^T \theta_j / q_k^T q_k \\ \theta_j &= \theta_j - r_{kj} q_k \\ Y_i &= Y_i - z_k q_k \end{aligned} \qquad (21)$$

When the procedure is completed, form the **R** matrix and solve for the network weights from (13). The selection of the tolerance (ρ) is the key to balancing the accuracy and complexity of the final network design. Chen et. al. [9] give alternative selection criteria for signal processing applications.

The strength of the RBF network is the localization of the kernel nodes and the linear learning rules, which allow for faster training than does backpropagation. To maintain this advantage, the only weights updated in this design are those between the kernel nodes and the output. In this case, even the backpropagation learning rule [16] is particularly simple:

$$\Delta\lambda_j = \gamma \sum_{i=1}^{m} \left[y_i - \left(\sum_{j=0}^{M} \lambda_j \, P_j \right) \right] P_i \tag{22}$$

for all j from 0 to M. Equation (22) gives the required weight change between the jth kernel node and the network output, for the ith exemplar. P_j is 1 if j=0 and θij otherwise. If we now define the network output for the ith exemplar as o_i,

$$o_i = \sum_{j=0}^{M} \lambda_j \, P_j \tag{23}$$

then equation (22) is simplified,

$$\Delta\lambda_j = \gamma \sum_{i=1}^{m} \, (y_i - o_i) \; P_i \tag{24}$$

for all j. In matrix form, (24) is

$$\Delta\mathbf{\Lambda} = \gamma\mathbf{\Theta}^T(\mathbf{Y} - \mathbf{\Theta\Lambda}) \tag{25}$$

which leads to the backpropagation law for this case,

$$\mathbf{\Lambda} = \mathbf{\Lambda} + \Delta\mathbf{\Lambda} \tag{26}$$

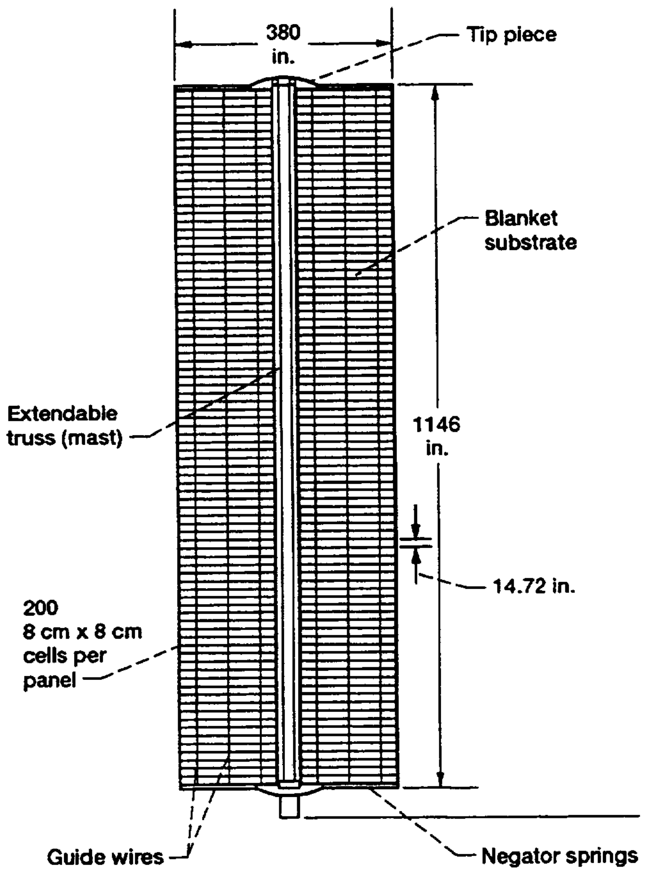

Figure 1.—Space Station split blanket solar array.

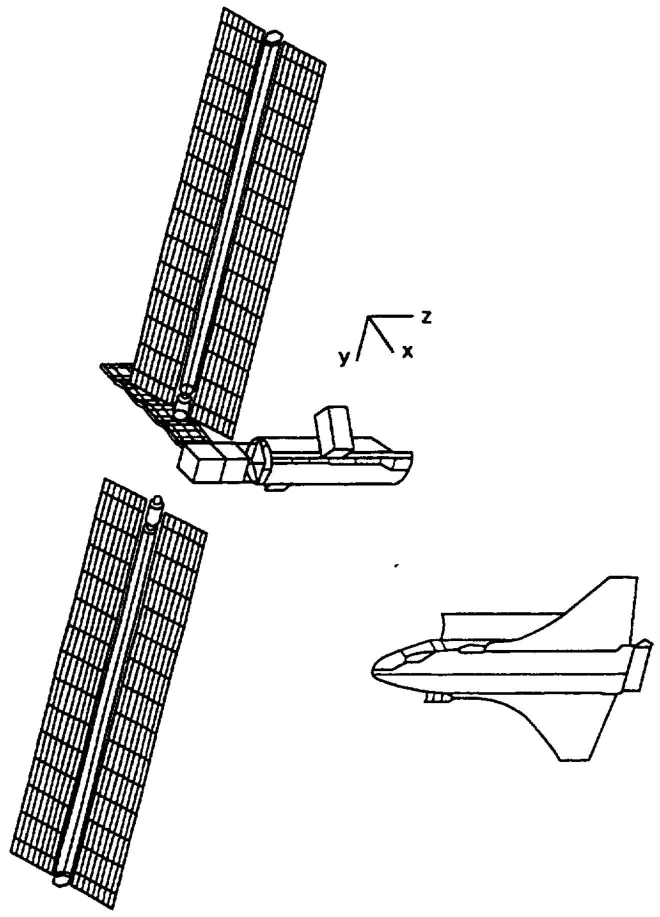

Figure 3.—Shuttle approach to SSF stage configuration SC-2.

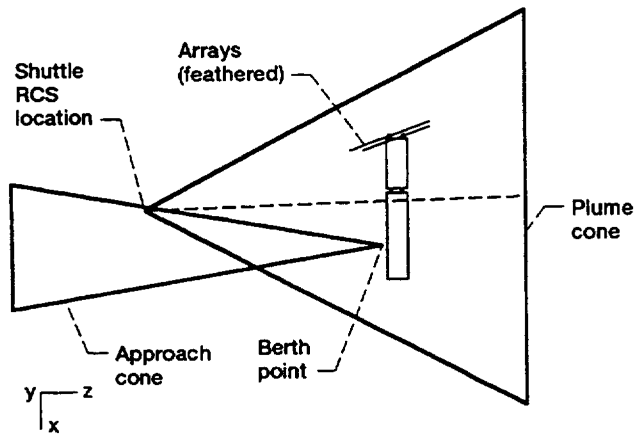

Figure 2.—Geometry of plume impingement.

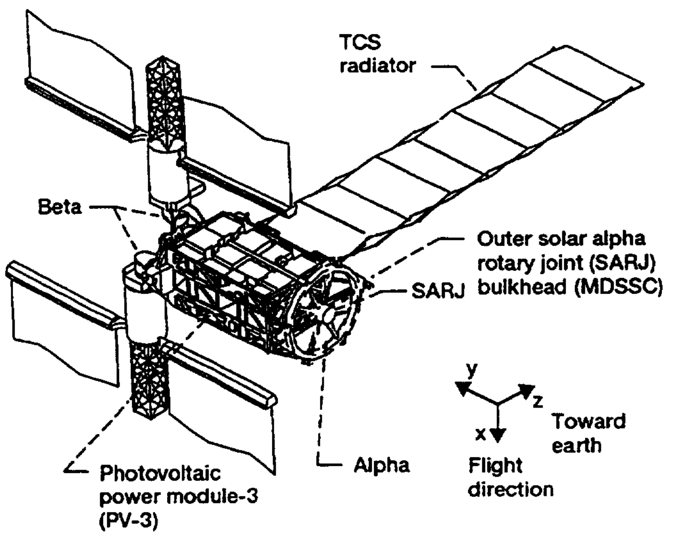

Figure 4.—Alpha and beta gimbals. From IEA delta PDR presentation package, Rocketdyne 02/92.

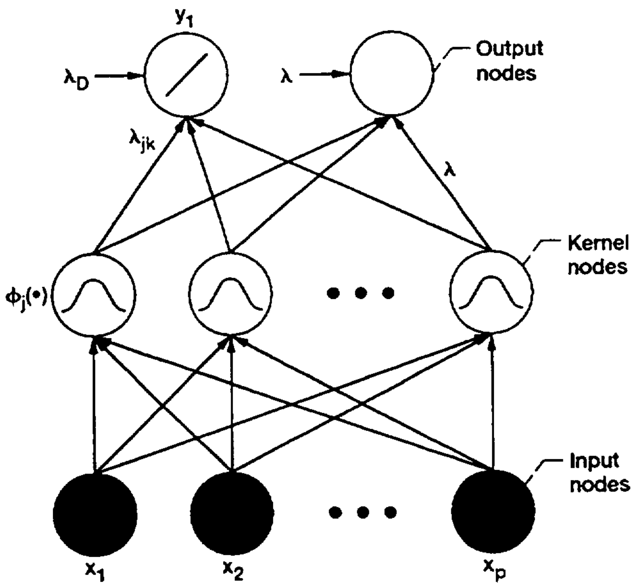

Figure 5.—RBF neural network for determination of the optimal beta gimbal angles.

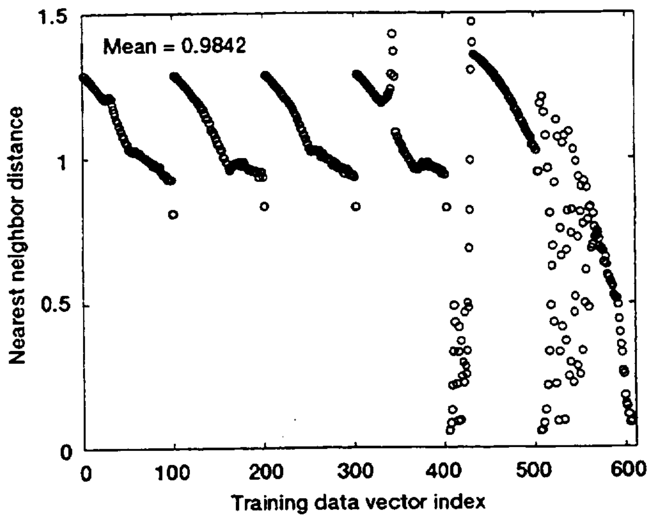

Figure 6.—Scatter plot of nearest neighbor distance for neural network training data.

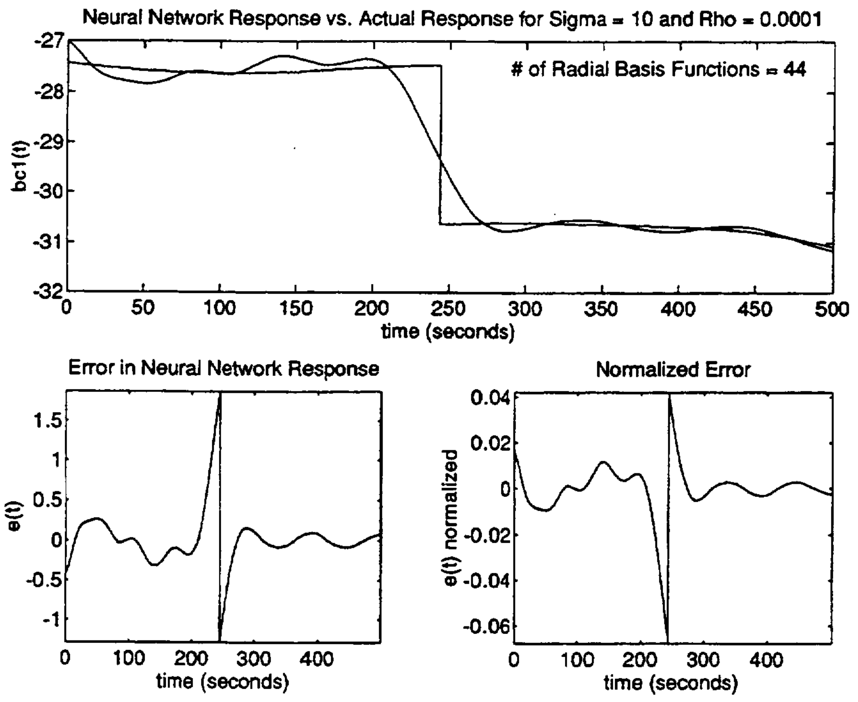

Figure 7.—RBF neural network design 4 response for testing run 1.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE February 1994	3. REPORT TYPE AND DATES COVERED Technical Memorandum

4. TITLE AND SUBTITLE

Optimal Space Station Solar Array Gimbal Angle Determination Via Radial Basis Function Neural Networks

5. FUNDING NUMBERS

6. AUTHOR(S)

Daniel J. Clancy, Ümit Özgüner, and Ronald E. Graham

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

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–8412

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

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM–106482

11. SUPPLEMENTARY NOTES

Daniel J. Clancy and Ümit Özgüner, Ohio State University, Columbus, Ohio 43210; and Ronald E. Graham, NASA Lewis Research Center. Responsible person, Ronald E. Graham, organization code 4110, (216) 433–5178.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories 13 and 31

12b. DISTRIBUTION CODE

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

The potential for excessive plume impingement loads on Space Station Freedom solar arrays, caused by jet firings from an approaching Space Shuttle, is addressed. An artificial neural network is designed to determine commanded solar array beta gimbal angle for minimum plume loads. The commanded angle would be determined dynamically. The network design proposed here involves radial basis functions as activation functions. Design, development and simulation of this network design are discussed.

14. SUBJECT TERMS

Neural networks; Pointing accuracy; Control dynamics

15. NUMBER OF PAGES
21

16. PRICE CODE
A03

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

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102