GPS Solut (2009) 13:183–190
DOI 10.1007/s10291-008-0111-2

ORIGINAL ARTICLE

A closed-form formula for GPS GDOP computation

Shing H. Doong

Received: 12 July 2008 / Accepted: 5 November 2008 / Published online: 20 November 2008

Abstract Geometric dilution of precision (GDOP) is often used for selecting good satellites to meet the desired positioning precision. An efficient closed-form formula for GDOP has been developed when exactly four satellites are used. It has been proved that increasing the number of satellites for positioning will always reduce the GDOP. Since most GPS receivers today can receive signals from more than four satellites, it is desirable to compute GDOP efficiently for the general case. Previous studies have partially solved this problem with artificial neural network (ANN). Though ANN is a powerful function approximation technique, it needs costly training and the trained model may not be applicable to data deviating too much from the training data. Using Newton's identities from the theory of symmetric polynomials, this paper presents a simple closed-form formula for computing GDOP with the inputs used in previous studies. These inputs include traces of the measurement matrix and its second and third powers, and the determinant of the matrix.

Keywords GPS · GDOP · Symmetric polynomials · Newton's identities

Introduction

In GPS applications the dilution of precision (GDOP) is often used to select a subset of satellites from all visible ones. In order to determine the position of a receiver, pseudoranges from n (≥ 4) satellites must be used at the same time. By linearizing the pseudorange equation with Taylor's series expansion at the approximate (or nominal) receiver position, the relationship between pseudorange difference ($\Delta\rho_i$) and positioning difference (Δx_i) can be summarized as follows (Jwo 2001):

$$\begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ \vdots \\ \Delta\rho_n \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & e_{13} & 1 \\ e_{21} & e_{22} & e_{23} & 1 \\ e_{31} & e_{32} & e_{33} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ e_{n1} & e_{n2} & e_{n3} & 1 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_3 \\ c\Delta t \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} \tag{1}$$

which is written in a compact form as

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v} \tag{2}$$

The $n \times 4$ matrix $\mathbf{H}$ is formed with direction cosines e_{i1}, $e_{i2,}$ and e_{i3} from the receiver to the ith satellite, and $\mathbf{v}$ denotes a random noise with an expected value of $\mathbf{0}$. The difference between the estimated and true receiver positions is given by

$$\tilde{x} = (\mathbf{H}^{\mathbf{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathbf{T}}\mathbf{v} \tag{3}$$

where $\mathbf{H}^{\mathbf{T}}$ denotes the transpose of $\mathbf{H}$, and $\mathbf{M} = \mathbf{H}^{\mathbf{T}}\mathbf{H}$, called the measurement matrix, is a 4×4 matrix no matter how large n is. It can be shown that the measurement matrix is symmetric and nonnegative, and thus it has four nonnegative eigenvalues (Hoffman and Kunze 1961). Assuming $E\{\mathbf{v}\mathbf{v}^{\mathbf{T}}\} = \sigma^2\mathbf{I}$, then $E\{\tilde{x}\tilde{x}^{\mathbf{T}}\} = \sigma^2(\mathbf{H}^{\mathbf{T}}\mathbf{H})^{-1}$. The GDOP factor is defined as the square root of the trace of the inverse measurement matrix

$$\text{GDOP} = \sqrt{\text{trace}(\mathbf{M}^{-1})} = \sqrt{\frac{\text{trace}[\text{adj}(\mathbf{M})]}{\det(\mathbf{M})}} \tag{4}$$

Because GDOP provides a simple interpretation of how much positioning precision can be diluted by a unit of

S. H. Doong (✉)
Department of Information Management, ShuTe University,
59 Hengshan Rd, Yanchao, Kaohsiung County 824, Taiwan
e-mail: tungsh@mail.stu.edu.tw

measurement error, it is desirable to choose the combination of satellites in a satellite constellation with GDOP as small as possible. Yarlagadda et al. (2000) showed that increasing the number of satellites will always reduce the GDOP. Thus, it makes sense for a receiver to use all its available channels to receive signals from visible satellites. Suppose that a receiver has five channels to receive signals from nine visible satellites, then a total of 9!/(4!5!) = 126 GDOPs need to be computed in order to decide the best combination of satellites. This can present challenges to some real time GPS applications.

By using cofactors and determinant in Eq. 4, the computation of GDOP does have a closed-form solution in terms of elements of **M**, and thus **H**. However, this approach is often less efficient than well-designed numerical algorithms such as the *LU* decomposition. Using the conventional *LU* decomposition, it takes a total of 160 floating point operations (add, subtract, multiply and divide) to invert a 4 × 4 matrix (Atkinson 1978).

Due to limited resources associated with many handheld GPS devices, previous studies have tried to compute GDOP without resorting to the cofactors approach or matrix inversion. Simon and El-Sherief (1995) used a function approximation procedure rooted in artificial neural network (ANN) to compute GDOP. Extensive studies following this track of GDOP computation have been conducted elsewhere (Jwo and Chin 2002; Jwo and Lai 2003, 2007).

There are a few problems associated with the neural based solutions to the GDOP computation. First, a model must be trained with substantial computing resources. Second, it is well known in the machine learning field that the resultant model depends critically on the set of training data. Thus, when the GPS application is moved to a geographically different area, new training data must be collected to train a new model.

In this paper, we show that using the same features in Simon and El-Sherief (1995) or Jwo and Lai (2007), the GDOP function can be explicitly written down with a very simple formula. The methodology behind this formula is Netwon's identities in the theory of symmetric polynomials. This paper is organized as follows. In the next section, a literature review of GDOP computation is discussed. This is followed by the theory of symmetric polynomials and Netwon's identities. The closed-form formula is then derived from Newton's identities. The paper is ended with a few concluding remarks.

Literature review

In this section, a brief review of GDOP computation is presented. This review is focused on a closed-form solution from Zhu (1992) and the ANN based approach. Other possible solutions based on machine learning tools are also discussed.

GDOP estimation

Yarlagadda et al. (2000) reviewed the GDOP metrics with a goal to estimate bounds of GDOP. For example, when there are exactly four satellites in the measurement of pseudorange, then GDOP GDOP $\geq \sqrt{2}$, and

$$\mathrm{GDOP} \geq \frac{2}{(\det(\mathbf{M}))^{1/8}} \tag{5}$$

Another interesting and important result of Yarlagadda et al. (2000) is, in a satellite constellation, increasing the number of satellites for positioning tasks will always reduce the GDOP. Thus, a GPS receiver should use all its available channels to receive signals from visible satellites. This has created a practical need to find alternative means other than the Zhu (1992) closed-form formula, which is valid only for the four-satellite situation. Most GPS receivers today can receive signals from more than four satellites.

A closed-form formula for four satellites

By denoting

$$E_{ij} = e_{i1}e_{j1} + e_{i2}e_{j2} + e_{i3}e_{j3} + 1, \quad 1 \leq i < j \leq 4 \tag{6}$$

where e_{i1}, e_{i2} and e_{i3} are direction cosines in Eq. 1, and using the fact that $e_{i1}^2 + e_{i2}^2 + e_{i3}^2 = 1$, Zhu (1992) derived a closed-form formula for GDOP

$$\mathrm{GDOP} = \sqrt{\frac{16 + b + c}{a + b + 2c}} \tag{7}$$

where a, b and c are intermediate variables defined as follows:

$$a = (E_{12}E_{34} + E_{13}E_{24} - E_{14}E_{23})^2 - 4(E_{12}E_{34}E_{13}E_{24})$$
$$b = 16 - 4(E_{12}^2 + E_{13}^2 + E_{14}^2 + E_{23}^2 + E_{24}^2 + E_{34}^2)$$
$$c = 2[E_{12}(E_{13}E_{23} + E_{14}E_{24}) + E_{34}(E_{13}E_{14} + E_{23}E_{24})]$$

Starting from elements of **H**, it has been determined that 39 multiplications, 34 additions, 1 division and 1 square root are needed to compute GDOP in Eq. 7. This is less than half of the operation count needed to invert the measurement matrix **M**, let alone that **M** must be computed from $\mathbf{H^T H}$ first. Unfortunately, Zhu's formula is valid only when $n = 4$ in Eq. 1 since it uses a critical equality valid only for this dimension

$$\mathrm{trace}(\mathbf{H^T H})^{-1} = \mathrm{trace}(\mathbf{H H^T})^{-1} \tag{8}$$

As Yarlagadda et al. (2000) suggested more satellites than four can be used to decrease GDOP, efficient GDOP

computation for any number (≥4) of satellites is desirable. Jwo (2001) has extended Zhu's work by considering the situation of three satellites aided by an altimeter.

Neural network based approximations

An ANN mimics the working of a human brain in learning. A neural cell (neuron) is a fundamental information processing unit in ANN. Neurons are connected via synapses with weights that can be optimized to fit the training data. Two types of neural based approaches have been proposed to handle the GDOP problem: approximation and classification (Simon and El-Sherief 1995; Jwo and Lai 2007). In the approximation approach, the GDOP is predicted from a trained ANN model as a real number. The model is often assessed with the mean absolute error

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - t_i|$$

where y_i and t_i are the ANN computed GDOP and target GDOP respectively, or the root mean square error

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2}$$

The training time for this kind of function approximation by ANN is usually long; thus a different type of ANN is introduced for GDOP computation. The classification approach uses a threshold to cut the GDOP range into two classes: acceptable and unacceptable. For example, if GPS applications accept a GDOP as high as 2.5, then the threshold can be set to 2.5. Therefore, if a set of satellites yields a GDOP higher than this number, this set is labeled unacceptable; otherwise, the set is labeled acceptable. Due to the fact that only a binary decision is needed in this case, training a classification ANN is very fast. A classification ANN model is usually assessed with the precision, recall and accuracy rates that are commonly used in binary prediction problems (Witten and Frank 2005). The binary classification approach can be extended to a multi-class classification approach when there is a need to divide the GDOP range in a finer manner; for example, two threshold values may be used to label a set of satellites as excellent, acceptable and unacceptable.

The ANN approach is a supervised learning procedure and thus there are two phases of operations. In the training phase, data with known input–output pairings are fed into the network to learn synaptic weights. The goal of this learning is to minimize differences between network outputs and target outputs. After the network is trained, the procedure enters the operational phase where inputs can be fed into the network to compute estimated outputs. The effectiveness of ANN is founded on an extended Kolmogorov's theorem by Hecht-Nielson (1987), which says that any continuous functional mapping $R^m \to R^n$ can be approximated by a three-layer ANN with $(2m + 1)$ neurons in the middle layer.

Simon and El-Sherief (1995) started the ANN approach for GDOP approximation. In their study, GDOP is considered as a function of four independent variables as follows. Since the measurement matrix $\mathbf{M} = \mathbf{H}^{\mathbf{T}}\mathbf{H}$ is symmetric, it has four real valued eigenvalues $\lambda_1, \lambda_2, \lambda_3, \lambda_4$. Assuming that $\mathbf{M}$ is nonsingular, then the GDOP can be expressed as

$$\text{GDOP} = \text{trace}(\mathbf{M}^{-1}) = \sqrt{\lambda_1^{-1} + \lambda_2^{-1} + \lambda_3^{-1} + \lambda_4^{-1}} \tag{9}$$

because eigenvalues of the inverse matrix are inverses of eigenvalues of the original matrix and the trace of a symmetric matrix equals the sum of all its eigenvalues (Hoffman and Kunze 1961).

In order to estimate GDOP in Eq. 9, Simon and El-Sherief (1995) adopted a special set of features

$$h_1(\lambda) \equiv \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = \text{trace}(\mathbf{M}) \tag{10}$$

$$h_2(\lambda) \equiv \lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 = \text{trace}(\mathbf{M}^2) \tag{11}$$

$$h_3(\lambda) \equiv \lambda_1^3 + \lambda_2^3 + \lambda_3^3 + \lambda_4^3 = \text{trace}(\mathbf{M}^3) \tag{12}$$

$$h_4(\lambda) \equiv \lambda_1\lambda_2\lambda_3\lambda_4 = \det(\mathbf{M}) \tag{13}$$

The last term in each equation is used to compute the components of $\mathbf{h}$, i.e., $h_1 = \text{trace}(\mathbf{M})$, $h_2 = \text{trace}(\mathbf{M}^2)$, $h_3 = \text{trace}(\mathbf{M}^3)$, and $h_4 = \det(\mathbf{M})$. These equalities hold because $\mathbf{M}$ and its powers are symmetric matrices (Hoffman and Kunze 1961). Thus, the researchers considered a real valued function

$$f : R^4 \to R, \quad \mathbf{h} \mapsto \text{GDOP} \tag{14}$$

to be approximated by ANN. The training data were collected from a GPS receiver located at 5,000 ft above San Francisco (37.5° latitude, 122° longitude), and the trained model was tested on a simulated missile trajectory originating from Vandenberg air force base in California.

Using a back-propagation technique, the function $\text{GDOP} = f(\mathbf{h})$ of Eq. 14 was approximated by an ANN with nine middle layer neurons. It took the researchers 4 h and 47 min on a VAX machine of 1990s to train a satisfactory ANN. The RMSE was 1.44%. On the other hand, an optimal interpolative (OI) net was used to classify sets of satellites according to a threshold of the GDOP value. The threshold was set from 2.4 to 3.4 with an increment of 0.2, and the classification accuracy was high (>87.4%) in all cases. The training time for this OI net was about 1 s, which is a 99.994% of savings from the back-propagation neural network (BPNN) for function approximation.

Jwo and Chin (2002) expanded the study of Simon and El-Sherief (1995) by using three more input–output mappings. In addition to the mapping in Eq. 14, called type 3 mapping in Jwo and Chin (2002), the researchers also considered the following mappings:

$$\mathbf{h} \mapsto \lambda^{-1} = (\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1}) \quad \text{(type1 mapping)}$$

$$\mathbf{H^T H} \mapsto \lambda^{-1} = (\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1}) \quad \text{(type2 mapping)}$$

$$\mathbf{H^T H} \mapsto \text{GDOP} \quad \text{(type4 mapping)}$$

In type 2 and type 4 mappings, the input $\mathbf{H^T H}$ denotes elements of the measurement matrix. Since the measurement matrix is a symmetric 4×4 matrix, there are ten input variables for the type 2 and type 4 mappings. Incidentally, the middle layer had 21 neurons according to Hecht-Nielson (1987), and the resultant network was much more complicated than the one for the other two scalar valued mappings. Only BPNN was explored in Jwo and Chin (2002) to approximate the four continuous mappings from R^4 (using $\mathbf{h}$) or R^{10} (using $\mathbf{H^T H}$) to R^1 (using GDOP) or R^4 (using λ^{-1}). The training and testing data were gathered with a receiver placed on the top of the building of the author's institute (25.15° latitude, 121.78° longitude and 62 m altitude). For the purpose of a direct GDOP approximation, it was found that the type 3 mapping had an error mean of −0.0022 and a SD of 0.0236, and the type 4 mapping had an error mean of −0.0127 and a SD of 0.0174. Both results were considered very good by the researchers, but the type 4 mapping needed much more time to learn the patterns due to its complicated structure.

On the other hand, Jwo and Lai (2003) considered the type 3 mapping with different types of ANN to classify sets of satellites. A GDOP threshold of 2.5 was assumed, and three types of ANN were considered, namely, BPNN, OI nets and probabilistic neural nets (PNN). It was found that PNN had provided the best performance in terms of classification accuracy and training efficiency.

Similarly, Jwo and Lai (2007) considered the type 3 mapping with different neural nets to approximate or classify the GDOP function. The general regression neural networks (GRNN) and BPNN were used to approximate the function and both types of ANN were found to be feasible in this task. All four neural nets (BPNN, OI net, PNN and GRNN) were found to provide satisfactory results for GDOP classification.

Other possible solutions for GDOP computation

Phillips (1984) proposed to select the four satellites with a maximum volume of tetrahedron formed by the tips of the unit vectors from the receiver to the satellites, i.e., vertices with the direction cosines (e_{i1}, e_{i2}, e_{i3}), $i = 1, 2, 3, 4$. This is because GDOP is approximately proportional to the inverse of this volume. However, this maximum volume method may not select desired satellites with the minimum GDOP. There are also other machine learning techniques that can be used to learn the GDOP function as ANN above. These will be briefly discussed in the following.

Support vector regression

An ANN uses the empirical risk minimization (ERM) principle to learn patterns. On the other hand, support vector regression (SVR) proposed by Vapnik (Cristianini and Shawe-Taylor 2000) uses the structural risk minimization (SRM) principle to train a model. ERM minimizes risks resulting from the training data only; thus it may overfit a model. On the other hand, SRM incorporates the model complexity into a learning process and may avoid the overfitting problem encountered in many supervised learning algorithms. Support vector based techniques have been used to solve many scientific and engineering problems; therefore, the type 3 mapping of Jwo and Chin (2002) may be learnt with a SVR procedure.

Genetic programming

Genetic programming (GP), a biologically inspired machine learning technique, may help in finding the size and shape of a regression model (Koza 1992). GP uses evolutionary principles to evolve better individuals in successive generations. An individual in GP is a program tree with input variables such as h_1, h_2, h_3, and h_4 above or a random constant as leaf nodes and operators such as addition, subtraction, multiplication, and division, etc., as internal nodes. This program tree is evaluated for evolutionary purpose against a test environment. In the GDOP application, a test environment is formed by the input–output pairings of the training data. The purpose of GP is to find a program tree (i.e. a formula relating the input variables to the output variable) such that the discrepancy between computed outputs and target outputs is minimized. Like the ANN approach, a GP-based symbolic regression can be used to learn the GDOP function.

Symmetric polynomials

A symmetric polynomial is a polynomial $p(X_1, X_2, \ldots, X_n)$ in n variables such that when any two variables are interchanged, the polynomial remains the same. This can be more precisely defined as follows:

$$p(X_{\sigma(1)}, X_{\sigma(2)}, \ldots, X_{\sigma(n)}) = p(X_1, X_2, \ldots, X_n) \tag{15}$$

where σ is any permutation of the sequence 1, 2,…, n. Symmetric polynomials have many applications in

combinatorics, group representation, and quantum theory among others. There are many types of symmetric polynomials, of which two are of special interest to this study, namely the power sum symmetric polynomials and elementary symmetric polynomials.

Power sum symmetric polynomials

The power sum symmetric polynomial of degree k in variables $X_1, X_2,\ldots, X_n$ is the sum of all kth powers of the variables

$$p_k(X_1, X_2, \ldots, X_n) = X_1^k + X_2^k + \cdots + X_n^k \tag{16}$$

The power sum symmetric polynomials form a type of building blocks for symmetric polynomials in the sense that every symmetric polynomial with rational coefficients may be expressed as a sum and difference of products of power sum symmetric polynomials with rational coefficients (Macdonald 1995). For example, the symmetric polynomial $p(X_1, X_2) = X_1^2X_2 + X_1X_2^2 + X_1X_2$ can be expressed as

$$p(X_1, X_2) = \frac{1}{2}p_1^3 - \frac{1}{2}p_1p_2 + \frac{1}{2}p_1^2 - \frac{1}{2}p_2$$

where p_k is defined in Eq. 16 with $n = 2$.

Elementary symmetric polynomials

Elementary symmetric polynomials form another type of building blocks in the theory of symmetric polynomials. An elementary symmetric polynomial of a positive degree $k \le n$ in variables $X_1, X_2,\ldots, X_n$ is defined as follows:

$$e_k(X_1, X_2, \ldots, X_n) = \sum_{1 \le j1 < j2 < \cdots < jk \le n} X_{j1}X_{j2}, \ldots, X_{jk} \tag{17}$$

For example, the third degree elementary symmetric polynomial in four variables can be explicitly written down as

$$e_3(X_1, X_2, X_3, X_4) = X_1X_2X_3 + X_1X_2X_4 + X_1X_3X_4 + X_2X_3X_4 \tag{18}$$

When $k > n$, it is convenient to define $e_k(X_1, X_2,\ldots, X_n) = 0$. The 0th degree elementary symmetric polynomial is defined by $e_0(X_1, X_2,\ldots, X_n) = 1$. It can be proved that every symmetric polynomial, including power sum symmetric polynomials, can be expressed as the sum and product of constant and elementary symmetric polynomials (Macdonald 1995). On the other hand, since an elementary symmetric polynomial has rational coefficients 1, it can be expressed by the power sum symmetric polynomials. The close relationship between these two types of symmetric polynomials is further explained by Newton's identities.

Newton's identities

Newtown's identities are also called Newton–Girard formulae. These identities were found by Isaac Newton and Albert Girard in 1600s independently, and they relate power sum symmetric polynomials to elementary symmetric polynomials as follows:

$$ke_k(X_1, \ldots, X_n) = \sum_{i=1}^{k} (-1)^{i-1} e_{k-i}(X_1, \ldots, X_n)p_k(X_1, \ldots, X_n) \tag{19}$$

The formulae are valid for any integer $k \ge 1$. Each identity of a special k can be verified by direct algebraic manipulations, while the validity of all identities needs a proof. A simple proof using formal power series expansion can be found in Berlekamp (1968). Mead (1992) provides another simple and natural proof of Newton's identities.

The closed-form formula for GDOP

Studies in Simon and El-Sherief (1995), Jwo and Lai (2003) and Jwo and Lai (2007) pointed out that the type 1 mapping $\mathbf{h} \mapsto \boldsymbol{\lambda}^{-1}$ or type 3 mapping $\mathbf{h} \mapsto$ GDOP is highly nonlinear and cannot be determined analytically. Thus, ANN has been employed to approximate these mappings. The type 1 mapping remains to be studied in the future. The other three types of mapping will be analyzed in the following. A simple closed-form formula for the type 3 mapping can be derived from Newton's identities.

The type 2 mapping: $\mathbf{H^T H} \mapsto \lambda^{-1}$

Since the measurement matrix $\mathbf{M} = \mathbf{H^T H}$ is a 4×4 matrix, the characteristic polynomial

$$\mathrm{p}(\lambda) = \det(\lambda \mathbf{I} - \mathbf{M}) = 0$$

is a fourth degree polynomial in λ. There *is* a closed-form solution to a general fourth degree polynomial, though this *quartic* formula is not as popular or well-known as the quadratic formula for second degree polynomial equations. Lodovico Ferrari discovered this quartic formula in 1540s by using finitely many radicals (i.e. nth root) and arithmetic operations on the coefficients of a fourth degree polynomial. Therefore, λ_i^{-1} can be computed analytically by using a finite number of radicals and arithmetic operations on elements of $\mathbf{M}$. Indeed, $n = 4$ is the highest degree of a polynomial equation that has a general solution formula using only radicals and arithmetic. The third degree polynomial equation was solved by Niccolo Tartaglia. Using Galois theory, it can be shown that a general polynomial

equation of degree greater than 4 is not solvable in radicals and arithmetic (Goldstein 1973). That is, there is no general solution formula using only finitely many radicals and arithmetic operations for a polynomial equation with degree higher than 4.

The type 3 and type 4 mappings:
$\mathbf{h} \mapsto$ GDOP, $\mathbf{H^T H} \mapsto$GDOP

Fortunately, in terms of GDOP computation, embedded symmetry in the formula has avoided many messy computations in the quartic formula of Ferrari. A simple algebraic manipulation of GDOP shows that

$$\text{GDOP} = \sqrt{\frac{\lambda_1\lambda_2\lambda_3 + \lambda_1\lambda_2\lambda_4 + \lambda_1\lambda_3\lambda_4 + \lambda_2\lambda_3\lambda_4}{\lambda_1\lambda_2\lambda_3\lambda_4}} \tag{20}$$

The denominator inside the radical is $h_4(\boldsymbol{\lambda})$, and the numerator is the third degree elementary symmetric polynomial of Eq. 18. Using Netwon's identities successively for $k = 3$, 2 and 1, it follows that

$$e_3 = \frac{1}{3}\left[\frac{1}{2}(p_1^2 - p_2)p_1 - p_1p_2 + p_3\right]$$

Since $h_1(\boldsymbol{\lambda})$, $h_2(\boldsymbol{\lambda})$ and $h_3(\boldsymbol{\lambda})$ are first, second and third degree power sums of the eigenvalues, a close-form formula for computing GDOP is given by

$$\text{GDOP} = \sqrt{\frac{0.5h_1^3 - 1.5h_1h_2 + h_3}{3h_4}} \tag{21}$$

Because each $h_i(\boldsymbol{\lambda})$can be computed from elements of the measurement matrix, Eq. 21 also provides an exact formula for the type 4 mapping $\mathbf{H^T H} \mapsto$GDOP in Jwo and Chin (2002). The work in Simon and El-Sherief (1995) or Jwo and Lai (2007) has tried to approximate the formula in Eq. 21 by using various kinds of ANN. This formula is a simple combination of $h_i(\boldsymbol{\lambda})$; thus it may be found by GP described above. However, due to the specific coefficients involved, it will take GP substantial efforts to find this formula. A flowchart for the proposed GDOP computation is depicted in Fig. 1.

An example using the proposed formula to compute GDOP is now illustrated. Suppose five satellites are available to a receiver with the direction cosines as follows:

$$\mathrm{H} = \begin{pmatrix} 0.408248 & 0.816497 & 0.408248 & 1 \\ 0.666667 & 0.333333 & 0.666667 & 1 \\ -0.408248 & 0.816497 & 0.408248 & 1 \\ 0.816497 & -0.408248 & 0.408248 & 1 \\ 0.408248 & -0.816497 & 0.408248 & 1 \end{pmatrix}$$

The measurement matrix $\mathbf{M} = \mathbf{H^T H}$ is computed as a 4×4 matrix and given in the following.

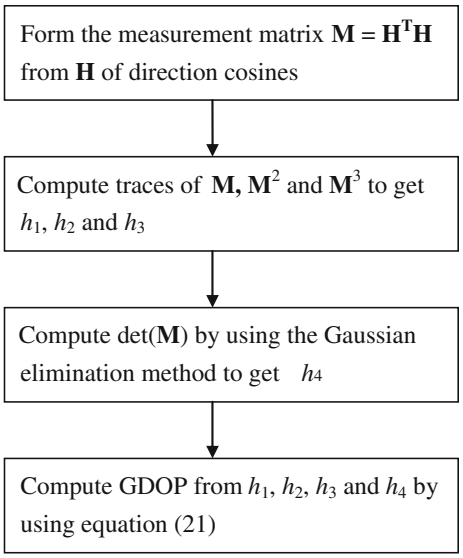

Fig. 1 GDOP computation using the proposed formula

$$\mathbf{M} = \begin{pmatrix} 1.61111 & -0.444444 & 0.944445 & 1.89141 \\ -0.444444 & 2.27778 & 0.388889 & 0.741582 \\ 0.944445 & 0.388889 & 1.11111 & 2.29966 \\ 1.89141 & 0.741582 & 2.29966 & 5 \end{pmatrix}$$

Computing the traces of $\mathbf{M}$, $\mathbf{M}^2$, and $\mathbf{M}^3$ yields $h_1 = 10$, $h_2 = 55.3316$ and $h_3 = 357.319$. The determinant of $\mathbf{M}$ gives $h_4 = 0.281962$, and according to Eq. 21, GDOP for this set of satellites is 5.68569.

Operation count and discussions

When there are exactly four satellites to compute GDOP, Zhu's formula (1992) provides the best computational efficiency. On the other hand, when there are more than four satellites to consider, one must first compute the matrix product $\mathbf{H^T H}$ to get the measurement matrix. This effort will not be considered in the following task of comparing operation count. The type 1 and type 3 mappings of Jwo and Chin (2002) require an intermediate step to compute h_i's from Eqs. 10–13. These computations need 144 floating operations divided as follows: three sums for h_1, 40 multiplications and 33 sums for h_2, 16 multiplications and 15 sums for h_3, and 23 multiplications and 14 sums for h_4. The determinant is computed by using the Gaussian elimination method (Atkinson 1978).

The presented formula

The presented formula in Eq. 21 needs six sums, two multiplications and one square root operation. Thus, the total operation count to compute GDOP from elements of $\mathbf{M}$ is 152 floating point operations and one radical operation.

Table 1 Operation counts to compute GDOP from elements of the measurement matrix

	The presented formula	Matrix inversion with *LU*	BPNN with type 3 mapping	BPNN with type 4 mapping
Floating	152	163	>244	>400
Radical	1	1	0	0
Exponent	0	0	10	22

The conventional LU decomposition for matrix inversion

Using the conventional *LU* decomposition to invert the measurement matrix requires 160 floating point operations (Atkinson 1978). Additionally, three sums are required to compute the trace of the inverse matrix, thus it takes a total of 163 floating point operations and one radical operation to compute GDOP, should this approach is adopted.

The ANN approach

For the ANN approach in Simon and El-Sherief (1995) or Jwo and Lai (2007), if type 3 mapping is used, then it needs a total of 144 floating point operations to compute intermediate variables. Additionally, with a BPNN structure of $4 \times 9 \times 1$, it needs more than 100 floating point operations to compute weighted sums and 10 exponential operations to compute the sigmoid function during the operational phase. On the other hand, if type 4 mapping is used, there is no need to compute the intermediate variables. However, a bigger network structure must be used to incorporate ten input variables now. Suppose a BPNN structure of $10 \times 21 \times 1$ is used, it will need more than 400 floating point operations to compute weighted sums and 22 exponential operations to compute the sigmoid function during the operational phase. The operation counts for different approaches are summarized in Table 1.

Conclusions

GPS applications use pseudoranges to locate the position of a receiver. The positioning accuracy is affected by measurement error amplified by GDOP. It is desirable to select a set of satellites with GDOP as small as possible. In real time GPS applications, fast computation of GDOP is needed to get the best performance from the system. Extensive studies using machine learning techniques have been conducted previously (Simon and El-Sherief 1995; Jwo and Lai 2003, 2007). Though these studies have shown that ANN provided a feasible means to solve the GDOP computation problem, there are a few questions left to be answered. First of all, machine learning techniques require computational resources to train a model, which needs to be retrained when substantially different geographic data are used. Second, machine learning techniques offer only approximations to the target values.

This study provides yet another method to compute GDOP without the labor associated with most machine learning techniques. Using Newton's identities in the theory of symmetric polynomials, a simple closed-form formula for GDOP was derived with the four features used previously (Simon and El-Sherief 1995; Jwo and Lai 2007). The comparison of operation count in Table 1 shows that this formula beats the direct matrix inversion method and neural based approaches in operational phase.

In contrast to the conventional approach, the closed-form formula does not need any training efforts, and nor does it incur any approximation error. The formula is a function of four real valued variables: traces of the measurement matrix $\mathbf{M}$, $\mathbf{M}^2$ and $\mathbf{M}^3$, and the determinant of $\mathbf{M}$. Using the embedded symmetry in the definition of GDOP and the theory of symmetric polynomials, a simple closed-form formula for GDOP has been derived. In addition to its practical applications in GDOP computation, the formula may also provide a means to estimate GDOP bounds in future studies.

Acknowledgments This work is supported in parts by a grant from the National Science Council of Taiwan under the contract number NSC95-2221-E-366-020-MY2. Computer time and facilities from the National Center for High-performance Computing (NCHC) of Taiwan are also appreciated. The author thanks the anonymous reviewers for their valuable comments. Helpful discussions with Dr. Chih-Hung Wu on the subject are appreciated.

References

Atkinson KE (1978) An introduction to numerical analysis. Wiley, New York

Berlekamp E (1968) Algebraic coding theory. McGraw-Hill, New York

Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge

Goldstein LJ (1973) Abstract algebra: a first course. Prentice-Hall, Englewood Cliffs

Hecht-Nielson R (1987) Kolmogorov's mapping neural network existence theorem. In: IEEE conference on neural networks, pp 11–14

Hoffman K, Kunze RA (1961) Linear algebra. Prentice-Hall, Englewood Cliffs

Jwo DJ (2001) Efficient DOP calculation for GPS with and without altimeter aiding. J Navig 54(2):269–279. doi:10.1017/S0373463301001321

Jwo DJ, Chin KP (2002) Applying back-propagation neural networks to GDOP approximation. J Navig 55(1):97–108. doi:10.1017/S0373463301001606

Jwo DJ, Lai CC (2003) Neural network-based geometry classification for navigation satellite selection. J Navig 56:291–304. doi:10.1017/S0373463303002200

Jwo DJ, Lai CC (2007) Neural network-based GPS GDOP approximation and classification. GPS Solut 11:51–60. doi:10.1007/s10291-006-0030-z

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection (complex adaptive systems). MIT Press, Cambridge

Macdonald IG (1995) Symmetric functions and Hall polynomials. Clarendon Press, Oxford

Mead DG (1992) Newton's identities. Am Math Mon 99(8):749–751. doi:10.2307/2324242

Phillips A (1984) Geometrical determination of PDOP. Navig J Inst Navig 31(4):329–337

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

Witten IH, Frank E (2005) Data mining, practical machine learning tools and techniques. Morgan Kauffman, San Francisco

Yarlagadda R, Ali I, Al-Dhahir N, Hershey J (2000) GPS GDOP metric. IEE Proc Radar Sonar Navig 147(5):259–264. doi:10.1049/ip-rsn:20000554

Zhu J (1992) Calculation of geometric dilution of precision. IEEE Trans Aerosp Elect Syst 28(3):893–894. doi:10.1109/7.256323