INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2021; 11(1):113-141

ANALYSIS OF DOUBLE-LAYER BARREL VAULTS USING DIFFERENT NEURAL NETWORKS; A COMPARATIVE STUDY

A. Kaveh[*, †] and A. Eskandari
School of Civil Engineering, Iran University of Science and Technology, Narmak, Tehran, Iran

ABSTRACT

The artificial neural network is such a model of biological neural networks containing some of their characteristics and being a member of intelligent dynamic systems. The purpose of applying ANN in civil engineering is their efficiency in some problems that do not have a specific solution or their solution would be very time-consuming. In this study, four different neural networks including FeedForward BackPropagation (FFBP), Radial Basis Function (RBF), Extended Radial Basis Function (ERBF), and Generalized Regression Neural Network (GRNN) have been efficiently trained to analyze large-scale space structures specifically double-layer barrel vaults focusing on their maximum element stresses. To investigate the efficiency of the neural networks, an example has been done and their corresponding results have been compared with their exact amounts obtained by the numerical solution.

Keywords: structural analysis; double-layer barrel vaults; neural networks; feedforward backpropagation; radial basis function; extended radial basis; generalized regression neural network; element stresses.

Received: 5 December 2020; Accepted: 12 February 2021

1. INTRODUCTION

1.1 Barrel vaults

The interest of utilizing skeletal space frames to cover huge column-free bays due to their remarkable advantages such as low cost, lightweight, small deflection because of high inherent stiffness, three-dimensional action in carrying loads, etc., has been increased.

[*]Corresponding author: School of Civil Engineering, Iran University of Science and Technology, Narmak, Tehran, Iran
[†]E-mail address: alikaveh@iust.ac.ir (A. Kaveh)

Distinct types of skeletal space frames, which are often categorized as grids, domes, and barrel vaults, are mainly put in practice in sports stadiums, exhibition centers, assembly halls, swimming pools, shopping arcades, and industrial buildings which are typical instances of structures where large unobstructed areas and minimum interference from internal supports are architecturally required.

Barrel vault, a very popular type of skeletal space frames, is one of the oldest forms which was improved in the nineteenth century by the use of iron bars [1]. These bars act as horizontal tie members to withstand the horizontal thrust from the vault and make structures supported on slender walls or columns. The earlier types of barrel vaults were constructed as single-layer structures and to cover large spans double-layer systems can be utilized which are statically indeterminate. In double-layer barrel vaults, due to the rigidity, the risk of instability can almost be neglected. The use of this type of barrel vaults escalates the stiffness of the vault structures and provides structural systems of great potential, capable of having bays in excess of 100 m. Consequently, they are considered large-scale structures, so their design requires enough precision to be optimum [2]. Regarding their optimization, Kaveh et al. have done many types of research [3-8].

Braced barrel vaults [1] are formed on a cylindrical surface with a circular basic curve. However, there is a possibility of utilizing other forms such as a parabola, ellipse, or funicular. Fig. (1) depicts the habitual arrangement of braced barrel vaults. The type of braced barrel vaults and the location of supports expressed as L/R, influence its structural behavior where L and R represent the distance between the supports in the longitudinal direction and the radius of curvature of the transverse curve, respectively.

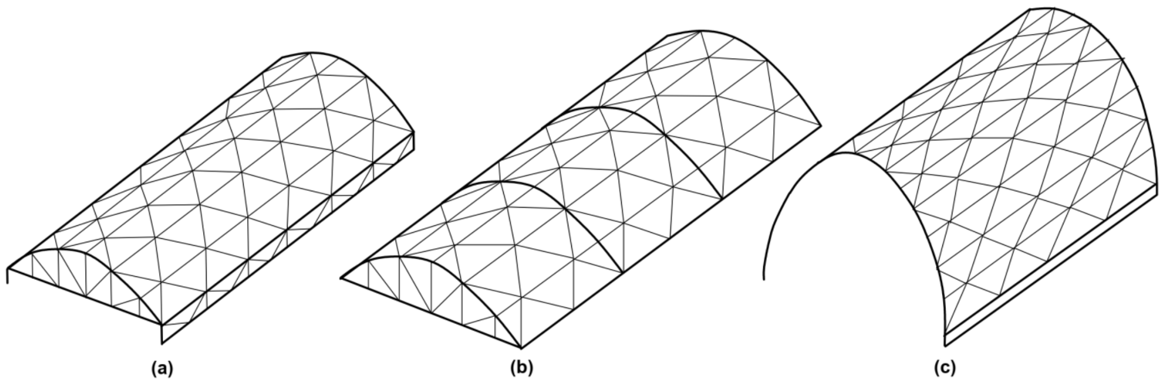


Figure 1. Braced barrel vaults

The response of braced barrel vaults while subjected to loading has been identified in three principal modes in the following [1]:

(a) Arch mode:

If the structure is only supported along its sides, it behaves as an arch structure, Fig. (2a). This type of structure relies on circumferential bending stiffness to resist displacement in the radial direction and also can be considered as a relatively soft structure.

(b) Beam mode:

If a barrel vault is only supported at its ends, it acts as a simple beam, Fig. (2b), which has longitudinal compressive stresses near the crown of the vault, longitudinal tensile stresses towards the free edges, and shear stresses towards the supports. To compare with the arch structure, this type of structure is relatively stiff and brittle. The end support could be provided by a thin diaphragm. If the diaphragm is rigid, the structure will behave as an encastre beam instead of a pinned one. Also, If the diaphragm is rigid but allowed to rotate, then this will only affect stresses near the end of the structure due to the Saint-Venant effect.

(c) Shell mode:

If a barrel vault is supported along both sides and both ends, Fig. (2c), the structure responds as a thin shell. Despite the fact that this structure looks as if it is partly an arch and partly a beam, there is a need for radial deflection of the structure to behave as an arch structure which is prevented by transverse shear in the shell. This means that the loads acting on the structure will be supported by a complex system of forces acting tangentially to the surface of the barrel vault.

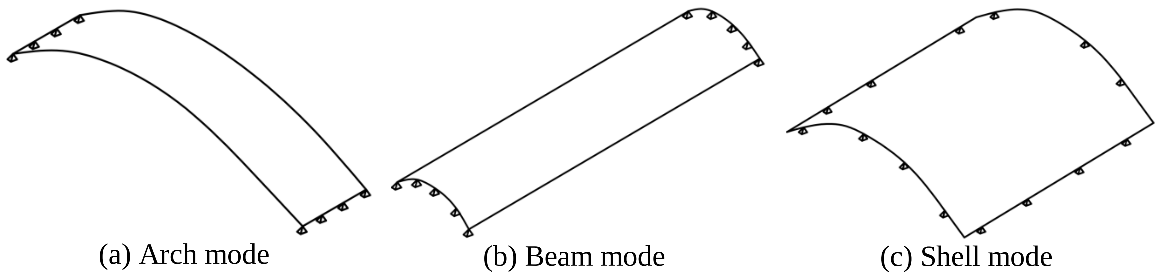


Figure 2. Principal modes of braced barrel vaults

1.2 Formex algebra and formian fundamentals

In the analysis of double-layer barrel vaults, numerous nodes and members are involved which make them time-consuming. So, configuration processing is an issue that can be simplified utilizing the concept of Formex algebra [9] or the theory of graphs [10]. Formex algebra is a mathematical system consisting of a set of abstract objects known as formices and a number of rules in accordance with which these objects can be manipulated. It implements a favorable basis for solutions to problems in data generation and computer graphics. This mathematical system has simplified configuration processing in large scale structures such as double-layer grids, barrel vaults, domes. The early ideas on which Formex algebra is based were developed by Prof. H. Nooshin during the years 1972-1973. The advantages of Formex formulation for data generation have been outlined by Nooshin [11] in the following:

- The process of data generation for structural systems is significantly simplified. The more complicated a structure is, the more beneficial it is to resort to Formex formulation.
- The decrease in storage requirements can be enormous. For large-scale structures, the storage needed data in the explicit form is several hundred times more than that demanded for data stored in the Formex format.
- While the design proceeds, the formulation can be modified as simply as changing a document on a word processor, and it can be manipulated to reflect the contemporary status after making any necessary changes in geometry, loading, or support conditions.

- It is drawing a configuration to generate the data for the intended structure, that is essential. However, once the Formex formulation is obtained, the upshot can be displayed on a monitor or printed.
- An implied or actual nodal ordering system is required neither for a Formex description of a configuration nor for obtaining stress analysis by utilizing the input data in a compatible stress analysis software.
- The Formex data generation is more advantageous compared with other available schemes that are analysis software dependent and consequently lack versatility. On the other hand, the input data produced by the Formex formulation can be employed, via Formian, in a variety of analysis software, such as SAP2000, ABAQUS, and LUSAS.

In the Formex algebra, structural elements are defined based on their positions towards three basic directions U1, U2, and U3 being considered as suitable counterparts to axes X, Y, and Z in the three-dimensional Cartesian system. Parallel lines relative to these basic directions are named normat lines, and points of their intersections are known as normat points. Programmers determine distances between these normat lines individually. In the following instance, Fig. (3), the distances are considered a unity. In the Formex algebra, for example, the location of node C set in the lower layer is represented by three successive numbers separated by colons and put inside square brackets as: [1, 1, 0] an element of the top layer located between nodes A and B, Fig. (3), is described as: [2, 2, 1; 4, 2, 1], where the third coordinate indicates the position of the lower or the top layer measured along with the third direction U3. Member CD located in the lower layer is described as: [1, 1, 0; 3, 1, 0], while the cross-brace AD is expressed as: [2, 2, 1; 3, 1, 0].

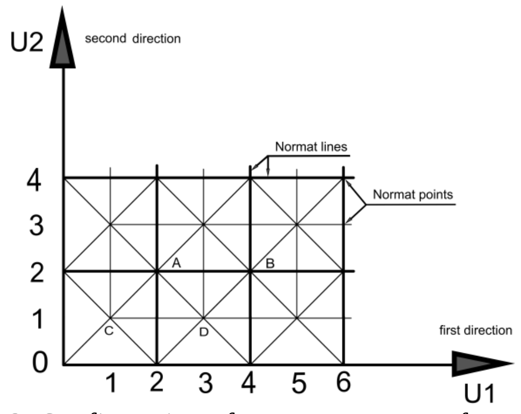


Figure 3. Configuration of component parts of space truss

In the following, a sample of the double-layer barrel vault, Fig. (4), and its corresponding programming language in Formian, Fig. (5), are shown.

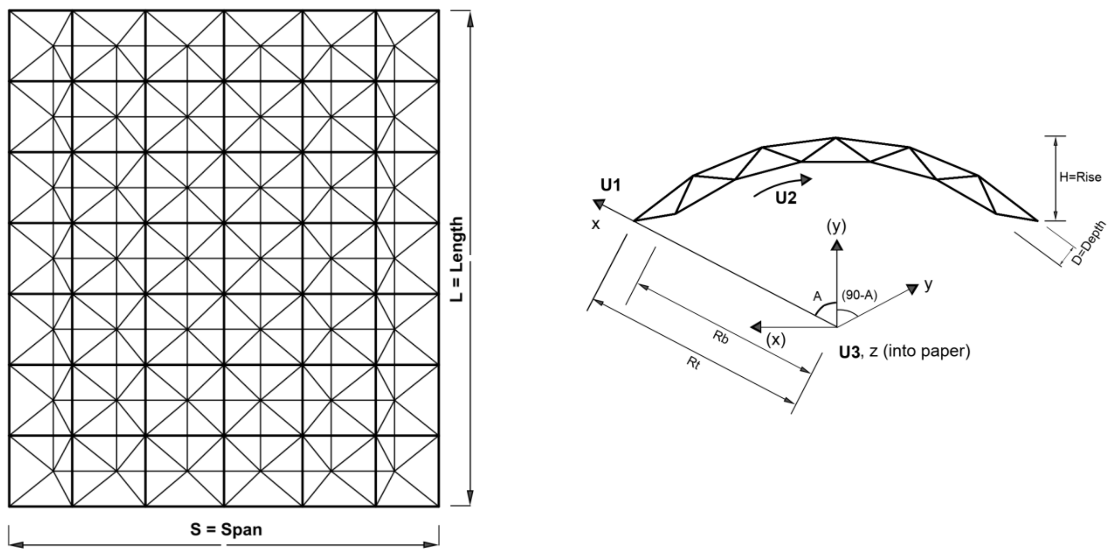


Figure 4. Plan and elevation of the double-layer barrel vault

```
(*) Double-layer Barrel Vault (*)
M= ; (*) top modules along with U2 (*)
N= ; (*) top modules along with U3 (*)
S= ; (*) span (*)
H= ; (*) rise (*)
D= ; (*) depth (*)
L= ; (*) length (*)
v=0; (*) view adjuster (*)
A=2*atan|(2*H/S); (*) sweep angle (*)
Rt=S/(2*sin|A); (*) top radius (*)
Rb=Rt-D; (*) bottom radius (*)
TOP=rinit(M,N+1,2,2)|[Rt,0,0; Rt,2,0]#
rinit(M+1,N,2,2)|[Rt,0,0; Rt,0,2];
BOT=rinit(M-1,N,2,2)|[Rb,1,1; Rb,3,1]#
rinit(M,N-1,2,2)|[Rb,1,1; Rb,1,3];
WEB=rinit(M,N,2,2)|lamit(1,1)|[Rt,0,0; Rb,1,1];
B=TOP#BOT#WEB;
B1=bc(1,A/M,L/(2*N))|B;
BV=verad(0,0,90-A)|B1;
use &,vm(2),vt(2),
vh(v,2.75*Rt,-Rt,0,0,Rt,0,1,Rt);
clear; draw BV;
```

Figure 5. Formian programming of the double-layer barrel vault

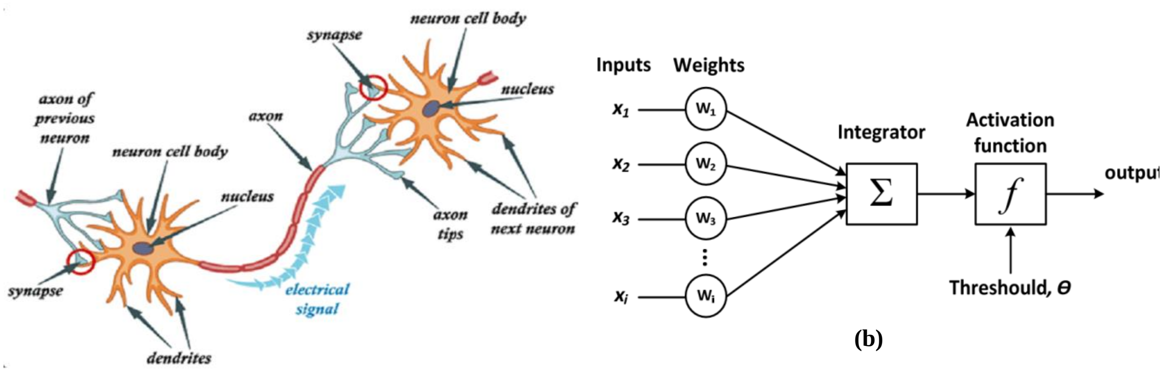
1.3 Neural networks

Artificial neural networks (ANNs) are not only massive parallel computational models but also mathematical machine learning techniques that imitate the mechanism of learning in biological organisms consisting of a large number of processing elements called neuron which are connected to one another by the use of axons and dendrites and connecting regions between axons and dendrites are referred to as synapses, Fig. (6a) [12, 13]. The structure of ANNs, which is similar to that of a human brain, contains a large number of processing elements each of which can have many inputs but only one output. Each output, however, branches out to the input of many other processing elements. The topology of ANNs might be in the form of either feed-forward or recurrent and their corresponding learning can be supervised or unsupervised depending on their topology. Feedforward neural networks are usually trained utilizing supervised training procedures. During the process, output signals are obtained by the translation of input signals using an appropriate transfer function. Weights, which are strengths for processing elements, are applied to input signals according to their importance. After receiving all input signals by processing elements, it computes the total input being received from its input paths based on these weights, Fig. (6b), i.e.,

$$net_{Pt} = \sum_{s} w_{st} o_{Ps} \tag{1}$$

where w_{st} is the connecting weight from s the source layer to t the target layer and o_{Ps} is the output produced using pattern p as the result of the input $o_{P(s-1)}$.

Although ANNs do not really solve the problem in a strictly mathematical sense, they demonstrate information processing characteristics that provide an approximate solution to a given problem. A trained network demonstrates some various pros over the mathematical process of computation. It provides a swift mapping of a given input into the desired output quantities. The application of ANNs is even more significant in nonlinear structural analysis and also as a rapid reanalyzer in optimal design. There has been significant interest in the application of neural networks in subjects of structural analysis and design [14-26]. In this paper, four artificial neural networks, namely FFBP, RBF, ERBF, and GRNN are applied to analyze space structures. Various aspects of these nets and parameters affecting the performance of each net are investigated. One example is presented and a comparison has been made on the performance of these nets. The response of each net for the same input has been compared to illustrate their efficiency.



(a) (b)

Figure 6. Neural network: (a) biological neuron, (b) artificial neural network

1.3.1 Feedforward backpropagation

1.3.1.1 Backpropagation neural nets

The backpropagation learning algorithm was first introduced by Parker [27] and Werbos [28] independently, and it became generalized and developed into a workable process by Rumelhart et al [29]. Various applications have been ascertained for backpropagation since 1986. Single-layer networks may not have the decision-making complexity required to solve a broad range of structural engineering problems. Due to this deficiency, multi-layer networks have been promoted from single-layer networks that are capable of learning any constant mapping to arbitrary accuracy. The training procedure of this net comprises three stages including the feedforward of the input training pattern, the calculation and backpropagation of the correlated error, and the adjustment of the weights. The delta error backpropagation algorithm is habitually employed for its supervised learning. It is essentially a steepest descent algorithm to adjust the weight connection strength. After training, the application of the net involves only the computations of the feedforward phase. Even if the training is time-consuming, a trained net can produce its output so rapidly. The method of training employed in this paper is backpropagation. This approach is one of the most successfully and widely utilized algorithms in artificial neural networks. In backpropagation, learning is accomplished when a set of training pairs is propagated through a network consisting of an input layer, one or more hidden layers, and an output layer. Each layer has its corresponding units and weight connections.

The procedure of the output layer is computing outputs, then subtracted from their exact amounts (target outputs) to obtain a sum of square error that illustrates the level at which the network has learned the input-output data and gradient of the learning process.

The output of a unit is obtained by passing the value of the net-sum through the activation function as:

$$O_{pt} = f_t(net_{pt}) \tag{2}$$

The sum of the squares of the error for a single pattern at the output units is calculated by:

$$E_p = \frac{1}{N_{output}} \sum_{outputs}^{network} (t_p - o_p)^2 \tag{3}$$

where E_p is the measure of the error of pattern p, t_p is the target output, o_p is the obtained output at the output nodes, and N_{output} is the number of the output units.

After processing all patterns, the total sum of the error is given by:

$$E = \frac{1}{PN_{output}} \sum_{P} \sum_{outputs}^{network} (t_p - o_p)^2 \tag{4}$$

where P is the number of all patterns in the training set. The delta signal for the output layer is defined as:

$$\mathrm{\delta_{pt}} = \left(\mathrm{t}_{tp} - \mathrm{o_{pt}}\right)\dot{\mathrm{f}}_{\mathrm{t}}\left(\mathrm{net_{pt}}\right) \tag{5}$$

and for the hidden layers:

$$\mathrm{\delta_{pt}} = \dot{\mathrm{f}}_{\mathrm{t}}\left(\mathrm{net_{pt}}\right)\sum_{\mathrm{k}} \mathrm{\delta_{pt}}\,\mathrm{w_{kt}} \tag{6}$$

The derivative of the activation function that is utilized in Eq. (5) is given as:

$$\dot{\mathrm{f}}_{\mathrm{t}} = \frac{\partial \mathrm{o}_{pt}}{\partial \mathrm{net}_{pt}} = \mathrm{o}_{pt}\left(1 - \mathrm{o}_{pt}\right) \tag{7}$$

where k indicates an upper layer unit (the output layer is the uppermost, and the input layer is the lowermost layer).

The learning rule associated with the backpropagation method is known as the generalized delta rule [31]. According to this rule, to modify the weights in the network, for a given pattern p, the following relation can be employed:

$$\Delta\mathrm{w_{st}}(n) = \eta\mathrm{\delta_{pt}}\mathrm{o}_{pt} + \alpha\mathrm{w_{st}}(\mathrm{n} - 1) \tag{8}$$

where n is the input-output presentation number and h is the learning rate and a is the momentum.

The momentum,a , is a fraction that is multiplied by the previous weight change and added to the current weight change; a should be positive and less than 1. These two terms are not independent and should be selected and adjusted as a pair. Although the dynamic adjustment of both parameters can generally accelerate convergence, a tremendous value for either h or a may lead to instability in the training process. Once the change to the weights is observed, the new weights are measured as:

$$\mathrm{w_{st}}(n + 1) = \mathrm{w_{st}}(n) + \Delta\mathrm{w_{st}}(n) \tag{9}$$

The errors are usually huge at the commencement of the learning process, which requires more changes in weights in the preceding stages of the learning process. By applying the backpropagation procedure, the network determines delta signals for the output layer and hidden layers, utilizing Eqs. (5) and (6), respectively. These deltas are employed to measure the changes for all amounts of weight according to Eq. (8) and the subsequent weight values using Eq. (9).

1.3.1.2 Feedforward neural networks

In artificial neural networks, feedforward neural networks (FFNN) incorporate a set of processing elements called neurons [30, 31]. Feedforward neural networks, trained with a backpropagation learning algorithm, Fig. (7), are the most popular neural networks employed for a wide variety of applications. There are no cycles or loops in feedforward networks. In such networks, the information moves only in the forward direction, from the input layer, through the hidden layers, and to the output layer. Through the procedure, each neuron computes the sum of the weights of the inputs at the presence of a bias and passes this sum through an activation function, such as the sigmoid function, so that the output is obtained. This process can be represented as follows:

$$h_j = \sum_{i=1}^{m} iw_{j,i}\, x_i + hb_j \tag{10}$$

where $iw_{j,i}$ is the weight connecting neurons [$i = (1,\ 2,...\ m)$, and [$j = (1,\ 2,...,\ [r,k])$ hb_j is a bias in the hidden layer, m is the total number of neurons in the input layer, x_i is the corresponding input data, r, k is the total number of neurons in the hidden layers, and n is the total number of neurons in the output layer.

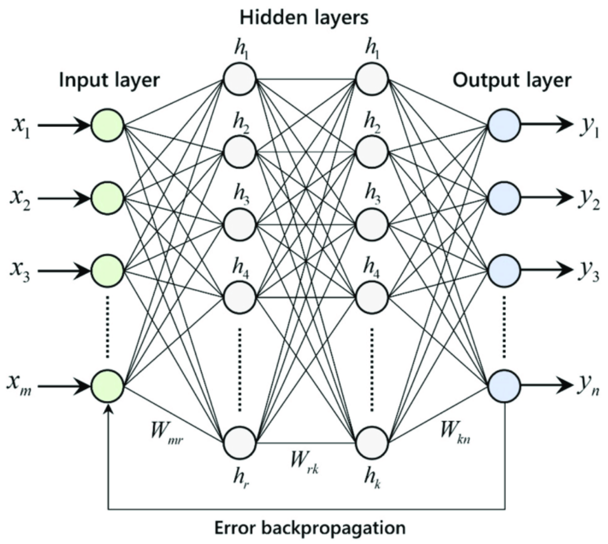


Figure 7. The topology of the feedforward backpropagation network with two hidden layers

The training process is attained to modify the weights and bias until some error criterion is observed. Above all, one predicament is to select a proper training algorithm. Moreover, it is very complex to design the neural network because of numerous elements, such as the number of neurons in the hidden layer, the interconnection between neurons and layer, error function, and activation function that influence the performance of training. The training procedure begins with arbitrary values of the weights, which might be random numbers, and proceeds iteratively. Each iteration of the complete training set is termed an epoch. In each one, the network adjusts the weights in the direction that decreases the error. As the iterative process of incremental adjustment continues, the weights progressively converge to the locally optimal set of values. A multitude of epochs is usually required before training is completed. For a supplied training set, the backpropagation learning proceeds in either pattern mode or batch mode. The former performs weight updating after each training pattern. The latter, however, does so after all training sets. The pattern mode is preferred over the batch mode because it demands less local storage for each synaptic connection. Furthermore, given that the patterns are haphazardly presented to the network, the use of pattern-by-pattern updating of weights makes the search in weight space stochastic, which reduces the probability of being trapped in a local minimum through the back-propagation algorithm. On the other hand, the use of the batch mode of training provides a more accurate estimation of the gradient vector [32, 33].

In the prediction mode, data flows forward through the network, from inputs to outputs. The network processes one example at a time, producing an estimation of the output value(s) based on the input value(s). The resulting error is utilized to evaluate the quality of prediction of the trained network. In back-propagation learning, the procedure commences with a training set and employs the back-propagation algorithm to compute the synaptic weights of the network. The aim is to design a generalized neural network. A network is termed generalized if it precisely computes the input-output relationship for input-output patterns never used in training the network [34]. When the learning process has been done by too many iterations (i.e. the neural network is overtrained or overfitted, while there is no difference between overtraining and overfitting), the neural network may memorize the training data which makes the neural network less capable of generalizing similar input-output patterns. The neural network produces nearly thorough results for data from the training set but fails for data from the test set. Overfitting can be compared to an improper choice of the degree of polynom in the polynomial regression. Severe overfitting can occur with noisy data, even when there are numerous training cases than weights. The basic condition for good generalization is the existence of an adequately large set of training cases. This training set must be at the same time and a representative subset of all cases that are the aim of generalization. The importance of this condition is related to the fact that there are two various types of generalization: interpolation and extrapolation. Interpolation uses cases that are more or less surrounded by nearby training cases; otherwise, that is extrapolation. In particular, cases that are outside the range of the training data need extrapolation. Although interpolation can often be done reliably, extrapolation is notably unreliable. Consequently, it is important to have sufficient training data to evade the need for extrapolation.

1.3.2 Radial basis function

The idea of employing RBFs as approximation functions was first introduced by Hardy [35] in 1971 when he applied the multiquadric RBFs to fit irregular topographical data. RBF neural networks were addressed in 1988 [36], which have recently drawn much attention due to their significant generalization capability and also simple network structures that limit unnecessary and lengthy calculations in comparison with the Multilayer Feedforward

Network (MFN). Some studies of universal approximation theorems on RBF have revealed that any nonlinear function over a compact set with arbitrary accuracy can be approximated by RBF neural network [37, 38]. Radial Basis Function Neural Networks (RBF) are universal approximators and a particular type of feedforward neural networks with radial basis functions employed as activation functions. RBF neural networks are commonly implemented for regression, classification, pattern recognition, and time series forecasting problems. Besides their great global approximation capability, RBFs benefit from other powerful features such as the compact structure, being able to approximate any continuous network, and their tolerance to noise. Similarly, to any other neural networks, a fundamental element in the performance of RBF is the learning procedure. The aspiration of this process is to tune the parameters of the neural network so as to minimize some error criterion. An RBF neural network with a typical topology of a single hidden layer has three principal parameters including the connection weights, widths, and centers. The conventional approach for training an RBF is to utilize two sequential stages training process. In the first stage, the centers of the hidden layer and the widths are obtained by taking the advantage of some unsupervised clustering algorithm such as k-means [39], vector quantization [40], or decision trees [41]. In the second stage, the connection weights between the hidden layer and the output layer are set. Habitually, these weights are determined linearly by applying the simple linear least squares (LS), the orthogonal least squares (OLS) algorithms [42, 43], or a gradient descent algorithm [44].

An RBF network in its simplest configuration is a three-layer feedforward neural network whose first layer corresponds to the inputs of the network, its second is a hidden layer consisting of several RBF nonlinear activation units, and the last one correlates to the final output of the network. Activation functions in RBF neural networks are conventionally implemented as Gaussian functions. Fig. (8) displays an illustration of the RBF topology.

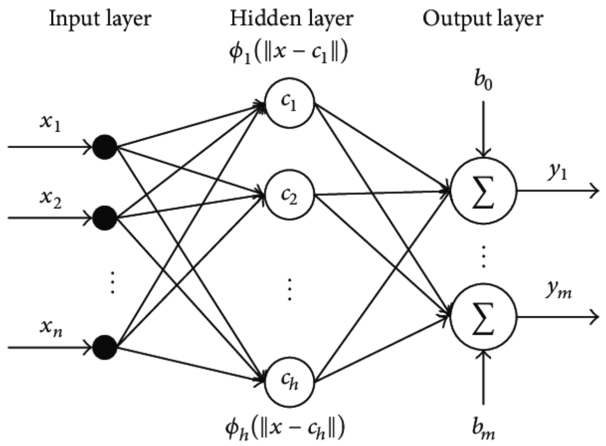


Figure 8. Structure of RBF neural network

To demonstrate the working flow of the RBF, a set of data D having N patterns of (x_p, y_p) is supposed where x_p is the input of the data set, and y_p is the actual output. The output of the i-th activation function ϕ_i in the hidden layer of the network can be computed applying

Eq. (11) based on the distance between the input pattern x and the center i.

$$\phi_i(\|\mathrm{x} - \mathrm{c}_i\|) = \exp\left(-\frac{\|\mathrm{x} - \mathrm{c}_i\|^2}{2{\sigma_i}^2}\right) \tag{11}$$

where $\| \,.\, \|$ is the Euclidean norm, c_i and σ_i are the center and width of the hidden neuron i, respectively.

Then, the output of the node k of the output layer of the network can be calculated utilizing the Eq. (12):

$$y_k = \sum_{j=1}^{n} \omega_{jk}\, \phi_j(x) \tag{12}$$

Most classical methods for training RBFs are conducted in two steps. In the first one, the centers and widths are determined using some unsupervised clustering algorithm, whereas in the second step the connection weights between the hidden layer and the output layer are found in a way such as an error criterion like Mean Squared Error (MSE) is minimized over all the data set.

The concept of Radial Basis Functions networks arises from the theory of function approximation. The main features of these networks are outlined as follows:

1. They are two-layer feedforward neural networks.
2. Their hidden nodes implement a set of radial basis functions (e.g. Gaussian functions).
3. Training the networks is divided into two stages; initially, the weights from the input to hidden layer are determined, and then the weights from the hidden to output layer are obtained.
4. The training/learning is pretty swift.
5. RBF networks are very accurate at interpolation.

1.3.2.1 Comparison between RBF networks and BP networks

There are various similarities and distinctions between RBF and BP:

Similarities

1. Being nonlinear feedforward neural networks.
2. Being universal approximators.
3. Being employed in alike application fields.

Distinctions

1. An RBF network has normally a single hidden layer, whereas BP neural networks can have any number of hidden layers.
2. RBFs are fully connected, while it is common for BP networks to be only partially connected.
3. In BPs, computation nodes in separate layers share a common neuronal model and not necessarily the same activation function. In RBF networks, however, hidden nodes (basis functions) perform very differently and have a very distinct purpose from output nodes.

4. In RBFs, the argument of each hidden unit activation function is the distance between the input and the weight (RBF centers), whereas, in Backpropagation, it is the inner outcome of the input and the weight.
5. BP nets are frequently trained with a single global supervised algorithm, while RBF neural networks are usually trained one layer at a time with the first layer unsupervised.
6. BPs construct global approximations to nonlinear input-output mapping with distributed hidden representations, whereas RBF nets conduce to apply localized nonlinearity at the hidden layer to construct local approximations. Although for approximating nonlinear input-output mappings, the RBF networks can be trained much quicker, BP may need fewer parameters.

1.3.3 Extended radial basis function

There are a few impediments to the conventional RBF approach, which might be limiting its effectiveness as a metamodeling strategy, hence introducing a novel extension to the RBF approach that successfully overcomes its contemporary deficiencies is required. More particularly, RBF neural networks generate an interpolating surface that is unique with respect to a supplied set of prescribed data points. Solving a square system of linear equations results in the interpolating surface. The incapability of preventing resulting spurious local minima is a considerable deficiency. Several researchers have suggested modifications to the conventional RBF networks to overcome some of their restrictions. In 1987, Powell [45] offered the theory of developing the performance of RBFs by augmenting them with a set of polynomial functions and imposing limitations, which leads to a scheme of linear equations. Although these modifications enhance the performance of RBF neural networks, they do not lead to an increase in their flexibility regarding the model building process. There have been significant strides in constructing metamodels with smoothing properties. Girosi [46] proposed the Support Vector Machine (SVM) technique from the context of data interpolation. The SVM technique takes smoothness constraints into the consideration to be incorporated into the model building process. Having considered the pros and cons of RBFs, there is a necessity of introducing an approach not only containing the advantages of RBF nets but also covering its drawbacks. Therefore, a method called the Extended Radial Basis Function (ERBF) approach has been proposed [47], which intentionally evades the notion of unique solvability. This new technique simplifies building metamodels for designers and ultimately leads to more precise and efficient metamodels. The following illustrates the development of a new type of basis functions known as the Nonradial Basis Functions (NRBFs), which will produce an integral component of the ERBF approach.

1.3.3.1 Nonradial basis functions

The multiquadric RBF is conceivably one of the most efficient basis functions among the available RBFs which was the motivation for embedding some of its characteristics within NRBFs. Nonradial basis functions, as can be realized by the name, are not functions of the Euclidean distance r. On the contrary, they perform as individual coordinates of generic points x relative to a given data point x^i, in each dimension independently. The coordinate vector is defined as $\xi^i = x - x^i$, which is a vector of *m* components, each corresponding to a single coordinate dimension. Hence, ξ^i_j is the coordinate of each point *x* relative to the data point x^i along the j-th dimension. Fig. (9) represents the variation between the Euclidean distance *r* employed in RBFs and the relative coordinates ξ applied to NRBFs for a two-dimensional case. The NRBFs for the i-th data point and the j-th dimension is expressed as ϕ_{ij} composed of three different components:

$$\phi_{ij}(\xi^i_j) = \alpha^L_{ij}\phi^L(\xi^i_j) + \alpha^R_{ij}\phi^R(\xi^i_j) + \beta_{ij}\phi^\beta(\xi^i_j) \tag{13}$$

where α^L_{ij}, α^R_{ij}, and β_{ij} are coefficients to be determined based on given problems; and the superscripts *L* and *R* represent left and right, respectively. Fig. (10) shows a generic basis function along with one of the dimensions for random values of α^L, α^R, and β. The functions of ϕ^L,ϕ^R, and ϕ^β are illustrated in Table 1. Four distinct regions (I–IV) are drawn in Fig. (10), each corresponding to a row in Table 1.

Table 1: Nonradial basis function

Region	**Range of ξ^i_j**	$\boldsymbol{\phi^L}$	$\boldsymbol{\phi^R}$	$\boldsymbol{\phi^\beta}$
I	$\xi^i_j \le -\gamma$	$(-n\gamma^{n-1})\xi^i_j + \gamma^n(1-n)$	0	ξ^i_j
II	$-\gamma \le \xi^i_j \le 0$	$(\xi^i_j)^n$	0	ξ^i_j
III	$0 \le \xi^i_j \le \gamma$	0	$(\xi^i_j)^n$	ξ^i_j
IV	$\xi^i_j \ge \gamma$	0	$(n\gamma^{n-1})\xi^i_j + \gamma^n(1-n)$	ξ^i_j

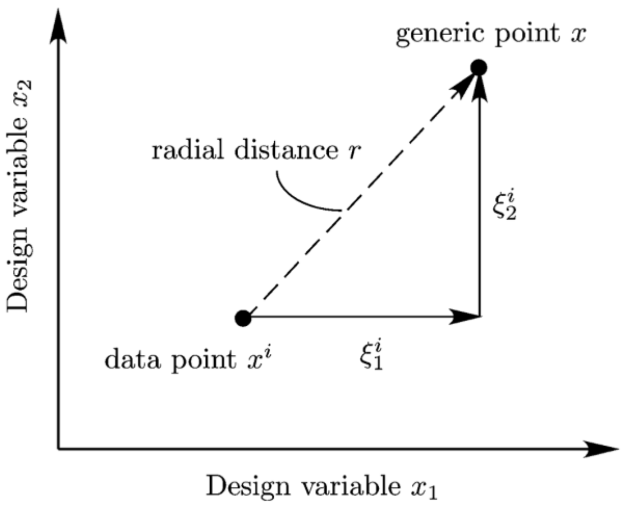
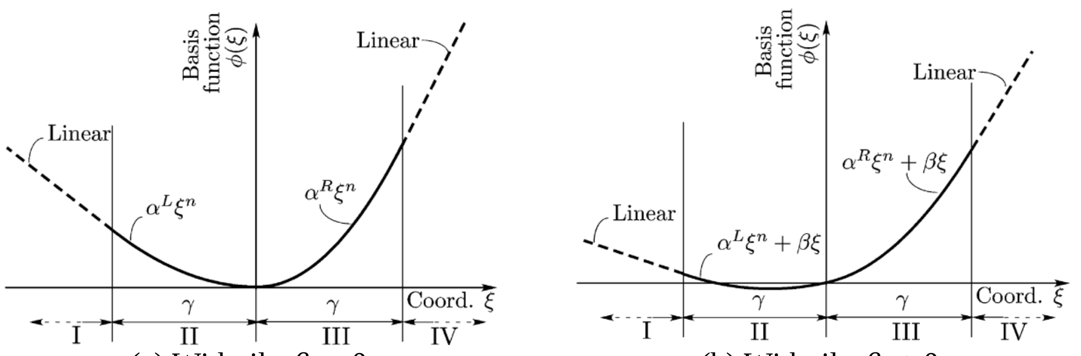


Figure 9. Definition of coordinate $\boldsymbol{\xi}$



(a) With tilt: $\beta = 0$ (b) With tilt: $\beta \neq 0$

Figure 10. Development of nonradial basis functions

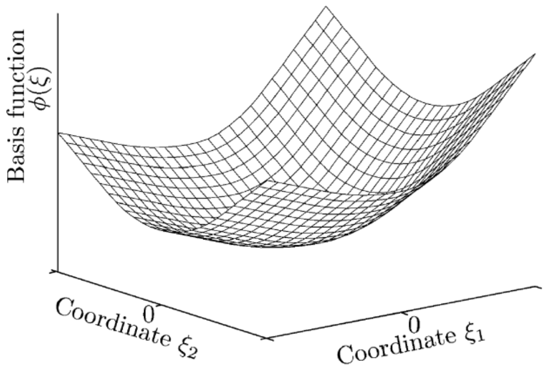


Figure 11. Nonradial basis function in two dimensions

The NRBF takes distinct forms in different regions. It is an n-th-order monomial $\alpha\xi^n$ (n ≥ 2 is an even integer) supplemented by a linear component or tilt ($\beta\xi$) in the central regions between $\xi = -\gamma$ and $\xi = +\gamma$ (regions II and III). Beyond these regions on either side (regions I and IV), the function is linear, such that the function and its first derivative are constant at $\xi = \pm\gamma$. A small value of γ would cause a smaller curved portion surrounding $\xi = 0$. So, γ can be considered as a smoothness parameter. Fig. (10a) shows the case where the coefficient $\beta = 0$ (no tilt), while Fig. 4b depicts the case where $\beta \neq 0$. Particularly, the parameters utilized to plot Fig. (10) are as follows: $n = 2$, $\gamma = 2.5$, $\alpha^L = 2$, $\alpha^R = 5$, and $\beta = 4$. For most real-life problems, values of $n = 2$ or 4 are suggested. There is no obvious benefit in terms of performance gain for greater values of n. Appropriate values for γ will be influenced by the magnitudes of the design variables. Nevertheless, normalization of the design space, say between 0 and 1, might prevent the necessity for the designer to choose a value for γ. In general, one would select a high value of γ for better smoothing properties of the metamodels. The thorough NRBF for the i-th data point is set as the sum of the individual basis functions in each dimension as:

$$\phi_i(\xi^i) = \sum_{j=1}^{m} \phi_{ij}(\xi_j^i) \tag{14}$$

1.3.3.2 Metamodeling using the ERBF approach

The ERBF is a metamodeling approach that utilizes a combination of radial and nonradial basis functions. This novel concept holds the appealing features of both classes of basis functions: the effectiveness of the multiquadric RBFs together with the flexibility of the NRBFs.

Combining Radial and Nonradial Basis Functions

As discussed beforehand, the ERBF approach promotes the notion of applying more than one basis function for each data point. By using these two different basis functions, the ERBF approximation function is as:

$$\tilde{f}(x) = \sum_{i=1}^{n_p} \sigma_i \psi\left(\|x - x^i\|\right) + \sum_{i=1}^{n_p} \sigma_i (x - x^i) \tag{15}$$

where $\psi(\|x - x^i\|)$ and $\phi_i(x - x^i)$ are obtained in Eqs. (11) and (14), respectively. Remark that every data point x_i is correlated with two different but complementary basis functions: a) the typical RBF represented in Eq. (11) and b) the basis function determined in Eq. (14). By taking the advantage of the definition of NRBFs provided in Eqs. (13) and (14) the following equation can be obtained:

$$\tilde{f}(x) = \sum_{i=1}^{n_p} \sigma_i \psi\left(\|x - x^i\|\right) + \sum_{i=1}^{n_p} \sum_{j=1}^{m} \left[\alpha_{ij}^L \phi^L(\xi_j^i) + \alpha_{ij}^R \phi^R(\xi_j^i) + \beta_{ij} \phi^\beta(\xi_j^i)\right] \tag{16}$$

where ϕ^L, ϕ^R, and ϕ^β are represented in Table 1. The following coefficient vectors and their sizes are defined as:

$$\alpha^L = \left\{\alpha_{11}^L \ \alpha_{12}^L \ \dots \ \alpha_{1m}^L \ \dots \ \alpha_{(n_p)(m)}^L\right\}_{(mn_p)\times(1)}^T \tag{17}$$

$$\alpha^R = \left\{\alpha_{11}^R \ \alpha_{12}^R \ \dots \ \alpha_{1m}^R \ \dots \ \alpha_{(n_p)(m)}^R\right\}_{(mn_p)\times(1)}^T \tag{18}$$

$$\beta = \left\{\beta_{11} \ \beta_{12} \ \dots \ \beta_{1m} \ \dots \ \beta_{(n_p)(m)}\right\}_{(mn_p)\times(1)}^T \tag{19}$$

$$\{\sigma\} = \left[\sigma_1 \ \ \sigma_2 \dots \sigma_{n_p}\right]^T \tag{20}$$

In addition to these vectors, the coefficient vector σ is obtained in Eq. (20). The vectors α^L, α^R, and β, just determined, contain mn_p elements each, and the vector σ includes n_p coefficients. So, the total number of coefficients to be calculated so as to fully specify the metamodel in Eq. (16) is given by $n_u = (3m + 1)n_p$.

The ERBF approach results in an underdetermined system of equations whose resulting freedom will be judiciously employed. Types of linear systems of equations for various metamodeling techniques are represented in Table 2.

Table 2: Type of linear system of equations for various metamodeling techniques

Method	Number of equations	Number of unknowns	Type of system	Interpolative solutions	Solution approach
RBF	n_p	n_p	Square	Unique	Matrix inverse
ERBF	n_p	$(3m+1)n_p$	Underdetermined	Family	Linear programming

1.3.3.3 Features

There are numerous significant advantages in employing extension, some of which are in the following: (1) it acts as a global nonlinear model to smoothly link together the various local linear models; (2) it extends the RBFs capability for extrapolating and generalizing more meaningfully; (3) it works as a unifying model that brings together the different approximators including splines and CMAC neural network models; and (4) this ERBF extension makes feasible the applications of statistical modeling and experiment design techniques to the investigation of general neural network approximation models.

1.3.4 Generalized regression neural network

GRNN is a type of supervised network which has been introduced by Specht [48]. GRNN is capable of producing constant value outputs. GRNNs are three-layer (input, hidden, and output layer) networks in which there is one hidden neuron for each training pattern in the hidden layer, Fig. (12). The generalized regression neural networks (GRNN) are memory-based networks that produce estimations of continuous variables and converge to the underlying regression surface. GRNNs are based on the estimate of probability density functions, fast training time, and being able to model non-linear functions. The GRNN is a one-pass learning algorithm with a massive parallel structure. It is that, even with sparse data in a multidimensional computation space, the algorithm implements smooth transitions from one perceived value to another. The algorithm can be employed for any regression problem in which an assumption of linearity is not considered. GRNNs are evaluated as a normalized Radial Basis Functions network in which there is a hidden unit centered at every training case. These Radial Basis Function units are density functions such as the Gaussian. The only weights that require to be determined are the widths of the RBF units. These widths are expressed as smoothing parameters (r). The main shortcoming of GRNN is not being able to ignore irrelevant inputs without considerable modifications to the basic algorithm. Consequently, GRNN is not likely to be the preferred choice when there are more than 5 or 6 number redundant inputs. The regression of a dependent variable, Y, on an independent variable, X, is the calculation of the most probable value of Y for each value of X based on a finite number of possibly noisy measurements of X and the associated amounts of Y. To implement system identification, it is habitually essential to assume some functional forms. In the case of linear regression, for example, the output Y is considered to be a linear function of the input, and the unknown parameters, a_i, are linear coefficients. The process does not need to assume a particular functional form. The Euclidean distance (d_i^2) is estimated between an input vector and the weights, which are then rescaled by the smoothing factor. The radial basis output is then the exponential of the negatively weighted distance.

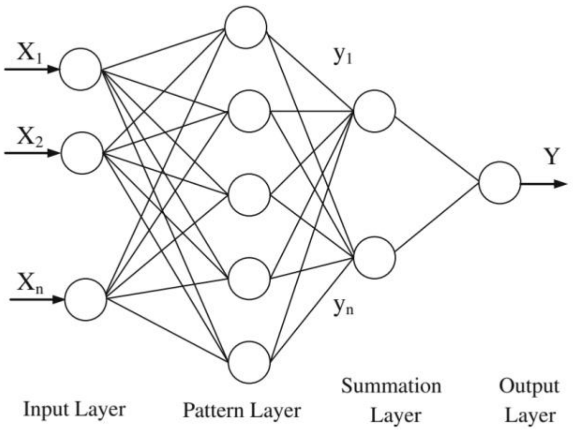


Figure 12. Schematic diagram of generalized regression neural network

The GRNN equations are as follows:

$$d_i^2 = (X - X^i)^T (X - X^i) \tag{21}$$

$$Y(X) = \frac{\sum_{i=1}^{n} Y_i \, exp\left(-\frac{d_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{n} exp\left(-\frac{d_i^2}{2\sigma^2}\right)} \tag{22}$$

The approximation function $Y(X)$ can be conceived as a weighted average of all observed amounts, Y_i, where each of which is weighted exponentially according to its Euclidian distance relative to X. And $Y(X)$ is the sum of the Gaussian distributions gathered at each training sample. However, the sum is not restrained to be the Gaussian. According to this theory, r expresses the smoothing factor, and the optimum smoothing factor can be calculated, after satisfying runs based on the mean squared error (MSE), Eq. (23), of the computed amounts, which must be kept at a minimum. This process refers to the training of the network. If several iterations pass without an increase in the mean squared error, the smoothing factor is specified as the optimum one for that data set. In the production step, the smoothing factor is applied to data sets that the network has not observed before. While employing the network to a new set of data, a raise in the smoothing factor would cause a reduction in the range of output values [48, 49]. In the GRNNs, there are no training parameters such as the learning rate, momentum, optimum number of neurons in the hidden layer, and learning algorithms as required to be determined in backpropagation neural networks. Besides, the GRNN has a high pace in approximation in comparison with the BPNNs. In the GRNNs structure, there is a smoothing factor that its optimum amount is determined by a try and error procedure. The smoothing factor needs to be greater than 0 and ranges from 0.1 to 1 with moderate upshots. The number of neurons in the input layer is as same as the number of inputs in the problem, the number of neurons in the output layer matches the number of outputs, and the number of hidden layer neurons is training patterns. The GRNN networks may be more accurate than the BPNNs when there are multiple outputs because the GRNN networks compute each output separately from the other outputs. The GRNN networks operate by distinguishing the discrepancy between the given sample patterns and patterns in the training set.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left[(Y_{GRNN})_i - \left(Y_{exp}\right)_i\right]^2 \tag{23}$$

1.3.4.1 Generalized regression neural networks VS. backpropagation neural networks

There are many distinctions between GRNN and BPNN. Initially, GRNNs are single-pass learning algorithm, whereas BPNNs requires two moves: forward and backward pass, which means GRNNs are significantly less time-consuming regarding training process. Also, the only free parameter in GRNNs is the smoothing parameter σ, but in BPNNs, more parameters are needed such as weights, biases, and learning rates. Although BPNN has a limited predefined size, since GRNN is an auto-associative memory network, it will store all the different input/output samples. Finally, GRNN is based on the general regression theory, whereas BPNNs are according to the gradient-descent iterative optimization method. The most remarkable advantage of GRNN over BPNN is the less training time, which proves its preference for dynamic systems modeling and control. Additionally, GRNN has less testing error, which means it has better generalization capacities than BPNN.

2. STRUCTURAL MODEL AND CONFIGURATION PROCESSING

In this paper, the considered model is a double-layer barrel vault, Fig. (13), which is generated utilizing FORMIAN, connected by MERO type of joints with the lengths of 8 m in x-direction divided into 8 spans and 16 m in y-direction divided into 12 spans and consisting of 768 bar elements whose diameters have been chosen haphazardly from 5 cm to 10 cm. Their corresponding section areas are the input data of neural networks. The height of the model is 4 m. Due to the symmetry of the model, one-fourth of the structure is considered in the formation of the neural nets so the elements have been classified into 202 groups of four and two. In other words, 202 sections have been made and the amount of their diameters have been selected randomly. The structure is supported along y-direction in the first and last rows. The sum of dead load and live load equal to 10 kN is applied as the concentrated load to each node of the top layer. All the nodes are considered to be ball-jointed. Then the analyses have been carried out 120 times utilizing section areas as input data and the maximum obtained element stresses are used as output data for neural networks. As illustrated, 120 pairs of input and output data consisting of section areas and element stresses are generated and 80 pairs of which are for training and 40 ones are for validation of the neural networks.

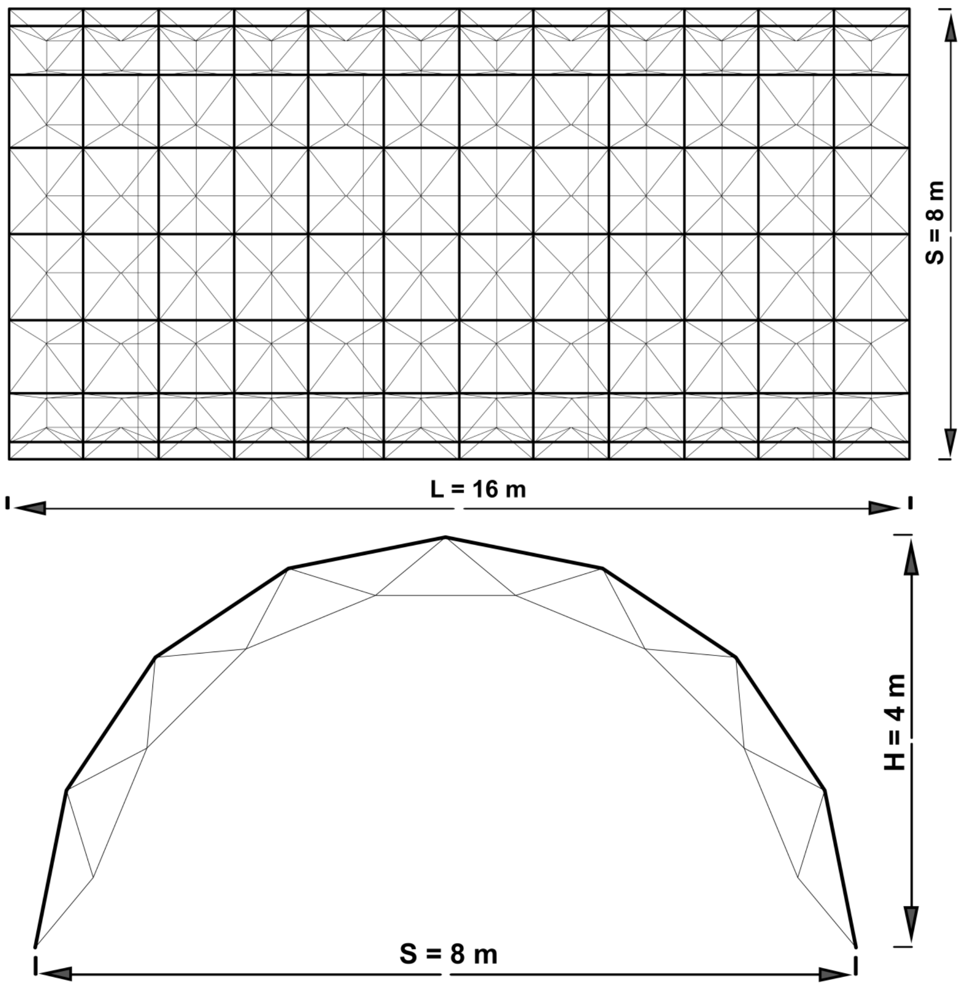


Figure 13 Plan and the elevation of the model

3. TRAINING AND TESTING THE NETWORKS

The numerical experiment has been carried out with two main aims. The former is the comparative study of the responses of the nets when large-scale structures are studied. The latter is investigating the features of each net. As the number of input-output units increases, there must be a rise in the number of training pairs leading to a reduction in the associated errors in the net response. In this example, the input units are the cross-sectional areas of the double-layer barrel vault, and the output units are their corresponding element stresses.

To train the networks, 120 pairs as input-output data, cross-sectional area, and element stresses are utilizes. The diameters of cross-sectional areas are generated haphazardly between 5 cm and 10 cm. One-tenth of the training pairs, as extra pairs, are created to verify the accuracy of the nets. To make a reasonable comparison among the nets, after the training process is accomplished, the same test data are given to the networks to show their efficiency.

3.1 Feedforward backpropagation neural network

The nets have been trained employing 120 pairs. The regression concerning training, test, validation, and all data is depicted in Fig. (14). The performance index is MSE. The performance of the FFBP employing 120 training pairs is shown in Fig. (15), which became the best at epoch 5.

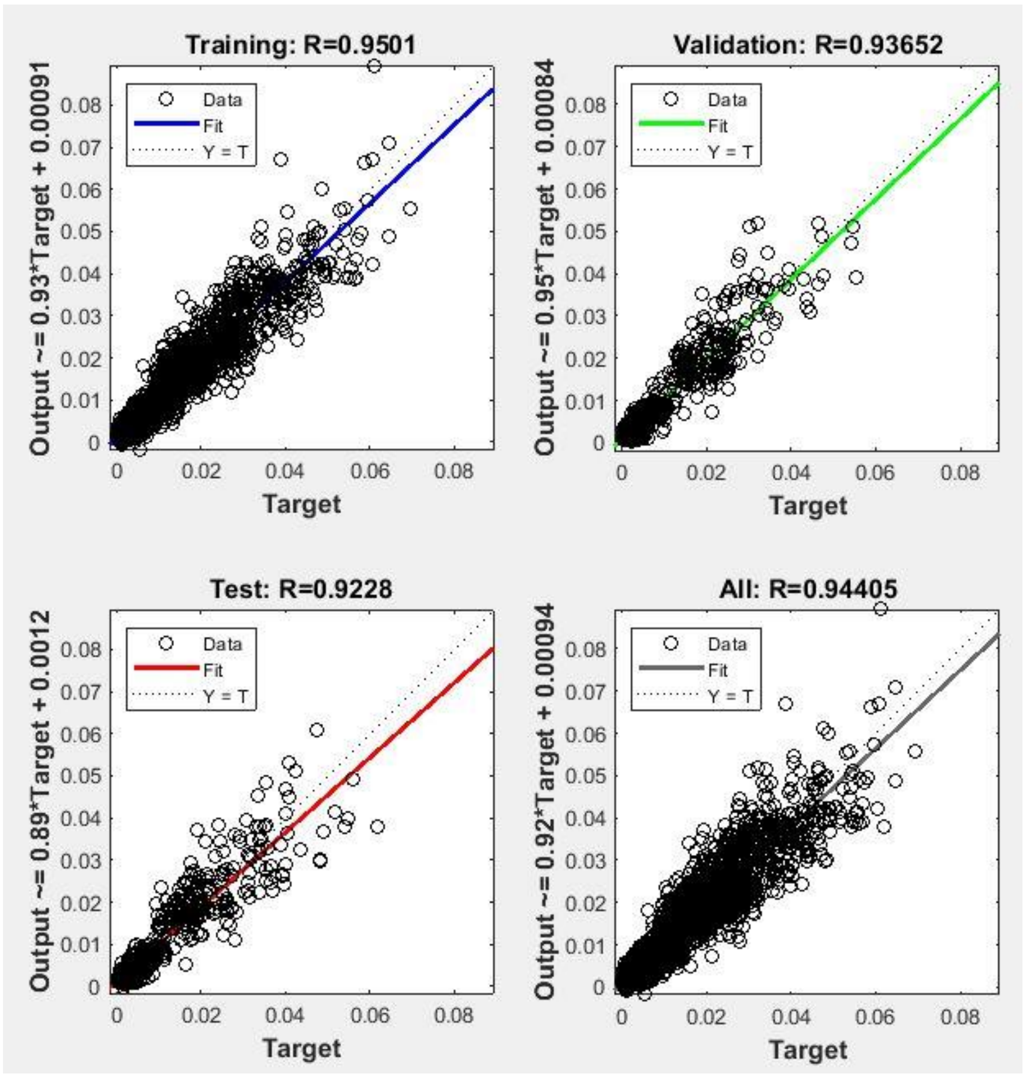


Figure 14. Regression of data in FFBP

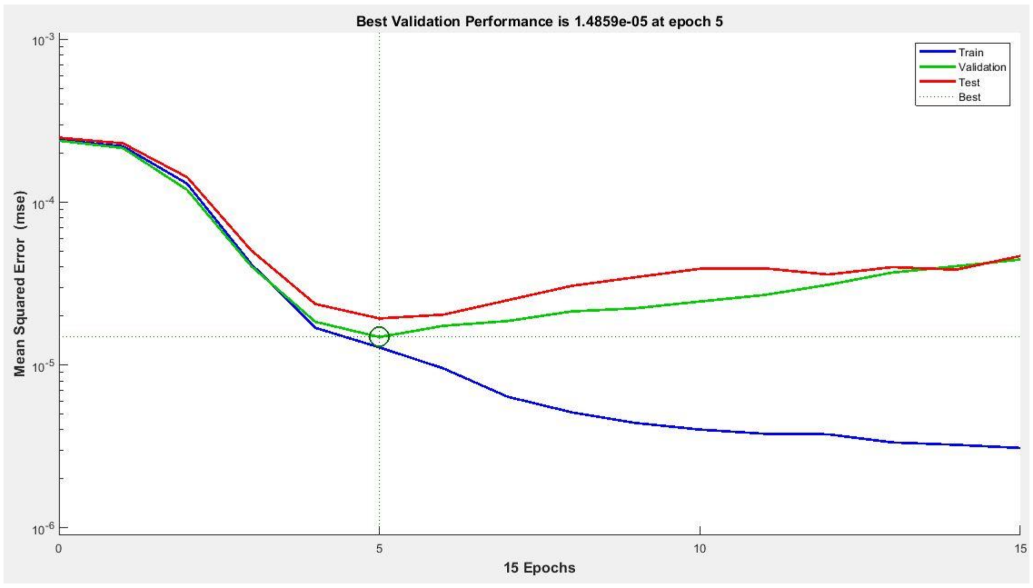


Figure 15. Performance of FFBP net

3.2 Radial basis function

The RBF networks have been also trained utilizing 120 training pairs. According to the training and testing the data, the regression of training, test, and all data is shown in Fig. (16). The performance is scrutinized using Mean Square Error being illustrated in Fig. (17).

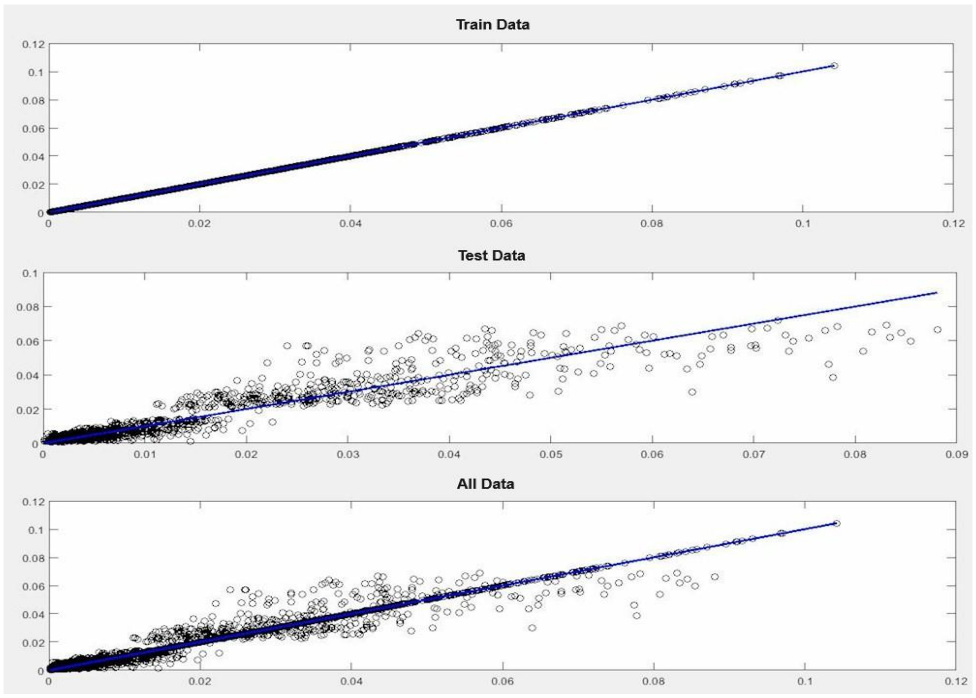


Figure 16. Regression of data in RBF

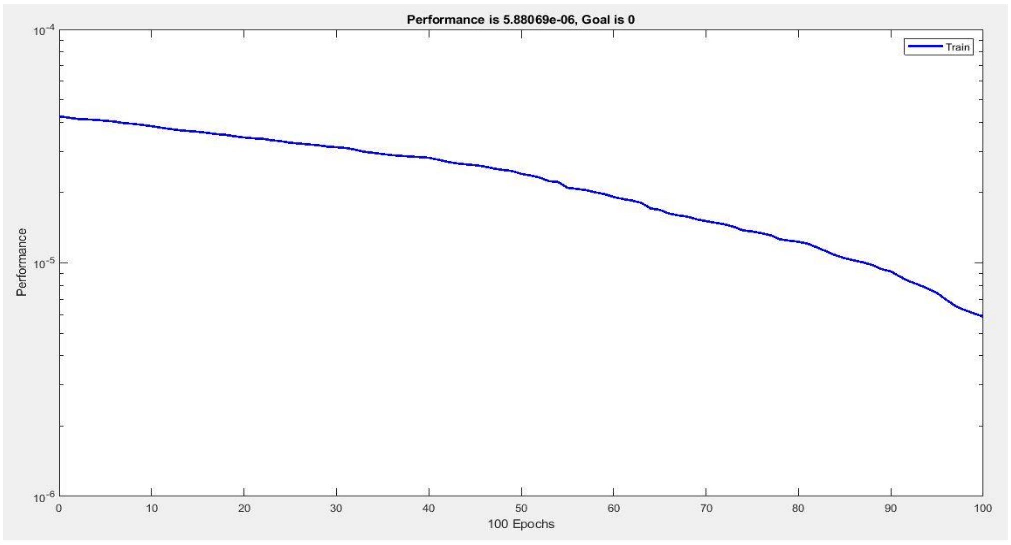


Figure 17. Performance of RBF net

3.3 Extended radial basis function

To train the ERBFs, 120 pairs have been employed so as to figure out the efficiency of the trained neural networks. After training is performed, based on the desired data and the upshot of the nets, the regression of data is obtained, Fig. (18), for training, test, and all data. Also, the MSE of the trained RBF, as its performance, is outlined in Fig. (19).

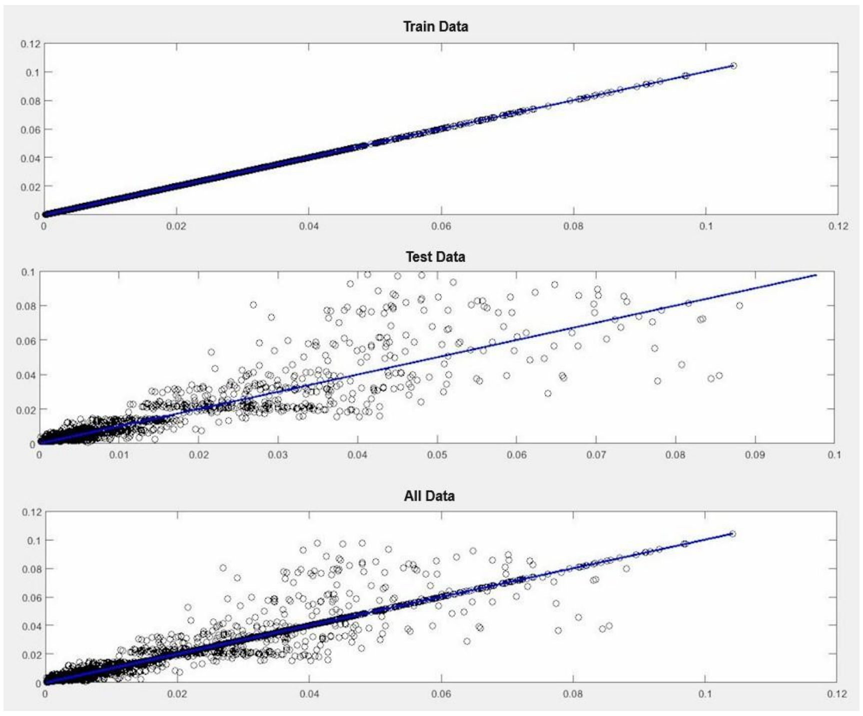


Figure 18. Regression of data in ERBF

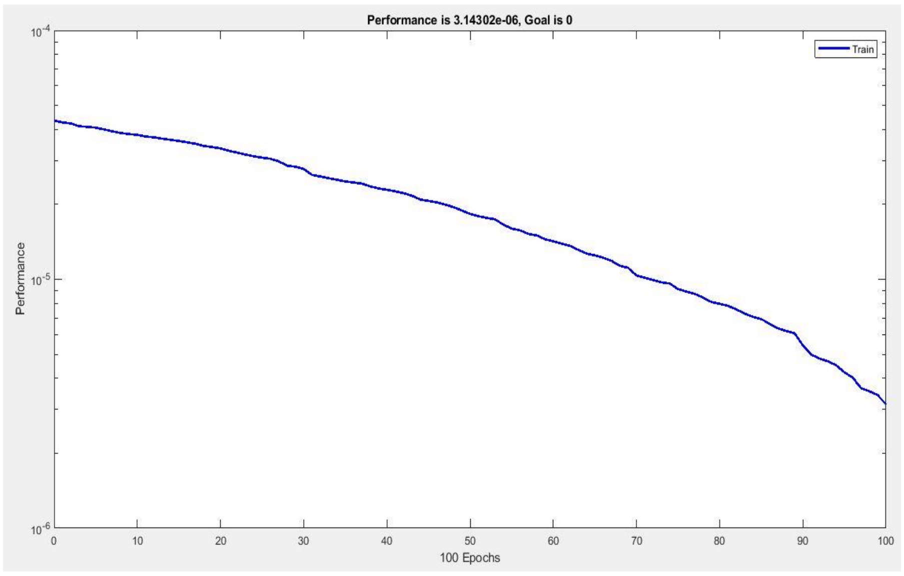


Figure 19. Performance of ERBF net

3.4 Generalized regression neural networks

The training procedure of GRNNs has been achieved the same as for the other neural networks, i.e., 120 training pairs have been applied. Also, for training, test, and all data regression diagrams are drawn in Fig. (20).

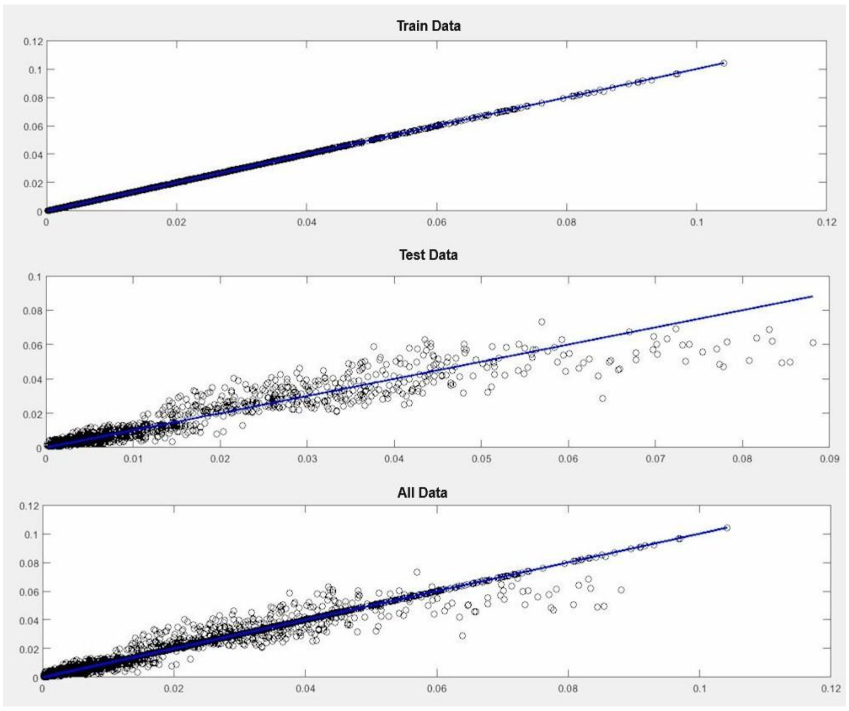


Figure 20. Regression of data in GRNN

3.5 Numerical comparison

To perceive how efficiently the nets have been trained and make a comparison among them, an untrained vector comprising 32 cross-sectional areas representing the whole model is given to each trained net. Then, their output, which is the maximum corresponding element stress, is recorded. Furthermore, the exact amount of maximum element stresses are obtained employing the numerical solution. Finally, the results of element stresses using all the networks and the numerical solution are written in Table. 3.

Table 3: The response NNs to a given untrained input vector

Section		**Element Stress** $\left(\frac{N}{cm^2}\right)$				
		Numerical	**Neural Network (120 Pairs)**			
Number	Area (cm^2)	Exact Amount	FFBP	RBF	ERBF	GRNN
1	46.74907	-546.0807	-560.1753	-560.4302	-540.84146	-532.12377
2	24.52289	-163.1838	-159.6127	-160.9418	-166.12039	-166.81372
3	43.04328	-395.3864	-387.5324	-405.0569	-390.73675	-402.71861
4	77.05776	-759.0589	-752.6798	-767.9333	-771.7176	-772.43881
5	44.93941	-436.0524	-447.6335	-426.9525	-439.61121	-420.5362
6	19.77579	299.5042	302.8521	296.1289	302.00252	288.48064
7	68.06063	176.9286	179.7267	182.7280	173.41278	181.17235
8	33.71613	266.3167	272.1016	261.0659	262.92627	276.39468
9	44.03739	-890.1339	-913.0991	-862.2477	-878.24619	-865.28523
10	67.11493	-90.0986	-91.95266	-88.3105	-88.820506	-87.593646
11	36.98962	-253.9258	-249.0007	-258.7874	-255.85594	-259.05353
12	75.34654	-280.4369	-272.7679	-269.7914	-285.96273	-272.34674
13	42.88528	-515.0185	-506.2473	-498.3544	-524.61768	-530.95858
14	35.07259	156.1594	160.0670	161.3561	152.62773	159.99420
15	60.82953	159.6092	156.1714	156.4226	160.59932	155.94600
16	40.29653	-921.8512	-937.6833	-908.3292	-932.99541	-896.00359
17	34.87867	-1069.3252	-1043.169	-1101.047	-1052.8461	-1035.104
18	23.42341	250.4017	256.3137	246.2489	247.34338	259.55390
19	30.66882	172.4522	169.6198	169.0479	174.88471	177.02557
20	52.86815	-222.0101	-216.6454	-230.8457	-219.33401	-229.5519
21	26.88380	220.3437	222.9577	225.1768	217.23190	211.75520
22	51.08055	170.5413	171.9706	165.6984	172.34549	174.40178
23	23.71080	167.7112	170.0481	169.8109	170.15136	173.48308
24	59.92057	-150.2564	-146.8854	-156.2614	-151.0266	-145.77504
25	64.00378	319.0812	313.6087	327.4176	312.49966	326.79678
26	25.99343	182.0039	187.0325	177.0407	186.16890	176.22752

27	35.26284	-216.6691	-218.8490	-210.9983	-221.61046	-211.6718
28	49.66660	362.9856	359.1206	369.1183	354.37263	351.76148
29	36.78939	-356.8501	-361.1355	-346.0283	-362.88385	-348.18525
30	27.09467	-670.3521	-654.0866	-659.3195	-681.09955	-686.31111
31	50.94889	-291.0786	-296.0048	-283.5514	-285.93122	-299.76055
32	71.63778	-115.6890	-119.1024	-111.1855	-114.56339	-120.2645

4. CONCLUDING REMARKS AND DISCUSSION

To conclude, not only can FFBP, RBF, ERBF, and GRNN neural networks be efficiently put into practice for the analysis of double-layer barrel vaults but also, they can be employed for the analysis, design, and optimization of other large-scale space structures such as grids and domes. The application of neural networks to problems in structural mechanics provides a near-optimal solution with a considerable reduction in computational time. Moreover, the following conclusions are derived:

- The performance of all four neural nets in the analysis of the double-layer barrel vault is satisfactory.
- The networks can efficiently be utilized for the analysis with at most 4% error.
- For a fixed number of training cycles, RBF works better than FFBP and results in better accuracy.
- Errors in FFBP are in general less than RBF for a regular number of training cycles.
- The use of Tangh for RBF and Sigmoid for FFBP causes a higher convergence rate and lower errors.
- The most significant advantage of GRNN is less training time.
- Employing extension in RBF (ERBF) expands the RBFs ability for extrapolating and generalizing leading to less error in the approximation.
- The FFBP has very accurate responses due to its prominent approximation ability.
- The best performance and the maximum accuracy occurred in ERBF.

5. ACKNOWLEDGMENT

The author would like to express great appreciation to P. Jelokhani for his valuable help and support during this research work.

REFERENCES

1. Makowski ZS. *Analysis, Design and Construction of Braced Barrel Vaults*, CRC Press, 1986.

2. Ramaswamy GS, Eekhout M. *Analysis, Design and Construction of Steel Space Frames*, Thomas Telford, 2002.
3. Kaveh A. *Applications of Metaheuristic Optimization Algorithms in Civil Engineering,* Springer, Cham, 2017.
4. Kaveh A, Farahani M, Shojaei N. Optimal design of barrel vaults using charged search system, *Int J Civil Eng* 2012; **10**: 301-8.
5. Kaveh A, Mirzaei B, Jafarvand A. Optimal design of double layer barrel vaults using improved magnetic charged system search, *Asian J Civil Eng* 2014; **15**: 135-54.
6. Kaveh A, Mahjoubi S. Optimum design of double-layer barrel vaults by lion pride optimization algorithm and a comparative study, *Struct* 2018; **13**: 213-29.
7. Kaveh A, Mirzaei B, Jafarvand A. Shape-size optimization of single-layer barrel vaults using improved magnetic charged system search, *Int J Civil Eng* 2014; **12**: 447-65.
8. Kaveh A, Eftekhar B. Optimal design of double layer barrel vaults using an improved hybrid big bang-big crunch method, *Asian J Civil Eng* 2012; **13**: 465-87.
9. Nooshin H. Space structures and configuration processing, *Prog Struct Eng Mat* 1998; **1**: 329-36.
10. Kaveh A. *Structural Mechanics: Graph and Matrix Methods*, Macmillan International Higher Education, 1992.
11. Nooshin H, Disney P. Formex configuration processing II, *Int J Space Struct* 2001; **16**: 1-56.
12. Fausett LV. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, Pearson Education India, 2006.
13. Patterson DW. *Artificial Neural Networks: Theory and Applications*. Prentice Hall PTR, 1998.
14. Berke L, Hajela P. *Applications of Artificial Neural Nets in Structural Mechanics*. In *Shape and Layout Optimization of Structural Systems and Optimality Criteria Methods*, Springer, Vienna, 1992, pp. 331-348.
15. Kaveh A, Khaleghi A. Prediction of strength for concrete specimiens using artificiai neural networks, *Asian J Civil Eng* 2000; **1**: 1-12.
16. Kaveh A, Iranmanesh A. Comparative study of backpropagation and improved counterpropagation neural nets in structural analysis and optimization, *Int J space struct* 1998; **13**: 177-85.
17. Kaveh A, Servati H. Design of double layer grids using backpropagation neural networks, *Comput Struct* 2001; **79**: 1561-8.
18. Hajela P, Berke L. Neural networks in structural analysis and design: an overview, *Comput Syst Eng* 1992; **3**: 525-38.
19. Iranmanesh A, Kaveh A. Structural optimization by gradient-based neural networks, *Int J Numer Meth Eng* 1999; **46**: 297-311.
20. Kaveh A, Elmieh R, Servati H. Prediction of moment-rotation characteristic for semi-rigid connections using BP neural networks, *Asian J Civil Eng* 2001; **2**: 131-42.
21. Kaveh A, Dehkordi M R. Neural networks for the analysis and design of domes, *Int J Space Struct* 2003; **18**: 181-93.
22. Hajela P, Berke L. Neurobiological computational models in structural analysis and design, *Comput Struct* 1991; **41**: 657-67.

23. Kaveh A, Servati H, Fazel D D. Prediction of moment-rotation characteristic for saddle-like connections using FEM and BP neural networks, *Asian J Civil Eng* 2001; **2**: 11-29.
24. Kaveh A, Gholipour Y, Rahami H. Optimal design of transmission towers using genetic algorithm and neural networks, *Int J Space Struct* 2008; **23**: 1-19.
25. Adeli H, Seon Park H. Counterpropagation neural networks in structural engineering, *J Struct Eng* 1995; **121**: 1205-12.
26. Rofooei F R, Kaveh A, Farahani F M. Estimating the vulnerability of the concrete moment resisting frame structures using artificial neural networks, *Int J Optim Civil Eng* 2011; **1**: 433-48.
27. Parker DB. *Learning Logic*, Center for Computational Research in Economics and Management Science, MIT-Press, Cambridge, 1985.
28. Werbos PJ. Generalization of backpropagation with application to a recurrent gas market model, *Neural Netw* 1988; **1**: 339-56.
29. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors, *Nature* 1986; **323**: 533-6.
30. Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks, *Chemome Intell Lab Syst* 1997; **39**: 43-62.
31. Faris H, Aljarah I, Mirjalili S. Training feedforward neural networks using multi-verse optimizer for binary classification problems, *Appl Intell* 2016; **45**: 322-32.
32. Hertz J, Krogh A, Palmer RG, Horner H. Introduction to the theory of neural computation, *Phys Today* 1991; **44**: 70.
33. Bertsekas DP, Tsitsiklis JN. *Neuro-Dynamic Programming*, Athena Scientific, 1996.
34. Wieland A, Leighton R. Geometric analysis of neural network capabilities, *1st IEEE Inernational Conference on Neural Networks* 1987: pp. 385.
35. Hardy RL. Multiquadric equations of topography and other irregular surfaces, *J Geophys Res* 1971; **76**: 1905-15.
36. Broomhead DS, Lowe D. *Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks*. Royal Signals and Radar Establishment Malvern, United Kingdom, 1988.
37. Hartman EJ, Keeler JD, Kowalski JM. Layered neural networks with Gaussian hidden units as universal approximations, *Neural Comput* 1990; **2**: 210-15.
38. Park J, Sandberg IW. Universal approximation using radial-basis-function networks, *Neural Comput* 1991; **3**: 246-57.
39. Sing JK, Basu DK, Nasipuri M, Kundu M. Improved k-means algorithm in the design of RBF neural networks, *IEEE Conference on Convergent Technologies for Asia-Pacific Region* 2003: pp. 841-845.
40. Vogt M. Combination of radial basis function neural networks with optimized learning vector quantization, *IEEE International Conference on Neural Networks* 1993: pp. 1841-6.
41. Kubat M. Decision trees can initialize radial-basis function networks, *IEEE Trans Neural Netw* 1998; **9**: 813-21.
42. Lin CL, Wang JF, Chen CY, Chen CW, Yen CW. Improving the generalization performance of RBF neural networks using a linear regression technique, *Expert Syst Appl* 2009; **36**: 12049-53.

43. Chen S, Wu Y, Luk BL. Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks, *IEEE Trans Neural Netw* 1999; **10**: 1239-43.
44. Neruda R, Kudovà P. Learning methods for radial basis function networks, *Future Gener Comput Syst* 2005; **21**: 1131-42.
45. Powell M J, *Algorithms for approximation*, Chap. 3, Oxford University Press, New York, 1987.
46. Girosi F. An equivalence between sparse approximation and support vector machines, *Neural Comput* 1998; **10**: 1455-80.
47. Mullur AA, Messac A. Extended radial basis functions: more flexible and effective metamodeling, *AIAA J* 2005; **43**: 1306-15.
48. Specht DF. A general regression neural network, *IEEE Trans Neural Netw* 1991; **2**: 568-76.
49. Beale MH, Hagan MT, Demuth HB. Neural network toolbox, *User's Guide, Math Works* 2010; **2**: 77-81.