Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2009-1394703

M.Sc. Thesis

A noise subspace approach for localization in wireless sensor networks

Sajid Aqeel

Abstract

Wireless sensor networks are becoming increasingly popular due to their low cost and wide applicability to support a large number of diverse application areas. Localization of sensor nodes is a fundamental requirement that makes the sensor data meaningful. Energy and cost constraints only allow to equip a few nodes with a GPS device and to localize the remaining nodes with the help of these known locations and a pair-wise range measurements. Multi-dimensional scaling is an attractive localization technique due to a closed-form solution. It however requires pairwise measurements between all nodes to obtain the unknown node coordinates. In this thesis we investigate the feasibility of an analytical solution when some of the dissimilarity measurements are missing. We propose a least squares method to obtain unknown node positions by projecting the squared distance matrix onto the noise subspace of the weight matrix. We evaluate the proposed method for fully connected, and partially connected networks. We show that the proposed method determines the absolute node locations for a fully connected sensor network. For partially connected networks, though it is infeasible to obtain the global node locations using our method, yet we present scenarios where realative node locations can be obtained.

A noise subspace approach for localization in wireless sensor networks

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Engineering

by

Sajid Aqeel
born in Tamman, Pakistan

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"A noise subspace approach for localization in wireless sensor networks"** by **Sajid Aqeel** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 27, 2009

Chairman:

Prof.Dr.ir. Alle-Jan Van der Veen, Technische Universiteit Delft

Advisor:

Dr.ir. Geert Leus, Technische Universiteit Delft

Committee Members:

Dr. Stefan Dulman Committee-Member

Abstract

Wireless sensor networks are becoming increasingly popular due to their low cost and wide applicability to support a large number of diverse application areas. Localization of sensor nodes is a fundamental requirement that makes the sensor data meaningful. Energy and cost constraints only allow to equip a few nodes with a GPS device and to localize the remaining nodes with the help of these known locations and a pair-wise range measurements. Multidimensional scaling is an attractive localization technique due to a closed-form solution. It however requires pairwise measurements between all nodes to obtain the unknown node coordinates. In this thesis we investigate the feasibility of an analytical solution when some of the dissimilarity measurements are missing. We propose a least squares method to obtain unknown node positions by projecting the squared distance matrix onto the noise subspace of the weight matrix. We evaluate the proposed method for fully connected, and partially connected networks. We show that the proposed method determines the absolute node locations for a fully connected sensor network. For partially connected networks, though it is infeasible to obtain the global node locations using our method, yet we present scenarios where realative node locations can be obtained.

Acknowledgments

I would like to thank GOD, Almighty who gave me courage, strength, and health to carry out this work. He is the Most Merciful Who has always bestowed His blessings upon me.

I would like to thank my advisor Dr. Geert Leus. I was fortunate to work under his supervision and from him i learnt the critical and objective way of thinking which is a great asset for my future. He always encouraged and assisted me in difficult times and this effort would never have been possible without his guidance. I would also like to thank the Circuits and Systems group in general and Professor, Dr. Alle-Jan. Van der Veen in particular for providing me the opportunity to work with these great people.

I would also take this opportunity to thank my parents, who always sacrificed their today for my tomorrow. They have always remain a source of inspiration for me. I would also thank my sisters who continously encouraged me to achieve my goals.

I have no words to express my thanks to my wife. I am privileged to have a life partner like her. She always stood with me in hard times and always prayed for my betterment. This little work is dedicated to her. I am also thankful to my little angels Ghania, and Sunnia whose love and support kept me focused.

Last but not the least i would like to thank my friends especially Mazhar, Wajid, Tariq and Shah Muhammad. It was hard to live here without my family but their support made it possible.

Sajid Aqeel
Delft, The Netherlands
August 27, 2009

Contents

1 Introduction

In this thesis we consider the localization problem for wireless sensor networks and present a new approach based on the noise subspace projection of the weight matrix.

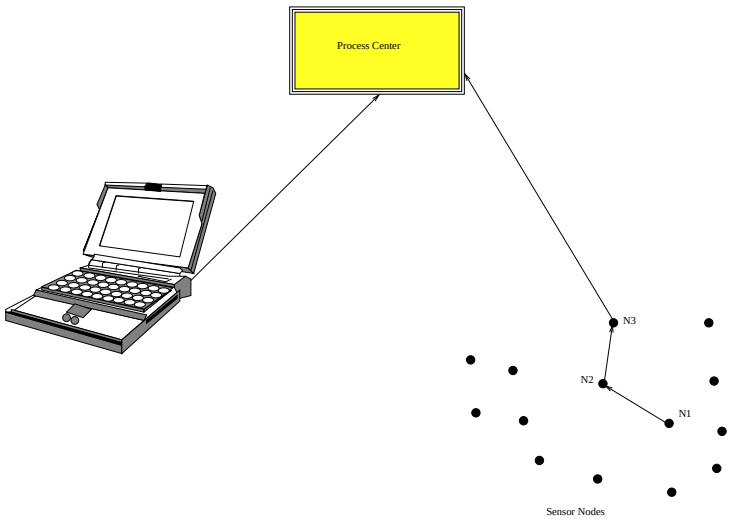
The purpose of this chapter is to define the problem, motivate the need for a new algorithm and to describe our main contributions.

1.1 Motivation: Importance of Localization

The recent advancement in micro-electronic-mechanical systems (MEMS) and wireless communications have made it feasible to design cost effective wireless sensor networks. A typical sensor network is composed of hundreds of tiny, inexpensive, and low-powered sensor nodes, where each node possess limited capabilities which mainly include data collection and communication with neighboring nodes. The cost effectiveness of such networks make them feasible to support a large number of diverse applications like battelfield surveillance, home automation, machine diagnosis and habitat monitoring to name a few [1]

Sensor nodes are deployed in a certain region where some activity needs to be monitored. The nodes group themselves in the form of an irregular ad-hoc network. The role of each node is to monitor its surrounding environment and transmit the captured data to the control center for further processing. For example a sensor network can be deployed in a large forest to warn about smoke indicating the possibility of fire. If there is any such activity, the nearby sensor is supposed to report this event to the control center. Upon receiving such a message the natural question is *Where ?*. Where the abnormality occurred? At which point the response team should immediately proceed? To answer these questions, one must know the precise location of the reporting sensor node (and thereby each node). The data itself is often meaningless unless coupled with the precise location information. Location information not only makes the data meaningful, but also helps in designing efficient routing and distributed signal processing algorithms which in turn help in energy conservation.

To support a large number of low cost applications, the cost of an individual sensor node must be at the lower end. Remote deployment of the sensor nodes require minimum energy consumption, which in turn requires that the sensor nodes should be able to operate for months and years without battery replacement. These two requirements pose a harsh constraint to the hardware with which a sensor node can be equipped. Thus energy and cost constraints do not allow to equip each node with a GPS devise. Moreover GPS can only be used in outdoor environments and is prone to jamming [2]. Manual configuration in which an administrator configures the coordinates during deployment, is also not feasible for large scale networks or in the situations where nodes are not stationary.



The approach normally followed to address the localization problem assumes that a set of nodes called anchor nodes, know their position *a priori* either through GPS or manual configuration, and the remaining nodes get their position estimates using the known anchor location information and pairwise distance information between a set of nodes.

1.2 Localization Challenges

The limited resources of the sensor nodes on one hand and the harsh field conditions on the other make localization quite a challenging task. Each node has a low-power processor, a modest amount of memory and a transceiver. They are battery operated and are expected to work for months and years without battery replacement which poses severe restrictions on their communication and computation capabilities. Nodes are randomly deployed (often dropped from an aeroplane) and organize themselves into a non uniform ad-hoc fashion. The cost and energy constraints do not allow to equip each node with a power amplifier, otherwise they would easily make measurements with a set of reference nodes (with known locations) and localize themselves using triangulation.

As each sensor node is able to transmit and receive, it can obtain some information about the locations of other nodes. The strength of a received signal provides some information regarding the distance of transmitting node. A node receives strong signals from nearby nodes and weaker signals from distant nodes. These pairwise measurements can be used to get a relative estimate of node positions. The accuracy of localization is directly related to the accuracy of distance estimates obtained from these measurements. However the hostile environment in which the sensor nodes are deployed, makes it very hard to obtain error free measurements. The wireless channel is subject to multipath effects and shadowing, hence no matter which ranging technique is used the distance estimates will always be noisy and the accuracy of localization will deteriorate.

Now we formally define the problem and give an overview of the techniques used

for localization.

1.3 Problem Statement

Suppose we have $M = k + m$ sensor nodes

$$\mathbf{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots\cdots \boldsymbol{\phi}_k, \boldsymbol{\phi}_{k+1}, \cdots\cdots \boldsymbol{\phi}_M] \tag{1.1}$$

Where $\boldsymbol{\phi}_i = [x_i \;\; y_i]^T$ is the location of $i-th$ node.

Let the first k nodes are the beacon nodes with known coordinates

$$\mathbf{\Phi}_a = [\boldsymbol{\phi}_1, \;\boldsymbol{\phi}_2, \cdots \boldsymbol{\phi}_k] \tag{1.2}$$

The localization task is then to estimate the coordinates of remaining m sensor nodes, given the coordinates of beacons and a number of pairwise dissimilarity measurements between a set of nodes. i.e

Estimate

$$\mathbf{\Phi}_u = [\boldsymbol{\phi}_{k+1}, \;\boldsymbol{\phi}_{k+2}, \cdots\cdots \boldsymbol{\phi}_M] \tag{1.3}$$

given

$$\mathbf{\Phi}_a = [\boldsymbol{\phi}_1, \;\boldsymbol{\phi}_2, \cdots\cdots \boldsymbol{\phi}_k] \tag{1.4}$$

and δ_{ij}, where δ_{ij} is the dissimilarity measurement between nodes i and j.

1.4 Localization Algorithms

Localization algorithms for wireless sensor networks fall into two main categories. *range based* and *range free*. In range based algorithms each node measures its distance to/from either its neighbors or a set of reference nodes using some ranging technology like (RSS, TDoA, AoA). While in range free algorithms only the connectivity information is used to measure the inter node distances. Range free techniques are much simpler but less accurate than range based techniques. Range based algorithms can further be differentiated from each other by the type of processing (centralized Vs distributed), ranging technology employed (RSS, TDoA, AoA), and the methodology used for localization (Collaborative Vs non collaborative) . Each of these aspects have their own advantages and disadvantages and can outperform the other in terms of accuracy, cost and complexity under different scenarios. Ideally a localization algorithm should find the position of each unknown node with high accuracy and low complexity. The performance of an algorithm depends upon range errors, connectivity (average number of neighbors per node), and total number of anchor nodes (Mostly for non collaborative type) [3].

1.4.1 Centralized Vs Distributed Algorithms

In centralized approaches the ranging data is communicated to a powerful central station which is responsible for computing the coordinates for all the nodes and to send the result back to the respective nodes. The central station responsible for extensive computations is often equipped with many resources. The sensor nodes measure the ranging data and send it to the base station, which computes the coordinates of each node and transmits the result back. For larger networks this two way communication requires efficient routing strategies, otherwise the energy of nodes lying closer to the base station which are responsible for transmitting this information back and forth will quickly be drained out. Hence centralized approach becomes impractical when the size of network exceeds beyond a few dozen nodes. Two famous centralized algorithms are MDS-MAP [4] (based on classical MDS) and convex position estimation[5] (based on SDP).

In distributed algorithms each node is responsible to localizes itself with the help of ranging data. Upon receiving the distance estimates form a set of nodes (at least three), each node finds its position using triangulation. Distributed algorithms are suitable even for the larger networks as they only require local broadcast for communication with their neighbors [3].

1.4.2 Ranging Methodology

Localization algorithms require a fraction of nodes with known coordinates and a set of pairwise measurements to determine the unknown node positions. A node measures the signal transmitted from another node which provides an indication of inter node distance. The commonly used ranging techniques include Received Signal Strength (RSS), Time Difference of Arrival (TDoA), and Angle of Arrival (AoA). The signal strength decays proportional to d^{-2},(in free space), where d is the distance between transmitter and receiver. RSS uses the signal strength to measure the distance between a transmitting and a receiving node. The technique works with the basic functionality already present on each node and does not require any specialized hardware which makes it the most appropriate choice for low cost applications, however it contains noise on the orders of several meters [6]. The main sources of noise in RSS measurements are multipath effects and shadowing [7]. The noise in RSS measurements are multiplicative which make the technique applicable over short distances only [2].

The Time Difference of Arrival (TDoA) is comparatively more accurate than RSS, in which each node is equipped with some additional hardware (a speaker and a microphone) to estimate the inter-node distance. The sources of noise are multipath effects and shadowing just like RSS, however the noise is additive [8] which makes the technique feasible to be used over large distances. However the requirement of extra hardware makes it inappropriate for low cost applications.

A third ranging methodology is Angle of Arrival (AoA) where the direction rather than the distance of a transmitting node is estimated using a sensor array at each node. Difference in arrival times of a transmitted signal at each of the array elements, provides an estimate of AoA. It is the best ranging methodology, however the requirement of much more hardware on each node makes it feasible only for some specialized

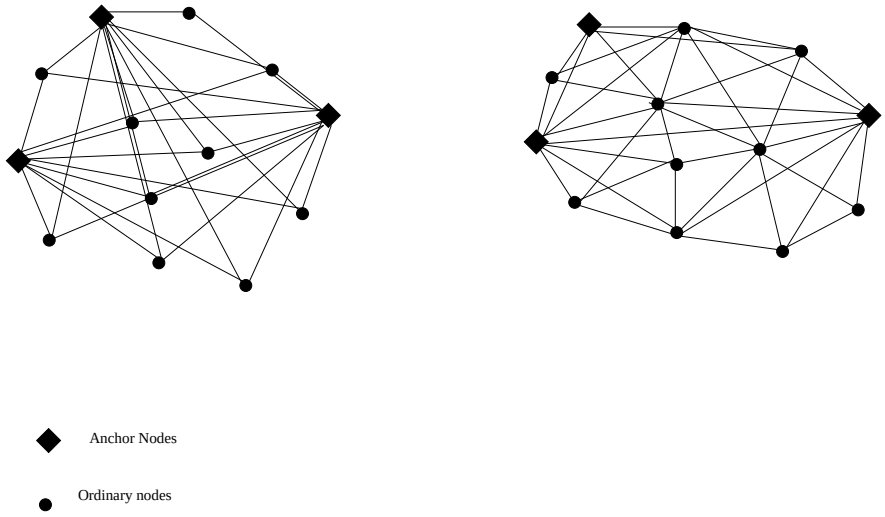


Figure 1.1: Collaborative Vs non collaborative localization

applications.

1.4.3 Collaborative Vs Non Collaborative techniques

There are two different approaches used for localization. In non collaborative localization [9], each node determines distance to the anchor nodes only, whereas in collaborative localization distance is measured between all possible pair of nodes. In non collaborative techniques a node can only be localized when it gets distance estimates from three anchor nodes, however in collaborative type a node can be localized when it is connected to any three nodes (anchors or ordinary nodes). Collaborative localization is a better choice as the addition of a single node reduces the CRB [2].

In this thesis we will assume RSS as the ranging technology due to its simplicity and wide applicability for a large number of low cost applications. We will also consider collaborative technique where each sensor node is able to make pairwise measurements with all possible neighbors (with whom it has a radio connectivity).

1.5 Outline and Contributions

Multidimensional scaling is a popular localization technique that provides a closed-form solution when noise free distance estimates are available between all node pairs. It is highly unlikely to obtain all pair distance measurements due to energy and bandwidth constraints in larger networks. Moreover the distance estimates are corrupted by noise due to multiplath and shadowing effects. Hence the performance of MDS degrades considerably. Many variants of MDS have been proposed in literature to address this drawback. Most of these algorithms iteratively minimize a local cost function and do not offer an analytical solution. There is no such algorithm (to the best of our knowledge) that provides a closed-form solution when some dissimilarity measurements are missing. A contribution of this thesis is to look for the scenarios and determine the conditions

under which the localization problem has a closed-form solution, even in the absence of (some of the) dissimilarity information. We shall also provide a least squares algorithm to determine the unknown node locations.

Ch-2 Multidimensional Scaling

In this chapter we will provide an overview of the MDS and we will discuss some of its variants. We end the chapter with some of the limitations of these techniques thus motivating the need for our algorithm.

Ch-3 A noise subspace approach for Localization

This chapter provides the first contribution of our thesis. We define a weighting matrix $\mathbf{W}$ with $w_{ij} = 1$, if node i and j make a measurement and 0 otherwise. We will then project the Hadamard product of squared distance matrix $\mathbf{R}$ and weight matrix $\mathbf{W}$ on to the noise subspace of $\mathbf{W}$. We show that the unknown node locations can be determined by multiplying the above projection with a known matrix. Simulation results indicate that the proposed method successfully determines the absolute node positions for a fully connected network, however it fails to localize a partially connected network.

Ch-4 Shortcomings of the proposed method

This chapter provides an explanation for the inability of the proposed method for partially connected networks. We provide the necessary and sufficient conditions for the applicability of the proposed method. We also provide some transformation techniques to reduce the rank of weighting matrix $\mathbf{W}$.

Ch-5 The proposed algorithm and rank reduction techniques

We provide a graphical explanation for the non existence of the proposed method for partially connected networks in this chapter. We also evaluate the performance of the proposed method with the rank reduction techniques. We further show that a certain weight matrix can indeed produce relative locations, though it is infeasible to obtain absolute positions.

Ch-6 Conclusions and Future work

In this chapter we conclude our thesis and provide some possible extensions of our work.

2 Multidimensional Scaling

The heart of any localization system is the pair-wise range measurements in which one sensor node measures the signal transmitted from another sensor. These measurements are made using acoustic or RF signals and commonly used ranging techniques are angle of arrival(AoA), time difference of arrival (TDoA), and received signal strength indication (RSSI). An error in the pairwise measurements(range error) greatly degrades the performance of localization algorithms. Range measurements are corrupted by two types of errors, i.e. noise due to multipath and the objects presence in the sensor field (trees,buildings etc) [2]. Received Signal Strength Indication (RSSI), is widely preferred for low cost applications as it works with the basic functionality present on a sensor node (transmit and receive). RSSI is also attractive from energy conservation point of view as the strength of signal can be measured during the normal data transmission. Although Time Difference of Arrival (TDoA), and Angle of Arrival (AoA) are relatively more accurate than RSSI, the extra hardware requirement makes them infeasible for a large number of low cost applications.

2.1 Received Signal Strength RSS

In free space the signal strength decays proportional to d^{-2}, where d is the distance between transmitter and receiver. In a real world, obstructed channel the strength decays proportional to $d^{-\alpha}$, where α is the path loss exponent. A receiving node measures the strength of signal at its received signal strength indicator circuit which provides an estimate to its distance from the transmitting node. Let P_{ij} is the measured RSS from node i to node j, then using path loss model [10]

$$P_{ij} = P_0 - 10\alpha log_{10}\frac{d_{ij}}{\triangle_0} \tag{2.1}$$

For $1 \leqslant$ i,j $\leqslant$ N, and i $\neq$ j. Here P_0 is the measured RSS at a reference distance $\triangle_0$, $d_{i,j}$ is the distance between nodes i and j and α is pathloss exponent which depends upon the environment in which sensor nodes are deployed and is known *a priori*. The RSS measurements suffer from multipath and shadowing effects [7]. Multipath effects give rise to frequency selective fading whose effects can be mitigated by averaging out the received power over a wide range of frequencies [2]. Shadowing arises due to the presence of physical objects in the environment in which the sensor nodes are deployed. From the received RSS measurement P_{ij} between nodes i and j, the MLE of distance between nodes i and j is [2]

$$\delta_{ij} = \triangle_0 10^{(P_0 - P_{ij})/10\alpha} \tag{2.2}$$

δ_{ij} is then used to estimate the unknown location of nodes. Received signal strength is an attractive technique due to its simplicity but the range errors encountered in RSS are multiplicative. The performance of a particular localization algorithm largely depends upon the selection of appropriate weights to combat the effects of range errors.

2.2 Multidimensional Scaling

Multidimensional scaling is a popular method that maps a set of measured dissimilarities to coordinates such that the distance between the node positions fits as much as possible to the measured dissimilarities. The method has been widely used in various fields like economy, behavioral sciences, psychology and political science. The most attractive feature of the MDS is a closed-form solution which makes it superior to its counterparts.

Suppose M sensor nodes are distributed in a certain field. The node coordinate matrix is given by

$$\mathbf{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots\cdots, \boldsymbol{\phi}_M] \tag{2.3}$$

Let the position of the $i-th$ sensor node is $\boldsymbol{\phi}_i = [x_i, y_i]^T$. The noise free distance between the $i-th$ and the $j-th$ sensor node is given as

$$d_{ij} = \parallel \boldsymbol{\phi}_i - \boldsymbol{\phi}_j \parallel = \sqrt{(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^T(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)} \tag{2.4}$$

$$d_{ij}^2 = (\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^T(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)$$

$$d_{ij}^2 = \boldsymbol{\phi}_i^T\boldsymbol{\phi}_i - 2\boldsymbol{\phi}_i\boldsymbol{\phi}_j + \boldsymbol{\phi}_j^T\boldsymbol{\phi}_j$$

Then the squared distance matrix $\mathbf{R}$ is given as

$$\mathbf{R} = \begin{pmatrix} 0 & d_{1,2}^2 & d_{1,3}^2 & \cdots & d_{1,M}^2 \\ d_{2,1}^2 & 0 & d_{2,3}^2 & \cdots & d_{2,M}^2 \\ d_{3,1}^2 & d_{3,2}^2 & 0 & \cdots & d_{3,M}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{M,1}^2 & d_{M,2}^2 & d_{M,3}^2 & \cdots & 0 \end{pmatrix}$$

The squared distance matrix $\mathbf{R}$ satisfies the conditions of self-similarity (i.e. $\delta_{ii} = 0$) and symmetry (i.e. $\delta_{ij} = \delta_{ji}$). If every node successfully makes a distance estimate with every other node, than the squared distance matrix will be fully known. The MDS multiplies the fully known squared distance matrix $\mathbf{R}$ by a centering matrix $\mathbf{J}$ on both sides to obtain an inner product matrix $\mathbf{K}$ (also called Gram matrix) as

$$\mathbf{K} = -0.5\mathbf{JRJ} = \mathbf{J}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{J} \tag{2.5}$$

Where $\mathbf{J}$ is defined as

$$\mathbf{J} = \mathbf{I}_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^T$$

where $\mathbf{I}_M$ is an $M \times M$ identity matrix and $\mathbf{1}_M$ is an $M \times 1$ vector of all ones.

Classical Multidimensional Scaling (CMDS) then performs the eigendecomposition of the inner product matrix $\mathbf{K}$ to extract the eigenvalues and eigenvectors

$$\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{2.6}$$

Let $\mathbf{\Lambda}_2 = diag(\lambda_1, \lambda_2)$ be the matrix of two largest eigenvalues values and $\mathbf{U}_2 = (\mathbf{u}_1, \mathbf{u}_2)$ is the matrix of corresponding eigenvectors then the coordinate matrix $\mathbf{\Phi}$ upto a translation, rotation and reflection is given as

$$\mathbf{\Phi} = \mathbf{U_2}\mathbf{\Lambda_2^{\frac{1}{2}}}$$

Assuming that a complete set of dissimilarity information is available, the Gram matrix $\mathbf{K}$ will be a rank 2 matrix, whose eigendecomposition will determine the relative node locations. The relative node locations can than be converted to absolute positions with the help of anchor nodes. It is an attractive technique due to an analytical solution, however it has a number of shortcomings

1. It requires pairwise distances between all M sensor nodes.
2. It is computationally very expensive as the eigendecomposition of Gram matrix $\mathbf{K}$ of size M requires $O(M^3)$ time.
3. MDS requires true distance estimates. However in real scenarios the distance measurements are corrupted by noise and the method performs poorly.

Due to energy and bandwith constraints, it is often not practical to obatin all pairwise distances. Moreover for larger networks it is quite infeasible to communicate the distance information to a central station. When distance estimates are corrupted by noise, the algorithm minimizes the squared error between actual distances and dissimilarities [11], which further deteriorates the performance.

MDS-MAP[4] is a direct application of the classical metric MDS. The method uses Dijkstra's algorithm to compute the shortest path distance between all nodes to populate the squared distance matrix $\mathbf{R}$. It then applies the MDS algorithm to generate the relative map, which is then transformed to a global map using a few known locations. MDS-MAP is inherently centralized in nature and computationally very expensive. It can only be used for small scale networks. Its performance also deteriorates for irregular shaped networks, where the shortest path distance differs from the Euclidean distance between nodes.

A number of different techniques have been proposed to reduce the computational complexity of CMDS [12],[13], [14]. Fastmap [12] selects two farthest objects as pivot objects and projects all the remaining objects on the line joining pivot objects thereby finding the first coordinate of each node. The $2 - nd$ coordinate is then obtained by projecting them on to a hyper-plane orthogonal to the line. The performance of the algorithm is largely dependent upon the careful selection of pivot objects, which must

be the two farthest objects (from each other), otherwise the algorithm may not be able to localize some nodes.

Another famous technique that addresses the complexity of MDS is Landmark MDS (LMDS) [14]. Unlike Fastmap that finds a one dimensional embedding at a time, LMDS finds a k-dimensional embedding of data points at once. The algorithm consists of following three steps

1. Designate a set of m Landmarks
2. Find a k-dimensional embedding of Landmark points using classical MDS
3. Embed the remaining points in R^k

LMDS finds a k-dimensional embedding both for the landmark points and remaining points. A slightly different approach Metricmap[13] selects $2-k$ objects and maps them into a k-dimensional space by retaining the k-largest eigenvalues. The remaining objects are then also mapped to the k-dimensional space.

Platt[15] has shown that LMDS, Fastmap and Metricmap are based on Nystrom approximation. Nystrom approximation permits to compute the eigenvalues and eigenvectors of a low-rank matrix for a subset of data. Classical MDS converts the distance matrix $\mathbf{R}$ to a kernel matrix(matrix of dot products) $\mathbf{K}$ through double centering. To explain the application of the Nystrom method, let m items are selected in distance and kernel matrices randomly, and assume that the m items constitute the first rows and columns of $\mathbf{K}$ and $\mathbf{R}$, i.e.

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{bmatrix}$$

Where $\mathbf{A}$ and $\mathbf{E}$ have dimensions $m \times m$, $\mathbf{B}$ and $\mathbf{F}$ have dimension $m \times (N-m)$ and $\mathbf{C}$ and $\mathbf{G}$ have dimensions $(N-m) \times (N-m)$. By Nystrom method the coordinates can be found by only utilizing the information in $\mathbf{A}$ and $\mathbf{B}$ only.

As $\mathbf{K}$ is positive semi-definite so it can be expressed as

$$\mathbf{K} = \begin{bmatrix} \mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{Y} \\ \mathbf{Y}^T\mathbf{X} & \mathbf{Y}^T\mathbf{Y} \end{bmatrix}$$

Thus

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} \tag{2.7}$$

and

$$\mathbf{B} = \mathbf{X}^T\mathbf{Y} \tag{2.8}$$

$\mathbf{X}$ can be obtained by the eigendecomposition of $\mathbf{A}$

$$\mathbf{X} = \mathbf{\Gamma}_k^{\frac{1}{2}}\mathbf{U}_k$$

Then from eq (2.8)

$$\mathbf{Y} = \mathbf{X}^{-T}\mathbf{B} = \mathbf{\Gamma}_k^{-1/2}\mathbf{U}_k\mathbf{B}$$

LMDS selects m points and embeds them into a k-dimensional subspace using the Nystrom method. Fastmap chooses two points and finds one dimensional embedding. The Metricmap chooses different dimensions of $\mathbf{A}$ $(2k \times 2k)$ and $\mathbf{B}$ $(k \times k)$. All these methods first solve the problem for a small number of points and than map the remaining points onto the subspace spanned by these points. The computational complexity associated with the MDS is thereby reduced as these methods eigendecompose a small matrix $(k \times k)$ instead of the original $(M \times M)$ matrix. Though these techniques address the computational complexity of MDS, yet they do not address the scalability issue.

2.2.1 Distributed MDS

The communication of all pairwise distances to a central station for further processing puts a lot of overhead from the energy dissipation point of view. For large scale networks consisting of hundreds of nodes it becomes a bottleneck to communicate this information to/from the central station. A number of variants of classical MDS have been proposed to address this problem.

In a distributed version of the MDS-MAP algorithm [16] each node applies MDS-MAP to nearby nodes only to get a local map . Only 2-hop neighbors are considered and the shortest path distance between them is computed to obtain local distance matrix $\mathbf{R}$. The local maps are generated by applying the MDS technique to all local distance matrices. The local maps thus formed are eventually combined together using a linear transformation to get a global map.

A sampling based FastMDS has been proposed in [17]. Since the submatrix along the diagonal of a dissimilarity matrix $\mathbf{R}$ is itself a dissimilarity matrix, the MDS algorithm is run on each submatrix instead of the original dissimilarity matrix. After finding a local solution a few points are selected from each submatrix and are put in an alignment matrix $\mathbf{M}_{align}$. The MDS algorithm is then run on $\mathbf{M}_{align}$ to get a global solution. The local and global solutions are than mapped through an affine mapping to find a common coordinate solution.

$$\mathbf{A}_i dMDS_i = mMDS_i$$

Once $\mathbf{A}_i$ is obtained, it is applied to non sampling points of each submatrix to get a solution in the global space.

2.2.2 Weighted MDS

The major drawbacks of the MDS are computational complexity (eigendecomposition of a dense $(M \times M)$ matrix) , communication overhead, the requirement of distance measurements between all nodes and unreliable performance under noisy conditions. Nystrom approximation provides a solution of complexity issue, while communication overhead can be reduced using a distributed approach. However none of the algorithms described so for address the third drawback of MDS.

Weighted MDS on the other hand, relaxes the requirement of pairwise distance measurement between all nodes. It can also mitigate the effects of noisy distance estimates by selecting suitable weights.

The objective of the MDS is to find the coordinates of the points such that the interpoint distance should match as closely as possible to the measured dissimilarities. The objective can be formulated in a least square sense as

$$S(X) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w_{ij}(d_{ij} - \delta_{ij})^2 \tag{2.9}$$

Where d_{ij} is the actual Euclidean distance between nodes i and j and δ_{ij} is the dissimilarity measure between them.

The above stress function can be minimized by selecting appropriate weights. The weight selection should be done in such a way so that the contribution of the less accurate measurements is minimized. The nodes from which there is no distance estimate available, can be excluded by selecting their respective weights as zero.

In [18] the selection of weights is done based on the dissimilarity measure between two nodes. If the distance between two nodes is small a larger weight is chosen and if it is large, a smaller weight is chosen. Their method outperforms CMDS, however intuitively the algorithm must have an unnecessary pull for a node which is surrounded by a large number of neighbors in one particular direction.

Costa et al. [11] uses the following stress function

$$\text{S}= 2 \sum_{1\le i\le n} \sum_{i\le j\le n+m} \sum_{1\le t\le K} w_{ij}^{(t)}(\delta_{ij}^t - d_{ij}(\mathbf{\Phi}))^2 + \sum_{1\le i\le n} r_i \|\phi_i - \phi^-(i)\|^2$$

Costa algorithm assumes that each node is able to obtain K dissimilarity measures from its neighbors. The K-neighbors are any set of nodes from which an estimate is available. The last term in the stress function takes care of some prior knowledge about the node positions (if any). The weights are chosen as

$$w_{ij} = \begin{cases} exp(-\delta_{ij}^2/h_{ij}^2) & \text{if } \delta_{ij} \text{ is measured} \\ 0 & \text{otherwise} \end{cases}$$

Where $h_{ij} = \max(\ \max\ x_k\delta_{ik},\ max x_k\delta_{kj})$. Neighbor selection is the most critical phase as because of the noisy distances, incorrect neighbors may be selected. To mitigate the noise effects a two step procedure is adopted. In the first step it sets $w_{ij} = 0$ if $\delta_{ij} > d_R$. A rough estimate of the coordinates is obtained after convergence so in the second step it refines the neighbor selection by setting $w_{ij} = 0$ if $\|x_i - x_j\| > d_R$.

A similar approach has been used in [19], which minimizes the distributed cost function with an additional term that corresponds to some prior knowledge about the network deployment.

$$\text{S}= 2 \sum_{1\le i\le n} \sum_{i\le j\le n+m} w_{ij}(\delta_{ij} - d_{ij}(\mathbf{\Phi}))^2 + \sum_{1\le i\le n} f(\phi_i)$$

The penalty function $f(x_i)$ is given as

$$f(x_i) = \begin{cases} \sum\limits_{j \in N_i} (min(d_{ij}(\mathbf{\Phi}), d_{min}) - d_{min})^2 & d_{ij}(\Phi) \leq d_{min} \\ \sum\limits_{j \in N_i} (max(d_{ij}(\mathbf{\Phi}), d_{max}) - d_{max})^2 & d_{ij}(\Phi) \geq d_{max} \\ 0 & j \notin N_i \end{cases}$$

Where N_i is the set of 2-hop neighbors while d_{min} and d_{max} are a priori minimum and maximum spacing between 2-hop nodes. The nodes violating the spacing constraints, and not having distance estimates are penalized by the penalty term.

All of the methods described above tend to minimize the cost function in a distributed manner to reduce the cost associated with centralized processing. The methods are applicable with a partial set of dissimilarity information which makes them a suitable candidate for real scenarios. However the major drawback of the weighted techniques is their possible convergence to a local optimum. Their convergence speed is also dependent upon perfect initialization.

2.3 Conclusion

Localization in the wireless sensor networks have got a lot of attention in the last few years. Many collaborative and non collaborative techniques have been proposed to address the issue. In the domain of collaborative localization, MDS is the most popular method due to an analytical solution. However its drawbacks often make it impractical to be utilized in noisy environments and in the situations where all pairwise measurements are not available between nodes. Weighted MDS aims to get a faithful embedding by only utilizing a few distance estimates. However the techniques do not have a closed-form solution and are subject to local minima. The neighborhood biasing also deteriorates the performance of these techniques

There is a need to determine the conditions under which the localization problem is solvable with missing dissimilarity information. It would be quite nice to have an analytical solution in the presence of missing dissimilarity information. In the next chapter we will introduce an algorithm for solving the problem with missing distances.

A noise subspace approach for localization

3

We discussed the multidimensional scaling and its limitations in the previous chapter. A significant limitation of MDS is that it requires dissimilarity information between all pair of nodes. In reality the power and bandwidth constraints make it impractical to get dissimilarity information between all nodes. When the dissimilarity information is missing between a set of nodes, MDS performs poorly. A number of algorithms have been proposed [11], [19] to address this drawback of MDS. These methods optimize a weighted cost function to find the node positions in presence of missing dissimilarity information. Only those nodes which make a dissimilarity measure, contribute in the optimization of the cost function. However unlike MDS, these techniques do not have a closed-form solution. Here we shall describe a method based on the noise subspace projection of the weight matrix and we will determine the conditions under which the localization problem with missing dissimilarity information can have a closed-form solution.

3.1 A noise subspace approach

Assume that M sensor nodes $\mathbf{\Phi} = [\boldsymbol{\phi}_1,\ \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_M]$ are randomly placed in a certain field. Let the position of the $i-th$ sensor node be $\boldsymbol{\phi}_i = [x_i, y_i]^T$. We also assume that the first k nodes know their position *a priori* through GPS or manual configuration. We wish to determine the coordinates of the remaining $M-k$ nodes with the help of k anchor nodes and a partial set of dissimilarity information.

The coordinate matrix $\mathbf{\Phi}$ can be decomposed into an anchor node matrix and an unknown node matrix as

$$\mathbf{\Phi} = [\mathbf{\Phi}_a \ \ , \ \ \mathbf{\Phi}_u]$$

where $\mathbf{\Phi}_a$ is the matrix related to the k anchor nodes, and $\mathbf{\Phi}_u$ is the matrix related to the $M-k$ unknown node positions to be determined.

We assume that the M sensor nodes measure the dissimilarity information between themselves, using any ranging technology (RSS, TOA or AoA). We further assume (for the time being) that the measured dissimilarity information is free of range errors. The distance between the $i-th$ and $j-th$ sensor node is given by

$$d_{ij} = \sqrt{(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)^T(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)} \tag{3.1}$$

The squared distance matrix $\mathbf{R}$ is given as

$$\mathbf{R} = \begin{bmatrix} 0 & d_{1,2}^2 & d_{1,3}^2 & \cdots & d_{1,M}^2 \\ d_{2,1}^2 & 0 & d_{2,3}^2 & \cdots & d_{2,M}^2 \\ d_{3,1}^2 & d_{3,2}^2 & 0 & \cdots & d_{3,M}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{M,1}^2 & d_{M,2}^2 & d_{M,3}^2 & \cdots & 0 \end{bmatrix} \tag{3.2}$$

Let $\mathbf{a}$ is an $M \times 1$ column vector defined as

$$\mathbf{a} = \begin{bmatrix} \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 \\ \vdots \\ \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \end{bmatrix} \tag{3.3}$$

and $\mathbf{1}$ is an $M \times 1$ column vector of all ones then

$$\mathbf{a1}^T = \begin{bmatrix} \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 \\ \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 \\ \vdots \\ \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 \\ \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M & \cdots & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \end{bmatrix} \tag{3.4}$$

and

$$\mathbf{1a}^T = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \cdots \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \\ \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \\ \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T \boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \end{bmatrix} \tag{3.5}$$

and finally $\mathbf{\Phi}^T \mathbf{\Phi}$

$$\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \boldsymbol{\phi}_2^T \\ \boldsymbol{\phi}_3^T \\ \vdots \\ \boldsymbol{\phi}_M^T \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \boldsymbol{\phi}_3 & \cdots \boldsymbol{\phi}_M \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_M \\ \boldsymbol{\phi}_2^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_M \\ \boldsymbol{\phi}_3^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_M^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \end{bmatrix} \quad (3.6)$$

Combining (3.4), (3.5) and (3.6), we obtain

$$\mathbf{a1}^T + \mathbf{1a}^T - 2\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \cdots & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 \\ \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \end{bmatrix} + \begin{bmatrix} \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \\ \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \\ \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \end{bmatrix}$$

$$-2 \begin{bmatrix} \boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_1^T\boldsymbol{\phi}_M \\ \boldsymbol{\phi}_2^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_2^T\boldsymbol{\phi}_M \\ \boldsymbol{\phi}_3^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_3^T\boldsymbol{\phi}_M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_M^T\boldsymbol{\phi}_1 & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_2 & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_3 & \cdots & \boldsymbol{\phi}_M^T\boldsymbol{\phi}_M \end{bmatrix}$$

$$\mathbf{a1}^T + \mathbf{1a}^T - 2\mathbf{\Phi}^T\mathbf{\Phi} = \begin{bmatrix} \phi_1^T\phi_1 + \phi_1^T\phi_1 - 2\phi_1^T\phi_1 & \phi_1^T\phi_1 + \phi_2^T\phi_2 - 2\phi_1^T\phi_2 & \cdots & \phi_1^T\phi_1 + \phi_M^T\phi_M - 2\phi_1^T\phi_M \\ \phi_2^T\phi_2 + \phi_1^T\phi_1 - 2\phi_2^T\phi_1 & \phi_2^T\phi_2 + \phi_2^T\phi_2 - 2\phi_2^T\phi_2 & \cdots & \phi_2^T\phi_2 + \phi_M^T\phi_M - 2\phi_2^T\phi_M \\ \vdots & \vdots & \ddots & \vdots \\ \phi_M^T\phi_M + \phi_1^T\phi_1 - 2\phi_M^T\phi_M & \phi_M^T\phi_M + \phi_2^T\phi_2 - 2\phi_M^T\phi_2 & \cdots & \phi_M^T\phi_M + \phi_M^T\phi_M - 2\phi_M^T\phi_M \end{bmatrix} \quad (3.7)$$

Comparing (3.2) and (3.7), we get

$$\mathbf{R} = \mathbf{a1}^T + \mathbf{1a}^T - 2\mathbf{\Phi}^T\mathbf{\Phi} \quad (3.8)$$

If every node makes a dissimilarity measurement with every other node in the network, then **R** will be non zero everywhere except at the diagonal and the network will be fully connected. The classical MDS multiplies the fully connected squared distance matrix **R** with a centering matrix on both sides to obtain an inner product matrix whose eigendecomposition then provides the required coordinates. MDS guarantees

a global minimum of the cost function for a fully connected network. However with missing dissimilarity information, the method cease to work.

We define an $M \times M$ weighting matrix $\mathbf{W}$ whose $(i,j)th$ entry will be one if a dissimilarity measurement between nodes i and j is made and zero otherwise :

$$w_{ij} = \begin{cases} 1 & \text{if } \delta_{ij} \text{ is measured} \\ 0 & \text{otherwise} \end{cases}$$

Now we take the Hadamard product between the squared distance matrix $\mathbf{R}$ and the weighting matrix $\mathbf{W}$. We will then perform an eigendecomposition of the weight matrix and we will project $\mathbf{R} \odot \mathbf{W}$ onto the noise subspace of $\mathbf{W}$. We will show that the unknown node positions can be determined from this product.

The Hadamard product between $\mathbf{R}$ and $\mathbf{W}$ can be written as

$$\mathbf{R} \odot \mathbf{W}=\text{diag}(\mathbf{a})\ \mathbf{W} + \mathbf{W}\ \text{diag}(\mathbf{a^T})\ \text{-2}\ (\mathbf{\Phi}^T\mathbf{\Phi}) \odot \mathbf{W}$$

Performing an SVD of $\mathbf{W}$, we get

$$\mathbf{W} = \mathbf{U}_s\mathbf{\Sigma}_s\mathbf{U}_s^T \tag{3.9}$$

where $\mathbf{U}_s = [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_r]$ is an $M \times r$ matrix containing the left/right singular vectors and $\mathbf{\Sigma}_s$ is an $r \times r$ diagonal matrix containing the singular values. Since $\mathbf{W}$ is a symmetric matrix of rank r, it follows

$$\mathbf{U}_n\mathbf{U}_n^T = \mathbf{I}_M - \mathbf{U}_s\mathbf{U}_s^T \tag{3.10}$$

where $\mathbf{U}_n = [\mathbf{u}_{r+1}, \mathbf{u}_{r+2}, ...\mathbf{u}_M]$ denotes the noise subspace and $\mathbf{I}_M$ is the $M \times M$ identity matrix.

Multiplying $\mathbf{R} \odot \mathbf{W}$ with noise subspace matrix $\mathbf{U}_n$ and its transpose on right and left respectively, we get

$$\begin{aligned} \mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n &= \mathbf{U}_n^T[(\text{diag}(\mathbf{a})\mathbf{W} + \mathbf{W}\text{diag}(\mathbf{a}^T) - 2(\mathbf{\Phi}^T\mathbf{\Phi}) \odot \mathbf{W})]\mathbf{U}_n \\ &= \mathbf{U}_n^T\text{diag}(\mathbf{a})\mathbf{W}\mathbf{U}_n + \mathbf{U}_n^T\mathbf{W}\text{diag}(\mathbf{a}^T)\mathbf{U}_n - 2\mathbf{U}_n^T[(\mathbf{\Phi}^T\mathbf{\Phi}) \odot \mathbf{W})]\mathbf{U}_n \end{aligned} \tag{3.11}$$

But since

$$\mathbf{U}_n^T\mathbf{W} = 0$$

and

$$\mathbf{W}\mathbf{U}_n = 0$$

we obtain

$$\mathbf{U}_n^T\text{diag}(\mathbf{a})\mathbf{W}\mathbf{U}_n = 0 \tag{3.12}$$

and

$$\mathbf{U}_n^T \mathbf{W} \text{diag}(\mathbf{a^T}) \mathbf{U}_n = 0 \tag{3.13}$$

which allows us to rewrite (3.11) as

$$\mathbf{U}_n^T (\mathbf{R} \odot \mathbf{W}) \mathbf{U}_n = -2\mathbf{U}_n^T (\mathbf{\Phi}^T \mathbf{\Phi} \odot \mathbf{W}) \mathbf{U}_n \tag{3.14}$$

We can determine the node positions from the above equation as the left hand side consists of all known quantities. To determine the node positions we rewrite $\mathbf{\Phi}^T \mathbf{\Phi} \odot \mathbf{W}$ as

$$\mathbf{\Phi}^T \mathbf{\Phi} \odot \mathbf{W} = \sum_{i=1}^{r} [\text{diag}(\mathbf{u}_i) \mathbf{\Phi}^T \mathbf{\Phi} \text{diag}(\mathbf{u}_i)] \lambda_i \tag{3.15}$$

The proof of (3.15) is given next

By definition, every element of the Hadamard product between $\mathbf{\Phi}^T \mathbf{\Phi}$ and the weight matrix $\mathbf{W}$ is given by

$$[\mathbf{\Phi}^T \mathbf{\Phi} \odot \mathbf{W}]_{jk} = [\mathbf{\Phi}^T \mathbf{\Phi}]_{jk} [\mathbf{W}]_{jk} \tag{3.16}$$

Since the rank of $\mathbf{W}$ is r. we can write

$$\mathbf{W} = \sum_{i=1}^{r} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$$

where the $\mathbf{u}_i' s$ are the left/right singular vectors and $\lambda_i' s$ are the singular values. Thus we can rewrite (3.16) as

$$[\mathbf{\Phi}^T \mathbf{\Phi} \odot \mathbf{W}]_{jk} = [\mathbf{\Phi}^T \mathbf{\Phi}]_{jk} [\sum_{i=1}^{r} \lambda_i \mathbf{u}_i \mathbf{u}_i^T]_{jk}$$

Since we can express

$$\sum_{i=1}^{r} \lambda_i \mathbf{u}_i \mathbf{u}_i^T = \sum_{i=1}^{r} \lambda_i \begin{bmatrix} u_{1i}u_{1i} & u_{1i}u_{2i} & \cdots & u_{1i}u_{Mi} \\ u_{2i}u_{1i} & u_{2i}u_{2i} & \cdots & u_{2i}u_{Mi} \\ \vdots & \vdots & \ddots & \vdots \\ u_{Mi}u_{1i} & u_{Mi}u_{2i} & \cdots & u_{Mi}u_{Mi} \end{bmatrix}$$

the Hadamard product between $\mathbf{\Phi}^T \mathbf{\Phi}$ and the weight matrix $\mathbf{W}$ can be written as

$$\mathbf{\Phi}^T \mathbf{\Phi} \odot \mathbf{W} = \sum_{i=1}^{r} \lambda_i \begin{bmatrix} \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^T \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_1^T \boldsymbol{\phi}_M \\ \boldsymbol{\phi}_2^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_2^T \boldsymbol{\phi}_M \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_M^T \boldsymbol{\phi}_1 & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_M^T \boldsymbol{\phi}_M \end{bmatrix} \odot \begin{bmatrix} u_{1i}u_{1i} & u_{1i}u_{2i} & \cdots & u_{1i}u_{Mi} \\ u_{2i}u_{1i} & u_{2i}u_{2i} & \cdots & u_{2i}u_{Mi} \\ \vdots & \vdots & \ddots & \vdots \\ u_{Mi}u_{1i} & u_{Mi}u_{2i} & \cdots & u_{Mi}u_{Mi} \end{bmatrix} \tag{3.17}$$

$$= \sum_{i=1}^{r} \lambda_i \begin{bmatrix} u_{1i}\boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 u_{1i} & u_{1i}\boldsymbol{\phi}_1^T\boldsymbol{\phi}_2 u_{2i} & \cdots & u_{1i}\boldsymbol{\phi}_1^T\boldsymbol{\phi}_M u_{Mi} \\ u_{2i}\boldsymbol{\phi}_2^T\boldsymbol{\phi}_1 u_{1i} & u_{2i}\boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 u_{2i} & \cdots & u_{2i}\boldsymbol{\phi}_2^T\boldsymbol{\phi}_M u_{Mi} \\ \vdots & \vdots & \ddots & \vdots \\ u_{Mi}\boldsymbol{\phi}_M^T\boldsymbol{\phi}_1 u_{1i} & u_{Mi}\boldsymbol{\phi}_M^T\boldsymbol{\phi}_2 u_{2i} & \cdots & u_{Mi}\boldsymbol{\phi}_M^T\boldsymbol{\phi}_M u_{Mi} \end{bmatrix} \tag{3.18}$$

Observing

$$\text{diag}(\mathbf{u}_i)\boldsymbol{\Phi}^T = \begin{bmatrix} u_{1i}\boldsymbol{\phi}_1^T \\ u_{2i}\boldsymbol{\phi}_2^T \\ \vdots \\ u_{Mi}\boldsymbol{\phi}_M^T \end{bmatrix}$$

and

$$\boldsymbol{\Phi}\text{diag}(\mathbf{u}_i) = \begin{bmatrix} \boldsymbol{\phi}_1 u_{1i} & \boldsymbol{\phi}_2 u_{2i} & \cdots & \boldsymbol{\phi}_M u_{Mi} \end{bmatrix}$$

we also have

$$\text{diag}(\mathbf{u}_i)\boldsymbol{\Phi}^T\boldsymbol{\Phi}\text{diag}(\mathbf{u}_i) = \begin{bmatrix} u_{1i}\boldsymbol{\phi}_1^T\boldsymbol{\phi}_1 u_{1i} & u_{1i}\boldsymbol{\phi}_1^T\boldsymbol{\phi}_2 u_{2i} & \cdots & u_{1i}\boldsymbol{\phi}_1^T\boldsymbol{\phi}_M u_{Mi} \\ u_{2i}\boldsymbol{\phi}_2^T\boldsymbol{\phi}_1 u_{1i} & u_{2i}\boldsymbol{\phi}_2^T\boldsymbol{\phi}_2 u_{2i} & \cdots & u_{2i}\boldsymbol{\phi}_2^T\boldsymbol{\phi}_M u_{Mi} \\ \vdots & \vdots & \ddots & \vdots \\ u_{Mi}\boldsymbol{\phi}_M^T\boldsymbol{\phi}_1 u_{1i} & u_{Mi}\boldsymbol{\phi}_M^T\boldsymbol{\phi}_2 u_{2i} & \cdots & u_{Mi}\boldsymbol{\phi}_M^T\boldsymbol{\phi}_M u_{Mi} \end{bmatrix} \tag{3.19}$$

Using (3.19) in (3.18), we obtain the result of (3.15)

Let us now define

$$\mathbf{N}_i = \boldsymbol{\Phi}\text{diag}(\mathbf{u}_i) \tag{3.20}$$

where $\mathbf{N}_i \in \Re^{2\times M}$. Vertically concatenating r such matrices, we obtain the $2r \times M$ matrix $\mathbf{N}$ as

$$\mathbf{N} = \begin{pmatrix} \mathbf{N}_1 \\ \mathbf{N}_2 \\ \vdots \\ \mathbf{N}_r \end{pmatrix} \tag{3.21}$$

Then (3.15) can be written as

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi} \odot \mathbf{W} = \mathbf{N}^T\mathbf{L}\mathbf{N} \tag{3.22}$$

where $\mathbf{L}$ is given by

$$\mathbf{L} = \text{diag}(\lambda_1, \lambda_1, \lambda_2, \lambda_2, \cdots \lambda_r, \lambda_r)$$

Comparing (3.14) and (3.22), we observe

$$\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n = -2\mathbf{U}_n^T\mathbf{N}^T\mathbf{L}\mathbf{N}\mathbf{U}_n \tag{3.23}$$

The left hand side of the above equation consists of all known quantities. In the following section we will show that the unknown node positions can be determined by multiplying the left hand side of (3.23) by a known matrix.

To obtain the unknown node positions we can partition $\mathbf{N}$ into an anchor node matrix $\mathbf{N}_a$ and an unknown node matrix $\mathbf{N}_u$ as

$$\mathbf{N} = [\mathbf{N}_a \ , \ \mathbf{N}_u] \tag{3.24}$$

where $\mathbf{N}_a \in \Re^{2r\times k}$ and $\mathbf{N}_u \in \Re^{2r\times(M-k)}$.
Since the anchor node positions are known *a priori*, $\mathbf{N}_a$ is a known matrix. We wish to determine the unknown node matrix $\mathbf{N}_u$ and thereby $\mathbf{\Phi}_u$.
To determine $\mathbf{N}_u$, the noise matrix $\mathbf{U}_n$ can be partitioned into two submatrices $\mathbf{U}_{na} \in \Re^{k\times(M-r)}$ and $\mathbf{U}_{nb} \in \Re^{(M-k)\times(M-r)}$ as

$$\mathbf{U}_n = \begin{pmatrix}\mathbf{U}_{na}\\ \mathbf{U}_{nb}\end{pmatrix} \tag{3.25}$$

Then we can write

$$\mathbf{N}\mathbf{U}_n = \mathbf{N}_a\mathbf{U}_{na} + \mathbf{N}_u\mathbf{U}_{nb} \tag{3.26}$$

Let us now define

$$\hat{\mathbf{N}}_u = [\mathbf{I}_{2r} \ , \ \mathbf{N}_u] \tag{3.27}$$

where $\mathbf{I}_{2r}$ is the $2r \times 2r$ identity matrix, and.

$$\hat{\mathbf{N}}_a = \begin{pmatrix}\mathbf{N}_a\mathbf{U}_{na}\\ \mathbf{U}_{nb}\end{pmatrix} \tag{3.28}$$

Then

$$\hat{\mathbf{N}}_u\hat{\mathbf{N}}_a = \begin{bmatrix}\mathbf{I}_{2r} \ , \ \mathbf{N}_u\end{bmatrix}\begin{bmatrix}\mathbf{N}_a\mathbf{U}_{na}\\ \mathbf{U}_{nb}\end{bmatrix}$$

$$= \mathbf{N}_a\mathbf{U}_{na} + \mathbf{N}_u\mathbf{U}_{nb} \tag{3.29}$$

Comparing the right hand side of (3.26) with (3.29), we get

$$\mathbf{N}\mathbf{U}_n = \hat{\mathbf{N}}_u \hat{\mathbf{N}}_a \tag{3.30}$$

The dimensions of $\hat{\mathbf{N}}_u$ and $\hat{\mathbf{N}}_a$ are $2r \times (M-k+2r)$ and $(M-k+2r) \times (M-r)$ respectively.
From (3.30), we get

$$\hat{\mathbf{N}}_u = \mathbf{N}\mathbf{U}_n \hat{\mathbf{N}}_a^{\dagger} \tag{3.31}$$

where

$$\hat{\mathbf{N}}_a^{\dagger} = \hat{\mathbf{N}}_a^T (\hat{\mathbf{N}}_a \hat{\mathbf{N}}_a^T)^{-1} \tag{3.32}$$

If $\hat{\mathbf{N}}_a$ has less rows than columns then there will be more constraining equations than free variables and a solution is generally not possible. Hence we need

$$M - r \geq 2r + (M-k)$$

or

$$k \geq 3r \tag{3.33}$$

The above equation provides us a condition for the minimum number of anchor nodes required to make $\hat{\mathbf{N}}_a$ a wide matrix. The minimum number of anchor nodes required must be greater than or equal to three times the rank of the weight matrix $\mathbf{W}$. If node i and j are unable to make a distance measurement, the corresponding entries w_{ij} and w_{ji} in the weight matrix will be zero. Hence for each missing link we have two zeros in the weight matrix, and the rank increases rapidly requiring more and more anchor nodes.

The $\hat{\mathbf{N}}_u$ given in (3.31) can not be used to compute $\mathbf{N}_u$ (and hence $\mathbf{\Phi}_u$) as it depends upon $\mathbf{N}$ which contains the unknown positions. To determine $\mathbf{N}_u$ we define $\acute{\mathbf{N}}_u \in \Re^{(2r+M-k)\times(M-k)}$ as

$$\acute{\mathbf{N}}_u = \begin{bmatrix} \mathbf{N}_u \\ -\mathbf{I}_{M-k} \end{bmatrix}$$

Then

$$\hat{\mathbf{N}}_u \acute{\mathbf{N}}_u = \begin{bmatrix} \mathbf{I}_{2r} & \mathbf{N}_u \end{bmatrix} \begin{bmatrix} \mathbf{N}_u \\ -\mathbf{I}_{M-k} \end{bmatrix}$$

$$= \mathbf{0}_{2r\times(M-k)} \tag{3.34}$$

Substituting $\hat{\mathbf{N}}_u$ from (3.31), we obtain

$$\mathbf{N}\mathbf{U}_n \hat{\mathbf{N}}_a^{\dagger} \acute{\mathbf{N}}_u = \mathbf{0}_{2r\times(M-k)} \tag{3.35}$$

But from (3.22), we have

$$\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n = -2\mathbf{U}_n^T\mathbf{N}^T\mathbf{L}\mathbf{N}\mathbf{U}_n$$

Post multiplying the above equation with $\hat{\mathbf{N}_a}^{\dagger}\acute{\mathbf{N}}_u$, we get

$$\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n\hat{\mathbf{N}}_a^{\dagger}\acute{\mathbf{N}}_u = -2\mathbf{U}_n^T\mathbf{N}^T\mathbf{L}\mathbf{N}\mathbf{U}_n\hat{\mathbf{N}}_a^{\dagger}\acute{\mathbf{N}}_u \tag{3.36}$$

But from (3.35)

$$\mathbf{N}\mathbf{U}_n\hat{\mathbf{N}}_a^{\dagger}\acute{\mathbf{N}}_u = \mathbf{0}$$

and thus

$$\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n\hat{\mathbf{N}}_a^{\dagger}\acute{\mathbf{N}}_u = \mathbf{0}_{(M-r)\times(M-k)} \tag{3.37}$$

From the above equation we obtain $\acute{\mathbf{N}}_u$ which has $2r + (M - k)$ rows and $M - k$ columns. The first $2r$ rows define the matrix $\mathbf{N}_u$.

Let us define

$$\mathbf{E} = \begin{bmatrix} \mathbf{I}_{2r} \\ \mathbf{0}_{M-k\times 2r} \end{bmatrix} \tag{3.38}$$

Then

$$\mathbf{N}_u = \mathbf{E}^T\acute{\mathbf{N}}_u \tag{3.39}$$

We can obtain the unknown node positions by taking the first two rows from $\mathbf{N}_u$ and by multiplying them with the inverse of $\text{diag}(\mathbf{u}_{1u})$.

$$\mathbf{\Phi}_u = \mathbf{N}_u\text{diag}(\mathbf{u}_{1u})^{-1} \tag{3.40}$$

where $\mathbf{u}_{1u}$ is an $(M - k) \times 1$ vector obtained by selecting the last $(M - k)$ elements from $\mathbf{u}_1$

Assuming that noise free range measurements are available between all pairs of nodes, eq (3.37) can be written as

$$\mathbf{U}_n^T\mathbf{R}\mathbf{U}_n\hat{\mathbf{N}}_a^{\dagger}\acute{\mathbf{N}}_u = \mathbf{0}_{(M-r)\times(M-k)} \tag{3.41}$$

thus the unknown node positions can be determined with the help of squared distance matrix $\mathbf{R}$ and a few known locations. The true and estimated node locations for a fully connected network of 100 nodes (using three of them as beacons) are shown in Figure (3.1)

However when some of the distance estimates are missing, i.e. the network is partially connected, the proposed method fails to find the unknown locations due to rank deficiency of $\hat{\mathbf{N}}_a$.

3.2 Conclusion

We proposed a least squares method based on the noise subspace projection of the weight matrix to find the unknown node locations for a wireless sensor network. The proposed method is able to determine the unknown node locations for a fully connected network, however for partially connected networks it fails due to rank deficiency of $\hat{\mathbf{N}}_a$. In the next chapter we shall describe the reasons for its failure for partial networks.

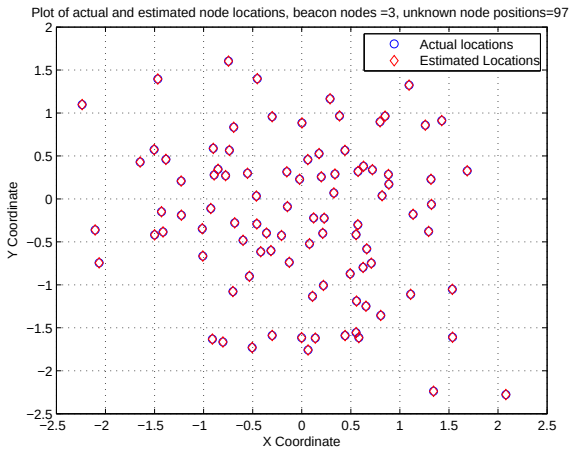


Figure 3.1: Actual and estimated node locations for a fully connected network

Shortcomings of the proposed method

4

The energy and bandwidth constraints pose a severe restriction on the computational and communication capabilities of a sensor node. The unfriendly environment in which sensor nodes are deployed also limits their communication range. Unless the network consists of a small number of nodes and the field conditions are perfect, it is highly unlikely for a node to get a noise free distance measurement from every other node. Multidimensional scaling requires a complete set of dissimilarity information for a closed-form solution. When the complete set of dissimilarity information is not available, the variants of MDS seek the required coordinates in an iterative manner [11], [19]. However these techniques are iterative in nature and may not be able to find a global optimum. We proposed a method to find a closed-form analytical solution to determine the node coordinates. In this chapter, we will determine the necessary and sufficient condition for the proposed method to have a solution with missing dissimilarity information.

4.1 Conditions for the existence of solution

In the previous chapter, we derived a least squares method to find the unknown node positions assuming that the noise free range measurements are available between some pairs of nodes. By projecting $\mathbf{R} \odot \mathbf{W}$ on to the noise subspace of the weight matrix and then multiplying with the pseudo inverse of a known matrix ($\hat{\mathbf{N}}_a$), we obtain the unknown node positions as given in (3.37)

$$\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n\hat{\mathbf{N}}_a^{\dagger}\acute{\mathbf{N}}_u = \mathbf{0}_{(M-r)X(M-k)}$$

where $\hat{\mathbf{N}}_a$ is defined as in (3.28)

$$\hat{\mathbf{N}}_a = \begin{pmatrix} \mathbf{N}_a\mathbf{U}_{na} \\ \mathbf{U}_{nb} \end{pmatrix}$$

$\hat{\mathbf{N}}_a$ must be a wide and full row rank matrix, as we need to compute its pseudo inverse which is then multiplied with $\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n$ to obtain the unknown node positions. To make $\hat{\mathbf{N}}_a$ a wide matrix, we already determined that the number of anchor nodes must be greater than or equal to three times the rank of the weight matrix $\mathbf{W}$. Now we determine the conditions for $\hat{\mathbf{N}}_a$ to be a full row rank matrix.

A necessary condition for $\hat{\mathbf{N}}_a$ to have full row rank is that the matrix $\mathbf{N}_a$ $(2r \times k)$ should have $2r$ independent rows (it must be full row rank). The matrix $\mathbf{N}_a$ is obtained by selecting the first k columns of $\mathbf{N}$ which in turn implies that $\mathbf{N}$ must have full row rank (i.e. $2r$).

The matrix $\mathbf{N}$ is defined as

$$\mathbf{N} = \begin{bmatrix} \mathbf{\Phi}\text{diag}(\mathbf{u}_1) \\ \mathbf{\Phi}\text{diag}(\mathbf{u}_2) \\ \vdots \\ \mathbf{\Phi}\text{diag}(\mathbf{u}_r) \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} \boldsymbol{\phi}_1 u_{11} & \boldsymbol{\phi}_2 u_{12} & \cdots & \boldsymbol{\phi}_M u_{1M} \\ \boldsymbol{\phi}_1 u_{21} & \boldsymbol{\phi}_2 u_{22} & \cdots & \boldsymbol{\phi}_M u_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_1 u_{r1} & \boldsymbol{\phi}_2 u_{r2} & \cdots & \boldsymbol{\phi}_M u_{rM} \end{bmatrix}$$

where u_{ij} is the $j-th$ component of the $i-th$ eigenvector, for $1 \leq i \leq r$ and $1 \leq j \leq M$. Since $\boldsymbol{\phi}_i = [x_i, y_i]^T$, $\mathbf{N}$ can also be written as

$$\mathbf{N} = \begin{bmatrix} x_1 u_{11} & x_2 u_{12} & \cdots & x_M u_{1M} \\ y_1 u_{11} & y_2 u_{12} & \cdots & y_M u_{1M} \\ x_1 u_{21} & x_2 u_{22} & \cdots & x_M u_{2M} \\ y_1 u_{21} & y_2 u_{22} & \cdots & y_M u_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_1 u_{r1} & x_2 u_{r2} & \cdots & x_M u_{rM} \\ y_1 u_{r1} & y_2 u_{r2} & \cdots & y_M u_{rM} \end{bmatrix}$$

Let $\mathbf{A}$ be a $2r \times M$ matrix obtained by stacking the r eigenvectors of $\mathbf{W}$, i.e. ,

$$\mathbf{A} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_r^T \\ \mathbf{u}_r^T \end{bmatrix} \tag{4.1}$$

$$= \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1M} \\ u_{11} & u_{12} & \cdots & u_{1M} \\ u_{21} & u_{22} & \cdots & u_{2M} \\ u_{21} & u_{22} & \cdots & u_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ u_{r1} & u_{r2} & \cdots & u_{rM} \\ u_{r1} & u_{r2} & \cdots & u_{rM} \end{bmatrix}$$

The rank of $\mathbf{A}$ will be r. Simulation results indicate that for a weighting matrix $\mathbf{W}$ corresponding to a partially connected network, many columns of $\mathbf{A}$ are equal. The question whether $\mathbf{N}$ is full row rank or not depends upon the distribution of these equal columns i.e. whether the equal columns of $\mathbf{A}$ fall in one set or are distributed over a number of sets.

Let $\mathbf{M}$ be a $2r \times s$ matrix obtained by selecting s equal columns of $\mathbf{A}$, where s is the largest set consisting of equal columns of $\mathbf{A}$, i.e. ,

$$\mathbf{M} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1s} \\ u_{11} & u_{12} & \cdots & u_{1s} \\ u_{21} & u_{22} & \cdots & u_{2s} \\ u_{21} & u_{22} & \cdots & u_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ u_{r1} & u_{r2} & \cdots & u_{rs} \\ u_{r1} & u_{r2} & \cdots & u_{rs} \end{bmatrix} \tag{4.2}$$

As the columns of $\mathbf{M}$ are equal, i.e. ,

$$\begin{aligned} u_{11} = u_{12} \cdots = u_{1s} \\ u_{21} = u_{22} \cdots = u_{2s} \\ u_{r1} = u_{r2} \cdots = u_{rs} \end{aligned}$$

$\mathbf{M}$ can be written as

$$\mathbf{M} = \begin{bmatrix} u_{11} & u_{11} & \cdots & u_{11} \\ u_{11} & u_{11} & \cdots & u_{11} \\ u_{21} & u_{21} & \cdots & u_{21} \\ u_{21} & u_{21} & \cdots & u_{21} \\ \vdots & \vdots & \ddots & \vdots \\ u_{r1} & u_{r1} & \cdots & u_{r1} \\ u_{r1} & u_{r1} & \cdots & u_{r1} \end{bmatrix}$$

which will be of rank 1 since all its columns are equal.

Now we determine the rank of $\mathbf{S}$ which is a submatrix of $\mathbf{N}$ obtained by taking an elementwise product between $\mathbf{M}$ and the corresponding node positions $\boldsymbol{\phi}_i = [x_i, y_i]^T$, i.e. ,

$$\mathbf{S} = \begin{bmatrix} x_1 & x_2 & \cdots & x_s \\ y_1 & y_2 & \cdots & y_s \\ x_1 & x_2 & \cdots & x_s \\ y_1 & y_2 & \cdots & y_s \\ \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & \cdots & x_s \\ y_1 & y_2 & \cdots & y_s \end{bmatrix} \odot \begin{bmatrix} u_{11} & u_{11} & \cdots & u_{11} \\ u_{11} & u_{11} & \cdots & u_{11} \\ u_{21} & u_{21} & \cdots & u_{21} \\ u_{21} & u_{21} & \cdots & u_{21} \\ \vdots & \vdots & \ddots & \vdots \\ u_{r1} & u_{r1} & \cdots & u_{r1} \\ u_{r1} & u_{r1} & \cdots & u_{r1} \end{bmatrix} = \begin{bmatrix} x_1u_{11} & x_2u_{11} & \cdots & x_su_{11} \\ y_1u_{11} & y_2u_{11} & \cdots & y_su_{11} \\ x_1u_{21} & x_2u_{21} & \cdots & x_su_{21} \\ y_1u_{21} & y_2u_{21} & \cdots & y_su_{21} \\ \vdots & \vdots & \ddots & \vdots \\ x_1u_{r1} & x_2u_{r1} & \cdots & x_su_{r1} \\ y_1u_{r1} & y_2u_{r1} & \cdots & y_su_{r1} \end{bmatrix} \tag{4.3}$$

Multiplying rows 1 and 2 by $(\frac{u_{21}}{u_{11}})$ and subtracting them from row 3 and 4 respectively, we get

$$\mathbf{S} = \begin{bmatrix} x_1u_{11} & x_2u_{11} & \cdots & x_su_{11} \\ y_1u_{11} & y_2u_{11} & \cdots & y_su_{11} \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x_1u_{r1} & x_2u_{r1} & \cdots & x_su_{r1} \\ y_1u_{r1} & y_2u_{r1} & \cdots & y_su_{r1} \end{bmatrix}$$

Similar row operations can be performed to other rows of $\mathbf{S}$, and finally we get

$$\mathbf{S} = \begin{bmatrix} x_1u_{11} & x_2u_{11} & \cdots & x_su_{11} \\ y_1u_{11} & y_2u_{11} & \cdots & y_su_{11} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Multiplying row 1 by $\frac{1}{x_1u_{11}}$ and row 2 by $\frac{1}{y_1u_{11}}$, we obtain

$$\mathbf{S} = \begin{bmatrix} 1 & \frac{x_2}{x_1} & \cdots & \frac{x_s}{x_1} \\ 1 & \frac{y_2}{y_1} & \cdots & \frac{y_s}{y_1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Subtracting row 2 from row 1, we get

$$\mathbf{S} = \begin{bmatrix} 1 & \frac{x_2}{x_1} & \cdots & \frac{x_s}{x_1} \\ 0 & \frac{y_2x_1 - x_2y_1}{y_1x_1} & \cdots & \frac{y_sx_1 - x_sy_1}{x_1y_1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Finally, multiplying row 2 by $\dfrac{y_1x_1}{y_2x_1 - x_2y_1}$, we get $\mathbf{S}$ in reduced echelon form

$$\mathbf{S} = \begin{bmatrix} 1 & \frac{x_2}{x_1} & \cdots & \frac{x_s}{x_1} \\ 0 & 1 & \cdots & \frac{(y_sx_1 - x_sy_1)y_1x_1}{x_1y_1(y_2x_1 - x_2y_1)} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Here the rank of $\mathbf{S}$ will be two as there are only two nonzero rows in its reduced echelon form. We selected $\mathbf{S}$ as a submatrix of $\mathbf{N}$ corresponding to s equal columns of $\mathbf{A}$. We also assumed that s is the largest set of equal columns present in $\mathbf{A}$. In the

worst case, all the equal columns present in **A** will fall in one set. As the rank of **A** is r, it implies that $r-1$ columns of **A** will be independent to this set. Hence in the worst case the rank of **N** will be

$$rank(\mathbf{N}) = rank(\mathbf{S}) + r - 1 \tag{4.4}$$

$$rank(\mathbf{N}) = r + 1$$

N can never be a full rank matrix in the worst case as $r+1$ can not be equal to $2r$ for any value of r (except possibly for $r = 1$).

To illustrate the worst case scenario let us assume a network of five sensor nodes with only one missing link between nodes 1 and 3 as shown in (4.1).

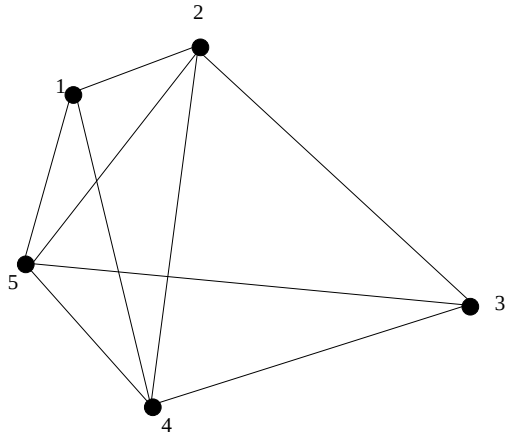


Figure 4.1: Partially connected network

The node coordinate matrix is given as

$$\mathbf{\Phi} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix}$$

The rank 3 weight matrix **W** and the three signal eigenvectors are given by

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{u}_1 = \begin{bmatrix} -0.3943 \\ -0.4792 \\ -0.3943 \\ -0.4792 \\ -0.4792 \end{bmatrix}$$

$$\mathbf{u}_2 = \begin{bmatrix} 0.7071 \\ 0 \\ -0.7071 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{u}_3 = \begin{bmatrix} 0.5869 \\ -0.3230 \\ 0.5869 \\ -0.3230 \\ -0.3230 \end{bmatrix}$$

The $2r \times M$ matrix $\mathbf{N}$ define in (3.21) is given by

$$\mathbf{N} = \begin{bmatrix} \mathbf{\Phi}\text{diag}(u_1) \\ \mathbf{\Phi}\text{diag}(u_2) \\ \mathbf{\Phi}\text{diag}(u_3) \end{bmatrix} = \begin{bmatrix} -0.3943x_1 & -0.4752x_2 & -0.3943x_3 & -0.4792x_4 & -0.4792x_5 \\ -0.3943y_1 & -0.4752y_2 & -0.3943y_3 & -0.4792y_4 & -0.4792y_5 \\ 0.7071x_1 & 0 & -0.7071x_3 & 0 & 0 \\ 0.7071y_1 & 0 & -0.7071y_3 & 0 & 0 \\ -0.5869x_1 & 0.3230x_2 & -0.5869x_3 & 0.3230x_4 & 0.3230x_5 \\ -0.5869y_1 & 0.3230y_2 & -0.5869y_3 & 0.3230y_4 & 0.3230y_5 \end{bmatrix}$$

The matrices $\mathbf{A}$, $\mathbf{M}$, and $\mathbf{S}$ defined in (4.1), (4.2), and (4.3) are thus given as

$$\mathbf{A} = \begin{bmatrix} -0.3943 & -0.4792 & -0.3943 & -0.4792 & -0.4792 \\ -0.3943 & -0.4792 & -0.3943 & -0.4792 & -0.4792 \\ 0.7071 & 0 & -0.7071 & 0 & 0 \\ 0.7071 & 0 & -0.7071 & 0 & 0 \\ -0.5869 & 0.3230 & -0.5869 & 0.3230 & 0.3230 \\ 0.5869 & 0.3230 & -0.5869 & 0.3230 & 0.3230 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} -0.4792 & -0.4792 & -0.4792 \\ -0.4792 & -0.4792 & -0.4792 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.3220 & 0.3220 & 0.3220 \\ 0.3220 & 0.3220 & 0.3220 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} -0.4792x_2 & -0.4792x_4 & -0.4792x_5 \\ -0.4792y_2 & -0.4792y_4 & -0.4792y_5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.3220x_2 & 0.3220x_4 & 0.3220x_5 \\ 0.3220y_2 & 0.3220y_4 & 0.3220y_5 \end{bmatrix}$$

As we have already shown the rank of $\mathbf{S}$ will be two. The addition of a linearly independent column to $\mathbf{S}$ will increase its rank by 1. Since column 1 and 3 of $\mathbf{N}$ are linearly independent to the columns of $\mathbf{S}$, hence the rank of $\mathbf{N}$ will be

$$rank(\mathbf{N}) = rank(\mathbf{S}) + 2 = 4$$

The above example depicts the worst case scenario in which all the equal columns of $\mathbf{A}$ fall in one set.

Now we assume that node 1 also misses a distance estimate from node 5. Then the weight matrix $\mathbf{W}$ and the corresponding signal eigenvectors are given as

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{u}_1 = \begin{bmatrix} -0.3069 \\ -0.5100 \\ -0.4390 \\ -0.5100 \\ -0.4390 \end{bmatrix}$$

$$\mathbf{u}_2 = \begin{bmatrix} 0.7702 \\ 0.1378 \\ -0.4294 \\ 0.1378 \\ -0.4294 \end{bmatrix}$$

$$\mathbf{u}_3 = \begin{bmatrix} -0.5590 \\ 0.4700 \\ -0.3505 \\ 0.4700 \\ -0.3505 \end{bmatrix}$$

The matrix $\mathbf{A}$ will now be given as

$$\mathbf{A} = \begin{bmatrix} -0.3069 & -0.5100 & -0.4390 & -0.5100 & -0.4390 \\ -0.3069 & -0.5100 & -0.4390 & -0.5100 & -0.4390 \\ 0.7702 & 0.1378 & -0.4294 & 0.1378 & -0.4294 \\ 0.7702 & 0.1378 & -0.4294 & 0.1378 & -0.4294 \\ -0.5590 & 0.4700 & -0.3505 & 0.4700 & -0.3505 \\ -0.5590 & 0.4700 & -0.3505 & 0.4700 & -0.3505 \end{bmatrix}$$

$\mathbf{A}$ now contains two set of equal columns, i.e. columns 2 and 4, and columns 3 and 5. Each set of equal columns when multiplied with the corresponding node positions will produce a rank 2 matrix. The column 1 of $\mathbf{A}$ is linearly independent to both of these sets. Hence the rank of $\mathbf{N}$ will now be

$$rank(\mathbf{N}) = rank(\mathbf{S}_1) + rank(\mathbf{S}_2) + 1 = 5$$

Generally for $\mathbf{N}$ to have full rank (i.e. $2r$), we require that $M - r$ equal columns of $\mathbf{A}$ (and hence $\mathbf{W}$) are distributed over r submatrices. Then each submatrix containing two or more equal columns of $\mathbf{A}$ will be a rank two matrix (as shown above), and only then the rank of $\mathbf{N}$ can be $2r$.

The beacon node matrix $\mathbf{N}_a$ consists of the first k columns of $\mathbf{N}$. Thus for $\mathbf{N}_a$ to have full row rank, we require that the first k columns must constitute r submatrices in which the equal columns of $\mathbf{W}$ are distributed.

The full rank condition of $\mathbf{N}_a$ is only a necessary condition for $\hat{\mathbf{N}}_a$ to have full rank. However it is not sufficient and $\hat{\mathbf{N}}_a$ can still be rank deficient.

We now determine the sufficient condition for $\hat{\mathbf{N}}_a$ to be a full rank matrix. $\hat{\mathbf{N}}_a$ is defined in (3.28) as

$$\hat{\mathbf{N}}_a = \begin{bmatrix} \mathbf{N}_a\mathbf{U}_{na} \\ \mathbf{U}_{nb} \end{bmatrix}$$

where $\mathbf{N}_a\mathbf{U}_{na}$ is given as

$$\mathbf{N}_a\mathbf{U}_{na} = \begin{bmatrix} x_1u_{11} & x_2u_{12} & \cdots & x_ku_{1k} \\ y_1u_{11} & y_2u_{12} & \cdots & y_ku_{1k} \\ x_1u_{21} & x_2u_{22} & \cdots & x_ku_{2k} \\ y_1u_{21} & y_2u_{22} & \cdots & y_ku_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_1u_{r1} & x_2u_{r2} & \cdots & x_ku_{rk} \\ y_1u_{r1} & y_2u_{r2} & \cdots & y_ku_{rk} \end{bmatrix} \begin{bmatrix} u_{1,r+1} & u_{1,r+2} & \cdots & u_{1,M} \\ u_{2,r+1} & u_{2,r+2} & \cdots & u_{2,M} \\ u_{3,r+1} & u_{3,r+2} & \cdots & u_{3,M} \\ u_{4,r+1} & u_{4,r+2} & \cdots & u_{4,M} \\ \vdots & \vdots & \ddots & \vdots \\ u_{k-1,r+1} & u_{k-1,r+2} & \cdots & u_{k-1,M} \\ u_{k,r+1} & u_{k,r+2} & \cdots & u_{k,M} \end{bmatrix}$$

$\mathbf{U}_n$ is an $M \times (M-r)$ noise matrix with full column rank (i.e. $M-r$). The matrix $\mathbf{U}_{na}$ is obtained by selecting the first k rows of $\mathbf{U}_n$ whereas $\mathbf{U}_{nb}$ consists of the last $M-k$ rows of $\mathbf{U}_n$. $\mathbf{N}_a\mathbf{U}_{na}$ will be full row rank only when $\mathbf{U}_{na}$ has k independent rows. Simulation results indicate that when the weight matrix is partially connected, the noise subspace matrix $\mathbf{U}_n$ contains a number of zero columns for leading rows. As $\mathbf{U}_{na}$ is obtained by selecting the first k rows, it will contain a number of zero columns, which make $\mathbf{N}_a\mathbf{U}_{na}$ rank deficient even if $\mathbf{N}_a$ is of full row rank.

Power constraints and physical obstacles make it infeasible for each sensor node to get an estimate from every other node. Thus we often end up with a partially connected network in which a sensor node is able to make an estimate only with its nearby nodes. The proposed method however, fails for partially connected networks due to rank deficiency of $\hat{\mathbf{N}}_a$.

Now we investigate the effects of some elementary operations on the weight matrix in the hope to get rid of this rank deficiency problem.

4.2 Operations to reduce the rank of weight matrix

For a fully connected network, every node makes a distance estimate with every other node in the network and the weight matrix $\mathbf{W}$ will be a rank 1 matrix. However when two nodes are unable to make an estimate between them, we need to place two zeros in the weight matrix for each missing link. Thus the rank of the weight matrix increases rapidly. The increase in rank of $\mathbf{W}$ is highly undesirable as it increases the number of anchor nodes.

Let us start with a simple example. Assume that there are 9 sensor nodes deployed randomly in a certain field. We further assume that nodes 1 and 6 are unable to make an estimate. Thus the weight matrix $\mathbf{W}$ will look like

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.5}$$

In this case the rank of $\mathbf{W}$ will be 3 and we require that the rank of $\mathbf{N}$ must be 6. However, the eigenvectors of $\mathbf{W}$ corresponding to nonzero eigenvalues are such that all the equal columns of $\mathbf{W}$ fall in one set. Hence in this case the rank of $\mathbf{N}$ will be

$$rank(\mathbf{N}) = r + 1 = 4$$

Although we just miss one distance estimate, and there still is a lot of redundancy, the proposed method will not be able to find the unknown node positions as $\mathbf{N}$ is rank deficient in this case.

The squared distance matrix $\mathbf{R}$ between the nodes is given as

$$\mathbf{R} = \begin{bmatrix} 0 & d_{1,2}^2 & d_{1,3}^2 & \cdots & d_{1,9}^2 \\ d_{2,1}^2 & 0 & d_{2,3}^2 & \cdots & d_{2,9}^2 \\ d_{3,1}^2 & d_{3,2}^2 & 0 & \cdots & d_{3,9}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{9,1}^2 & d_{9,2}^2 & d_{9,3}^2 & \cdots & 0 \end{bmatrix}$$

Since the diagonal entries of $\mathbf{R}$ are zero, and we take an elementwise product between $\mathbf{R}$ and $\mathbf{W}$, the diagonal entries of $\mathbf{W}$ can be modified without any effect on the resultant product. The modification in the diagonal entries is aimed at reducing the rank of $\mathbf{W}$.

Now let us change the above $\mathbf{W}$ as

$$\mathbf{W} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The first column ($\mathbf{c}_1$) of $\mathbf{W}$ can now be written as a linear combination of the second and sixth columns as

$$\mathbf{c}_1 = 2\mathbf{c}_2 - \mathbf{c}_6$$

Thus the rank of the weight matrix $\mathbf{W}$ has been reduced to two. Now if some other nodes do not make distance estimates, we can perform similar operations to $\mathbf{W}$ i.e. we add a 1 to the diagonal entry of each row having a zero. For example if nodes (2,3) and (7,9) do not make a measurement between them, performing similar operations on $\mathbf{W}$ yields

$$\mathbf{W} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 2 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 2 \end{bmatrix}$$

Another possible operation to reduce the rank of $\mathbf{W}$ is to decrement the diagonal entry of each row having a missing link. For instance for a missing link between nodes 1 and 6, the weight matrix $\mathbf{W}$ defined in (4.5) can be changed to

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

To summarize, for each missing link in any row we increment (or decrement) the diagonal entry in that row by 1. By doing so the rank of $\mathbf{W}$ is increased by at most one while earlier it was being increased by at most 2 for each missing link. By such techniques we can reduce the rank of $\mathbf{W}$, thereby requiring fewer anchor nodes.

4.3 Conclusion

In this chapter we described the reasons for failure of the proposed method in case of partially connected networks. The rank deficiency of the anchor node matrix $\mathbf{N}_a$ and the sparsity of the noise subspace matrices make the method inapplicable for partial networks. We have also introduced two rank reduction techniques to minimize the requirement of anchor nodes and to alleviate the rank deficiency of $\hat{\mathbf{N}}_a$. We shall show in the next chapter that the two rank reduction techniques though look similar, produce different node projections. We shall also determine whether the rank reduction techniques indeed solve the rank deficiency of $\hat{\mathbf{N}}_a$.

The proposed algorithm with rank reduction techniques

5

For a fully connected sensor network, classical MDS provides a closed-form solution to determine the unknown node positions. We wish to find a similar solution for a partially connected network by projecting the Hadamard product of the squared distance matrix $\mathbf{R}$ and weight matrix $\mathbf{W}$, onto the noise subspace of $\mathbf{W}$. In the previous chapter we described the inability of the proposed method for partially connected networks due to rank deficiency of $\hat{\mathbf{N}}_a$. We have also introduced two techniques to reduce the rank of weight matrix $\mathbf{W}$. When the rank of weight matrix is reduced, intuitively we expect to solve the rank deficiency of $\hat{\mathbf{N}}_a$ (rank of $\hat{\mathbf{N}}_a$ must be $2r + M - k$)). The important questions which we will consider in this chapter are: Whether the localization problem with missing dissimilarity information is solvable? and if yes whether the proposed method is able to determine the global node positions? To answer the first question, we will take the node projections on to the noise subspace of different weight matrices and we will determine the conditions for a possible solution. To answer the secound question we will take some examples to demonstrate whether the rank reduction techniques indeed assist us to find a possible solution.

Let us assume that $M = 5$ sensor nodes are randomly placed in a certain field and every sensor node gets a distance estimate from every other node as shown below. Let the position of the $i-th$ node be $\boldsymbol{\phi}_i = [x_i, y_i]^T$. Thus the coordinate matrix $\boldsymbol{\Phi}$ is given

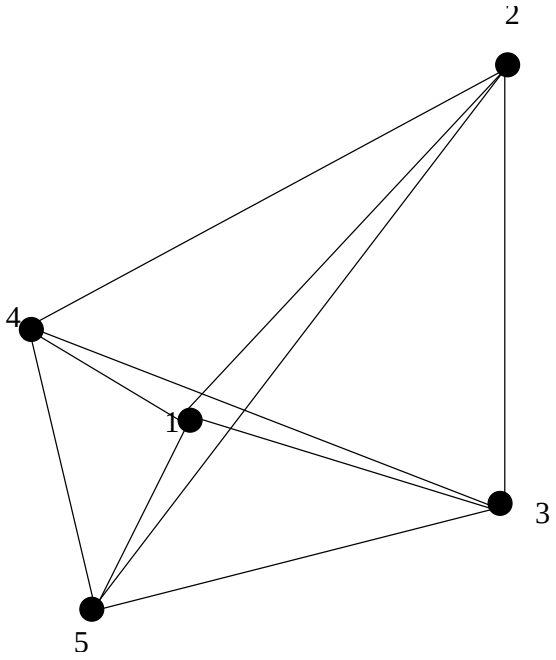


Figure 5.1: Fully connected network of five nodes

as

$$\mathbf{\Phi} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix}$$

Since the network is fully connected, the weight matrix $\mathbf{W}$ is given as

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The single signal eigenvector of $\mathbf{W}$ will be

$$\mathbf{u}_1 = \begin{bmatrix} \frac{1}{\sqrt{M}} \\ \frac{1}{\sqrt{M}} \\ \frac{1}{\sqrt{M}} \\ \frac{1}{\sqrt{M}} \\ \frac{1}{\sqrt{M}} \end{bmatrix}$$

and as a result

$$\mathbf{u}_1\mathbf{u}_1^T = \begin{bmatrix} \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \end{bmatrix}$$

Then the projection matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \mathbf{I} - \mathbf{u}_1\mathbf{u}_1^T$$

$$\mathbf{P} = \begin{bmatrix} 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} \end{bmatrix} \tag{5.1}$$

Taking a left projection between the node coordinate matrix $\mathbf{\Phi}^T$ and the projection matrix $\mathbf{P}$, we obtain

$$\mathbf{P}\mathbf{\Phi}^T = \begin{bmatrix} 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \\ x_5 & y_5 \end{bmatrix}$$

$$= \begin{bmatrix} (1-\frac{1}{M})x_1 - \frac{1}{M}(x_2+x_3+x_4+x_5) & (1-\frac{1}{M})y_1 - \frac{1}{M}(y_2+y_3+y_4+y_5) \\ (1-\frac{1}{M})x_2 - \frac{1}{M}(x_1+x_3+x_4+x_5) & (1-\frac{1}{M})y_2 - \frac{1}{M}(y_1+y_3+y_4+y_5) \\ (1-\frac{1}{M})x_3 - \frac{1}{M}(x_2+x_1+x_4+x_5) & (1-\frac{1}{M})y_3 - \frac{1}{M}(y_2+y_1+y_4+y_5) \\ (1-\frac{1}{M})x_4 - \frac{1}{M}(x_2+x_3+x_1+x_5) & (1-\frac{1}{M})y_4 - \frac{1}{M}(y_2+y_3+y_1+y_5) \\ (1-\frac{1}{M})x_5 - \frac{1}{M}(x_2+x_3+x_4+x_1) & (1-\frac{1}{M})y_5 - \frac{1}{M}(y_2+y_3+y_4+y_1) \end{bmatrix}$$

Thus by taking a projection between the node coordinate matrix $\mathbf{\Phi}^T$ and $\mathbf{P}$, the node positions can be determined with a possible translation as shown in (5.2). To determine the unknown node positions with the help of square distance matrix $\mathbf{R}$ and a few known locations, we define $\mathbf{N}$ for the above case as

$$\mathbf{N} = \mathbf{\Phi}\text{diag}(\mathbf{u}_1)$$

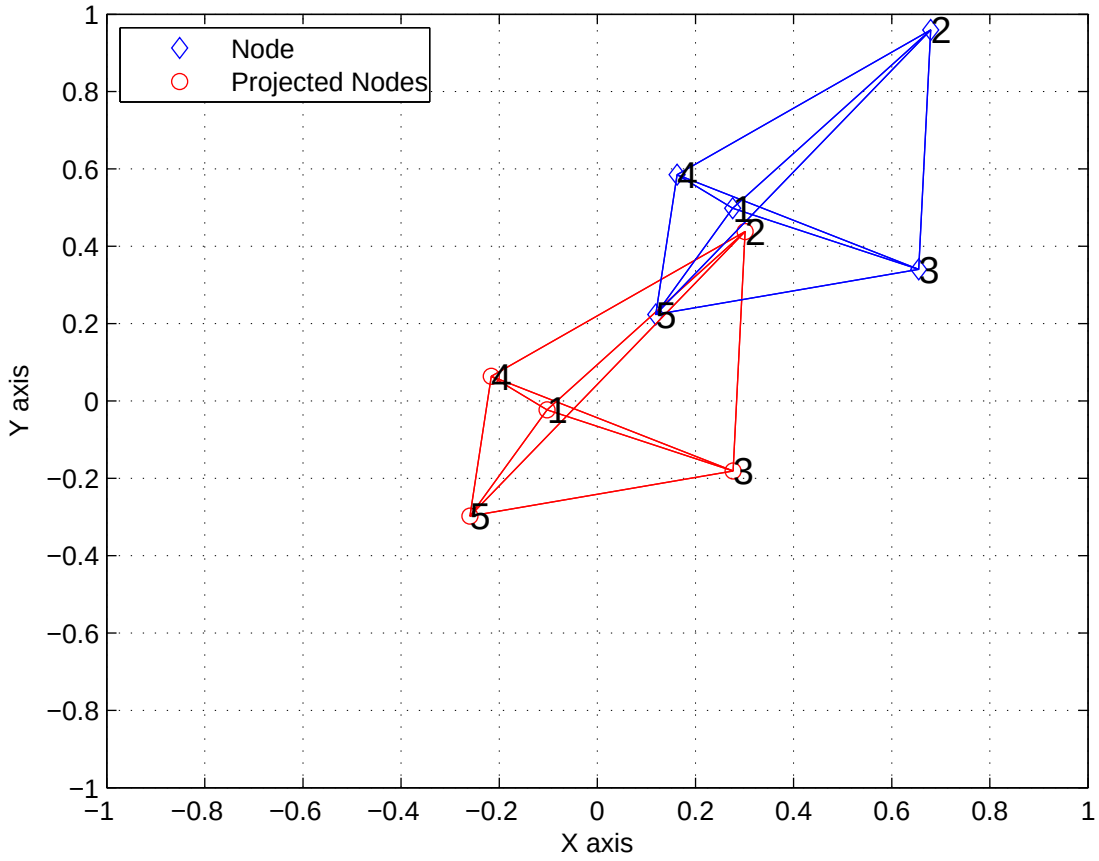


Figure 5.2: Original and projected node positions for a fully connected network

$$\mathbf{N} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix} \begin{bmatrix} \frac{-1}{\sqrt{M}} & 0 & 0 & \cdots & 0 \\ 0 & \frac{-1}{\sqrt{M}} & 0 & \cdots & 0 \\ 0 & 0 & \frac{-1}{\sqrt{M}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{-1}{\sqrt{M}} \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} \frac{-1}{\sqrt{M}}x_1 & \frac{-1}{\sqrt{M}}x_2 & \frac{-1}{\sqrt{M}}x_3 & \cdots & \frac{-1}{\sqrt{M}}x_5 \\ \frac{-1}{\sqrt{M}}y_1 & \frac{-1}{\sqrt{M}}y_2 & \frac{-1}{\sqrt{M}}y_3 & \cdots & \frac{-1}{\sqrt{M}}y_5 \end{bmatrix}$$

The proposed method requires that the number of beacon nodes k must be at least three times the rank of the weight matrix $\mathbf{W}$. Since the rank of $\mathbf{W}$ is one, we assume that the first three sensor nodes are acting as beacons. Hence the beacon node matrix $\mathbf{N}_a$ is given by

$$\mathbf{N}_a = \begin{pmatrix} \frac{-1}{\sqrt{M}}x_1 & \frac{-1}{\sqrt{M}}x_2 & \frac{-1}{\sqrt{M}}x_3 \\ \frac{-1}{\sqrt{M}}y_1 & \frac{-1}{\sqrt{M}}y_2 & \frac{-1}{\sqrt{M}}y_3 \end{pmatrix}$$

The beacon node matrix $\mathbf{N}_a$ will be full rank as the nodes are deployed randomly. To determine the unknown node positions with the help of distance information between node pairs and the beacon nodes, we define a matrix $\hat{\mathbf{N}}_a$ as

$$\hat{\mathbf{N}}_a = \begin{pmatrix} \mathbf{N}_a\mathbf{U}_{na} \\ \mathbf{U}_{nb} \end{pmatrix} \tag{5.2}$$

The node positions are then determined by multiplying the pseudo inverse of $\hat{\mathbf{N}}_a$ with $\mathbf{U}_n^T(\mathbf{R} \odot \mathbf{W})\mathbf{U}_n$. Thus we require that $\hat{\mathbf{N}}_a$ must have full rank. For a fully connected network the condition is generally satisfied and hence the unknown node positions can be determined by the proposed method.

For a fully connected network of five sensor nodes where three of them are acting as beacons, the unknown node positions determined by the proposed method are shown in Figure(5.3)

It is quite appealing to compare our algorithm with the classical MDS solution at this point. The MDS multiplies the squared distance matrix $\mathbf{R}$ with a centering matrix $\mathbf{J}$ on both sides to obtain an inner product matrix $\mathbf{B}$, i.e.

$$\mathbf{B} = -0.5\mathbf{JRJ} = \mathbf{J}\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{J} \tag{5.3}$$

where $\mathbf{J}$ is given by

$$\mathbf{J} = \mathbf{I}_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^T$$

It then performs an eigendecomposition of the inner product matrix $\mathbf{B}$ to obtain relative node positions. The relative node positions are subject to a possible translation, rotation, and reflection and are mapped to absolute positions using a few anchor nodes by a procedure called Procrestus analysis [18]. Procrestus analysis translates, rotates and reflects a given configuration of points to match as closely as possible, to the other.

For a fully connected network of M sensor nodes, the projection matrix $\mathbf{P}$ defined in (5.1) is same as the centering matrix $\mathbf{J}$ used by the MDS. Our method however, is able to find the absolute node positions in a single step as shown in Figure (5.3) and unlike MDS no final alignment step is required.

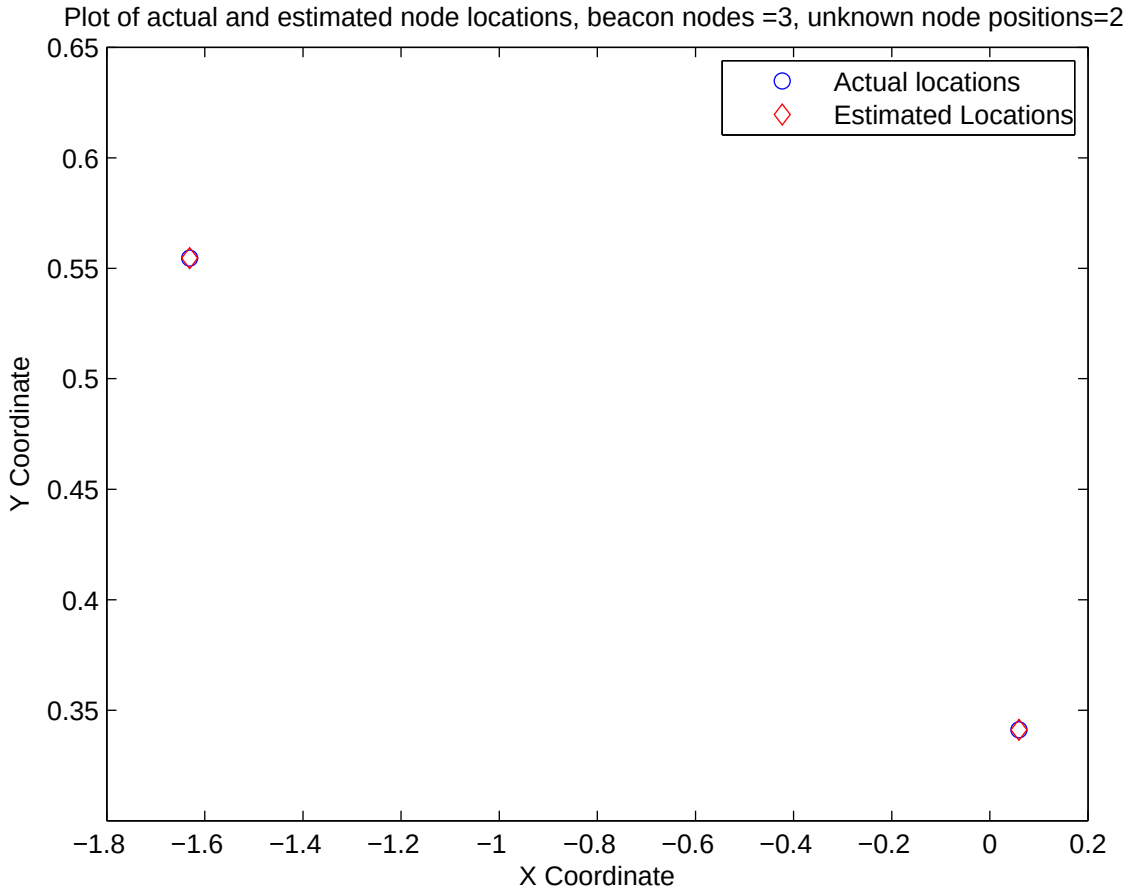


Figure 5.3: Actual and estimated locations

5.1 Performance of the algorithm with the first rank reduction technique

Now we assume that node 3 does not get an estimate from node 5. Then the weight matrix $\mathbf{W}$ will be

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 \end{bmatrix}$$

The signal subspace matrix $\mathbf{U}_s$ will now be equal to

$$\mathbf{U}_s = \begin{bmatrix} \frac{1}{\sqrt{M}} & 0 \\ \frac{1}{\sqrt{M}} & 0 \\ \frac{1}{\sqrt{M}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{M}} & 0 \\ \frac{1}{\sqrt{M}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

and we obtain

$$\mathbf{U}_s\mathbf{U}_s^T = \begin{bmatrix} \frac{1}{\sqrt{M}} & 0 \\ \frac{1}{\sqrt{M}} & 0 \\ \frac{1}{\sqrt{M}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{M}} & 0 \\ \frac{1}{\sqrt{M}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{M}} & \frac{1}{\sqrt{M}} & \frac{1}{\sqrt{M}} & \frac{1}{\sqrt{M}} & \frac{1}{\sqrt{M}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} + \frac{1}{2} & \frac{1}{M} & \frac{1}{M} - \frac{1}{2} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} & \frac{1}{M} \\ \frac{1}{M} & \frac{1}{M} & \frac{1}{M} - \frac{1}{2} & \frac{1}{M} & \frac{1}{M} + \frac{1}{2} \end{bmatrix}$$

Thus, the projection matrix $\mathbf{P}$ will now be

$$\mathbf{P} = \begin{bmatrix} 1 - \frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & 1 - \frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & \frac{1}{2} - \frac{1}{M} & -\frac{1}{M} & \frac{1}{2} - \frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & 1 - \frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & \frac{1}{2} - \frac{1}{M} & -\frac{1}{M} & \frac{1}{2} - \frac{1}{M} \end{bmatrix}$$

and

$$\mathbf{P}\mathbf{\Phi}^T = \begin{bmatrix} 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & \frac{1}{2}-\frac{1}{M} & -\frac{1}{M} & \frac{1}{2}-\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & -\frac{1}{M} & 1-\frac{1}{M} & -\frac{1}{M} \\ -\frac{1}{M} & -\frac{1}{M} & \frac{1}{2}-\frac{1}{M} & -\frac{1}{M} & \frac{1}{2}-\frac{1}{M} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \\ x_5 & y_5 \end{bmatrix}$$

$$= \begin{bmatrix} (1-\frac{1}{M})x_1 - \frac{1}{M}(x_2+x_3+x_4+x_5) & (1-\frac{1}{M})y_1 - \frac{1}{M}(y_2+y_3+y_4+y_5) \\ (1-\frac{1}{M})x_2 - \frac{1}{M}(x_1+x_3+x_4+x_5) & (1-\frac{1}{M})y_2 - \frac{1}{M}(y_1+y_3+y_4+y_5) \\ (\frac{1}{2}-\frac{1}{M})(x_3+x_5) - \frac{1}{M}(x_2+x_1+x_4) & (\frac{1}{2}-\frac{1}{M})(y_3+y_5) - \frac{1}{M}(y_2+y_1+y_4) \\ (1-\frac{1}{M})x_4 - \frac{1}{M}(x_2+x_3+x_1+x_5) & (1-\frac{1}{M})y_4 - \frac{1}{M}(y_2+y_3+y_1+y_5) \\ (\frac{1}{2}-\frac{1}{M})(x_3+x_5) - \frac{1}{M}(x_2+x_4+x_1) & (\frac{1}{2}-\frac{1}{M})(y_3+y_5) - \frac{1}{M}(y_2+y_4+y_1) \end{bmatrix}$$

The original and projected node positions are shown in Figure (5.4). The fully connected nodes (i.e. nodes 1, 2, and 4) are mapped to a point centered at origin while keeping the inter node distances intact, while the nodes with a missing link (nodes 3 and 5) are mapped to a point which is sum of their original positions besides a translation. It is impossible to determine the exact positions of node 3 and node 5 using this projection.

To investigate whether the proposed method will be able to determine the unknown node positions for a partially connected network, we now assume that 9 nodes are deployed in a certain region. We further assume that all of them are able to make an estimate, except nodes 1 and 7.

The weight matrix $\mathbf{W}$ is now given as

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

With a missing link between nodes 1 and 7, two elements in the weight matrix are zero and its rank is 3. We can reduce the rank of $\mathbf{W}$ by adding 1 to the diagonal entries of

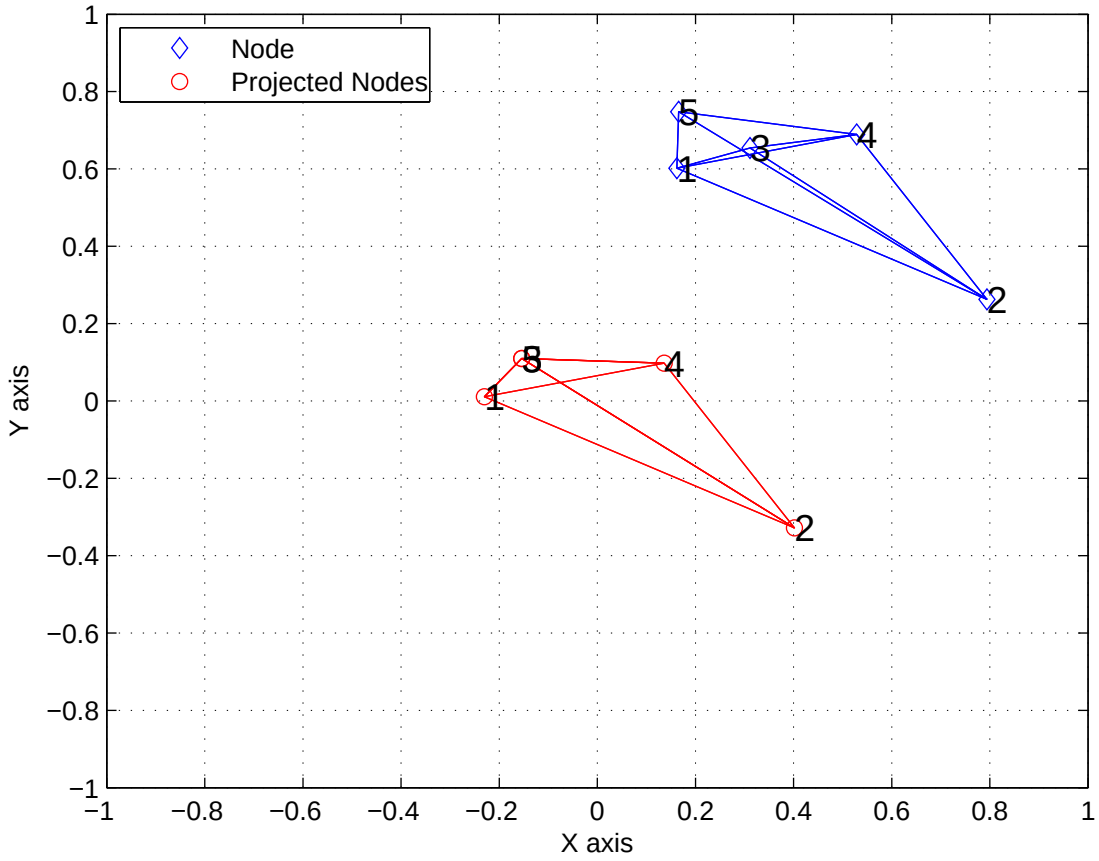


Figure 5.4: Original and projected node locations when the link between node 3 and node 5 is missing

the rows having a zero as

$$\mathbf{W} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

With this transformation the rank of $\mathbf{W}$ is reduced to two and the eigenvectors $\mathbf{u}_1$ and $\mathbf{u}_2$ corresponding to two nonzero singular values are given as

$$\mathbf{u}_1 = \begin{pmatrix} \frac{-1}{\sqrt{M}} \\ \frac{-1}{\sqrt{M}} \\ \vdots \\ \frac{-1}{\sqrt{M}} \end{pmatrix}$$

$$\mathbf{u}_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{-1}{\sqrt{2}} \\ 0 \\ 0 \end{pmatrix}$$

$\mathbf{N}$ will now be given as

$$\mathbf{N} = \begin{pmatrix} \mathbf{\Phi}\text{diag}(\mathbf{u}_1) \\ \mathbf{\Phi}\text{diag}(\mathbf{u}_2) \end{pmatrix}$$

$$\mathbf{N} = \begin{bmatrix} \frac{-1}{\sqrt{M}}x_1 & \frac{-1}{\sqrt{M}}x_2 & \frac{-1}{\sqrt{M}}x_3 & \frac{-1}{\sqrt{M}}x_4 & \frac{-1}{\sqrt{M}}x_5 & \frac{-1}{\sqrt{M}}x_6 & \frac{-1}{\sqrt{M}}x_7 & \frac{-1}{\sqrt{M}}x_8 & \frac{-1}{\sqrt{M}}x_9 \\ \frac{-1}{\sqrt{M}}y_1 & \frac{-1}{\sqrt{M}}y_2 & \frac{-1}{\sqrt{M}}y_3 & \frac{-1}{\sqrt{M}}y_4 & \frac{-1}{\sqrt{M}}y_5 & \frac{-1}{\sqrt{M}}y_6 & \frac{-1}{\sqrt{M}}y_7 & \frac{-1}{\sqrt{M}}y_8 & \frac{-1}{\sqrt{M}}y_9 \\ \frac{1}{\sqrt{2}}x_1 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}}x_7 & 0 & 0 \\ \frac{1}{\sqrt{2}}y_1 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}}y_7 & 0 & 0 \end{bmatrix}$$

The beacon node matrix $\mathbf{N}_a$ will consist of the first six columns of $\mathbf{N}$ (As the rank of $\mathbf{W}$ is 2).

$$\mathbf{N}_a = \begin{bmatrix} \frac{-1}{\sqrt{M}}x_1 & \frac{-1}{\sqrt{M}}x_2 & \frac{-1}{\sqrt{M}}x_3 & \frac{-1}{\sqrt{M}}x_4 & \frac{-1}{\sqrt{M}}x_5 & \frac{-1}{\sqrt{M}}x_6 \\ \frac{-1}{\sqrt{M}}y_1 & \frac{-1}{\sqrt{M}}y_2 & \frac{-1}{\sqrt{M}}y_3 & \frac{-1}{\sqrt{M}}y_4 & \frac{-1}{\sqrt{M}}y_5 & \frac{-1}{\sqrt{M}}y_6 \\ \frac{1}{\sqrt{2}}x_1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}}y_1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In the previous chapter we derived the conditions under which $\mathbf{N}$(or $\mathbf{N}_a$) will have full rank. We define a matrix $\mathbf{A}$ consisting of signal eigenvectors of $\mathbf{W}$. As there are only two eigenvectors corresponding to nonzero eigenvalues for the above example, $\mathbf{A}$ will be

$$\mathbf{A} = \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \mathbf{u}_2^T \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} \\ \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & \frac{-1}{\sqrt{2}} & 0 & 0 \end{bmatrix}$$

Then $\mathbf{M}$ and $\mathbf{S}$ defined in (4.2) and (4.3) are given as

$$\mathbf{M} = \begin{bmatrix} \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} \\ \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} & \frac{-1}{\sqrt{M}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} \frac{-1}{\sqrt{M}}x_2 & \frac{-1}{\sqrt{M}}x_3 & \frac{-1}{\sqrt{M}}x_4 & \frac{-1}{\sqrt{M}}x_5 & \frac{-1}{\sqrt{M}}x_6 & \frac{-1}{\sqrt{M}}x_8 & \frac{-1}{\sqrt{M}}x_9 \\ \frac{-1}{\sqrt{M}}y_2 & \frac{-1}{\sqrt{M}}y_3 & \frac{-1}{\sqrt{M}}y_4 & \frac{-1}{\sqrt{M}}y_5 & \frac{-1}{\sqrt{M}}y_6 & \frac{-1}{\sqrt{M}}y_8 & \frac{-1}{\sqrt{M}}y_9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We have shown in the previous chapter that the rank of $\mathbf{S}$ will be two. If we add a linearly independent column to $\mathbf{S}$, its rank will be increased by one. Thus the rank of $\mathbf{N}$ will be

$$\mathbf{N} = rank(\mathbf{S}) + 2 = 4$$

as column 1 and column 7 of $\mathbf{N}$ are independent to $\mathbf{S}$.

The rank of the beacon node matrix will be

$$\mathbf{N}_a = rank(\mathbf{S}) + 1 = 3$$

as the beacon node matrix $\mathbf{N}_a$ does not include column 7.

The beacon node matrix will only be full rank when a beacon node misses a distance estimate from another beacon node. However if an ordinary node does not obtain a distance estimate either from a beacon node or from another ordinary node, $\mathbf{N}_a$ will be rank deficient.

Even if the beacon node matrix $\mathbf{N}_a$ is full rank, we further require that the noise subspace matrices $\mathbf{U}_{na}$ and $\mathbf{U}_{nb}$ must also be full rank. Simulation results indicate that this condition is not true even in the case of just one missing link. Thus the proposed method is unable to find node locations for partially connected networks.

5.2 Performance of the proposed method with secound rank reduction technique

Another possible transformation to reduce the rank of $\mathbf{W}$ is to subtract 1 from the diagonal entry of a row containing a zero as described in the previous chapter. For a network of five sensor nodes with a missing link between nodes 3 and 5, the weight matrix $\mathbf{W}$ with this transformation is given by

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The projection matrix $\mathbf{P}$ will now be given as

$$\mathbf{P} = \mathbf{I} - \mathbf{U}_s\mathbf{U}_s^T = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & 0 & -\frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & \frac{2}{3} & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

and

$$\mathbf{P}\mathbf{\Phi}^T = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & 0 & -\frac{1}{3} & 0 \\ -\frac{1}{3} & \frac{2}{3} & 0 & -\frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & \frac{2}{3} & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \\ x_5 & y_5 \end{bmatrix}$$

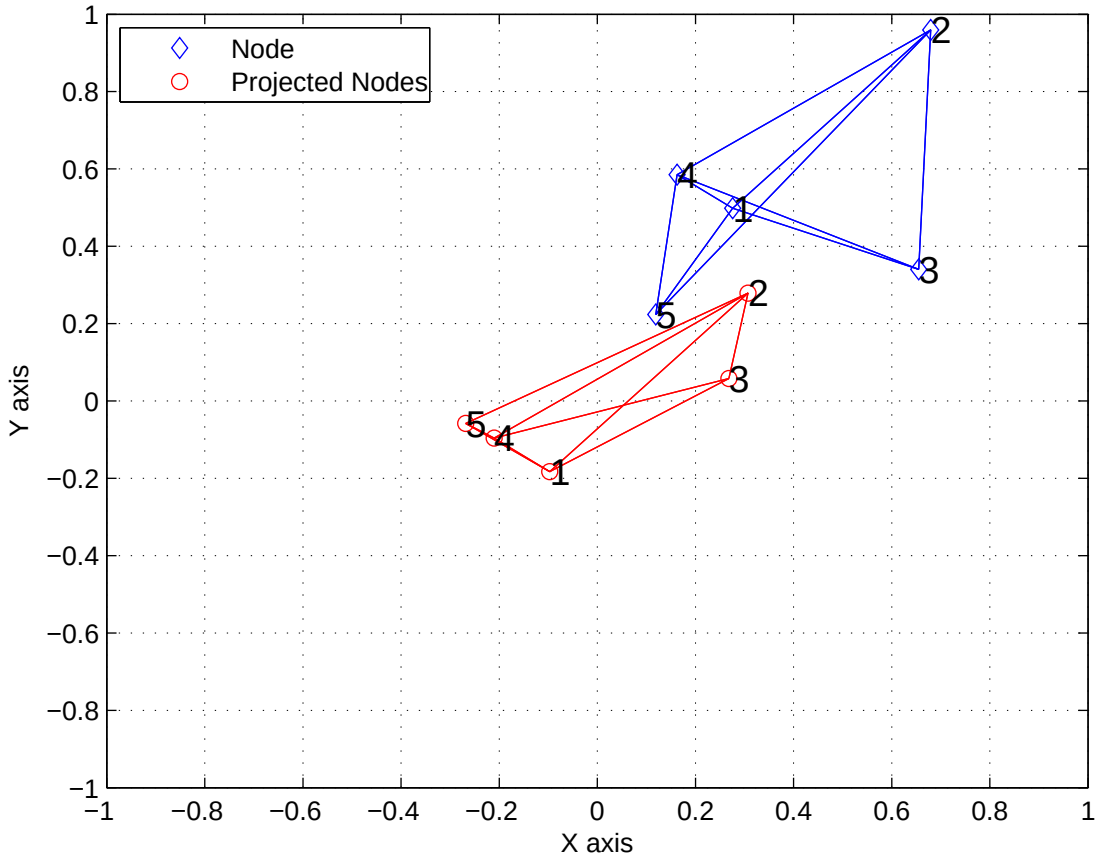


Figure 5.5: Original and projected node locations

$$= \begin{bmatrix} \frac{2}{3}x_1 - \frac{1}{3}(x_2 + x_4) & \frac{2}{3}y_1 - \frac{1}{3}(y_2 + y_4) \\ \frac{2}{3}x_2 - \frac{1}{3}(x_1 + x_4) & \frac{2}{3}y_2 - \frac{1}{3}(y_1 + y_4) \\ \frac{1}{2}x_3 - \frac{1}{2}x_5 & \frac{1}{2}y_3 - \frac{1}{2}y_5 \\ \frac{2}{3}x_4 - \frac{1}{3}(x_1 + x_2) & \frac{2}{3}y_4 - \frac{1}{3}(y_1 + y_2) \\ -\frac{1}{2}x_3 + \frac{1}{2}x_5 & -\frac{1}{2}y_3 + \frac{1}{2}y_5 \end{bmatrix}$$

The original and projected nodes with this transformation technique are shown in Figure (5.5). This transformation technique maps the fully connected cluster (nodes 1, 2, and 4) centered at origin, while ignoring the connections with nodes 1 and 3. It also maps the nodes having a missing link to a location which is the difference of their original positions multiplied with a scaling factor. Now if we assume that one of them (say node 3) is a beacon node, the the position of the other node can be determined.

The necessary condition for $\hat{\mathbf{N}}_a$ to be a full rank matrix is that the beacon node

matrix $\mathbf{N}_a$ must be full rank. We have shown that the full rank condition for beacon node matrix $\mathbf{N}_a$ is only satisfied when a beacon node misses a distance estimate from another beacon node. Thus to satisfy the full rank condition for $\mathbf{N}_a$, we require that both node 3 and node 5 must have known locations. However even with that assumption, the sufficient condition is not satisfied. Simulation results indicate that even with this transformation on $\mathbf{W}$ and with the assumption that missing link is between beacon nodes, the sparsity of noise subspace matrix $\mathbf{U}_{na}$ disallows the computation of node positions with the proposed method.

5.3 Conclusion

We provided a graphical interpretation by projecting the node locations onto the noise subspace of the weight matrix. We have shown that for a fully connected network, our method is able to find the global node positions in a single step. We have also demonstrated that the different weight transformations produce different node projections. One such transformation can produce relative node locations. However it is still infeasible to find global positions, even with the help of rank reduction techniques.

6 Conclusions and Future Work

We briefly conclude our work and provide possible options for future research in this chapter.

6.1 Conclusions

In this thesis, we have introduced a noise subspace approach for the localization in wireless sensor networks. We have shown that the proposed method determines the absolute node coordinates as opposed to MDS which finds local positions, for a fully connected network. When certain dissimilarity measurements are missing, the proposed method however fails to localize the network due to rank deficiency of $\hat{\mathbf{N}}_a$. We have also introduced some rank reduction techniques to overcome the rank deficiency problem of $\hat{\mathbf{N}}_a$. Though one such technique is able to estimate the relative positions, yet they fail to alleviate the rank deficiency associated with $\hat{\mathbf{N}}_a$. Hence the global node coordinates can not be determined for partially connected networks by the proposed method.

In chapter 3, we derived the least squares method by introducing a weight matrix whose elements depict the presence or absence of the corresponding dissimilarity measure. We take the Hadamard product between the squared distance matrix and the weight matrix and project this product on the noise subspace of the weight matrix $\mathbf{W}$. We have shown that the unknown node positions can be determined by multiplying this projection with a known matrix. We have provided the simulation results to indicate that the proposed method is able to find absolute node positions for a fully connected network.

In chapter 4, we provided a detailed analysis for the failure of the proposed method when certain dissimilarity measurements are missing. We derived the necessary and sufficient full row rank conditions of $\hat{\mathbf{N}}_a$. Unfortunately the conditions are not satisfied even with one missing link. Some transformations to reduce the rank of the weight matrix have also been introduced in the hope to solve the rank deficiency problem.

Chapter 5 provides a graphical explanation, where we have depicted the node projections onto the noise subspace of different weighting matrices. The full row rank condition for the anchor node matrix is only satisfied when the missing link exists between the anchor nodes. Even if that condition is satisfied, yet the method is unable to determine absolute node coordinates due to sparsity of noise subspace matrices.

6.2 Suggestions for Future Work

Localization in wireless sensor networks is an active research area and a number of different algorithms proposed recently address the computational complexity, communica-

tion overhead and the performance under noisy, incomplete dissimilarity measurements. Below we present some of the possible directions for further research.

- We have shown that a particular transformation on the weight matrix can possibly generate the relative node locations for partially connected networks. It would be appealing to formulate it further and to look for the possibility of finding global locations similar to MDS. An interesting aspect will be the fraction of anchor nodes required for such a solution.
- A number of distance matrix completion algorithms have been proposed recently [20] [21]. The application of these techniques in the domain of wireless sensor networks is yet to be investigated. As a pre phase the incomplete distance matrix can first be completed before the application of our proposed method. It would be quite interesting to compare the complexity and feasibility of distance matrix completion coupled with our algorithm with the weighted MDS techniques [11], [19].

Bibliography

[1] *Wireless Sensor Network Designs*. Hoboken, NJ: Wiley, 2003.

[2] N.Patwari, "Location estimation in sensor networks," Ph.D. dissertation, University Of Michigan, 2005.

[3] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," vol. 43, pp. 499–518, 2003.

[4] Y.Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Transactions on Parallel and Distributed Networks*, vol. 15, no. 11, pp. 961–974, Nov 2004.

[5] L.Dohetry, L. Ghauoi, and K. Psiter, "Convex position estimation in wireless sensor networks," 2001.

[6] P. Bhal. and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," *INFOCOMM*, 2000.

[7] T. Rappaport, *Wireless Communication, Principles and Practice*. Prentice Hall Inc, 1996.

[8] N.Patwari, A. Hero, and A. Perkins, "Relative location estimation in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 51, pp. 2137–2148.

[9] D. Niculescu and B. Nath, "Ad-hoc positioning system," *IEEE GlobeCom*, 2001.

[10] X.Li, "Collaborative localization with received-signal strength in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 6, no. 6, pp. 3807–3817, Nov 2007.

[11] J. Costa, N.Patwari, and A. Hero, "Distributed weighted multidimensional scaling for node localization in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, pp. 39–64.

[12] C.Faloutsos and K. Lin, "Fastmap: a fast algorithm for indexing, data mining and visualization," in *Proc. of ACM SIGMOD*, 1995, pp. 163–174.

[13] J. Wang, K.Lin, and D.Shasha, "Evaluating a class of distance mapping algorithms for data mining and clustering," in *Proc. of ACM KDD*, 1999, pp. 307–311.

[14] V.Silva and J. Tannenbaum, "Sparse multidimensional scaling using landmark points," 2001.

[15] J.C.Platt, "Fastmap, metricmap and landmark mds are all nystrom algorithms," in *10th International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 261–268.

[16] Y.Shang and W.Ruml, "Improved mds based localization," *INFOCOMM*, March 2004.

[17] T.Yang, J.Liu, and W.Wang, "A fast approximation to mds," in *INFOCOMM*, 2003.

[18] V. Duc, V. Nhat, and S. Challa, "Weighted mds for sensor localization," in *Proceedings of the International Conference on Computational Science and its Applications*, 2008, pp. 409–418.

[19] F.Zaho, Y.Lin, and Q.Yuan, "Distributed weighted least squares scaling with soft-constraints for node localization in wireless sensor networks," in *CHINACOM 07*, Aug 2007.

[20] E.Candes, J. Romberg, and T. Tao, "Robust unecrtainty priniciples: exact signal reconstruction from highly incomplete," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–502, Feb 2006.

[21] D. Petros, M. Ismail, and M. Gopal, "Distance matrix reconstruction from incomplete distance information for sensor network localization," in *IEEE Annual Communications society on Sensors and Ad-hoc Communications networks SECON*, 2006, pp. 163–174.