5-1995

Navigation Satellite Selection Using Neural Networks

Daniel J. Simon
Cleveland State University, d.j.simon@csuohio.edu

Hossny El-Sherief
TRW System Integration Group

Publisher's Statement

NOTICE: this is the author's version of a work that was accepted for publication in Neurocomputing. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Neurocomputing, 7, 3, (05-01-1995); 10.1016/0925-2312(94)00024-M

Original Citation

Dan Simon, Hossny El-Sherief. (1995) Navigation satellite selection using neural networks. Neurocomputing, 7(3), 247-258, doi: 10.1016/0925-2312(94)00024-M.

Repository Citation

Navigation satellite selection using neural networks

Dan Simon [a,*], Hossny El-Sherief [b]

[a] *TRW Test Laboratory, 4051 N. Higley Road, Mesa, AZ 85215, USA*
[b] *Manager, Guidance Systems and Control Department, TRW Systems Integration Group, Building SB2, Room 1051, PO Box 1310, San Bernardino, CA 92402, USA*

Abstract

The application of neural networks to optimal satellite subset selection for navigation use is discussed. The methods presented in this paper are general enough to be applicable regardless of how many satellite signals are being processed by the receiver. The optimal satellite subset is chosen by minimizing a quantity known as Geometric Dilution of Precision (GDOP), which is given by the trace of the inverse of the measurement matrix. An artificial neural network learns the functional relationships between the entries of a measurement matrix and the eigenvalues of its inverse, and thus generates GDOP without inverting a matrix. Simulation results are given, and the computational benefit of neural network-based satellite selection is discussed.

Keywords: Neural networks; Global Positioning System; Geometric dilution of precision; Approximation; Classification

1. Introduction

A Global Positioning System (GPS) receiver generates a user position and time by measuring the range from the user to four or more GPS satellites [8,9,3], but a GPS receiver can process only a subset of available satellite signals. For instance, there may be nine satellites visible, but the receiver hardware may be limited to processing no more than six satellites. So before processing, the receiver must decide which subset to use. The optimal choice can be made by using the subset which results in the smallest magnification of satellite errors onto resultant user

position and time. This magnification can be determined for each satellite subset by inverting a 4×4 matrix.

Phillips [16] presented a simple and geometrically intuitive approach to this problem under the assumption that the GPS receiver processes exactly four satellite signals, and the user is not concerned with obtaining an accurate time reference. He showed that the optimal satellite set is that set which minimizes a certain geometrical measurement of a tetrahedron formed by the four satellites and the user.

The approach taken in this paper is applicable to any number of satellite signals. It is based on the learning properties of artificial neurons, and as such gives an approximate rather than an exact answer. Its primary advantage lies in the fact that no matrix inversions are required. This translates into less required computational time for satellite subset selection, and the ability to begin navigating sooner with the best satellite subset.

2. Geometric dilution of precision (GDOP)

A user's GPS receiver measures a set of n ranges $(R_1, R_2, \ldots, R_n)$ between the user and n GPS satellites. The GPS satellites are at positions (x_i, y_i, z_i), $(i = 1, \ldots, n)$. The four unknowns which the user needs to determine are the offset T between receiver time and GPS time, and the user position (x, y, z). We denote the user's best estimate of time offset and position as $\hat{T}$ and $(\hat{x}, \hat{y}, \hat{z})$. We denote the corresponding best estimates of range as $(\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_n)$. The errors between the true and estimated quantities are denoted by

$$\Delta x = x - \hat{x} \tag{1}$$

$$\Delta y = y - \hat{y} \tag{2}$$

$$\Delta z = z - \hat{z} \tag{3}$$

$$\Delta T = T - \hat{T} \tag{4}$$

$$\Delta R_i = R_i - \hat{R}_i. \tag{5}$$

The errors of the user's estimate of time and position can be determined by solving the following n simultaneous nonlinear equations for Δx, Δy, Δz, and ΔT [10].

$$(\hat{x} + \Delta x - x_i)^2 + (\hat{y} + \Delta y - y_i)^2 + (\hat{z} + \Delta z - z_i)^2 = \left(\hat{R}_i + \Delta R_i - c\hat{T} - c\Delta T\right)^2 \tag{6}$$

$$(i = 1, \ldots, n)$$

where c is the speed of light. These equations can be linearized to obtain the matrix equation

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & 1 \\ a_{21} & a_{22} & a_{23} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & 1 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\Delta T \end{pmatrix} = \begin{pmatrix} \Delta R_1 \\ \Delta R_2 \\ \vdots \\ \Delta R_n \end{pmatrix}. \tag{7}$$

The $n \times 4$ matrix in Eq. 7 is called the *measurement matrix*, and its elements are given by

$$\begin{aligned} a_{i1} &= (\hat{x} - x_i)/\left(\hat{R}_i - c\hat{T}\right) \\ a_{i2} &= (\hat{y} - y_i)/\left(\hat{R}_i - c\hat{T}\right) \\ a_{i3} &= (\hat{z} - z_i)/\left(\hat{R}_i - c\hat{T}\right). \end{aligned} \tag{8}$$

Eq. 7 can be written more compactly as

$$A\vec{x} = \vec{r}. \tag{9}$$

The least-squares solution for $\vec{x}$ is given by [15]

$$\vec{x} = (A^T A)^{-1} A^T \vec{r}. \tag{10}$$

$A^T A$ is invertible if A has full rank. The uncertainty of the solution of user position and time is therefore related to the uncertainty of the measured ranges by follows.

$$\mathrm{cov}(\vec{x}) = (A^T A)^{-1} A^T \mathrm{cov}(\vec{r}) \left[(A^T A)^{-1} A^T\right]^T. \tag{11}$$

If the covariance of $\vec{r}$ is normalized to an identity matrix, we obtain a simplified expression for the covariance of user position and time as

$$\mathrm{cov}(\vec{r}) = I \Rightarrow \mathrm{cov}(\vec{x}) = (A^T A)^{-1}. \tag{12}$$

A useful scalar measure of the uncertainty of the solution of the user position and time is the trace of the above matrix. This quantifies the magnification of GPS range measurement errors (e.g. due to satellite and receiver inaccuracies) onto user position and time errors. GDOP is thus defined as

$$\mathrm{GDOP} = \left[\mathrm{trace}(A^T A)^{-1}\right]^{1/2}. \tag{13}$$

So GDOP can be calculated by inverting a 4×4 matrix. But how can GDOP be computed without resorting to matrix inversion? The way that humans learn most effectively is through induction, which is reasoning from the particular to the general. A computer algorithm can be designed to inductively generate a mathematical function by generalizing from known input/output relationships ([22], p. 182). Two ways of doing this are reviewed in the following sections.

3. Neural network-based approximation

The term *backpropagation* refers to a general learning rule which is implemented in an artificial neural network. Good overviews can be found in [1,13,19]. A typical backpropagation network has three layers of neurons – an input layer, a middle layer (also called a hidden layer), and an output layer. The outputs of the input layer neurons are weighted to activate the middle layer neurons, and the outputs of the middle layer neurons are weighted to activate the output layer

neurons. The effectiveness of backpropagation in learning complex, multidimensional functions can be partially explained by Kolmogorov's Theorem, which was extended to neural networks by Hecht-Nielson [1,5,7]. This theorem states that any functional $\mathscr{R}^m \rightarrow \mathscr{R}^n$ mapping can be exactly represented by a three-layer neural network with $(2m + 1)$ middle-layer neurons, assuming that the input components are normalized to lie in the range [0,1].

Knowing that GDOP is equal to the square root of the trace of $(A^TA)^{-1}$ (see Eq. 13), we will use the following general facts about the trace and eigenvalues of a matrix to compute GDOP.

(1) The trace of a matrix is equal to the sum of its eigenvalues ([6], p. 40). More specifically, the GDOP of a GPS navigation solution is equal to the square root of the sum of the eigenvalues of $(A^TA)^{-1}$, where A is defined in Eq. 7-9.
(2) If (A^TA) has eigenvalues λ_i then $(A^TA)^{-1}$ has the eigenvalues λ_i^{-1} ([15], p. 292).
(3) The determinant of a matrix is equal to the product of its eigenvalues ([4], p. 332).
(4) If (A^TA) has eigenvalues λ_i then $(A^TA)^k$ has the eigenvalues λ_i^k, where k is any positive integer ([6], p. 43).

Using $\vec{\lambda}$ to denote the four-element vector of the eigenvalues of A^TA, we can define the four functions

$$f_1(\vec{\lambda}) = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = \text{trace}(A^TA) \tag{14}$$

$$f_2(\vec{\lambda}) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 = \text{trace}\left[(A^TA)^2\right] \tag{15}$$

$$f_3(\vec{\lambda}) = \lambda_1^3 + \lambda_2^3 + \lambda_3^3 + \lambda_4^3 = \text{trace}\left[(A^TA)^3\right] \tag{16}$$

$$f_4(\vec{\lambda}) = \lambda_1\lambda_2\lambda_3\lambda_4 = \det(A^TA). \tag{17}$$

Using the above notation, the GDOP which we wish to calculate is precisely given by

$$\text{GDOP} = \left(\lambda_1^{-1} + \lambda_2^{-1} + \lambda_3^{-1} + \lambda_4^{-1}\right)^{1/2}. \tag{18}$$

So GDOP can be viewed as the scalar functional of the $\mathscr{R}^4 \rightarrow \mathscr{R}^4$ mapping $\vec{f}(\vec{\lambda})$

$$\text{GDOP} = \text{GDOP}\left[\vec{f}(\vec{\lambda})\right]. \tag{19}$$

The mapping from $\vec{f}(\vec{\lambda})$ to GDOP cannot be determined analytically. But this complex, nonlinear mapping is the type of problem at which neural networks excel. A neural network can be designed with four easily computable inputs (f_1, f_2, f_3, f_4), one hidden layer, and four outputs $(\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1})$. The outputs can then be summed to give the square of GDOP.

At this point it is of interest to investigate the solution space of Eq. 14–17. That is, given f_i $(i = 1, 2, 3, 4)$, is there a unique solution for λ_i $(i = 1, 2, 3, 4)$? We know that there is at least one solution. But if there is more than one solution, then a neural network may converge to the wrong solution and hence produce an

incorrect GDOP for a given satellite subset. The following lemma states that the solution to Eq. 14–17 is unique, and thus implies that a well-trained network will produce an accurate value for GDOP.

Lemma 1. *Consider two* 4×4 *matrices* A *and* B *such that* $\text{trace}(A) = \text{trace}(B)$, $\text{trace}(A^2) = \text{trace}(B^2)$, $\text{trace}(A^3) = \text{trace}(B^3)$, *and* $\det(A) = \det(B)$. *Then* A *and* B *have the same eigenvalues.*

Proof. See Appendix A. □

4. Neural-network based classification

Note that in picking a satellite subset to use for navigation, the receiver does not need to compute GDOP for every satellite subset. It only needs to find a subset which gives a satisfactorily low GDOP. So it can be argued that using backpropagation to find an approximating function to GDOP $[(\vec{f}(\vec{\lambda})]$ (see Eq. 19) entails more work than necessary. A more efficient approach is to create a network which classifies satellite subsets according to GDOP. If a network can be trained to classify a satellite group into one of n sets $(S_1, \ldots, S_n)$ according to GDOP, then those satellite groups which are classed in the best set are candidates for navigation use.

Classification may be the most popular application of neural networks. So it is not surprising that there are many different network architectures which have been proposed for classification. One recently proposed architecture is the Optimal Interpolative Net (OI Net) [20,21]. The OI Net is a three-layer classification network which grows during training according to how many neurons are necessary for correct classification. The efficient recursive learning formulation presented in [20] makes the OI Net an attractive architecture. In addition, fault tolerance can be implemented in OI Net training in a straightforward manner [17,18].

The OI Net can be used to select a good group of satellites with which to navigate by classifying groups according to GDOP. Given a group G of satellites, the OI Net can be trained to classify the group as

$$G \in S_i \quad \text{iff} \quad \text{GDOP}(G) < T_i \quad (i = 1, \ldots, n) \tag{20}$$

where $\text{GDOP}(G)$ is the GDOP of satellite group G, and T_i $(i = 1, \ldots, n)$ is some set of thresholds.

Now the issue becomes one of determining suitable thresholds T_i to use in the classification. If too many thresholds are used, it becomes difficult to find a diverse enough set of training samples, and the probability of misclassification increases. But if too few thresholds are used, the classification becomes too coarse, too many satellite groups are correctly classified in the best set, and there is less chance of selecting the best satellite group.

One method of solving the dilemma of how many classes to use is to implement multiple OI Nets. Each OI Net uses all of the training samples, and is trained to

classify a satellite group in one of two sets. Each OI Net uses a different threshold for classification. So we use n OI Nets, and the jth OI Net classifies a satellite group according to whether its GDOP is less than or greater than the threshold T_j. We will use the convention $T_1 < T_2 < \cdots < T_n$. A sequence of OI Nets running in series operates as follows.

(1) Set $j = 1$.
(2) If the jth OI Net classifies any of its inputs as being less than T_j, then any of the corresponding satellite groups can be used for navigation. Exit the loop.
(3) If the test in step (2) fails, i.e. if the jth OI Net classifies all of its inputs as being greater than T_j, then increment j by one and repeat step (2).

Note that if all n OI Nets classify the inputs as being greater than their respective thresholds, then all of the satellite groups have a GDOP greater than T_n. In this case, any satellite group can be used for navigation.

At this point it is of interest to inquire as to the optimal number of OI Nets to use. Fewer OI Nets means less operational complexity and less computational effort, but also results in a more coarse separation of classes. A coarse separation of classes means that there is less likelihood of selecting the single best satellite group for navigation use. So the choice of how many OI Nets to use is a trade-off between complexity and desired navigation accuracy.

In determining the optimal number of OI Nets to use, we will assume the following.

(1) We assume that the best GDOP is a random variable taken from a uniform distribution on $[T_0, T_{n+1}]$.
(2) We assume that the n OI Nets have thresholds $T_i (i = 1, \ldots, n)$ which are equally spaced by δT.
(3) We assume that the relative importance of operational complexity is W, while the relative importance of navigation accuracy is $1 - W$.

Under these assumptions we determine the optimal number of OI Nets as follows.

Theorem 1. *The optimal number n of OI Nets is given by*

$$n = \max\left[1, \operatorname{round}\left(\sqrt{(1-W)(T_{n+1}-T_0)/W} - 1\right)\right]. \tag{21}$$

Proof. See Appendix B. □

5. Simulation results

The two neural networks described in this paper were simulated on a VAX 8650 computer. Training took place for a GPS receiver located at 5000 feet above San Francisco (37.5 degrees latitude, 122 degrees longitude) in an 18-satellite constellation. Once each hour, for 12 hours, the measurement matrix (A) was generated for each visible four-satellite subset. The functions $f_i(\cdot)$ $(i = 1, 2, 3, 4)$ (see Eq. 14–17) were calculated, and λ_i^{-1} $(i = 1, 2, 3, 4)$ were calculated using the IMSL math

library. The function values f_i were normalized to the range [0.2, 0.8], saved in a training file, and then used to train the neural networks. If $f_4(\cdot)$ (the determinant of A^TA) was less then 0.12, the satellite set was immediately discarded from consideration. Such a low determinant can be shown by simulation to correspond to a GDOP too high for possible use. At each measurement time there were between five and seven visible satellites. There were thus between 15 and 35 four-satellite sets from which to choose the best set.

The networks were then tested on a simulated missile trajectory originating from Vandenberg Air Force Base in California. The powered portion of the flight lasted for 120 seconds, during which the missile travelled about 3.5° in longitude, 0.2° in latitude, and 150 miles in altitude. The trained neural networks were used to choose the best satellite group every two seconds. There were between five and seven satellites visible during the 120-second boost phase, and the satellite configuration with the best GDOP changed twice during that time (from a high of about 3.1 to a low of about 2.3).

5.1. Backpropagation simulation

Backpropagation was used to learn the correct network weights and external inputs so that the four $f_i(\vec{\lambda})$ inputs were mapped into the correct eigenvalue inverses. The backpropagation network had four input neurons corresponding to f_i ($i = 1, 2, 3, 4$), nine hidden layer neurons, and four output neurons corresponding to λ_i^{-1} ($i = 1, 2, 3, 4$) (see Eqs. 14–17). The network was trained with a learning rate of 0.9 and a momentum rate of 0.7. In order to achieve good training, we found it necessary to generate weight changes only for output errors above a specified error tolerance. In other words, any errors below the tolerance were considered perfect responses [2]. We used an initial tolerance of 50%, and decreased it as learning progressed. Backpropagation is a notoriously slow learning method, and it lived up to its reputation. The network converged to a minimum after about 23400 learning iterations (four hours and 47 minutes of VAX CPU time).

It can be easily shown from Eqs. 7–9 that trace $(A^TA) = 2n$, where n is the number of satellites processed by the receiver. So the input $f_1(\cdot)$ is a constant. This constant was still used as an input to the neural network, however, for the sake of generality. In addition, this constant input increased the size of the network, thus affording more flexibility in learning. The backpropagation net calculated a total of 1036 GDOPs during the simulated flight with a maximum error of 4.00% and an rms error of 1.44%.

5.2. Optimal interpolative net simulation

Several OI Nets were trained for several different thresholds of GDOP. Each OI Net had only three input neurons corresponding to (f_2, f_3, and f_4) (recall that f_1 is constant). Each OI Net had two output neurons. Ideally, a satellite group with a GDOP less than the threshold would have an output vector of [1,0] and a group

Table 1
Optimal interpolative net characteristics (150 training inputs)

GDOP threshold	Number of hidden neurons	Classification rate (%)
2.4	12	97.0
2.6	14	93.1
2.8	17	87.4
3.0	16	94.0
3.2	15	100
3.4	12	92.3

with a GDOP greater than the threshold would have an output vector of [0,1]. The number of hidden layer neurons varied from one OI Net to the next as determined by the learning algorithm. Each OI Net used a fitting parameter $\rho = 0.1$, an ill-conditioning threshold $\gamma_1 = 1e - 8$, and an error reduction threshold $\gamma_2 = 1e - 3$ (see Sin and deFigueiredo [20] for descriptions of these parameters). Each OI Net learned the training data in about one second of VAX CPU time. Table 1 shows the characteristics of several OI Nets which were trained and tested with the data described earlier in this section.

5.3. Comparison of backpropagation and OI Net learning

The most notable difference between backpropagation approximation and OI Net classification is the huge difference in training time. Backpropagation required about four hours and 47 minutes of VAX CPU time, while OI Net training required about one second of training time. So using an OI Net results in a savings of about 99.994% of the training time required for backpropagation. However, backpropagation gives a more accurate approximation than the OI Net. Backpropagation can be used for arbitrarily exact approximation, but the OI Net can only be used for either/or classification. The decision of which type of network to use must be made on the basis of the relative importance of training time versus exactness.

6. Conclusion

Two approximate methods of choosing an optimal subset of visible navigation satellites have been presented. The methods are based on the learning abilities of artificial neurons, and apply regardless of how many satellites the navigation system is tracking. A simulation study was shown to verify the applicability of the proposed approach. The results given in this paper clearly demonstrate the feasibility of neural network-based satellite selection.

The strength of using a neural net to determine GDOP is in its computational

efficiency. If a conventional LU-decomposition method [14] is used to invert a 4×4 matrix in order to determine GDOP, a total of 160 floating point operations are required for each matrix inversion. (A floating point operation is an add, subtraction, multiply, or divide.) If there are nine visible satellites and the user's receiver has the capability of processing signals from four or five of those satellites, there are a total of 126 possible satellite configurations from which to choose. This would require a total of $(126)(160) = 20160$ floating point operations in order to determine the best satellite set to use. The computational effort required could result in a short but noticeable delay before a navigation solution could be generated. This delay could be important for aircraft, missiles, or other platforms for which immediate navigation data are desired. An electronic implementation [12] of a neural network of the type described in this paper would be able to determine the best satellite configuration (in terms of GDOP) virtually instantaneously. This would enable the use of a high-integrity navigation solution without the delay required for many matrix inversions.

Of course, a GPS receiver designer and user wants some assurance that the GDOP computation is not only quick, but also accurate under all possible conditions. The universal approximation property of multilayer networks can be used to give some assurance of the accuracy of neural network based GDOP compuation. Consider the GDOP estimator as a machine which inputs the elements of the A^TA matrix and outputs GDOP. Since the eigenvalues of a matrix depend continuously on the elements of the matrix [6, Appendix D], and since GDOP is a continuous function of the eigenvalues of the A^TA matrix, we know that GDOP is a continuous function of the entries of the input matrix. We also know that there exists a feedforward neural network with one hidden layer which can approximate any given continuous function arbitrarily well [7]. Therefore, there exists a neural network with one hidden layer which can map the elements of the A^TA matrix to GDOP with arbitrarily good accuracy. Of course, the GDOP networks in this paper include the intermediate step of computing the determinant and traces of A^TA. Nevertheless, the universal approximation result of [7] gives us good reason to believe a properly trained GDOP network will give good results. As with any approximation method (neural network based or otherwise), extensive testing should be conducted to satisfy oneself of the accuracy of the approximation.

The simulation results presented in this paper show that the backpropagation network gives very accurate results at the expense of long training time. Therefore a less accurate interpolative classification network which can be trained more quickly was presented as an alternative. Of course, other architectures and training algorithms could also be used. In general, one of the key questions which a neural network engineer needs to answer is, 'Which network architecture and training algorithm should I use?' The contribution of this paper is to demonstrate the general feasibility of navigation satellite selection using neural networks. The question of how to choose a network architecture and training method for a particular problem is an important issue which is not addressed in this paper, but which deserves consideration, and which has been receiving increased attention in the neural network community [11].

Appendix A – Proof of Lemma 1

Consider two 4×4 matrices A and B such that $\text{trace}(A) = \text{trace}(B)$, $\text{trace}(A^2) = \text{trace}(B^2)$, $\text{trace}(A^3) = \text{trace}(B^3)$, and $\det(A) = \det(B)$. Denote the characteristic equations of A and B as

$$\Delta(A) = \lambda^4 + a_1\lambda^3 + a_2\lambda^2 + a_3\lambda + a_4 \tag{22}$$

$$\Delta(B) = \lambda^4 + b_1\lambda^3 + b_2\lambda^2 + b_3\lambda + b_4. \tag{23}$$

Then, using Fadeev's method ([15] p. 285), the a_k can be found using the recursion

$$P_{k+1} = AP_k + a_k I,\ P_1 = I \tag{24}$$

$$a_k = -\frac{1}{k}\,\text{trace}(AP_k) \tag{25}$$

and similarly for the b_k. This gives

$$a_1 = -\text{trace}(A) \tag{26}$$

$$a_2 = -\tfrac{1}{2}\text{trace}\left(A^2 + a_1 A\right) \tag{27}$$

$$a_3 = -\tfrac{1}{3}\text{trace}\left(A^3 + a_1 A^2 + a_2 A\right) \tag{28}$$

$$a_4 = \det(A) = -\tfrac{1}{4}\,\text{trace}\left(A^4 + a_1 A^3 + a_2 A^2 + a_3 A\right) \tag{29}$$

and similarly for the b_k. Using these expressions for the a_k and b_k, it can be shown using the premise of the lemma that $\Delta(A) = \Delta(B)$, and hence $\lambda(A) = \lambda(B)$. □

Appendix B – Proof of Theorem 1

We want to choose the number of OI Nets to minimize some combination of total computational effort and GDOP. More OI Nets results in more computational effort, but also results in finer separation between GDOP classes. The finer separation results in a stronger possibility of choosing the optimal satellite set.

Assume that the best GDOP is a random variable uniformly distributed on $[T_0, T_{n+1}]$. Assume that OI Net$_j$ classifies each input as having a GDOP either above or below threshold $T_j (j = 1, \ldots, n)$, and $T_{j+1} - T_j = \delta T$ $(j = 0, \ldots, n)$. Assume that the OI Nets are executed in sequential order. Let m be the number of OI Nets which must be exercised before a GDOP is classified as less than the corresponding threshold. That is, if GDOP_0 is the best available GDOP, then $T_0 < T_1 < \cdots < T_{m-1} < GDOP_0 < T_m < \cdots < T_{n+1}$. Since GDOP_0 is assumed to be a random variable uniformly distributed on $[T_0, T_{n+1}]$, m is a discrete random variable with probabilities $P(m = k) = (T_k - T_{k-1})/(T_{n+1} - T_0)$ $(k = 1, \ldots, n)$. Therefore the expected value of m is given by $\text{E}(m) = n/2$. Note that $m = k \Rightarrow T_m = T_k$ $(k = 1, \ldots, n-1)$. But if $m = n$ then either $T_m = T_n$ (if the mth OI Net classifies at least one GDOP less than T_n), or $T_m = T_{n+1}$ (if the mth OI Net

classifies all GDOPs greater than T_n). So T_m is a discrete random variable with probabilities $P(T_m = T_k) = 1/(n+1)$ $(k = 1, \ldots, n+1)$. Therefore the expected value of T_m is given by $\mathrm{E}(T_m) = T_0 + \delta \mathrm{T}(n+2)/2$. So the problem of finding an optimal number n of OI Nets can be stated as

$$\min_n \left[WE(m) + (1-W)E(T_m) \right] \tag{30}$$

where W is the relative importance of computational effort, and $1-W$ is the relative importance of navigation accuracy. The above problem reduces to

$$\min_n \left\{ Wn/2 + (1-W)\left[T_0 + \frac{(T_{n+1} - T_0)(n+2)}{2(n+1)} \right] \right\} \tag{31}$$

which has the solution

$$n = -1 \pm \sqrt{(1-W)(T_{n+1} - T_0)/W}\,. \tag{32}$$

Given the fact that n must be a positive integer, we obtain the result of Theorem 1. □

Acknowledgement

The proof of Lemma 1 was provided by Reid Reynolds of the TRW Space and Technology Group.

References

[1] M. Caudill, Neural networks primer: Part III, *AI Expert* (June 1988) 53–59.

[2] M. Caudill, Neural network training tips and techniques, *AI Expert* (Jan. 1991) 56–61.

[3] S. Gilbert, ed., *Global Positioning System, Vol. 3* (Institute of Navigation, Washington, DC, 1986).

[4] G. Golub and C. Van Loan, *Matrix Computations* (Johns Hopkins University Press, Baltimore, MD, 1990).

[5] R. Hecht-Nielson, Kolmogorov's mapping neural network existence theorem, *IEEE Conf. on Neural Networks*, San Diego, CA (1987) 11–14.

[6] R. Horn and C. Johnson, *Matrix Analysis* (Cambridge University Press, New York, NY, 1990).

[7] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward neural networks are universal approximators, *Neural Networks* 2 (5) (1989) 359–366.

[8] P. Janiczek, ed., *Global Positioning System, Vol. 1* (Institute of Navigation, Washington, DC, 1980).

[9] P. Janiczek, ed., *Global Positioning System, Vol. 2* (Institute of Navigation, Washington, DC, 1984).

[10] P. Jorgensen, Navstar/Global Positioning System 18-satellite constellations, in: P. Janiczek, ed., *Global Positioning System, Vol. 2* (The Institute of Navigation, Washington, DC, 1984) 1–12.

[11] J. McDonnell and D. Waagen, Evolving neural network connectivity, *Proc. IEEE Conf. on Neural Networks*, San Francisco, CA (1993) 863–868.

[12] C. Mead, *Analog VLSI and Neural Systems*, (Addison-Wesley, Reading, MA, 1989).

[13] P. Melsa, Neural networks: A conceptual overview, Report TRC-89-08, Tellabs Research Center, Mishawaka, IN, 1989.

[14] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, *Numerical Recipes* (Cambridge University Press, New York, NY, 1986).

[15] B. Noble and J. Daniel, *Applied Linear Algebra* (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[16] A. Phillips, Geometrical determination of PDOP, *Navigation: J. Inst. Navigation* (31) 4 (1984) 329–337.
[17] D. Simon and H. El-Sherief, A fault tolerant optimal interpolative net, *IEEE Conf. Neural Networks*, San Francisco, CA (1993) 825–830.
[18] D. Simon and H. El-Sherief, Fault tolerant training for optimal interpolative nets, *IEEE Trans. Neural Networks* (in print).
[19] P. Simpson, *Artificial Neural Systems* (Pergamon Press, New York, NY, 1990).
[20] S. Sin and R. deFigueiredo, An evolution-oriented learning algorithm for the optimal interpolative net, *IEEE Trans. Neural Networks* (3) 2 (1992) 315–323.
[21] S. Sin and R. deFigueiredo, Efficient learning procedures for optimal interpolative nets, *Neural Networks* (6) (1993) 99–113.
[22] C. Townsend and D. Feucht, *Designing and Programming Personal Expert Systems* (Tab Books, Blue Ridge Summit, PA, 1986).