GEODESY AND CARTOGRAPHY
ISSN 2029-6991 / eISSN 2029-7009
2015 VOLUME 41(3): 113–118
http://dx.doi.org/10.3846/20296991.2015.1086118

UDK 528.14

PRACTICAL METHOD TO SOLVE LARGE LEAST SQUARES PROBLEMS USING CHOLESKY DECOMPOSITION

Ghadi YOUNIS

Surveying & Geomatics Engineering, Palestine Polytechnic University, Wad-Alharia, Hebron, Palestine
E-mail: ghadi@ppu.edu

Received 13 April 2015; accepted 15 August 2015

Abstract. In Geomatics, the method of least squares is commonly used to solve the systems of observation equations for a given number of unknowns. This method is basically implemented in case of having number observations larger than the number of unknowns. Implementing the large least squares problems would require a large storage on the hard drive to store the different matrices for applying the solution. The computational time for solution would extremely increase with increasing number of unknowns and observations. The calculation of the inverse of the normal equation matrix will get more complex using the traditional methods with higher numbers of unknowns. Here, practical methods to eliminate the required storage and computations times during the solution are introduced. The Cholesky decomposition will be used to solve the systems of equations in order to avoid the complexity of the matrix inversion and to guarantee faster solutions. A block matrix implementation of Cholesky decomposition is to be used to enable the management of the memory and its limitations through the solutions. The principle of threading, which is supported in most of the programming languages like C++ or Java, is implemented to use the computer resources especially all available central processing units (CPU). This principle can be implemented over networks of computers to use of the resources of more available computers working under common servers.

Keywords: systems of equations, least squares solutions, matrix, matrix inverse, Cholesky decomposition, block matrix, normal equations matrix, threading, parallel processing.

Introduction to least squares

In surveying and Geomatics, observations must often satisfy established numerical relationships known as geometric constraints. As examples in a closed polygon traverse, horizontal angle and distance measurements should conform to the geometric constraints, and in a differential leveling loop, the elevation differences should sum to a given quantity. However, because the geometric constraints meet perfectly rarely, if ever, the data are adjusted. The fundamental principle of a least squares adjustment is the most common method for adjusting the observations in the Geomatics branches like surveying, GNSS, Geodesy and Photogrammetry. The principle of least squares takes the advantage of the redundancies in observation for computing the unknowns. The basic principle of least squares method is to minimize the summation of the squared residuals in the observations (Ghilani, Wolf 2008).

$$\sum v^2 = v_1^2 + v_2^2 + \ldots + v_n^2 = \min . \qquad (1)$$

The more general case of least squares adjustment assumes that the observations have varying degrees of precision, and thus varying weights. Then the weighted least squares principle reads:

$$\sum wv^2 = w_1 v_1^2 + w_2 v_2^2 + \ldots + w_n v_n^2 = \min . \qquad (2)$$

For an observation l_i with a residual v_i, the observation equation reads:

$$l_i + v_i = \boldsymbol{A}_i \hat{\boldsymbol{x}} . \qquad (3)$$

The row vector $\boldsymbol{A}_i$ is a row in the design matrix $\boldsymbol{A}$. The elements of $\boldsymbol{A}_i$ are the coefficients of the unknown parameters in an observation equation. The least squares solution for the system of linear equations according to Equation (3) is given in a matrix form as:

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{l} . \qquad (4)$$

In Equation (4), the square matrix $\boldsymbol{W}_{ll} = \sigma_0^2 \boldsymbol{C}_{ll}^{-1}$ is the weight matrix of the observations using their covariance matrix $\boldsymbol{C}_{ll}$ (Jäger *et al.* 2005). σ_0^2 is the

assumed reference variance of the solution before the adjustment. is the column vector of the observations. Using the solved unknowns in the vector matrix $\boldsymbol{x}$ of equation (4), the residuals vector $\boldsymbol{v}$ of the observations reads:

$$\boldsymbol{v} = \boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{l}\,. \tag{5}$$

Using the calculated residuals, the calculated reference variance of the adjustment with r redundant observations reads:

$$\hat{s}_0^2 = \frac{\boldsymbol{v}^T \boldsymbol{W}_{ll} \boldsymbol{v}}{r}\,. \tag{6}$$

Using the laws of error propagation, the covariance matrix of the unknowns $\boldsymbol{C}_{xx}$ reads (Niemeir 2002):

$$\boldsymbol{C}_{xx} = \hat{s}_0^2 (\boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{A})^{-1} = \hat{\boldsymbol{s}}_{\boldsymbol{0}}^{\boldsymbol{2}} \boldsymbol{N}^{-1} = \hat{\boldsymbol{s}}_{\boldsymbol{0}}^{\boldsymbol{2}} \boldsymbol{Q}_{xx}\,. \tag{7}$$

The covariance matrix $\boldsymbol{C}_{\hat{l}\hat{l}}$ of the adjusted observations is given by:

$$\boldsymbol{C}_{\hat{l}\hat{l}} = \hat{s}_0^2\, \boldsymbol{A}(\boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{A})^{-1}\, \boldsymbol{A}^T = \hat{s}_0^2\, \boldsymbol{A}\boldsymbol{N}^{-1}\boldsymbol{A}^T = \hat{s}_0^2\, \boldsymbol{Q}_{\hat{l}\hat{l}}\,. \tag{8}$$

1. Memory optimization

The solution of a least squares problem using a system of linear equations given in equation (4) can be written as (Ghilani, Wolf 2006):

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{A})^{-1}\, \boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{l} = \boldsymbol{N}^{-1} \boldsymbol{c}\,. \tag{9}$$

Here, $\boldsymbol{N}$ is the normal equations matrix, and $\boldsymbol{c}$ is the constants matrix. $\boldsymbol{N}$ and $\boldsymbol{c}$ read:

$$\boldsymbol{N} = (\boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{A})\,; \tag{10}$$

$$\boldsymbol{c} = \boldsymbol{A}^T \boldsymbol{W}_{ll} \boldsymbol{l}\,. \tag{11}$$

In the case of a large number of observations and unknowns, significant memory would be required to store the matrices $\boldsymbol{A}$, $\boldsymbol{W}$, $\boldsymbol{N}$ and $\boldsymbol{l}$. The Matrix $\boldsymbol{N}$ and the vector matrix $\boldsymbol{c}$ can be calculated directly using the observations and their weights (Jäger *et al.* 2005). In this way, the size required to store the matrices A will be reduced to only one row of the matrix to be stored temporarily during the computations. As the normal matrix $\boldsymbol{N}$ is symmetric, only the upper or the lower part of the $\boldsymbol{N}$ matrix must be saved. The calculations for the elements of the upper part of $\boldsymbol{N}$ and the elements of the $\boldsymbol{c}$ vector using only one row of the design matrix A related to the i-th observation read (Niemeier 2002):

$$n_{m,n} = n_{m,n} + a_m a_n w_i\,; \tag{12}$$

$$c_m = c_m + a_m w_i l_i\,. \tag{13}$$

In the case of a very large number of unknowns, memory limits may stop the calculations during the solution of Equation (10). To avoid this problem, the matrix $\boldsymbol{N}$ is divided into sub-matrices (blocks). Only one or a limited number of blocks are then loaded in the memory (Smith 2001). The other blocks are stored in the hard drive in ASCII or Binary files. Afterwards, they are loaded when required. The block matrix form of the normal equations matrix $\boldsymbol{N}$ reads (Okrah 2005):

$$\boldsymbol{N} = \begin{bmatrix} \boldsymbol{N}_{11} & \boldsymbol{N}_{12} & \boldsymbol{N}_{13} & \cdots & \boldsymbol{N}_{1m} \\ & \boldsymbol{N}_{22} & \boldsymbol{N}_{23} & \cdots & \boldsymbol{N}_{2m} \\ & & \boldsymbol{N}_{33} & \cdots & N_{3m} \\ & & & \ddots & \vdots \\ & & & & \boldsymbol{N}_{mm} \end{bmatrix}. \tag{14}$$

2. Cholesky decomposition

Cholesky decomposition is a useful method that can be used for efficient numerical solutions. It is basically used to solve a system of linear equations (Press *et al.* 1992). It is commonly used for Geomatics applications for applying the least squares solutions. Other applications in the field of electronics were introduced by (Aquilante *et al.* 2011). Other application is the in the reconstruction of 3D scenes in photogrammetry using 2D images (Hartley *et al.* 1992). The Cholesky decomposition of the covariance matrices was used to decrease the computational times of a two-sample procedure for testing the equality of the generalized Frobenius means of two independent populations on the space of symmetric positive matrices (Osborne *et al.* 2013).

The calculation time required to find the solution to a system of linear Equations (4) can be reduced by implementing the Cholesky decomposition, in which a positive symmetric definite matrix $\boldsymbol{N}$ can be represented by a lower triangle $\boldsymbol{L}$ matrix or an upper triangle matrix $\boldsymbol{U}$ multiplied by its transpose (Rothberg, Gupta 1994).

$$\boldsymbol{N} = \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{U}^T \boldsymbol{U}\,; \tag{15a}$$

$$\boldsymbol{L} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}; \tag{15b}$$

$$\boldsymbol{U} = \begin{bmatrix} U_{11} & U_{11} & \cdots & U_{1n} \\ 0 & U_{22} & \cdots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & U_{nn} \end{bmatrix}. \tag{15c}$$

The factorization of the lower triangle matrix $\boldsymbol{L}$ according to the Cholesky decomposition reads:

$$l_{jj} = \sqrt{n_{jj} - \sum_{i=1}^{j-1} l_{ji}^2}\ ; \tag{16}$$

$$l_{vj} = \left(n_{vj} - \sum_{i=1}^{j-1} l_{ji} l_{vi} \right) \Big/ l_{jj}\ . \tag{17}$$

By having the factorized $\boldsymbol{L}$ matrix, the solution with respect to the unknowns can be obtained without the need to invert the normal equations matrix $\boldsymbol{N}$. The matrix form of the linear system of equations according to the Cholesky decomposition reads:

$$\boldsymbol{Nx = LL^T x = c}\ . \tag{18}$$

Equation (18) can be rewritten as follows (Ghilani, Wolf 2006):

$$\boldsymbol{Ly = c}\ ; \tag{19}$$

$$\boldsymbol{L^T y = x}\ . \tag{20}$$

This makes it easy to solve the system of equations without the need to invert the matrix $\boldsymbol{L}$ or $\boldsymbol{L^T}$. This is because $\boldsymbol{L}$ is a lower triangle matrix (Aledeld, Mayer 1993). Solving the system of equations $\boldsymbol{Ly = c}$ with respect to y is called forward substitution. The elements of the vector matrix y are (Press *et al.* 2002):

$$y_j = \left(c_j - \sum_{i=1}^{j-1} l_{ji} y_i \right) \Big/ l_{jj}\ . \tag{21}$$

The solution of the system of equations $\boldsymbol{L^T y = x}$ with respect to the unknowns vector x is called backward substitution. The values of the elements of the unknowns' vector x are:

$$x_j = \left(y_j - \sum_{i=j+1}^{n} l_{ij} x_i \right) \Big/ l_{jj}\ . \tag{22}$$

One concern with the Cholesky decomposition facrorizations in Equations (16) and (17), the square roots are using in the factorization of the Cholesky decomposition. The results are always position in the well-conditioned solutions. Where the matrices are positive definitive. In many applications the matrices are ill-conditioned (positive semi-definitive matrix). The numbers under the square roots can get negative by the round off errors. Adding a diagonal correction to the matrices can solve the problem. But the accuracy of the solution will be less (Fang, O'Leary 2006).

3. Cholesky block matrix decomposition

The implementation of the block matrix Cholesky decomposition is applied similarly to the normal Cholesky decomposition. Each block is handled as an element of a matrix (Nool 1992). The factorization of the block lower triangle matrices reads (Schaefer 2003):

$$\boldsymbol{L_{jj}} = Decomp\left(\boldsymbol{N_{jj}} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{j-1}} \boldsymbol{L_{ji} L_{ji}^T} \right); \tag{23}$$

$$\boldsymbol{L_{l\,j}} = \left[Bforsub\left(\boldsymbol{L_{jj}}\ , \boldsymbol{N_{jl}} - \sum_{i=1}^{j-1} \boldsymbol{L_{ji} L_{li}^T} \right) \right]^{\boldsymbol{T}}\ . \tag{24}$$

In Equation (23), the term $Decomp(\boldsymbol{A})$ is the application of Cholesky factorization of Matrix $\boldsymbol{A}$ using Equations (16) and (17). The term $Bforsub(\boldsymbol{A,B})$ is calculated by using the forward substitution in Equation (21) of the matrix $\boldsymbol{A}$ and the columns of $\boldsymbol{B}$ in Equation (25). Having $\boldsymbol{B_i}$ as the i-th column of matrix $\boldsymbol{B}$, $Bforsub(\boldsymbol{A,B})$ reads:

$$Bforsub(\boldsymbol{A,B}) = \\ (Forsub(\boldsymbol{A,B_1}), Forsub(\boldsymbol{A,B_2}), Forsub(\boldsymbol{A,B_3})...). \tag{25}$$

The forward substitution to find the block element $\boldsymbol{Y_j}$ of the vector y using the block elements $\boldsymbol{C_j}$ of the vector c reads:

$$\boldsymbol{Y_j} = Forsub\left(\boldsymbol{L_{jj}}\ , \boldsymbol{C_j} - \sum_{i=1}^{j-1} \boldsymbol{L_{ji} Y_i} \right). \tag{26}$$

The backward substitution to find the block element $\boldsymbol{X_j}$ of the unknowns' vector x using the block elements $\boldsymbol{Y_j}$ of the vector $\boldsymbol{y}$ reads:

$$\boldsymbol{X_j} = Backsub\left(\boldsymbol{L_{jj}^T}\ , \boldsymbol{Y_j} - \sum_{i=j+1}^{k} \boldsymbol{L_{ij}^T X_i} \right). \tag{27}$$

The function $Backsub(\ )$ is the implementation of backward substitution in Equation (22).

4. Parallel processing

In the modern computers, many processors are available (e.g. duo processor, quad processor, i3, i5, i7). Classical programming using the programming languages (e.g. VC++, VB, Java ... etc.) use only one processor at a time. Most of these languages support the principle of threading. Here, the functions in the program are run multiple times in parallel. Each time, a function uses new input and output values. Implementing the principle of threading enables the use of all processors or a customized number of processors (Schiebl 1999) (see Fig. 1). The principle of

calculating the normal matrix in a block matrix form directly from the observations enables application of threading principles (Breymann 2005). This means that many blocks can be calculated in simultaneously from the same observations. While each block reserving a single processor partially or totally (Nool 2001). The use of the parallel processing principle enables the use of high performance computers that has multiple CPUs. In addition the calculations can be applied over servers and networks. This makes use of more computers to apply the calculations. In this case, the observations are stored on the server drive. Where all computers over the network can read them and use them to calculate specific parts of the normal equations matrix $\boldsymbol{N}$.

5. Results and analysis

The calculations were applied over different numbers of unknowns. First test was to solve a system of 16,000 equations with 10,000 unknowns. The calculations of the unknowns using direct solution given in Equation (9) had a computations time of approximately 16 in single processing unit. The calculation required 320 KB of storage for the unknowns matrix $\boldsymbol{x}$ in a binary format. While the matrices $\boldsymbol{A}$, $\boldsymbol{A^T}$, $\boldsymbol{N}$, $\boldsymbol{l}$ and $W_{diagonal}$ required approximately 154 MB, 154 MB, 100 MB, 511 KB, and 511 KB. The total required RAM and storage memory was approximately 420 MB. When memory optimization in Equations (12) to (14) was applied by dividing the matrix $\boldsymbol{N}$ for (10×10) blocks as given in Equation (14), only the upper part of the matrix $\boldsymbol{N}$ was stored using 92 MB and the $\boldsymbol{c}$-matrix using 324 KB with approximately 4 MB per block stored as a separate file. While only 11 MB were need for the RAM to apply the solutions in approximately 11 hours. In the next step, the Cholesky decomposition was applied to solve the problem in a single matrix form as shown in Equations (15) to (22) and in a block matrix using the principle given in Equations (23) to (27). The solution time has decreased dramatically to 4 and 2.5 hours, respectively. Finally, the solution was applied using block matrix Cholesky decomposition applying the principle of parallel processing over 4 CPUs. The calculations time was reduced to approximately 43 minutes.

A final test was applied to solve a system of linear equations to calculate the coefficients of a local potential/geoid model using the principle of Adjusted Spherical Cap Harmonics (ASCH). Where the observation equation reads (Younis 2015):

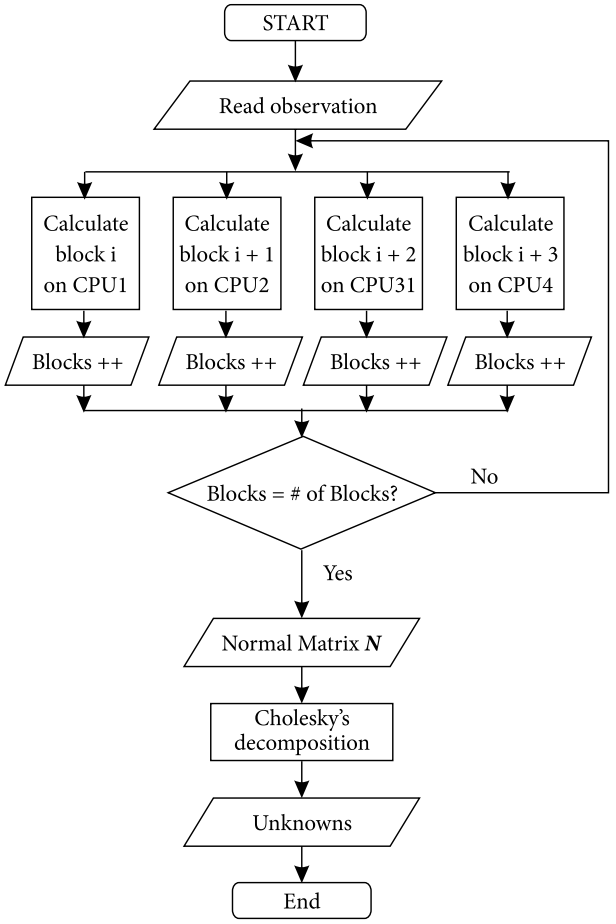

Fig. 1. Flowchart for calculating unknowns using parallel processing

$$V(r,\alpha,\vartheta)=\frac{GM}{r}\sum_{k=0}^{k\max}\left(\frac{R}{r}\right)^{n(k)}\cdot$$
$$\sum_{m=0}^{k}\left(C'_{km}\cos m\alpha+S'_{km}\sin m\alpha\right)\overline{P}_{km}(\cos\vartheta). \quad (28)$$

In equation (28), $\overline{P}_{nm}(\cos\vartheta)$ is the fully normalized Legendre function of integer order k and degree m. (α,θ,r) are the scaled spherical cap coordinates (Franceschi, De Santis 1994). (S'_{nm},C'_{nm}) are the ASCH coefficients. Here, we have to consider that having a maximum degree and order n of the model will have $(n+1)^2$ coefficients (S'_{nm},C'_{nm}) (Younis 2013). The final solution was applied with a maximum degree and order of 300. The number of unknowns/coefficients (S'_{nm},C'_{nm}) were 90,601. The observation were 105 height fitting points, 15,000 terrestrial gravity points and the ASCH coefficients of a locally transformed global model (Younis *et al.* 2011). The total number of observations was 105 706. The normal equations matrix was divided to 1000×1000 blocks in total size for the upper triangular part of approximately 320 GB. The observations were stored on server folder enabling 40 computers with quad processor over the

network to access the observations and solve for specific parts of the matrix $\boldsymbol{N}$. A total time for the computations was around 36 hours. This method have been successfully implemented in the Digital Finite Elements Height Reference Surface – Software version 5.0 (http://www.dfhbf.de/software.v5.php) for the computation of the height reference surface using combined observations (height fitting points, terrestrial gravity data, deflections of vertical, etc.) (Younis 2013).

Conclusions

The principle of Cholesky decomposition could efficiently be implemented successfully to apply least squares solutions. The use of Cholesky decomposition has enabled faster solutions compared to traditional methods. One basic point that makes difference is that Cholesky decompositions solves for the unknowns without the need to apply the inverse of a matrix. Also it was possible to calculate the $\boldsymbol{N}$ and $\boldsymbol{c}$ matrices in Equation (11) directly from the observations without the need to store all matrices of the traditional method of the solution like $\boldsymbol{A}$, $\boldsymbol{A^T}$ and $\boldsymbol{W}$ in Equation (4). The principle of matrix partition as explained in Equations (12) to (14) has enabled the use personal or older computer in solving large system equations. This enabled slow/normal computers with less RAM memory to load the matrices during the computations partially and not as a whole unit. This was necessary for many types of Geomatics problems. For example, the calculation of geoid model coefficients or the transformation parameters between coordinates systems.

The principle of threading enabled the use of multiple CPUs in parallel on a single computer. For example, a computer with Quad-processors will implement the computations by using the methods of threading approximately 4 times faster the implementing the computations over a single CPU. Here, we get used of the advantages of matrix partition to apply the Cholesky block matrix decomposition over more CPU. This principle was successfully applied over networks. The project observations were stored over a server drive. Each computer over the networks runs for a specific part of the normal equations matrix $\boldsymbol{N}$ and stores that element as a single file in network project folder. This method was effective to solve the large least squares problems with the need for High performance computing unit (e.g. HP XC3000 in Karlsruhe Institute of Technology-KIT http://www.scc.kit.edu/dienste/hc3.php). While the computer labors are mostly available in most of research institutes and universities.

References

Aledeld, G.; Mayer, G. 1993. The Cholesky method for interval data, *Linear Algebra and its Applications* 194: 161–182. Elsevier Science Inc. http://dx.doi.org/10.1016/0024-3795(93)90120-d

Aquilante, F.; Boman, L.; Bostrom, J.; Koch, H.; Lindh, R.; Sanchez de Meras, A.; Pedersen, T. B. 2011. Cholesky decomposition techniques in electronic structure theory, *Challenges and Advances in Computational Chemistry and Physics* 13: 301–343. Springer. http://dx.doi.org/10.1007/978-90-481-2853-2_13

Breymann, U. 2005. *C++ Einführung und professionelle Programmierung* [*C++ introduction and professional programming*]. 8th ed. München-Wien: Carl Hanser Verlag. ISBN 3-446-40253-5 (in German).

Fang, H.; O'Leary, D. 2006. Modified Cholesky algorithms: a catalog with new approaches. Maryland: University of Maryland.

Franceschi, G.; De Santis, A. 1994. Ionospheric mapping by means of spherical harmonic analysis: new developments, *Advances in Space Research* 14(12): 61–64. http://dx.doi.org/10.1016/0273-1177(94)90240-2

Ghilani, C.; Wolf, P. 2006. Adjustment computations: spatial data analysis. 4th ed. New Jersey: Jones Willey and Sons Inc.

Ghilani, C.; Wolf, P. 2008. Elementary surveying: an introduction to geomatics. 12th ed. New Jersey: Pearson Prentice Hall.

Hartley, R.; Gupta, R.; Chang, T. 1992. Stereo from uncalibrated cameras, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 15–18 June 1992, Champaign, IL, USA, 761–764. http://dx.doi.org/10.1109/CVPR.1992.223179

Jäger, R.; Müller, T.; Saler, H.; Schwäble, R. 2005. *Klassische und robuste Ausgleichungsverfahren* [Classical and Robust Adjustement Procedure]. Heidelberg: Herbert Wichmann Verlag (in German).

Niemeier, W. 2002. *Ausgleichungsrechnung* [Adjustement Computation]. Berlin-New York: Walter de Gruyter (in German).

Nool, M. 1992. Block-Cholesky for parallel processing, *Applied Numerical Mathematics* 10(1): 37–57. Elsevier Science Inc. http://dx.doi.org/10.1016/0168-9274(92)90054-h

Nool, M. 2001. Explicit parallel block Cholesky algorithms on the CRAY APP, *Applied Numerical Mathematics* 19(1–2): 91–114. Elsevier Science Inc. http://dx.doi.org/10.1016/0168-9274(95)00078-9

Okrah, B. 2005. *Computation of a DFHRS Database for Florida (USA) and implementation of a block matrix inversion as a C++ based DLL into DFHRS Software*: Master Thesis, Hochschule Karlsruhe, Karlsruhe.

Osborne, D.; Patrangenaru, V.; Ellingson, L.; Groisser, D. A.; Schwartzman, A. 2013. Nonparametric two-sample tests on homogeneous Riemannian manifolds, Cholesky decompositions and Diffusion Tensor Image analysis, *Journal of Multivariate Analysis* 119: 163–175. http://dx.doi.org/10.1016/j.jmva.2013.04.006

Press, W.; Teukolsky, S.; Vetterling, V.; Flannery, B. 1992. Numerical recipes in C: The art of scientific computing. 2nd ed. Cambridge: Cambridge University Press. ISBN 0-521-43108-5.

Press, W.; Teukolsky, S.; Vetterling, V.; Flannery, B. 2002. Numerical recipes in C++: the art of scientific computing. 2nd ed. Cambridge: Cambridge University Press. ISBN 978-0521880688.

Rothberg, E.; Gupta, A. 1994. An efficient block-oriented approach to parallel sparse Cholesky factorization, *Journal SIAM Journal on Scientific Computing* 15(6): 1413–1439. http://dx.doi.org/10.1137/0915085

Schaefer, U. 2003. Aspects for a block version of the interval Cholesky algorithm, *Journal of Computational and Applied Mathematics* 152(1–2): 481–491.

Schiebl, H. 1999. *Visual C++6.0 für Einsteiger und Fortgeschrittene* [Visual C++6.0 for beginners and advanced], München-Wien: Carl Hanser Verlag. ISBN 3-446-19548-3 (in German).

Smith, S. 2001. The factorability of symmetric matrices and some implications for statistical linear models, *Linear Algebra and its Applications* 335(1–3): 63–80. Elsevier Science Inc. http://dx.doi.org/10.1016/s0024-3795(00)00238-x

Younis, G. 2013. *Regional gravity field modeling with adjusted spherical cap harmonics in an integrated approach*. Darmstadt: Schriftenreihe Fachrichtung Geodäsie der Technischen Universität Darmstadt. ISBN 978-3-935631-28-0.

Younis, G. 2015. *Local earth gravity/potential modeling using ASCH*. Berlin/Heidelberg: Springer. ISSN: 1866-7538. http://dx.doi.org/10.1007/s12517-014-1767-2

Younis, G.; Jäger, R.; Becker, M. 2011. Transformation of global spherical harmonic models of the gravity field to a local adjusted spherical cap harmonic model, *Arabian Journal of Geosciences* 6(2): 375–381. Berlin/Heidelberg: Springer http://dx.doi.org/10.1007/s12517-011-0352-1

Ghadi YOUNIS is a Licensed Land Surveyor and a lecturer for Surveying and Geomatics at Palestine Polytechnic University since 2013. He earned his Master Degree in Geomatics/Geodesy from the Karlsruhe University of Applied Sciences. He also earned his PhD in Civil Engineering and Geodesy from the Technical University of Darmstadt.

Research interests: surveying, GNSS, geodesy, photogrammetry and close range photogrammetry, GIS and cartography.