AN INTERACTIVE TOOL TO INVESTIGATE THE INFERENCE PERFORMANCE OF NETWORK DYNAMICS FROM DATA

Veenadhar Katragadda

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2012

APPROVED:

Yan Wan, Major Professor
Shengli Fu, Committee Member
Kamesh Namuduri, Committee Member
Murali Varanasi, Chair of the Department of Electrical Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Mark Wardell, Dean of the Toulouse Graduate School

Veenadhar, Katragadda. An Interactive Tool to Investigate the Inference Performance of Network Dynamics from Data. Master of Science (Electrical Engineering), August 2012, 108 pp., 14 figures, references, 31 titles.

Network structure plays a significant role in determining the performance of network inference tasks. An interactive tool to study the dependence of network topology on estimation performance was developed. The tool allows end-users to easily create and modify network structures and observe the performance of pole estimation measured by Cramer-Rao bounds. The tool also automatically suggests the best measurement locations to maximize estimation performance, and thus finds its broad applications on the optimal design of data collection experiments. Finally, a series of theoretical results that explicitly connect subsets of network structures with inference performance are obtained.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr.Yan Wan for her guidance and support. As an advisor she always motivated me with her immense knowledge and made me enthusiastic to make my research successful. She is a great teacher and her classes are always fun and the way she present a topic is outstanding and her exams are always challenging and motivating. She is one of my favorite proffesors I met in my life.

Besides my advisor, I would like to thank Dr. Shengli Fu, Dr. Kamesh Namuduri, for their support as committee members of my thesis. A very special thanks to Dr. Xinrong Li for giving me insights of estimation thoery. I am so grateful for the support of my parents, my sister and my friends. Generally, I appreciate all the people I know for helping me grow up.

CONTENTS

CHAPTER 1

INTRODUCTION

Systems have become structurally more complex and larger in size in a variety of applications, such as autonomous network control, transportation management, sensor networking, and power network surveillance, among others. Many of these systems are characterized by interconnected network topologies. While extensive advances have been achieved in the analysis, evaluation, and design of these networked systems, the inference of network dynamics/structure from data has just started to gain attention [6, 24, 17]. This growing interst is motivated by the fact that network inference tasks are critical for a range of design and control problems in networks. For instance, epidemic control relies on the inference of propagation dynamics from noisy outbreak data [28]. Genetic engineering is contingent upon the identification of gene regulatory networks from experimental microarray data [18, 17, 4]. Securing power networks is concerned with designing power network structure that is difficult for malicious attackers to estimate the configuration and dynamics of power systems from local observations (so as to disturb the network or to steal power) [12]. These network inference applications have motivated two critical research needs: 1) designing smart data-collection experiments to better understand a network's behavior, and 2) securing a networked system effectively against malicious probings and attacks. I note that from dynamical systems point of view, security is concerned with preventing attackers from using observation data to detect a system's internal dynamics [30].

I find that the key step to address the above two needs is to understand the pivotal factors impacting the performance of inference tasks in networks. This knowledge can then be utilized to design strategies to enhance security or surveillance capability. Previous ef-

forts suggested that network topology has significant impact on the performance of inference tasks from noisy observation data [27]. In particular, estimation performance of system pole locations—as critical indicators of network dynamics—can be closely tied to network structure. However, mathematically capturing this structure dependence is nontrivial for general networks, as this problem is fundamentally concerned with directly relating eigenvectors, or equivalently zero-locations with network structure characteristics.

Computer-aided tools can significantly complement analytical limitations by providing automated analysis and design. The automated solutions can in turn provide rich insights to analytical discoveries. As such, in this thesis, I construct an interactive graphical user interface (GUI) tool that allows end users to easily modify a network's structure and observe the impact of such structural change on inference performance [15]. The tool is available to download at [1] with the interface shown in Figure 1.1. This stand-alone GUI tool has the following features:

1) End users can choose between reading the network structure from a file or directly drawing the network's graph structure in a plot area.
2) End users can select measurement noise levels and actuation and measurement locations to set up the network inference tasks. The tool then automatically displays the inference performance expressed in terms of Cramer-Rao bounds (CRB).
3) For any designated network structure, the tool displays the best measurement locations to observe network dynamics (e.g., with the tightest CRB). This information helps with the smart design of experiments to maximize inference performance.
4) End users can instantaneously observe the impact of structure change on inference performance, by directly modifying the graph structure through e.g., adding/removing nodes/edges, and dragging nodes.

Figure 1.1. Interactive tool: Graphical User Interface

In this thesis, I illustrate the functionalities of the tool, the theoretical background and algorithms for constructing the tool, and the analytical development of results obtained using the tool. In particular, I focus on the inference of system pole locations from noisy measurement data, and utilize the tool to help with: 1) evaluating the impact of network structural change on inference performance measured by CRB, 2) identifying classes of network structures whose structural characteristics can be explicitly linked to inference performance, and 3) designing measurement locations to maximize inference performance. Building upon our previous development [27], I develop these results using meshed control theory, graph theory, and estimation theory. I envision that this interactive tool and resulting theoretical findings will provide us rich insights into structure-exploiting experimental design for smart data collection and strategies to enhance security.

1.1. Overview of Thesis Content

In this thesis, I illustrate the functionality, construction and the use of the GUI tool in studying the relationship between network structure and estimation performance in terms of CRB. The thesis is organized as follows. In Chapter 2, I introduce the broad background on estimation theory that is related to the development of this work. Chapter 3 presents the functional description of the GUI. Chapter 4 presents the theoretical background behind the implementation of the tool. Chapter 5 contains examples on utilizing this tool for scientific discovery. In particular, the tool allows us to identify subclasses of graphs, for which I provide theoretical proofs on the explicit relationship of them with the estimation performance. Finally, Chapter 6 includes a brief conclusion and future work.

CHAPTER 2

BACKGROUND ON ESTIMATION THEORY AND CRAMER-RAO BOUNDS

This chapter provides the background on estimation theory, Cramer-Rao bounds and dynamical system theory that allow us to investigate the relationship between estimation performance and network structure.

2.1. Estimation Theory

In this section, I review the theory behind estimators, estimation process and how estimations are achieved in several applications. Estimation theory is an important subject in the fields of statistics and signal processing. Modern estimation theory is applied in a variety of signal processing applications. The goal of estimation theory is to extract some parameters from noisy observations and provide optimal estimates of these parameters. The process of estimation is different for different estimators. Estimator is a standard or a rule for measuring an estimate of a quantity based on observed data [5].

Estimators can be classified into two categories: *point estimators* and *interval estimators*. A point estimator yields a single value and the estimation involves calculating a single statistic which is the best estimate of the unknown parameter using the sample data. Minimum variance un-biased estimators (MVUE), best linear unbiased estimator (BLUE), minimum mean squared error estimator (MMSE) and maximum likelihood estimator (MLE) are the methods to derive point estimators [5].

Estimation error, *Cramer-Rao bound* and *bias* are typical metrics to evaluate the performance of point estimators [25]. The optimality of an estimate is determined by the *estimation error* which is the estimate itself minus the actual or true value of the parameter $e = \hat{\theta} - \theta$. If this error is high then the estimated value is not optimal and in contrast if the

error is zero then the estimate is a perfect optimal one. The variance on estimation error of a unknown parameter is defined by a lower bound called Cramer-Rao bound (CRB). A detailed discussion on CRB is given in the section 2.2. *Bias* is defined as the difference between the expected value of the parameter that is to be estimated and it's true value. If $\hat{\theta}$ serves as an estimator to an unknown parameter θ then,

$$Bias[\hat{\theta}] = \mathbb{E}[\hat{\theta}] - \theta = \mathbb{E}[\hat{\theta} - \theta] \tag{1}$$

If bias is zero then the estimator is un-biased. The estimator which is un-biased and having minimum . An estimate is said to be *consistent* if the mean-squared estimation error tends to zero as the number of observations increases $\lim_{L\to\infty} \mathbb{E}[e^T e] = 0$ [16] and the estimator with this behavior is *asymptotically unbiased*. The estimator which has minimum varience and bias = 0 (unbiased) is called minimum varience unbiased estimator (MVUE).

Interval estimator results in a range of possible values for an unknown parameter using the sample data. Single-valued result obtained from a point estimator can be expressed as a single function of a vector, whereas in interval estimators the results can be a range of multiple functions. I am not reviewing the details on interval estimator, as in this thesis, I am only interested in finding an MVU estimator, which falls into the category of point estimators.

The general estimation process is given by Figure 2.1. Major steps include:

- Determine probability distribution function (PDF) of the observed data. The PDF shows the dependence of the data on the parameters to be estimated.
- Develop an estimator for the above data model.
- Computer simulations can be done to test optimality and performance of the estimator.

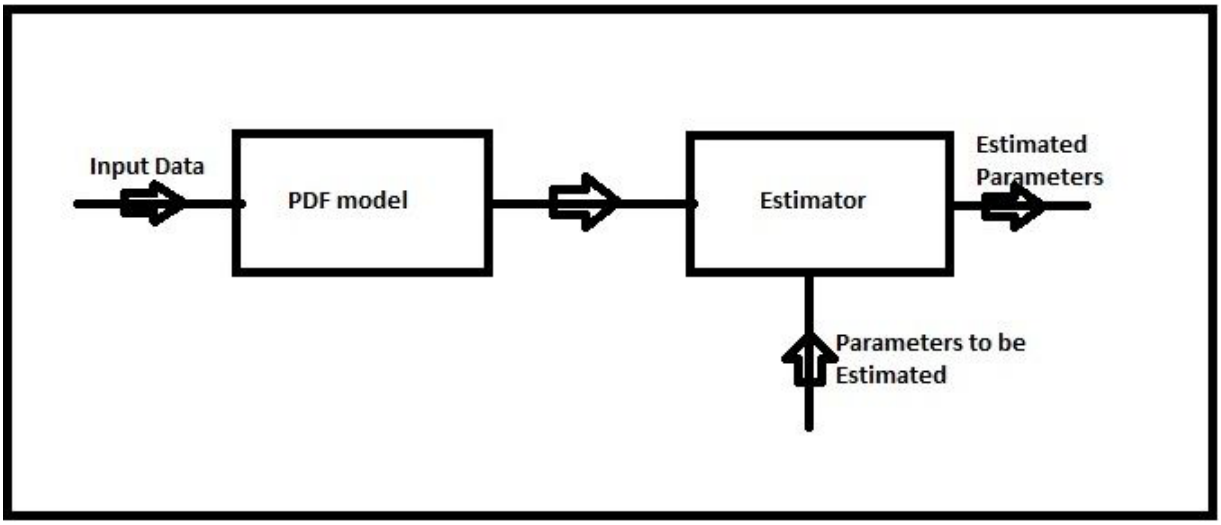

Figure 2.1. General estimation process

Estimation is performed in several applications including Radar, Sonar, Speech processing, Biomedicine, Control and Seismology. In all of these applications it is important to estimate a parameter or group of parameters based upon the experimental data which contains noisy random components. In Radar and Sonar, the interested parameters are the range and velocity of a target. In speech recognition, the parameter to be estimated can be some letter or word which is matched with the words already stored in the memory. In seismology the parameter to be estimated can be the distance of the underground oil deposit [16].

Pole estimation has its own significance in different applications. For example in signal processing, the poles of the harmonics are estimated to find the performance of detecting and tracking in a multi-channel setup [14]. The estimation in this work is performed using the total least square technique. The performance of the maximum likelihood estimator (MLE) estimating the time-of-arrival, amplitude parameters and poles from a noisy signal is studied in [9]. The work is motivated by underground acoustic testing where in the arrival time of the signal is unknown because of uncertainties. In this thesis, I am interested in developing

a tool (refer Chapter(3) for details on the tool) for studying the relation between network structure and the performance of estimator, estimating poles of the network. I find CRB's for this study on estimation performance (refer 2.2 for background on CRB).

2.2. Cramer Rao Lower Bound

In the previous section, I discussed that the minimum variance unbiased estimator is always an efficient estimator. In practice, all the estimators are not MVU and sometimes there can be a possibility where no MVU estimator exsists. The non-existence of MVU estimator is shown in Figure 2.2. I can see that for $\theta < \theta_0$, the estimator θ_2 has low variance and for $\theta > \theta_0$, the estimator θ_3 has low variance. Therefore, it is undecidable for an optimal estimator or there is no MVU estimator. *Cramer Rao Bound* (CRB) is the tool which helps in finding a MVU estimator. If an estimator reaches this bound, it is MVU and hence an optimum estimator.

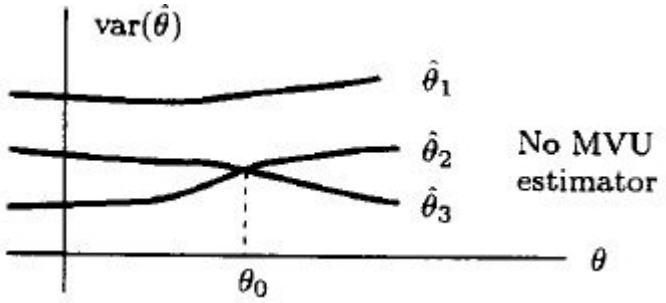

Figure 2.2. Variance of different estimators (From [16] topic: Existence of MVU estimator).

The CRB inequality or *information inequality* establishes the lower limit on how much information about an unknown probability distribution parameter, a set of measurements carries. In other words, it establishes the lower limit on *variance* of an estimator, estimating an unknown parameter θ having probability distribution $p(x;\theta)$. CRB says that the lower limit on variance of unbiased estimator is at least the inverse of the *Fisher information*, $I(\theta)$

(equation (4)) [20]. Fisher information is the variance of the *score* (V). Score is defined as the gradient (partial derivative) with respect to unknown parameter θ of the natural logarithm of likelihood function [11], given in equation (2).

$$V = \frac{\partial}{\partial\theta} log L(\theta; M) = \frac{1}{L(\theta; M)} \frac{\partial L(\theta; M)}{\partial\theta} \tag{2}$$

where L is the Likelihood function, M is the observation and θ is the unknown parameter.

To prove the inequality condition of CRB, let's assume g as an unbiased estimator and V is the score, then from *Cauchy-Schwartz* inequality [16] I have ,

$$(\mathbb{E}[(V - \mathbb{E}[V])(g - \mathbb{E}[g])])^2 \leq \mathbb{E}[(V - \mathbb{E}[V])^2]\mathbb{E}[(g - \mathbb{E}[g])^2] \tag{3}$$

As g is an Unbiased estimator $\mathbb{E}[g] = \theta$ further simplifying equation(3) will result equation(4) (See [16] Topic: Cramer Rao bound, for entire proof).

$$Var(g) \geq \frac{1}{I(\theta)} \tag{4}$$

where $I(\theta) = \mathbb{E}[V^2] = -\mathbb{E}[\frac{\partial^2 lnp(M;\theta)}{\partial\theta^2}]$ is the *Fisher Information*.

The conditions under which the bound is valid is, [11]

- The bound is only valid to un-biased estimators while the biased estimators violate the lower bound.
- Maximum Likelihood Estimators (MLE) achieve the lower bound as the size of the observation data increases.
- In practice the bound may be unreachable.

2.3. System Topology, Eigenvalues and Performance

Our efforts on developing an interactive tool is to study the connection between the *graph topology* and the estimation performance of the system. This section explains the relation between the *system topology*, *eigenvalues* and overall performance of the system. Several works have applied parameter estimation and system identification to study genetic networks in biological motivated applications, but the need for study on inference based on network/graph topology was commented [4]. Network structure plays an important role in studying inference performance. The topology of the network specifies about where in the network, measurements are to be made for definite inference of dynamics. Studying on the topological structure of the networks is encouraged in several fields. In *Medicine*, study on the brain regions which are interconnected by white matter tracts where in the network topology is important to understand the information sharing between brain regions and observe data for studying brain related diseases [13]. The work suggests role of network topology in studying the disease *Schizophrenia*. In *computer security*, analyzing the vulnerability issues based on graph structure are widely studied and several tools are developed which can detect network intrusion[21].

Now let us study the importance of poles in inference performance. Poles are the critical parameters indicating the performance and stability of the system. In this work, MLE of *poles* of single input single output (SISO) system from noisy data is considered (see chapter 4 for insights). The system dynamics are related to *pole- residue* or *pole-zero* form, which are shown in equations (5) and (6) [29].

$$H[z] = \sum_{i=0}^{n-1} \frac{A_i}{z - p_i} \tag{5}$$

where A_i are the residues and p_i are the poles.

$$H[z] = \frac{\prod_{i=1}^{m} z - z_i}{\prod_{i=0}^{n-1} z - p_i} \tag{6}$$

where z_i are the zeros and p_i are the residues. The location(value) of the pole determines the performance of the system. The following instances where poor performance of the system is observed are:-

- If the pole is nearer to zero, then numerator and the denominator of the transfer function given in equation(6) are nearly equal and so they get cancelled, resulting in poor dynamics and performance.
- If the residue A_i is nearly zero, then from the equation(5) the numerator goes nearly zero resulting poor dynamics.
- If poles are repeated, then it is hard to study the behaviour of the system and hence the performance of the system cannot be analyzed.

In biological applications, modeling of virus spread is related to the network structure and the allocation of the resources in these networks are studied for minimizing the virus spread [28]. In this work, the network structure relating to the spread, determines the system matrices whose elements contains all the network interactions. The largest eigenvalue of the system matrix specifies the rate of expansion/contraction of the spread. In mathematical terms, minimizing the spread of the virus is to minimize the dominant eigenvalue. In our work, the system matrix which is related to the network structure is *stochastic*. The largest eigenvalue of the system matrix is always '1' because of it's stochastic nature. As already discussed, if the poles are repeated, one cannot infer much about the estimation performance. So, I will consider the study on inference of estimation performance at second largest eigenvalue. Similar work

where in study on the second largest eigenvalue which accounts for the convergence rate of the network to infer consensus from network structure was done in [26]. Thus the system topology and the eigenvalues are critical in determining the estimation performance.

CHAPTER 3

FUNCTIONAL DESCRIPTION OF INTERACTIVE TOOL

In this chapter, I will describe in detail the functionality and interface of the GUI tool. As this tool is designed to assist with studying a network structure's impact on estimation performance, I integrate the following features into the tool: 1) The graph capturing network structure is displayed to allow an intuitive and visualized observation; 2) End users can directly modify the network structure through dragging nodes or adding/deleting edges/nodes and observe the change of estimation performance automatically; and 3) The best measurement location is automatically generated for any designated actuation locations.

The interface of the tool is composed of three parts: 1) the *Options* panel where end users can set up the network for estimation performance analysis, 2) the *Plot* panel where the graph capturing network structure is displayed, and 3) the *Output* panel where the system dynamics and a variety of information to help with interpreting estimation performance is displayed. Here let us describe each of these three panels in detail.

3.1. Options Panel

The *Options* panel allows users to select various input modes and set up the network for estimation performance analysis. This panel includes four major sub-panels: the *File* sub-panel, the *Plot Options* sub-panel, the *Settings* sub-panel and the *Scaling Factor* sub-panel.

3.1.1. File Sub-Panel

The *File* sub-panel allows end-users to select the source of network and save the current network structure. The drop-down menu has three options:

- *Plot a Network*: This is the default option. When this option is chosen, the user can draw a new network structure in the plot area.

- *Open Saved Network*: This option allows the user to load an existing network structure into the GUI. After the network structure is loaded, the associated graph will appear in the plot area, and the user can then freely modify it. The network structure is stored in a **.mat* format. In particular, the **.mat* file contains two matrices: $NodeLocation \in R^{n\times 2}$ and $Adjacency \in R^{n\times n}$. Matrix $NodeLocation$ stores the x location (first column) and y location (second column) of each node respectively. Matrix $Adjacency$ is the adjacency matrix of the network, storing the connectivity information among the nodes. For instance, if there is a directed edge from vertex i to j, then the $Adjacency$ matrix has a value '1' at the ith row and jth column entry, denoted as $Adjacency_{ij}$. Similarly, if there is an undirected edge between two vertices i and j, both the entries $Adjacency_{ij}$ and $Adjacency_{ji}$ are '1'.
- *Save Current Network*: This option allows the user to store the current network structure in the plot area into a **.mat* file. The format of the file is discussed above.

3.1.2. Plot Options Sub-Panel

This sub-panel allows end users to create and modify a network structure in the plot area. The sub-panel has three drop-down menus, namely *Draw/Delete/Drag*, *Vertex/Edge*, and *Directed/Undirected*. Here let us discuss the functionality for each combination of the selections defined by the three drop-down menus.

- *Draw + Vertex*: This combination allows the user to draw nodes in the plot area. Once the mouse is clicked, a node appears in the plot area and is labeled with a natural number starting from 1. The current x and y locations of the mouse appear at the top of the plot area to facilitate the creation of a desired structure.
- *Delete + Vertex*: With this combination, once the mouse is clicked, the vertex that is closest to the click and all edges connected to the vertex will disappear.

- *Draw + Edge + Directed/Undirected*: At this mode, when the user presses the mouse close to one vertex i, moves it to close to another vertex j, and then releases it, an edge between i and j is drawn in the undirected case, and from i to j is drawn in the directed case. The analysis developed in this thesis is only on the undirected network structures.
- *Delete + Edge + Directed/Undirected*: Similar to the above case, the edge that is the closest to the line defined by the press-release actions will be deleted.
- *Drag Nodes*: At this mode, when the user presses a node, moves it to another location and then releases it, the node is dragged to the new location and all edges connected to the node are moved simultaneously. The dragging capability is particularly useful to study the impact of network structural change on estimation performance.

3.1.3. Settings Sub-Panel

This panel allows the user to define actuation locations, measurement locations, and also the variance of Gaussian noise that the network is subject to.

- *Actuation Location*: The actuation location specifies the index of the node at which the network is actuated. In particular, I assume that a unit impulse input is applied at this location. Impulse-type inputs are widely used to capture short-time disturbances to systems (such as power failure in power grids, heat short to rats in biological experiments, etc.) [22, 23].
- *Measurement Location*: This is where the system response is measured. I assume that the network is measured at only one location. Moreover, an infinite length of sampled measurement sequence is available, so as to obtain the maximum estimation performance available at this measurement location.
- *Noise Variance*: I assume that the measurement is corrupted by a Gaussian noise

with the variance σ^2 specified here. As I will show in Section 4, the variance of noise scales proportionally with CRB.

3.1.4. Scaling Factor Sub-Panel

In this thesis, I consider typical spread dynamics [28], which requires the state matrix A to be a stochastic matrix that satisfies two properties: 1) row sum being 1, and 2) all the elements being in the range of 0 and 1 (i.e., $0 < A_{ij} < 1$). To ensure this requirement, a scaling process is required to convert distance-based edge weights into probability-like state matrix entries. This sub-panel displays the automatically selected scaling factor, and also allows the user to change it or fix its value so as to allow a fair comparison of estimation performance among networks subject to topological change. This sub-panel include three parts.

- *Display/Edit Box*: This component serves as a display box, showing the current scaling factor when *fix scaling factor* is unchecked. In this case, the scaling factor is automatically determined. It also serves as an edit box and allows the user to type the scaling factor when *Fix Scaling Factor* is checked.
- *Fix Scaling Factor Check-Box*: When this check box is checked, the scaling factor maintains the same despite all structural changes; otherwise it is determined automatically for each particular structure instance. This function is particularly useful when the user wishes to study the impact of structural change on estimation performance. Keeping the same scaling factor allows a fair comparison among different network structures. If the current scaling factor violates the properties of state matrices (e.g., with entries beyond the range of 0 to 1), the tool pops up an error message saying "Further change is not allowed because of the scaling constraint". When this occurs, the user can type a valid scaling factor directly in the *Display/Edit* box.

- *Ok button*: This button enables the typed scaling factor. All results are recalculated based upon the new scaling factor. The *Ok* button is only valid when *fix scaling factor* is checked.

3.2. Plot Panel

This panel includes the plot area and three buttons: *Enter*, *Undo*, and *Redraw*.

- *Enter*: After completing the network construction, pressing the *Enter* button displays the estimation performance results in the *Output* panel. At the mean time, the actuation, measurement, and best measurement locations are marked in blue, red, and green respectively. Moreover, edge weights are also displayed in the plot area.
- *Undo*: This button cancels the last change (e.g., draw/change/delete) to the network.
- *Redraw*: Pressing this button will pop up a window asking the user to delete the entire network and start with drawing a new network. Pressing *Yes* will clear everything and this operation cannot be undone.

3.3. Output Panel

This panel displays the network dynamics, and all results that assist users in assessing estimation performance. Specifically, the panel includes the following components:

3.3.1. System Dynamics Sub-Panel

The system dynamics associated with the network structure displayed in the plot area is represented in a state-space form (refer to [27] for more details):

$$\mathbf{x}[k+1] = A\mathbf{x}[k] + \mathbf{B}u[k] \tag{7}$$

$$y[k] = \mathbf{C}\mathbf{x}[k] + w[k],$$

where $\mathbf{x}[k] \in R^n$ are the state variables representing the responses at each of the n nodes in the network, $u[k]$ is a unit impulse signal, and $y[k]$ represents the measurement signal. As there is only one actuation location in the network, I indicate the actuation location in the vector $\mathbf{B} \in R^{n\times 1}$. Specifically, if the actuation is located at node i, $\mathbf{B}$ is a vector with its ith entry equal to 1, and all the other entries equal to 0. Similarly, $\mathbf{C} \in R^{1\times n}$ is a row vector, with its jth entry being 1 representing the measurement location, while all the other entries being 0. Moreover, $w[k]$ is the additive Gaussian noise with mean 0 and variance σ^2.

State matrix $A \in R^{n\times n}$ is a stochastic matrix that captures the network structure shown in the plot area. In particular, A is calculated from the network graph in the following. The tool first calculates the weight matrix $W \in R^{n\times n}$ where W_{ij} is the inverse of the distance between node i and node j. The Laplacian matrix L is then generated from W in such a way that each row sum of L equals 0. In another words, the diagonal entries of L equal the negative sum of all off-diagonal entries in the same row. Matrix L is further scaled with a scaling factor δ such that all entries of L are between 0 and 1. The default scaling factor δ is given in Equation (8).

$$\delta = \frac{1}{1 + max(diag(L))}, \tag{8}$$

where $diag()$ means placing the diagonal entries of L into a vector. Finally, A matrix is formed as $A = I - L$, where I is the identity matrix.

3.3.2. Pole-Residue Representation Sub-Panel

The pole-residue representation of the system transfer function $H[z]$ is the key toward the Cramer-Rao analysis, as it separates system poles as estimation parameters (see the detailed discussion in Section 4 and [27, 19]). In particular, the representation is in the

following format:

$$H[z] = \sum_{i=0}^{n-1} \frac{A_i}{z - p_i}, \tag{9}$$

where A_i are the residues and p_i are the poles of the network. All eigenvalues of A lies between -1 and 1, with the largest eigenvalue at 1.

3.3.3. Right Eigenvectors Sub-Panel

The sub-panel displays the right eigenvector associated with each eigenvalue. As seen from Section 5, estimation performance studies rely on algebraic graph theory results that relate eigenvectors with structural change [7, 3, 8].

3.3.4. Fisher Information Sub-Panel

This sub-panel displays the Fisher Information matrix $I(\theta)$. Fisher Information matrix specifies the level of uncertainty in the observation sequence y with respect to the unknown parameter vector θ (see [16, 19] for the details).

$$I(\theta) = -\mathbb{E}\left(\frac{\partial Inp(y;\theta)^T}{\partial \theta} \frac{\partial Inp(y;\theta)}{\partial \theta}\right) \tag{10}$$

where y is the observation signal, $p(y;\theta)$ is the probability function of y expressed in terms of θ, ^{T} represents matrix transpose, and $E()$ is the expectation operator.

3.3.5. Cramer-Rao Bounds for Poles Sub-Panel

Cramer-Rao bound specifies a lower bound on the variance of estimated parameter θ. System poles and their associated CRB are displayed in this sub-panel. CRB is obtained by inverting the Fisher Information matrix as shown in Equation (11) [16].

(11) $$var(\theta) \geq I(\theta)^{-1}$$

3.3.6. Efficient Observation Placement Sub-Panel

Given a fixed actuation location, this sub-panel provides the best measurement location to observe the second largest eigenvalue, or named the Fiedler eigenvalue [7]. The Fiedler eigenvalue is a critical indicator of system dynamics and has received extensive study (see e.g., [7, 8, 28, 2, 26]). For instance, it captures the mixing speed of a markov chain [2], the convergence speed of consensus building [26], and the spread speed of an epidemic [28]. The CRB associated with this optimal observation is also displayed.

CHAPTER 4

THEORETICAL BACKGROUND

In this chapter, I describe the theoretical foundations of the tool. In particular, I first show how the CRBs for pole estimations are obtained from the observation sequence. This analysis is based upon the pole-residue representation which relates system response (captured by the observation sequence) with pole locations. Moreover, in many applications, finding observation locations associated with the best estimation performance are interested. These locations represent the most effective places to ascertain system dynamics, or equivalently, the most vulnerable sites to release system dynamics if seized by adversaries. I thus also discuss the calculation of optimal measurement locations.

4.1. Fisher Information Matrix

In this section, I consider the system shown in Equation (7), and discuss how the Fisher information matrix for pole estimation can be obtained. Most of the analysis discussed here can be found in [27]. I summarize the results here for the completeness of our presentation.

I start from the pole-residue representation of system dynamics as shown in Equation (9). I denote the parameter of interest θ as a cascade of the parameters $A = [A_0\ A_1\ A_2\ ...]$ and $p = [p_0\ p_1\ p_2\ ...]$ in the pole-residue form. In particular, $\theta = [\ A\ \ P\ ]$. Moreover, as the output sequence $y[k]$ can be written as [27]

$$y[k] = \sum_{i=0}^{n-1} A_i p_i^k + w[k], \tag{12}$$

we can easily obtain the Fisher information matrix as [27, 19]

$$(13) \qquad F(\theta) = \frac{1}{\sigma^2}\frac{\partial y_j^T}{\partial \theta}\frac{\partial y_j}{\partial \theta} = \frac{1}{\sigma^2}\begin{bmatrix} \frac{\partial y_j^T}{\partial A} \\ \frac{\partial y_j^T}{\partial p} \end{bmatrix}\begin{bmatrix} \frac{\partial y_j}{\partial A} & \frac{\partial y_j}{\partial p} \end{bmatrix}$$

$$= \frac{1}{\sigma^2}\begin{bmatrix} 1 & p_0 & p_0^2 & p_0^3 & .... \\ 1 & p_1 & p_1^2 & p_1^3 & .... \\ . & . & . & . & .... \\ . & . & . & . & .... \\ 1 & p_{n-1} & p_{n-1}^2 & p_{n-1}^3 & .... \\ 0 & A_0 & 2A_0p_0 & 3A_0p_0^2 & .... \\ 0 & A_1 & 2A_1p_1 & 3A_1p_1^2 & .... \\ . & . & . & . & .... \\ . & . & . & . & .... \\ 0 & A_{n-1} & 2A_{n-1}p_{n-1} & 3A_{n-1}p_{n-1}^2 & .... \end{bmatrix}\begin{bmatrix} 1 & 1 & .... & 1 & 0 & 0 & .... & 0 \\ p_0 & p_1 & .... & p_{n-1} & A_0 & A_1 & .... & A_{n-1} \\ p_0^2 & p_1^2 & .... & p_{n-1}^2 & 2A_0p_0 & 2A_1p_1 & .... & 2A_{n-1}p_{n-1} \\ . & . & .... & . & . & . & .... & . \\ . & . & .... & . & . & . & .... & . \\ . & . & .... & . & . & . & .... & . \\ . & . & .... & . & . & . & .... & . \\ . & . & .... & . & . & . & .... & . \\ . & . & .... & . & . & . & .... & . \\ . & . & .... & . & . & . & .... & \end{bmatrix}$$

$$= \frac{1}{\sigma^2}\begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \end{bmatrix},$$

where

$$a_1 = \begin{bmatrix} (1+p_0^2+p_0^4+....) & (1+p_0p_1+p_0^2p_1^2+....) & .... & (1+p_0p_{n-1}+p_0^2p_{n-1}^2+....) \\ (1+p_0p_1+p_0^2p_1^2+....) & (1+p_1^2+p_1^4+....) & .... & (1+p_1p_{n-1}+p_1^2p_{n-1}^2+....) \\ . & . & .... & . \\ . & . & .... & . \\ (1+p_0p_{n-1}+p_0^2p_{n-1}^2+....) & (1+p_1p_{n-1}+p_1^2p_{n-1}^2+....) & .... & (1+p_{n-1}^2+p_{n-1}^4+....) \end{bmatrix}$$

$$(14) \qquad = \begin{bmatrix} \frac{1}{1-p_0^2} & \frac{1}{1-p_0p_1} & .... & \frac{1}{1-p_0p_{n-1}} \\ \frac{1}{1-p_0p_1} & \frac{1}{1-p_1^2} & .... & \frac{1}{1-p_1p_{n-1}} \\ . & . & .... & . \\ . & . & .... & . \\ \frac{1}{1-p_{n-1}p_0} & \frac{1}{1-p_{n-1}p_1} & .... & \frac{1}{1-p_{n-1}^2}, \end{bmatrix}$$

$$a_2 = \begin{bmatrix} (p_0A_0 + p_0^2(2A_0p_0) + ...) & (p_1A_0 + p_1^2(2A_0p_0) + ...) & .... & (p_{n-1}A_0 + p_{n-1}^2(2A_0p_0) + ...) \\ (p_0A_1 + p_0^2(2A_1p_1) + ...) & (p_1A_1 + p_1^2(2A_1p_1) + ...) & .... & (p_{n-1}A_1 + p_{n-1}^2(2A_1p_1) + ...) \\ . & . & .... & . \\ . & . & .... & . \\ (p_0A_{n-1} + p_0^2(2A_{n-1}p_{n-1}) + ...) & (p_1A_{n-1} + p_1^2(2A_{n-1}p_{n-1}) + ...) & .... & (p_{n-1}A_{n-1} + p_{n-1}^2(2A_{n-1}p_{n-1}) + ...) \end{bmatrix}$$

$$(15) \quad = \begin{bmatrix} \frac{A_0p_0}{(1-p_0^2)^2} & \frac{A_0p_1}{(1-p_0p_1)^2} & .... & \frac{A_0p_{n-1}}{(1-p_0p_{n-1})^2} \\ \frac{A_1p_0}{(1-p_1p_0)^2} & \frac{A_1p_1}{(1-p_1^2)^2} & .... & \frac{A_1p_{n-1}}{(1-p_1p_{n-1})^2} \\ . & . & .... & . \\ . & . & .... & . \\ \frac{A_{n-1}p_0}{(1-p_{n-1}p_0)^2} & \frac{A_{n-1}p_1}{(1-p_{n-1}p_1)^2} & .... & \frac{A_{n-1}p_{n-1}}{(1-p_{n-1}^2)^2} \end{bmatrix}$$

$$a_3 = \begin{bmatrix} (p_0A_0 + p_0^2(2A_0p_0) + p_0^3(3A_0p_0^2) + ...) & .... & (p_0A_{n-1} + p_0^2(2A_{n-1}p_{n-1}) + p_0^3(3A_{n-1}p_{n-1}^2) + ...) \\ (p_1A_0 + p_1^2(2A_0p_0) + p_1^3(3A_0p_0^2) + ...) & .... & (p_1A_{n-1} + p_1^2(2A_{n-1}p_{n-1}) + p_1^3(3A_{n-1}p_{n-1}^2) + ...) \\ . & .... & . \\ . & .... & . \\ (p_{n-1}A_0 + p_{n-1}^2(2A_0p_0) + p_{n-1}^3(3A_0p_0^2) + ...) & .... & (p_{n-1}A_{n-1} + p_{n-1}^2(2A_{n-1}p_{n-1}) + p_{n-1}^3(3A_{n-1}p_{n-1}^2) + ...) \end{bmatrix}$$

$$(16) \quad = \begin{bmatrix} \frac{A_0p_0}{(1-p_0^2)^2} & \frac{A_1p_0}{(1-p_0p_1)^2} & .... & \frac{A_{n-1}p_0}{(1-p_0p_{n-1})^2} \\ \frac{A_0p_1}{(1-p_1p_0)^2} & \frac{A_1p_1}{(1-p_1^2)^2} & .... & \frac{A_{n-1}p_1}{(1-p_1p_{n-1})^2} \\ . & . & .... & . \\ . & . & .... & . \\ \frac{A_0p_{n-1}}{(1-p_{n-1}p_0)^2} & \frac{A_1p_{n-1}}{(1-p_{n-1}p_1)^2} & .... & \frac{A_{n-1}p_{n-1}}{(1-p_{n-1}^2)^2} \end{bmatrix}$$

and

$$a_4 = \begin{bmatrix} (A_0^2 + (2A_0p_0)^2 + (3A_0p_0^2)^2 + ...) & .... & (A_0A_{n-1} + (2A_0p_0)(2A_{n-1}p_{n-1}) + ...) \\ (A_0A_1 + (2A_1p_1)(2A_0p_0) + (3A_1p_1^2)(3A_0p_0^2) + ...) & .... & (A_1A_{n-1} + (2A_1p_1)(2A_{n-1}p_{n-1}) + ...) \\ . & .... & . \\ . & .... & . \\ (A_0A_{n-1} + (2A_{n-1}p_{n-1})(2A_0p_0) + (3A_{n-1}p_{n-1}^2)(3A_0p_0^2) + ...) & .... & (A_{n-1}^2 + (2A_{n-1}p_{n-1})^2 + (3A_{n-1}p_{n-1}^2)^2 + ...) \end{bmatrix}$$

$$(17) \quad = \begin{bmatrix} \frac{A_0^2(1+p_0^2)}{(1-p_0^2)^3} & \frac{A_0A_1(1+p_0p_1)}{(1-p_0p_1)^3} & \cdots & \frac{A_0A_{n-1}(1+p_0p_{n-1})}{(1-p_0p_{n-1})^3} \\ \frac{A_0A_1(1+p_0p_1)}{(1-p_0p_1)^3} & \frac{A_1^2(1+(p_1)^2)}{(1-p_1^2)^3} & \cdots & \frac{A_1A_{n-1}(1+p_1p_{n-1})}{(1-p_1p_{n-1})^3} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \frac{A_0A_{n-1}(1+p_0p_{n-1})}{(1-p_0p_{n-1})^3} & \frac{A_1A_{n-1}(1+p_1p_{n-1})}{(1-p_1p_{n-1})^3} & \cdots & \frac{A_{n-1}^2(1+p_{n-1}^2)}{(1-p_{n-1}^2)^3} \end{bmatrix}$$

4.2. Cramer-Rao Bounds for Pole Estimation

The CRB for the estimate of parameter θ_i is the ith diagonal entry of the inverse of the Fisher information matrix:

$$(18) \qquad C_i = (F(\theta)^{-1})_{ii}.$$

Of our particular interest, the CRB for the ith pole p_i (denoted as $C(p_i)$) equals C_{n+i+1}. Because the calculation of CRB is involved with inverting a large matrix that is prone to be ill-conditioned, I instead find a lower-bound of the CRB. In particular, the CRB associated with the ith pole is bounded by [27]

$$(19) \qquad C(p_i) \geq \left(\frac{1}{\sigma^2} \ \frac{A_i^2(1+p_i^2)}{(1-p_i^2)^3} \right)^{-1} = \sigma^2 \frac{(1-p_i^2)^3}{A_i^2(1+p_i^2)}$$

The above result shows that CRB scales with the variance of Gaussian noise. Moreover, the residues play an important role in the performance of pole estimation. Generally, the poles associated with large residues are easier to estimate than the ones with smaller residues [27].

4.3. Effective Sensor Placement

In many sensor network applications such as bridge health monitoring and environmental monitoring, sensors are placed ad hoc. However, as noted recently, smartly choosing

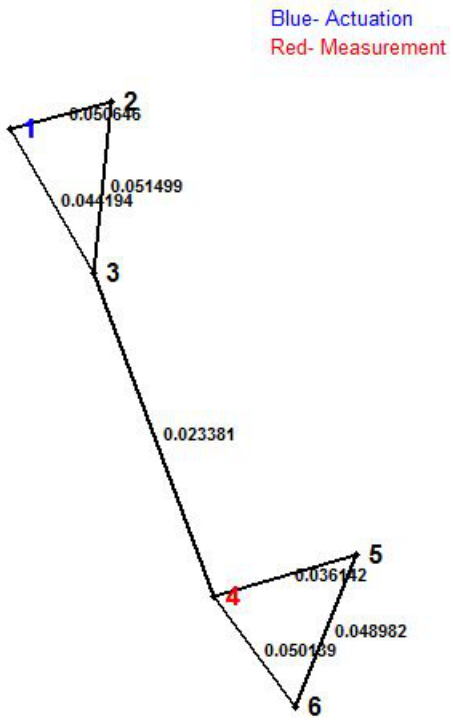

Figure 4.1. Example of input network structure.

sensor locations can significantly enhance estimation performance [10, 31]. Our philosophy is that some basic understanding of network structure can lead to a structure-exploiting design of effective sensor locations.

In this tool, the best observation location is displayed for a fixed network structure and actuation location. The best observation location is achieved by looping through all possible locations and comparing the CRBs of the Fiedler eigenvalue associated with each location.

4.3.1. Algorithm to Find the Effective Measurement Location

As I relate the structure of the network to the discrete SISO system dynamics, The first step is to get the dynamics from the input structure and relate to the equation (7).

%%% To get matrix A %%%

Get the distance between all the n vertices from the input graph

Find the edge weights by the inverting the above distances

Find the Laplacian Matrix L whose elements are the edge weights

IF the elements of Laplacian matrix are > 1 or <0

Scale the edge weights to be in the range (0,1)

ENDIF

Negate all the elements of L

Find the State matrix A using $A = I - L$

B and C represents the actuation and measurement locations.

Get the n*1 matrix B and put '1' in the matrix at the actuation location . Rest of the elements are zero

Get the 1*n matrix C and put '1' in the matrix at measurement location. Rest of the elements are zero

$D = 0$

%%% To get the Cramer - Rao bounds %%%

Get the transfer function from the state-space form

Find the Poles and Residues from transfer function

The Fisher matrix is obtained using equation(13)

Find CRB by inverting the Fisher matrix (equation(18))

%%% To get the Efficient Cramer - Rao bounds %%%

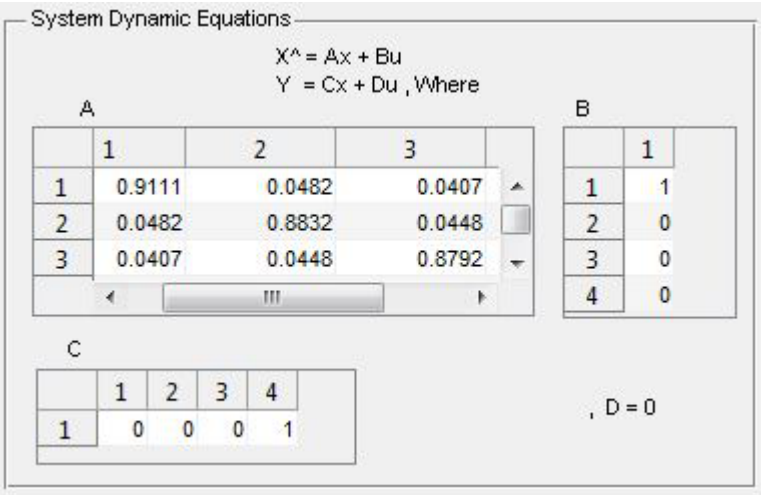

Figure 4.2. System dynamics.

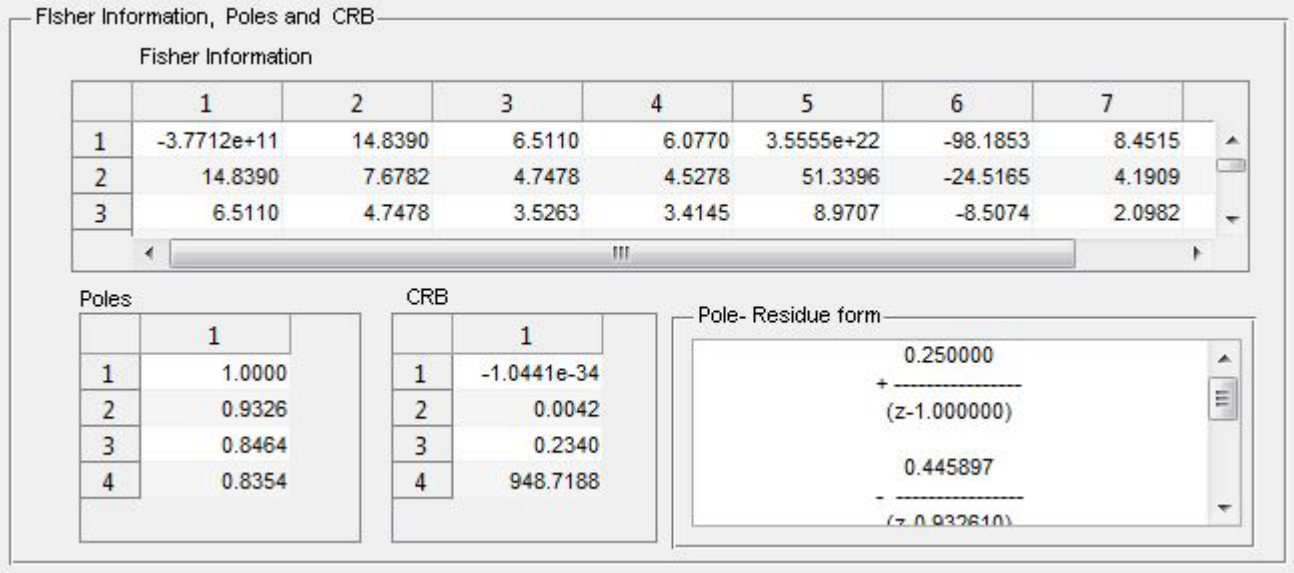

Figure 4.3. Pole-residue, Fisher matrix and CRB.

FOR each node in the network

Find the CRB

Get the value of the bound at Fiedler eigenvalue

END FOR

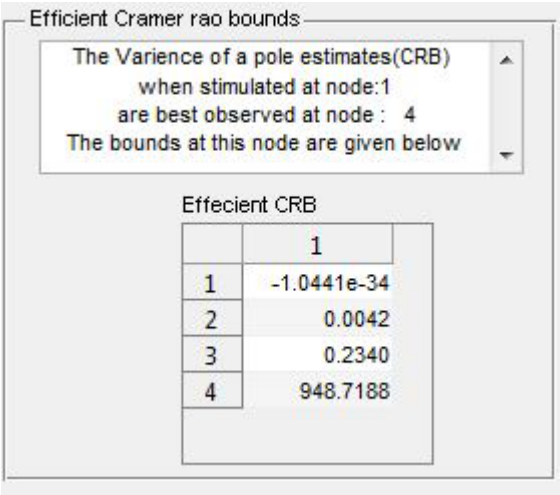

Figure 4.4. Efficient CRB

Compare all the values of the bound at Fiedler eigenvalue for smallest. Get measurement location at which the bound is small

Get the location and CRB at that measurement location

CHAPTER 5

SOME GRAPHICAL RESULTS ON ESTIMATION PERFORMANCE FOR SUBSETS OF NETWORK STRUCTURES

The tool allows us to investigate network structure's impact on the performance of pole estimation. The graphical representation of network structure and the modification features permitted on the graph facilitate the discovery of theoretical results. In this chapter, I present some preliminary graphical results on estimation performance and their proofs.

The first theorem suggests that exchanging actuation and measurement locations does not change the estimation performance of any pole.

Theorem 5.1. *Consider the estimation of an eigenvalue λ_r from a network with the dynamics described in Equation (7). If the network is actuated at node i and measured at node j, the CRB for λ_r does not change if the actuation and measurement locations are switched.*

Proof. According to Theorem 6 in [27], the CRB for λ_r scales inversely with $(v_{ri}v_{rj})^2$, where v_{ri} is the ith entry in the eigenvector associated with the eigenvalue λ_r. After the actuation and measurement locations are switched, let us denote the new eigenvector entries associated with actuation location j and measurement location i as v'_{rj} and v'_{ri} respectively. The CRB now scales inversely with $(v'_{rj}v'_{ri})^2$. As exchanging actuation and measurement locations does not change the right eigenvector of a system, I have $v_r = v'_r$. As such, the CRB in the new setting now scales inversely with $(v'_{rj}v'_{ri})^2 = (v_{rj}v_{ri})^2 = (v_{ri}v_{rj})^2$. The proof is complete. □ □

In Figure 5.1, the actuation is at node 3 and the measurement is at node 1. In Figure 5.2, the actuation is at node 3 and the measurement is at node 3. By comparing the outputs of two figures, I can observe that CRB's are the same even if the actuation and measurement

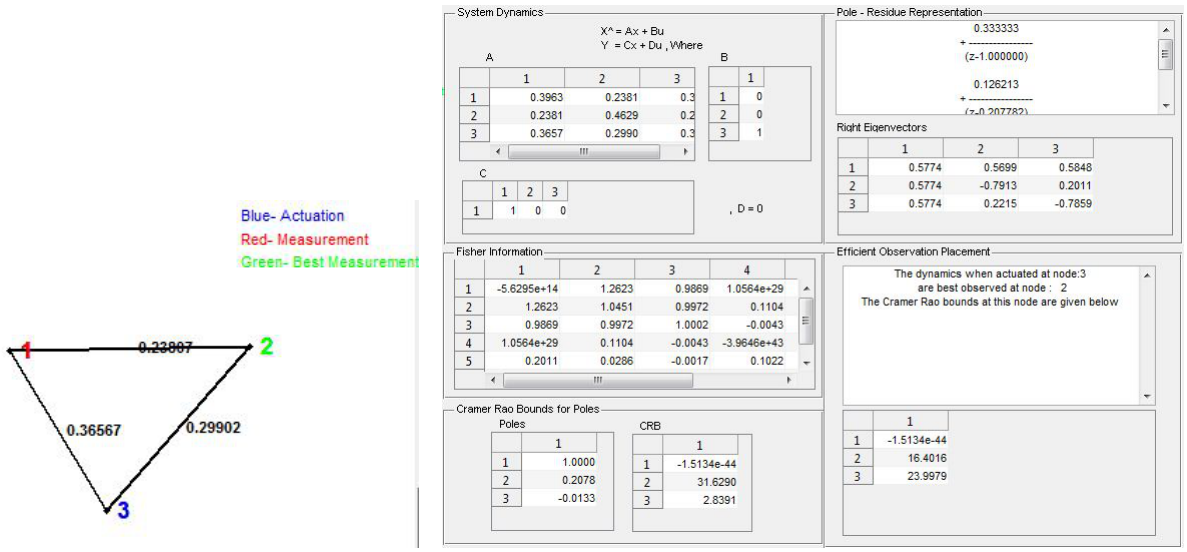

Figure 5.1. Triangle structure: actuated at node 3 and measured at node 1 (Left: Input, Right: Output).

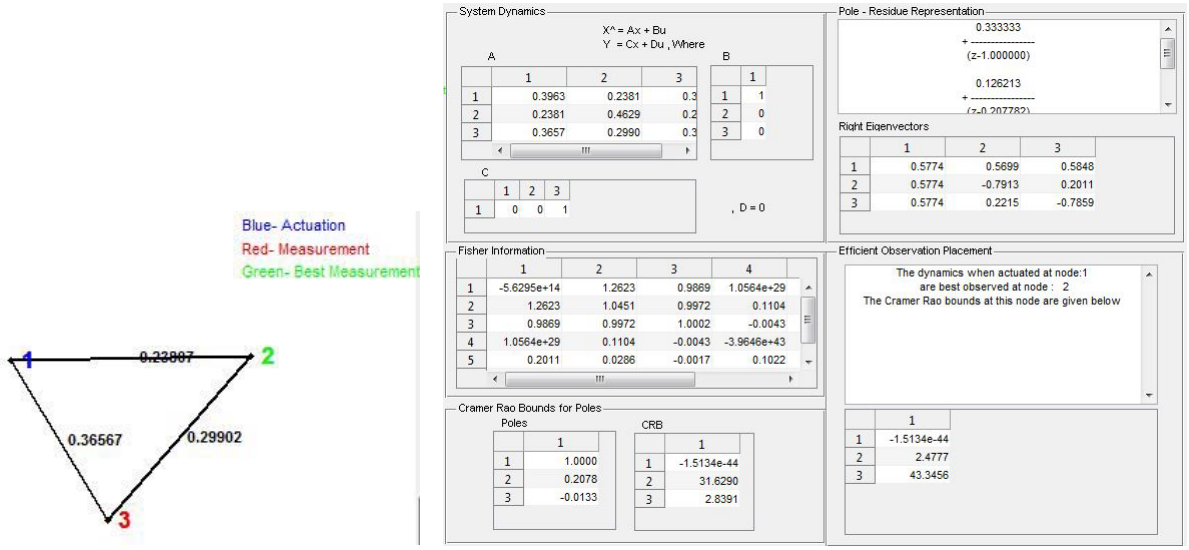

Figure 5.2. Triangle structure: actuated at node 1 and measured at node 3 (Left: Input, Right: Output).

locations are exchanged. The second theorem extends the result in [27] and informs the selection of observation locations in a tree graph when the network is actuated at a leaf of the tree.

Theorem 5.2. *Let us consider estimating the Fiedler eigenvalue of a network with a tree structure. If the network is actuated at a leaf, the minimum CRB is achieved when the measurement location is also at the leaves of the tree. Moreover, the worst observation location is among the nodes adjacent to the algebraic center of the tree.*

Proof. The estimation performance of the Fiedler eigenvalue scales inversely with $(v_{1i}v_{1j})^2$, where i is the actuation location, j is the measurement location, and v_1 is the eigenvector associated with the Fiedler eigenvalue, commonly called the Fiedler eigenvector. Moreover, according to [8, 7], entries in the Fiedler eigenvector increase or decrease monotonically from the algebraic center of the tree. As such, the maximum values of eigenvector entries are achieved at the leaves and the minimum are among the nodes adjacent to the algebraic center of the tree. Combining the above two results leads to the conclusion. □ □

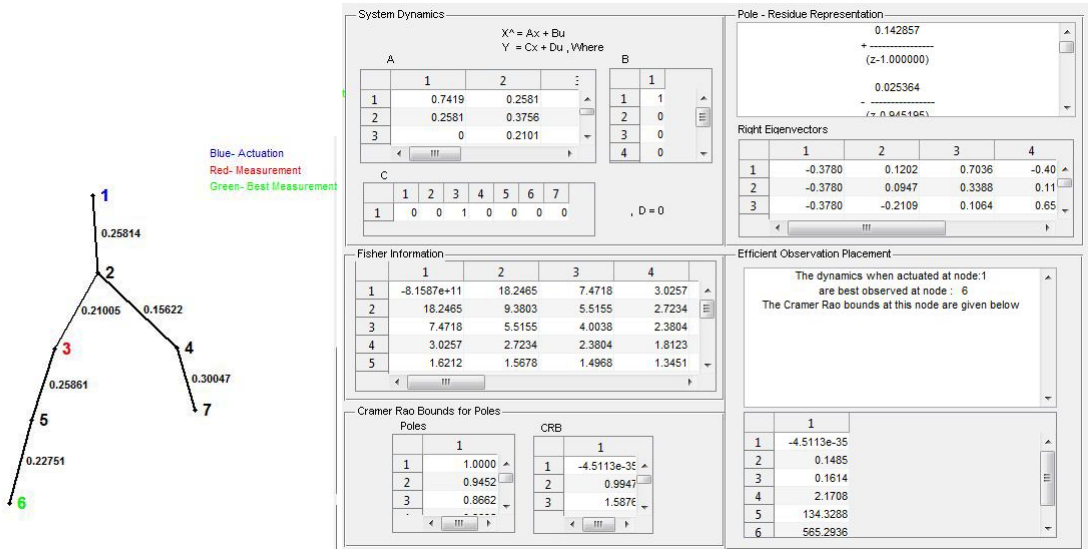

Figure 5.3. Example of input tree structure (Left: Input, Right: Output).

As seen in Figure 5.3, the actuation is placed at the leaf node 1 and measurement is at node 3. I can observe that the effective CRB(minimum CRB) is at the leaf which is at node 6. Similar characteristic holds for line structure i.e. in the line structure, the effective

measurement location is at the end of the line, farthest from the actuation location. The example of line structure is shown in Figure 5.4.

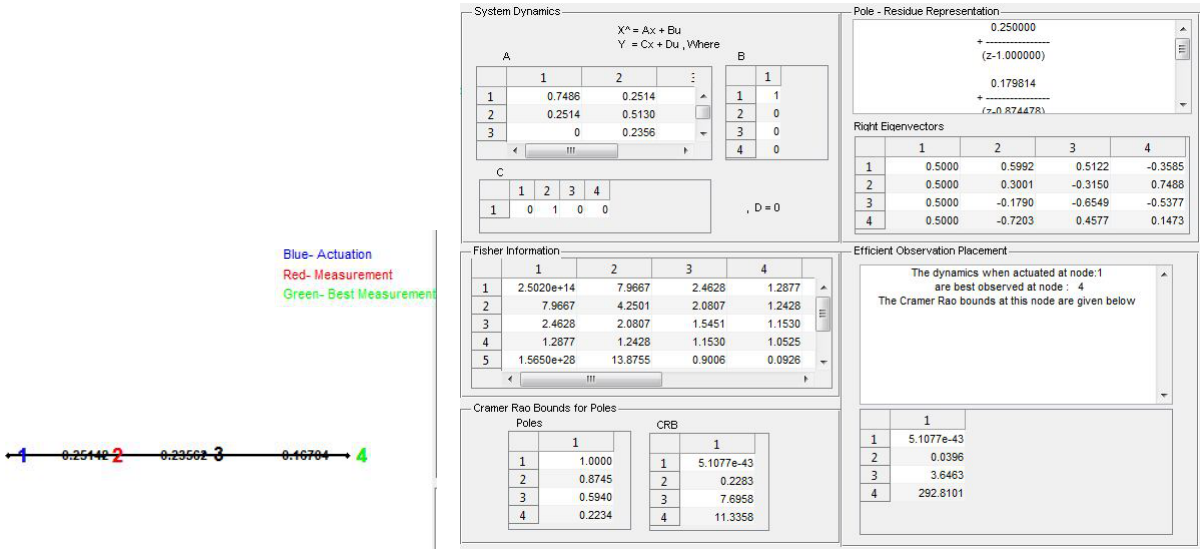

Figure 5.4. Example of line structure.

The next theorem suggests the node in a tree graph where the estimation of Fiedler eigenvalue is impossible.

Theorem 5.3. *Consider the estimation of the Fiedler eigenvalue of a network with a tree structure. If the algebraic center is a node in the tree, inference is not possible if this particular node is either the actuation or measurement locations.*

Proof. If the algebraic center is a node (denoted as k) in a tree, the entry v_{1k} in the Fielder eigenvector is 0 according to Theorem 4.3 in [7]. As the estimation performance of the Fiedler eigenvalue scales inversely with $(v_{1i}v_{1k})^2$ if the network is actuated at node i, or $(v_{1k}v_{1j})^2$ if the network is measured at node j, I obtain that the CRB is ∞ in each case, indicating that estimating the Fiedler eigenvalue is impossible. □ □

The above theorem naturally leads to the conclusion that in any symmetric tree graph,

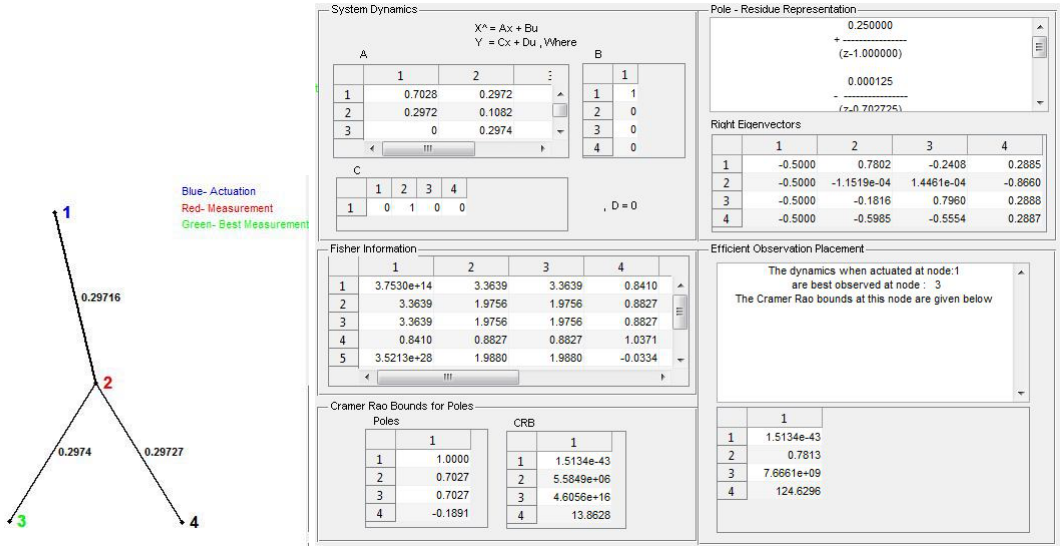

Figure 5.5. Example of symmetric tree graph.

the inference of the Fiedler eigenvalue is impossible if the network is actuated or measured at the center of the symmetry. Examples of symmetric trees include star graphs with all edge weights the same. As shown in Figure 5.5, the edge weights are nearly equal.The CRB at Fielder eigenvalue is very large or close to infinity. Therefore, inference on estimation performance is not possible.

The next theorem is concerned with the trend of CRB when the edge weight is changed in a simple graph with only two nodes.

Theorem 5.4. *Consider a graph with two nodes. Increasing the edge weight between the two nodes from* 0 *to* ∞ *causes the CRB to increase monotonically and then decrease monotonically. The largest CRB occurs when the edge weight is* 0.5.

Proof. In the two-node case, as suggested by Equation (17), the CRB scales inversely with $\frac{(1+p_1^2)}{(1-p_1^2)^3}$ and A_1, where $A_1 = (v_{11}v_{12})^2$. Because the first norm of the Fielder eigenvector equals 0, I have $v_{11} = -v_{12}$. Moreover, as $v_{11}^2 + v_{12}^2 = 1$, I can easily obtain that $v_{11} = v_{12} = \sqrt{0.5}$, independent of the length between the two nodes. Because A_1 is a constant, CRB is only

affected by the value of the pole p_1. As the length between the two nodes increases from 0, the Fielder eigenvalue p_1 increases from -1 to 1, and as such causing $\frac{(1+p_1^2)}{(1-p_1^2)^3}$ to decrease first and then increase, with the minimum occurring while $p_1 = 0$. This occurs when the edge weight between the two nodes is 0.5. □ □

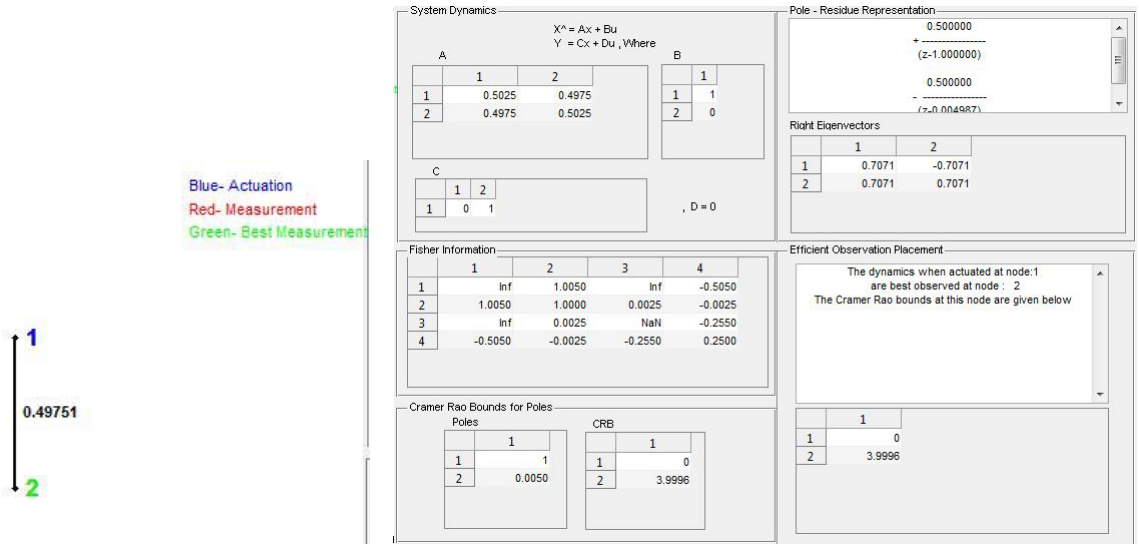

Figure 5.6. Two node graph structure.

Figure 5.6 shows the input(left) and output(right) of a two node structure when the CRB is maximum. The Figure 5.7 shows a graph representing that as the edge weights are increasing, the CRB increases first and then decreases. The maximum CRB is at a edge weight close to 0.5. (The graph is plotted from the values taken from the tool by increasing edge weights for a structure shown in Figure 5.6).

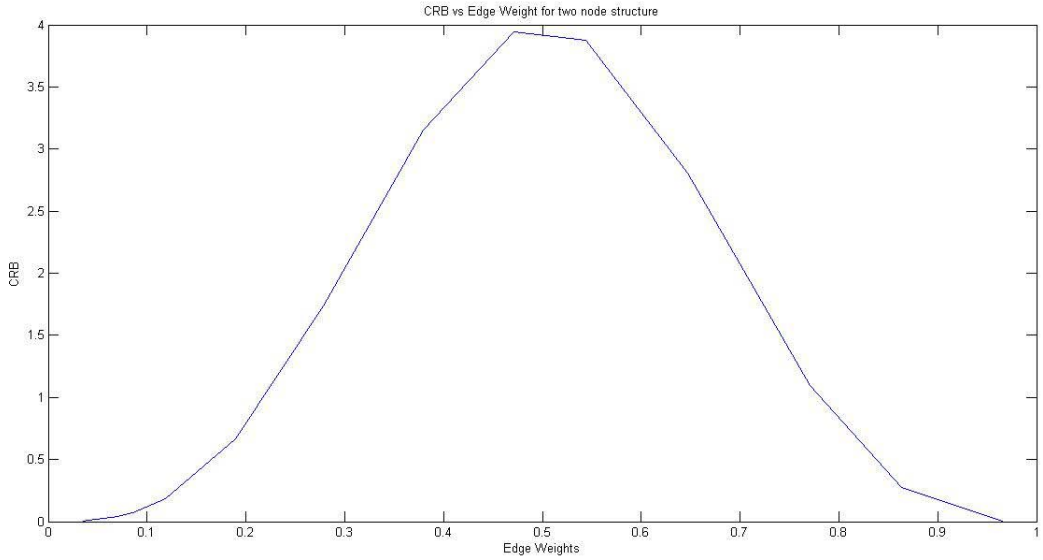

Figure 5.7. CRB vs Edge Weights for a two node graph structure.

CHAPTER 6

CONCLUDING REMARK AND FUTURE WORK

In this thesis, I introduce an interactive tool to investigate the performance of network dynamics from noisy data. The contributions of the tool is that 1) it facilitates the theoretical analysis of network structure's impact on network estimation performance, and 2) it suggests the best measurement location to maximize network estimation performance, and thus provides guidance for the design of data collection experiments. In future, one can utilize the tool to develop further analytical results for non-tree structures. Moreover, I will extend the network structure from single layer to hierarchical structures, and consider network dynamics defined upon Laplacian and adjacency matrices.

APPENDIX A

MATLAB IMPLEMANTATION OF THE TOOL USING GUIDE-GUI MAIN FUNCTION

```
%%%% The main function for GUI %%%%
%%%% Developer: Veenadhar Katragadda %%%%
function varargout = crb(varargin)
% CRB MATLAB code for crb.fig
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @crb_OpeningFcn, ...
'gui_OutputFcn',  @crb_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before crb is made visible.
function crb_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
24 %%% As a opening function we declare array variables here
25 xy=[]; %% XY is the n*2 matrix which stores the location(X and
       Y points) of n vertices.
26 A=[]; %% A is the n*n matrix which has the edge definitions (
      undirected or directed) between vertices
27 %%% rest of the variables used for intermediate calcuations
28 eid=[];
29 id1=[];
30 xytemp=[];
31 Atemp=[];
32 enchk=[];
33 enchk=0;
34 prev_scale = 1;
35 %%% Updates the handles structure so that these variables can
      be
36 %%% used in different fucntions.
37 handles.xy = xy;
38 handles.A=A;
39 handles.eid=eid;
40 handles.id1=id1;
41 handles.xytemp = xytemp;
42 handles.Atemp = Atemp;
43 handles.enchk=enchk;
```

```
44 handles.prev_scale=prev_scale;
45 handles.fir_cnt = [];
46 guidata(hObject, handles);
47 set(handles.enter,'enable','off'); %% set visibility of enter
     button to off
48 set(handles.undo,'enable','off'); %% set visibility of undo
     button to off
49 set(handles.uok,'enable','off');
50 set(handles.check,'enable','off');
51 set(handles.scale,'enable','inactive');
52 set(handles.check2,'enable','off');
53 % Choose default command line output for crb
54 handles.output = hObject;
55 % Update handles structure
56 guidata(hObject, handles);
57 % --- Outputs from this function are returned to the command
     line.
58 function varargout = crb_OutputFcn(hObject, eventdata, handles
     )
59 varargout{1} = handles.output;
60 % --- Executes on button press in enter.
61 function enter_Callback(hObject, eventdata, handles,varargin)
62 handles.enchk=1;
```

```
63 guidata(hObject, handles);
64 c2 = get(handles.check2,'value');
65 set(handles.draw,'value',1); %% set the draw pop-up menu to
     draw mode.
66 set(handles.vertex,'Value',1); %% set the vertex pop-up menu
     to vertex mode.
67 set(handles.check2,'enable','on');
68 if c2 == 0
69 set(handles.uok,'enable','off');
70 set(handles.check,'enable','on');
71 set(handles.scale,'enable','inactive');
72 elseif c2 == 1
73 set(handles.uok,'enable','off');
74 set(handles.check,'enable','off');
75 set(handles.scale,'enable','inactive');
76 end
77 %%% get the input data from the user from "input data" section
     .
78 in_act = str2double(get(handles.actuation,'String'));
79 in_meas= str2double(get(handles.measurement,'String'));
80 in_noise= str2double(get(handles.noise,'String'));
81 [m,o]= size(handles.xy);
82 if in_act > m %if the input is higher than the number of nodes
```

```
83 %this is the first line of the msgbox
84 msgboxText{1} =  'You have entered a value that is higher than
       the number of nodes at actuation location input';
85 %this command creates the actual message box
86 msgbox(msgboxText,'Input number incorrect', 'error');
87 elseif isempty(in_act) %if the input is not a number
88 %this is the first line of the msgbox
89 msgboxText{1} =  'You have tried to input something that is
     NOT a number at actuation location input.';
90 %this command creates the actual message box
91 msgbox(msgboxText,'Input not a number', 'error');
92 elseif isnan(in_act) %if the input is not a number
93 %this is the first line of the msgbox
94 msgboxText{1} =  'Nothing is entered at actuation location
     input.';
95 %this command creates the actual message box
96 msgbox(msgboxText,'Input empty', 'error');
97 elseif in_meas > m %if the input is higher than the number of
     nodes
98 %this is the first line of the msgbox
99 msgboxText{1} =  'You have entered a value that is higher than
       the number of nodes at measurement location input';
100 %this command creates the actual message box
```

```
101 msgbox(msgboxText,'Input number incorrect', 'error');
102 elseif isempty(in_meas) %if the input is not a number
103 %this is the first line of the msgbox
104 msgboxText{1} =  'You have tried to input something that is
      NOT a number at measurement location input.';
105 %this command creates the actual message box
106 msgbox(msgboxText,'Input not a number', 'error');
107 elseif isnan(in_meas) %if the input is not a number
108 %this is the first line of the msgbox
109 msgboxText{1} =  'Nothing is entered at measurement location
      input input.';
110 %this command creates the actual message box
111 msgbox(msgboxText,'Input empty', 'error');
112 elseif isempty(in_noise) %if the input is not a number
113 %this is the first line of the msgbox
114 msgboxText{1} =  'You have tried to input something that is
      NOT a number at Noise varience input.';
115 %this command creates the actual message box
116 msgbox(msgboxText,'Input not a number', 'error');
117 elseif isnan(in_noise) %if the input is not a number
118 %this is the first line of the msgbox
119 msgboxText{1} =  'Nothing is entered at Noise input.';
120 %this command creates the actual message box
```

```
121 msgbox(msgboxText,'Input empty', 'error');
122 else
123 %%% Call the matrix function to calculate several outputs for
      the
124 %%% current input.
125 sc=1;
126 en=0;
127 if c2 == 1
128 fixval = handles.fix1;
129 fix =1;
130 else
131 fix = 0;
132 fixval = 1;
133 end
134 [mat2,p,r,F1,CRLBmat2,A1,B,C,vstr,xymid,ns,CRLBest,posi,kk,D,
      errchk] = matrix(handles.xy,handles.A,in_act,in_meas,
      in_noise,sc,en,fix,fixval,handles.prev_scale);
135 handles.prev_scale = kk;
136 guidata(hObject,handles);
137 if errchk == 1
138 errordlg('The scaling factor cannot be possible','error');
139 else
140 [m3,o3]=size(handles.A);
```

```
count=0;
cla;
hold on
%%% Display the edge weights on respective edges.
for e=1:m3
for f=e+1:m3
if handles.A(e,f) == 1
count = count + 1;
text(xymid(count,1),xymid(count,2),['  ' num2str(A1(e,f))],'
    Color','k','FontSize',8,'FontWeight','b');
else
count = count + 1;
end
end
end
%%% Display and plot all outputs on GUI.
if ~isempty(handles.xy)
plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
[ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy);
n2 = size(handles.A,1);
plot(handles.axes,ux,uy,'k-','LineWidth',2)
plot(handles.axes,dxu,dyu,'r:')
plot(handles.axes,dxl,dyl,'b--')
```

```
163 % plot(handles.axes,ix,iy,'ko')
164 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
165 for k = 1:n2
166 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
167 end
168 end
169 text(handles.xy(in_act,1),handles.xy(in_act,2),['  ' num2str(
      in_act)],'Color','b','FontSize',12,'FontWeight','b')
170 text(handles.xy(in_meas,1),handles.xy(in_meas,2),['  ' num2str
      (in_meas)],'Color','r','FontSize',12,'FontWeight','b')
171 text(handles.xy(posi,1),handles.xy(posi,2),['  ' num2str(posi)
      ],'Color','g','FontSize',12,'FontWeight','b')
172 text(7,9.15,'Blue- Actuation','Color','b','FontSize',8)
173 text(7,8.83,'Red- Measurement','Color','r','FontSize',8)
174 text(7,8.51,'Green- Best Measurement','Color','g','FontSize'
      ,8)
175 if length(dxl)>=2
176 if  ~isnan(dxl(1,1)+dxl(2,1))
177 text(0.52,9.47,'Directed edges:','FontSize',8)
178 text(0.52,9.15,'-- Blue large to small node','Color','b','
      FontSize',8)
```

```
179 text(0.52,8.83,'-- Red small to large node','Color','r','
      FontSize',8)
180 else
181 end
182 else
183 end
184 if length(dxu)>=2
185 if ~isnan(dxu(1,1)+dxu(2,1))
186 text(0.52,9.47,'Directed edges:','FontSize',8)
187 text(0.52,9.15,'-- Blue large to small node','Color','b','
      FontSize',8)
188 text(0.52,8.83,'-- Red small to large node','Color','r','
      FontSize',8)
189 else
190 end
191 else
192 end
193 set(handles.scale,'Units','normalized', 'String', kk);
194 set(handles.rev,'Units','normalized', 'Data', D);
195 set(handles.poles,'Units','normalized', 'Data', p);
196 set(handles.fisher,'Units','normalized', 'Data', F1);
197 set(handles.crb,'Units','normalized', 'Data', CRLBmat2);
198 set(handles.a, 'Units','normalized','Data', A1);
```

```
199 set(handles.b, 'Units','normalized', 'Data', B);
200 set(handles.b,'ColumnWidth',{25});
201 set(handles.c,'Units','normalized', 'Data', C);
202 set(handles.c,'ColumnWidth',{25});
203 set(handles.note,'String',ns);
204 set(handles.crbb,'Units','normalized', 'Data', CRLBest);
205 set(handles.pr,'String',vstr);
206 end
207 checkk = get(handles.check,'value');
208 if checkk == 1
209 set(handles.uok,'enable','on');
210 set(handles.scale,'enable','on');
211 else
212 set(handles.uok,'enable','off');
213 set(handles.scale,'enable','inactive');
214 end
215 end
216 %--- Executes on selection change in save.
217 function save_Callback(hObject, eventdata, handles)
218 switch get(handles.save,'Value') %%% Switch for different
      cases.
219 case 1  %% To plot a new network
220 if ~isempty(handles.xy)
```

```
221 set(handles.vertex,'Value',1); %% Set the vertex pop up menu
      to vertex mode.
222 set(handles.draw,'Value',1); %% Set the draw popup menu to
      draw mode.
223 else
224 %%% Make the variables empty if they are not.
225 xy=[];
226 A=[];
227 eid=[];
228 id1=[];
229 xytemp=[];
230 Atemp=[];
231 enchk=[];
232 enchk=0;
233 handles.xy = xy;
234 handles.A=A;
235 handles.eid=eid;
236 handles.id1=id1;
237 handles.xytemp = xytemp;
238 handles.Atemp = Atemp;
239 handles.enchk=enchk;
240 guidata(hObject, handles);
241 end
```

```
242 case 2 %% Open saved netwrok.
243 if ~isempty(handles.xy)
244 %%% Create a dialogue box.
245 ans = questdlg('Are you sure you want to open a saved graph?
      This operation cannot be undone.', ...
246 'Open a Graph', ...
247 'Yes','No','No');
248 switch ans
249 case 'Yes' %% Open a user interface box to select to open the
      saved network.
250 A=[];
251 xy=[];
252 uiopen('*.mat'); %% open a user interface box.
253 handles.A = A;
254 handles.xy = xy;
255 guidata(hObject, handles);
256 if ~isempty(handles.xy)
257 hold on
258 cla; %% clear the axes
259 %%% Plot the saved network on the axes of GUI.
260 if ~isempty(handles.xy)
261 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
262 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy);
```

```
263 n2 = size(handles.A,1);
264 plot(handles.axes,ux,uy,'k-','LineWidth',2)
265 plot(handles.axes,dxu,dyu,'r:')
266 plot(handles.axes,dxl,dyl,'b--')
267 % plot(handles.axes,ix,iy,'ko')
268 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
269 for k = 1:n2
270 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
271 end
272 end
273 %%% Set motion function to off and make it on to avoid the
      delay of
274 %%% the function
275 set(gcf,'WIndowButtonMotionFcn','');
276 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
277 set(handles.enter,'enable','on'); %% Set visibility of enter
      button to on
278 set(handles.undo,'enable','off'); %% Set visibility of undo
      button to off
279 set(handles.save,'value',1); %% Set the save pop menu to "plot
       a network" option.
```

```
280 set(handles.new,'enable','on');
281 else
282 set(handles.enter,'enable','off'); %% Set visibility of enter
      button to off
283 set(handles.undo,'enable','off'); %% Set visibility of undo
      button to off
284 set(handles.save,'value',1); %% Set the save pop menu to "plot
       a network" option.
285 end
286 otherwise
287 set(handles.save,'value',1); %% Set the save pop menu to "plot
       a network" option.
288 end
289 else
290 A=[];
291 xy=[];
292 uiopen('*.mat');
293 handles.A = A;
294 handles.xy = xy;
295 guidata(hObject, handles);
296 if ~isempty(handles.xy)
297 hold on
298 cla;
```

```
299 if ~isempty(handles.xy)
300 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
301 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy);
302 n2 = size(handles.A,1);
303 plot(handles.axes,ux,uy,'k-','LineWidth',2)
304 plot(handles.axes,dxu,dyu,'r:')
305 plot(handles.axes,dxl,dyl,'b--')
306 % plot(handles.axes,ix,iy,'ko')
307 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
308 for k = 1:n2
309 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
310 end
311 end
312 set(gcf,'WIndowButtonMotionFcn','');
313 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
314 set(handles.enter,'enable','on');
315 set(handles.undo,'enable','on');
316 set(handles.save,'value',1);
317 set(handles.new,'enable','on');
318 else
319 set(handles.enter,'enable','off');
```

```
320 set(handles.undo,'enable','off');
321 set(handles.save,'value',1);
322 end
323 end
324 case 3
325 if isempty(handles.xy)
326 msgbox('There is nothing to save','Info','error');
327 set(handles.save,'value',1);
328 set(handles.draw,'value',1);
329 set(handles.undirected,'value',1);
330 else
331 A = handles.A;
332 xy = handles.xy;
333 uisave({'A','xy'});
334 msgbox('Saved','Info');
335 set(handles.save,'value',1);
336 set(handles.draw,'value',1);
337 set(handles.undirected,'value',1);
338 end
339 otherwise
340 end
341 % --- Executes on selection change in vertex.
342 function vertex_Callback(hObject, eventdata, handles)
```

```
343 v = get(handles.draw,'value');
344 v1 = get(handles.vertex,'value');
345 if v==3
346 set(handles.vertex,'value',1);
347 msgbox('Cannot use edge mode','Info','error');
348 end
349 if v1==2
350 if isempty(handles.xy)
351 msgbox('Cannot draw/delete edges without Vertices/Nodes','
      Error in GUI','error');
352 set(handles.vertex,'value',1);
353 end
354 else
355 end
356 % --- Executes on selection change in draw.
357 function draw_Callback(hObject, eventdata, handles)
358 v = get(handles.draw,'value');
359 v1 = get(handles.vertex,'value');
360 if v==2
361 if isempty(handles.xy)
362 msgbox('Cannot delete Vertices/Edges without Nodes','Error in
      GUI','error');
363 set(handles.draw,'value',1);
```

```
364 end
365 else
366 end
367 if v==3
368 if isempty(handles.xy)
369 set(handles.draw,'value',1);
370 msgbox('Cannot drag without Vertices/Nodes','Error in GUI','
      error');
371 elseif ~isempty(handles.xy)
372 set(handles.vertex,'Value',1);
373 set(gcf,'WIndowButtonMotionFcn','');
374 set(gcf,'WindowButtonUpFcn',{@figure1_WindowButtonUpFcn,
      handles});
375 set(handles.enter,'enable','off');
376 else
377 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
378 set(handles.enter,'enable','on');
379 end
380 end
381 ch1=all(all(handles.A==0));
382 if ch1==1
383 set(handles.undo,'enable','off');
```

```
384 end
385 function figure1_WindowButtonMotionFcn(hObject, eventdata,
      handles)
386 clc;
387 axx=axis;
388 pttt=getPoint;
389 if (pttt(1,1)<axx(1,1))
390 set(handles.xytext,'String','');
391 elseif(pttt(1,1)>axx(1,2))
392 set(handles.xytext,'String','');
393 elseif(pttt(1,2)<axx(1,3))
394 set(handles.xytext,'String','');
395 elseif(pttt(1,2)>axx(1,4))
396 set(handles.xytext,'String','');
397 else
398 v3 = get(handles.draw,'value');
399 v4 = get(handles.vertex,'value');
400 v5 = get(handles.undirected,'value');
401 pt = getPoint;
402 if v3==1 && v4==1
403 set(handles.xytext,'String',[' x = ' num2str(pt) ' = y ']);
404 elseif ~isempty(handles.xy)
405 id = getNearestVertex(pt,handles.xy);
```

```
406 set(handles.xytext,'String',[' id = ' num2str(id)]);
407 else
408 set(handles.xytext,'String','');
409 end
410 if v3 == 3
411 if isempty(handles.xy)
412 set(handles.draw,'value',1);
413 else
414 ch= all(all(handles.A==0));
415 ax=axis;
416 pt1 = getPoint;
417 handles.xy(handles.id1,:) = pt1;
418 guidata(hObject, handles);
419 n1 = size(handles.A,1);
420 if ch==1
421 set(handles.undo,'enable','off');
422 cla;
423 hold on
424 if ~isempty(handles.xy)
425 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
426 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
427 n2 = size(handles.A,1);
428 plot(handles.axes,ux,uy,'k-','LineWidth',2)
```

```
429 plot(handles.axes,dxu,dyu,'r:')
430 plot(handles.axes,dxl,dyl,'b--')
431 % plot(handles.axes,ix,iy,'ko')
432 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
433 for k = 1:n2
434 text(handles.xy(k,1),handles.xy(k,2),[' ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
435 end
436 end
437 elseif ch==0
438 cla;
439 hold on
440 if ~isempty(handles.xy)
441 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
442 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
443 n2 = size(handles.A,1);
444 plot(handles.axes,ux,uy,'k-','LineWidth',2)
445 plot(handles.axes,dxu,dyu,'r:')
446 plot(handles.axes,dxl,dyl,'b--')
447 % plot(handles.axes,ix,iy,'ko')
448 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
449 for k = 1:n2
```

```
450 text(handles.xy(k,1),handles.xy(k,2),['    ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
451 end
452 in_act = str2double(get(handles.actuation,'String'));
453 in_meas= str2double(get(handles.measurement,'String'));
454 in_noise= str2double(get(handles.noise,'String'));
455 [m,o]= size(handles.xy);
456 if in_act > m %if the input is higher than the number of nodes
457 %this is the first line of the msgbox
458 msgboxText{1} =  'You have entered a value that is higher than
       the number of nodes at actuation location input';
459 %this command creates the actual message box
460 msgbox(msgboxText,'Input number incorrect', 'error');
461 elseif isempty(in_act) %if the input is not a number
462 %this is the first line of the msgbox
463 msgboxText{1} =  'You have tried to input something that is
      NOT a number at Noise varience input at actuation location
      input.';
464 %this command creates the actual message box
465 msgbox(msgboxText,'Input not a number', 'error');
466 elseif isnan(in_act) %if the input is not a number
467 %this is the first line of the msgbox
```

```
468 msgboxText{1} =  'Nothing is entered at actuation location
      input.';
469 %this command creates the actual message box
470 msgbox(msgboxText,'Input empty', 'error');
471 elseif in_meas > m %if the input is higher than the number of
      nodes
472 %this is the first line of the msgbox
473 msgboxText{1} =  'You have entered a value that is higher than
       the number of nodes at measurement location input';
474 %this command creates the actual message box
475 msgbox(msgboxText,'Input number incorrect', 'error');
476 elseif isempty(in_meas) %if the input is not a number
477 %this is the first line of the msgbox
478 msgboxText{1} =  'You have tried to input something that is
      NOT a number at Noise varience input at measurement
      location input.';
479 %this command creates the actual message box
480 msgbox(msgboxText,'Input not a number', 'error');
481 elseif isnan(in_meas) %if the input is not a number
482 %this is the first line of the msgbox
483 msgboxText{1} =  'Nothing is entered at measurement location
      input input.';
484 %this command creates the actual message box
```

```
485 msgbox(msgboxText,'Input empty', 'error');
486 elseif isempty(in_noise) %if the input is not a number
487 %this is the first line of the msgbox
488 msgboxText{1} =  'You have tried to input something that is
      NOT a number at Noise varience input.';
489 %this command creates the actual message box
490 msgbox(msgboxText,'Input not a number', 'error');
491 elseif isnan(in_noise) %if the input is not a number
492 %this is the first line of the msgbox
493 msgboxText{1} =  'Nothing is entered at Noise input.';
494 %this command creates the actual message box
495 msgbox(msgboxText,'Input empty', 'error');
496 else
497 sc=1;
498 en=0;
499 c2 = get(handles.check2,'value');
500 if c2 == 1
501 fixval = handles.fix1;
502 fix =1;
503 else
504 fix = 0;
505 fixval = 1;
506 end
```

```
507 [mat2,p,r,F1,CRLBmat2,A1,B,C,vstr,xymid,ns,CRLBest,posi,kk,D,
      errchk] = matrix(handles.xy,handles.A,in_act,in_meas,
      in_noise,sc,en,fix,fixval,handles.prev_scale);
508 if handles.prev_scale ~= kk
509 set(gcf,'WIndowButtonMotionFcn','');
510 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
511 end
512 handles.prev_scale = kk;
513 guidata(hObject, handles);
514 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
515 if errchk == 1
516 set(gcf,'WIndowButtonMotionFcn','');
517 errordlg('Further dragging cannot be possible beacause of
      scaling constraint','error');
518 else
519 [m3,o3]=size(handles.A);
520 count=0;
521 for e=1:m3
522 for f=e+1:m3
523 if handles.A(e,f) == 1
524 count = count + 1;
```

```
525 text(xymid(count,1),xymid(count,2),['   ' num2str(A1(e,f))],'
      Color','k','FontSize',8,'FontWeight','b');
526 else
527 count = count + 1;
528 end
529 end
530 end
531 hold on
532 text(handles.xy(in_act,1),handles.xy(in_act,2),['   ' num2str(
      in_act)],'Color','b','FontSize',12,'FontWeight','b')
533 text(handles.xy(in_meas,1),handles.xy(in_meas,2),['   ' num2str
      (in_meas)],'Color','r','FontSize',12,'FontWeight','b')
534 text(handles.xy(posi,1),handles.xy(posi,2),['   ' num2str(posi)
      ],'Color','g','FontSize',12,'FontWeight','b')
535 text(7,9.15,'Blue- Actuation','Color','b','FontSize',8)
536 text(7,8.83,'Red- Measurement','Color','r','FontSize',8)
537 text(7,8.51,'Green- Best Measurement','Color','g','FontSize'
      ,8)
538 if length(dxl)>=2
539 if  ~isnan(dxl(1,1)+dxl(2,1))
540 text(0.52,9.47,'Directed edges:','FontSize',8)
541 text(0.52,9.15,'-- Blue large to small node','Color','b','
      FontSize',8)
```

```
542 text(0.52,8.83,'-- Red small to large node','Color','r','
      FontSize',8)
543 else
544 end
545 else
546 end
547 if length(dxu)>=2
548 if ~isnan(dxu(1,1)+dxu(2,1))
549 text(0.52,9.47,'Directed edges:','FontSize',8)
550 text(0.52,9.15,'-- Blue large to small node','Color','b','
      FontSize',8)
551 text(0.52,8.83,'-- Red small to large node','Color','r','
      FontSize',8)
552 else
553 end
554 else
555 end
556 set(handles.scale,'Units','normalized', 'String', kk);
557 set(handles.rev,'Units','normalized', 'Data', D);
558 set(handles.poles,'Units','normalized', 'Data', p);
559 set(handles.fisher,'Units','normalized', 'Data', F1);
560 set(handles.crb,'Units','normalized', 'Data', CRLBmat2);
561 set(handles.a, 'Units','normalized','Data', A1);
```

```
562 set(handles.b, 'Units','normalized', 'Data', B);
563 set(handles.b,'ColumnWidth',{25});
564 set(handles.c,'Units','normalized', 'Data', C);
565 set(handles.c,'ColumnWidth',{25});
566 set(handles.note,'String',ns);
567 set(handles.crbb,'Units','normalized', 'Data', CRLBest);
568 set(handles.pr,'String',vstr);
569 end
570 end
571 end
572 end
573 end
574 end
575 end
576 clc
577 %--- Executes on mouse press over figure background, over a
      disabled or
578 %--- inactive control, or over an axes background.
579 function figure1_WindowButtonUpFcn(hObject, eventdata, handles
      )
580 axx=axis;
581 pttt=getPoint;
582 if (pttt(1,1)<axx(1,1))
```

```
583 elseif(pttt(1,1)>axx(1,2))
584 elseif(pttt(1,2)<axx(1,3))
585 elseif(pttt(1,2)>axx(1,4))
586 else
587 v6 = get(handles.draw,'value');
588 v7 = get(handles.vertex,'value');
589 v8 = get(handles.undirected,'value');
590 if v7==2 && ~isempty(handles.xy)
591 pt = getPoint;
592 ax = axis;
593 if pt(1)>=ax(1) && pt(1)<=ax(2) && pt(2)>=ax(3) && pt(2)<=ax
      (4)
594 handles.edid = getNearestVertex(pt,handles.xy);
595 guidata(hObject, handles);
596 if handles.eid==handles.edid
597 msgbox('the starting and ending nodes should be different to
      draw or delete the edge ','Error in draw/delete edge','
      error');
598 set(handles.undo,'enable','off');
599 end
600 if v6==1
601 handles.A(handles.eid,handles.edid) = 1;
602 guidata(hObject, handles);
```

```
603 elseif v6==2
604 handles.A(handles.eid,handles.edid) = 0;
605 guidata(hObject, handles);
606 else
607 end
608 if  v8==1
609 if handles.edid ~= handles.eid
610 handles.A(handles.edid,handles.eid) = handles.A(handles.eid,
      handles.edid);
611 guidata(hObject, handles);
612 else
613 handles.A(handles.edid,handles.eid) = 0;
614 guidata(hObject, handles);
615 end
616 elseif v8==2
617 if handles.edid == handles.eid
618 handles.A(handles.edid,handles.eid) = 0;
619 guidata(hObject, handles);
620 else
621 end
622 end
623 end
624 cla;
```

```
625 hold on
626 if ~isempty(handles.xy)
627 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
628 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
629 n2 = size(handles.A,1);
630 plot(handles.axes,ux,uy,'k-','LineWidth',2)
631 plot(handles.axes,dxu,dyu,'r:')
632 plot(handles.axes,dxl,dyl,'b--')
633 %% plot(handles.axes,ix,iy,'ko')
634 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
635 for k = 1:n2
636 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
637 end
638 end
639 set(gcf,'WIndowButtonMotionFcn','');
640 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
641 elseif v6==3
642 set(gcf,'WIndowButtonMotionFcn','');
643 end
644 end
645 %--- Executes on button press in undo.
```

```
646 function undo_Callback(hObject, eventdata, handles)
647 v9 = get(handles.draw,'value');
648 v10 = get(handles.vertex,'value');
649 v11 = get(handles.undirected,'value');
650 ax=axis;
651 handles.xy = handles.xytemp;
652 handles.A = handles.Atemp;
653 guidata(hObject, handles);
654 if v9==3
655 set(gcf,'WIndowButtonMotionFcn','');
656 hold on
657 cla;
658 if ~isempty(handles.xy)
659 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
660 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
661 n2 = size(handles.A,1);
662 plot(handles.axes,ux,uy,'k-','LineWidth',2)
663 plot(handles.axes,dxu,dyu,'r:')
664 plot(handles.axes,dxl,dyl,'b--')
665 %% plot(handles.axes,ix,iy,'ko')
666 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
667 for k = 1:n2
```

```
668 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
669 end
670 end
671 else
672 hold on
673 cla;
674 if ~isempty(handles.xy)
675 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
676 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
677 n2 = size(handles.A,1);
678 plot(handles.axes,ux,uy,'k-','LineWidth',2)
679 plot(handles.axes,dxu,dyu,'r:')
680 plot(handles.axes,dxl,dyl,'b--')
681 %% plot(handles.axes,ix,iy,'ko')
682 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
683 for k = 1:n2
684 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
685 end
686 end
687 set(gcf,'WIndowButtonMotionFcn','');
```

```
688 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
689 end
690 set(handles.undo,'enable','off');
691 if isempty(handles.xy)
692 set(handles.enter,'enable','off');
693 set(handles.new,'enable','off');
694 else
695 end
696 %--- Executes on mouse press over figure background, over a
      disabled or
697 %--- inactive control, or over an axes background.
698 function figure1_WindowButtonDownFcn(hObject, eventdata,
      handles)
699 axx=axis;
700 pttt=getPoint;
701 if (pttt(1,1)<axx(1,1))
702 elseif(pttt(1,1)>axx(1,2))
703 elseif(pttt(1,2)<axx(1,3))
704 elseif(pttt(1,2)>axx(1,4))
705 else
706 handles.xytemp = handles.xy;
707 handles.Atemp = handles.A;
```

```
708 guidata(hObject, handles);
709 set(handles.enter,'enable','on');
710 v = get(handles.draw,'value');
711 v1 = get(handles.vertex,'value');
712 v2 = get(handles.undirected,'value');
713 pt = getPoint;
714 ax = axis;
715 if pt(1)>=ax(1) && pt(1)<=ax(2) && pt(2)>=ax(3) && pt(2)<=ax
      (4)
716 if v1==2
717 if ~isempty(handles.xy)
718 handles.eid = getNearestVertex(pt,handles.xy);
719 guidata(hObject, handles);
720 end
721 elseif v==1
722 n = size(handles.xy,1);
723 handles.xy(n+1,:) = pt;
724 handles.A(n+1,n+1) = 0;
725 guidata(hObject, handles);
726 hold on
727 cla;
728 if ~isempty(handles.xy)
729 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
```

```
730 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
731 n2 = size(handles.A,1);
732 plot(handles.axes,ux,uy,'k-','LineWidth',2)
733 plot(handles.axes,dxu,dyu,'r:')
734 plot(handles.axes,dxl,dyl,'b--')
735 % plot(handles.axes,ix,iy,'ko')
736 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
737 for k = 1:n2
738 text(handles.xy(k,1),handles.xy(k,2),[' ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
739 end
740 end
741 set(gcf,'WIndowButtonMotionFcn','');
742 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
743 elseif v==3
744 if ~isempty(handles.xy)
745 handles.id1 = getNearestVertex(pt,handles.xy);
746 guidata(hObject, handles);
747 end
748 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
749 else
```

```
750 if ~isempty(handles.xy)
751 id = getNearestVertex(pt,handles.xy);
752 handles.xy(id,:) = [];
753 handles.A(id,:) = [];
754 handles.A(:,id) = [];
755 guidata(hObject, handles);
756 hold on
757 cla;
758 if ~isempty(handles.xy)
759 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
760 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy,ax);
761 n2 = size(handles.A,1);
762 plot(handles.axes,ux,uy,'k-','LineWidth',2);
763 plot(handles.axes,dxu,dyu,'r:');
764 plot(handles.axes,dxl,dyl,'b--');
765 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
766 for k = 1:n2
767 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
768 end
769 end
770 set(gcf,'WIndowButtonMotionFcn','');
```

```
771 set(gcf,'WIndowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
772 end
773 end
774 end
775 if ~isempty(handles.xy)
776 set(handles.new,'enable','on');
777 else
778 end
779 set(handles.undo,'enable','on');
780 set(gcf,'WindowButtonUpFcn',{@figure1_WindowButtonUpFcn,
      handles});
781 end
782 % --- Executes on button press in new.
783 function new_Callback(hObject, eventdata, handles)
784 choice = questdlg('Are you sure you want to clear everything
      and redraw ? This operation cannot be undone.', ...
785 'Draw a New Graph', ...
786 'Yes','No','No');
787 % Handle response
788 switch choice
789 case 'Yes'
790 set(handles.undo,'enable','off');
```

```
791 cla;
792 xy=[];
793 A=[];
794 eid=[];
795 id1=[];
796 xytemp=[];
797 Atemp=[];
798 enchk=[];
799 enchk=0;
800 handles.xy = xy;
801 handles.A=A;
802 handles.eid=eid;
803 handles.id1=id1;
804 handles.xytemp = xytemp;
805 handles.Atemp = Atemp;
806 handles.enchk=enchk;
807 handles.prev_scale = 1;
808 handles.fir_cnt = [];
809 guidata(hObject, handles);
810 p=[];F1=[];CRLBmat2=[];A1=[];B=[];C=[];ns=[];CRLBest=[];vstr
      =[];D=[];
811 set(handles.rev,'Units','normalized', 'Data', D);
812 set(handles.poles,'Units','normalized', 'Data', p);
```

```
813 set(handles.fisher,'Units','normalized', 'Data', F1);
814 set(handles.crb,'Units','normalized', 'Data', CRLBmat2);
815 set(handles.a, 'Units','normalized','Data', A1);
816 set(handles.b, 'Units','normalized', 'Data', B);
817 %set(handles.b,'ColumnWidth',{25});
818 set(handles.c,'Units','normalized', 'Data', C);
819 %set(handles.c,'ColumnWidth',{25});
820 set(handles.note,'String',ns);
821 set(handles.crbb,'Units','normalized', 'Data', CRLBest);
822 set(handles.pr,'String',vstr);
823 set(handles.enter,'enable','off');
824 set(handles.vertex,'value',1);
825 set(handles.draw,'value',1);
826 set(handles.save,'value',1);
827 set(handles.undirected,'value',1);
828 set(handles.scale,'Enable','inactive');
829 set(handles.uok,'Enable','off');
830 set(handles.check,'Enable','off');
831 set(handles.check,'Value',0)
832 set(handles.check2,'value',0)
833 set(handles.check2,'enable','off')
834 set(gcf,'WindowButtonMotionFcn',{
      @figure1_WindowButtonMotionFcn,handles});
```

```
835 case 'No'
836 t= get(handles.draw,'value');
837 if  t==3
838 set(gcf,'WindowButtonMotionFcn','');
839 else
840 end
841 otherwise
842 end
843 if isempty(handles.xy)
844 set(handles.new,'enable','off');
845 else
846 end
847 %--- Executes on button press in uok.
848 function uok_Callback(hObject, eventdata, handles)
849 uokk = str2double(get(handles.scale,'String'));
850 if isnan(uokk)
851 errordlg('The scaling factor should be a number','error')
852 else
853 %%% get the input data from the user from "input data" section
      .
854 in_act = str2double(get(handles.actuation,'String'));
855 in_meas= str2double(get(handles.measurement,'String'));
856 in_noise= str2double(get(handles.noise,'String'));
```

```
857 [m,o]= size(handles.xy);
858 if in_act > m %if the input is higher than the number of nodes
859 %this is the first line of the msgbox
860 msgboxText{1} =  'You have entered a value that is higher than
       the number of nodes at actuation location input';
861 %this command creates the actual message box
862 msgbox(msgboxText,'Input number incorrect', 'error');
863 elseif isempty(in_act) %if the input is not a number
864 %this is the first line of the msgbox
865 msgboxText{1} =  'You have tried to input something that is
     NOT a number at actuation location input.';
866 %this command creates the actual message box
867 msgbox(msgboxText,'Input not a number', 'error');
868 elseif isnan(in_act) %if the input is not a number
869 %this is the first line of the msgbox
870 msgboxText{1} =  'Nothing is entered at actuation location
     input.';
871 %this command creates the actual message box
872 msgbox(msgboxText,'Input empty', 'error');
873 elseif in_meas > m %if the input is higher than the number of
     nodes
874 %this is the first line of the msgbox
```

```
875 msgboxText{1} =  'You have entered a value that is higher than
       the number of nodes at measurement location input';
876 %this command creates the actual message box
877 msgbox(msgboxText,'Input number incorrect', 'error');
878 elseif isempty(in_meas) %if the input is not a number
879 %this is the first line of the msgbox
880 msgboxText{1} =  'You have tried to input something that is
      NOT a number at measurement location input.';
881 %this command creates the actual message box
882 msgbox(msgboxText,'Input not a number', 'error');
883 elseif isnan(in_meas) %if the input is not a number
884 %this is the first line of the msgbox
885 msgboxText{1} =  'Nothing is entered at measurement location
      input input.';
886 %this command creates the actual message box
887 msgbox(msgboxText,'Input empty', 'error');
888 elseif isempty(in_noise) %if the input is not a number
889 %this is the first line of the msgbox
890 msgboxText{1} =  'You have tried to input something that is
      NOT a number at Noise varience input.';
891 %this command creates the actual message box
892 msgbox(msgboxText,'Input not a number', 'error');
893 elseif isnan(in_noise) %if the input is not a number
```

```
894 %this is the first line of the msgbox
895 msgboxText{1} =  'Nothing is entered at Noise input.';
896 %this command creates the actual message box
897 msgbox(msgboxText,'Input empty', 'error');
898 else
899 %%% Call the matrix function to calculate several outputs for
        the
900 %%% current input.
901 handles.fix1=str2double(get(handles.scale,'String'));
902 guidata(hObject, handles);
903 sc = uokk;
904 en=1;
905 c2 = get(handles.check2,'value');
906 if c2 == 1
907 fixval = handles.fix1;
908 fix =1;
909 else
910 fix = 0;
911 fixval = 1;
912 end
913 [mat2,p,r,F1,CRLBmat2,A1,B,C,vstr,xymid,ns,CRLBest,posi,kk,D,
        errchk] = matrix(handles.xy,handles.A,in_act,in_meas,
        in_noise,sc,en,fix,fixval,handles.prev_scale);
```

```
914 handles.prev_scale = kk;
915 guidata(hObject,handles);
916 if errchk == 1
917 errordlg('Scaling factor given by user in not valid','error');
918 else
919 [m3,o3]=size(handles.A);
920 count=0;
921 cla;
922 hold on
923 %%% Display the edge weights on respective edges.
924 for e=1:m3
925 for f=e+1:m3
926 if handles.A(e,f) == 1
927 count = count + 1;
928 text(xymid(count,1),xymid(count,2),['  ' num2str(A1(e,f))],'
      Color','k','FontSize',8,'FontWeight','b');
929 else
930 count = count + 1;
931 end
932 end
933 end
934 %%% Display and plot all outputs on GUI.
935 if ~isempty(handles.xy)
```

```
936 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'.')
937 [ux,uy,dxu,dyu,dxl,dyl] = plotGraph(handles.A,handles.xy);
938 n2 = size(handles.A,1);
939 plot(handles.axes,ux,uy,'k-','LineWidth',2)
940 plot(handles.axes,dxu,dyu,'r:')
941 plot(handles.axes,dxl,dyl,'b--')
942 % plot(handles.axes,ix,iy,'ko')
943 plot(handles.axes,handles.xy(:,1),handles.xy(:,2),'k.')
944 for k = 1:n2
945 text(handles.xy(k,1),handles.xy(k,2),['  ' num2str(k)],'Color'
      ,'k','FontSize',12,'FontWeight','b')
946 end
947 end
948 text(handles.xy(in_act,1),handles.xy(in_act,2),['  ' num2str(
      in_act)],'Color','b','FontSize',12,'FontWeight','b')
949 text(handles.xy(in_meas,1),handles.xy(in_meas,2),['  ' num2str
      (in_meas)],'Color','r','FontSize',12,'FontWeight','b')
950 text(handles.xy(posi,1),handles.xy(posi,2),['  ' num2str(posi)
      ],'Color','g','FontSize',12,'FontWeight','b')
951 text(7,9.15,'Blue- Actuation','Color','b','FontSize',8)
952 text(7,8.83,'Red- Measurement','Color','r','FontSize',8)
953 text(7,8.51,'Green- Best Measurement','Color','g','FontSize'
      ,8)
```

```
954 if length(dxl)>=2
955 if  ~isnan(dxl(1,1)+dxl(2,1))
956 text(0.52,9.47,'Directed edges:','FontSize',8)
957 text(0.52,9.15,'-- Blue large to small node','Color','b','
      FontSize',8)
958 text(0.52,8.83,'-- Red small to large node','Color','r','
      FontSize',8)
959 else
960 end
961 if length(dxu)>=2
962 if ~isnan(dxu(1,1)+dxu(2,1))
963 text(0.52,9.47,'Directed edges:','FontSize',8)
964 text(0.52,9.15,'-- Blue large to small node','Color','b','
      FontSize',8)
965 text(0.52,8.83,'-- Red small to large node','Color','r','
      FontSize',8)
966 else
967 end
968 else
969 end
970 set(handles.scale,'Units','normalized', 'String', kk);
971 set(handles.rev,'Units','normalized', 'Data', D);
972 set(handles.poles,'Units','normalized', 'Data', p);
```

```
973 set(handles.fisher,'Units','normalized', 'Data', F1);
974 set(handles.crb,'Units','normalized', 'Data', CRLBmat2);
975 set(handles.a, 'Units','normalized','Data', A1);
976 set(handles.b, 'Units','normalized', 'Data', B);
977 set(handles.b,'ColumnWidth',{25});
978 set(handles.c,'Units','normalized', 'Data', C);
979 set(handles.c,'ColumnWidth',{25});
980 set(handles.note,'String',ns);
981 set(handles.crbb,'Units','normalized', 'Data', CRLBest);
982 set(handles.pr,'String',vstr);
983 end
984 end
985 end
986 % --- Executes on button press in check.
987 function check_Callback(hObject, eventdata, handles)
988 % hObject    handle to check (see GCBO)
989 % eventdata  reserved - to be defined in a future version of
      MATLAB
990 % handles    structure with handles and user data (see GUIDATA
      )
991 c1 = get(handles.check, 'value');
992 if c1 == 0
993 set(handles.uok,'Enable','off');
```

```
994 set(handles.scale,'Enable','inactive');
995 else
996 set(handles.uok,'Enable','on');
997 set(handles.scale,'Enable','on');
998 end
999 %--- Executes on button press in check2.
1000 function check2_Callback(hObject, eventdata, handles)
1001 fix1 = [];
1002 handles.fix1=fix1;
1003 guidata(hObject, handles);
1004 c1 = get(handles.check, 'value');
1005 c2 = get(handles.check2,'value');
1006 if c2 == 1
1007 handles.fix1=str2double(get(handles.scale,'String'));
1008 guidata(hObject, handles);
1009 set(handles.check,'value',0);
1010 set(handles.uok,'Enable','on');
1011 set(handles.scale,'Enable','on');
1012 set(handles.check,'Enable','off');
1013 elseif c2 ==0
1014 handles.fix1=[];
1015 guidata(hObject, handles);
1016 set(handles.check,'enable','on');
```

```
1017 %set(handles.uok,'Enable','off');
1018 set(handles.scale,'Enable','inactive');
1019 set(handles.uok,'Enable','off');
```

APPENDIX B

MATRIX FUNCTION: RETURNS DIFFERENT OUTPUTS IN GUI

```
1 %%% This function calculates different outputs for the current
      user input
2 %%% and retuns to display %%%
3 function[mat2,p,r,F1,CRLBmat2,A1,B,C,vstr,xymid,ns,CRLBest,
     posi,k,D,errchk] = matrix(xy,A,in_act,in_meas,in_noise,sc,
     en,fix,fixval,prev_scale)
4 %%% Get the xy locations and calcuate the distance between the
      vertices
5 [m,o]= size(xy);
6 niCr= m^2;
7 xymid=[];
8 t1=1;
9 for e=1:m
10 for f=e+1:m
11 xymid(t1,:) = (xy(e,:) + xy(f,:)) / 2;
12 t1=t1+1;
13 if t1 == niCr + 1;
14 end
15 end
16 end
17 [m4,o4]=size(xymid);
18 t=1;
19 temp = [niCr,2];
```

```
20 tempsq = [niCr,2];
21 for e=1:m
22 for f=1:m
23 temp(t,:) = xy(e,:) - xy(f,:);
24 t=t+1;
25 if t == niCr + 1;
26 end
27 end
28 end
29 tempsq = temp.^2;
30 tempsqrt = (tempsq(:,1) + tempsq(:,2)).^(0.5);
31 matemp = vec2mat(tempsqrt, m)
32 mat=[];
33 one = eye(m);
34 mat1 = one + matemp;
35 for j= 1:m
36 for k= 1:m
37 mat(j,k) = (mat1(j,k))^-1;
38 end
39 end
40 mat2 = mat .* A;
41 %%% Form the Laplacian matrix L %%%
```

```
42 Ln = (-1)*mat2; % Ln, the values of the input matrix to
       negative values
43 I = eye(size(Ln)); % I, identity matrix, has size of the input
        matrix
44 dg = sum((Ln)'); % dg, vector that stores the values of the
      row sums of the input matrix Ln
45 dg1=-dg;
46 %k = numel(num2str(int32(var)))
47 DD = diag(dg, 0); % D, diagonal matrix of zeros and a main
      diagonal dg (already defined above)
48 L1 = Ln + ((-1)*DD); % L, laplacian matrix  with the inverted
      input entries and diagonal entries equal to the negative of
       row sums.
49 if en==1
50 L3 = L1*sc;
51 k = sc;
52 else
53 end
54 if fix == 1
55 L3 = L1* fixval;
56 k = fixval;
57 elseif fix == 0
58 var = max(dg1(:));
```

```
59 A3 = I - (L1*prev_scale) ;% A, State matrix (Identity matrix -
       Laplacian matrix)
60 cnt2 = 0;
61 for e1= 1:m
62 for f1= 1:m
63 if (A3(e1,f1)<0) || (A3(e1,f1)>1)
64 cnt2 = cnt2 + 1;
65 else
66 end
67 end
68 end
69 sumA3 = sum((A3)');
70 cnt3 = sum(sumA3);
71 if cnt2 ~= 0 || cnt3 ~=m
72 kt = 1/(var + 1);
73 L3 =L1 * kt;
74 k = kt;
75 else
76 L3 = (L1*prev_scale);
77 k=prev_scale;
78 end
79 end
80 L = L3;
```

```
A2 = I - L ;% A, State matrix (Identity matrix - Laplacian
    matrix)
cnt = 0;
for e= 1:m
for f= 1:m
if (A2(e,f)<0) || (A2(e,f)>1)
cnt = cnt + 1;
else
end
end
end
sumA2 = sum((A2)');
cnt1 = sum(sumA2);
if cnt ~= 0 || cnt1 ~=m
errchk = 1;
else
errchk = 0;
end
A1 = I - L;
[D,V]=eigs(full(A1));
B = zeros(m, 1); % ei, vector of zeros, has a row size of the
    input matrix Lin
B(in_act) = 1; % Changing the first entry of ei vector to 1
```

```
C = zeros(1,m);
C(in_meas) = 1;
[b, a] = ss2tf(A1,B,C,0);% Transfer function from state space
[r, p] = residue(b,a); % Respresenting the transfer function
    in a pole-residue form
%%% Calcuate the CRB for user inputted mesurement location %%%
[F,CRLBmat]=crbcalc(p,A1,r,in_noise);
F1=F;
CRLBmat2 = CRLBmat;
%%% Find the location of the best measurement location when
    actauted from
%%% the user inputted actuation point %%%
CRLBes=[];
es=[];
l =1;
for i = 1:m
CRLBmatt=[];
C1 = zeros(1,m);
C1(i) = 1;
[b1, a1] = ss2tf(A1,B,C1,0);
[r2, p2] = residue(b1,a1);
[F2,CRLBmatt]=crbcalc(p2,A1,r2,in_noise);
es(:,l) = CRLBmatt;
```

```
123 l=l+1;
124 end
125 CRLBes=es(2,:);
126 CRLBes(1,in_act) = 1000;
127 least = [];
128 posi=[];
129 least = min(CRLBes(:));
130 for i =1:m
131 if CRLBes(1,i) == least
132 posi =i;
133 else
134 end
135 end
136 CRLBest = [];
137 Best = [];
138 C2 = zeros(1,m);
139 C2(posi) = 1;
140 [b2, a2] = ss2tf(A1,B,C2,0);
141 [r3, p3] = residue(b2,a2);
142 [F3,Best]=crbcalc(p3,A1,r3,in_noise);
143 CRLBest = Best;
144 %%% Calculate the Pole - Residue form %%%
145 p1 = p;
```

```
r1 = r;
for i = 1:m
if p(i,1) < 0 && r(i,1) < 0
p1(i,1) = -p1(i,1);
r1(i,1) = -r1(i,1);
vstr{i} = sprintf('%f\n%s\n%s%f%c\n',r1(i,1), '-
    ----------------', '(z+', p1(i,1), ')');

elseif p(i,1) < 0 && r(i,1) > 0
p1(i,1) = -p1(i,1);
vstr{i} = sprintf('%f\n%s\n%s%f%c\n',r(i,1), '+
    ----------------', '(z+', p1(i,1), ')');

elseif p(i,1) > 0 && r(i,1) > 0
vstr{i} = sprintf('%f\n%s\n%s%f%c\n',r(i,1), '+
    ----------------', '(z-', p(i,1), ')');
else
r1(i,1) = -r1(i,1);
vstr{i} = sprintf('%f\n%s\n%s%f%c\n',r1(i,1), '-
    ----------------', '(z-', p(i,1), ')');
end
end
```

```
ns = sprintf('%s%u\n%s%u\n%s','The dynamics when actuated at
    node:', in_act,' are best observed at node :   ',posi,'The
    Cramer Rao bounds at this node are given below');
%%% the function calculates the Fisher matrix and CRB %%%
function [F,CRLBmat] = crbcalc(p,A1,r,in_noise)
CRLBmat = [];
temp1 = [];
[n2,m2] = size(A1);
for i=1:n2
for j=1:n2
XA(i,j) = 1 / (1-(p(i)*p(j)));
end
end
for i=1:n2
for j=1:n2
XB(i,j) = (r(i)*p(j)) / ((1-(p(i)*p(j)))^2);
end
end
XC = XB';
for i=1:n2
for j=1:n2
XD(i,j) = (r(i)*r(j)*(1-(p(i)*p(j))^2)) / ((1-(p(i)*p(j)))^4);
end
```

```
185 end
186 FA = vertcat(XA,XB);
187 FB = vertcat(XC,XD);
188 F = [FA FB];
189 for i1=1:length(A1)
190 temp1(i1) = ((1-(p(i1)^2))^3)/((r(i1)^2)*(1+(p(i1)^2)));
191 end
192 CRLBmat= (temp1 * in_noise)';
193 end
194 end
```

APPENDIX C

OTHER FUNCTIONS REQUIRED FOR PLOTTING NETWORKS

```
1 %%% Function will plot the graph structure on the GUI axis %%%
2
3 function[ux,uy,dxu,dyu,dxl,dyl] = plotGraph(A,xy,ax)
4                 if nargin < 3
5                     ax = axis;
6                 else
7                     minx = min([ax(1); xy(:,1)]);
8                     maxx = max([ax(2); xy(:,1)]);
9                     miny = min([ax(3); xy(:,2)]);
10                    maxy = max([ax(4); xy(:,2)]);
11                    ax = [minx maxx miny maxy];
12                end
13              [ux,uy,dxu,dyu,dxl,dyl] = dgplot(A,xy);
14               axis(ax);
15          function[ux,uy,dxu,dyu,dxl,dyl] = dgplot(A,xy)
16          n = size(A,1);
17          A = max(0,min(1,ceil(abs(A))));
18          uA = A.*A'.*(1-eye(n));
19          [ux,uy] = makeXY(uA,xy);              % undirected edges
                (2-way)
20          [dxu,dyu] = makeXY(triu(A-uA),xy); % directed edges
                (1-way, sm2lg)
```

```
21          [dxl,dyl] = makeXY(tril(A-uA),xy); % directed edges
                (1-way, lg2sm)
22          %[ix,iy] = makeXY(A.*eye(n),xy)      % self-connecting
                edges
23
24          function [x,y] = makeXY(A,xy)
25              x = NaN;
26              y = NaN;
27              if any(A(:))
28                  [ii,jj] = find(A);
29                  m = length(ii);
30                  xmat = [xy(ii,1) xy(jj,1)]';
31                  ymat = [xy(ii,2) xy(jj,2)]';
32                  x = [xmat; NaN(1,m)]; x = x(:);
33                  y = [ymat; NaN(1,m)]; y = y(:);
34              end
35          end
36          end
37  end
```

C.2. GET NEAREST VERTEX FUNCTION

```
1 %%% This function returns the nearest vertex ID number which
      is nearer to
```

```
%%% the mouse location.%%%
function id = getNearestVertex(pt,xy)
        dsq = sum((xy-pt(ones(size(xy,1),1),:)).^2,2);
        id = find(dsq==min(dsq));
        id = id(1);
end
```

C.3. MAKE XY FUNCTION

```
function [x,y] = makeXY1(A,xy)
            x = NaN;
            y = NaN;
            if any(A(:))
                [ii,jj] = find(A);
                m = length(ii);
                xmat = [xy(ii,1) xy(jj,1)]';
                ymat = [xy(ii,2) xy(jj,2)]';
                x = [xmat; NaN(1,m)]; x = x(:);
                y = [ymat; NaN(1,m)]; y = y(:);
            end
end
```

C.4. GET POINT FUNCTION

```
1 %%% This function returns the location of the mouse pointer on
      GUI axis.%%%
2 function pt = getPoint()
3         cpt = get(gca,'CurrentPoint');
4         pt = cpt(1,1:2);
5             latch = 0.01;
6             if ~isnan(latch)
7                 pt = round(pt/latch)*latch;
8             end
9 end
```

BIBLIOGRAPHY

[1] `http://www.ee.unt.edu/public/wan/`.

[2] S. Boyd, P. Diaconis, and L. Xiao, *Fastest mixing markov chain on a graph*, SIAM Review 46, 667–689.

[3] R. Dhal, S. Roy, Y. Wan, A. Saberi, and P. Corrigan, *Majorizations for the eigenvectors of graph-adjacency matrices*, submitted.

[4] Mary J. Dunlop and Richard M. Murray, *Towards biological system identification: fast and accurate estimates of parameters in genetic regulatory networks*, Proceedings of Conference on Decision and Control, December 2006.

[5] George Casella E.L. Lehmann, Theory of Point Estimation (2nd edition), Springer-Verlag New York, Inc., 1998.

[6] Byron Ellis and Wing Hung Wong, *Learning causal bayesian network structures from experimental data*, Journal of the American Statistical Association 103 (2008).

[7] M Fiedler, *Absolute algebraic connectivity of trees*, Linear and Multilinear Algebra 26, 85–106.

[8] M. Fiedler, *Eigenvectors of acyclic matrices*, Czechoslovak Mathematics Journal 100, 607–618.

[9] J.D. George, R.R. Muise, and J.S. Abel, *Simultaneous estimation of time-of-arrival, pole, and amplitude parameters for transient exponential signals*, Proceedings of Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on, March 1992.

[10] Robert F Guratzsch and Sankaran Mahadevan, *Structural health monitoring sensor place-*

ment optimization under uncertainty, American Institute of Aeronautics and Astronautics(AIAA Journal 2010) 48 (2010), no. 7, 1281–1289.

[11] Cramer Harald, Mathematical Methods of Statistics, Princeton Univ press, Princeton, NJ, 1946.

[12] H Hashimoto, *Distributed cyber attack detection for power network systems*, Proceedings of 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), December 2011.

[13] Martijn P Van Den Heuvel, Rene C W Mandl, Cornelis J Stam, Rene S Kahn, and Hilleke E Hulshoff Pol, *Aberrant frontal and temporal complex network structure in schizophrenia: a graph theoretical analysis*, Journal of Neuroscience, November 2010.

[14] Lieven De Lathauwer Jean-Michel Papy and Sabine Van Huffel, Common pole estimation in multi-channel exponential data modeling, 2006.

[15] Veenadhar Katragadda, Marigona Bokshi, Yan Wan, and Xinrong Li, *An interactive tool to investigate the inference performance of network dynamics from data*, Submitted to the Proceedings of Infotech@Aerospace 2012, American Institute of Aeronautics and Astronautics (AIAA) (California, USA), June 19 - 21 2012.

[16] Steven M. Kay, *Cramer-rao lower bound*, PH sinal processing series, no. 5035, Prentice Hall, Inc. Upper Saddle River, New Jersy 07458, 1993.

[17] Haoni Li, Nan Wang, Ping Gong, Edward J Perkin, and Chaoyang Zhang, *Learning the structure of gene regulatory networks from time series gene expression data*, Proceedings of The 2010 International Conference on Bioinformatics and Computational Biology (BIOCOMP 2010): Genomics (Las Veagas, NV, USA), July 2010.

[18] I.A. Maraziotis, A. Dragomir, and A. Bezerianosi, *Gene networks reconstruction and time-series prediction from microarray data using recurrent neural fuzzy networks*, IET

Systems Biology 1 (2007), 41–50.

[19] L.T. McWhorter and L.L. Scharf, *Cramer-rao bounds for deterministic modal analysis*, IEEE Trans. Signal Proc. SP-41 (1993), 1847–1862.

[20] Jean-Philippe Ovarlez, *Cramer rao bound computation for velocity estimation in the broad-band case using the mellin transform*, Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on, April 1993.

[21] Cynthia Phillips and Laura Painton Swiler, *A graph-based system for network-vulnerability analysis*, Workshop on New security paradigms NSPW 98 (1998), 1998.

[22] Li Shi-Hua and Tang De-Shan, *Power grid crises management and empirical research on small disturbance character of power grid*, Proceedings of Power and Energy Engineering Conference, 2009. APPEEC 2009, March 2009.

[23] Rakesh Kumar Sinha and Amit Kumar Ray, *Effect of acute and chronic heat exposure on frequency of eeg components in different sleep-wake state in young rats*, Iranian Biomedical Journal 8 (2004), no. 2, 69–75.

[24] V. Anne Smith, Jing Yu, Tom V. Smulders, Alexander J. Hartemink, and Erich D. Jarvis, *Computational inference of neural information flow networks*, PLoS Computational Biology 2 (2006), 1436–1449.

[25] Harry L. Van Trees, Detection, Estimation and Modulation Theory, Part I, John Wiley and Sons, 1968.

[26] Y. Wan, K. Numuduri, S. Akula, and M. Varanasi, *The impact of multi-group multi-layer network structure on the performance of distributed consensus building strategies*, International Journal of Robust and Nonlinear Control (available online before print, February 2012).

[27] Y. Wan and S. Roy, *On inference of network time constants from impulse response*

data: graph-theoretic cramer-rao bounds, Proceedings of the 48th IEEE Conference on Decision and Control (Shanghai, China), December 2009.

[28] Y. Wan, S. Roy, and A. Saberi, *Designing spatially-heterogeneous strategies for the control of virus spread*, IET Systems Biology 2 (2008), 184–201.

[29] Yan Wan and Sandip Roy, *On inference of network time constants from impulse response data: Graph-thoretic cramer-rao bounds*, Proceedings of Joint 48th IEEE Conference on Decision and Control 28th Chinese Control Conference (Shanghai, P.R. China), December 2009.

[30] M Xue, S Roy, Y Wan, and S Das, *Security and vulnerability of cyber-physical infrastructure networks: A control-theoretic approach*, (2012).

[31] Ting Hua Yi, Hong Nan Li, and Ming Gu, *Optimal sensor placement for structural health monitoring based on multiple optimization strategies*, The Structural Design of Tall and Special Buildings 20 (2011), no. 7, 891–900.