l Payload, Δpay (lbs) (All variables are normalized by their respective baseline values).

NEURAL NETWORK AND RESPONSE SURFACE METHODOLOGY FOR ROCKET ENGINE COMPONENT OPTIMIZATION

Rajkumar Vaidyanathan[1]*, Nilay Papila[1]*, Wei Shyy[1]‡, P. Kevin Tucker[2]§, Lisa W. Griffin[2]¶,
Raphael Haftka[1]#, and Norman Fitz-Coy[1]**

[1]Department of Aerospace Engineering, Mechanics & Engineering Science,
University of Florida, Gainesville, FL 32611-6250 USA
[2]NASA Marshall Space Flight Center, Huntsville, AL 35812.

ABSTRACT

The goal of this work is to compare the performance of response surface methodology (RSM) and two types of neural networks (NN) to aid preliminary design of two rocket engine components. A data set of 45 training points and 20 test points, obtained from a semi-empirical model based on three design variables, is used for a shear coaxial injector element. Data for supersonic turbine design is based on six design variables, 76 training data and 18 test data obtained from simplified aerodynamic analysis. Several RS and NN are first constructed using the training data. The test data are then employed to select the best RS or NN. Quadratic and cubic response surfaces, radial basis neural network (RBNN) and back-propagation neural network (BPNN) are compared. Two-layered RBNN are generated using two different training algorithms, namely, *solverbe* and *solverb*. A two-layered BPNN is generated with Tan-Sigmoid transfer function. Various issues related to the training of the neural networks are addressed, including number of neurons, *error goals*, *spread constants*, and the accuracy of different models in representing the design space. A search for the optimum design is carried out using a standard, gradient-based optimization algorithm over the response surfaces represented by the polynomials and trained neural networks. Usually a cubic polynomial performs better than the quadratic polynomial but exceptions have been noticed. Among the NN choices, the RBNN designed using *solverb* yields more consistent performance for both engine components considered. The training of RBNN is easier as it requires linear regression. This coupled with the consistency in performance promise the possibility of it being used as an optimization strategy for engineering design problems.

* Graduate Student Assistant, Student Member AIAA
† Graduate Student Assistant, Student Member AIAA
‡ Professor and Dept. Chair., Associate Fellow AIAA
§ Aerospace Engineer, Member AIAA
¶ Aerospace Engineer, Member AIAA
# Distinguished Professor, Fellow AIAA
** Associate Professor, Member AIAA

1 INTRODUCTION

1.1 General Background

Advanced rocket propulsion systems are being proposed to meet goals for increased performance, robustness, and safety while concurrently decreasing weight and cost. These new goals are forcing consideration of design variables over ranges and in combinations not typically employed, thereby increasing the design space complexity. Objective and efficient evaluation of these new and complex designs can be facilitated by development and implementation of systematic techniques. Accordingly, Response Surface Methodology[1] (RSM) and Neural Network[2] (NN) techniques have been used to generate surrogate models representing data obtained from complex numerical and experimental simulations. An optimization algorithm is then used to interrogate these models for optimum design conditions, based on specified constraints. In this study, the preliminary design issues related to rocket propulsion components, including gas-gas injectors and supersonic turbines have been investigated. The objective of this effort is to assess relative performance of RSM and NN techniques in representing the design space.

A polynomial-based RSM, in which the design space is represented with quadratic and cubic polynomials in the dependent variables, is used. The polynomial coefficients are obtained by linear regression. The maximum or the minimum of the surface can then be located using a gradient search method. Response Surface methodologies have been used before for rocket engine component design. For example, Tucker et al.[3] have used RSM for rocket injector design. The approach is not tied to any specific data type or source. The dimensionality of the data is not a concern, and data obtained through both numerical and experimental methods can be effectively used. RSM enables the designer to combine any number of design variables for different types of injectors and propellant combinations. This generality allows the consideration of information at varying levels of breadth (i.e., scope of design variables) and depth (i.e., details of the design variables).

The RSM is effective in representing the global characteristics of the design space and it filters noise associated with design data. Depending on the order of polynomial employed and the shape of the actual response surface, the RSM can introduce substantial errors in certain regions of the design space. Shyy et al.[4] have showed it that

(lbs) (All variables are normalized by their respective baseline values).

for a given injector design, a third order response surface performs better than a second order surface. Generation of polynomial based surfaces can be costly for cases involving many of design variables due to the amount of data required to evaluate the coefficients. In fact, the number of coefficients increases rapidly with the order of polynomial. For example, a complete second-order polynomial of N design variables has $(N+1)(N+2)/(2!)$ coefficients. A complete cubic model has $(N+1)(N+2)(N+3)/(3!)$ coefficients. The choice of order of the polynomial and the terms to be included depends on the design problem. Many combinations of terms may have to be tried to represent the design space before the best one can be selected.

An optimization scheme requiring large amounts of data and evaluation time to generate meaningful results is of limited value. While the preliminary designs can be accomplished with empirically based information, detailed designs often require use of data from experiments and/or computational fluid dynamics (CFD) analyses. This data can be time consuming and expensive to generate in large quantities. Recently, NN have been used to represent the models instead of the more typical polynomial RSM. Work in the area of NN by Shyy et al.[4] and Papila et al.[5] have shown that some NN can perform well even when a modest amount of data is available. In particular radial basis neural networks (RBNN) like polynomial based RSM require only linear regression for training and have proven to be particularly accurate. Norgaad et al.[6] and Ross et al.[7] have investigated the feasibility of reducing wind tunnel test times by using NN to interpolate between measurements and demonstrated cost savings. These works have focused on using the NN to predict data. Attempts to use the network as a function evaluator and then to link it to the optimizer have been made by Protzel et al.[8], Rai and Madavan[9] and Greenman and Roth[10].

NN are highly flexible in functional form and hence can offer significant potential for representing complex functions. Networks, like RBNN, that are flexible and employ linear regression methods can use both of these properties to improve the performance. The number of neurons in the network, size of the region over which the neuron is sensitive, and the training accuracy of the network are some of the parameters that need to be selected in a network. These can be determined by comparing the performance of NN designed with different values of these parameters. Neural networks can be effectively used in two ways. First, they can be used in conjunction with RSM. In complex regions of the surface, the NN can be trained using the existing data. The trained NN can then be used to generate additional data to augment existing data, thus possibly enhancing the accuracy of the surface in that particular area. Such an approach was investigated by Shyy et al[4]. This work demonstrated that the NN could indeed yield additional information to help generate more accurate polynomial-based response surfaces. Second, NN can generate data to be used directly in conducting gradient-based optimization. In other words, NN can perform the role of either enhancing the fidelity of a polynomial-based response surface, as in the first approach, or generating information as input to an optimizer by itself without resorting to a polynomial representation, as in the second approach. Either way, the only function evaluations required are for the points sought by the optimizer, which searches the design space based on the sensitivity of the response to the perturbations in the design variables.

1.2 Scope

The present work is aimed at a direct comparison of the RSM and NN techniques in terms of accuracy and efficiency; the hybrid RSM-NN scheme noted above will not be used here. Both techniques are applied to data used in the design of two rocket engine components: a shear co-axial injector and a supersonic turbine. Variations of each technique are evaluated. Both second and third order polynomials will be used for the Response Surface (RS). Two NN schemes, radial basis and the more commonly used back-propagation NNs are used. The same database for each component will be used to train both the RS and the NN. Both will then be linked to an optimization procedure. There is little rigorous theory in the literature to establish the desired framework for a clear comparison between the performances of the two techniques. However, this work provides an assessment of the techniques regarding their practical use in the rocket engine component design process.

2 APPROACHES

2.1 Summary of Analytical Models and Design Variables

Two components of a rocket propulsion system have been considered here, the injector and the turbine. First, a shear coaxial injector element that uses gaseous oxygen (GO_2) and gaseous hydrogen (GH_2) as propellants is used to investigate the relative performance of RSM and NN in the design of rocket engine injectors. The original data set from Tucker et al[3] (45 design points) is used to generate quadratic and cubic response surfaces for both, energy release efficiency (ERE), a measure of injector performance, and chamber wall heat flux (Q). These 45 design points are evenly distributed over the design space. ERE was obtained using correlations taking into account combustor length, L_{comb} (length from injector to throat), and the propellant velocity ratio, V_f/V_o. The nominal chamber wall heat flux at a point just downstream of the injector, Q_{nom}, was calculated using a modified Bartz equation. It was then correlated with propellant mixture ratio, O/F, and propellant velocity ratio, V_f/V_o, to yield the actual chamber wall heat flux, Q. The accuracy of each polynomial fit on the original data set is evaluated. Two different types of radial basis NN (RBNN) and a back propagation NN (BPNN) are also trained to represent ERE and Q. Each surface is then used to conduct design optimization over the same range of independent variables. The optimal design points are compared with exact points calculated from the empirical model of Calhoon et al[11]. The range of design variables considered in this study is shown in Table 1. Twenty additional data points that are not used in the generation of response surfaces or the neural networks are

used to assess the accuracy of different variants of RSM and NN.

The other propulsion system component examined is a supersonic turbine where the preliminary design is conducted by one-dimensional aerodynamic analysis using *FpgenML*[12]. *FpgenML* generates a flowpath and runs a preliminary meanline calculation on this flowpath. In this study, a single stage turbine has been considered. There are six design parameters and four output variables involved in this design process. There are 76 design points available for training. These 76 points were selected by using a face centered composite (fcc) design. Instead of 77 design points, as would be provided by a *fcc* design for six variables, only 76 were available since the meanline code could not converge for one of the designs. The design variables are the mean diameter, D, RPM, blade annulus area, A_{ann}, vane axial chord, C_v, blade axial chord, C_b, and stage reaction, k_r. These are parameters influencing the structural properties and performance of the turbine. Overall efficiency of the turbine, η, turbine weight, W, a lumped inertia measure, $(AN)^2$ ($A_{ann} \times (RPM)^2$) and speed at pitchline, V_{pitch} ($D \times RPM$) are chosen as dependent variables. The goal is to maximize the incremental payload (Δpay), which is derived from turbine weight (W) and efficiency (η). Therefore, the objective is a design where W is minimized and η is maximized. Due to the structural considerations, constraints have to be imposed on $(AN)^2$ and V_{pitch}.

Using 18 additional simulations, distributed within the design space, the accuracy of the models is tested. The ranges considered for the design variables and the dependent variables are shown in Eqs. (1) and (2).

$$\left.\begin{array}{l} \text{For the design variables:} \\ 1.496 > D > 0.0502 \\ 1.4 > RPM > 0.6 \\ 1.3 > A_{ann} > 0.699 \\ 1.706 > C_v > 0.394 \\ 1.143 > C_b > 0.264 \\ 0.0 > k_r > 0.5 \end{array}\right\} \quad (1)$$

$$\left.\begin{array}{l} \text{For dependent variables:} \\ 1.116 > \eta > 0.223 \\ 0.801 > W > 0.422 \\ 2.197 > (AN)^2 > 0.343 \\ 1.849 > V_{pitch} > 0.0484 \end{array}\right\} \quad (2)$$

All the variables involved in the design process are normalized by their respective baseline values.

2.2 Objective Functions

When attempting to optimize two or more different objective functions, conflicts between them arise because of the different relationships they have with the independent parameters. To solve this problem, a multi-objective[12] approach is investigated in this study. Here, competing objective functions are combined to a single composite objective function. The maximization of the composite function effectively provides a compromise between the individual functions. An average of some form is normally used to represent the composite function. For example, Tucker et al[3] used a geometric mean to combine their two objectives, *ERE* and Q. The composite desirability is of the form

$$D = \left(\prod_{i=1}^{l} d_i \right)^{1/l} \quad (3)$$

where D is the composite objective function, d_i's are normalized values of the objective functions and l is the number of objective functions.

Another way of constructing a composite function is to use a weighted sum of the objective functions. The composite desirability function can then be expressed as

$$D = \sum_{i=1}^{l} \alpha_i f_i \quad (4)$$

where D is the composite objective function and f_i's are the non-normalized objective functions. The α_i's are dimensional parameters that control the importance of each objective function.

For the injector, the goal is to maximize the energy release efficiency, *ERE* while minimizing the chamber wall heat flux, Q. This is achieved by maximizing a composite objective function given by Eq (5).

$$D = (d_{ERE} d_Q)^{1/2} \quad (5)$$

where the normalized functions are defined in Eqs. (6) and (7). In the case where a response should be maximized, such as *ERE*, the normalized function takes the form:

$$d_{ERE} = \left(\frac{ERE - A}{B - A} \right)^s \quad \text{for } A \le ERE \le B \quad (6)$$

where B is the target value and A is the lowest acceptable value. We set $d_{ERE} = 1$ for any $ERE > B$ and $d_{ERE} = 0$ for $ERE < A$. The choice of s is made based on the subjective importance of this objective in the composite desirability function. In the case where a response is to be minimized, such as Q, the normalized function takes on the form:

$$d_Q = \left(\frac{E - Q}{E - C} \right)^t \quad \text{for } C \le ERE \le E \quad (7)$$

where C is the target value and E is the highest acceptable value. We set $d_Q = 1$ for any $Q < C$ and $d_Q = 0$ for $Q > E$. A, B, C, and E are chosen according to the designer's priorities or, as in the present study, simply as the boundary values of the domain of *ERE* and Q. The value of t is again chosen to reflect the importance of the objectives in the design. In the study A and B are equal to 95.0 and 99.9, respectively. Values of C and E are equal to 0.48 and 1.1, respectively. Both s and t were set to a value of 1.

In the case of the turbine, a weighted sum of the two objectives η and W has been used. The expression, in the context of the turbine gives the incremental value of the payload with the change in W and η. The goal is to maximize this incremental value, which in turn results in minimum W and maximum η.

$$D = \Delta pay = C_1 \times 100 \times (\eta - \eta_b) - C_2 \times (W - W_b) \quad (8)$$

where C_1 = the amount of payload increment capacity for any efficiency gain
C_2 = the amount of payload increment capacity for any weight gain
η = the calculated efficiency
η_b = the baseline efficiency
W = calculated weight
W_b = the baseline weight.

The baseline efficiency and weight are obtained using existing design knowledge without benefiting from an optimization strategy. The weight associated with η expressed in percentage, by multiplying it with 100, is C_1 and the weight associated with W is C_2. This relationship is developed based on detailed turbopump design processes. For one percent increase in efficiency a payload increase of C_1 lbs can be achieved, and as the weight of the turbine increases the payload has to be correspondingly decreased by a factor of C_2.

2.3 Response Surface Methodology (RSM)

Polynomial RSM constructs polynomials of assumed order and unknown coefficients based on regression analysis. The solution for the set of coefficients that best fits the training data is a linear least square problem. The number of coefficients to be evaluated depends on the order of polynomial and the number of design parameters involved.

According to the injector model developed by Calhoon et al[11], injector performance, as measured by *ERE* depends only on the velocity ratio, V/V_o, and combustion chamber length, L_{comb}. Therefore, only 15 distinct design points are available for *ERE*. Since chamber wall heat flux depends only on the velocity ratio, V/V_o, and the oxidizer to fuel ratio, *O/F*, there are 9 distinct design points for *Q*. The design space for this problem is depicted in Figure 1. For *ERE*, the 5 distinct chamber lengths offer the potential for a fourth-order polynomial fit in L_{comb}, while the three different velocity ratios limit the fit in V/V_o to second order. Quadratic and cubic response surfaces for both *ERE* and *Q* have been generated for evaluation. The above-noted limitations on the data, limits the cubic surfaces to be third order in L_{comb} only.

As already mentioned, to construct a complete quadratic polynomial of N design variables, the number of coefficients required is $(N+1)(N+2)/(2!)$. In the turbine case with 6 design variables, we would need to estimate 28 coefficients. A complete cubic model would require $(N+1)(N+2)(N+3)/(3!)$ or 84 coefficients and four levels. Since the data available is not sufficient to evaluate all the cubic terms, reduced cubic models are employed.

The response surfaces were generated by standard least-squares regression using JMP[14], a statistical analysis software package. JMP is an interactive, spreadsheet-based program having a variety of statistical analysis tools. Statistical techniques are also available for identifying polynomial coefficients that are not well characterized by the data. A stepwise regression procedure based on t-statistics is used to discard terms and improve the prediction accuracy. The t-statistic, or t-ratio, of a particular coefficient is given by the value of the coefficient divided by the standard error of the coefficient, which is an estimate of its standard deviation. The accuracy of different surfaces at points different from the training data can be estimated by comparing the adjusted root mean square error defined as:

$$\sigma_a = \sqrt{\frac{\sum e_i^2}{n - n_p}} \quad (9)$$

Here e_i is the error at i^{th} point of the training data, n is the number of training data points and n_p is the number of coefficients. When the data contains uncorrelated Gaussian noise, σ_a provides an unbiased estimate of that noise. Even when the error is not solely due to noise σ_a provides a good overall comparison among the different surface fits.

The accuracy of the models in representing the objective functions is also gauged by comparing the values of the objective function at test design points, different from those used to generate the fit. The root mean square error, σ, for the test set is given by:

$$\sigma = \sqrt{\frac{\varepsilon^2}{m}} \quad (10)$$

In this equation ε_i is the error at the i^{th} test point and m is the number of test points.

2.4 Neural Networks

Two different types of NN have been used, namely radial basis[15] and back-propagation[15]. The training process of the network is a cyclic process and the weights and biases of the nodes of the network are adjusted until an accurate mapping is obtained. This trained network can then predict the values of the objective for any new set of design variables in the design space. The neural network toolbox[15] available in *Matlab* is used for the current analysis.

2.4.1 Radial Basis Neural Networks (RBNN)

Radial-basis neural networks are two-layer networks with a hidden layer of radial-basis transfer function and a linear output layer (Figure 2). RBNN requires large number of neurons, depending on the size of the data set, but they can be designed in a small amount of time. This is due to the fact that the process of determining the weights associated with the large number of neurons uses linear regression. Thus, they may be efficient to train when there are large amounts of data available for training.

The transfer function for radial basis neuron is *radbas*, which is shown in Figure 2b. *Radbas* has maximum and minimum outputs of 1 and 0, respectively. The output of the function is given by

$$a = radbas\left(dist\left(w, p\right)\times b\right) \tag{11}$$

where *radbas* is the transfer function, *dist* is the vector distance between the network weight vector, w and the input vector, $\boldsymbol{p}$, and b is the bias. In a radial basis network (Figure 2a) each neuron in the *radbas* hidden layer is assigned weights, w_1 which are equal to the values of one of the training input design points. Therefore, each neuron acts as a detector for a different input. The bias for each neuron in that layer, b_1 is set to 0.8326/*sc*, where *sc* is the spread constant, a value defined by the user. This defines the region of influence by each neuron. The whole process is then reduced to the evaluation of the weights, w_2, and biases, b_2, in the output linear layer, which is a linear regression problem. If the input to a neuron is identical to the weight vector, the output of that neuron is 1, since the effective input to the transfer function is zero. When a value of 0.8326 is passed through the transfer function the output is 0.5. For a vector distance equal to or less than 0.8326/*b*, the output is 0.5 or more. The *spread constant* defines the radius of the design space over which a neuron has a response of 0.5 or more. Small values of *sc* can result in poor response in a domain not closely located to neuron positions, that is, for inputs that are far from the training data as compared to the defined radius, the response from the neuron will be negligible. Large values will result in low sensitivity of neurons. Since the radius of sensitivity is large, neurons whose weights are different from the input values by a large amount will still have high output thereby resulting in a flat network. The best value of the *spread constant* for some test data can be found by comparing σ for networks with different *spread constants*.

In *Matlab*, radial-basis networks can be designed using two different design procedures, *solverbe* and *solverb*. *Solverbe* designs a network with zero error on the training vectors by creating as many radial basis neurons as there are input sets. Therefore, *solverbe* may result in a larger network than required and map the network exactly, thereby fitting numerical noise. A more compact design in terms of network size is obtained from *solverb*, which creates one neuron at a time to minimize the number of neurons required. At each epoch or cycle, neurons are added to the network till a user specified RMS error is reached or until the network has the maximum number of neurons possible. The design parameters for *solverb* are the *spread constant*, a user defined RMS *error goal*, and the maximum number of epochs whereas it is only the *spread constant* for *solverbe*.

In case of the injector design there are two objectives, namely *ERE* and Q and for turbine the objectives are η and W. Figures 3 and 4 give the variation of σ for the network design with *solverbe* for the objective functions of the two engine components. In case of *solverb* the *error goal* during training also defines the accuracy of the network. An objective of fitting a numerical model is to remove the noise associated with the data. A model, which maps exactly as *solverbe* does, will not eliminate the noise, whereas *solverb* will. Figures 5 and 6 give the variation of σ for the network design with *solverb* for the objective functions of the two engine components.

By comparing Figures 3-6 it can be seen that for low values of *spread constant* the NN network has a poor performance. As the *spread constant* increases σ asymptotically decreases. However, as demonstrated by Figure 5a the performance of the network can deteriorate for higher values of the *spread constant*. The region with a large variation in σ is highly unreliable because this indicates a high sensitivity of the model to a small variation of *spread constant* and possibly the test data, in this region. Hence the desirable *spread constant* is selected from the region where the performance of the network is relatively consistent.

Figures 5 and 6 also show the influence of *error goal* on the network. Generally if a network maps the training data accurately it can be expected to perform efficiently with the test data. However, accurately mapping noisy data may result in poor prediction capabilities for the network. The variation in the performance is not significant except for the *ERE* and Q network (Figure 5), where the poor performance of the network at high values of *spread constant* improves for a larger *error goal*. This may indicate the presence of noise in the data for *ERE*, which *solverb* is able to eliminate with an appropriate *error goal*. Figure 7 shows variations in number of epochs and σ with the variation of *error goal* for a given *spread constant* when RBNN is designed with *solverb*. The number of neurons in the network is one more than the number of epochs. One expects that as the *error goal* increases the number of epochs becomes smaller and the network performs less accurately as in Figures 7a and 7b. However as demonstrated by Figures 7c and 7d, a more stringent *error goal* for the training data does not necessarily result in better predictive capability against the test data. Less accurate network can be designed for these objectives, which have smaller prediction error.

When choosing an appropriate network the above-mentioned features have to be considered. The performance of the constructed NN is best judged by comparing the prediction error as given in Eq. (10), for different networks. Using *solverbe*, networks are designed with varying *spread constants* and the one that yields the smallest error is selected. When *solverb* is used, networks are designed for different *spread constants* and *error goals*. The network that gives the smallest error for the test data is used. The details of the networks selected are discussed in later sections.

2.4.2 Back-propagation Neural Networks (BPNN)

Back-propagation networks are multi-layer networks with hidden layers of sigmoid transfer function and a linear output layer (Figure 8). The transfer function in the hidden layers should be differentiable and thus, either log-sigmoid or tan-sigmoid functions are typically used. In this study, a single hidden layer with a tan-sigmoid transfer

function, *tansig*, (Figure 8b) is considered. The output of the function is given by

$$a = tansig\left(w\bar{}p + b\right) \tag{12}$$

where *tansig* is the transfer function, w is the weight vector, p is the input vector and b is the bias vector. The maximum and minimum outputs of the function are 1 and –1, respectively.

The number of neurons in the hidden layer of a back-propagation network is a design parameter. It should be large enough to allow the network to map the functional relationship, but not too large to cause overfitting. Once it has been chosen, the network design is reduced to adjusting the weight matrices and the bias vectors. Since for BPNN the unknown weights are in the nonlinear function, the training process requires nonlinear regression, which is an optimization process. This optimization is usually performed using grad

ient methods. In *Matlab*, back-propagation networks can be trained by using three different training functions, *trainbp*, *trainbpx* and *trainlm*. The first two are based on the steepest descent method. Simple back-propagation with *trainbp* is usually slow since it requires small learning rates for stable learning. *Trainbpx*, applying momentum or adaptive learning rate, can be considerably faster than *trainbp*, but *trainlm*, applying *Levenberg-Marquardt* optimization[15], is the most efficient since it is based on a more efficient optimization algorithm.

The design parameters for *trainlm* are the number of neurons in the hidden layer, a user defined *error goal*, and the maximum number of epochs. The training continues until either the *error goal* is reached, the minimum error gradient occurs or the maximum number of epochs has been met.

For BPNN, the initial weights and biases are randomly generated and then the optimum weights and biases are evaluated through an iterative process. The weights and biases are updated by changing them in the direction of down slope with respect to the sum-squared error of the network, which is to be minimized. The sum-squared error is the sum of the squared error between the network prediction and the actual values of the output. In BPNN (Figure 8a) the weights, w_1, and biases, b_1, in the hidden *tansig* layer are not fixed as in the case of RBNN. Hence, the weights have a nonlinear relationship in the expression between the inputs and the outputs. This results in a nonlinear regression problem, which takes a longer time to solve than RBNN. Depending upon the initial weights and biases, the convergence to an optimal network design may or may not be achieved. Due to the randomness of the initial guesses, if one desires to mimic the process exactly for some purpose, it is impossible to re-train the network with the same accuracy or convergence unless the process is reinitiated exactly as before. The initial guess of the weights is a random process in Matlab. Hence to re-train the network the initial guess has to be recorded.

The architecture is decided based on past experience with similar kind of dataset. For a given objective the *error goal* is fixed and the number of hidden layer neurons are varied between 2 and the total number of inputs. Each network is retrained few times so as to start the search from random initial weights and biases. The networks that do not achieve the *error goal* are discarded. Among the converged networks the selection of the best network is made based on the value of σ. The goal is to attain as low a value for σ as possible. The number of neurons in the hidden layer is increased one at a time till the error goal is achieved and a small value of σ is obtained. Although this method may not be the best way to obtain the best BPNN, it is considered adequate for the current study. At times larger network has a high value of σ, which maybe due to overfitting of the design space. To prevent the model from converging to a local minimum, an iterative method is used as suggested by Stepniewski et al[16]. The obtained network is retrained with initial weights obtained by perturbing the weights of the obtained network.

$$w = w_o + \lambda r w_o \tag{13}$$

where w is the initial weight vector for the network to be trained, w_o is the weight vector of the obtained network, λ is the level of perturbation (0.1) and r is a matrix of random numbers between –1 to 1.

2.5 Design Optimization Process

The entire optimization process can be divided into two parts:

1) RS/NN training phase for establishing an approximation,
2) Optimizer phase.

In the first phase, RS or NN are generated with the available training data set. In the second phase the optimizer uses the RS/NN during the search for the optimum until the final converged solution is obtained. The initial set of design variables is randomly selected from within the design space. The flowchart of the process is shown in Figure 9.

The optimization problem at hand can be formulated as $min\{f(x)\}$subject to $lb \le x \le ub$, where lb is the lower boundary vector and ub is the upper boundary vector of the design variables vector x. If the goal is to maximize the objective function then $f(x)$ can be written as $-g(x)$, where $g(x)$ is the objective function. Additional linear or nonlinear constraints can be incorporated if required. The present design process does not have any such additional constraints. The optimization toolbox[17] in *Matlab* used here employs a sequential quadratic-programming algorithm.

3 RESULTS AND DISCUSSION

The RS and NN are constructed using the training data. The test data is then employed to select the best RS or NN. Specifically in RSM, the difference between the RS and the training data, as given by Eq. (9), is normally used to judge the performance of the fit. The additional use of the test data helps to evaluate the performance of different polynomials over design points not used during the training phase. This gives a complementary insight into the quality

of the RS over the design space. For both the rocket engine components, different polynomials were tried. Table 2 compares the performance of different polynomials used to represent the two objective functions of the injector case, *ERE* and Q. Starting with the all the possible cubic terms in the model, revised models are generated by removing and adding terms. Similar kind of analysis is also done for the turbine case. The best polynomial is selected based on a combined evaluation between σ_a and σ.

For the NN, the test data helps evaluate the accuracy of networks with varying neurons in BPNN and varying spread constant in RBNN. Thus the test data are part of the evaluation process to help select the final NN. Based on the RSM or NN model, a search for optimum design is carried out using a standard, gradient-based optimization algorithm over the response surfaces represented by the polynomials and trained neural networks.

3.1 Shear-Coaxial Injector

According to the available data, the injector performance, *ERE*, depends only on the velocity ratio, V_f/V_o, and combustion chamber length, L_{comb}, which indicates 15 distinct design points for *ERE*. The chamber wall heat flux, Q, depends on velocity ratio, V_f/V_o, and oxidizer to fuel ratio, O/F, and has nine distinct points. For *ERE*, as seen from Figure 1, five distinct levels for L_{comb} offers the potential for a fourth-order polynomial fit in the same, while three different velocity ratios and oxidizer to fuel ratio limit the fit in these variables to second order.

A reduced quadratic and an incomplete cubic response surfaces are used for the two objective functions. The first model in Table 2a and the sixth model in Table 2b are the selected cubic models for *ERE* and Q, respectively. There is no noticeable improvement among the remaining cubic model for *ERE*. For Q, the selected model is the best in terms of σ_a, although there are other models with identical value of σ.

$$ERE = 70.43 + 1.580V_f/V_o + 6.208L_{comb} - 0.190(V_f/V_o)L_{comb} - 0.331(L_{comb})^2 \quad (14)$$

$$Q = 0.479 - 0.046O/F + 0.191V_f/V_o - 0.009(O/F)^2 - 0.028(O/F)V_f/V_o \quad (15)$$

$$ERE = 50.059 + 3.758V_f/V_o + 14.573L_{comb} - 0.05(V_f/V_o)^2 - 0.777(V_f/V_o)L_{comb} - 1.459(L_{comb})^2 + 0.002(V_f/V_o)^2 L_{comb} + 0.046V_f/V_o(L_{comb})^2 + 0.047(L_{comb})^3 \quad (16)$$

$$Q = -0.566 - 0.358O/F + 0.383V_f/V_o - 0.0191(O/F)^2 - 0.107(O/F)V_f/V_o - 0.003(V_f/V_o)^2 + 0.005(O/F)^2 V_f/V_o + 0.002(O/F)(V_f/V_o)^2 \quad (17)$$

Equations (14) and (15) are the reduced quadratic responses and Eqs. (16) and (17) represent the reduced cubic polynomials used for the two objective functions. The t-statistics for the coefficients in Eq. (14) vary between 49.30 and 8.06. For the coefficients in Eq. (15), they vary between 6.28 and 0.52. In Eqs. (16) and (17), the t-statistics of the coefficents vary between 14.69 and 0.31 and 3.36 and 0.74, respectively.

The radial basis networks designed with solverbe are the largest with 15 neurons in the hidden layer for *ERE* network and nine neurons for the Q network. *Solverb* designs a network for *ERE* with 14 neurons in the hidden layer and a network for Q with eight neurons. Compared to RBNN, BPNN has fewer neurons, the number of neurons in the hidden layer are eight and four for the *ERE* and Q networks, respectively. Details of the networks used are listed in Table 3. The *spread constant* used for RBNNs and the *error goal* of the training data is also given in Table 3. The *spread constant* values are selected from the region where the performance of the network is consistent with the variation of *spread constant* (Figures 3-6). The *error goal*, in the case of *solverb*, is selected based on the network with the best performance for the ideal *spread constant* (Figure 7).

The error in predicting the values of the objective function by different schemes is given in Table 4. Several observations can be readily made.

1. Both NNs perform better than the RSM for this data set.
2. Both *solverbe* and *solverb* are of comparable performance.
3. The BPNN helps generate smaller networks and performs at par in comparison to RBNN.
4. The cubic polynomial is more accurate than the quadratic one.

The various models generated are compared with test data in Figures 10 and 11. The curves representing the NN predictions are closer to the data obtained from the injector model than the RSs thereby demonstrating that NN models are able to predict better than the RSs. BPNN performs as well as RBNN but tends to be flat. Due to its lower order, the quadratic polynomial is flat. The cubic polynomial is able to perform better than quadratic.

The optimum solution obtained from various schemes is shown in Table 5 and Figures 12 and 13. The aim is to maximize *ERE* and minimize Q. The trend of the objective functions in the design space is monotonic and hence every model is able to select identical optimum design for the given constraints. The flatness of the polynomials results in bad predictive values of the objective function for the optimum design. The cubic polynomial is more flexible than quadratic but is not consistent. For a V_f/V_o constraint of 4 the quadratic polynomial is more accurate but for higher values of V_f/V_o the cubic polynomial is more accurate. In contrast, the NN models are able to perform well. Since the optimum design happens to be the same as one of the training points, *solverbe* is able to predict the values of the objective function accurately. *Solverb* performs equally well, thereby showing the capability of performance with fewer neurons. Performance of BPNN is not as satisfactory as suggested in Table. 4. For lower constraints of V_f/V_o, it performs poorly

but for higher values of V_f/V_o it is good. This may be due to the selection of fewer neurons in the hidden layers of the networks. Overall, it is still better than to the RSM and demonstrates the flexibility of NN over RS.

As stated by Papila et al[5], when it comes to choosing between NN and polynomials, polynomials are easy to compute. The number of coefficients might be numerous but the linearity of the system expedites the process of coefficient evaluations. This is also the reason RBNN train fast. On the other hand, the weights of BPNN are evaluated through a nonlinear optimization, which slows the training process. Of all the NN presented here, the one designed with the help of *solverbe* is the fastest to train since the values of the weights are set to values of the input dependent variables. *Solverb* trains with the addition of one neuron at a time with weights similar to the input and hence is slower.

3.2 Supersonic Turbine

The generation of RS and the training of the NNs are done with the 76 design points in Table 5A. The analysis was initially done without the constraints and then with the constraints on $(AN)^2$ and V_{puch}.

A quadratic RS was initially generated. Then, cubic terms were included. Cubic terms that are products of three different variables were included because of the number of data available and the number of levels being three. The trend of the design data also suggests the presence of some of these terms. Therefore, the initial cubic equation has 45 terms. A reduced third order RSs for η and W was selected based on the relative performances of different polynomials obtained by removing terms from the initial cubic equation based on t-statistics. The cubic equation was selected based on the evaluated value of σ_a and σ. Table 6 suggests that the reduced cubic polynomial is better than the quadratic polynomial since σ_a is better for the former. The values of σ are comparable.

The t-statistics for the coefficients in the response surface of η varies between 179.72 and 1.2. The coefficients in the response surface of W have t-statistics varying between 822.66 and 0.68. The response surfaces for η and W are as follows:

$$\begin{aligned}\eta &= 0.654 + 2.917D + 5608.217RPM + 1.0287A_{ann}\\ &-0.00729C_v + 0.00544C_b - 0.0399k_r - 4.282D^2\\ &-16283.057D\cdot RPM - 1.572\times10^{7}RPM^2 + 5.228DA_{ann}\\ &-13461.823A_{ann}RPM + 0.0247C_vD + 114.467C_bRPM\\ &-0.00647C_v^2 - 0.0124C_b^2 - 0.163k_rD - 300.440k_rRPM\\ &-0.429k_rA_{ann} - 0.00608k_rC_v - 0.00362k_rC_b - 0.0128k_r^2\\ &+54719.62DA_{ann}RPM + 387.74DC_bRPM - 1.743DA_{ann}k_r\\ &-0.037DC_bk_r - 5384.729A_{ann}RPMk_r\\ &-113.868C_bRPMk_r \end{aligned} \quad (18)$$

$$\begin{aligned}W &= 0.644 + 1.509D - 961.842RPM\\ &-0.627A_{ann} - 0.00452C_v + 0.00412C_b - 0.0255k_r\\ &-3.805D^2 - 7040.351D\cdot RPM + 2.248DA_{ann}\\ &-13.012A_{ann}^2 + 0.00856C_vD + 10.744C_vRPM\\ &-0.00342C_v^2 - 0.0104C_bD - 23.359C_bRPM\\ &+0.0127C_bA_{ann} - 0.00609C_b^2 - 0.0686k_rD\\ &+93.527k_rRPM - 0.227k_rA_{ann} - 0.00324k_rC_v\\ &-0.00183k_rC_b - 0.00673k_r^2 + 93.193DC_vRPM\\ &-162.604DC_bRPM + 921.053Dk_rRPM\\ &+0.342DA_{ann}C_b - 0.692DA_{ann}k_r - 0.0162DC_vk_r\\ &-11.311C_vRPMk_r \end{aligned} \quad (19)$$

The networks designed with *solverb* have 37 and 75 neurons for η and W, respectively in the hidden layer, while those designed with *solverbe* has 76 neurons each. The BPNN uses significantly less number of neurons by generating networks with five and 60 neurons for η and W, respectively, in a single hidden layer. The NN architectures chosen are listed in Table 7.

The accuracy of the various models is tested with the data available in Table 6A and the error is shown in Table 8. *Solverbe* has a poor prediction for η, which might be due to overfitting, but performs well for W. The outcome of Table 8 for the supersonic turbine is similar to that of Table 4 for the injector, except that BPNN is clearly inferior to RBNN. Overall, based on the two cases, it seems that *solverb* is most consistent among all methods evaluated.

The optimum solutions subjected to the constraints, of $(AN)^2$ limited to less than 1.132 (normalized with baseline value) and V_{puch} is limited to less than 1.148 (normalized with baseline value), are presented in Table 9. Since $(AN)^2$ is proportional to the product of square of *RPM* and A_{ann}, and V_{puch} is proportional to D times *RPM*, no NN/RS is generated for them. By comparing the predicted optimal design by the various methods, one observes that *solverbe* and BPNN yield noticeably larger errors in η and W, respectively. *Solverb* and the response surface are more consistent with both η and W. Judged by the error in predicting Δpay, it seems that the RSM is most accurate. However, since the real goal is to maximize Δpay, it is important to note that the actual value of Δpay for the optimal design chosen by the RSM is the worst. Clearly, the large multiplier in Eq. (8) causes bias in relative weighting between η and W, which in turn causes different "apparent" accuracy levels by various methods.

From a design perspective, it is interesting to understand the impact of the constraints from A_{ann} and V_{puch} on the optimal turbine parameters. Such an assessment is offered in Figures 14 and 15. As D, *RPM* and A_{ann} decrease, η, W, V_{puch}, AN^2 and Δpay decrease. C_b and C_v are almost constant over the design space and they do not have any noticeable effect on the objective functions and constraints. In the case of C_v, the BPNN shows a small perturbation for the analysis with the constraint. This might be due to the mapping of some noise by BPNN. Otherwise it is unaffected by the inclusion of the constraints. The stage reaction, K_r, is unaffected as expected, since we are dealing

only with the single stage of the turbine. Hence there is no split on the stage reaction.

4 SUMMARY AND CONCLUSIONS

In the present study, the RS and NN are first constructed using the training data. The test data are then employed to assess the performance of various polynomials and to offer insight into model improvement by removing and adding terms. The best polynomial is selected based on a combined evaluation between σ_a and σ. For the NN, the test data helps evaluate the accuracy of networks with varying neurons in BPNN and varying spread constants in RBNN. Thus the test data are adopted to help select appropriate RSM and NN models. Once an RSM or NN model is constructed, a search for optimum design is carried out using a standard, gradient-based optimization algorithm over the response surfaces represented by the polynomials and trained neural networks.

Based on the results obtained, we have reached the following conclusions.

1. Higher order polynomials perform better than lower order polynomials as they have more flexibility. However, appropriate statistical measure needs to be taken to determine the best terms to include.
2. In the present study, both NN and RSM can perform comparably for modest data sizes.
3. Among all the NN configurations, RBNN designed with solverb seems to be more consistent in performance for both injector and turbine cases.
4. Radial basis networks, even when designed efficiently with *solverb*, tend to have many more neurons than a comparable back-propagation with tan-sigmoid or log-sigmoid neurons in the hidden layer. The basic reason for this is the fact that the sigmoid neurons can have outputs over a large region of the input space, while radial basis neurons only respond to relatively small regions of the input space. Thus, larger input spaces require more radial basis neurons for training.
5. Configuring a radial basis network often takes less time than that for a back-propagation network because the training process for the former is a linear in nature.
6. RBNN with the combined feature of flexibility and linear regression is more accurate than BPNN, which is nonlinear.

Based on the results shown in Tables 4 and 8, it is seen that the RBNN technique performs consistently, and holds promise for the design/optimization of advanced rocket propulsion components. The method adopted here to generate BPNN is not necessarily the most efficient. Given a better method of making the selection of the number of neurons in the hidden layer, BPNN, might be able to perform better. Future work would be aimed at implementing a better designing procedure for back-propagation networks. The work has been carried out with modest data sizes and the training is fast for such cases. Issues related to the number of design variables and training data size are critical for practical design applications, and should be addressed in the future.

5 ACKNOWLEDGMENT

The present study has been supported by NASA Marshall Space Flight Center.

6 REFERENCE:

1. Myers, R. H. and Montgomery, D. C., *Response Surface Methodology – Process and Product Optimization Using Designed Experiment*, John Wiley & Sons, 1995.
2. Hertz, J., Krogh, A. and Palmer, R. G., *Introduction to the Theory of Neural Computation, Lecture Notes Volume I*, Addison-Wesley Publishing Company, 1992.
3. Tucker, P. K., Shyy, W. and Sloan, J. G., "*An Integrated Design/Optimization Methodology For Rocket Engine Injectors*," 34th AIAA / ASME / SAE / ASEE Joint Propulsion conference and Exhibit, July 13-15, 1998 AIAA-98-3513, Cleveland, OH.
4. Shyy, W., Tucker, P. K. and Vaidyanathan, R., "*Response Surface and Neural Network techniques for Rocket Engine Injector Optimization*," 35th AIAA / ASME / SAE / ASEE Joint Propulsion Conference and Exhibit, June 20-24, 1999, AIAA-99-2455, Los Angeles, CA.
5. Papila, N., Shyy, W., Fitz-Coy, N. and Haftka, R. T., "*Assessment of Neural Net and Polynomial-Based Techniques for Aerodynamic Applications*," 17th Applied Aerodynamics Conference, June 28-July 1, 1999, AIAA-99-3167, Norfolk, VA.
6. Norgaard, M., Jorgenson, C. C. and Ross, J. C., "*Neural Network Prediction of New Aircraft Design Coefficients*," NASA TM-112197, 1997.
7. Ross, J. C., Jorgenson, C. C. and Norgaard, M., "*Reducing Wind Tunnel Data Requirements Using Neural Networks*," NASA TM-112193, 1997.
8. Protzel, P. W., Palumbo, D. L. and Arras, M. K., "*Fault Tolerance of Artificial Neural Networks with Applications in Critical Systems*," NASA Technical Paper 3187.
9. Rai, M. M and Madavan, N. K., "*Aerodynamic Design Using Neural Networks*," AIAA Paper No. 98-4928.
10. Greenman, R. M. and Roth, K. R., "*High-Lift Optimization Design Using Neural Networks on a Multi-Element Airfoil*," Proceedings of DETC 98, ASME 1998 Computers in Engineering Conference.
11. Calhoon, D., Ito, J. and Kors, D., "*Investigation of Gaseous propellant Combustion and Associated Injector-Chamber Design Guidelines*," Aerojet liquid rocket company, NASA Cr-121234, Contract NAS3-13379, July 1973.
12. Papila, N., Shyy, W., Griffin, L. W., Huber, F. and Tran, K., "*Preliminary Design Optimization For A Supersonic Turbine For Rocket Propulsion*," 36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Paper No. 2000-3242 (2000), Huntsville, AL.

13. Haftka, R. T., Gurdal, Z. and Kamat, M. P., *Elements of Structural Optimization*, 3rd edition, Kluwer Academic Publishers, 1990.
14. SAS Institute Inc. (1995). *JMP version 3*. Cary, NC.
15. Dernuth, H. and Beale M., *Matlab Neural Network Toolbox*, The Math Works Inc, 1992.
16. Stepniewski, S. W., Greenman, R. M., Jorgensen, C. C. and Roth, K. R., "*Designing Compact Feedforward Neural Models with Small Training Data Sets*," 38th AIAA Aerospace Sciences Meeting and Exhibit, January 10-13, 2000.
17. Coleman, T., Branch, M. A. and Grace, A., *Optimization Toolbox for Use with Matlab*, Version 2, The Math Works Inc, 1999.

O/F	V/V_o	L_{comb}, in.
4,6,8	4	4,5,6,7,8
4,6,8	6	4,5,6,7,8
4,6,8	8	4,5,6,7,8

Table 1: Range of design variables considered for the shear coaxial injector element.

Model #	Coefficient = 0	Terms Removed	Terms Included	σ_a (%)	σ (%)
1		$(V/Vo)^3$	Quadratic and less	0.218	0.280
2	V/Vo^2			0.0857	0.212
3	V/Vo^2	$(V/Vo)^2 L_{comb}$		0.0799	0.214
4		$(V/Vo)^2 L_{comb}$, $(V/Vo)^2$		0.0799	0.214
5		$(V/Vo)^2 L_{comb}$, (V/Vo)	$(L_{comb})^2$	0.0859	0.213
6		$(V/Vo)^2 L_{comb}$, $(V/Vo)^2$	$(L_{comb})^2$, $(V/V_o)^2 (L_{comb})^2$	0.0936	0.212
7		$(V/Vo)^2 L_{comb}$, $(V/Vo)^2$	$(L_{comb})^2$, $(V/V_o)^2 (L_{comb})^2$, $V/Vo(L_{comb})^2$	0.0988	0.212

Table 2(a): Different cubic polynomials for *ERE*. (Dependent variables: V/V_o and L_{comb}, 15 training points, 10 test points) (Errors are given in percentages of the mean value of the responses).

Model #	Coefficient = 0	Terms Removed	Terms Included	σ_a (%)	σ (%)
1		$(O/F)^3$	Quadratic and less	5.445	3.490
2	$(V/Vo)^2$, $(O/F)^3$			5.584	2.234
3	$(O/F)^3$	$(V/Vo)^2$		5.584	2.094
4		$(V/Vo)^2$, $(O/F)^3$		5.584	2.094
5		$(V/Vo)^2$, $(O/F)^3$		5.584	2.234
6		$(V/Vo)^2$, $(O/F)^3$, $(V/Vo)^3$		3.909	2.094
7		$(V/Vo)^2$, $(O/F)^3$, $(V/Vo)^3$	$(V/Vo)^2(O/F)^3$	5.584	2.094

Table 2(b): Different cubic polynomials for Q. (Dependent variables: O/F and V/V_o, 9 training points, 4 test points) (Errors are given in percentages of the mean value of the responses).

Scheme	# of Layers	# of neurons in the hidden layer		# of neurons in the output layer		*Error goal* aimed for during training	
		ERE	Q	*ERE*	Q	*ERE*	Q
RBNN (*Solverbe*)	2	15	9	1	1	0.0 {*sc* = 3.25}	0.0 {*sc* = 1.20}
RBNN (*Solverb*)	2	14	8	1	1	0.001 {*sc* = 1.05}	0.001 {*sc* = 1.05}
BPNN	2	8	4	1	1	0.01	0.01

Table 3: Neural Network architectures used to design the model for shear coaxial injector element. {*sc* = *spread constant*}

Scheme	σ for *ERE* (%)	σ for Q (%)
RBNN (*Solverbe*)	0.207	1.396
RBNN (*Solverb*)	0.133	1.536
BPNN	0.180	0.832
Partial Cubic RS	0.213	2.234
Quadratic RS	0.280	3.490

Table 4: RMS error in predicting the values of the objective function by various schemes for the shear coaxial injector element (Errors are given in percentages of the mean value of the responses).

V_f/V_o	Scheme	O/F	L_{comb}, in.	ERE, %	Q, Btu/in^2-sec
4	RBNN (*Solverbe*)	8.0	7.0	98.60 (0.00)	0.588 (0.00)
	RBNN (*Solverb*)	8.0	7.0	98.60 (0.00)	0.588 (0.00)
	BPNN	8.0	6.9	98.64 (0.14)	0.578 (1.70)
	Partial Cubic RS	8.0	7.0	98.61 (0.01)	0.594 (1.02)
	Quadratic RS	8.0	7.0	98.67 (0.07)	0.591 (0.51)
	Model	8.0	7.0	**98.60**	**0.588**
	Model	8.0	6.9	**98.50**	**0.588**
6	RBNN (*Solverbe*)	8.0	7.0	99.20 (0.00)	0.512 (0.00)
	RBNN (*Solverb*)	8.0	7.0	99.20 (0.00)	0.512 (0.00)
	BPNN	8.0	7.0	99.18 (0.02)	0.513 (0.20)
	Partial Cubic RS	8.0	7.0	99.15 (0.05)	0.500 (2.34)
	Quadratic RS	8.0	7.0	99.17 (0.03)	0.531 (3.71)
	Model	8.0	7.0	**99.20**	**0.512**
8	RBNN (*Solverbe*)	8.0	7.0	99.40 (0.00)	0.493 (0.00)
	RBNN (*Solverb*)	8.0	7.0	99.40 (0.00)	0.493 (0.00)
	BPNN	8.0	7.0	99.41 (0.01)	0.500 (1.42)
	Partial Cubic RS	8.0	7.0	99.42 (0.02)	0.499 (1.22)
	Quadratic RS	8.0	7.0	99.67 (0.27)	0.471 (4.46)
	Model	8.0	7.0	**99.40**	**0.493**

Table 5: Optimal Solutions for fixed values of V_f/V_o and given range of O/F and L_{comb} obtained with NN and RSM schemes for the shear coaxial injector element. (Constraints: $4 \le O/F \le 8$, $4 \le L_{comb} \le 7$) (Errors are given in parenthesis for each prediction is in %)

Type of RS	σ_a for η (%)	σ for η (%)	σ_a for W (%)	σ for W (%)
Quadratic RS	2.507	0.863	0.788	1.281
Reduced Cubic RS	1.949	1.031	0.402	1.223

Table 6: Training and predicting error for different response surfaces of the objective functions of the supersonic turbine. (Errors are given in percentages of the mean value of the responses)

Scheme	# of Layers	# of neurons in the hidden layer		# of neurons in the output layer		*Error goal* aimed for during training	
		η	W	η	W	η	W
RBNN (*Solverbe*)	2	76	76	1	1	0.0 (*sc* = 9.50)	0.0 (*sc* = 9.45)
RBNN (*Solverb*)	2	37	75	1	1	0.001 (*sc* = 6.50)	0.001 (*sc* = 8.35)
BPNN	2	5	60	1	1	0.001	0.001

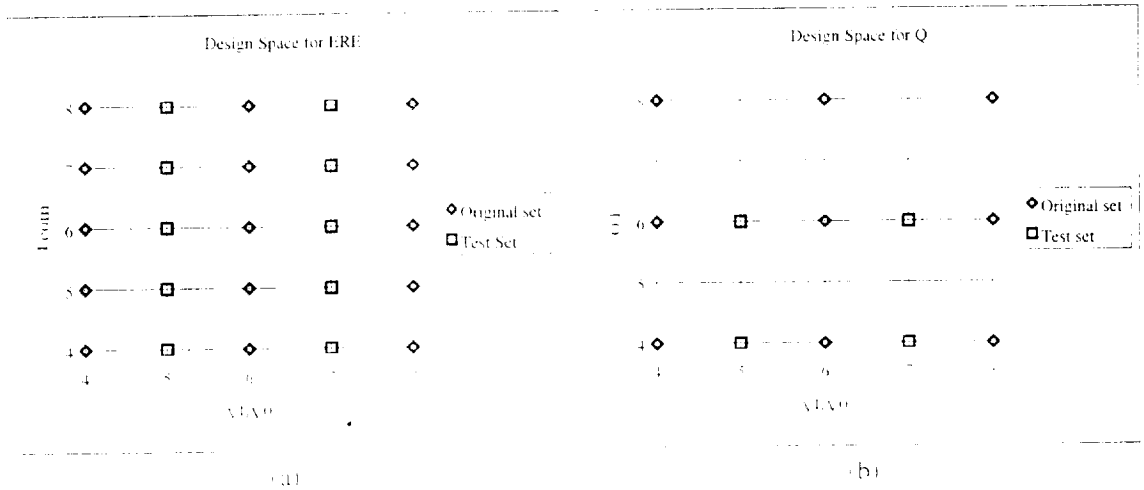
Table 7: Neural Network architectures used to design the models for η, W and $V_{cp,n}$ of the supersonic turbine. (*sc* = *spread constant*).

Scheme	σ for η (%)	σ for W (%)
RBNN (*Solverbe*)	1.251	1.096
RBNN (*Solverb*)	0.292	1.102
BPNN	0.777	2.563
Reduced Cubic RS	1.031	1.223

Table 8: RMS error in predicting the values of the objective function by various schemes for the supersonic turbine. (Error are given in percentages of the mean value of the responses)

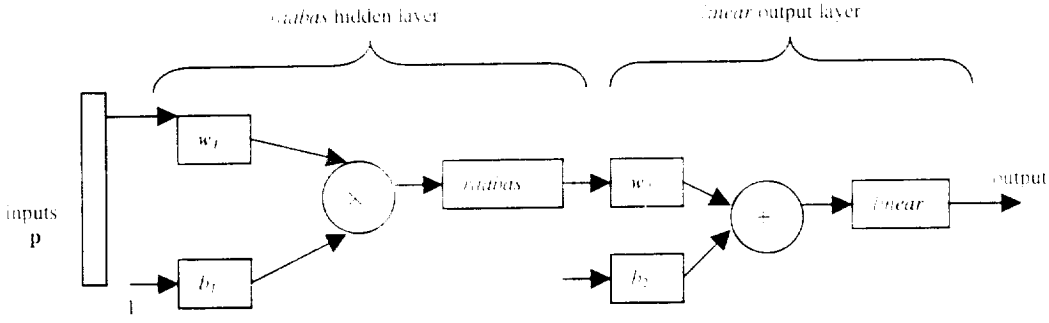
Scheme	D	RPM	A_{ann}	C_v	C_b	K_r	η	W	V_{pitch}	AN^2	Δpay
RBNN (*Solverbe*)	0.972	1.181	0.811	1.443	0.836	0.0	0.810 (5.80)	0.636 (0.74)	1.148	1.132	-0.139 (29.80)
Meanline	0.972	1.181	0.811	1.443	0.836	0.0	**0.766**	**0.641**	**1.148**	**1.132**	**-0.197**
RBNN (*Solverb*)	0.999	1.149	0.857	1.483	0.792	0.0	0.785 (1.75)	0.653 (0.17)	1.148	1.132	-0.177 (9.16)
Meanline	0.999	1.149	0.857	1.483	0.792	0.0	**0.772**	**0.654**	**1.148**	**1.132**	**-0.194**
BPNN	1.024	1.121	0.901	1.168	1.143	0.0	0.793 (2.49)	0.608 (8.63)	1.148	1.132	-0.153 (21.49)
Meanline	1.024	1.121	0.901	1.168	1.143	0.0	**0.772**	**0.666**	**1.148**	**1.132**	**-0.195**
Reduced Cubic RS	0.903	1.272	0.700	1.706	0.871	0.0	0.758 (1.50)	0.591 (2.10)	1.148	1.132	-0.194 (8.40)
Meanline	0.903	1.272	0.700	1.706	0.871	0.0	**0.746**	**0.604**	**1.148**	**1.132**	**-0.211**

Table 9: Optimal Solutions with constraints on V_{pitch} and AN^2 for a supersonic turbine. (Error given in parenthesis for each prediction is in %). (All variables are normalized by their respective baseline values)



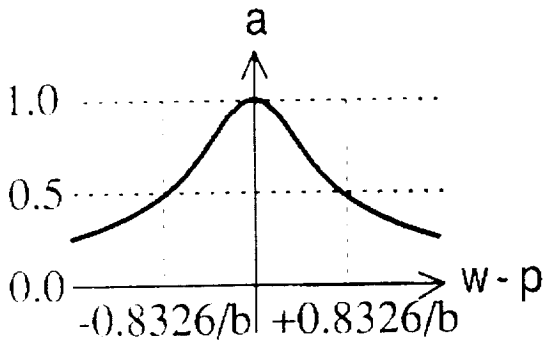
(a) (b)

Figure 1: Design space for (a) *ERE*. (15 Training points, 10 Test points) (b) *Q*. (9 Training points, 4 Test points) for the injector.



(a)

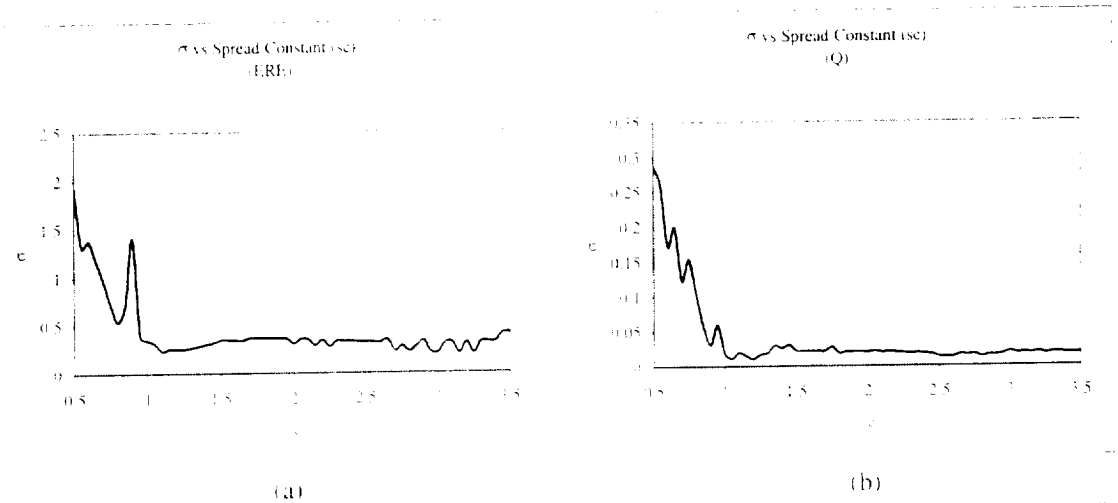
Figure 2: (a) Radial basis network.



$$a = radbas\left(dist\left(w, p\right)\times b\right)$$

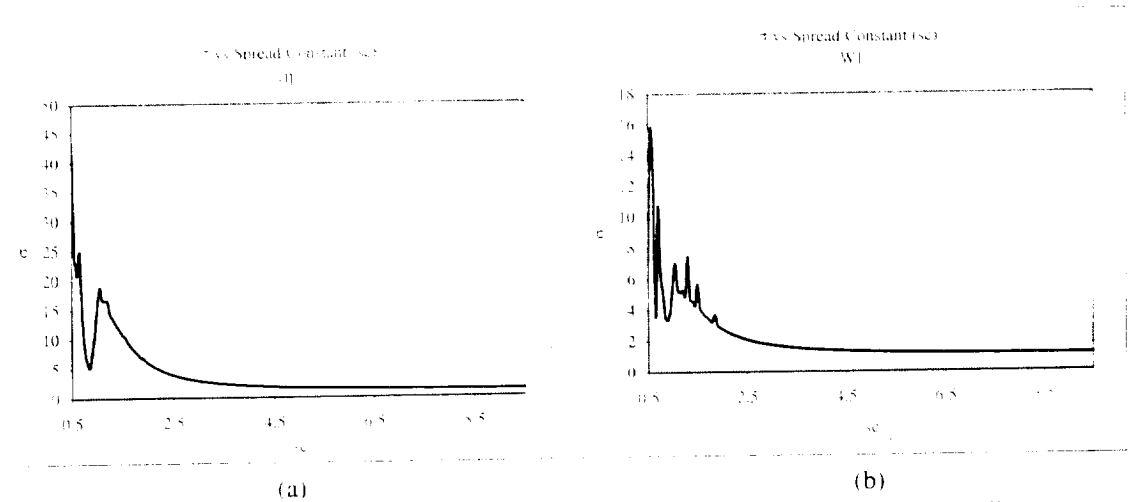
(b)

Figure 2: (b) Transfer function, *radbas*.



(a) (b)

Figure 3: Comparison of σ for different NN designed with *solverbe* for (a) *ERE* (%) and (b) *Q* (Btu/in^2-sec).



(a) (b)

Figure 4: Comparison of σ for different NN designed with *solverbe* for (a) η (%) and (b) *W* (lbs).

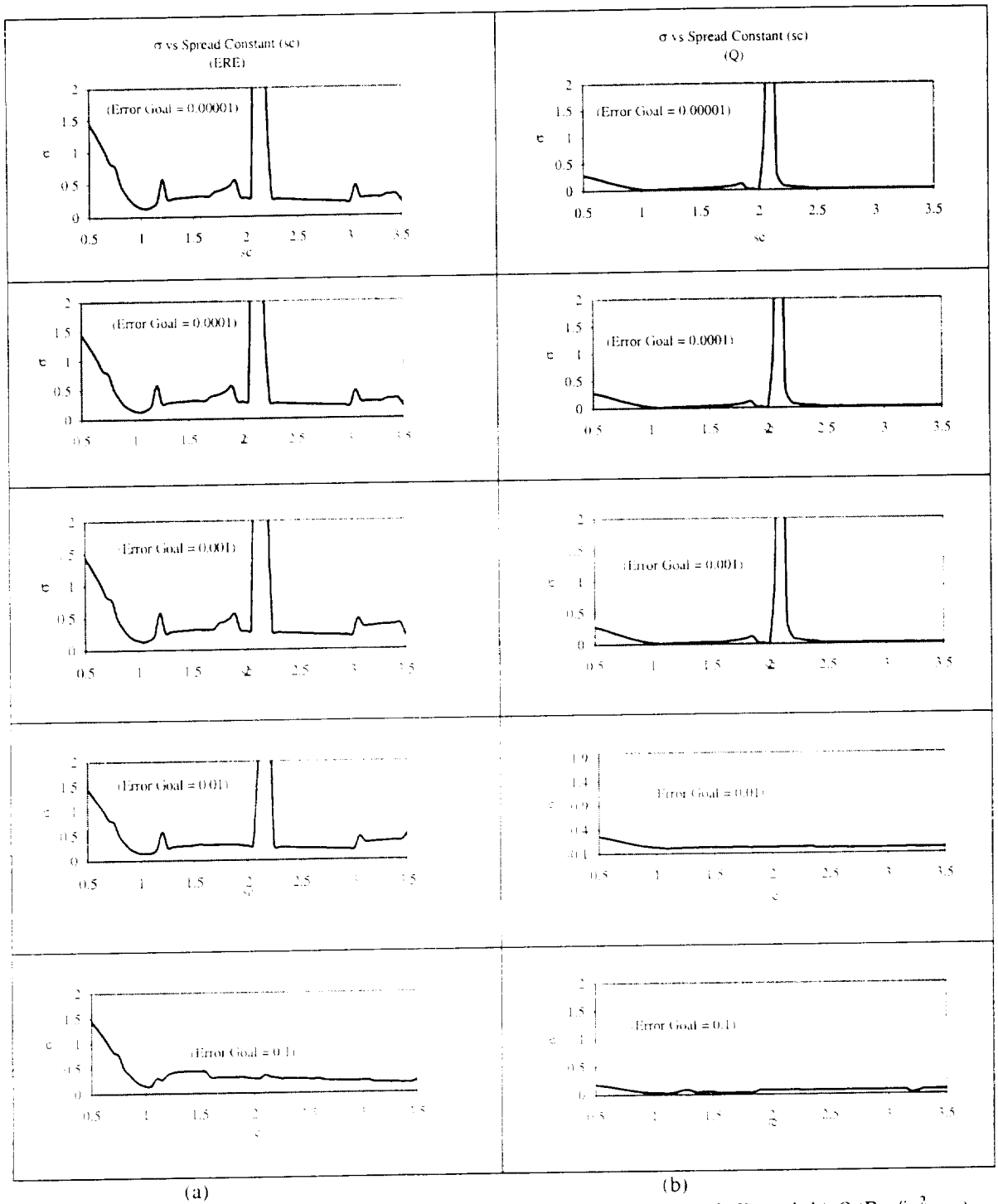


Figure 5: Comparison of σ for different NN designed with *solverb* for (a) *ERE* (%) and (b) *Q* (Btu/in^2-sec).

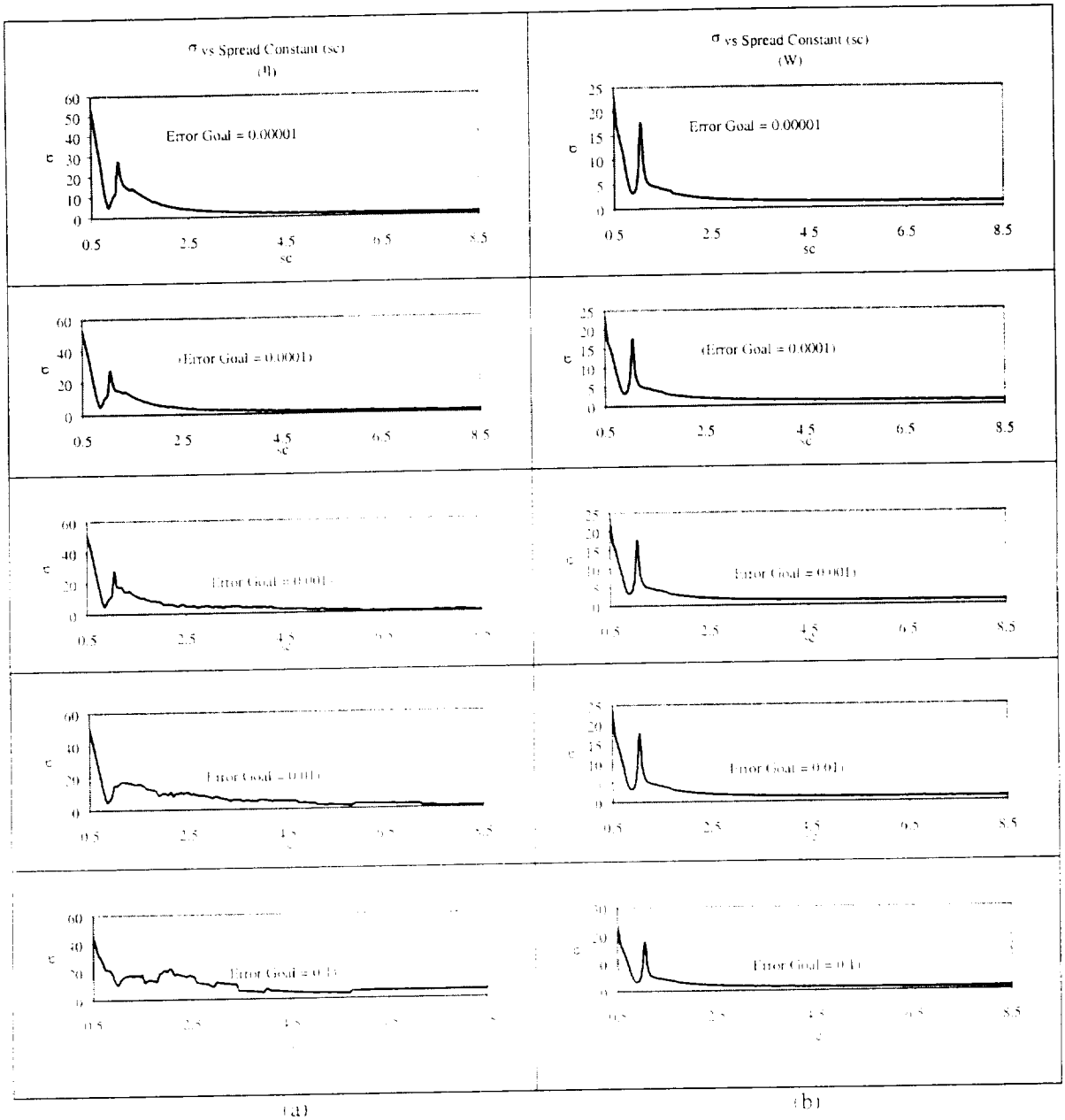
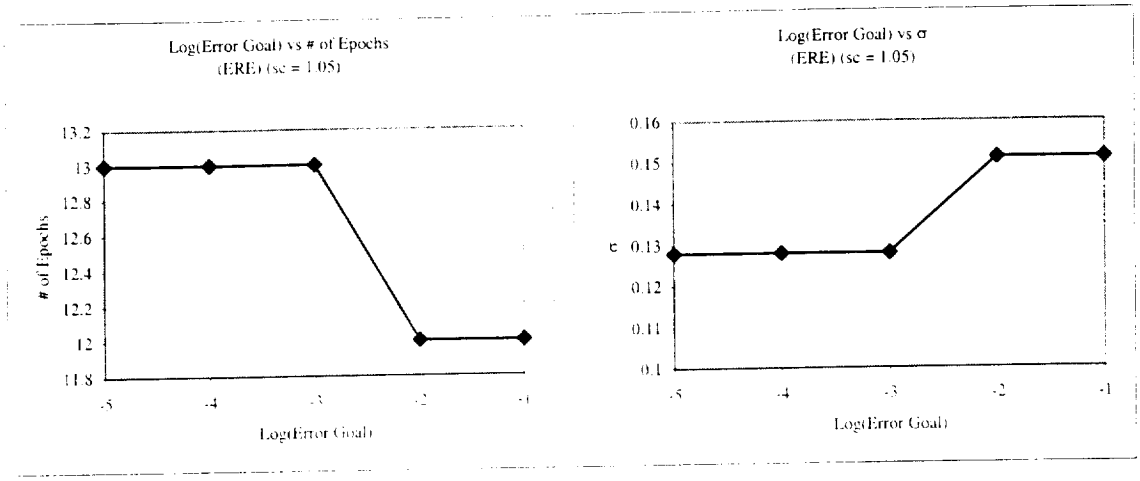
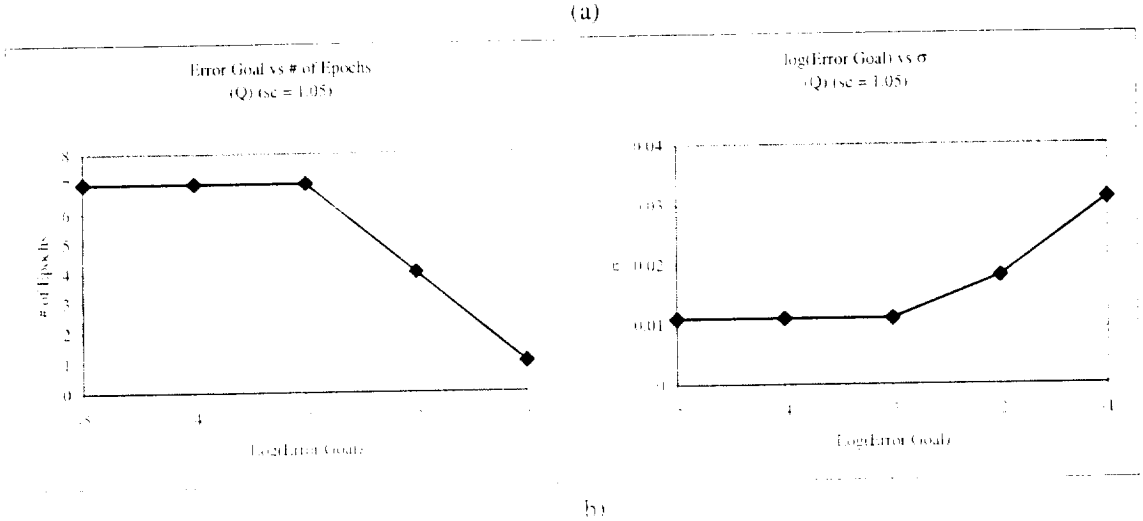


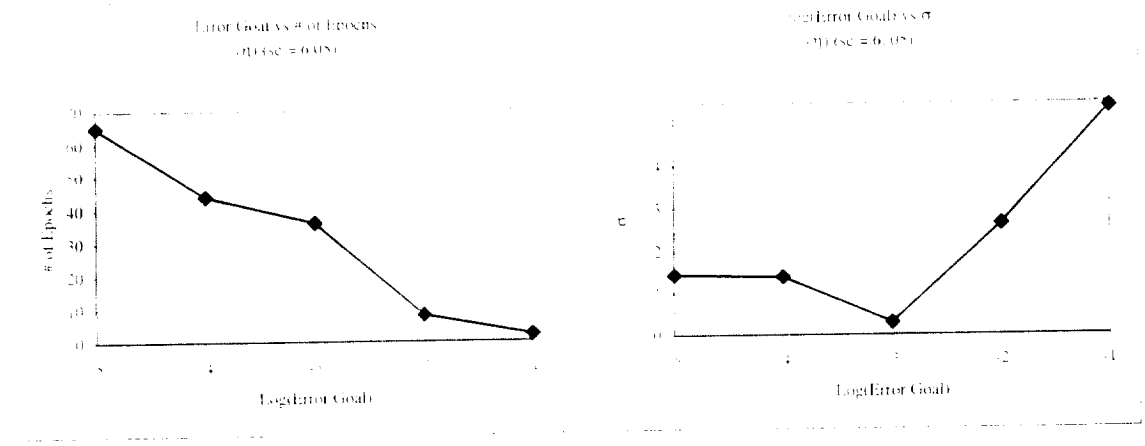
Figure 6: Comparison of σ for different NN designed with *solverb* for (a) η (%) and (b) W (lbs).



(a)

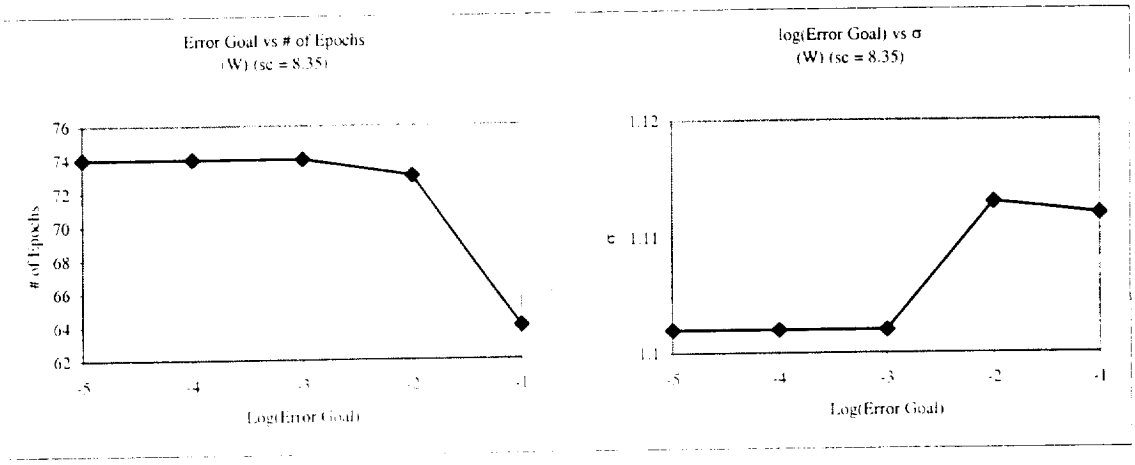


(b)



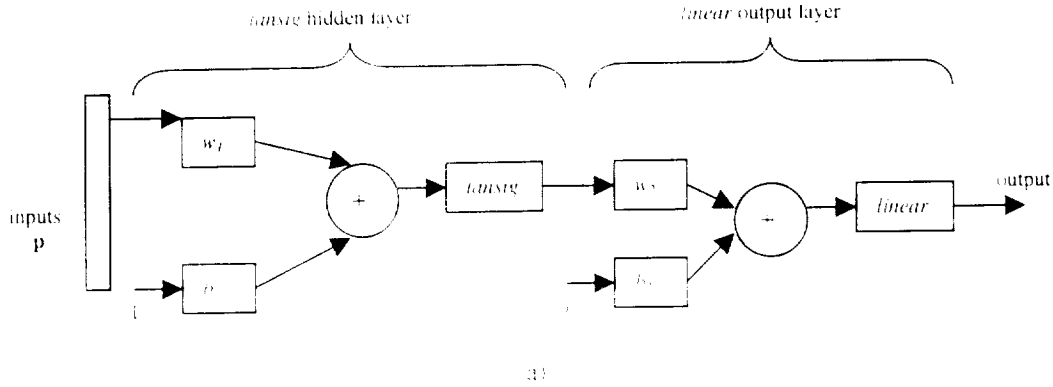
(c)

Figure 7: Comparison of *Error goal* vs. number of Epochs and σ for networks trained with *solverb*. (a) *ERE* (%), (b) *Q* (Btu/in^2-sec), (c) *η* (%) and (d) *W* (lbs). (Continued)

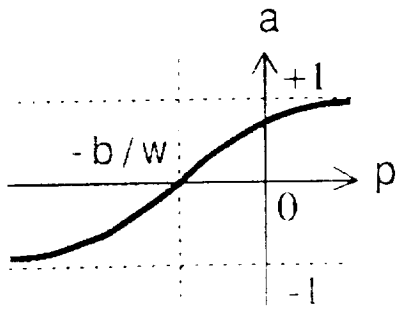


(d)

Figure 7: Comparison of *Error goal* vs. number of Epochs and σ for networks trained with *solverb*. (a) *ERE* (%), (b) Q (Btu/in^2-sec), (c) η (%) and (d) W (lbs).



(a)



$a = tansig(w \cdot p + b)$

(b)

Figure 8: (a) Back-propagation Network, (b) Transfer function, *tansig*.

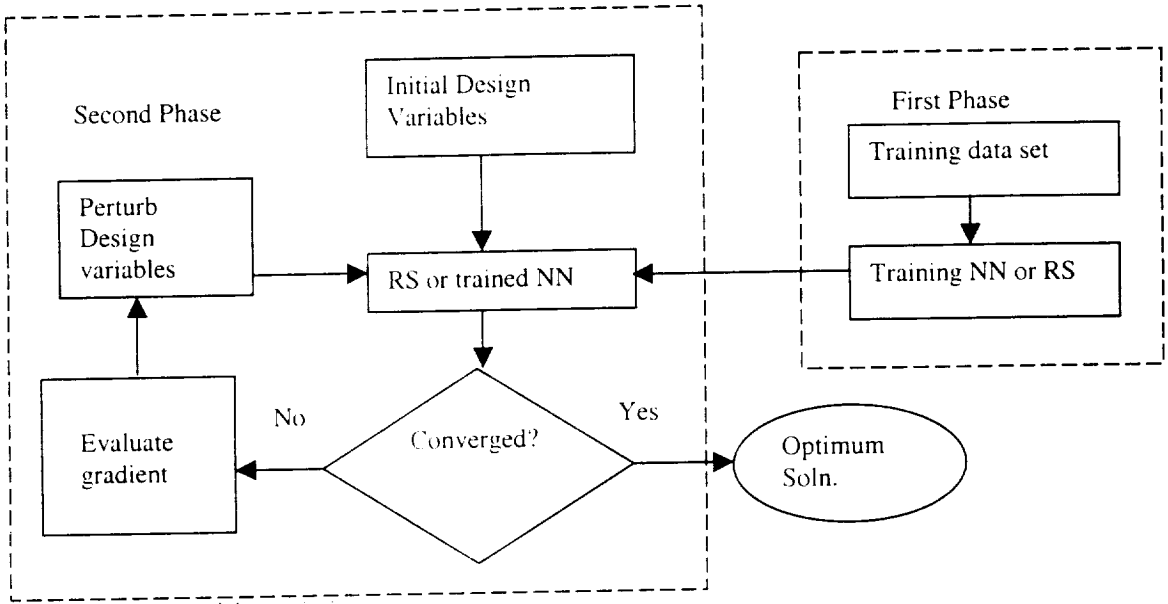


Figure 9: Schematic of the optimization process.

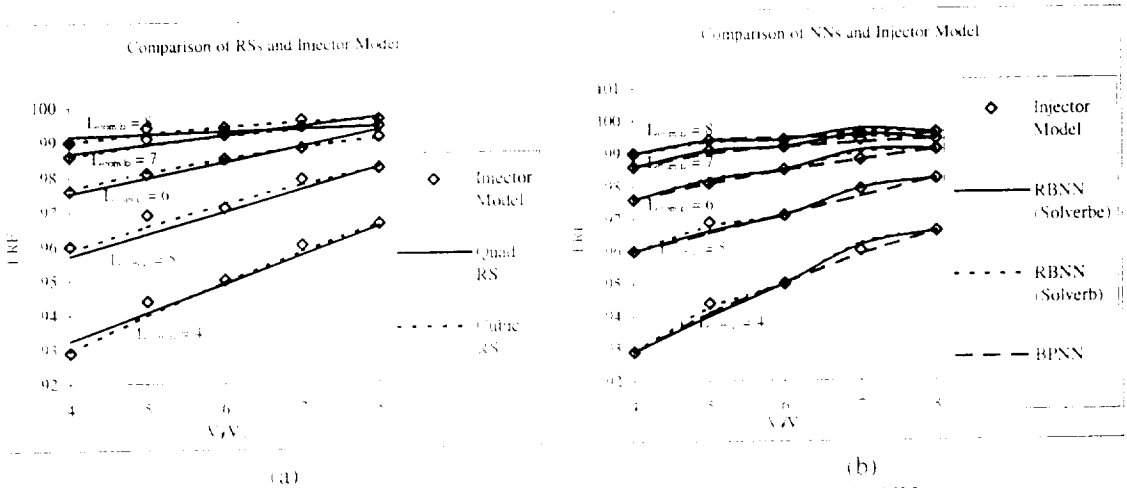


Figure 10: Comparison of models with test data for ERE of the injector. (a) RSs (b) NNs.

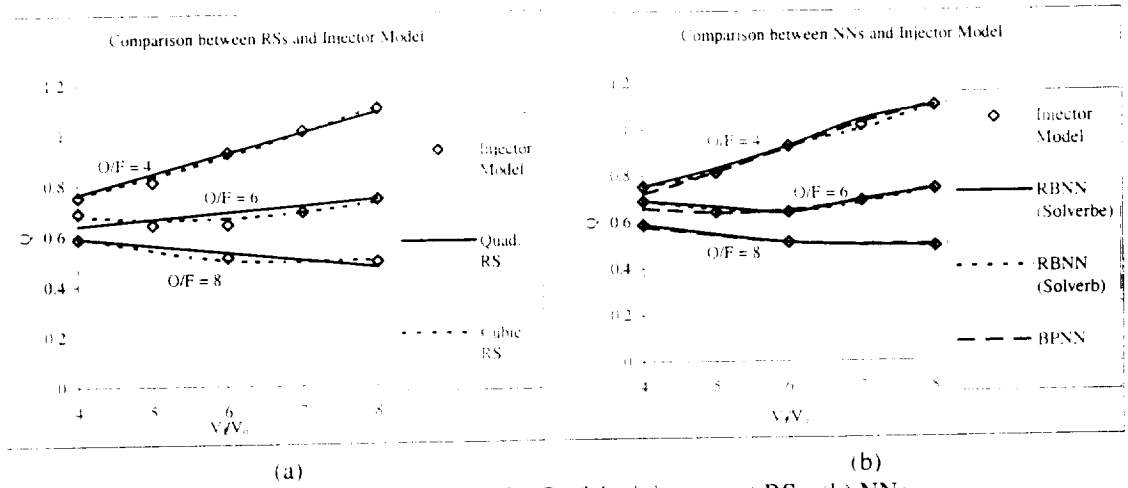
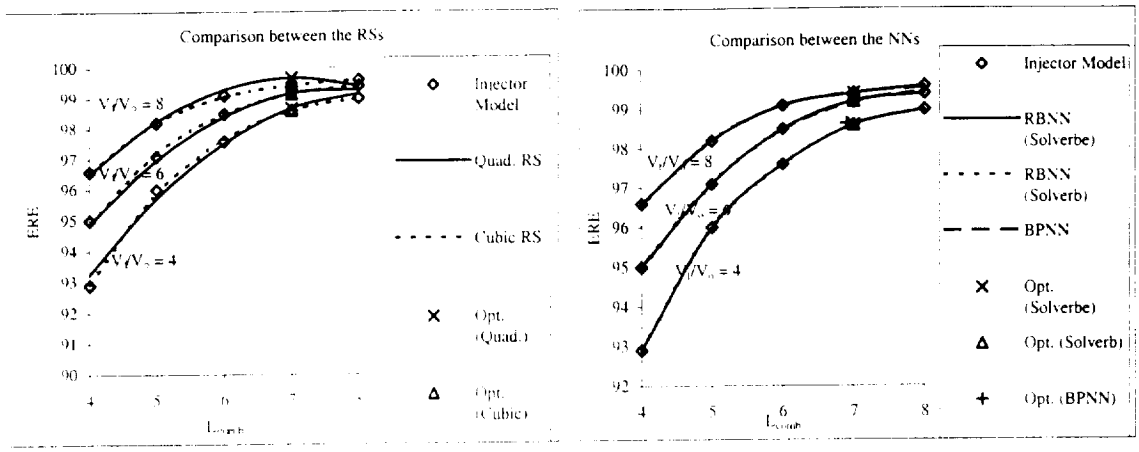
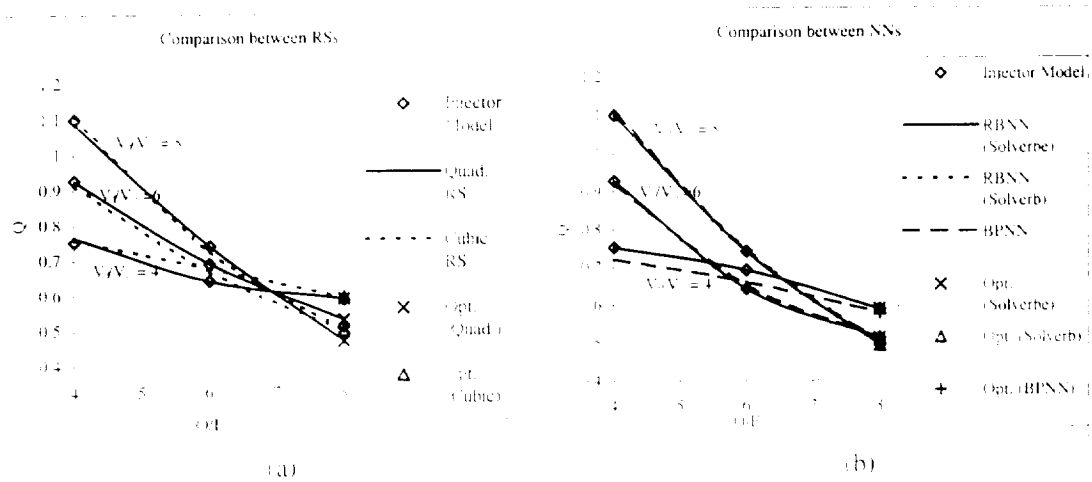


Figure 11: Comparison of models with test data for Q of the injector. (a) RSs. (b) NNs.



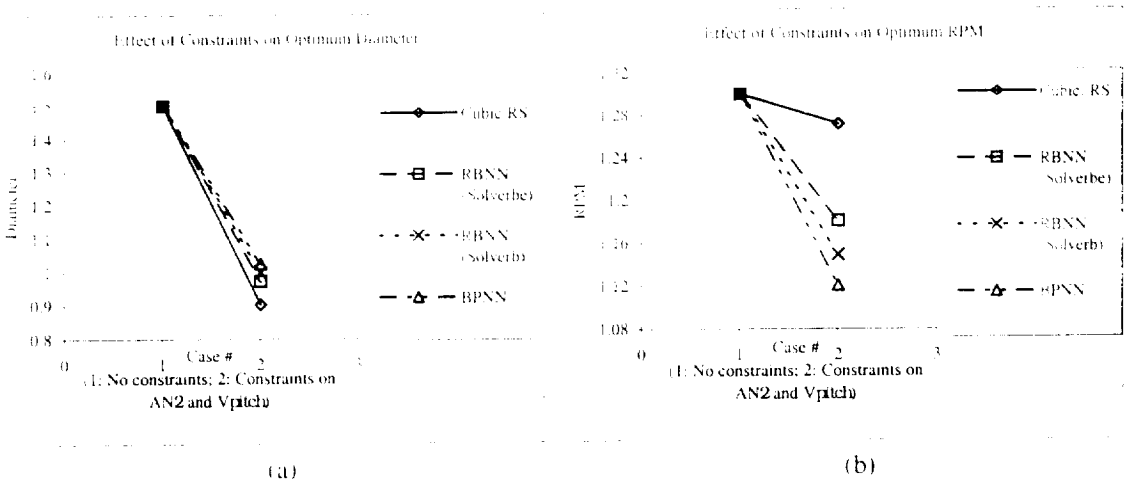
(a) (b)

Figure 12: Performance of models for ERE of the injector. (a) RSs (b) NNs.



(a) (b)

Figure 13: Performance of models for Q of the injector. (a) RSs (b) NNs.



(a) (b)

Figure 14: Effect due to presence (case 1) or lack of constraints (case 2) on design variables. (a) Optimum Diameter, D (in.), (b) Optimum RPM, (c) Optimum Annulus Area, A_{ann} (in.^{2}), (d) Optimum Vane Axial Chord, C_v (in.), (e) Optimum Blade Axial Chord, C_b (in.) and (f) Stage Reaction, K_r (%)(All variables are normalized by their respective baseline values) (Continued).

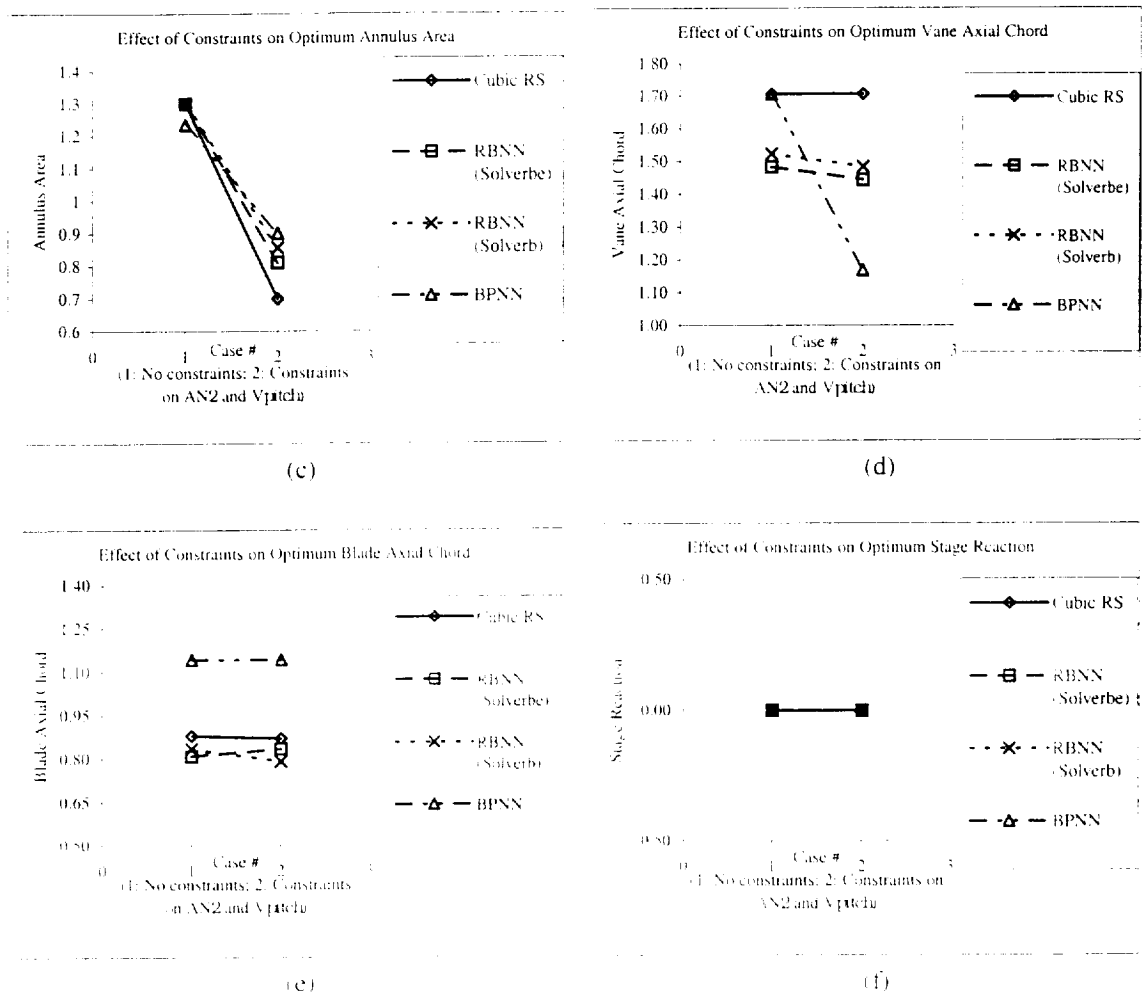


Figure 14: Effect due to presence (case 1) or lack of constraints (case 2) on design variables. (a) Optimum Diameter, D (in.), (b) Optimum RPM, (c) Optimum Annulus Area, A_{annl} (in.^{2}), (d) Optimum Vane Axial Chord, C_v (in.), (e) Optimum Blade Axial Chord, C_b (in.) and (f) Stage Reaction, K_r (%) (All variables are normalized by their respective baseline values).

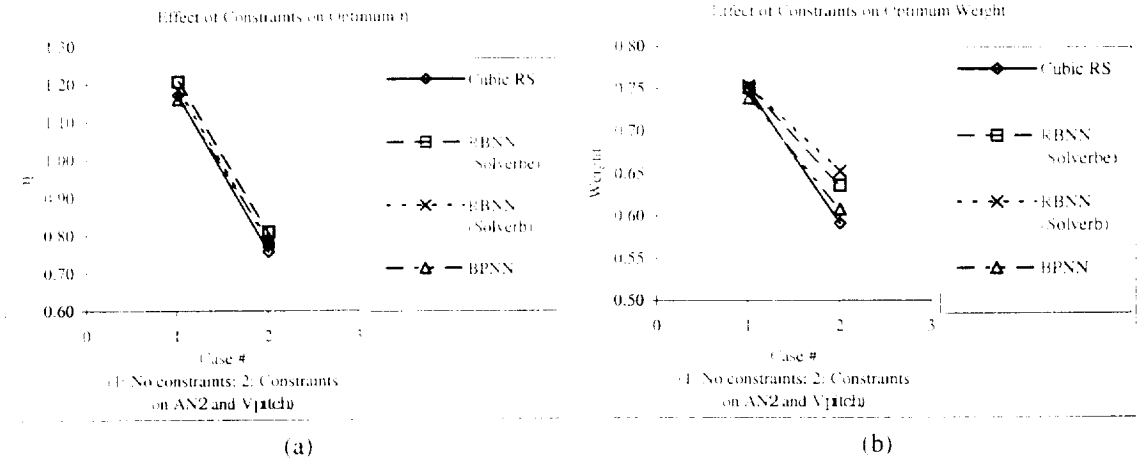
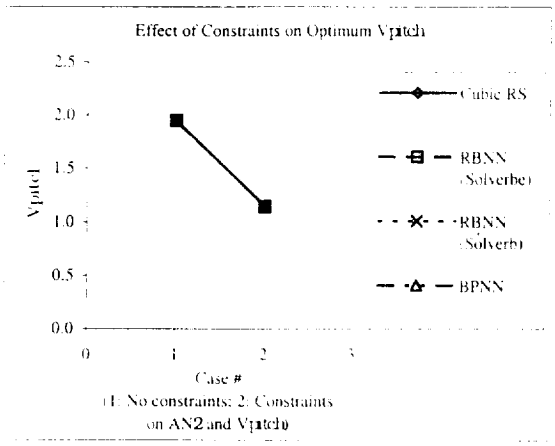
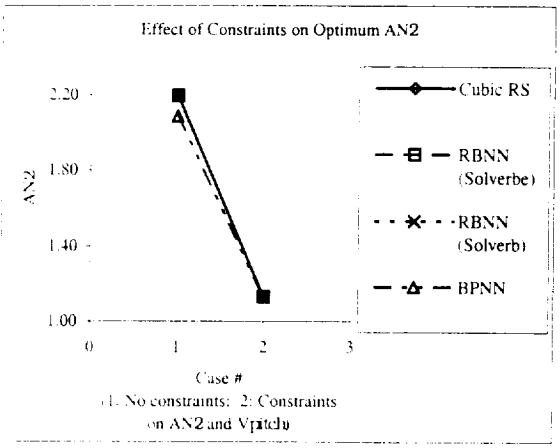


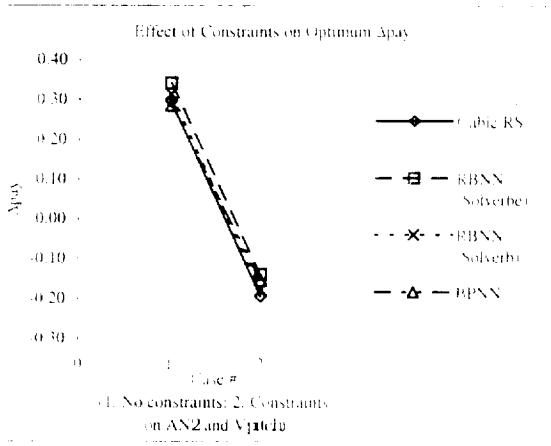
Figure 15: Effect due to presence (case 1) or lack of constraints (case 2) on objective functions. (a) Optimum Efficiency, η (%), (b) Optimum Weight, W (lbs), (c) Optimum pitch speed, V_{pitch} (in. /sec), (d) Optimum Annulus Area X RPM, AN^2 (in^2*rpm^2) and (e) Optimum Incremental Payload, Δpay (lbs) (All variables are normalized by their respective baseline values) (Continued).



(c)



(d)



(e)

Figure 15: Effect due to presence (case 1) or lack of constraints (case 2) on objective functions. (a) Optimum Efficiency, η (%), (b) Optimum Weight, W (lbs), (c) Optimum pitch speed, V_{pitch} (in. /sec), (d) Optimum Annulus Area × *RPM*, AN^2 ($in^2 \times rpm^2$) and (e) Optimum Incremental Payload, Δpay (lbs) (All variables are normalized by their respective baseline values).