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:

$$\sum w_i v_i^2 = w_1 v_1^2 + w_2 v_2^2 + \ldots + w_n v_n^2 = \min.$$  \hspace{1cm} (2)

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

$$l_i + v_i = A_i \hat{x}.$$  \hspace{1cm} (3)

The row vector $A_i$ is a row in the design matrix $A$. The elements of $A_i$ 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:

$$\hat{x} = (A^T W_{ll} A)^{-1} A^T W_{ll} l.$$  \hspace{1cm} (4)

In Equation (4), the square matrix $W_{ll} = \sigma_0^2 C_{ll}^{-1}$ is the weight matrix of the observations using their covariance matrix $C_{ll}$. $\sigma_0^2$ 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 $x$ of equation (4), the residuals vector $v$ of the observations reads:

$$v = Ax - I.$$  \hspace{1cm} (5)

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

$$\hat{s}_0^2 = \frac{v^T W_\mu v}{r}. \hspace{1cm} (6)$$

Using the laws of error propagation, the covariance matrix of the unknowns $C_{xx}$ reads (Niemeier 2002):

$$C_{xx} = \hat{s}_0^2 (A^T W_\mu A)^{-1} \hat{s}^2 N^{-1} = \hat{s}_0^2 Q_{xx}. \hspace{1cm} (7)$$

The covariance matrix $C_{ll}$ of the adjusted observations is given by:

$$C_{ll} = \hat{s}_0^2 A (A^T W_\mu A)^{-1} A^T = \hat{s}_0^2 AN^{-1} A^T = \hat{s}_0^2 Q_{ll}. \hspace{1cm} (8)$$

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):

$$\hat{x} = (A^T W_\mu A)^{-1} A^T W_\mu I = N^{-1} c. \hspace{1cm} (9)$$

Here, $N$ is the normal equations matrix, and $c$ is the constants matrix. $N$ and $c$ read:

$$N = (A^T W_\mu A); \hspace{1cm} (10)$$

$$c = A^T W_\mu l. \hspace{1cm} (11)$$

In the case of a large number of observations and unknowns, significant memory would be required to store the matrices $A$, $W$, $N$ and $I$. The Matrix $N$ and the vector matrix $c$ 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 $N$ is symmetric, only the upper or the lower part of the $N$ matrix must be saved. The calculations for the elements of the upper part of $N$ and the elements of the $c$ vector using only one row of the design matrix $A$ related to the $i$-th observation read (Niemeier 2002):

$$n_{m,n} = n_{m,n} + a_m a_n w_i; \hspace{1cm} (12)$$

$$c_m = c_m + a_m w_i l_1. \hspace{1cm} (13)$$

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 $N$ is divided into sub-matrices (blocks). Only one or a limited number of blocks are then loaded into 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 $N$ reads (Okrah 2005):

$$N = \begin{bmatrix}
N_{11} & N_{12} & N_{13} & \cdots & N_{1m} \\
N_{22} & N_{23} & \cdots & \cdots & N_{2m} \\
N_{33} & \cdots & \cdots & \cdots & N_{3m} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
N_{nm} & \cdots & \cdots & \cdots & N_{mm}
\end{bmatrix}. \hspace{1cm} (14)$$

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 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 $N$ can be represented by a lower triangle $L$ matrix or an upper triangle matrix $U$ multiplied by its transpose (Rothberg, Gupta 1994).

$$N = LL^T = U^T U; \hspace{1cm} (15a)$$

$$L = \begin{bmatrix}
l_{11} & 0 & \cdots & 0 \\
l_{21} & l_{22} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
l_{m1} & l_{m2} & \cdots & l_{mm}
\end{bmatrix}; \hspace{1cm} (15b)$$

$$U = \begin{bmatrix}
U_{11} & U_{11} & \cdots & U_{1m} \\
0 & U_{22} & \cdots & U_{2m} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & U_{mm}
\end{bmatrix}. \hspace{1cm} (15c)$$
The factorization of the lower triangle matrix $L$ according to the Cholesky decomposition reads:

\[ l_{jj} = \sqrt{n_{jj} - \sum_{i=1}^{j-1} l_{ij}^2}; \quad (16) \]

\[ l_{ij} = \frac{n_{ij} - \sum_{i=1}^{j-1} l_{ij} l_{ji}}{l_{jj}}. \quad (17) \]

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

\[ N x = L^T x = c. \quad (18) \]

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

\[ L y = c; \quad (19) \]

\[ L^T y = x. \quad (20) \]

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

\[ y_j = \left( c_j - \sum_{i=1}^{j-1} l_{ij} y_i \right) / l_{jj}. \quad (21) \]

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

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

One concern with the Cholesky decomposition factorizations in Equations (16) and (17), the square roots are used in the factorization of the Cholesky decomposition. The results are always positive in the well-conditioned solutions. Where the matrices are positive definite. In many applications the matrices are ill-conditioned (positive semi-definite 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):

\[ L_{jj} = \text{Decomp} \left( N_{jj} - \sum_{i=1}^{j-1} L_{ji} L_{ji}^T \right); \quad (23) \]

\[ L_{ij} = \left[ \text{Bforsub} \left( L_{jj}, N_{ji} - \sum_{i=1}^{j-1} L_{ji} L_{ii}^T \right) \right]^T. \quad (24) \]

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

\[ \text{Bforsub}(A, B) = (\text{Forsub}(A, B_1), \text{Forsub}(A, B_2), \text{Forsub}(A, B_3) \ldots ). \quad (25) \]

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

\[ Y_j = \text{Forsub} \left( L_{jj}, C_j - \sum_{i=1}^{j-1} L_{ji} Y_i \right). \quad (26) \]

The backward substitution to find the block element $X_j$ of the unknowns’ vector $x$ using the block elements $Y_j$ of the vector $y$ reads:

\[ X_j = \text{Backsub} \left( L_{jj}^T, Y_j - \sum_{i=j+1}^{k} L_{ij}^T X_i \right). \quad (27) \]

The function $\text{Backsub}(\phantom{x})$ 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 $N$.

### 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 $\mathbf{x}$ in a binary format. While the matrices $\mathbf{A}$, $\mathbf{A}^T$, $\mathbf{N}$, $\mathbf{I}$ and $\mathbf{W_{\text{diagonal}}}$ 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 $\mathbf{N}$ for $(10 \times 10)$ blocks as given in Equation (14), only the upper part of the matrix $\mathbf{N}$ was stored using 92 MB and the $\mathbf{c}$-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):
network to access the observations and solve for specific parts of the matrix \( N \). A total time for the computations was around 36 hours. This method has 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 \( N \) and \( c \) matrices in Equation (11) directly from the observations without the need to store all matrices of the traditional method of the solution like \( A \), \( A^T \) and \( W \) 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 coordinate 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 \( N \) 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


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.