THE JOURNAL OF NAVIGATION (2002), **55**, 97–108. © The Royal Institute of Navigation
DOI: 10.1017/S0373463301001606 Printed in the United Kingdom

Applying Back-propagation Neural Networks to GDOP Approximation

Dah-Jing Jwo and Kuo-Pin Chin

(*Institute of Maritime Technology*, *National Taiwan Ocean University*)

In this paper, back-propagation (BP) neural networks (NN) are applied to the GPS satellite Geometric Dilution of Precision (GDOP) approximation. The methods using BPNN are general enough to be applicable regardless of the number of satellite signals being processed by the receiver. BPNN is employed to learn the functional relationships firstly, between the entries of a measurement matrix and the eigenvalues and thus generate GDOP, and secondly, between the entries of a measurement matrix and the GDOP, both without inverting a matrix. Consequently, two sets of entries and two sets of output variables, respectively, are used that in total yield four types of mapping architectures. Simulation results from these four architectures are presented. The performance and computational benefit of neural network-based GDOP approximation are explored.

KEY WORDS

1. GPS. 2. Data. 3. GDOP.

1. INTRODUCTION. The navigation accuracy of GPS is normally determined by two causes: the errors in each signal observable, and the geometry formed by the observables employed for positioning or navigation. The GPS measurements are normally corrupted by several error sources, such as ionospheric delay, tropospheric delay, satellite clock and receiver clock offsets, receiver noise and multi-path. The Geometric Dilution of Precision, usually referred to as the GDOP, is a geometrically determined factor that describes the effect of geometry on the relationship between measurement error and position determination error. It is used to provide an indication of the quality of the solution. Some receiver hardware may be restricted to processing a limited number of visible satellites. Therefore, it is sometimes necessary to select the satellite subset that offers the best or most acceptable solution. The optimal satellite subset is usually chosen by minimizing the GDOP.

Neural networks are trainable, dynamic systems that can estimate input-output functions. They have been applied to a wide variety of problems because they are model-free estimators, i.e. without a mathematical model. The back-propagation neural network (BPNN) has been the most popular learning algorithm throughout all neural applications. BPNN is a neural system with a back-propagation algorithm that can learn input-output functions from a series of samples. It is a gradient-based algorithm, in the sense that the weight update is performed along the direction of the gradient of an appropriate error function. The BPNN is simple and requires a minimal amount of storage.

GDOP calculation using neural networks was first proposed by Simon and El-Sherief (1995), who employed the BP and optimal interpolative (OI) net for implementing the function approximation and classification, respectively. In their research, the BPNN was used to learn the functional relationships between the entries of a measurement matrix and the eigenvalues of its inverse, and thus generate GDOP. The methods using BP are general enough to be applicable regardless of the number of satellite signals being processed by the receiver. In this paper, the architecture of a neural network for GDOP function approximation will be re-examined and improved. In addition to the architecture implemented by Simon and El-Sherief (1995), three other input-output relationships will be employed. Simulation results will be given; the computational benefit for different types of mapping will be compared and discussed.

2. GPS MEASUREMENTS AND GDOP. The GPS measurements, errors, and the GDOP are briefly reviewed in this section. Consider the vectors depicted in Figure 1 relating the Earth's centre, the satellites and the user's position. The vector

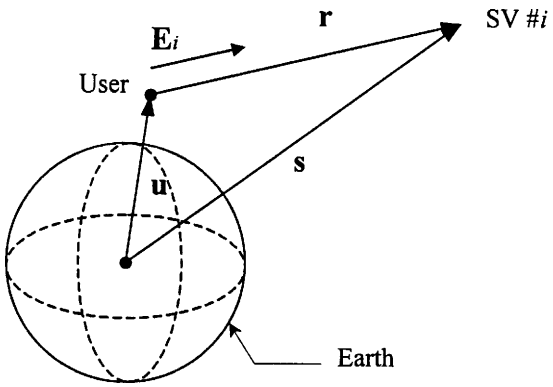

Figure 1. Definition of vectors.

s represents the vector from the Earth's centre to a satellite, **u** represents the vector from the Earth's centre to the user's position, and **r** represents the vector from the user to satellite, we can write the vector relation:

$$\mathbf{r} = \mathbf{s} - \mathbf{u}. \tag{1}$$

The distance $\|\mathbf{r}\|$ is computed by measuring the propagation time from the transmitting satellite to the user/receiver. The GPS pseudorange ρ_i is defined for the i-th satellite by:

$$\rho_i = \|\mathbf{s}_i - \mathbf{u}\| + ct_b + v_{\rho_i} = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + ct_b + v_{\rho_i}, \tag{2}$$

where: (x_u, y_u, z_u) and (x_i, y_i, z_i) denote the user and ith satellite's positions in three dimensions, respectively; c is the speed of light; t_b is the receiver clock offset from system time; and v_{ρ_i} is the pseudorange measurement noise.

Equation (2) is linearised by expanding Taylor's series around the approximate (or nominal) user position $(\hat{x}_n, \hat{y}_n, \hat{z}_n)$ and neglecting the higher-order terms. Defining $\hat{\rho}_i$ as ρ_i at $(\hat{x}_n, \hat{y}_n, \hat{z}_n)$ gives:

$$\Delta\rho_i = \rho_i - \hat{\rho}_i = e_{i1}\Delta x_u + e_{i2}\Delta y_u + e_{i3}\Delta z_u + ct_b + v_{\rho_i}, \tag{3}$$

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

where:

$$e_{i1} = \frac{\hat{x}_n - x_i}{\hat{r}_i}; \quad e_{i2} = \frac{\hat{y}_n - y_i}{\hat{r}_i}; \quad e_{i3} = \frac{\hat{z}_n - z_i}{\hat{r}_i}, \tag{4}$$

and $\hat{r}_i = \sqrt{(\hat{x}_n - x_i)^2 + (\hat{y}_n - y_i)^2 + (\hat{z}_n - z_i)^2}$. The vector $(e_{i1}, e_{i2}, e_{i3}) \equiv \mathbf{E}_i$, $i = 1, \ldots, n$, denotes the line-of-sight vector from the user to the satellites. Equation (3) can be written in a matrix formulation:

$$\begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \vdots \\ \Delta\rho_n \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & e_{13} & 1 \\ e_{21} & e_{22} & e_{23} & 1 \\ e_{31} & e_{32} & e_{33} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ e_{n1} & e_{n2} & e_{n3} & 1 \end{bmatrix} \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ c\Delta t_b \end{bmatrix} + \begin{bmatrix} v_{\rho_1} \\ v_{\rho_2} \\ v_{\rho_3} \\ \vdots \\ v_{\rho_n} \end{bmatrix}, \tag{5}$$

which can be represented as:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v}. \tag{6}$$

The dimension of the geometry matrix $\mathbf{H}$ is $n \times 4$ with $n \geqslant 4$.

The least squares solution to Equation (6) is given by

$$\hat{\mathbf{x}} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{z}, \tag{7}$$

and the quality of navigation solution for a linearised pseudorange equation is represented by taking the difference between the estimated and true positions:

$$\tilde{\mathbf{x}} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{v}, \tag{8}$$

where: $\mathbf{v}$ has zero mean, and so does $\tilde{\mathbf{x}}$. The covariance between the errors in the components of the estimated position is:

$$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\mathrm{T}}\} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}E\{\mathbf{v}\mathbf{v}^{\mathrm{T}}\}\mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1},$$

where: $E\{\cdot\}$ is the expected value operator. If all components of $\mathbf{v}$ are pairwise uncorrelated and have variance σ^2, then $E\{\mathbf{v}\mathbf{v}^{\mathrm{T}}\} = \sigma^2\mathbf{I}$, and consequently:

$$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\mathrm{T}}\} = \sigma^2(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}. \tag{9}$$

The GDOP factor is defined as:

$$\mathrm{GDOP} = \sqrt{\mathrm{trace}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}} = \sqrt{\frac{\mathrm{trace}[\mathrm{adj}(\mathbf{H}^{\mathrm{T}}\mathbf{H})]}{\det(\mathbf{H}^{\mathrm{T}}\mathbf{H})}}. \tag{10}$$

It is seen from Equation (9) that the GDOP factor gives a simple interpretation of how much one unit of measurement error contributes to the derived position solution error for a given situation. It determines the magnification factor of the measurement noise that is translated into the derived solution.

Existing methods for GDOP calculation generally include:

(a) Matrix inversion by computer;
(b) Closed-form algorithm; and
(c) Maximum volume of a tetrahedron.

The most straightforward approach for obtaining GDOP is to use matrix inversion or the closed-form solution to all combinations and select the minimum one.

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

However, the matrix inversion and closed-form algorithm by computer present a significant computational burden to the navigation computer especially when the number of satellites is large. Four satellites will generally be required to provide sufficient information for an acceptable three-dimensional position fix. For the case of processing four satellite signals, it has been shown that GDOP is approximately inversely proportional to the volume of the tetrahedron formed by four satellites. Therefore, it is optimum to select satellites such that the volume is as large as possible, which is sometimes called the maximum volume method. The disadvantage of this method is that it does not guarantee an optimum selection of satellites.

3. THE BACK-PROPAGATION NEURAL NETWORKS. Artificial neural networks (ANNs), or simply neural networks (NNs), have been applied to a wide variety of problems. They have been studied for more than three decades since Rosenblatt (1962) first applied single-layer perceptrons to pattern classification learning in the late 1950s. A NN is a network structure consisting of a number of nodes connected through directional links. Each node represents a process unit, and the links between nodes specify the casual relationship between the connected nodes. The learning rule specifies how these parameters should be updated to minimize a prescribed error measure, which is a mathematical expression that measures the discrepancy between the network's actual output and a desired output.

The importance of a NN includes the way a neuron is implemented and how their interconnection/topology are made. The procedure of finding a gradient vector in a network structure is generally referred to as back-propagation (BP) since the gradient is calculated in the direction opposite to the flow of the output of each node. The BPNN has been most popular throughout all neural applications. It is a feed forward type (multi-layer perceptron) supervised learning network. A feed forward network maps a set of input vectors to a set of output vectors. The basic principle of the BP is to use the gradient steepest descent method to minimize the cost function. Standard BP can be used for batch training or sequential training. In sequential-style training, the weight updating is performed after the presentation of each training pattern, while in batch-style training, the weight updating is performed after the presentation of all training patterns. The training of neural networks is traditionally based on minimization of the cost function. Suppose a set of training samples is available, the problem can be characterized as choosing the weights (or coupling strengths) of a given network such that the following total squared error is minimized:

$$E = \frac{1}{2} \sum_{k=1}^{n} (d_k - y_k)^2, \tag{11}$$

where: n is the number of output variables.

Most units in NN transform their net input by using a scalar-to-scalar node transfer function called 'activation function'. The activation functions of the hidden layer and the output layer are typically sigmoid functions:

$$f(u) = \frac{1}{(1+e^{-u})}, \tag{12}$$

where: $u \in (-\infty, \infty)$, and $f(u) \in (0, 1)$. This function is smooth and continuous, and differentiable:

$$\dot{f}(u) = u(1-u). \tag{13}$$

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

Sigmoid hidden and output units usually use a 'bias' or 'threshold' term in computing the net input to the unit. A bias term can be treated as a connection weight from a special unit with a constant activation value. The BP topology employed in this paper is made of three layers, including one input layer, one hidden layer, and one output layer, as shown in Figure 2. More detailed discussion on NN can be found

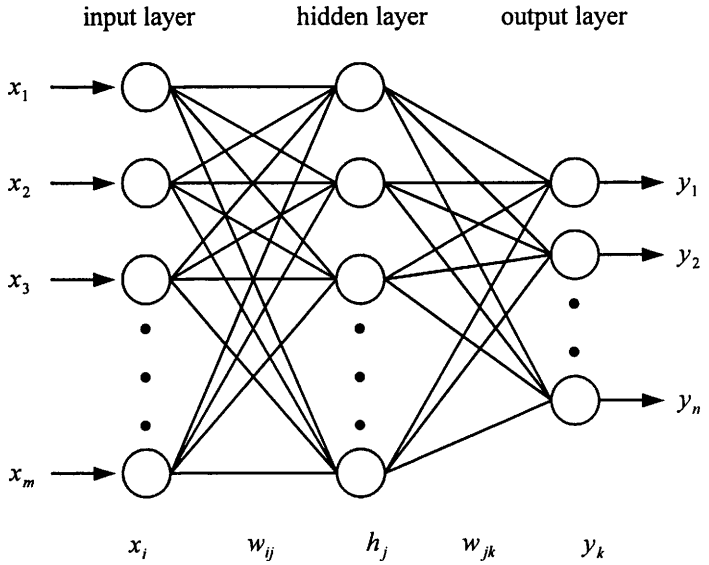

Figure 2. The topology of a three-layer back-propagation neural network.

in many literatures; such as: Widrow and Lehr (1990), Poggio and Girosi (1990), Chester (1993) and Haykin (1999). The training procedure is summarized as follows.

(a) Set η, α and initial values of w_{ij}, w_{jk}, θ_j, θ_k, where the initial values are uniformly distributed random variables.
(b) Input training samples, $\mathbf{x} = (x_1, x_2 \ldots, x_m)^{\mathrm{T}}$ and $\mathbf{d} = (d_1, d_2, \ldots, d_n)^{\mathrm{T}}$.
(c) Calculate the input v_j and output h_j for the jth hidden neuron

$$v_j = \sum_{i=1}^{m} w_{ij}x_i - \theta_j : h_j = f(v_j) = \frac{1}{1+\exp(-v_j)}.$$

(d) Calculate the input r_k and output y_k for the kth output neuron

$$r_k = \sum_{j=1}^{N_{hid}} w_{jk}x_j - \theta_k;\ y_k = f(r_k) = \frac{1}{1+\exp(-r_k)}.$$

(e) Calculate error for the kth output neuron:

$$\delta_k = (d_k - y_k)\dot{f}(r_k) = y_k(1-y_k)(d_k - y_k).$$

(f) Calculate error for the jth hidden neuron:

$$\delta_j = \dot{f}(v_j)\sum_{k=1}^{n} w_{jk}\delta_k = h_j(1-h_j)\sum_{k=1}^{n} w_{jk}\delta_k.$$

(g) Calculate Δw_{jk} and $\Delta\theta_k$: $\Delta w_{jk} = \eta\delta_k h_j + \alpha\Delta w_{jk}$, $\Delta\theta_k = -\eta\delta_k + \alpha\Delta\theta_k$
(h) Calculate Δw_{ij} and $\Delta\theta_j$: $\Delta w_{ij} = \eta\delta_j x_i + \alpha\Delta w_{ij}$, $\Delta\theta_j = -\eta\delta_j + \alpha\Delta\theta_j$
(i) Update w_{jk} and θ_k: $w_{jk} = w_{jk} + \Delta w_{jk}$, $\theta_k = \theta_k + \Delta\theta_k$

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

(j) Update w_{ij} and θ_j: $w_{ij} = w_{ij} + \Delta w_{ij}$, $\theta_j = \theta_j + \Delta\theta_j$
(k) Calculate the cost function: $E = \frac{1}{2}\sum_{k=1}^{n}(d_k - y_k)^2$
(l) Repeat steps 3–11 until the network is convergent (i.e. the cost function reaches a sufficiently small threshold).

where the notations are defined as follows:

η: learning rate,
α: momentum,
x_i: value of the ith input neuron,
h_j: output of the jth hidden neuron,
d_k: desired output of the kth output neuron,
y_k: actual output of the kth output neuron,
w_{ij}: interconnection weight between the ith input neuron and the jth hidden neuron,
w_{jk}: interconnection weight between the jth hidden neuron and the kth output neuron,
θ_j: bias/threshold values of the jth hidden neuron,
θ_k: bias/threshold values of the kth output neuron.

4. GDOP APPROXIMATION USING BPNN. To reduce the training time, all input and output variables are normalized in the range [0, 1]. In Simon and El-Sherief's paper (1995a), the Hecht-Nielson's approach for the effectiveness of BP in learning complex, multi-dimensional functions were employed. The Hecht-Nielson's approach was based on Kolmogorov's Theorem and extended to neural networks. The theorem states that any functional $\mathscr{R}^m \to \mathscr{R}^n$ mapping can be exactly represented by a three-layer NN with $(2m+1)$ middle-layer neurons, assuming that the input components are normalized to lie in the range [0, 1]:

$$fn\colon [0, 1]^m \subset \mathscr{R}^m \to \mathscr{R}^n. \tag{14}$$

Since $\mathbf{H}^T\mathbf{H}$ is a 4×4 matrix, it has four eigenvalues, λ_i $(i = 1 \ldots 4)$. It is seen that the four eigenvalues of $(\mathbf{H}^T\mathbf{H})^{-1}$ will be λ_i^{-1}. Based on the fact that the trace of a matrix is equal to the sum of its eigenvalues, Equation (10) can be represented as

$$\text{GDOP} = \sqrt{\text{trace}(\mathbf{H}^T\mathbf{H})^{-1}} = (\lambda_1^{-1} + \lambda_2^{-1} + \lambda_3^{-1} + \lambda_4^{-1})^{1/2}. \tag{15}$$

Very accurate results can be obtained when using BP to perform the function approximation of Equation (15) after a long training time.

Four types of mapping will be performed, depending on the input-output relationships, which is shown in Figure 3. These three-layer networks have the

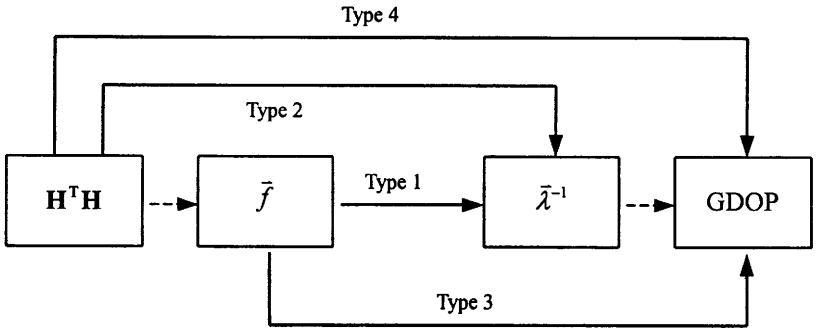

Figure 3. Input-output relationships for four types of mapping using BPNN.

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

'$m-N_{hid}-n$' structures. Based on the Kolmogorov's Theorem, the number of middle-layer neurons $N_{hid}=2m+1$ is selected. The four types of mapping are described as follows:

4.1. *Type 1: Four Inputs Mapped to Four Outputs.* The first type of mapping employs the following property (Simon and El-Sherief, 1995a):

$$f_1(\vec{\lambda}) = \lambda_1+\lambda_2+\lambda_3+\lambda_4 = \text{trace}(\mathbf{H}^{\mathrm{T}}\mathbf{H}) \tag{16a}$$

$$f_2(\vec{\lambda}) = \lambda_1^2+\lambda_2^2+\lambda_3^2+\lambda_4^2 = \text{trace}[(\mathbf{H}^{\mathrm{T}}\mathbf{H})^2] \tag{16b}$$

$$f_3(\vec{\lambda}) = \lambda_1^3+\lambda_2^3+\lambda_3^3+\lambda_4^3 = \text{trace}[(\mathbf{H}^{\mathrm{T}}\mathbf{H})^3] \tag{16c}$$

$$f_4(\vec{\lambda}) = \lambda_1\lambda_2\lambda_3\lambda_4 = \det(\mathbf{H}^{\mathrm{T}}\mathbf{H}). \tag{16d}$$

The $\vec{\lambda}^{-1}$ here is viewed as a functional $\mathscr{R}^4 \to \mathscr{R}^4$ mapping from $\vec{f}$ to $\vec{\lambda}^{-1}$: $\vec{\lambda}^{-1} = \text{fn}(\vec{f})$. The network has the (4-9-4) structure with the following input-output pairs:

$$\text{Input: } (x_1, x_2, x_3, x_4)^{\mathrm{T}} = (f_1, f_2, f_3, f_4)^{\mathrm{T}}$$
$$\text{Output: } (y_1, y_2, y_3, y_4)^{\mathrm{T}} = (\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1})^{\mathrm{T}}$$

The mapping from $\vec{f}$ to $\vec{\lambda}^{-1}$ is highly nonlinear and cannot be determined analytically but can be precisely approximated by the BPNN.

4.2. *Type 2: Ten Inputs Mapped to Four Outputs.* The second type of mapping is trained to approximate the matrix eigenvalue inverses directly from the matrix elements. Because $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ is a 4×4 symmetric matrix, there are only ten elements (e.g., upper triangle elements) required for mapping:

$$\mathbf{H}^{\mathrm{T}}\mathbf{H} = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 \\ & g_5 & g_6 & g_7 \\ & & g_8 & g_9 \\ \text{sym} & & & g_{10} \end{bmatrix}$$

where: $g_k = (\mathbf{H}^{\mathrm{T}}\mathbf{H})_{i,j}$, $1 \leqslant i \leqslant j \leqslant 4$, for $k = 1 \ldots 10$. The $\vec{\lambda}^{-1}$ here is a functional $\mathscr{R}^{10} \to \mathscr{R}^4$ mapping from $\mathbf{H}^{\mathrm{T}}\mathbf{H}$ to $\vec{\lambda}^{-1}$: $\vec{\lambda}^{-1} = \text{fn}(\mathbf{H}^{\mathrm{T}}\mathbf{H})$. The network now has the (10-21-4) structure with the following input-output pairs:

$$\text{Input: } (x_1, x_2, \ldots, x_{10})^{\mathrm{T}} = ((\mathbf{H}^{\mathrm{T}}\mathbf{H})_{1,1}, (\mathbf{H}^{\mathrm{T}}\mathbf{H})_{1,2}, \ldots, (\mathbf{H}^{\mathrm{T}}\mathbf{H})_{4,4})^{\mathrm{T}}$$
$$\text{Output: } (y_1, y_2, y_3, y_4)^{\mathrm{T}} = (\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1})^{\mathrm{T}}$$

4.3. *Type 3: Four Inputs Mapped to One Output.* In the above two types of mapping, the output variables used for training are the eigenvalue inverses. In this case, the network is designed to perform a $\mathscr{R}^4 \to \mathscr{R}^1$ mapping from $\vec{f}$ directly to GDOP, i.e. GDOP = fn($\vec{f}$). Consequently, the network has the simpler (4-9-1) structure:

$$\text{Input: } (x_1, x_2, x_3, x_4)^{\mathrm{T}} = (f_1, f_2, f_3, f_4)^{\mathrm{T}}$$
$$\text{Output: } y = \text{GDOP}$$

4.4. *Type 4: Ten Inputs Mapped to One Output.* The network has the (10-21-1)

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

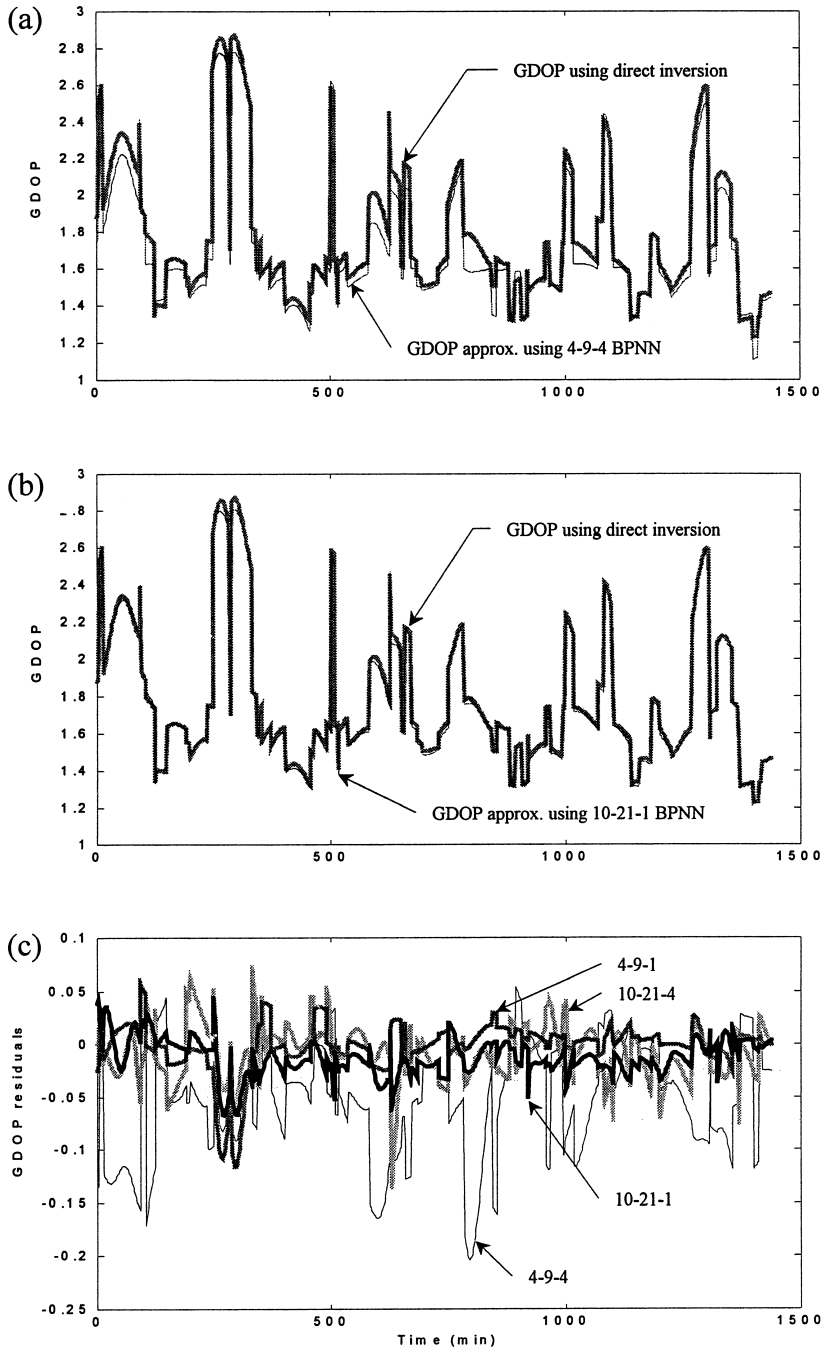

Figure 4. GDOP approximation using (a) the 4-9-4 BPNN; (b) the 10-21-1 BPNN, and (c) comparison of GDOP residuals for four types of mapping.

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

structure, which directly maps $\mathbf{H}^T\mathbf{H}$ to GDOP, i.e. GDOP = fn($\mathbf{H}^T\mathbf{H}$). This is a functional $\mathscr{R}^{10} \rightarrow \mathscr{R}^1$ mapping:

$$\text{Input: } (x_1, x_2, \ldots, x_{10})^T = ((\mathbf{H}^T\mathbf{H})_{1,1}, (\mathbf{H}^T\mathbf{H})_{1,2}, \ldots, (\mathbf{H}^T\mathbf{H})_{4,4})^T$$
$$\text{Output: } y = \text{GDOP}$$

Simulation was conducted to investigate and compare the performance for the four types of mapping. The receiver is simulated to be located at the top of the building of this Institute, which has an approximate position of North 25·15 degrees, East 121·78 degrees, at an altitude of 62 metres ($[-3042329{\cdot}2\ 4911080{\cdot}2\ 2694074{\cdot}3]^T$ m in WGS-84 ECEF coordinates) in a 24-satellite constellation. A 24-hour duration was simulated. The GDOP was computed every one minute, and then collected in two files every other minute, one for training and the other for testing purpose. The BPNN is made of three layers (one input layer, one hidden layer, and one output layer). For comparing training performance, all the four types of mapping are trained with 20000 learning iterations. The learning rate and momentum are selected as 0·5 and 0·6, respectively. The input variables, i.e. f_i and $\mathbf{H}^T\mathbf{H}$, were normalized to the range [0, 1]; and the output variables, i.e. λ_i^{-1} and GDOP, were normalized to the range [0·2, 0·8].

Results showed that, after 20000 learning iterations, the one-output architectures, i.e. (4-9-1) and (10-21-1) types, provided better GDOP mapping accuracy than the four-output architectures, i.e. (4-9-4) and (10-21-4) types. The (4-9-4) type provided relatively poor accuracy while the other three types provided mapping accuracies on about the same order of magnitude. Figure 4 provides the approximate GDOP obtained by the four types of mapping. When using the Multilayer Functional-Link Network (MLFN), which was formed by adding additional inputs using logarithmic and exponential functions, the improved accuracy was not significant.

Since the original outputs of four-output structures (4-9-4) and (10-21-4) are eigenvalue inverses, the mapping accuracies for the eigenvalue inverse are now explored. For the four-output cases, the (10-21-4) architecture predicts the eigenvalue inverses with much better accuracy than does the (4-9-4) architecture. The (4-9-4) type of mapping, after 20000 learning iterations, does not approximate the eigenvalue inverses with very good accuracy, while the eigenvalue inverses obtained by the ten-input (10-21-4) architectures are predicted with very good accuracy. The four eigenvalue inverses approximated by (4-9-4) and (10-21-4) types of mapping are shown in Figure 5. With regard to the ten-input architectures, (10-21-4) and (10-21-1), because ten input variables are required for these two architectures, the number of hidden neurons is increased from 9 to 21, respectively. This makes the minimum dimensions of w_{ij} from 4×9 to 10×21, respectively (based on Kolmogorov's Theorem). However, the ten-input architectures avoid the intermediate step of computing f_i as in Equation (16). In addition, when the one-output architecture (10-21-1) is employed, the above w_{jk} dimension immediately decreases from 21×4 to 21×1, respectively. Also, when using the (4-9-1) architecture instead of the (4-9-4) architecture, the dimension of w_{jk} decreases from 9×4 to 9×1. A brief summary of the mapping performance is shown in Table 1.

5. CONCLUSION.

The BPNN-based GDOP approximation was successfully conducted. Four types of functional mapping were performed and their performances explored. While it is recognized that the BPNN has been most popular throughout

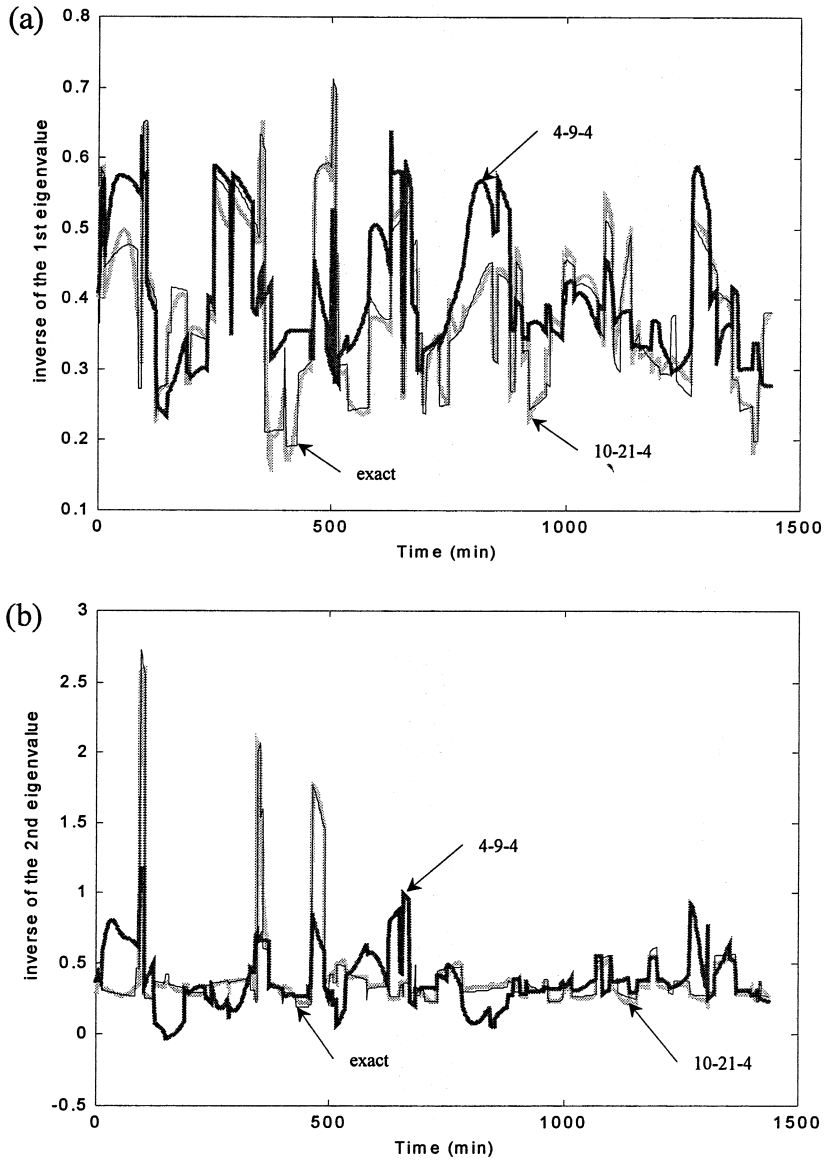

Figure 5. For legend see opposite.

all neural applications, it is also well known to have some drawbacks such as slow learning. To improve the BPNN, one can adjust the learning rates to improve the training time. Much of the NN research literature is devoted to attempts to speed up BP. Instead of presenting better BPNN algorithms, the objective of this paper was mainly on the discussion of mapping performance among four types of network architectures for approximating the GDOP function, hence only the standard back-propagation algorithm was employed. Moreover, the results obtained in this paper can also be used for other applications, such as, for determining the matrix eigenvalues.

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

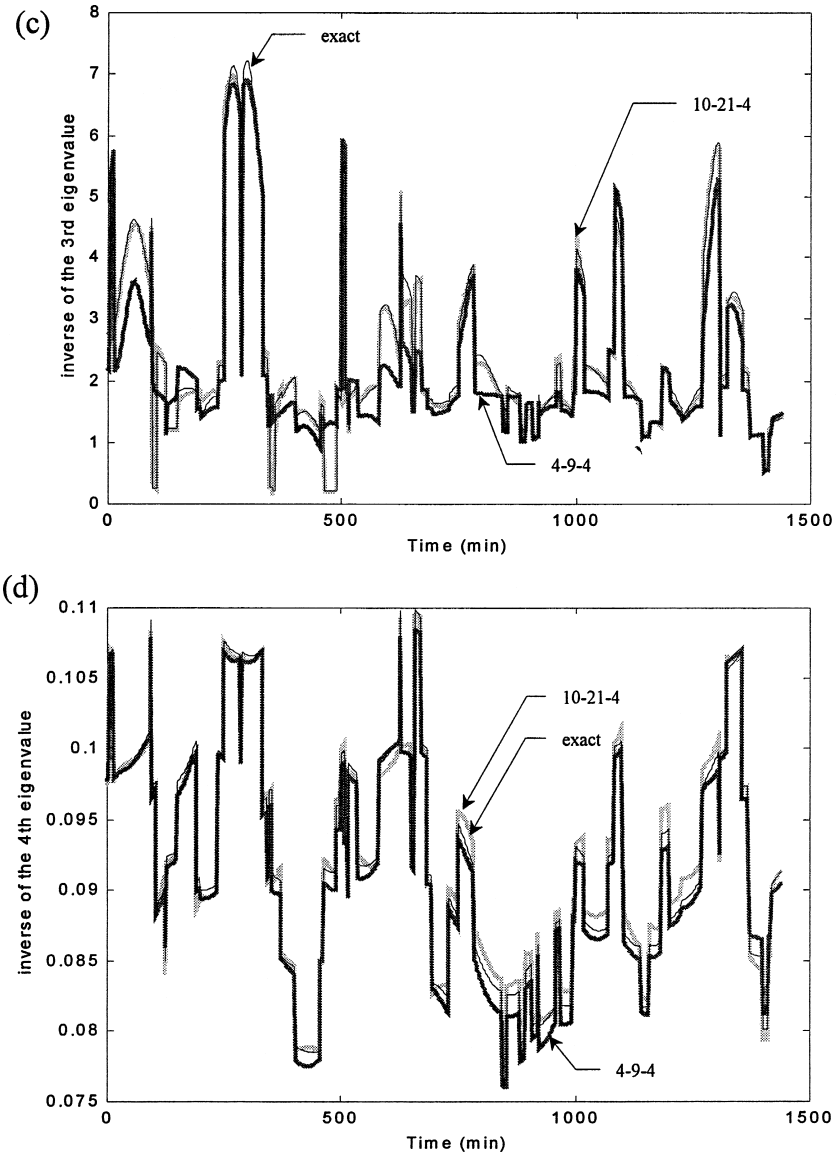

Figure 5. Approximation of eigenvalue inverses: (a) λ_1^{-1}; (b) λ_2^{-1}; (c) λ_3^{-1}; and (d) λ_4^{-1}, using the 4-9-4 and 10-21-4 types of BPNN.

Table 1. Performances of four types of mapping after 20000 iterations of training.

Architecture	4-9-4	10-21-4	4-9-1	10-21-1
GDOP mapping accuracy	good	very good	very good	very good
Error mean	−0·0553	−0·0084	−0·0022	−0·0127
Error standard deviation	0·0526	0·0251	0·0236	0·0174
$\vec{\lambda}^{-1}$mapping accuracy	not as good	very good	N/A	N/A
Intermediate step (i.e. calculation of $\vec{f}$) required?	yes	no	yes	no

https://doi.org/10.1017/S0373463301001606 Published online by Cambridge University Press

ACKNOWLEDGEMENTS

The authors wish to thank Professor Dan Simon of Cleveland State University for sharing his valuable experience.

REFERENCES

Simon, D. and El-Sherief, H. (1995a). Navigation satellite selection using neural networks. *Neurocomputing* **7**, pp. 247–258.

Simon, D. and El-Sherief, H. (1995b). Fault-tolerant training for optimal interpolative nets. *IEEE Trans. Neural Networks* **6** (3), pp. 1531–1535.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, New York.

Kihara, M. and Okada, T. (1984). A satellite selection method and accuracy for the global positioning system. *NAVIGATION*: *Journal of the Institute of Navigation* **31** (1), pp. 8–20.

Stein, B. A. (1985). Satellite selection criteria during altimeter aiding of GPS. *NAVIGATION*: *Journal of the Institute of Navigation* **32** (2), pp. 149–157.

Yarlagadda, R., Ali, I., Al-Dhahir, N. and Hershey, J. (2000). GPS GDOP metric. *IEE Proc.-Radar*, *Sonar Navig*. **147** (5), pp. 259–264.

Jwo, D.-J. (2001). Efficient DOP calculation for GPS with and without altimeter aiding. *This Journal* **54** (2), pp. 269–279.

Widrow, B. and Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptron, madaline, and back-propagation. *Proc. of the IEEE* **78** (9), pp. 1415–1442.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proc. of the IEEE* **78** (9), pp. 1481–1497.

Chester, M. (1993). *Neural Networks: A Tutorial*. Prentice-Hall.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice-Hall.

Yu, X.-H., Chen, G.-A. and Cheng, S.-X. (1995). Dynamic learning rate optimization of the back-propagation algorithm. *IEEE Trans. Neural Networks* **6** (3), pp. 669–677.

Reyneri, L. M. and Filippi, E. (1990). Modified back-propagation algorithm for fast learning in neural networks. *Electron. Lett*. **26** (19), pp. 1564–1566.

Yam, Y. F. and Chow, T. W. S. (1993). Extended back-propagation algorithm. *Electron. Lett*. **29** (19), pp. 1701–1702.

Holt, M. J. J. and Semnani, S. (1990). Convergence of back-propagation in neural networks using a log-likelihood cost function. *Electron. Lett*. **26** (23), pp. 1964–1965.

Polycarpou, M. M. and Ioannou, P. A. (1992). Learning and convergence analysis of neural-type structures networks. *IEEE Trans. Neural Networks* **3** (1), pp. 39–50.