THE JOURNAL OF NAVIGATION (2007), **60**, 291–301. © The Royal Institute of Navigation
doi:10.1017/S0373463307004158 Printed in the United Kingdom

DGPS Correction Prediction Using Artificial Neural Networks

M. Mohasseb, A. El-Rabbany, O. Abd El-Alim and R. Rashad

(*Ryerson University*, *Toronto*)
(Email: rabbany@ryerson.ca)

This paper focuses on modelling and predicting differential GPS corrections transmitted by marine radio-beacon systems using artificial neural networks. Various neural network structures with various training algorithms were examined, including Linear, Radial Biases, and Feedforward. Matlab Neural Network toolbox is used for this purpose. Data sets used in building the model are the transmitted pseudorange corrections and broadcast navigation message. Model design is passed through several stages, namely data collection, preprocessing, model building, and finally model validation. It is found that feedforward neural network with automated regularization is the most suitable for our data. In training the neural network, different approaches are used to take advantage of the pseudorange corrections history while taking into account the required time for prediction and storage limitations. Three data structures are considered in training the neural network, namely all round, compound, and average. Of the various data structures examined, it is found that the average data structure is the most suitable. It is shown that the developed model is capable of predicting the differential correction with an accuracy level comparable to that of beacon-transmitted real-time DGPS correction.

KEY WORDS

1. Differential GPS. 2. Pseudorange differential correction. 3. Artificial Neural Networks.

1. INTRODUCTION.

To enhance the safety of marine navigation some countries around the world have established networks of reference stations around their coastal areas which continuously broadcast real-time differential GPS (DGPS) corrections in the RTCM format. These corrections are digitally modulated using a special form of frequency modulation – the minimum shift keying (MSK). The modulated correction data is then transmitted from the radio beacon at rates between 25 and 200 bps. Typical rates, however, are 100 and 200 bps. In most of the cases, an integrity monitoring unit is co-located with the reference station to monitor its performance (El-Rabbany, 2006).

In the past, a high transmission rate was necessary to mitigate the rapid changes in selective availability (SA). Fortunately, SA was switched off on May 2, 2000, which improved the standalone GPS positioning by a factor of seven or more (El-Rabbany, 2006). In addition, with SA switched off, the differential pseudorange correction (PRC) changes relatively slowly over time (i.e. shows short-term stability). As a result, a lower transmission rate is required which helps reduce the cost of installing and operating a DGPS system. Furthermore, with the deactivation of SA the PRC is expected to have a high day-to-day repeatability (i.e. a long-term stability). This makes

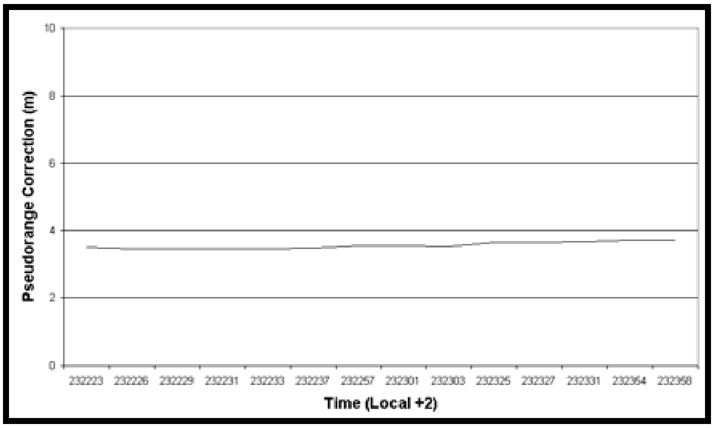

Figure 1. PRC for PRN 01 transmitted by Alexandria DGPS radio-beacon from 232223 to 232358 06/01/2003.

the PRC predictable, which helps further reduce the cost of beacon-based DGPS systems. For example, if the PRC can be predicted with a high degree of accuracy, it could be compared with the real-time PRC to monitor the system performance. This would eliminate the need for integrity monitoring stations, which in turn translates into lower operational costs.

This paper takes advantage of the day-to-day repeatability of differential corrections to produce a predicted set of DGPS corrections using a neural network-based model. Data sets used in building the model include the transmitted differential corrections and the broadcast ephemeris. Model design is passed through several stages, namely data collection, preprocessing, model building and finally model validation. It is shown that the developed neural network-based model is capable of generating predicted differential corrections which match that of the receiver-generated real-time corrections.

2. PRC VARIABILITY.

Prior to developing our neural network model, it was necessary to verify the short- and long-term variability of the PRC data. PRC data transmitted by Alexandria DGPS reference station were collected at different time windows for this purpose. As shown in Figure 1, the PRC changes rather slowly over time. The figure shows that the differential correction could be delivered to the DGPS receiver with a latency of 90 seconds with no noticeable degradation in the DGPS positioning accuracy. Similarly, Figure 2 shows the PRC data over three days for PRN 20. As can be seen there is a high day-to-day repeatability, which suggests that the PRC data of a particular day can be predicted using data from previous days.

3. NEURAL NETWORK MODELLING.

An Artificial Neural Network (ANN) consists of a set of processing elements called neurons that interact by sending signal to one another along weighted connections. These elements are inspired by biological nervous systems. The connection weights, which can be determined adaptively, specify the precise knowledge representation. The advantageous and distinguishing feature of neural networks is their ability to learn. The network in

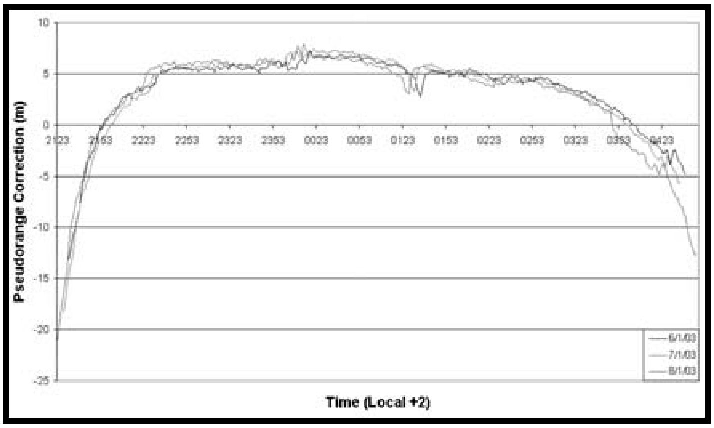

Figure 2. PRC for PRN 20 transmitted by Alexandria reference station from 06/01/2003 to 08/01/2003.

the adaptive mode abstracts and generalizes the function character in the process of learning from training patterns. The learning algorithm is an optimization method capable of finding weight coefficients and thresholds for a given neural network and a training set. Two approaches are used in training ANN: batch and incremental. Batch training of a network proceeds by making weight and bias changes based on an entire set (batch) of input vectors. Incremental training changes the weights and biases of a network as needed after presentation of each individual input vector. Incremental training is sometimes referred to as "on line" or "adaptive" training. ANN has been used extensively in GPS and other fields (Suna and Kalenderli, 2003; Watts, 2001; Chiang, 2004; Jwo and Pai, 2004).

To select the appropriate network architecture, modelling is divided into two stages, short span fitting and long span fitting. PRC data transmitted by Alexandria DGPS reference station were collected on 17 and 18 October, 2004 for this purpose. The first 1000 samples of PRN 01 on 17/10/2004 were used for model training and the first 1000 samples on 18/10/2004 were used for model testing. If the network succeeded in short span fitting then it was used for the long span fitting where the pseudorange correction of PRN 01 on 17/10/2004 was used (for the whole period it was over the horizon, 21 400 samples) in model training and the pseudorange correction of PRN 01 on 18/10/2004 in model testing.

Three types of neural network design were tested using MATLAB 6.5 (Hagan and Denuth, 1995) to find the most appropriate one that matched the nature of the correction data. In the entire neural network the input data were arranged in matrix of concurrent vector. The three network types are Linear Layer, Radial Bases, and Feedforward. Table 1 lists the training algorithms used along with its category and MATLAB abbreviation.

3.1. *Linear Neural Network.* The linear neural network has the following architecture:

- Number of layers: one.
- Number of neurons: one.
- Transfer function: Linear Transfer Function.
- Learning rules: Least Square Error.

Table 1. Backpropagation training algorithms for Feedforward Neural Network architecture used in modelling the pseudorange correction.

MATLAB abbreviation	Algorithm	Category
traingdx	1-1-Variable Learning Rate	1-Gradient Descent
trainrp	1-2-Resilient Back	
triancgp	2-1-Polak-Ribiere Update	2-Conjugate Gradient
triancgb	2-2-Powelle-Beale Restarts	
trianscg	2-3-Scaled Conjugate Gradient	
trianbgf	3-1-BFGS	3-Quasi-Newton
trainoss	3-2-One Step Secant	
trainlm	4-1-Levenberg-Marquardt	4-Levenberg-Marquardt
trainbr	4-2-Levenberg-Marquardt with Automated Regularization	

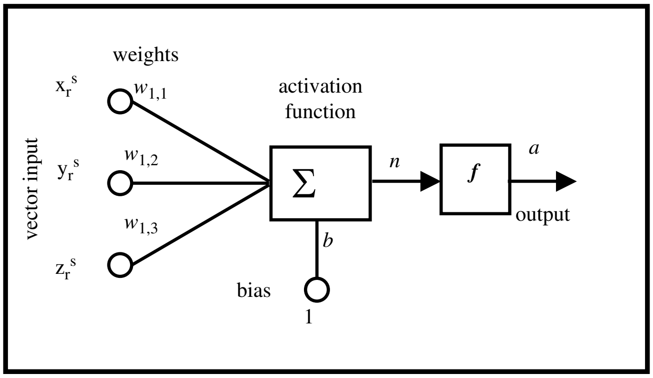

Figure 3. Single neuron with three elements input vector.

The construction of the linear layer is illustrated in Figure 3, where the figure shows one neuron with three elements input vector. Figure 4 shows the same structure but with MATLAB notation which will be used in the rest of this paper.

The output of liner neuron is given by:

$$a = f_{lin}(n) = f_{lin}(Wp + b) = (Wp + b) \tag{1}$$

Where weights W and bias b are calculated from inputs vector p and targets T by solving the linear equation to minimize the mean square error given by:

$$[Wb] * [p;1_1 \ \dots \ 1_q] = T \tag{2}$$

Where q is the number of training patterns. Linear layer managed to model the short span training data set (when simulated with same data used in the training). However, when the network was simulated with the test data set, it failed to retrieve the correction again.

3.2. *Radial Basis Neural Network*. A Radial Basis network is shown in Figure 5 and has the following architecture:

- Number of layers: two.
- Number of neurons: variable in the first layer and one in the second layer.

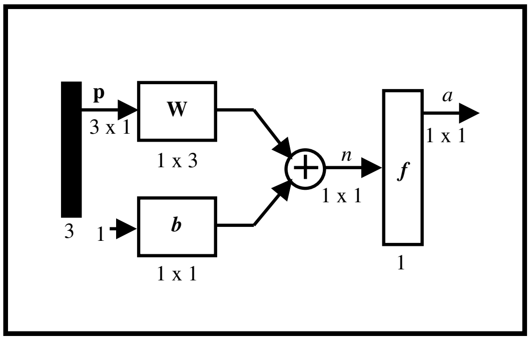

Figure 4. Single neuron with three elements input vector in block diagram notation.

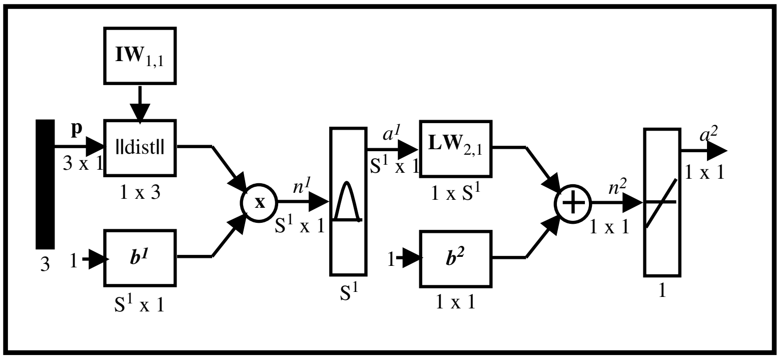

Figure 5. Radial Basis Neural Network architecture for modelling PRC.

- Transfer function: Radial Biases Function in the first layer and Linear Transfer Function in the second layer.

Three types of Radial Basis Network are tested, namely Exact Design, More Efficient, and Generalized Regression Network. The output of both Exact Design and More Efficient is given by:

$$a^2 = f_{lin}(n^2) = f_{lin}(LW_{2,1}a^1 + b^2) = (LW_{2,1}a^1 + b^2) \tag{3}$$

$$a_i^1 = f_{rad}(n^1) = f_{rad}(prod(\|_i IW_{1,1} - p\|, b_i^1)) \tag{4}$$

$$\text{where: } f_{rad}(n) = exp(-n^2) \tag{5}$$

$$\|w - P\| = \sqrt{\sum (w-P)^2} \tag{6}$$

where: $H = prod(X, Y)$ for $i = 1$ to $lH_i = X_i * Y_i$

The output of Generalized Regression Network is given by:

$$a^2 = f_{lin}(n^2) = f_{lin}(nprod(LW_{2,1}a^1)) \tag{7}$$

where $nprod(w, p) = \dfrac{wp}{\sum P}$.

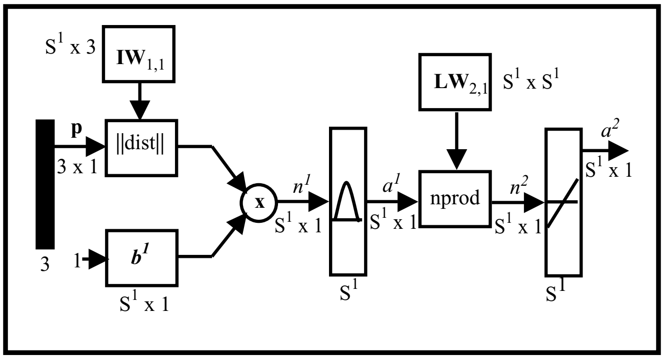

Figure 6. Generalized Regression Neural Network architecture for modelling PRC.

In Exact Design, weights in the first layer are set to the inputs vector *P*' where the weights of second layer is given by solving the linear equation to minimize the mean square error between a^1 and T as in Equation 2. In More Efficient, a similar approach is used, the only difference that it adds a single neuron each iteration until the sum of the squared error falls beneath the error goal or the maximum number of neurons has been reached. In Generalized Regression Network, weights of first layer are set as Exact Design but weights in second layer are set to T.

3.2.1. *Exact Design.* Although the Exact Design managed to model the short-term training data set, it failed with the test data set. Even if the Exact Design had managed to model the test data set, it could be used because it created a neuron for every record in its training data. For the long-term data set, the Exact Design would need 2400 neurons, which is not practical for pseudorange correction data.

3.2.2. *More Efficient Design.* More Efficient Design behaves in the same way as Exact Design; it managed to model the short-term training data set, but failed with the test data set.

3.2.3. *Generalized Regression Network.* Generalized Regression Network is similar in architecture with the Radial Basis except that the input to the second layer is normalized, as shown in Figure 6. Generalized Regression Neural Network managed to model both the short-term training data set and test data set. However the design of Generalized Regression Network is not efficient, it uses one neuron per record in the training data. As such, it is not practical to use this network in PRC modelling.

3.3. *Feedforward Neural Network.* The Feedforward Neural Network has a wide variety of architectures and backpropagation training algorithms. To find a suitable Feedforward architecture, one design was adopted and several training algorithms were used to train the neural network. The Feedforward architecture selected was based on guidelines in (Masters, 1993) and is shown in Figure 7.

- Number of layers: three.
- Number of neurons: three in the first layer, twenty in the second layer, one in the third layer.
- Transfer function: Tansig Function in both the first and the second layer, and Linear Transfer Function in the third layer.

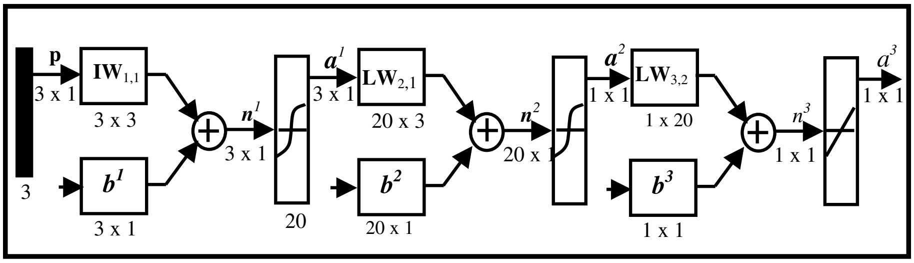

Figure 7. Feedforward Neural Network architecture for modelling the PRC.

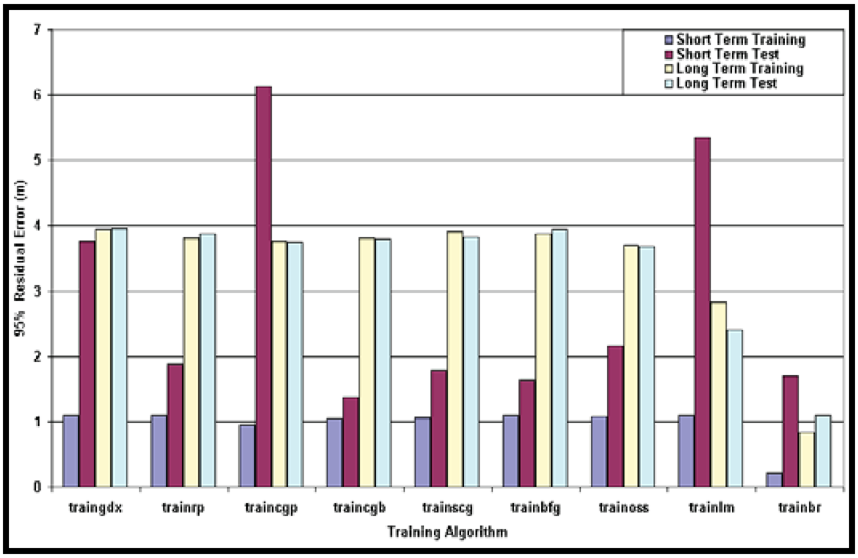

Figure 8. Results of training the Feedforward neural network with different backpropagation training algorithms for modelling pseudorange correction.

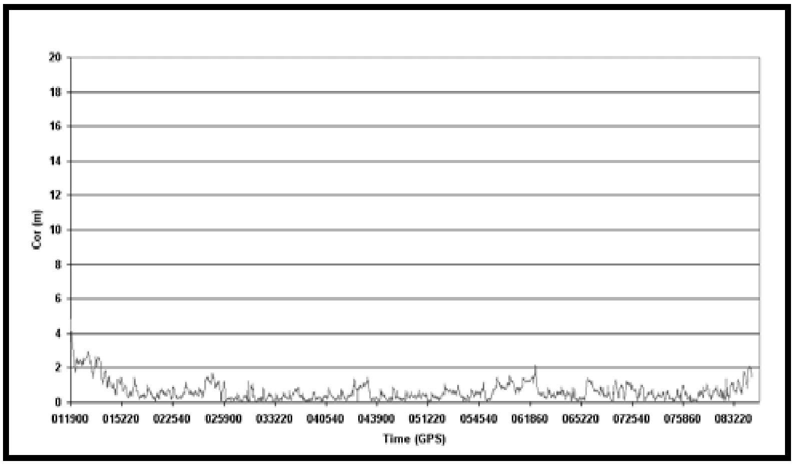

Figure 9. Difference between the predicted and transmitted PRC correction for PRN 13.

The output of Feedforward Neural Network is given by:

$$\begin{aligned} a^3 &= f_{lin}(n^3) = f_{lin}(LW_{3,2}a^2 + b^3) = (LW_{3,2}a^2 + b^3) \\ a^2 &= f_{hts}(n^2) = f_{hts}(LW_{2,1}a^1 + b^2) \\ a^1 &= f_{hts}(n^1) = f_{hts}(W_{1,1}P + b^1) \end{aligned} \tag{8}$$

where; $f_{hts}(n) = \dfrac{2}{(1+exp(-2n))} - 1$

The backpropagation training algorithms used were classified into four categories, Gradient Descent, Conjugate Gradient, Quasi-Newton, and Levenberg-Marquardt (Bose and Liang, 1996; Fausett, 1994).

3.3.1. *Gradient Descent.* In Gradient Descent (steepest descent) weights are updated according to:

$$W(k+1) = W(k) + \alpha d_k \tag{9}$$

where α is the learning rate parameter and $\boldsymbol{d}_k$ is the negative of the gradient

$$d_k = -g_k \tag{10}$$

To speed the convergence of the steepest descent, two approaches were used; Variable Learning Rate and Resilient Back. In Variable Learning Rate, the magnitude of the learning rate was increased if the learning in the previous step had decreased the total error function. Conversely, if the error function had increased, the learning rate was decreased. The objective of the Resilient backpropagation training algorithm is to eliminate the harmful effects of sigmoid functions on the magnitudes of the partial derivatives, because the slope of sigmoid functions approaches zero as the input gets large. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value.

3.3.2. *Conjugate Gradient.* In Conjugate Gradient weights are updated according to

$$W(k+1) = W(k) + \alpha_k d_k \tag{11}$$

where α_k is the conjugate vector coefficient (step length) and $\boldsymbol{d}_k$ is conjugate gradient direction. Initially $\boldsymbol{d}_k$ is taken to be the direction of the steepest descent.

$$d_0 = -g_0 \tag{12}$$

In the following iteration $\boldsymbol{d}_k$ is computed from:

$$d_{k+1} = -g_{k+1} + \beta_k d_{k+1} \tag{13}$$

and in Polak-Ribiere, update β_k is computed from:

$$\beta_k = \frac{\Delta g_{K-1}^T - g_K}{g_K^T g_{K-1}} \tag{14}$$

In Powelle-Beale Restarts, the conjugate gradient direction is set to the negative of the gradient if there is little orthogonality between the current gradient and the previous gradient. Computing the conjugate vector coefficient (α_k) requires too much computation, Scaled Conjugate Gradient is an algorithm developed to avoid such computation.

3.3.3. *Quasi-Newton*. In Newton methods weights are updated according to

$$W(k+1) = W(k) + \alpha_k H^{-1}(-g_k) \tag{15}$$

Where α_k is one, g_k is the gradient (as in the steepest descent), and $\boldsymbol{H}^{-1}$ is the inverse of the Hessian matrix. For Quasi-Newton methods $\boldsymbol{H}^{-1}$ is reached through a series of matrices beginning with the unit matrix $\boldsymbol{I}$ and ending with the inverse of the Hessian matrix $\boldsymbol{H}^{-1}$. The Quasi-Newton algorithm that employs the BFGS (Broyden, Fletcher, Golfarb, Shanno) formula for updating the Hessian matrix is called the BFGS Algorithim.

3.3.4. *Levenberg-Marquardt*. In the Levenberg-Marquardt method weights are updated according to

$$W(k+1) = W(k) - [JJ^T + IJ]^{-1} J^T e \tag{16}$$

Where $\boldsymbol{J}$ is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases and $\boldsymbol{e}$ is a vector of network errors.

All algorithms managed to train the proposed Feedforward Neural Network to model the pseudorange correction with its input. The network modelled both short and long span data sets. The best performance for accuracy and stability is the Levenberg-Marquardt with Automated Regularization. Figure 8 shows the residual error of modelling the short and long-term data sets using the training algorithms listed in Table 1.

4. RESULTS AND DISCUSSION.

The performance of the proposed neural network-based model was assessed using a PRC data set collected over 24 hours from 14:00 21/10/2004 to 14:00 22/10/2004. The ANN-based predicted DGPS (PDGPS) corrections were first compared with the transmitted PRC counterpart. As can be seen in Figure 9, the difference between the predicted and transmitted PRC corrections for PRN 13 is less than 2m most of the time. The PRC of other satellites exhibit similar error differences. To further assess the performance of the proposed neural network-based model, we compared the horizontal positioning accuracies from three positioning modes, namely standalone GPS, DGPS, and PDGPS. Figures 10 and 11 show the results. It can be seen from Figure 10 that, although the accuracy of the standalone GPS has dramatically improved as a result of SA removal, it is still low compared to that of DGPS. On the other hand, the accuracies of DGPS and PDGPS are comparable (Figure 11), which shows that the ANN-based model is capable of modelling and predicting the differential corrections.

5. CONCLUSIONS.

It has been shown in this paper that artificial neural networks can effectively be used to predict the pseudorange differential corrections with a high accuracy level. In comparison with beacon-transmitted real-time DGPS corrections, the ANN-based PDGPS model developed was found to deviate by a maximum of 2m most of the time. This resulted in a positioning accuracy comparable to that of DGPS. Practical applications of the proposed model include its use to monitor the quality and integrity of beacon-transmitted real-time DGPS corrections. This would eliminate the need for integrity monitoring stations, which in turn translates into lower operational costs.

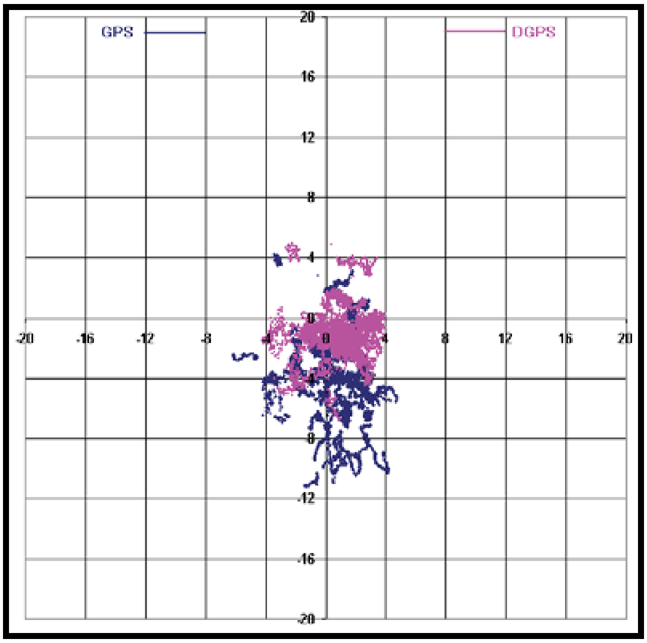

Figure 10. Horizontal positioning accuracy from standalone GPS and DGPS.

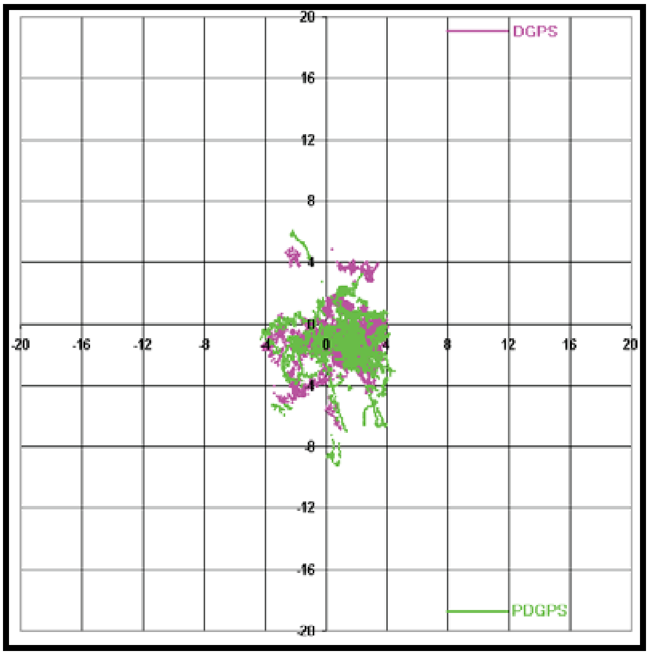

Figure 11. Horizontal positioning accuracy from PDGPS and DGPS.

REFERENCES

Bose, N., and Liang, P. (1996). *Neural Network Fundamentals with Graphs, Algorithms, and Application*. McGraw-Hill, International Edition, USA.

Chiang, K.-W. (2004). *INS/GPS Integration Using Neural Networks For Land Vehicle Navigation Application*. PhD thesis. The University of Calgary, Canada.

El-Rabbany, A. (2006). *Introduction to GPS: The Global Positioning System*. Second edition. Artech House Publishers.

Fausett, L. (1994). *Fundamentals of Neural Networks Architectures, Algorithms and Applications*. Prentice Hall International, Inc., USA.

Hagan, M., and Denuth, M. (1995). *Neural Network Design*. Thomson Learning, USA.

Jwo, D., and Pai, C. (2004). Incorporation of Neural Network State Space Estimator for GPS Attitude Determination. *The Journal of Navigation*, **57**, 117–134.

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press, Inc., USA.

Suna, B., and Kalenderli, O. (2003). Electrode Contour Optimization By Artificial Neural Network With Levenberg-Marquartdt Algorithm. *IJCI Proceedings of Intl. XII. Turkish Symposium on Artificial Intelligence and Neural Networks*, **1**, 1.

Watts, D. C. (2001). *Land Cover Mapping by Combinations of Multiple Artificial Neural Networks*. MSc thesis. The University of Calgary, Canada.