SPARSE MATRIX LIBRARY FOR POWER SYSTEM STATE ESTIMATION BASED ON FULL KNUTH'S METHOD

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TUNA YILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONIC ENGINEERING

JUNE 2021

Approval of the thesis:

SPARSE MATRIX LIBRARY FOR POWER SYSTEM STATE ESTIMATION BASED ON FULL KNUTH'S METHOD

submitted by **TUNA YILDIZ** in partial fulfillment of the requirements for the degree of **Master of Science** i**n Electrical and Electronic Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, **Graduate School of Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of the Department, **Electrical and Electronics Eng.** _____

Assoc. Prof. Dr. Murat Göl
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Ali Nezih Güven
Electrical and Electronics Engineering Dep., METU _____

Assoc. Prof. Dr. Murat Göl
Electrical and Electronics Engineering Dep., METU _____

Prof. Dr. Ece Güran Schmidt
Electrical and Electronics Engineering Dep., METU _____

Assoc. Prof. Dr. Ozan Keysan
Electrical and Electronics Engineering Dep., METU _____

Assist. Prof. Dr. Oğuzhan Ceylan
Administrative and Social Sciences, Kadir Has Uni. _____

Date: 18.06.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Tuna Yıldız

Signature :

ABSTRACT

SPARSE MATRIX LIBRARY FOR POWER SYSTEM STATE ESTIMATION BASED ON FULL KNUTH'S METHOD

Yıldız, Tuna
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Murat Göl

June 2021, 72 pages

Considering the increase in power system size and the number of PMUs, it is essential to use a computationally efficient state estimator. The Fast Decoupled State Estimation is the most common method used in industrial applications, thanks to its computational efficiency and ease of implementation. However, it can be improved further by using sparse storage techniques, thanks to the sparse structure of the state estimation matrices.

In literature, there are several types of sparse storage algorithms, however, only a few of them is suitable for the power system state estimation operations. Considering the possible frequent topology changes, Knuth's method has a superiority in power system applications. However, even Knuth's Method can be enhanced further by using additional information of the matrices.

This thesis proposes the full Knuth's Method for sparse storage algorithm. Considering that sparse storage libraries for real-time power system applications are not available as open-source, firstly modified sparse storage library is built. After that, by using the created sparse storage library, the features of the power system

state estimator are built. Thanks to the designed sparse storage library, the computational performance is increased further for power system state estimation.

Keywords: Sparse Storage, Knuth’s Method, Full Knuth’s Method, State Estimation, Sparse Matrix Inversion

ÖZ

GÜÇ SİSTEMİ DURUM KESTİRİMİ İÇİN TAM KNUTH YÖNTEMİNE DAYALI SEYREK MATRİS KÜTÜPHANESİ

Yıldız, Tuna
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Murat Göl

Haziran 2021, 72 sayfa

Güç sistemi boyutundaki ve PMU'ların sayısındaki artış dikkate alındığında, hesaplama açısından verimli bir durum kestirimcisi kullanmak önemlidir. Fast Decoupled Durum Kestirimi, hesaplama verimliliği ve uygulama kolaylığı sayesinde endüstriyel uygulamalarda kullanılan en yaygın yöntemdir. Buna ek olarak Fast Decoupled Durum Kestirimi, güç sistemi durum kestiriminde kullanılan matrislerin seyrek yapısı sayesinde seyrek depolama gibi depolama teknikleri kullanılarak daha da geliştirilebilir.

Literatürde bir çok seyrek depolama algoritması vardır, ancak bunlardan sadece birkaçı güç sistemi durum kestirimi işlemleri için uygundur. Olası sık topoloji değişiklikleri göz önüne alındığında, Knuth Yönteminin güç sistemi uygulamalarında bir üstünlüğü vardır. Bunlara ek olarak, Knuth Yöntemi bile matrislerle ilgili ek bilgiler kullanılarak daha da geliştirilebilir.

Bu tez, seyrek depolama algoritması için tam Knuth Yöntemini önermektedir. Gerçek zamanlı güç sistemi uygulamaları için seyrek depolama kütüphanelerinin açık kaynak olara mevcut olmadığı düşünülerek, öncelikle modifiye edilmiş seyrek depolama kitaplığı oluşturulmuştur. Daha sonra oluşturulan seyrek depolama

kütüphanesi kullanılarak güç sistemi durum kestirimcisinin özellikleri oluşturulmuştur. Tasarlanan seyrek depolama kütüphanesi sayesinde, güç sistemi durum kestirimi için hesaplama performansı daha da artırılmıştır.

Anahtar Kelimeler: Seyrek Depolama, Knuth Yöntemi, Geliştirilmiş Knuth Yöntemi, Durum Kestirimi, Seyrek Matris Tersi

To my family

Osman Yıldız

Ulviye Yıldız

Gizem Yıldız

ACKNOWLEDGMENTS

First, I would like to thank my supervisor Murat Göl for his precious friendship, support, and guidance. It was a great pleasure to work three years with him. He always supported me and answered my questions whenever I had a problem with the research. Also, my academic view was highly influenced by his guidance.

I would really appreciate Scientific and Technological Research Council of Turkey (TUBITAK) for financial support via BIDEB 2210-A funding.

I would also to thank dearest experts who were involved in this thesis project: Emre Rızvanoğlu, Ozkan Tanrıverdi, Umut Can Çay, Etki Açılan, and Batuhan Bülbül. Without their help and participation in project, the validation of the thesis cannot have successful results.

Significant support came from my family as specially my parents Osman Yıldız, and Ulviye Yıldız. They have always encouraged me to continue my academic life without considering the financial issues and mentally supported me in tough times.

Last but not least support came from my best friend Zeynep Suvacı. Thanks to her love, patience, and morale support, I got to that point.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

WLS	Weighted Least Squares
FD-WLS	Fast Decoupled Weighted Least Squares
SE	State Estimation
CSR	Compressed Sparse Row
CSC	Compressed Sparse Column
BLUE	Best Linear Unbiased Estimator
RAM	Read Access Memory

CHAPTER 1

INTRODUCTION

In power systems, a state estimation process is an essential tool for monitoring the system. To solve the state estimation for monitoring purposes, it is necessary to have the required number of measurements gathered from the field, and the number of measurements is increasing day after day with the increasing system size. Those measurement devices mainly measure the " Active and Reactive Power Injections, Active and Reactive Power Flows and Voltages." However, in recent years, the technologies behind the measurement devices are evolving rapidly and PMU devices, which measure the voltage magnitude and voltage angle, current magnitude, and current angle, penetrate the power systems. As a result of the high number of measurements and the increased system size, the data processed in state estimation increased. Due to these issues, the solution time of state estimation processes increases, and taking action for problematic situations can be delayed. Therefore, to monitor the power system properly, state estimators have to solve the given measurement set before the next measurement set is collected from the field.

State estimators mainly contain two steps, and those steps are "Estimation of States" and "Bad Data Analysis." In the power system state estimation process, the states are defined as "Voltage Magnitudes of Buses (V)" and "Voltage Angles of Buses (θ)." The purpose of the state estimators is to use the provided measurements that are gathered from the field and trying to estimate the "V" and "θ" of each bus of the power system. However, the outcome of the state estimation process may be affected due to malfunctioned measurements included in the provided measurement set. Therefore, bad data analysis becomes important to detect and correct malfunctioned measurements among the provided measurement set.

In literature, there are several state estimators, which have their own advantages and disadvantages to each other [1-4]. However, the most common state estimator in the field applications is “WLS (Weighted Least Squares)” since it is easy to implement and the computational time of the WLS is superior to other state estimators. In addition, there are several shapes of the WLS state estimators to decrease the time consumption of the estimation process. The most used one in the field is the “FD-WLS (Fast Decoupled Weighted Least Squares)” state estimator [5]. In the FD-WLS state estimator, the time consumption of the estimation processed decreased based on the observations that states of “V” are strongly related with reactive power measurements and states of “θ” are strongly related with active power measurements [5]. As a result of these observations, the Jacobian matrix is calculated only once during the process and is used for all iterations until the system converged. Therefore, the time consumption of building the Jacobian matrix in each iteration is eliminated.

Besides the state estimators, the next important step of the state estimation process is “Bad Data Analysis.” Once the states of the system are obtained by using the “WLS State Estimator” or “Fast Decoupled State Estimator,” bad data analysis is performed to detect and identify the malfunctioned data among the provided measurement set [6-9]. This identification and detection process of malfunctioned measurement is performed under two steps. These steps are written as follows;

- Chi-square test,
- Normalize residual test.

In the bad data analysis process, the Chi-square test is performed for detection of whether there are malfunctioned measurements among the provided measurement set or not. The Normalized residual test is performed for identification of the bad data, which is detected by Chi-square [7].

In the state estimation algorithms, for both processes of “Estimation of States” and “Bad Data Analysis” the main time consumptions occurred during the three main matrix operations, which are matrix multiplications, Cholesky decompositions, and

matrix inversion. Although the solution time improvements were obtained with the "FD-WLS" state estimation, there are still a significant amount of time consumptions that occurred during these processes due to included zero-entry calculations in matrix operations. These unnecessary zero-entry calculations become important with increasing the number of measurements and the system sizes.

Thanks to the super sparse structures of matrices in state estimators, it is possible to use sparse storage algorithms for further improvement to eliminate this unnecessary time consumption that occurred by zero entries and accelerate the state estimation and bad data analysis process. Sparse storage is a method that keeps the information of non-zero entries in matrices via linked lists to eliminate the zero-entry calculations during the matrix operations.

In this thesis, it is aimed to propose a suitable solution for state estimation processes in order to decrease the solution time of solving each measurement set for a given power system structure by using sparse storage algorithms.

In literature, there are several sparse storage methods [10-15]. However, sparse storage is mostly case needed process. The algorithms such as Compressed Sparse Row (CSR), Compressed Sparse Column (CSC), Skyline Storage (SKS) etc., are mostly used for fixed-size matrices, but in power system state estimation process matrix sizes are changing during the operations in real-time. Therefore, those algorithms do not provide enough flexibility for the power system state estimation.

The most common sparse storage methods that satisfy power system state estimator needs are "Gustavson's Method" and "Knuth's Method." Both methods contain two types of solutions under themselves. Those solutions are forming linked lists with row information of non-zero elements and column information of non-zero elements [13,14]. The privilege among these solutions is determined according to the programming language that is used for the applications.

In power system state estimation process, flexibility and the linked list search time are considered as two key features of the sparse storage algorithms due to frequent

alterations in the topology, which reflects on the used matrices in applications. Considering these two key features, Knuth's Method becomes superior to the Gustavson's Method due to lack of flexibility in Gustavson's Method. Therefore, in this work, "Knuth's Method" is enhanced by combining both row ordered linked lists and column ordered linked lists together. Beside the increase flexibility, the time consumption of the linked list search is also decreased as a further enhancement with the help of the utilization of the proposed method. Nevertheless, thanks to the improvement of the technology behind the memory storage, using extra space in "RAMs (Read Access Memories)" does not create any burden due to the proposed method.

There are several sparse storage methods in the literature, however, there are not any open-source algorithms that perform sparse storage for the power system state estimation process. Therefore, this work also develops an open-source sparse storage library for the state estimation process based on the utilized storage technique.

In this work, it will be shown that the proposed method, which is combining the row ordered linked list and column ordered linked list for "Knuth's Method" gives the desired results and enhances the "Knuth's Method" in matrix operations for power system state estimator.

As a result, this thesis provides new perspective for the Knuth's Method which will contribute to the decreases in terms of solution time of power system state estimation processes.

The main contributions of the proposed method are listed below;

- The linked list search time was further decreased compared to the "Knuth's Method",
- The flexibility of linked lists operations for sparse matrix operations was increased,
- The solution time of the matrix operations were significantly decreased,
- The open-source sparse storage library was provided.

In this thesis, in order to explain the proposed method, the work is divided into eight chapters.

In the first chapter, which is the introduction part, the definition of the problem is explained. Moreover, the existing solutions in the literature for this defined problem are reviewed in this chapter. In addition, the innovation of the proposed method is given in the introduction part. Moreover, the contributions of the proposed method are given.

In the second chapter, the background information, the details of the state estimation process, the details of the bad data analysis process will be given under the "Weighted Leas Square" and "Bad Data Analysis" parts. Under the "Bad Data Analysis" part, two steps which are "Chi-square Test" and "Normalize Residual Test" will be explained in detail. These explained processes in those parts are used in order to build a power system state estimator and then the built state estimator is further improved with the proposed method.

In the third chapter, the sparse storage techniques which are suitable for the power system state estimation process will be explained in detail. The main operations, which are addition, deletion, and changing a value of a non-zero element in linked lists for sparse storage, will be explained. Moreover, the advantages and disadvantages of those sparse storage methods will be given.

In the fourth chapter, matrix multiplication, the details of sparse matrix multiplication using "Knuth's Method" and the proposed method will be explained. A comparison between the "Knuth's Method" and the proposed method will be given. Moreover, the advantages of the proposed method over the "Knuth's Method" will be shown. Then the results of the proposed method for the matrix multiplication process and the comparison between the proposed method with "Knuth's Method" will be given.

In the fifth chapter, the processes of Cholesky decomposition and the importance of the decomposition process for state estimation will be explained. Then the results of

the comparison of the proposed method for Cholesky decomposition with other methods will be shown.

The importance of the matrix inversion process and the details of the used "Takahashi Method" will be given in chapter six. Then the results of the comparison for the proposed method with other solutions will be given.

In chapter seven, the details of the built real-time state estimator will be given. In this state estimator, the solution time results of the state estimation and bad data analysis processes that utilized the proposed method will be shown.

In the final chapter of this thesis, the discussion of the proposed method, observations regarding the test results of the proposed method in state estimation process and future works will be provided.

CHAPTER 2

BACKGROUND INFORMATION

In order to describe the importance of sparse storage algorithm for power system monitoring, firstly state estimation process should be analyzed. In the state estimation process there are several steps in order to estimate bus voltages and bus voltage angles [6]. Those steps can vary, however, the main idea is the same for all state estimators that are used for the power system monitoring.

In this part of the thesis, the well-known state estimator which is called as "Weighted Least Squares – WLS" state estimator, will be discussed. The benefits of WLS over the other estimators are:

- It is easy to implement,
- Computational performance is better among others,
- And it is the best linear unbiased estimator "(BLUE)[6]".

And later, bad data analysis, which is performed after the WLS-SE in order to find the corrupted bad data among the measurement set, will be analyzed.

In this Chapter, the discussion about the estimator will be given in the Weighted Least Squares part, and the discussion about bad data analysis will be given under the Bad Data Analysis part.

2.1 Weighted Least Squares State Estimator

In the power system monitoring, state estimation has a crucial role in finding the system states, namely, bus voltage magnitudes and bus voltage angles, using the provided measurement set. Those measurements are mainly divided into two measurement types which are "SCADA" and " PMU (Phasor Measurement Unit)" measurements. SCADA measurements consist of power flow measurements, power injection measurements, and voltage magnitude measurements, and PMU

measurements consist of voltage magnitude, voltage angle, current magnitude, and current angle measurements. The state estimator aims to use those measurements to find an optimum solution for system states via optimizing the objection function.

In order to achieve this, first of all, system components which are transmission lines, transformers, shunt capacitors or rectors, and tap changing or phase shifting transformers, should be modeled. To model these components, the generalized π model is used.

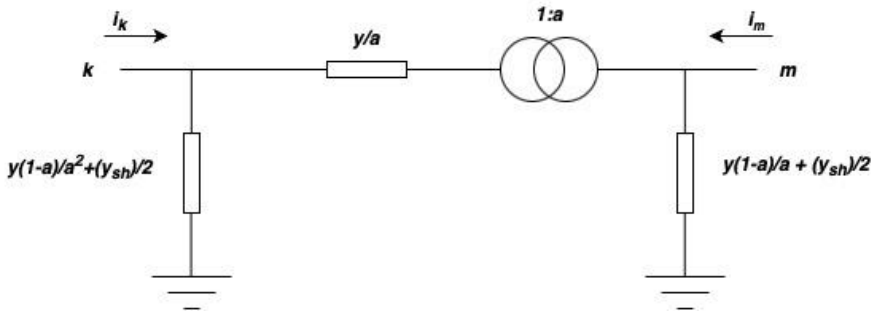

Figure 2.1: The generalized π model of a transformer.

After modeling the components, the y_{bus} matrix is obtained according to formulation in [6]:

$$\begin{bmatrix} i_k \\ i_m \end{bmatrix} = \begin{bmatrix} y/a^2 + y_{sh}/2 & -y/a \\ -y/a & y + y_{sh}/2 \end{bmatrix} \begin{bmatrix} v_k \\ v_m \end{bmatrix} \quad (2.1)$$

where,

y is the series admittance value of the line (leakage admittance for transformer),

a is the tap value of the transformer ($a = 1$ if this is transmission line),

y_{sh} is the line charging susceptance value.

With the use of equation (2.1) network model formed as follow:

$$I = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1N} \\ Y_{21} & Y_{22} & \dots & Y_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{N1} & Y_{N2} & \dots & Y_{NN} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} = Y\,V \qquad (2.2)$$

where,

i_k is the net current injection at bus k,

v_k is the voltage phasor at bus k, and Y_{km} is the $(k, m)^{th}$ element of Y matrix.

This network model is one of the key elements for WLS-SE since the calculation of measurement function of state estimation needs network models. Once the network model is obtained, according to [6], the definition of the state estimation formulation can be written as follows:

$$z = h(x) + e \qquad (2.3)$$

where,

h(.) represents the measurement function which makes a relationship between measurements to state vector x $(n \times 1)$,

x represents the true state vector with the size of $(n \times 1)$,

e corresponds to the measurement error vector with the size of $(m \times 1)$,

z represents the measurement vector with the size of $(m \times 1)$,

n is the number of states, and *m* is the number of measurements.

In WLS-SE formulation, there are several assumptions are made for measurement errors such as;

- E[e] = 0
- $E[e_i e_j] = 0$
- $cov(e) = E[ee^T] = R$

where,

R is named as measurement error covariance matrix, and it is a diagonal matrix.

The objective function of the WLS-SE can be written as follows:

$$J(x) = \sum_{i=1}^{m} W_{ii} \, (z_i - \, h_i(x))^2 \qquad (2.4)$$

In equation (2.4),

subscript *i* shows the i^{th} entry of the related vector,

subscript *ii* represents the ii^{th} entry of the related vector, and W is equal to the R^{-1}.

It can be seen that with the minimization of the *x* gradient of equation (2.4) will be zero according to the first order optimality condition. For this reason, the following relation will be held:

$$g(x) = \frac{\partial J(x)}{\partial x} = \, -H^T(x)W[z - h(x)] = 0 \qquad (2.5)$$

Since $g(x)$ is a nonlinear function, to solve the nonlinear problem, an iterative solution is required. Therefore, equation (2.5) is linearized around the state vector x^k by using the Taylor Series Expansion, and the following iterative solution formulation is written as follows:

$$\Delta x^{k+1} = G(x^k)^{-1} H^T(x^k) W[z - h(x^k)] \qquad (2.6)$$

where,

$\Delta x^{k+1} = \, x^{k+1} - \, x^k$,

$G(x^k) = H^T(x^k) W H(x^k)$,

$H(x^k)$ represents the measurement Jacobian matrix with a size of $(m \times n)$,

x^k represents the state vector that is estimated at iteration *k* and

h(.) is the measurement function that creates measurements by using x^k.

After forming equation (2.6), to solve the state estimation problem, the measurement function is formed as follows;

$$H = \begin{bmatrix} \frac{\partial P_{inj}}{\partial \theta} & \frac{\partial P_{inj}}{\partial V} \\ \frac{\partial P_{flow}}{\partial \theta} & \frac{\partial P_{flow}}{\partial V} \\ \frac{\partial Q_{inj}}{\partial \theta} & \frac{\partial Q_{inj}}{\partial V} \\ \frac{\partial Q_{flow}}{\partial \theta} & \frac{\partial Q_{flow}}{\partial V} \\ 0 & \frac{\partial V_{mag}}{\partial V} \end{bmatrix} \tag{2.7}$$

The expressions for each partition in equation (2.6) are given as follows;

- Elements corresponding to real power injection measurements:

$$\begin{aligned} \frac{\partial P_i}{\partial \theta_i} &= \sum_{j=1}^{N} V_i V_j \left(-G_{ij} sin\theta_{ij} + B_{ij} cos\theta_{ij}\right) - V_i^2 B_{ii} \\ \frac{\partial P_i}{\partial \theta_j} &= V_i V_j \left(G_{ij} sin\theta_{ij} - B_{ij} cos\theta_{ij}\right) \\ \frac{\partial P_i}{\partial V_i} &= \sum_{j=1}^{N} V_j (G_{ij} cos\theta_{ij} + B_{ij} sin\theta_{ij}) + V_i G_{ii} \\ \frac{\partial P_i}{\partial V_j} &= V_i \left(G_{ij} cos\theta_{ij} + B_{ij} sin\theta_{ij}\right) \end{aligned} \tag{2.8}$$

- Elements corresponding to reactive power injection measurements:

$$\frac{\partial Q_i}{\partial \theta_i} = \sum_{j=1}^{N} V_i V_j \left(G_{ij} cos\theta_{ij} + B_{ij} sin\theta_{ij}\right) - V_i^2 G_{ii}$$

$$\frac{\partial Q_i}{\partial \theta_j} = V_i V_j\left(-G_{ij}cos\theta_{ij} - B_{ij}sin\theta_{ij}\right)$$
$$\frac{\partial Q_i}{\partial V_i} = \sum_{j=1}^{N} V_i\, V_j\left(G_{ij}cos\theta_{ij} + B_{ij}cos\theta_{ij}\right) - V_i B_{ii} \quad (2.9)$$
$$\frac{\partial Q_i}{\partial V_j} = V_i V_j\left(-G_{ij}cos\theta_{ij} - B_{ij}sin\theta_{ij}\right)$$

- Elements corresponding to real power flow measurements:

$$\frac{\partial P_{ij}}{\partial \theta_i} = V_i V_j\left(g_{ij}sin\theta_{ij} - b_{ij}cos\theta_{ij}\right)$$
$$\frac{\partial P_{ij}}{\partial \theta_j} = -V_i V_j\left(g_{ij}sin\theta_{ij} - b_{ij}cos\theta_{ij}\right)$$
$$\frac{\partial P_{ij}}{\partial V_i} = -V_j\left(g_{ij}cos\theta_{ij} + b_{ij}sin\theta_{ij}\right) + 2(g_{ij} + g_{si})\, V_i \quad (2.10)$$
$$\frac{\partial P_{ij}}{\partial V_j} = -V_i\left(g_{ij}cos\theta_{ij} + b_{ij}sin\theta_{ij}\right)$$

- Elements corresponding to reactive power flow measurements:

$$\frac{\partial Q_{ij}}{\partial \theta_i} = -V_i V_j\left(g_{ij}cos\theta_{ij} + b_{ij}sin\theta_{ij}\right)$$
$$\frac{\partial Q_{ij}}{\partial \theta_j} = V_i V_j\left(g_{ij}cos\theta_{ij} + b_{ij}sin\theta_{ij}\right)$$
$$\frac{\partial Q_{ij}}{\partial V_i} = -V_i\left(g_{ij}sin\theta_{ij} - b_{ij}cos\theta_{ij}\right) - 2V_i\left(b_{ij} + b_{si}\right) \quad (2.11)$$
$$\frac{\partial Q_{ij}}{\partial V_j} = -V_i\left(g_{ij}sin\theta_{ij} - b_{ij}cos\theta_{ij}\right)$$

- Elements corresponding to voltage magnitude measurements:

$$\frac{\partial V_i}{\partial V_i} = 1, \frac{\partial V_i}{\partial V_j} = 0, \frac{\partial V_i}{\partial \theta_i} = 0, \frac{\partial V_i}{\partial \theta_j} = 0 \quad (2.12)$$

where,

V_i, θ_i are the voltage magnitude and phase angle at bus i, θ_{ij} is equal to phase angle differences between bus i and bus j,

$G_{ij} + jB_{ij}$ is the ij^{th} element of the bus admittance matrix,

$g_{ij} + jb_{ij}$ is the admittance value of the branch which is connecting the bus i and j,

$g_{si} + jb_{si}$ is the admittance of the shunt branch connected to bus i,

N_i is the bus number that is directly connected to bus i.

In the power system, with the existence of super sparse structure in the network model, there occurs only a few non-zero elements for equations (2.8), (2.9), (2.10), (2.11), and (2.12). The example structure of the Jacobian *"H"* matrix for IEEE-30 Bus system can be seen in Figure 2.2.

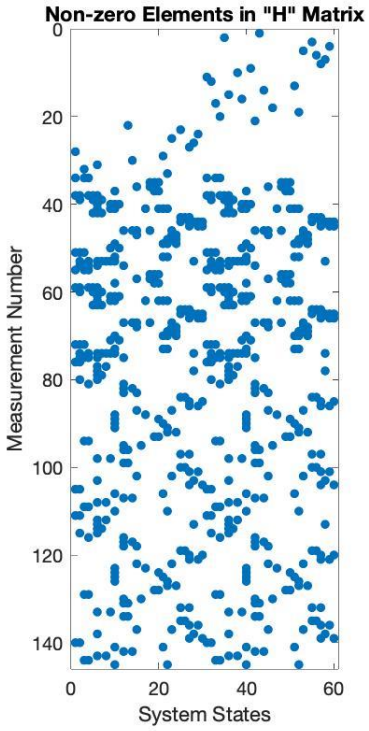

Figure 2.2: The *"H"* matrix sparse structure.

As is seen in Figure 2.2, the sparsity of the *"H"* matrix is around 9%. According to [16], the shape as *"H"* matrices is defined as a super sparse structure. Moreover, in equation (2.6), the Gain *"(G)"* matrix is constructed with the *"H"* matrix. Therefore, thanks to a sparse structure of the "*H"* matrix, the "*G"* matrix is also has a super sparse structure. The example figure for the Gain matrix (*"G"*) is in Figure 2.3.

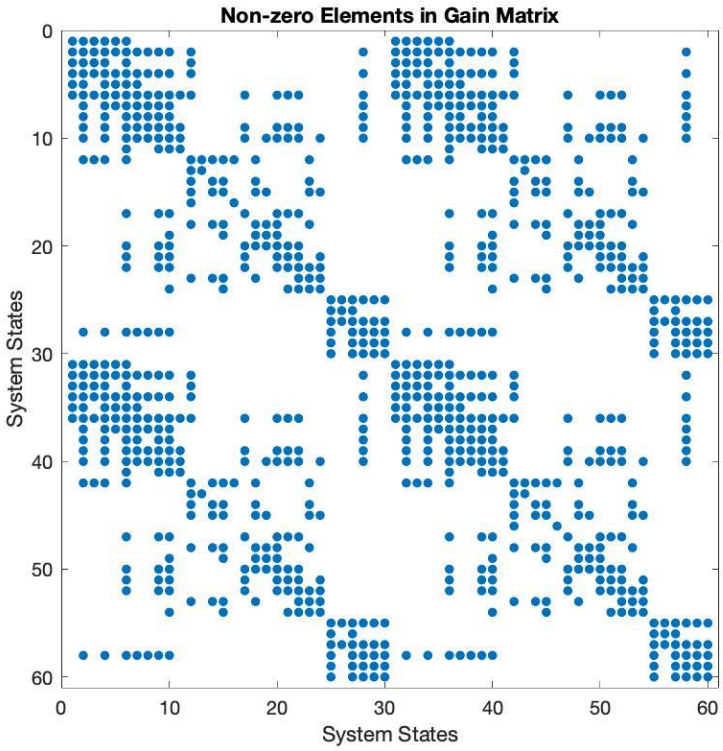

Figure 2.3: The Gain Matrix sparse structure.

Once the *"G"* matrix and *"H"* matrix is calculated, equation (2.6) should be solved for each iteration. However, since there is an inverse matrix operation in equation (2.6), not to take an inverse of the *"G"* matrix, the "Cholesky Decomposition" process is applied to the *"G"* matrix.

As a result, the "Cholesky Decomposition" method is utilized instead of taking an inverse of the *"G"* to decrease computational time. According to [17,18], "Cholesky Decomposition" formulation can be written as follows:

$$A = LL^T = \begin{bmatrix} x & 0 & 0 \\ x & x & 0 \\ x & x & x \end{bmatrix} \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix} \tag{2.13}$$

$$L_{ij} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}\left(L_{jk}\right)^*}$$
$$L_{ij} = \frac{A_{jj} - \sum_{k=1}^{j-1} L_{jk}\left(L_{jk}\right)^*}{\sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}\left(L_{jk}\right)^*}} \tag{2.14}$$

where,

i and *j* are the row and column indices of the matrix.

Once the calculation is performed for "Cholesky Decomposition," equation (2.14) is solved for Δx^{k+1} at each iteration until the determined threshold for the convergence is satisfied.

During this calculation process of each iteration, a sparse storage method can be applied for further improvement in terms of computational speed. Since the sparsity of matrices is increasing with the increasing power system size, a zero-entry calculations cause an undesired time-consumptions fo processes such as *"Gain Matrix"* calculation and "Cholesky Decomposition", since those calculations include a lot of multiplication, addition, and subtraction operations. Therefore, this time-consuming calculations, which are occurred due to the zero entries in matrices, can be eliminated with the help of sparse storage in significant order.

2.2 Bad Data Analysis

Once WLS-SE is performed, the bad data analysis checks presence of an erroneous measurement that biases system states during the state estimation process. In general, bad data analysis is formed of two steps [6-9]. The first step is detecting the

erroneous measurement, and the second step is identifying the detected measurement and eliminating them if possible. Those erroneous data occurs due to various reasons such as having a finite accuracy among the meters, telecommunication medium, etc. Therefore, eliminating those bad data among the measurement set provided to the state estimator is crucial for power system operators in terms of proper system monitoring.

To achieve this, two well-known approaches, which are the "Chi-Square Test" and "Normalized Residual Test," are used for the detection and identification purposes of bad data, respectively [6, 19, 20].

2.2.1 Chi-Square Test

The main purpose of the Chi-square test is to detect the existence of the erroneous measurement. According to [19], this is achieved by utilizing the x^2 distribution.

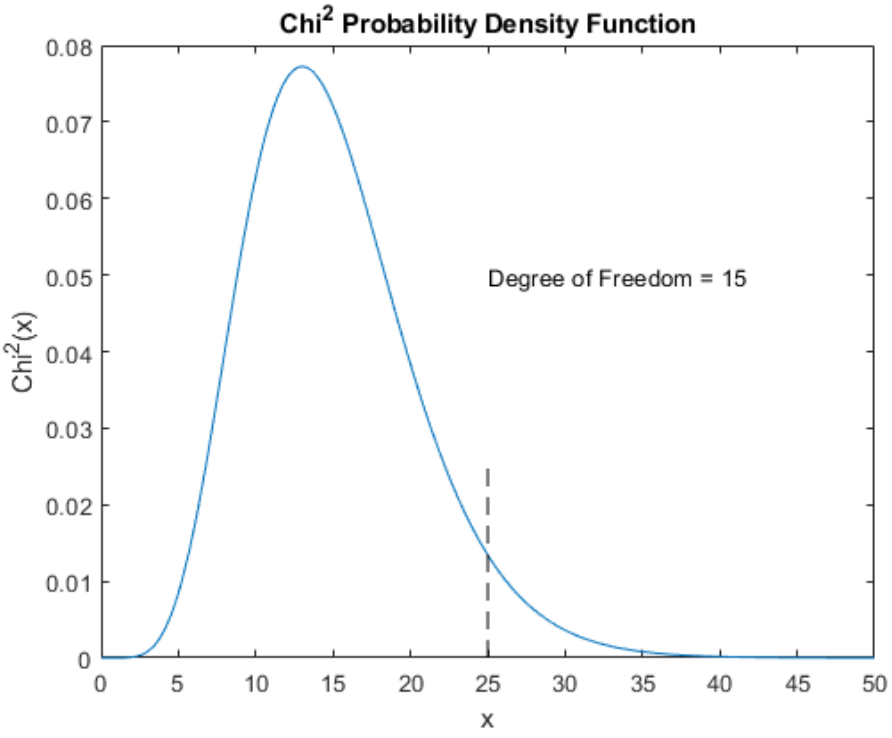

Figure 2.4: Probability density function for x^2.

Degree of Freedom	Probability of Exceeding the Critical Value									
	0,99	0,95	0,9	0,8	0,75	0,5	0,25	0,1	0,05	0,01
1	0,00	0,00	0,02	0,06	0,10	0,45	1,32	2,71	3,84	6,63
2	0,02	0,10	0,21	0,45	0,58	1,39	2,77	4,61	5,99	9,21
3	0,11	0,35	0,58	1,01	1,21	2,37	4,11	6,25	7,81	11,34
4	0,30	0,71	1,06	1,65	1,92	3,36	5,39	7,78	9,49	13,28
5	0,55	1,15	1,61	2,34	2,67	4,35	6,63	9,24	11,07	15,09
6	0,87	1,64	2,20	3,07	3,45	5,35	7,84	10,64	12,59	16,81
7	1,24	2,17	2,83	3,82	4,25	6,35	9,04	12,02	14,07	18,48
8	1,65	2,73	3,49	4,59	5,07	7,34	10,22	13,36	15,51	20,09
9	2,09	3,33	4,17	5,38	5,90	8,34	11,39	14,68	16,92	21,67
10	2,56	3,94	4,87	6,18	6,74	9,34	12,55	15,99	18,31	23,21
11	3,05	4,57	5,58	6,99	7,58	10,34	13,70	17,28	19,68	24,72
12	3,57	5,23	6,30	7,81	8,44	11,34	14,85	18,55	21,03	26,22
13	4,11	5,89	7,04	8,63	9,30	12,34	15,98	19,81	22,36	27,69
14	4,66	6,57	7,79	9,47	10,17	13,34	17,12	21,06	23,68	29,14
15	5,23	7,26	8,55	10,31	11,04	14,34	18,25	22,31	25,00	30,58
16	5,81	7,96	9,31	11,15	11,91	15,34	19,37	23,54	26,30	32,00
17	6,41	8,67	10,09	12,00	12,79	16,34	20,49	24,77	27,59	33,41
18	7,01	9,39	10,86	12,86	13,68	17,34	21,60	25,99	28,87	34,81
19	7,63	10,12	11,65	13,72	14,56	18,34	22,72	27,20	30,14	36,19
20	8,26	10,85	12,44	14,58	15,45	19,34	23,83	28,41	31,41	37,57
21	8,90	11,59	13,24	15,44	16,34	20,34	24,93	29,62	32,67	38,93
22	9,54	12,34	14,04	16,31	17,24	21,34	26,04	30,81	33,92	40,29
23	10,20	13,09	14,85	17,19	18,14	22,34	27,14	32,01	35,17	41,64
24	10,86	13,85	15,66	18,06	19,04	23,34	28,24	33,20	36,42	42,98
25	11,52	14,61	16,47	18,94	19,94	24,34	29,34	34,38	37,65	44,31
26	12,20	15,38	17,29	19,82	20,84	25,34	30,43	35,56	38,89	45,64
27	12,88	16,15	18,11	20,70	21,75	26,34	31,53	36,74	40,11	46,96
28	13,56	16,93	18,94	21,59	22,66	27,34	32,62	37,92	41,34	48,28
29	14,26	17,71	19,77	22,48	23,57	28,34	33,71	39,09	42,56	49,59
30	14,95	18,49	20,60	23,36	24,48	29,34	34,80	40,26	43,77	50,89

Figure 2.5: The part of the Chi-Square table.

In Figure 2.4 area under the probability density function is related to the probability of finding X in the corresponding region. In other words

$$Pr(X \geq x_{th}) = \int_{x_{th}}^{\infty} x^2(u).du \tag{2.15}$$

Equation (2.15) represents the probability of X being larger than a certain threshold x_{th}. With the increasing values of x_{th} value, the probability of X being a specified region decreases since the tail of the distribution is decaying. In Figure 2.4, the dashed line corresponds to a threshold value, representing the largest acceptable

value for X that will not imply any erroneous measurement. If the value of X exceeds the thresold value, then it is flagged as a bad data suspicion [6, 20]. These threshold values can be found by using the chi-square table seen in Figure 2.5. Therefore, with the utilization of the value of X that corresponds to the cost of the objective function in equation (2.4) and the threshold value found by using the chi-square table, erroneous measurements can be detected among the given measurement set.

2.2.2 Normalize Residual Test

In the previous chapter, the detection of the erroneous measurement is explained for the bad data analysis. In this part, the identification method will be given in detail for measurements detected in the "Chi-Square Test."

According to [6, 19], for WLS-SE equation shown in equation (2.3) changed to a linearized measurement equation as follows:

$$\Delta\hat{x} = H\Delta x + e \tag{2.16}$$

where, $E(e) = 0$ and

$cov(e) = R.$

Then, the linearized state vector for WLS-SE can be written as:

$$\begin{aligned} \Delta\hat{x} &= (H^T R^{-1} H)^{-1} H^T R^{-1} \Delta z \\ &= G^{-1} H^T R^{-1} \Delta z \end{aligned} \tag{2.17}$$

and the esitmated value of Δz is wrriten as:

$$\Delta\hat{z} = H\Delta\hat{x} = K\Delta z \tag{2.18}$$

where,

$K = HG^{-1}H^{T}R^{-1}$ which is called a "hat matrix" for putting a hat on Δz.

The structure of the hat matrix can be seen in Figure 2.6. To proceed further, firstly, the properties of the *"K"* matrix should be investigated. Those properties are;

$$K.K.K.K \ldots . K = K \tag{2.19}$$

$$KH = H \tag{2.20}$$

$$(I-K)\,H = 0 \tag{2.21}$$

Moreover, the residuals of measurements can be described as follows:

$$\begin{aligned} r &= \Delta z - \Delta\hat{z}\ as \\ &= (I-K)\Delta z \\ &= (I-K)(H\Delta x + e) \\ &= (I-K)e \\ &= Se \end{aligned} \tag{2.22}$$

where,

S is named as residual sensitivity matrix, and it represents the sensitivity of measurements residual to the measurement errors, and it has the following properties:

$$S.S.S \ldots S = S \tag{2.23}$$

$$S.R.S^{T} = SR \tag{2.24}$$

By using the relation between the residuals of measurements and errors with using equation (2.22), covariance and the mean of the measurement residuals can be obtained as follows:

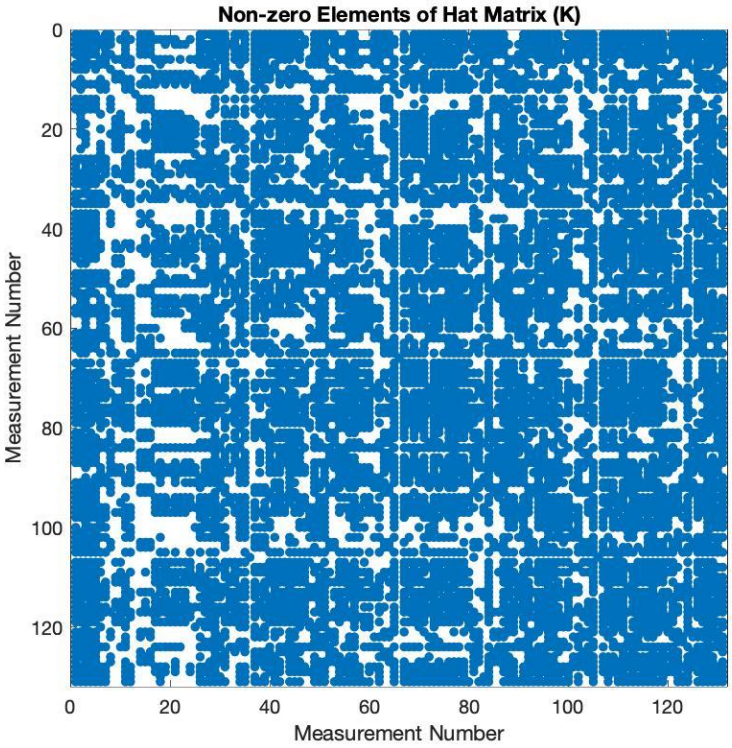

Figure 2.6: The structure of the hat matrix (*K*).

$$
\begin{aligned}
E(r) &= E(S.e) = SE(e) = 0 \\
Cov(r) &= \Omega = E[rr^T] \\
&= SE[ee^T]S^T \\
&= SRS^T \\
&= SR
\end{aligned}
\tag{2.25}
$$

Once the "*S*" matrix is calculated, the next step, which is finding the normalized residuals, is utilized by using the following formulations:

$$r_i = z_i - h_i(\hat{x}), i = 1, \ldots, m \tag{2.26}$$

$$r_i^N = \frac{|r_i|}{\sqrt{\Omega_{ii}}}, i = 1, \ldots, m \tag{2.27}$$

where,

r_i^N represents the normalized value of residuals at i^{th} indices and

Ω_{ii} represents the i^{th} diagonal entry of the residual sensitivity matrix.

Once the normalized residuals are calculated, the information of erroneous measurement is reached by searching the maximum of the normalized residuals. The maximum of normalized residuals identifies the measurement, which has errors and biases the state estimation process. To eliminate the biasing problem that occurred due to the bad data, identified measurement in normalize residual test should be eliminated or corrected. In general, system operators choose to correct instead of eliminating since once the bad data is eliminated, whole matrices such as measurement Jacobian matrix, Gain matrix, etc., should be reformed. However, if the found erroneous measurement is corrected, then only a simple modification is required among those matrices, and it is a faster operation.

The steps for detecting and identifying the bad data among measurements set is given as;

- Solving the WLS-SE and obtain the estimated states
- With obtained estimated states cost of the objective function is calculated
- Check whether there is erroneous data or not
- If there is erroneous data, "Normalize Residual Test" is performed
- In "Normalize Residual Test," first residuals are found by using the estimated states and measurements
- Then hat matrix (*K*) is calculated, and then sensitivity matrix (*S*) is found
- Finally, with the calculation of normalized residuals using the sensitivity matrix, bad data is found among the measurement set.

In the state estimation process, the identification of the bad data consumes a significant amount of time if there are multiple bad data in the measurement set due to the calculation of an inverse operation mentioned in equation (2.17) and

performation of state estimation for each bad data. Therefore, with the help of sparse storage, time consumption can be reduced.

CHAPTER 3

SPARSE STORAGE

In the power system state estimation process, there are several storage techniques to improve the computational speed [5]. However, the most suitable and the most common one is the sparse storage method since matrices in state estimation have a super sparse structure, and eliminating the processes of zero elements during matrix calculation provides desired computational improvement. In terms of the sparse storage method, there are several different approaches to store non-zero elements of the matrices, but all those approaches are based on the same principle, which is creating linked lists for non-zero elements and using these linked lists to make matrix calculations [10-15]. Among those sparse storage methods, two methods are commonly used in the power system state estimation process, namely "Gustavson's Method" and "Knuth's Method," and with varying linked list creation in each method, the benefits from those can be further increased. Each of these methods has advantages and disadvantages over each other. However, in this thesis, "Knuth's Method" is used for a storage technique to decrease the computational time of the state estimation process since it has flexibility for matrix reformations during the iterations of the state estimation process.

In this Chapter, the details about the two main sparse storage techniques in state estimation will be given. The comparison between "Gustavson's Method" and "Knuth's Method" is given in the "Comparison of Sparse Storage Techniques" part.

3.1 Gustavson's Method

Gustavson's method is the most common sparse storage technique in terms of improving the computational performance of processes that have sparse structure matrices. In literature, it is called CRS (Compressed Row Storage), and the other

variation of "Row Storage" is called CCS (Compressed Column Storage) [11]. The usage of those methods varies in terms of programming language in such that, for columned-based programming languages, CCS should be used to utilize sparse storage, and for rowed-based programming languages, CRS should be used.

In Gustavson's method, the way of storing non-zero elements of matrices is creating the three vectors in linked list, which are value vector, row index vector and index vector. The sample of method can be seen as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n \times k} \tag{3.1}$$

where,

n is the row number and

k is the column number of the matrix.

In equation (3.1), the sample sparse matrix can be seen. Moreover, the formulation of sparsity calculation of matrix can be written as follows;

$$\delta = \frac{number\ of\ non-zeros}{number\ of\ total\ elements} \times 100 \tag{3.2}$$

By using equation (3.2) for the matrix in equation (3.1), the sparsity of the sample matrix is calculated as "32%". As seen, the non-zero elements place less than half of the total elements of the matrix. In power system applications, this sparsity percentage decreases with increasing system size.

For sparse matrices exampled in equation (3.1), column-based linked lists are created for Gustavson's method as follows;

$$
\begin{aligned}
value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\
column\ index &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\
index &= [1 \quad 3 \quad 5 \quad 6 \quad 7 \quad 9]_{1\times(n+1)}
\end{aligned} \tag{3.3}
$$

where,

m is the number of non-zero elements and

n is the row number.

Moreover, the same linked list with using row-based vectors for matrix in equation (3.1) can be written as follows;

$$
\begin{aligned}
value\ array &= [1 \quad 2 \quad 4 \quad 5 \quad 4 \quad 3 \quad 6 \quad 2]_{1\times m} \\
row\ index &= [1 \quad 4 \quad 5 \quad 2 \quad 3 \quad 1 \quad 5 \quad 2]_{1\times m} \\
index &= [1 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9]_{1\times(k+1)}
\end{aligned} \tag{3.4}
$$

where,

m is the non-zero elements and

k is the column number of the matrix.

In both CRS and CCS methods, the approach is based on creating a linked list for non-zero elements of matrices.

In the compressed row storage method, the way of creating a linked list is following non-zero elements in each row and storing their value in value array, column number in column index vector, and the number of non-zero elements for each row in index vector. For example, in equation (3.3), the number of non-zero elements in i^{th} row can be found by investigating the difference between (i^{th}+1) and i^{th} column of index array such that in the first row, there are two non-zero elements ($index(2) - index(1) = 2$), in second row there are two non-zero elements ($index(3) -$

$index(2) = 2$) etc. Therefore, with using these linked lists, non-zero elements in matrices can be stored, and also sparse stored matrices can be recreated.

Besides creating the linked lists, there are three main operations for sparse stored matrices which are;

- Adding an additional non-zero element to the matrix,
- Deleting a non-zero element from the matrix,
- Changing the value of non-zero elements of the matrix.

These operations are given detailed in the following sub-sections.

3.1.1 Adding a Non-Zero Element

According to [11], adding a non-zero element to existed linked list of Gustavson's method has three steps. Those steps are;

- Finding the location where the column number or row number of added value takes place in column index vector or row index vector depending on the utilized method
- Adding new value to value vector in found location
- Changing the total number of non-zero elements in the index array

The steps are visualized for matrix in equation (3.1) with the following equations.

$$\begin{bmatrix} 1 & 0 & 8 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.5}$$

The new added value is shown with red color in equation (3.5). The new value is added to the first row and third column of the matrix. After adding the new value, linked lists for row storage method in equation (3.3) and column storage method in (3.4) changed respectively as follows;

$$
\begin{aligned}
value\ array &= [1 \quad 8 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\
column\ index &= [1 \quad 3 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\
index &= [1 \quad 4 \quad 6 \quad 7 \quad 8 \quad 10]_{1\times(n+1)}
\end{aligned} \tag{3.6}
$$

$$
\begin{aligned}
value\ array &= [1 \quad 2 \quad 4 \quad 5 \quad 8 \quad 4 \quad 3 \quad 6 \quad 2]_{1\times m} \\
row\ index &= [1 \quad 4 \quad 5 \quad 2 \quad 1 \quad 3 \quad 1 \quad 5 \quad 2]_{1\times m} \\
index &= [1 \quad 4 \quad 5 \quad 7 \quad 9 \quad 10]_{1\times(k+1)}
\end{aligned} \tag{3.7}
$$

As seen in equations (3.6), (3.7), the value vector and column/row index vector size increased with the number of non-zero elements added to the matrix. However, index vector size does not change since it shows only the total number of non-zero elements in rows or columns for row storage or column storage, respectively, yet the corresponded values of the index array increase.

3.1.2 Deleting a Non-Zero Element

According to [11], deleting a non-zero element from existed linked list of Gustavson's method has three steps as follows;

- Finding the location where the column number or row number of deleted value takes place in column index vector or row index vector depending on utilized method,
- Deleting the desired value from the value vector by using the found location,
- Changing the total number of non-zero elements in the index array.

The steps are visualized for matrix in equation (3.1) as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.8}$$

In equation (3.8), the value in the fourth row and the first column is changed from "2" to "0" in other words, it is deleted from the linked list. The changed linked lists for matrix in equation (3.8) is written as follows;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 4 \quad 6]_{1\times m} \\ column\ index &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 4]_{1\times m} \\ index &= [1 \quad 3 \quad 5 \quad 6 \quad 6 \quad 8]_{1\times(n+1)} \end{aligned} \tag{3.9}$$

$$\begin{aligned} value\ array &= [1 \quad 4 \quad 5 \quad 4 \quad 3 \quad 6 \quad 2]_{1\times m} \\ row\ index &= [1 \quad 5 \quad 2 \quad 3 \quad 1 \quad 5 \quad 2]_{1\times m} \\ index &= [1 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7]_{1\times(k+1)} \end{aligned} \tag{3.10}$$

As seen in equations (3.9) and (3.10), the value vector and column/row index vector size decreased with the number of non-zero elements deleted from the matrix. However, index vector size does not change since it shows only the total number of non-zero elements in rows or columns for row storage or column storage, respectively, yet the corresponded value of the index array decreases.

3.1.3 Changing the Value of Non-Zero Element

According to [11], changing a non-zero element in the linked list has two steps, and those steps are;

- Finding the location where the column number or row number of changed non-zero element takes place in column index vector or row index vector, respectively,

- Updating the value vector by using the found location in the first step.

The steps are visualized for matrix in equation (3.1) as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 10 & 0 \end{bmatrix}_{n\times k} \tag{3.11}$$

In equation (3.11), the existed non-zero element value in the fifth row and the fourth column is changed from "6" to "10". The new linked lists are formed as follows;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 10]_{1\times m} \\ column\ index &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ index &= [1 \quad 3 \quad 5 \quad 6 \quad 7 \quad 9]_{1\times(n+1)} \end{aligned} \tag{3.12}$$

$$\begin{aligned} value\ array &= [1 \quad 2 \quad 4 \quad 5 \quad 4 \quad 3 \quad 10 \quad 2]_{1\times m} \\ row\ index &= [1 \quad 4 \quad 5 \quad 2 \quad 3 \quad 1 \quad 5 \quad 2]_{1\times m} \\ index &= [1 \quad 4 \quad 5 \quad 6 \quad 8 \quad 9]_{1\times(k+1)} \end{aligned} \tag{3.13}$$

As seen in equations (3.12) and (3.13), the only changes occurred in the value vector since the change is made for existed non-zero element, and all other properties for non-zero elements in the linked list are kept same.

3.2 Knuth's Method

Besides Gustavson's method, there is another option for sparse storage named Knuth's method for the power system state estimation process. In Knuth's method, instead of three vectors to store information of non-zero elements, four vectors are utilized, namely as value vector, column/row vector, begin row/column vector, and next row/column vector. As Gustavson's method, Knuth's method also has two

options to store non-zero elements in row order or in column order depending on the programming language that is used for processes.

In order to visualize Knuth's method, the same matrix formed in equation (3.1) is used. The linked list that is created by utilizing column order and row order is formed as follows;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.14}$$

$$\begin{aligned} value\ erray &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ row\ array &= [1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5]_{1\times m} \\ nextC &= [6 \quad 8 \quad -1 \quad -1 \quad -1 \quad 7 \quad -1 \quad -1]_{1\times m} \\ begin\ column &= [1 \quad 3 \quad 5 \quad 2 \quad 4]_{1\times k} \end{aligned} \tag{3.15}$$

where,

$value\ array$: storing the non-zero values of the matrix (can be arbitrary order),

$column\ array$: Column index of the corresponding elements stored in value vector ($column\ array\ (i)$ is the column index of $value\ array\ (i)$)

$row\ array$: Row index of the corresponding elements stored in value vector (row $array\ (i)$ is the row index of $value\ array\ (i)$)

$nextC$: This array contains the pointer to the next non-zero element location in the same row ($nextC\ (i) = z \Rightarrow value\ array\ (z)$ is the next non-zero element of $value\ array(i)$)

$nextR$: This array contains the pointer to the next non-zero element location in the same column ($nextR\ (i) = z \Rightarrow value\ array\ (z)$ is the next non-zero entry of $value\ array\ (i)$)

$begin\,row$: This array contains the pointers to the beginning of each row ($begin\,row\,(i) = z$, first non-zero entry of row *i* is $value\ array\ (z)$)

$begin\,column$: This array contains the pointer to the beginning of each column ($begin\,column\,(i) = z$, first non-zero entry of column *i* is $value\ array\ (z)$)

m is the number of non-zero elements in the matrix

n is the total column number, and k is total the row number of the matrix

To form linked lists given in equations (3.14) and (3.15) for sparse matrix operations, there are three main processes to be considered. Those processes are;

- Adding the additional non-zero element to the matrix,
- Deleting the non-zero element from the matrix,
- Changing the non-zero element of the matrix.

3.2.1 Non-Zero Element Addition

Adding an additional non-zero element to the linked list is more complicated than "Gustavson's Method" since linked lists in "Knuth's Method" can be formed arbitrarily [14]. Therefore, the operation for reforming linked lists varies with the location of the newly added non-zero element. In order to visualize the reformation of linked lists, the row order method is used. The processes are the same for the column order method as well.

In order to visualize the reformation of linked lists, the used matrix for adding a non-zero element at the beginning of the row is given as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 8 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.16}$$

As seen in the matrix in equation (3.16), the new non-zero element "8" is added to the second row and first column.

When the non-zero element is added at the beginning of the row, then the following process is performed the update linked lists.

$$\begin{aligned} &value\ array\ (m+1) = M \\ &column\ array\ (m+1) = j \\ &nextR\ (m+1) = begin\ row\ (i) \\ &begin\ row\ (i) = m+1 \end{aligned} \tag{3.17}$$

where,

m is the number of the non-zero element before the new non-zero element,

i is the row number of newly added non-zero element,

j is the column number of new non-zero element and

M is the value of the non-zero element.

With utilizing the equation (3.17), the linked lists in equation (3.14) are updated as follows;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6 \quad 8]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4 \quad 1]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1 \quad 3]_{1\times m} \\ begin\ row &= [1 \quad 9 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.18}$$

The used matrix for adding a non-zero element neither the first entry nor the last entry is given as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 8 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.19}$$

In the matrix in equation (3.19), the new non-zero element is added between the first column and fourth column of the fifth row. In order to update such cases, the following procedure was performed.

$$\begin{aligned} &value\ array\ (m+1) = M \\ &column\ array\ (m+1) = j \\ &nextR\ (m+1) = \ prev \\ &nextR\ (prev) = m+1 \end{aligned} \tag{3.20}$$

where,

$prev$ is the previous non-zero element index in the same row of newly added non-zero element at value array.

Therefore, with utilizing equation (3.20), the linked lists created in equation (3.14) are updated as follows;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6 \quad 8]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4 \quad 3]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 9 \quad -1 \quad 8]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.21}$$

Finally, the used matrix for adding a non-zero element to the end of the row visualized as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 8 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.22}$$

$$\begin{aligned} &value\ array\ (m+1) = M \\ &column\ array\ (m+1) = j \\ &nextR\ (m+1) = -1 \\ &nextR\ (prev) = m+1 \end{aligned} \tag{3.23}$$

In the matrix given in equation (3.22), the non-zero element is added at the end of the first row. For such cases, the process in equation (3.23) is performed to update the linked lists.

With the utilization of equation (3.23), linked lists created in equation (3.14) can be reformed as follows;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6 \quad 8]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4 \quad 5]_{1\times m} \\ nextR &= [2 \quad 9 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1 \quad -1]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.24}$$

3.2.2 Deleting a Non-Zero Element

Besides “Gustavson’s Method”, deleting a non-zero element from the matrix is different in “Knuth’s Method.” During the deletion process of a non-zero entry in “Knuth’s Method”, deleted value is kept in linked lists, however, with the utilization of the “$nextR$” vector or “$nextC$” vector, the value that deleted from the matrix is skipped during the linked list search. There are three main consideration of deletion processes which are,

- Deleting a non-zero element from the beginning of a row,

- Deleting a non-zero element from neither beginning nor end of a row,
- Deleting a non-zero element from at the end of the row.

To visualize the mentioned processes, the matrix in equation (3.1) is changed with the corresponding deletion process.

The first problem is deleting a non-zero element from the beginning of a row. The changed matrix is given as follows;

$$\begin{bmatrix} 0 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.25}$$

In order to reform the linked list in equation (3.14), the utilized step is given as follows;

$$begin\ row\ (i) = nextR\ (prev) \tag{3.26}$$

By using equation (3.26), the created linked lists are reformed. The changed linked lists are;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1]_{1\times m} \\ begin\ row &= [2 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.27}$$

When utilizing linked search to recreate a matrix by using the linked lists in equation (3.27), it is seen that the first value of the first row is skipped. In other words, it is deleted from a matrix.

The second problem and third problem have the same approach for deleting a non-zero element from the middle of a row or the non-zero element at the end of the row.

Therefore, the following matrix was created to visualize deleting a non-zero element from at the end of the row.

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \end{bmatrix}_{n\times k} \tag{3.28}$$

The process for updating linked lists can be written as follow;

$$nextR\ (prev) = nextR\ (i) \tag{3.29}$$

By using the equation (3.29), the created linked lists in equation (3.14) changed.

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.30}$$

As seen in equation (3.30), the end of the fifth row, which is the value of "6", is eliminated during the linked list search of the matrix recreation process.

3.2.3 Changing a Non-Zero Value

Changing a non-zero value is another process in "Knuth's Method." In order to change the desired non-zero value in the matrix, the linked list search was performed. During the process of linked list search, when the index of column number for changed value is found in column array, the value is changed to the desired value in value array at found index. For example, the first column at the first-row entry change from "1" to "10" for the matrix in equation (3.1).

$$\begin{bmatrix} 10 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{3.31}$$

For matrix in equation (3.31), the linked lists are changed as follows;

$$\begin{aligned} value\ array &= [10 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \end{aligned} \tag{3.32}$$

3.3 Comparison of Sparse Storage Techniques

Sparse storage algorithms are formed according to need, and each of these sparse storage techniques has own advantages and disadvantages. The advantages of "Gustavson's Method" are;

- The process of building a linked list and recreation of the matrix from a linked list is easy,
- It takes up less memory space,
- The computational speed for adding, deleting, and changing a non-zero entry is high.

However, there is one significant disadvantage of "Gustavson's Method" for power system state estimation. The disadvantage is;

- Linked list cannot be reformed if the new column or row is added.

In power system state estimation, several matrices, such as seen in equations (2.1) and (2.7), are formed in random order. Therefore, "Gustavson's Method" does not meet the requirements for the state estimation process. In order to have the ability to

create linked lists in arbitrary order, the “Knuth’s Method” becomes the best suitable solution. The advantages of “Knuth’s Method” are;

- Linked lists can be formed in arbitrary order,
- The flexibility of adding/deleting a column or row to a matrix.

The disadvantages of “Knuth’s Method” are;

- It consumes more memory space,
- The implementation is complicated.

Although “Knuth’s Method” desires high memory space, with the help of the significant improved memory technology in recent years, this disadvantage of the “Knuth’s Method” vanished. Therefore, the consumption of memory space is no longer to be considered as an important factor. Hence the “Knuth’s Method” is utilized to build a sparse storage library that contains all matrix operations such as multiplication, Cholesky Factorization, and inverse of a matrix in this thesis. The detailed information for these operations is given in Chapter 4.

CHAPTER 4

THE PROPOSED METHOD

In previous chapters, background information of "WLS State Estimator, Bad Data Detection and Identification" are given. The importance of using the sparse storage techniques for these processes were mentioned. After that, the sparse storage methods namely Gustavson's Method and the basis of the proposed method which is Knuth's Method", were explained in detail.

According to "Knuth's Method," there are two options for building a linked list of a matrix. The first one is using the column method, and the second one is using the row method. In this thesis, to obtain a faster sparse library, these two methods combined and utilized a new linked list containing all seven vectors mentioned in Chapter 3.2. According to the following matrix, the example linked list that is used for the sparse storage technique is written as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{4.1}$$

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ row\ array &= [1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1]_{1\times m} \\ nextC &= [6 \quad 8 \quad -1 \quad -1 \quad -1 \quad 7 \quad -1 \quad -1]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad 6 \quad 7]_{1\times n} \\ begin\ column &= [1 \quad 3 \quad 5 \quad 2 \quad 4]_{1\times k} \end{aligned} \tag{4.2}$$

In equation (4.2), the new sparse storage technique is utilized to perform matrix operations mentioned state estimation operations. As it seen from the linked list that is created for the proposed method, the number of vectors in linked list are increased to keep additional information of non-zero entries. The required memory of the proposed method and Knuth's method are shown as below;

- Required memory of Knuth's Method is $(3 \times nnz \times m)$
- Required memory of The Proposed Method is $(5 \times nnz + m + n)$

where,

nnz is the total non-zero element number,

m is the row number of the matrix,

n is the column number of the matrix,

However, the use of additional space is not considered as a problem thanks to the improvements in "RAMs" technologies. Therefore, with the help of these additional informations linked list search time is further decreased.

In this way, the operations for matrices such as multiplication, addition, and subtraction are aimed to be accelerated. The implementation of matrix multiplication process is given in Chapter 4.2.

4.1.1 Adding a Non-Zero Element

In order to visualize the reformation of linked list for the proposed method, the used matrix for adding a non-zero element is given as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 8 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n \times k} \tag{4.3}$$

As seen in the matrix in equation (4.3), the new non-zero element "8" is added to the second row and first column. Addition a non-zero element to matrix is different in the proposed method since the proposed method is utilized both row ordered, and column ordered linked lists together. In example, when non-zero entry is added at the beginning of the row it does not mean that it must be for the beginning entry of the column. Therefore, to obtained linked list for the proposed method, all processes mentioned in Chapter 3.2.1 should be processed for vectors related with row ordered linked list and column ordered linked list separately. The reformed linked list for the prosed method when the new non-zero entry is added to matrix is shown below;

$$
\begin{aligned}
&value\ array = [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6 \quad 8]_{1\times m}\\
&column\ array = [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4 \quad 1]_{1\times m}\\
&\text{row array} = [1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \quad 2]_{1\times m}\\
&\text{nextC} = [9 \quad 8 \quad -1 \quad -1 \quad -1 \quad 7 \quad -1 \quad -1 \quad 6]_{1\times m}\\
&nextR = [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1 \quad 3]_{1\times m}\\
&begin\ row = [1 \quad 9 \quad 5 \quad 6 \quad 7]_{1\times n}\\
&begin\ column = [1 \quad 3 \quad 5 \quad 2 \quad 4]_{1\times k}
\end{aligned}
\tag{4.4}
$$

As it seen in equation (4.4), to update the linked list, the process of adding a non-zero entry to beginning of the row in Chapter 3.2.1 is followed for *"$column\ array$, $nextR$,* and *$begin\ row$"*, and the process of adding a non-zero element neither the first entry nor the last entry in Chapter 3.1.1 is followed for "$row\ array$, $nextC$ and $begin\ column$".

4.1.2 Deleting a Non-Zero Element

In order to visualize the reformation of linked list for the proposed method, the used matrix for deleting a non-zero element at the beginning of the row is given as follows;

$$\begin{bmatrix} 1 & 0 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 2 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 6 & 0 \end{bmatrix}_{n\times k} \tag{4.5}$$

As seen in the matrix in equation (4.5), non-zero element "2" was deleted from the fourth row and first column of matrix in equation. The deletion process of a non-zero entry from the matrix has same considerations such as when non-zero entry is deleted from the beginning of the row it does not mean that it must be located at the beginning of the column. Therefore, the processes mentioned in Chapter 3.2.2, should be applied separately for vectors related with row ordered link list and column ordered link list. The reformed linked list for the prosed method when the non-zero entry is deleted from the matrix is shown below;

$$\begin{aligned} value\ array &= [1 \quad 3 \quad 5 \quad 2 \quad 4 \quad 2 \quad 4 \quad 6]_{1\times m} \\ column\ array &= [1 \quad 4 \quad 2 \quad 5 \quad 3 \quad 1 \quad 1 \quad 4]_{1\times m} \\ row\ array &= [1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5]_{1\times m} \\ nextR &= [2 \quad -1 \quad 4 \quad -1 \quad -1 \quad -1 \quad 8 \quad -1]_{1\times m} \\ nextC &= [7 \quad 8 \quad -1 \quad -1 \quad -1 \quad 7 \quad -1 \quad -1]_{1\times m} \\ begin\ row &= [1 \quad 3 \quad 5 \quad -1 \quad 7]_{1\times n} \\ begin\ column &= [1 \quad 3 \quad 5 \quad 2 \quad 4]_{1\times k} \end{aligned} \tag{4.6}$$

As it seen in equation (4.6), to update the linked list, the process of deleting a non-zero entry from the beginning of the row in Chapter 3.2.2 is followed for *"column array, nextR*, and *begin row"*, and the process of deleting a non-zero entry from neither the first entry nor the last entry in Chapter 3.2.2 is followed for "*row array, nextC* and *begin column"*.

In conclusion, to update linked list for the proposed method, when non-zero entry is added, deleted, or changed, all processes under Chapter 3.2 should be considered for related vectors separately.

4.2 Matrix Multiplication

Matrix multiplication process is the key operation in power system state estimation, since in each step of the state estimation, there is a multiplication process between super sparse matrices. Therefore, in order to speed up the operations, sparse storage plays a significant role. Before explaining the method that is used for the thesis, the matrix operation is visualized as follows;

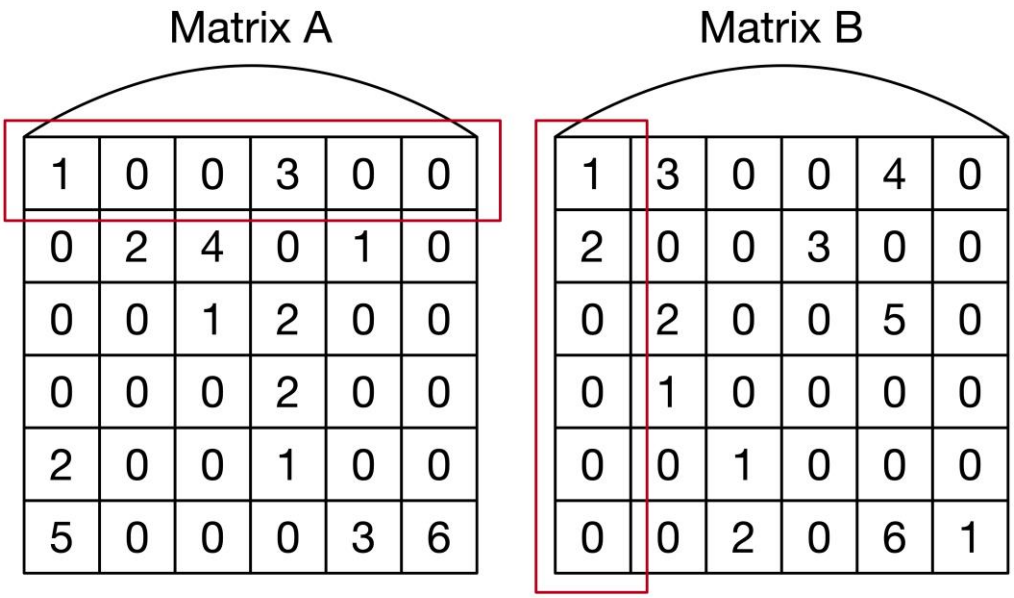

Figure 4.1: The visualization of matrix multiplication for first row first column

In matrix multiplication seen in Figure 4.1, the time complexity is $O(n^3)$ since each entry of matrix is considered during the process even the result is equal to "0". However, when the row method of "Knuth's Method" is used, the time complexity decreases from $O(n^3)$ to $O(n^2 \times m \times link\ list\ search\ time)$, where "$m$" is the total number of intersections of non-zero elements indexes between rows and columns. Linked list search time is the time for finding a required non-zero element in obtained linked lists. When the size of the matrix is increased, the time difference between normal operation and sparse storage technique increases as well. Therefore, sparse storage has a crucial role to obtain faster operations for bigger size of systems.

The visualization of the linked list search in the row ordered “Knuth’s Method” is given as follows;

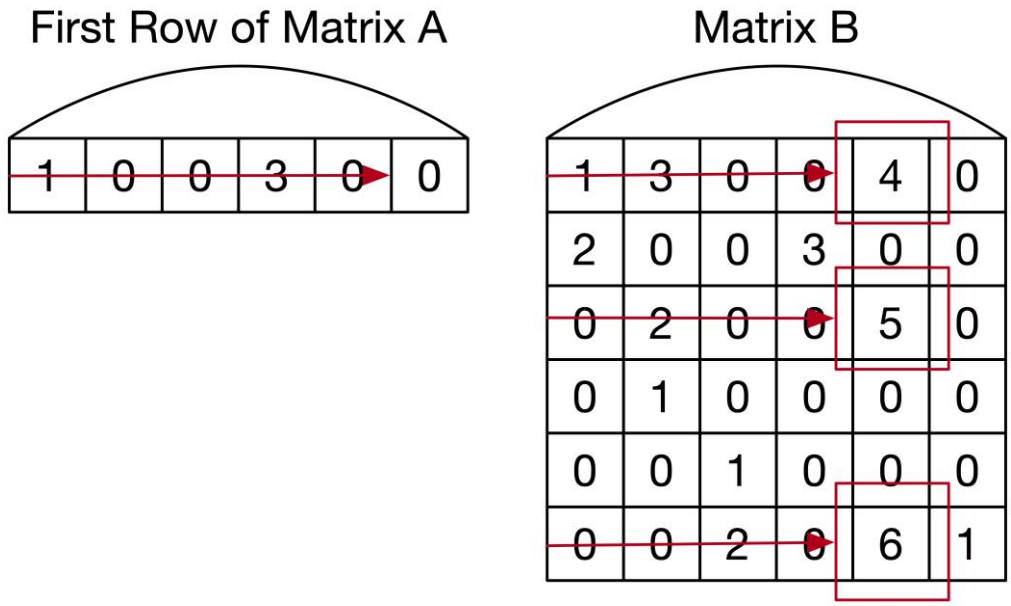

Figure 4.2: The linked list search of row order “Knuth’s Method” in the multiplication of the first row of Matrix A and fifth column of Matrix B

As seen in Figure 4.2, to find the locations of non-zero elements in the fifth column of Matrix B, the linked list search is performed from the beginning for each row of the examined column. This issue makes additional time consumption, and the time consumption increases with increasing the size of matrices.

Thanks to the proposed strategy in equation (4.2) for sparse storage, the time consumption of the multiplication process is decreased further by decreasing the linked list search time. This time reduction is done by eliminating the extra linked list searches to find column indexes of each row of Matrix B in the multiplication process by storing the non-zeros with row order and column order at the same time. The visualization of the multiplication process with the proposed sparse storage technique is given as follows;

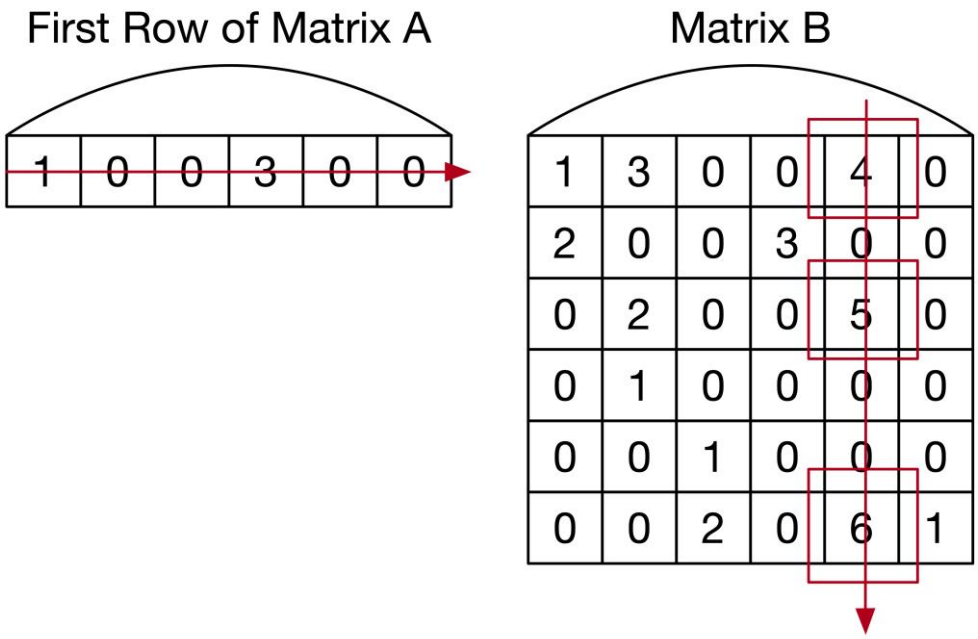

Figure 4.3: The linked list search of the proposed method in the multiplication process of the first row of Matrix A and fifth column of Matrix B

In Figure 4.3, it is seen that the linked list search can be performed both in row order and column order. In other words, instead of searching each row to find whether there is a non-zero element or not in the examined column, the linked list search performed directly to the desired column to find row indexes of non-zero elements.

In that way, multiplication is performed if the column index of the non-zero element in a row of Matrix A is intersected with the row index of the non-zero element in a column of Matrix B.

To achieve this process following steps are performed for each iteration:

1. $prev\ row = 1$ (investigated row number = 1),
2. $prev\ column = 5$ (investigated column number = 5),
3. $column\ array\ (prev\ row) = 1$,
4. $row\ array\ (prev\ column) = 1$,

5. If column of the value and row of the value are same, multiplication occurs,
6. $column\ array\ (nextR\ (prev\ row)) = 4$,
7. $row\ array\ (nextC\ (prev\ column)) = 3$
8. If $column\ array\ (nextR\ (prev\ row)) = 4 >$ $row\ array\ (nextC\ (prev\ column)) = 3$, $prev\ column$ value updated as $prev\ column = nextC\ (prev\ column)$,
9. New value of $row\ array\ (prev\ column) = 6$,
10. If $column\ array\ (nextR\ (prev\ row)) = 4 <$ $row\ array\ (nextC\ (prev\ column)) = 6$, $prev\ row$ value updated as $prev\ column = nextR(prev\ row)$,
11. When value of $colum\ array\ (prev\ row)$ and $row\ array\ (prev\ column)$, multiplication process occurs,
12. When one of the $nextR\ (prev\ row)$ or $nextC\ (prev\ column)$ reaches to " -1" the multiplication process of that iteration terminates.

This approach decreases the time complexity of the linked list search algorithm further to obtain a faster multiplication process where the time complexities of the proposed method and Knuth's method for matrix multiplication operation are shown as below;

- The Proposed Method has $O(m \times \sum_{i=1}^{n}(nnz_i + D))$
- Knuth's Method has $O(m \times \sum_{i=1}^{n}(nnz_i + D \times nnz_i)$

where,

nnz_i is the total non-zero element number in i^{th} column,

m is the row number of the matrix A,

n is the column number of the matrix B,

D is the link list search time to find the non-zero entry for desired location.

4.3 Cholesky Decomposition

As seen in equation (2.6), there is an inverse calculation of the "Gain Matrix", therefore, with applying the "Cholesky Decomposition" to "Gain Matrix," the solution time of equation (2.6) decreased with the elimination of the inversion operation. In literature, there are several types of factorization algorithms [23-25]. Among these algorithms, the most used decomposition algorithm is the Cholesky Decomposition for power system state estimation processes since it is a computationally cheap algorithm for positive definite symmetrical matrices, as the gain matrix used in WLS procedure. After applying the Cholesky decomposition process to the "Gain Matrix" mentioned in (2.13), the new equation can be written as follows;

$$LL^T\hat{x} = H^T(z - h(\hat{x})) \qquad (4.7)$$

where,

L is the lower triangular matrix of the Gain matrix,

L^T is the upper triangular matrix of Gain matrix,

z is the measurements and

$h(\hat{x})$ is the measurements which are created with measurement function by using the estimated states.

The visualization of the lower and upper triangular matrices of the gain matrix in Figure 2.3 is given below.

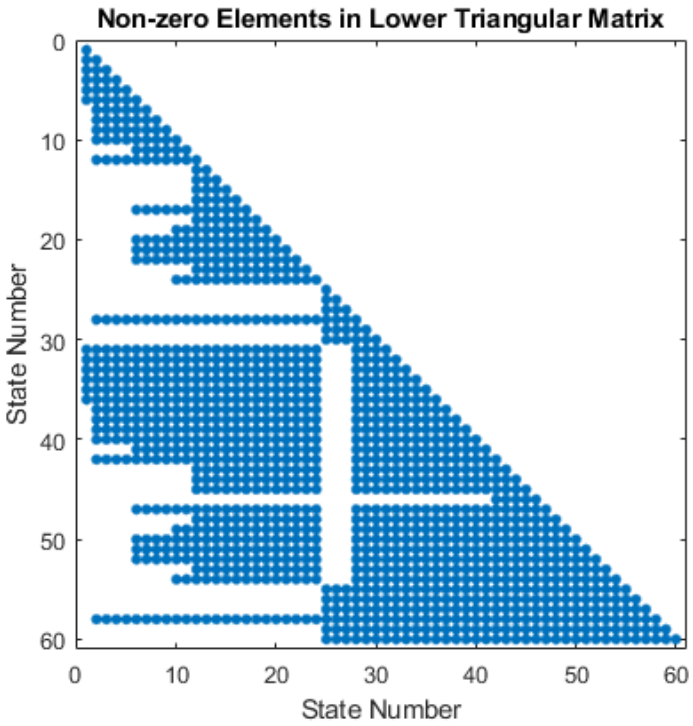

Figure 4.4: The lower triangular matrix of the gain matrix in Figure 2.3

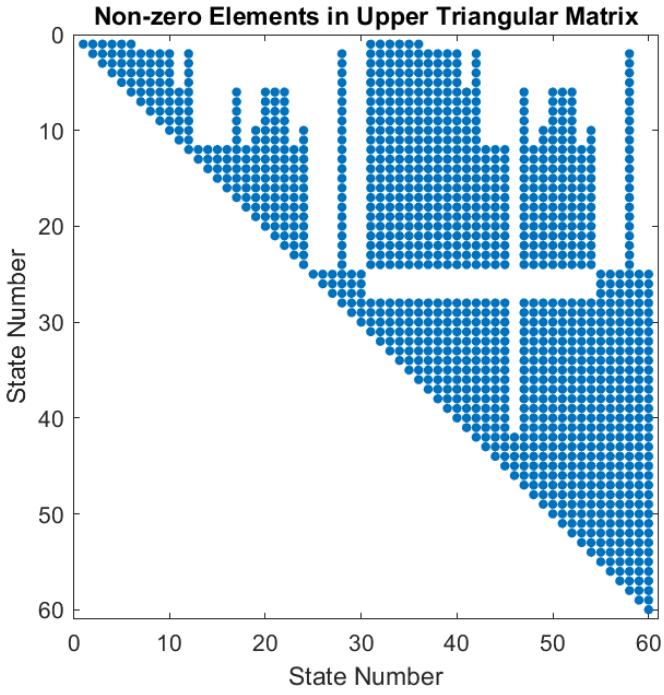

Figure 4.5: The upper triangular matrix of the gain matrix in Figure 2.3

There are several methods of computation of the "Cholesky Decomposition" to find lower and upper triangular matrices in the literature. However, according to [24], the following algorithm is one of the best algorithms in terms of time complexity, among others. The Doolittle's algorithm for "Cholesky Decomposition" is given in below.

Algorithm 1 Cholesky Decomposition (Doolittle's Algorithm)

1: ***for*** $j = 1{:}n$
2: ***if*** $j > 1$
3: $A(j{:}n, j) = A(j{:}n, j) - A(j{:}n, 1{:}j-1)A(j, 1{:}j-1)^T$
4: ***end***
5: $A(j{:}n, j) = A(j{:}n, j)/\sqrt{A(j, j)}$
6: ***end***

where,

n is the column number of matrix A.

After obtaining the lower and upper triangular matrices of the Gain matrix by using Algorithm 1, with the help of the forward and the backward substitution, the estimated states are calculated by using the following formula;

$$\begin{aligned} t &= H^T(z - h(\hat{x})) \\ L^T\hat{x} &= y \\ Ly &= t \end{aligned} \tag{4.8}$$

In equation (4.8), since *L*, L^T and *t* are known, *y* is calculated by using the forward substitution, and then, with the calculated *y*, estimated states are found by using the backward substitution.

4.4 The Matrix Inversion

The matrix inversion is the most time-consuming process among all other matrix operations such as multiplication, addition, subtraction, etc. Unfortunately, the state estimation process (bad data identification) contains an inverse operation during the calculation of the hat matrix in equation (2.18). However, as a result of the super sparse structure of the Gain Matrix, this time consumption, which occurrs due to the matrix inversion, can be eliminated by using the proposed sparse library. In literature, there are several approaches for inverse operation, however, in this thesis, the "Takahashi Method" is applied to the sparse algorithms to calculate matrix inversion [23]. The details of a "Takahashi Method" are given in Chapter 4.4.1.

4.4.1 Takahashi Method

According to [26], the Takahashi method utilizes the "LDU" factorization for computing the inverse of a given matrix. In order to calculate matrix inversion, there are two equations written as follows;

$$Z = D^{-1}L^{-1} + (I - U)Z \tag{4.9}$$

$$Z = U^{-1}D^{-1} + Z(I - L) \tag{4.10}$$

where,

A is the given matrix and A = *LDU* (*L, U, and D* are unit lower triangular, unit upper, and diagonal matrices, respectively) and

$Z = A^{-1}$.

According to [26], by utilizing the equations (4.9) and (4.10), some observations are made for positive definite symmetrical matrices. These observations are;

- The product of $(D^{-1}L^{-1})_{ii} = D_{ii}^{-1}$. This observation is used to eliminate calculation of the inverse of lower triangular matrix *"L,"*

- $(I - U)$ is the strictly upper triangular matrix since U is the unit upper triangular matrix,

Using these two observations, the Z matrix can be computed without calculating the L^{-1} matrix. The formulation to the calculation of inverse elements to the diagonal and upper triangular party of Z matrix can be written as follows;

$$z_{ij} = d_{ij}^{-1} \sum_{k>i}^{n} u_{ik} z_{kj} \quad for\ i \leq j \tag{4.11}$$

According to [26], the example of performing formulation of the equation (4.11), can be seen below.

$$A = \begin{bmatrix} x & 0 & x & x \\ 0 & x & x & 0 \\ x & x & x & 0 \\ x & 0 & 0 & x \end{bmatrix} \tag{4.12}$$

$$L + U = \begin{bmatrix} x & 0 & x & x \\ 0 & x & x & 0 \\ x & x & x & x \\ x & 0 & x & x \end{bmatrix} \tag{4.13}$$

$$\begin{aligned} z_{44} &= d_{44}^{-1} \\ z_{34} &= -u_{34} z_{44} \\ z_{33} &= d_{33}^{-1} - u_{34} z_{43} \\ z_{32} &= -u_{23} z_{33} \\ z_{22} &= d_{22}^{-1} - u_{23} z_{32} \\ z_{14} &= -u_{13} z_{34} - u_{14} z_{44} \\ z_{13} &= -u_{13} z_{33} - u_{14} z_{44} \\ z_{11} &= d_{11}^{-1} - u_{13} z_{31} - u_{14} z_{41} \end{aligned} \tag{4.14}$$

where,

A is the given matrix,

d_{ij} is the i^{th} row and j^{th} column of the diagonal matrix,

u_{ij} is the i^{th} row and j^{th} column of the unit upper triangular matrix and

red values are “fill-in” values that occurred during the decomposition process.

The used “Takahashi Algorithm” for matrix inversion is given as follows;

Algorithm 2 Takahashi Algorithm

```
1:for j = m × 2:−1:1
2:   for i = m × 2:−1:1
3:     if i == j
4:       value = 0;
5:       for k = i + 1:m × 2
6:           value = value + U(i,k) × Z(k,j);
7:       end
8:       Z(i,j) = (1/D(i,j)) − value;
9:       Z(j,i) = Z(i,j);
10:    elseif i < j
11:      value = 0;
12:      for k = i + 1:m × 2
13:          value = value − U(i,k) × Z(k,j);
14:      end
15:      Z(i,j) = value;
16:      Z(j,i) = Z(i,j);
17:    end
18:end
```

where,

m is the size of matrix A,

Z is the A^{-1},

U is the unit upper triangular matrix and

D is the diagonal matrix.

In equation (4.13), it is seen that, after the decomposition process applied to the given matrix, some non-zero elements, which do not exist in the given matrix, come up in lower triangular and upper triangular matrices. These non-zero entries create an additional time consumption during the matrix inversion process. Therefore, in the state estimation process, to decrease the number of fill-ins in matrices, the "Reverse Cuthill-McKee" algorithm is used to reorder bus numbers of system structure to centralize the non-zero entries around diagonals [27].

In that way, the number of the "fill-in" values in matrices are being reduced. In an example, when the "Reverse Cuthill-McKee" algorithm applied to the Gain matrix in Figure 2.3, the new structure of the matrix becomes as follows;

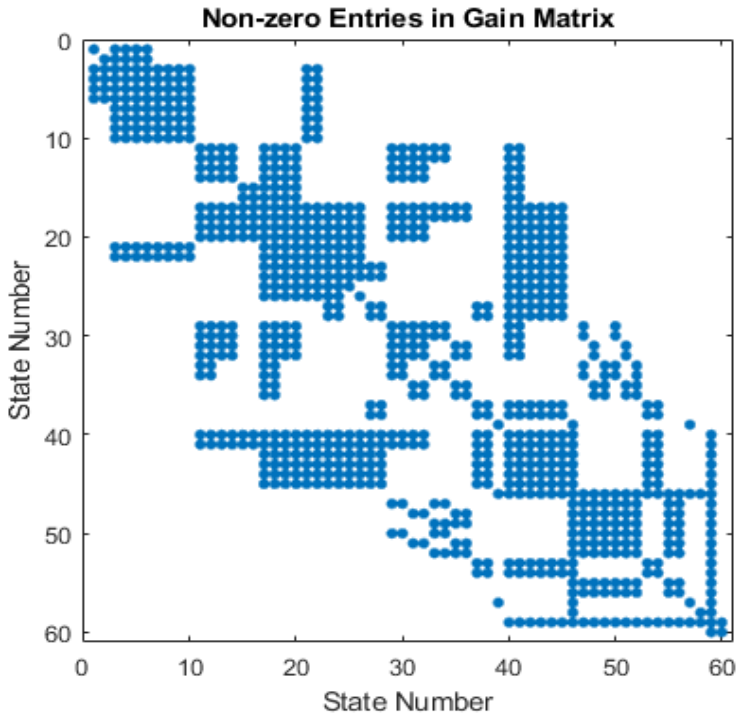

Figure 4.6: The Gain matrix after Reverse Cuthill McKee algorithm applied

Even the Gain matrix in Figure 4.6 and Figure 2.3 is the same matrix, when the decomposition applied, the lower and upper triangular matrix differs as follows;

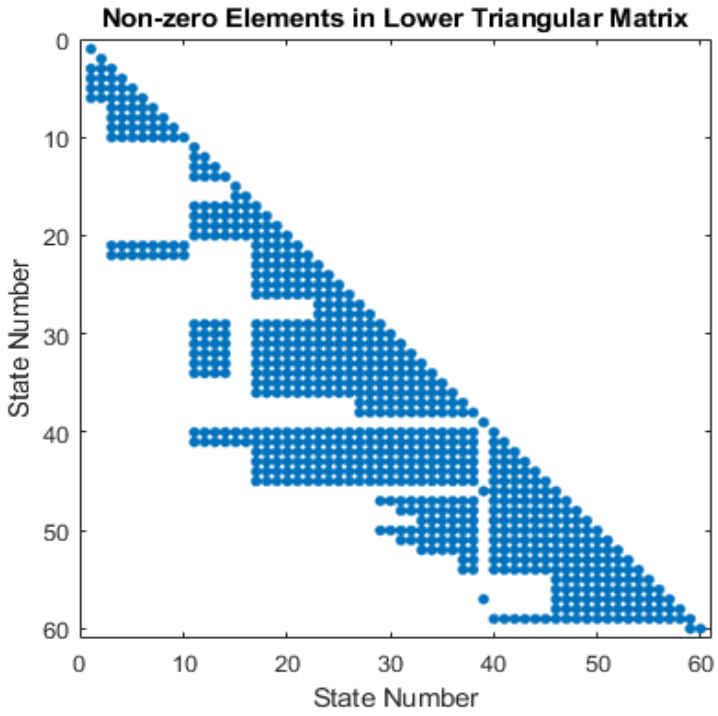

Figure 4.7: The lower triangular matrix of the Gain matrix after the decomposition process

In Figure 4.7, it is seen that the number of non-zero entries is less than the non-zero entries in Figure 4.4. Therefore, this example shows the importance of the ordering process. The details of the "Reverse Cuthill-McKee" algorithm can be found in [24].

According to [28], for further time improvement during the inversion operation of the matrix, only non-zero entry locations of Gain Matrix are calculated in equation (4.11). However, when system size increases, the fill-ins become inevitable. As a result of this, calculating only non-zero entry locations of the Gain matrix, leads to the wrong solution in the inverse operation of the matrix. Therefore, instead of using the non-zero entry locations of the Gain matrix, utilizing the non-zero entry locations of the lower triangular matrix of the Gain matrix gives the desired result since, during the decomposition process, fill-ins are considered. The test results of the proposed

method, which is the sparse storage applied to the Takahashi method using only the non-zero entries locations of the lower triangular matrix, the Takahashi method with calculating all entry, the built-in function of MATLAB, and conventional Takahashi method is given in Chapter 5.3.

CHAPTER 5

VALIDATION OF THE PROPOSED METHOD WITH STATE ESTIMATOR

In previous chapters the details of the operations in WLS State Estimation, Bad Data Analysis were given. Beside of these, to decrease the computational time during these operations, the importance of the sparse storage methods was mentioned. Moreover, the proposed method and the further improvements with the help of the proposed method for the matrix multiplication, Cholesky decomposition and the matrix inversion operation was given in detail.

In this thesis, to test the proposed strategy in real-life matrix operations, the state estimator is built. This state estimator contains the state estimation and bad data analysis. With the use of a built state estimator, all matrix operations mentioned in Chapter 4 are tested with the proposed sparse library, and the solution time of the proposed methods is investigated. The proposed method built in MATLAB environment with “Object Oriented” manner. However, MATLAB has still an issue of the solution time consumption of “Object Oriented” algorithms. In other words, when algorithm is implemented as “Object Oriented” in MATLAB environment it takes more time to be computed.

In this thesis, three main methods are proposed. These methods are;

- Sparse multiplication with the “Full Knuth’s Method”
- Sparse Cholesky Decomposition with the “Full Knuth’s Method”
- Sparse Takahashi matrix inversion with the “Full Knuth’s Method”

Besides these three main methods, a complete sparse library for the “Full Knuth’s Method” is established, which contains entry search, entry deletion, graph search algorithms, etc., to perform state estimation process in real life.

The solution time of state estimation and bad data analysis with built sparse library for IEEE 30-Bus system is given below [29].

Table 5.1: The solution time of proposed method of state estimation and bad data analysis process in IEEE 30-Bus system

System Size	*Density of H Matrix*	*Density of Gain Matrix*	*Solution Time of State Estimation (ms)*	*Solution Time of Bad Data Analysis (ms)*
IEEE 30-Bus	9%	29%	15.6	9.4

In Table 5.1, the solution time of the state estimation process and bad data analysis process for the IEEE 30-Bus system is shown. For the state estimation process, the solution time is dependent on the iteration number. For the IEEE 30-Bus system, system states converged to a threshold value in 10 iterations. Therefore, for each iteration, the solution time of the state estimation with the proposed method is equal to 2.56 ms. In bad data analysis of the IEEE 30-Bus system, the main time consumption is the matrix multiplication since in equation (2.18), during the calculation of hat matrix "*K,"* there is an inverse of the Gain matrix. The inverse of the Gain matrix is almost a full matrix, and this issue causes the extra linked list search time.

The solution time of the state estimation and bad data analysis for the IEEE 118-Bus system is given below.

Table 5.2: The solution time of proposed method of state estimation and bad data analysis process in IEEE 118-Bus system

System Size	*Density of H Matrix*	*Density of Gain Matrix*	*Solution Time of State Estimation (ms)*	*Solution Time of Bad Data Analysis (ms)*
IEEE 118-Bus	2%	10%	90.2	84.3

5.1 Test Results of Proposed Method in Multiplication Process

In the power system state estimation, the multiplication process has a crucial role for time consumption. Therefore, with the proposed method, this time kept as minimum as possible. The time consumption for these multiplication processes in state estimation is compared with the multiplication process without using the sparse storage technique seen in Algorithm 3. To compare the proposed method and the normal method, matrices are randomly created with different sizes and sparse density. In addition to that, the real 2383-bus Polish power system, real 3120-bus power system and 9241-bus power system grid are used to investigate the computation time difference between the proposed method with other methods [30]. The methods are tested with, Intel i9 9900 2.3 GHz 8 Core processor and 16 GB 2666 MHz RAM in MATLAB 2020b environment.

Algorithm 3 Conventional Matrix Multiplication

1: ***for*** $i = 1{:}m$
2: ***for*** $j = 1{:}n$
3: ***for*** k=1:m
4: $\mathrm{S}(i,j) = S(i,j) + A(i,k) \times A(k,j)$
5: ***end***
6: ***end***
7: ***end***

where,

m, n are row number and column number of matrix A respectively, and

S is the resulted matrix for multiplication.

Table 5.3: The time consumption of matrix multiplication using the proposed method, reference tool, and conventional multiplication.

Matrix Size	*Density of Matrix*	*The Proposed Method (ms)*	*Reference Tool (ms)*	*Algorithm 3 Conventional Algorithm (ms)*
(60x60)	1%	0.19794	0.03680	0.1450
(60x60)	5%	0.2662	0.04341	0.1449
(236x236)	1%	1.8	0.05003	7.2
(236x236)	5%	5.2	0.056652	9.6
(600x600)	1%	19.2	1.5	115.5
(2383x2383)	0.36%	508.4	76.8	22165.8
(3120x3120)	0.25%	1187.0	194.2	58623.4
(9241x9241)	0.12%	8502.3	4107.3	~

In Table 5.3, the solution times for the algorithm of the proposed method, the reference matrix multiplication and conventional algorithm are given. The algorithms are compared in MATLAB environment. However, according to [31], MATLAB uses “c++” for the built-in functions, and according to [32], the algorithms written in MATLAB are a few hundred times slower than the algorithms written with “c++” language. Moreover, built-in functions of MATLAB are well-optimized and uses all available cores to utilize the parallel processing. On the contrary, codes written in MATLAB environment are using only a single core to perform an algorithm.

Therefore, instead of directly comparing the computation time of reference tool with the proposed method, the scaling of computation time between two different matrix sizes for the reference tool and the proposed method should be compared.

In Table 5.3, it is seen that when the size increases, the solution time of conventional algorithm increases more than the proposed method and, it is seen that, computation

time of the reference tool, which is the built-in function of MATLAB, is approaching the computation time of proposed method, with increasing measurement size, which means that the scaling of the built-in functions is higher than the proposed method. Beside of that, for the conventional algorithm, when the matrix size increases the solution time of the matrix operation increases dramatically. Therefore, the computation time of the conventional algorithm is shown as "~" symbol.

In Table 5.3, when density increases, using sparse storage methods becomes meaningless since the time consumption of the number of linked list searches increases. However, thanks to matrix structures in the power system in real life, the density of matrices is less then the 1%. Therefore, using a sparse structure to hold matrices in linked list form is important.

5.2 The Test Results of the Proposed Method in Cholesky Decomposition Process

In the power system state estimation process, as seen in Figure 2.3, the sparsity of the Gain matrix is around 25%. However, when the size of the system increases, the sparsity of the matrix decreases less than the 1%. Therefore, it is important to use sparse storage methods to reduce the time consumption of non-zero elements in matrices.

The proposed strategy mentioned in Chapter 3.2 is applied to the "Cholesky Decomposition" algorithm given in Algorithm 1. After applying the sparse storage technique, the time results of built Cholesky Decomposition with sparse storage technique, the MATLAB built-in function for Cholesky Decomposition, and the Doolittle's algorithm given in Algorithm 1 are investigated and given in the table below.

In Table 5.4, it is seen that when the proposed sparse algorithm is applied to the "Cholesky Decomposition," the solution time decreases. As it is mentioned in Chapter 5.1, the built-in functions of MATLAB are processed with well-optimized

"c++" algorithm. As a result, there is a huge solution time gap between the proposed method and the built-in function of MATLAB, but the scaling of the built-in function between two matrix size is higher than the proposed method and the computation time of the built-in function is approaching the computation time of proposed method. Again, the computation time of the conventional algorithm is shown as "~" symbol since the computation time of the conventional algorithm increases dramatically when the matrix size increases.

Table 5.4: The time results of Cholesky Decomposition with the proposed method, MATLAB built-in function, and Doolittle's algorithm.

Matrix Size	*Density of Matrix*	*Proposed Method (ms)*	*Built-in Function (ms)*	*Algorithm 1 Doolittle's Algorithm (ms)*
(60x60)	1%	0.34	0.01745	0.9118
(60x60)	5%	0.36	0.01656	0.9043
(236x236)	1%	5.6	0.23671	13.1
(236x236)	5%	7.2	0.27640	14.3
(600x600)	1%	11.2	0.77697	85.2
(2383x2383)	0.36%	253.8	24.6	16533.1
(3120x3120)	0.25%	439.4	55.8	65732.1
(9241x9241)	0.12%	1023.1	950.5	~

5.3 Test Results of Proposed Method for Matrix Inversion

In bad data analysis of state estimation process, to calculate the hat matrix mentioned in equation (2.18), the proposed sparse method is applied to Algorithm 2 for Takahashi method. As opposed to Algorithm 2, the proposed method avoids calculating *"0"* values while performing the inversion process. The methods are

tested with different matrix sizes and matrix sparsity densities. The test results are given in below.

In Table 5.5, solution times of the inverse operation with different methods for different matrices are investigated. It is seen that the proposed method has better solution time comparing with the conventional matrix inversion. However, the proposed method still slower than the built-in function of MATLAB due to the programming language difference mentioned in Chapter 5.1. In addition, there is one more performance improvement with the proposed method, as seen in **Error! Not a valid bookmark self-reference.**. That performance improvement is achieved by calculating only the non-zero entry locations in the lower triangular matrix for the inverse of the Gain matrix instead of calculation all entries of the inverse of the Gain matrix.

Table 5.5. The time results of the proposed method, the built-in function of MATLAB, conventional matrix inversion, and calculation of all entries of the inverse of Gain matrix.

Matrix Size	*Density of Matrix*	*Proposed Method (ms)*	*Built-in Function (ms)*	*Algorithm 2 Conventional Takahashi Method (ms)*	*Sparse Method with Calculation All Entries (ms)*
(60x60)	1%	0.9042	0.0599	0.23875	0.9571
(60x60)	5%	1.8	0.067325	0.24427	2.4
(236x236)	1%	3.8	1.5	13.4	4.7
(236x236)	5%	5.6	1.5	19.9	30.9
(600x600)	1%	14.9	11.4	378.7	69.3
(2383x2383)	0.36%	2732.3	166.1	40342.2	65907.6
(3120x3120)	0.25%	5670.2	360.8	~	~
(9241x9241)	0.12%	9998.5	6401.9	~	~

CHAPTER 6

CONCLUSION

In order to meet the power demand of the customers, power systems are enlarging each year, and with the increasing system size new measurements are placed to gather data from the field for improve the situational awareness. In addition, PMUs are also deployed in power systems in the recent years. The high refresh rates of those devices creates an additional computational burden for the monitoring systems and energy management systems. Considering this situation, the SE has a crucial role in real time monitoring of the power system. Thanks to the sparse matrix structures of the SE applications, sparse matrix storage methods are utilized to improve the computational performance.

In the literature there are several types of sparse storage methods, however, power system operation has unique properties and hence, only few of sparse storage methods can satisfy the flexibility condition for power system SE. One of those proper techniques is the well-known Knuth's Method. Despite the widely known necessity for sparse storage in state estimation applications, there is no open-source sparse storage library.

In this thesis, the main purpose is to build the open-source sparse library for matrix operations and decrease the computation time of the matrix operations which are included in power system state estimation processes. Therefore, the proposed method, which is the full Knuth's Method is built, and with the help of the proposed method, the major time-consuming processes such as "Matrix Multiplication, Cholesky Decomposition and Matrix Inversion", are improved and the computation time of the overall SE process is decreased further with decreasing the linked list search time. In order to achieve the decrease the computation time of the linked list search process, Knuth's Method is enhanced with utilizing both the column ordered

method and row ordered method together which are utilized separately for sparse storage algorithms. By this way, undesired linked list searches are minimized during the matrix operations.

The main challenge encountered during the implementation of the proposed method is, storing the row ordered and column ordered linked list together for the results of the matrix operations since, the matrix operations are accomplished in one way, which is either row manner or column manner. This issue has been overcome and flexibility of the sparse storage for SE processes is achieved with the proposed method.

In this thesis three different algorithms were compared for main matrix operations in SE which includes “WLS-SE and Bad Data Analysis” processes. The results show that, the MATLAB built-in functions have lower computation time results than the proposed method in smaller size matrices, since MATLAB built-in functions are implemented with the “c++” language which is a few hundred times faster than the codes written directly in MATLAB environment. However, when the system size increases, it is seen from the results, the scaling of the proposed method is lower than the built-in functions of MATLAB. With increasing matrix sizes the results show that the computation time of the built-in functions becomes closer to the proposed method even the built-in functions are written with “c++” language. In addition, the importance of using sparse storage algorithms is revealed, since the computation times of normal matrix processing algorithms written in MATLAB environment are considerably higher than the proposed method.

In order to validate the proposed method, full state estimation process, which contains all type of matrix operations, is built and tested in different IEEE bus systems. During the tests, it was seen that the matrix operations work properly, and provides satisfactory computation time results for the full state estimation process considering the performance of MATLAB environment.

Note that, the proposed method is not well-optimized. In order to optimize the proposed method in future, first of all, parallel processing can be added as a feature

wherever it is appliable. Moreover, with investigating the properties of the state estimation process, further decrease in computation time can be achieved by utilizing the block calculations in matrix operations since the matrices matrix in state estimation processes has a specific shape. By this way, the solution time of matrix operations such as matrix multiplications, matrix addition, Cholesky Decomposition, matrix inversion etc. can be decreased dramatically. Finally, for the proposed method to reach its real capacity, the algorithms can be written in the "c++" language. In addition to that, in this thesis for ordering purposes the Reverse Cuthill McKee algorithm is used. To further improve the ordering process Tinney-2 algorithm can be implemented instead of Cholesky Decomposition process.

REFERENCES

[1] M. Meriem, C. Bouchra, B. Abdelaziz, S. O. B. Jamal, E. M. Faissal and C. Nazha, "Study of state estimation using weighted-least-squares method (WLS)," 2016 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM), 2016, pp. 1-5.

[2] M. Göl and A. Abur, "LAV Based Robust State Estimation for Systems Measured by PMUs," in IEEE Transactions on Smart Grid, vol. 5, no. 4, pp. 1808-1814, July 2014.

[3] L. Mili, M. G. Cheniae and P. J. Rousseeuw, "Robust state estimation of electric power systems," in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 41, no. 5, pp. 349-358, May 1994.

[4] A. Kumar and S. Chakrabarti, "ANN-based hybrid state estimation and enhanced visualization of power systems," ISGT2011-India, 2011, pp. 78-83.

[5] A. Monticelli and A. Garcia, "Fast decoupled state estimators," in IEEE Transactions on Power Systems, vol. 5, no. 2, pp. 556-564, May 1990.

[6] Expósito, A.G., & Abur, A. (2004). Power System State Estimation: Theory and Implementation (1st ed.). CRC Press.

[7] G. D'Antona and L. Perfetto, "Bad data detection and identification in power system state estimation with network parameters uncertainty," 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), 2015, pp. 26-31.

[8] E. Handschin, F. C. Schweppe, J. Kohlas and A. Fiechter, "Bad data analysis for power system state estimation," in IEEE Transactions on Power Apparatus and Systems, vol. 94, no. 2, pp. 329-337, March 1975, doi: 10.1109/T-PAS.1975.31858., J., Michel, J. and Westaway, R.W.C. 2000.

Neogene and Quaternary volcanics of southeastern Turkey. The Geological Society, London, Special Publications, 173,459-487

[9] A. Monticelli, "Reliable Bad Data Processing for Real-Time State Estimation," in IEEE Power Engineering Review, vol. PER-3, no. 5, pp. 31-32, May 1983.

[10] M. Shah, "Sparse Matrix Sparse Vector Multiplication - A Novel Approach," 2015 44th International Conference on Parallel Processing Workshops, 2015, pp. 67-73.

[11] Farzaneh, Aiyoub & Kheiri, Hossein & Abbaspour, Mehdi. (2009). An efficient storage format for large sparse matrices. Communications de la Faculté des Sciences de l'Université d'Ankara. Séries A1: Mathematics and Statistics. 58. 10.1501.

[12] R. C. Agarwal, F. G. Gustavson and M. Zubair, "A high performance algorithm using pre-processing for the sparse matrix-vector multiplication," Supercomputing '92:Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, 1992, pp. 32-41.

[13] Dongarra, Jack & Lumsdaine, Andrew & Niu, Xinhiu & Pozo, Roldan & Remington, Karin. (1997). A Sparse Matrix Library in C++ for High Performance Architectures. Proceedings of the Second Object Oriented Numerics Conference.

[14] Knuth, D. (1973). The Art Of Computer Programming, vol. 3: Sorting And Searching. Addison-Wesley.

[15] Java Sparse Matrix Library. Available: https://java-matrix.org/

[16] N. Hurley and S. Rickard, "Comparing Measures of Sparsity," in IEEE Transactions on Information Theory, vol. 55, no. 10, pp. 4723-4741, Oct. 2009.

[17] A. Krishnamoorthy and D. Menon, "Matrix inversion using Cholesky decomposition," 2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2013, pp. 70-72.

[18] Ntekim, OE & Esuabana, Ita & Edeke, Uwe. (2013). On Lu Factorization Algorithm With Multipliers. Global Journal of Mathematical Sciences. 12. 10.4314/gjmas.v12i1.3.

[19] T. P. Vishnu, V. Viswan and A. M. Vipin, "Power system state estimation and bad data analysis using weighted least squares method," 2015 International Conference on Power, Instrumentation, Control and Computing (PICC), 2015, pp. 1-5.

[20] E. Handschin, F. C. Schweppe, J. Kohlas, and A. Fiechter, “Bad data analysis for power systems state estimation,” IEEE Trans. Power App. Syst., vol. 94, pp. 329–337, Mar./Apr. 1975.

[21] A. Monticelli and A. Garcia, "Reliable Bad Data Processing for Real-Time State Estimation," in IEEE Transactions on Power Apparatus and Systems, vol. PAS-102, no. 5, pp. 1126-1139, May 1983.

[22] Y. Saad, ModiÖed from SPARSKIT: a basic tool kit for sparse matrix computations,(June6, 1994).

[23] J. Chen, K. Ji, Z. Shi and W. Liu, "Implementation of Block Algorithm for LU Factorization," 2009 WRI World Congress on Computer Science and Information Engineering, 2009, pp. 569-573.

[24] Golub, Gene H. & Van Loan, Charles F. (1983), “Matrix computations,” Baltimore: Johns Hopkins University Press.

[25] H. Yamashita and E. Nakamae, "A pivot ordering algorithm aimed at minimizing computation time," in IEEE Transactions on Circuits and Systems, vol. 25, no. 8, pp. 634-637, August 1978.

[26] Campbell, Yogin & Davis, Tim. (1995). Computing The Sparse Inverse Subset: An Inverse Multifrontal Approach.

[27] E. Cuthill and J. McKee, Reducing the bandwidth of sparse symmetric matrices, Proc. 24th Nat. Conf., ACM Publ. p. 69, 1122 Ave. of the Americas, New York, N.Y. 1969.

[28] B. Bilir and A. Abur, "Bad data processing when using the coupled measurement model and Takahashi's sparse inverse method," IEEE PES Innovative Smart Grid Technologies, Europe, 2014, pp. 1-5.

[29] Ali R. Al-Roomi (2015). Power Flow Test Systems Repository [https://al-roomi.org/power-flow]. Halifax, Nova Scotia, Canada: Dalhousie University, Electrical and Computer Engineering.

[30] R. D. Zimmerman, C. E. Murillo-Sanchez, Matpower (2021).

[31] MATLAB. (2010). version 7.10.0 (R2010a). Natick, Massachusetts: The MathWorks Inc.

[32] Andrews, Tyler. (2012). Computation Time Comparison Between Matlab and C++ Using Launch Windows.