Chapter 29

Feeding Neural Network Models with GPS Observations: A Challenging Task

R.F. Leandro
Department of Geodesy and Geomatics Engineering,
University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, Canada, E3B 5A3

C.A.U. Silva
Department of Civil construction, Federal Technologic Learning Centre, Belém, Para, Brazil

M.C. Santos
Department of Geodesy and Geomatics Engineering,
University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, Canada, E3B 5A3

Abstract. Much has been done in terms of functional and stochastic modelling of observations in space geodesy, aiming at the development of adequate adjustment models. One of the techniques, which has been the focus of more attention in the last years, is the Neural Network model. Although not trivial to be used, this kind of model provides an extreme adaptation capability, which can be an issue of fundamental importance for certain applications. In this paper we discuss the use of GPS observations in Neural Networks models, providing a brief description how a neural model works and what are its restrictions, as well as how to treat the GPS observations in order to satisfy them.

A Neural Network is an information processing system formed by a big number of simple processing elements, called artificial neurons. Typically the input values must be normalized, with typical range [0,1], or alternatively [-1,1]. After processed, the signal can be transformed back to its original origin and amplitude. When dealing with GPS observations, namely ranges and range rates, the absolute numerical values are usually pretty large (e.g. order of 20 millions of meters for ranges) coupled with precisions in the order of mm for carrier-phase and meter for pseudoranges. The observations need to be modified to avoid degrading their precision during the normalization, in order to make the application of neural models suitable for GPS data.

In this work methods to make the use of GPS data possible in neural models are discussed and showed with real examples. The analysis is made for both pseudoranges and carrier-phases. It is demonstrated that with the adequate treatment the use of those observables can be made without degradation of precision.

Keywords. Neural Networks, GPS.

1 Introduction

Modelling plays a fundamental role in Geodesy. Signal processing, physical phenomena functional modelling, interpolation, forecasting, stochastic modelling are a few examples of the applications that require modelling in Geodesy. In most cases adjustments are used, and in this case, the most used technique is the least squares procedure. Filters are also widely used, such as Kalman filter, which involves some of the least squares technique concepts. In the ninety's a new technique appeared to be useful in Geodesy, called Neural Networks. Primarily developed for computing applications, such as pattern recognition, neural networks have been adapted to be used in several fields of science, including Geodesy. Those adaptations are needed because usually the situations and problems encountered in computer science are sometimes very different than in other fields. Geodesy was not an exception in this case, and because of that, the Neural Network analyst for geodesy needs to have a good knowledge in neural data processing in order to be able to use this technique successfully. Adaptation of the geodetic data may be needed in some cases to make the data useful in a neural model. This is the case explored in this paper, where the problem of using GPS data in neural networks is shown. Compatibility between GPS and the neural model has to be made possible by means of some modification in the original GPS data. This innovating synergy has made necessary the development of novel techniques in terms of GPS data handling.

2 Neural Network Models

A Neural Network is an information processing system formed by a big number of simple processing elements, called artificial neurons, or simply neurons. The first artificial neuron model was presented by Rosenblatt (1958), who called it *perceptron*.

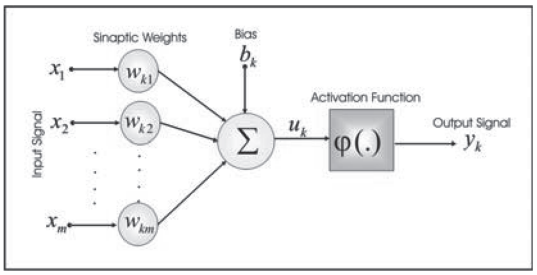

Fig. 1 Nonlinear artificial neuron model (Adapted from Haykin, 1999).

A perceptron computes its input as a linear combination of its input signal by using the synaptic weights. The synaptic weights play the role of parameters, which are adjusted at the training process (this procedure will be discussed later in this section). The synaptic weights hold the knowledge of the network. After that an activation function is applied to the neuron input to generate the neuron output (in the case of a single neuron it is already the output signal). One neuron can have one or more outputs, always with the same value. In the case of a identity activation function, the neuron plays the role of a linear model. The processing of a neuron k can be represented by:

$$y_k = \varphi\left(\sum_{i=1}^{m}(x_i \cdot w_{ki}) + b_k\right), \qquad (1)$$

where y_k is the neuron output, φ is the activation function, m is the number of input parameters, x_i is the i-th input parameter, w_{ki} is the i-th synaptic weight and b_k is the bias.

Typically the order of normalized amplitude of a neuron output is in within the range [0,1], or alternatively [-1,1]. This range depends on the type of activation function used. The neural model also includes a term that is applied externally, called bias and represented in Figure 1 by b_k. The bias has the function of increase or decrease the neuron input.

It is possible to introduce a functional link into the network as an additional layer of neurons, called a hidden layer. This layer can be composed of one or more neurons. The input signal of the hidden layer neurons is generated by the output signal of the input layer. The output signal of the hidden layer is used to generate the input signal to the output layer. It is also possible to introduce not just one, but several hidden layers into the model.

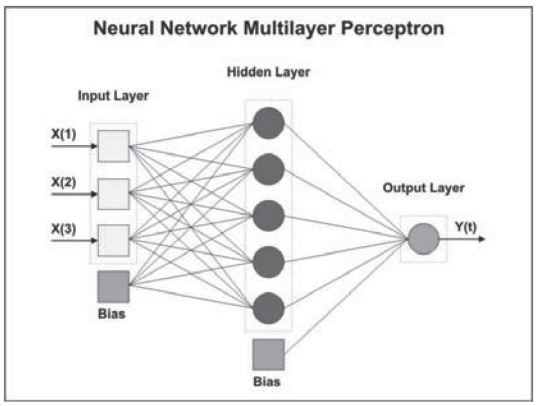

Fig. 2 Neural Network Multilayer Perceptron.

Figure 2 shows a scheme of a neural network with one hidden layer. In this example, x(1), x(2) and x(3) are the input parameters and y(t) is the output parameter. Each element, excepting the biases, is a neuron. Each of these neurons is a processing element that works according to equation (1). The synaptic links (the lines in the draw) connect the different layers, carrying the output signal of a previous one to generate the input signal of the next one. Each synaptic link of the network has a corresponding synaptic weight that is applied to the flowing signal that is going through it.

Another issue of a neural network model is the number of neurons of each layer. This number is fixed to the input and output layers, in function of the input and output parameters. For the hidden layers this number is arbitrary. The model resulting from adding hidden layers between the input and output layers is called Multilayer Perceptron (MLP). The MLP is not the only type of neural network model, but is one of the most popular ones, due to its high adaptation capability and its applicability to a wide group of different applications.

It is necessary not just to know which model will be used, but also all its characteristics, such as the number of hidden layers, the number of neurons in each hidden layer, the activation function of each

layer, etc. There are others more specific characteristics that will not be discussed here.
Once we have a model defined, it is necessary to train the neural network with real data. Such data is composed by a set of known input and output parameters. The training process is not more than an adjustment of the synaptic weights to the data set. This adjustment attempts to decrease the residuals of the output of the network. The residuals are the difference between the computed output and the known output. Based on these residuals it is performed an update of the synaptic weights. Due to the complexity of neural networks, the adjustment can not be done with a direct computation, so the so called training algorithms, which are a type of iterative adjustment of the synaptic weights, are used. One of these algorithms is the backpropagation training algorithm, which is composed by two steps. The first one is the feed-forward, when the Network is fed with a set of inputs that are propagated through the links, from the input layer to the output layer. After that the output value is compared with the known output (from the same set) and the residuals are computed. The second step is the feed-backward. In this step the network is fed with the errors of the previous step, which are propagated through the network from the output layer to the input layer (backwards). During the feed-backward step the synaptic weights are adjusted, using the partial derivatives of each activation function with respect to each synaptic weight. In this step, an additional parameter is also used, which is the learning rate. The learning rate controls how much the weights are going to be updated, given the derived correction. These two steps are carried out several times, for all training sets (which are also called training patterns), and for several epochs (one epoch is the cycle where all patterns are used once in the training process). It is made several times to each parameter up to the residuals converge to a desired threshold value. After the training process we have a Neural Network Model with adjusted synaptic weights according to the training parameters. Figure 3 shows a plot of a training process.

In the example above, the goal of the training process was to achieve a value for the summation of the squared residuals (or errors) of 0.001 (the straight line in the plot). This number is dimensionless because of the normalization process that all data has to go through before it can be used

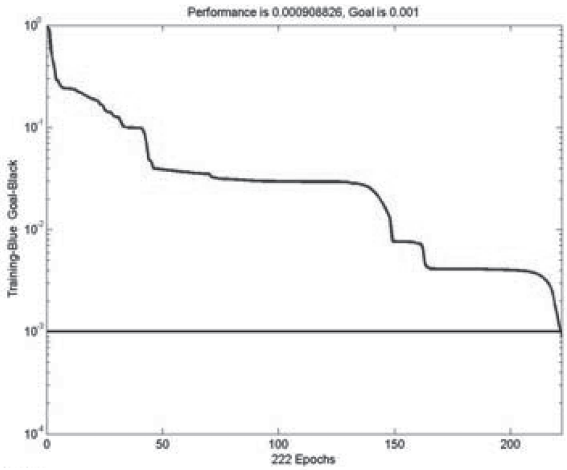

Fig. 3 Neural network training process

in the neural model. In the case of this figure, the goal was achieved in 222 epochs, when the network got a value less then the imposed goal. Usually the initial value of the residuals is high, because the weights of the network have to be initialized with arbitrary or randomic values.

Neural networks can be used in three distinct ways: as intelligence simulation systems, as real time adaptation processors or as data analysis systems. One of the advantages of neural networks models is their high flexibility, allowing solving problems for dynamic and non linear systems. The generalization capability allows estimations of values not used in the training process. In this case, neural models can be successfully used for approximation, interpolation, extrapolation, time prediction. Their complex nature allows the use of hyper surfaces to represent the phenomena modeled, and the training process makes them self-adaptative. These characteristics make the neural networks an attractive solution, since they are theoretically capable of approximate any surface (and therefore modeling any phenomena) without a high knowledge of the phenomena modeled.

In order to successfully conceive a neural network model, some steps are necessary. One of the steps is the definition of the optimal architecture of the network, for the specific application it is going to be used. The architecture consists of the definition of the number of layers, and the number of neurons in each layer. Together with this step, there are also other parameters to be established, such as the learning rate, type of activation functions and

normalization interval. All of them play an important role in the training process, and an important role in the generalization capability of the neural model. Seeking the optimal set of parameters and architecture for a neural network can be a hard work. Although some authors define some rules to setup a model, there is not a defined procedure to follow in order to build a model.

In the testing procedure (which is performed to test the efficiency of candidate models for certain application) depending on the complexity of the architectures the training procedure can be time consuming, even for fast computers. The training time of a neural network can vary from a few seconds to several days. Even though it is necessary to spend some time to set an optimal configuration, the results of the neural network estimations usually pay off the effort made.

3 Neural Networks for Geodesy

Neural networks showed to be an attractive alternative in terms of modeling for several areas, and it was not different for geodesy.

Since early nineties several authors have published papers reporting the use of neural models in geodesy, for different applications. These applications include navigation (e.g. Dumville and Tsakiri (1994), Chansarkar (1999), Vickery and King (2002)), geoid approximation (e.g. Kuhar et al (2001)), weather parameters (atmosphere and space) interpolation and forecasting (e.g. Xenos and Stergiou (2002), Leandro (2004), Leandro and Santos (2004)), prediction of Earth orientation parameters (e.g. Schuh et. al. (2002)), design of networks (e.g. Chang et. al. (1996)) and others.

Usually the utilization of neural models is attempted in order to solve problems where deterministic and statistic tools still requires improvements, or where there isn't a very good knowledge about the modeled phenomena. A potential application is also the dynamic filters where the real time adaptation capability of neural networks fits pretty well. This is the case for example of the navigation applications, when the neural model has to "learn" the pattern of the movement, playing the role of a predictive filter.

The models designed for approximation and interpolation usually target phenomena for which the modeling with linear models is complicated, or the deterministic functions don't fit very well on the approximated surface. As said in the previous section, a neural network can theoretically approximate any surface, depending only on whether an optimal configuration (architecture and other parameters of the neural model) is found or not for the specific case. So, in theory, with the optimal configuration, neural network models are capable of perfect fitting for these cases. The difficulties to achieve it are the determination of the optimal configuration, and the size of the data set available to train the network.

In terms of forecasting, neural networks have also a good advantage, given by their adaptation capability. While when using deterministic models a beforehand, complete knowledge of the modeled effect is necessary, in case of neural networks the model can adapt itself to the behavior of the time series, in order to be able of forecast future occurrences. Also, when it's considered that the series has a stochastic component, there is need of using a stochastic model coupled with the deterministic model. In case of neural networks there is no such need, because a single neural model can be used for either or both components. Leandro and Santos (2004) have showed this kind of application, using neural networks for space weather parameters forecasting. Another example that involves time series is the case of GPS cycle slip detection and correction, where neural models can be used to predict values for GPS carrier phase measurements.

Far from being a concurrent to the actual modeling techniques, such as least squares, neural networks can play an important role as an additional key for complex problems. Neural models differ from least squares adjustments in several aspects, such as:

- ✓ Synaptic weights in neural models play the role of parameters in least squares;
- ✓ The architecture of a neural model plays the role of the functional model in least squares;
- ✓ Patterns used in the training process can play the role of observations and/or constraints in a least squares adjustment, depending on the training RMS goal;
- ✓ Test patterns play the role of control points;
- ✓ Errors as called in neural processing are similar to observation residuals in adjustments;
- ✓ The training process plays the role of the parameters adjustment;
- ✓ The learning rate plays the role of an observation weight (it controls the update of the parameters/synaptic weights);

- ✓ An epoch in neural training process plays the role of an iteration in an adjustment;
- ✓ In least squares adjustment, the functional model has to be built to represent the phenomena. In neural networks, the architecture is built in order to be able of adapting itself to the phenomena.

Even thought there are several differences between them, in some cases neural networks and least squares can be along side. This is the case of using neural networks as the transition part of Kalman filtering. For example one can take advantage of the well known functional model for positioning to build the update adjustment of a navigation filter, together with a neural network playing a role of a self learning navigation estimator as the transition.

Neural networks, when used properly, are a very attractive tool that can make a difference enhancing actual techniques with a high adaptation capability. Its potential applications are countless, being restricted only by the geodesist's imagination to link a neural model to a geodesy problem. As the proposed subject of this work, some of those techniques are discussed in next section.

4 Feeding Neural Networks with GPS observations

When we talk about using GPS data in neural networks as a relevant subject, the first question that arises is why should the data need special treatment to be useful. Actually, there are some reasons to that and the first one (maybe the more important one) is the fact that in neural processing, all values should be set between 0 and -1, or between -1 and 1. This implies that values of the order of tens of thousands of kilometers (in the case of pseudoranges) or up to hundreds of thousands of cycles (in the case of carrier phase) must be fit into small intervals, and the model still needs to be capable of providing estimations with precision compatible with the original data. Other aspect to be considered is the compatibility between training patterns from different observations and the compatibility between training and testing patterns. The generalization capacity of the model depends on this coherency when transforming the original data set.

The value which is most used in neural networks for performance analysis is the MSE (Mean Squared Error). The MSE can be computed with the equation:

$$MSE = \frac{1}{np}\sum_{i=1}^{np} e^2 , \qquad (2)$$

where np represents the number of patterns used in the training process. It is easy to notice that the MSE is nothing more than the variance of the neural network estimations. Therefore, with this value we can relate the performance of the network with estimated variances for real data. However, since the data handled by neural models is normalized, a transformation is needed, which can be easily derived from error propagation law:

$$\sigma^2 = NF^2 \cdot MSE , \qquad (3)$$

where σ^2 is the estimated variance and NF is the normalization factor. Given expressions 2 and 3, one can say that the variance of neural network estimation is function of the MSE (in this case, the performance of the network) and the normalization factor used to fit the data for neural processing usage. Therefore, the value of the normalization actor plays a fundamental role in the estimation variance, when the estimations are transformed back from the normalized domain to their real values.

In the examples used in this work, the neural network model was designed to estimate L2 observations (pseudorange and carrier phase), given the observations at frequency L1. In order to do so, the training process was performed with data from a GPS network, and the simulation (with the neural network already trained to be able of estimate L2 observables) was made for a given receiver. Figure 4 shows the scheme of the data processing.

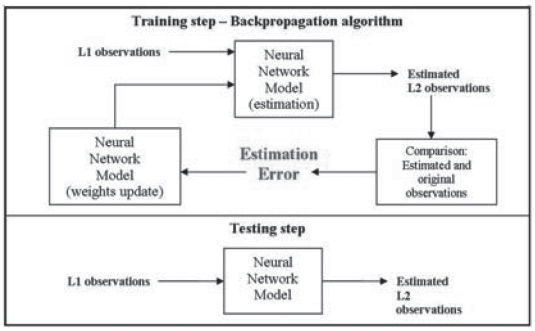

Fig. 4 Data processing scheme.

In this case the first obstacle was to fit carrier phase measurements into a normalized interval. The first decision made was setting the normalized interval to be between -1 and 1, which provides a

normalization factor two times smaller than the alternative option (0 and 1). Figure 5 shows an example of carrier phase measurements for L1 and L2.

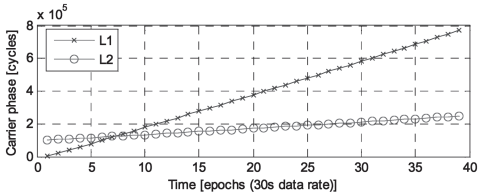

Fig. 5 Carrier phase measurements (L1 and L2).

As can be seen in the figure above, there is a difference of reference between the two frequencies, given by the different numbers assigned for the initial cycle ambiguity, and a difference in rate, given by the different wavelengths. Because of these differences, it would be hard to relate L1 and L2, as wanted. This is a typical problem of coherence between patterns generated from different observables. Since the initial value of the phase counter is arbitrary, mistreating this difference can lead to high normalization factors. A second implication is that each receiver-satellite pair would have a different value for this difference. To solve this problem the meter was used as common unit for both frequencies. Also the counters were synchronized, assigning the same initial phase count for both (in this case zero) as shown in Figure 6.

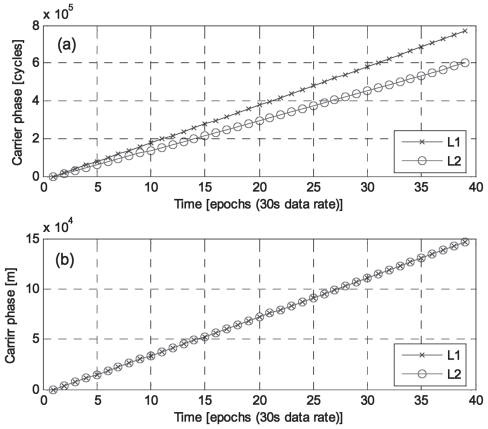

Fig. 6 Carrier phase measurements (L1 and L2) after synchronization (a) and transformed to meters unit (b).

After the above modifications, the problem to relate the two frequencies is solved. However the values for normalization factors would be still high, because of the wide range of carrier phase values, and there are a few techniques that can be used.

The first one to be mentioned is to add pseudo slips in the phase counting. This technique consists of adding false cycle slips in the observations, what allow setting the maximum value of the counter, in order to maintain the normalization factor under a certain value. The result of applying pseudo slips in the carrier phase measurements is shown in Figure 7.

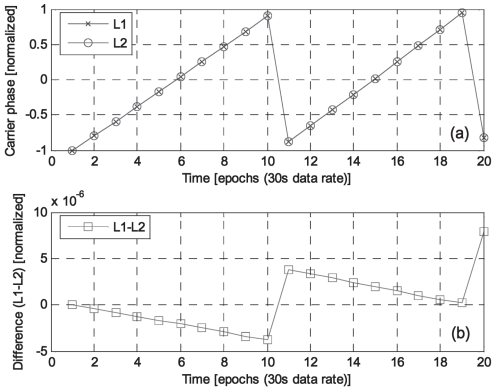

Fig. 7 Carrier phase measurements (L1 and L2) with pseudo slips (a) and the difference between normalized L1 and L2 (b).

The disadvantage of using this technique comes from the added slips themselves. For long time series, several slips have to be used, and for each subsequent series between slips, the errors of the estimations get accumulated when building a slip free estimated series. Figure 7b shows how small the differences between L1 and L2 are when the data is normalized. The neural model should be sensible to these variations. Implicitly, by minimizing the normalization factor these small differences are more easily detected when processed in a neural network.

Another technique that can be used is the time differencing of the observations. This approach reduces the range in which the phase counters vary. However, similarly to the previous case, the errors in the prediction get accumulated through time when rebuilding a slip free time series. This effect should be compensated by the much smaller value of the normalization factor. Depending on the type of series and generalization capability of the network it can be a good option. However, usually it does not perform very well for long time series. Figure 8 shows an example of the application of this technique.

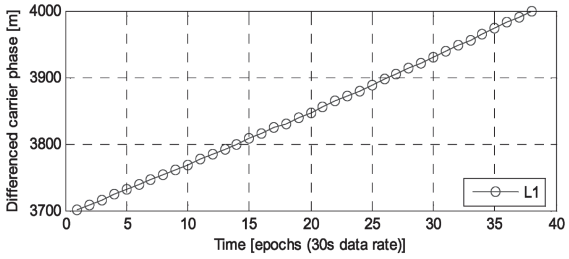

Fig. 8 Time differenced carrier phase.

It can be noticed that the range of the vales is much smaller than the one shown in Figure 6b, for example. Comparing the two cases, the normalization factor would be approximately 500 times smaller for this case. For instance, assuming the same MSE, the variance of the estimations would be much smaller in this last situation. This technique was used for the analysis mentioned at the beginning of this section. It demonstrated the best results in terms of carrier phase estimations.

Another technique that can be used is the inverse truncation procedure. It consists of truncating the number at a maximum size of significant numbers. It is called inverse, because in this technique instead of holding the left hand side of the number, it is held the right hand side. It is a very useful technique when one wants to relate two very similar quantities, not depending on their absolute values, but only up to a certain level. For instance, we wanted to relate C1 and P2 pseudoranges. Since we know that the difference between the two is below a hundred meters, we could hold the numbers just up to hundreds and assume that the rest of the values (the left hand side) should be always the same. Of course this procedure also causes a big decreasing in normalization factors. Figure 9 shows the results of this technique, applied to pseudoranges. Figure 9a shows the pseudoranges (C1 and P2) with their original values. Figure 9b shows the values after the application of the inverse truncation. The range of the values drop from the order of 10^7 to the order of 10^3. Figure 9c shows the difference of C1 and P2 values (after modified). In Figure 9d the left hand part of the pseudoranges (eliminated in the modification) is shown and Figure 9e shows the difference between these values (C1 – P2). It can be noticed from Figure 9e that the eliminated part of the numbers are always exactly the same for both observables, What maintains the assumption of coherence between patterns generated from different observables.

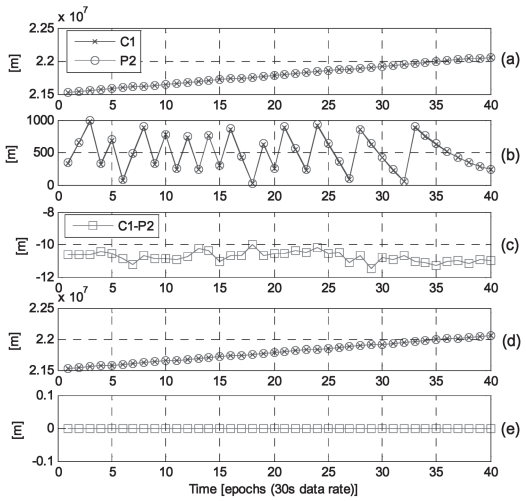

Fig. 9 Inverse truncation technique.

Other techniques than the ones shown in this work can also be used. One example of this is the functional reduction used by Leandro et. al. (2005) to modify pseudorange time series in order to feed stochastic models. This reduction consists of removing the mathematical distance and other systematic effects from the observations. In terms of neural processing this approach can be useful, however when in some cases it can bring some difficulties. For example for the sake of the coherence between patterns, coordinates of different receivers should be well known. In case one or more receivers have unknown position the technique could introduce biases in the reduced series, what can complicate the training process. This is a reason why this technique was not used in this specific case. However it can be useful in other situations (e.g. in applications dealing with only one receiver).

In terms of the data processing where all those techniques were applied, the best results were found when the time differencing technique was used for carrier phases and the inverse truncation was used for pseudoranges. Although these techniques performed better in this case, doesn't mean they are better than the other ones, but that they were more appropriate for the situation explored. In other cases, the other shown techniques might be more adequate.

5 Conclusions

Neural network models require data fitted into normalized intervals, what require modification in the original data set. In the process of modifying the data several techniques can be used. Some of the

techniques that can be useful for using neural models in geodesy, as well as their advantages and disadvantages, were show in this paper.

Definitively the choice of the adequate procedure when preparing the data to be used by neural models can produce large variations in quality of the estimations when brought back to their original range.

The techniques shown in this work are not all existent alternatives and other ones maybe even better than those might exist. The variety of solutions for this kind of problem can be as large as the possibilities when using neural model. However, an important contribution of this work is showing the importance of certain problems when dealing with data from geodetic measurements (such as GPS), as well as some useful solutions. If they are not applicable for a specific case, alternatives can arise from the concepts explored in them.

It is impossible to point which one of the shown techniques is the best one, because in each case a different solution can be the optimal. It was the case shown in this work, where the technique used for pseudoranges was different from the one used for carrier phase.

Although the data used in this work were GPS measurements, the procedures can be potentially applied to any type of measurement with same characteristics.

6 Acknowledgements

This search was partially funded by NSERC.

7 References

Chang, Y. M.; Chen, C. H.; Chen, C. S. (1996). Optimal Observation Design of Surveying Network using Artificial Neural Network. Geomatics Research Australasia, No.64, June, 1996, pp. 1-16.

Chansarkar, M. (1999). GPS Navigation using Neural Networks. 12th International Technical Meeting of the Satellite Division of the Institute of Navigation, September 14-17, 1999, Nashville Convention Center, Nashville, Tennessee.

Dumville, M. and Tsakiri, M. (1994). An Adaptive Filter for Land Navigation Using Neural Computing. 7th International Technical Meeting of The Satellite Division of The Institute of Navigation, September 20-23, 1994, Salt Palace Convention Center - Salt Lake City, UT.

Haykin, S. (1999). Neural Networks – A Comprehensive Foundation. Prentice Hall – Upper Saddle River, New Jersey.

Kuhar, M.; Stopar, B.; Turk, G.; Ambrozic, T. (2001). The use of artificial neural network in geoid surface approximation. Allgemeine Vermessungs-Nachrichten, Vol.108, No.1, 2001, pp. 22-27.

Leandro, R. F. (2004). A New Technique to TEC Regional Modeling using a Neural Network. ION GNSS 2004, September, 2004, Long Beach, California.

Leandro, R. F. and Santos, M. C. (2004). Comparison between autoregressive model and neural network for forecasting space environment parameters. Bollettino di Geodesia e Scienze Affini, Vol.63, No.3, 2004, pp. 197-212.

Maia, T.C.B., Silva C.A.U., Leandro R.F., Segantine P.C.L., Romero R.A.F. (2002). Predição da Contagem de Ciclos da Portadora GPS Utilizando uma Modelagem Conexionista Temporal – FIR MLP. XVI Brazilian Symposium on Neural Networks. Porto de Galinhas, Recife, Brazil.

Schuh, H.; Ulrich, M.; Egger, D.; Mueller, J.; Schwegmann, W. (2002). Prediction of Earth orientation parameters by artificial neural networks. Journal of Geodesy, Vol.76, No.5, 2002, pp. 247-258.

Vickery, J. L. and King, L. R. (2002). Use of Neural Networks and Expert Systems for Rapid Differential GPS Navigation. ION GPS 2002, September 24-27, 2002, Oregon Convention Center, Portland, Oregon.

Xenos, T. D. and Stergiou, D. C. (2002). One day before foF2 neural network based prediction models: A performance comparison between ordinary, fuzzy and recurrent neural networks. Acta Geodaetica et Geophysica Hungarica, Vol.37, No.2-3, 2002, pp. 293-296.