UNIVERSITY OF SOUTHAMPTON

FACULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES

Mathematical Sciences

Numerical Methods for Constrained Euclidean Distance Matrix Optimization

by

Shuanghua Bai

Thesis for the degree of Doctor of Philosophy

July 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES
Mathematical Sciences

Doctor of Philosophy

NUMERICAL METHODS FOR CONSTRAINED EUCLIDEAN DISTANCE MATRIX OPTIMIZATION

by Shuanghua Bai

This thesis is an accumulation of work regarding a class of constrained Euclidean Distance Matrix (EDM) based optimization models and corresponding numerical approaches. EDM-based optimization is powerful for processing distance information which appears in diverse applications arising from a wide range of fields, from which the motivation for this work comes. Those problems usually involve minimizing the error of distance measurements as well as satisfying some Euclidean distance constraints, which may present enormous challenge to the existing algorithms. In this thesis, we focus on problems with two different types of constraints. The first one consists of spherical constraints which comes from spherical data representation and the other one has a large number of bound constraints which comes from wireless sensor network localization.

For spherical data representation, we reformulate the problem as an Euclidean distance matrix optimization problem with a low rank constraint. We then propose an iterative algorithm that uses a quadratically convergent Newton-CG method at its each step. We study fundamental issues including constraint nondegeneracy and the nonsingularity of generalized Jacobian that ensure the quadratic convergence of the Newton method. We use some classic examples from the spherical multidimensional scaling to demonstrate the flexibility of the algorithm in incorporating various constraints.

For wireless sensor network localization, we set up a convex optimization model using EDM which integrates connectivity information as lower and upper bounds on the elements of EDM, resulting in an EDM-based localization scheme that possesses both efficiency and robustness in dealing with flip ambiguity under the presence of high level of noises in distance measurements and irregular topology of the concerning network of moderate size.

To localize a large-scale network efficiently, we propose a patching-stitching localization scheme which divides the network into several sub-networks, localizes each sub-network separately and stitching all the sub-networks together to get the recovered network. Mechanism for separating the network is discussed. EDM-based optimization model can be extended to add more constraints, resulting in a flexible localization scheme for various kinds of applications. Numerical results show that the proposed algorithm is promising.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Shuanghua Bai, declare that the thesis entitled *Numerical Methods for Constrained Euclidean Distance Matrix Optimization* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: (Bai et al., 2015; Bai and Qi, 2016)

Signed:.....

Date:.....

Acknowledgements

First of all, I would like to state my sincerest thanks to my supervisor Professor Houduo Qi for his continuous support and guidance throughout my postgraduate studies. His excellent mathematical knowledge and invaluable advice contributed enormously to the accomplishment of this work. I feel very fortunate to have him as my supervisor and mentor.

I am also grateful to my advisers Dr. Tri-Dung Nguyen and Professor Huifu Xu for their help and suggestions during my studies. I would like to thank my examiners Prof. Michal Kočvara from University of Birmingham and Dr. Tri-Dung Nguyen for their careful reading, supports as well as their comments, which have greatly improved this thesis.

My thanks also go to the previous and present members in the Operational Research Group. I specially thank former research fellow Chao Ding for his enlightening suggestions and numerous discussions regarding the technical aspects in the research on sensor network localization. I am grateful to the optimization research group in National University of Singapore for their generous sharing and patient discussion which help me a lot in developing my coding skills. I would also like to thank the research group in Beijing Jiaotong University led by Professor Naihua Xiu for their sharing on the research trend and information in many interesting optimization topics. I also place my sense of gratitude to my office mates and fellow PhD students for their company over the years.

Finally and most importantly, I would like to thank my parents and my brother Shunhua for their endless and unconditional love and support all through my life.

Chapter 1

Introduction

In this thesis, we focus on designing algorithms for a class of Euclidean Distance Matrix (EDM) based optimization problems. In particular, we are interested in EDM-based optimization problems with two types of constraints: spherical constraints and bound constraints. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be n points in $\mathbb{R}^r$, where $r > 0$ is known as the embedding dimension of those points. The primary information that is available for those points is the measured Euclidean distances among them

$$d_{ij} \approx \|\mathbf{x}_i - \mathbf{x}_j\|, \quad \text{for some pairs } (\mathbf{x}_i, \mathbf{x}_j), \tag{1.1}$$

which may be incomplete or noisy, or both. The aim of EDM-based optimization is to recover the (relative or global) coordinates of these points in a target space $\mathbb{R}^r$ purely based on those available distances. Such problems are usually encountered with cone constraints and rank constraints, which would bring nonsmoothness and nonconvexity to the optimization model. So algorithms need to be designed for solving the problems with specific constraints accurately and efficiently.

This chapter is split into three sections. In Section 1.1, we cover the background to Euclidean Distance Matrix which is the fundamental concept of our modelling process and algorithm design. In Section 1.2, we give an introduction to semismooth

Newton method that is the main approach to deal with spherical constraints. In Section 1.3, we cover a novel convergent Alternating Direction Method with Multipliers (ADMM) which allows us to deal with large amount of bound constraints in conic programming.

1.1 Background on Euclidean Distance Matrix

Let $\mathcal{S}^n$ denote the space of $n \times n$ symmetric matrices equipped with the standard inner product $\langle A, B\rangle = \mathrm{Tr}(AB)$ for $A, B \in \mathcal{S}^n$. Let $\|\cdot\|$ denote the induced Frobenius norm. Let $\mathcal{S}^n_+$ denote the cone of positive semidefinite matrices in $\mathcal{S}^n$ (often abbreviated as $X \succeq 0$ for $X \in \mathcal{S}^n_+$). The so-called *hollow* subspace $\mathcal{S}^n_h$ is defined by (":=" means define)

$$\mathcal{S}^n_h := \{A \in \mathcal{S}^n : \quad \mathrm{diag}(A) = 0\},$$

where $\mathrm{diag}(A)$ is the vector formed by the diagonal elements of A. For subsets α, β of $\{1, \ldots, n\}$, denote $A_{\alpha\beta}$ as the submatrix of A indexed by α and β (α for rows and β for columns). A_α denotes the submatrix consisting of columns of A indexed by α, and $|\alpha|$ is the cardinality of α. Throughout the thesis, vectors are treated as column vectors. For example, x^T is a row vector for $x \in \mathbb{R}^n$. The vector e is the vector of all ones and I denotes the identity matrix, whose dimension is clear from the context. When it is necessary, we use I_n to indicate its dimension n. Let e_i denote the ith unit vector, which is the ith column of I. We also need the following two important linear transformations.

The first one is Householder transformations, which are orthogonal transformations that describe reflections about hyperplanes containing the origin. Let

$$v := [1, \ldots, 1, 1 + \sqrt{n}]^T = e + \sqrt{n} e_n.$$

Then

$$Q = I_n - \frac{2}{v^T v} v v^T$$

is the Householder transformation that maps $e \in \mathbb{R}^n$ to the vector $[0, \ldots, 0, -\sqrt{n}]^T \in \mathbb{R}^n$.

The second one is the geometric centering transformation, which centers a set of points at their geometric center. Consider a collection of n points in $\mathbb{R}^r$, ascribed to the columns of matrix $X \in \mathbb{R}^{r \times n}$, $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, $\mathbf{x}_i \in \mathbb{R}^r$. The centroid is the mean of all the points

$$\mathbf{x}_c = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i = \frac{1}{n} X e.$$

By subtracting this vector from all the points in the set, we have the set of centralized points as

$$X_c = X - \mathbf{x}_c e^T = X(I_n - \frac{1}{n} e e^T).$$

Then the geometric centering transformation is defined as

$$J := I_n - \frac{1}{n} e e^T. \tag{1.2}$$

We often use the following properties:

$$J^2 = J, \quad Q^2 = I \quad \text{and} \quad J = Q \begin{bmatrix} I_{n-1} & 0 \\ 0 & 0 \end{bmatrix} Q. \tag{1.3}$$

1.1.1 Squared Euclidean Distance Matrix

A matrix D is a (squared) EDM if $D \in \mathcal{S}_h^n$ and there exist points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^r$ such that $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ for $i, j = 1, \ldots, n$. $\mathbb{R}^r$ is often referred to as the

embedding space and r is the embedding dimension when it is the smallest such r. Consider the following example of EDM for the case $n = 3$.

Example 1.1. *It is easy to check the following matrix D is an EDM. One of the sets of points in $\mathbb{R}^2$ that generate D is depicted in Figure 1.1*

$$D = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} = \begin{bmatrix} 0 & D_{12} & D_{13} \\ D_{21} & 0 & D_{23} \\ D_{31} & D_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 5 \\ 1 & 0 & 4 \\ 5 & 4 & 0 \end{bmatrix}$$

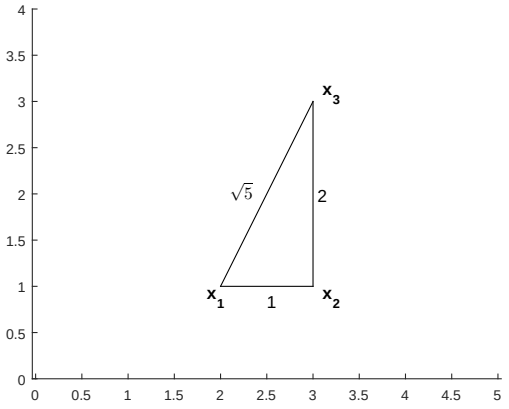


Figure 1.1: A set of points in $\mathbb{R}^2$ that generate an example of EDM

We note that any rotation, reflection or translation of a set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ would give the same D (i.e. the Euclidean distances between points remain unchanged under rigid transformation). In other words, there are infinitely many sets of points that correspond to the same D. Therefore, given the EDM, only relative coordinates of points can be recovered. To achieve global coordinates in a desired coordinate system, additional information (points with known positions) and procedure (Procrustes analysis) are required. We will introduce the details in the following sections.

1.1.2 Characterizations of EDM

It is well-known that a matrix $D \in \mathcal{S}^n$ is an EDM if and only if

$$D \in \mathcal{S}_h^n \quad \text{and} \quad J(-D)J \succeq 0. \tag{1.4}$$

The origin of this result can be traced back to Schoenberg (1935) and an independent work by Young and Householder (1938). See also Gower (1985) for a nice derivation of (1.4). Moreover, the corresponding embedding dimension is $r = \text{rank}(JDJ)$.

From the definition in (1.2), it is noted that the matrix J, when treated as an operator, is the orthogonal projection onto the subspace $e^\perp := \{x \in \mathbb{R}^n : \ e^T x = 0\}$. Characterization (1.4) simply means that D is an EDM if and only if $D \in \mathcal{S}_h^n$ and D is negative semidefinite on the subspace $e^\perp$:

$$-D \in \mathcal{K}_+^n := \left\{A \in \mathcal{S}^n : \quad x^T A x \geq 0, \ \forall \ x \in e^\perp\right\}.$$

It follows that $\mathcal{K}_+^n$ is a closed convex cone (known as the almost positive semidefinite cone). This gives us a window of using conic programming in dealing with distance related problems. Let $\Pi_{\mathcal{K}_+^n}(D)$ denote the orthogonal projection of $D \in \mathcal{S}^n$ onto $\mathcal{K}_+^n$:

$$\Pi_{\mathcal{K}_+^n}(D) := \arg\min \ \|D - Y\| \quad \text{s.t} \quad Y \in \mathcal{K}_+^n.$$

A nice property is that this projection can be done through the orthogonal projection onto the positive semidefinite cone $\mathcal{S}_+^n$ and is due to Gaffke and Mathar (1989)

$$\Pi_{\mathcal{K}_+^n}(D) = D + \Pi_{\mathcal{S}_+^n}(-JDJ) \qquad \forall \ D \in \mathcal{S}^n. \tag{1.5}$$

The other formula for computing $\Pi_{\mathcal{K}_+^n}$ is due to Hayden and Wells (1988, Thm. 2.1):

$$D \in \mathcal{K}_+^n \iff QDQ := \begin{bmatrix} \hat{D} & \hat{d} \\ \hat{d}^T & \hat{d}_0 \end{bmatrix} \quad \text{and} \quad \hat{D} \in \mathcal{S}_+^{n-1}, \tag{1.6}$$

and

$$\Pi_{\mathcal{K}_+^n}(D) = Q \begin{bmatrix} \Pi_{\mathcal{S}_+^{n-1}}(\hat{D}) & \hat{d} \\ \hat{d}^T & \hat{d}_0 \end{bmatrix} Q, \quad \forall D \in \mathcal{S}^n. \tag{1.7}$$

Because of (1.7), the cone $\mathcal{K}_+^n$ can be described as follows:

$$\mathcal{K}_+^n = \left\{ Q \begin{bmatrix} Z & z \\ z^T & z_0 \end{bmatrix} Q : \begin{array}{ll} Z \in \mathcal{S}_+^{n-1} & \\ z \in \mathbb{R}^{n-1} & z_0 \in \mathbb{R} \end{array} \right\}. \tag{1.8}$$

Its polar cone $(\mathcal{K}_+^n)^\circ$ is then given by

$$(\mathcal{K}_+^n)^\circ = \left\{ Q \begin{bmatrix} Z & 0 \\ 0 & 0 \end{bmatrix} Q : Z \in -\mathcal{S}_+^{n-1} \right\}. \tag{1.9}$$

We will use (1.5) for the implementation of our algorithm and (1.8) and (1.9) for theoretical analysis.

1.1.3 Coordinates recovery from EDM

In this section, we mainly introduce process for recovering the coordinates of points from EDM. If D is an EDM, from the definition introduced in section 1.1.1,

$$D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_i - 2\mathbf{x}_i\mathbf{x}_j + \mathbf{x}_j^T\mathbf{x}_j.$$

Let $X \in \mathbb{R}^{r\times n}$, $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ be a collection of n points, then

$$D = e\mathrm{diag}(X^TX)^T - 2X^TX + \mathrm{diag}(X^TX)e^T, \tag{1.10}$$

which is an obvious relation between coordinates of points X and the EDM D. Define the matrix

$$G := X^T X, \tag{1.11}$$

which is always called Gram matrix. From Gower (1982), the set of coordinates can be obtained through the decomposition:

$$-\frac{1}{2}JDJ = X^T X. \tag{1.12}$$

We note that the decomposition is possible because the matrix $(-JDJ)$ is positive semidefinite according to (1.4).

The results in (1.4) and (1.12) are true when D is a true EDM. What should one do if D is not a true EDM? The most popular method is the classical Multidimensional Scaling (cMDS) (Cox and Cox, 2000; Borg and Groenen, 2005), which simply computes the nearest positive semidefinite matrix from $(-JDJ)$ and is obtained through the following optimization:

$$\min_{Y} \ \|J(Y-D)J\|^2 \ \text{ s.t. } \ -JYJ \succeq 0 \text{ and } Y \in \mathcal{S}_h^n. \tag{1.13}$$

The optimal solution is just the orthogonal projection of $(-JDJ)$ onto $\mathcal{S}_+^n$ and is denoted by $\Pi_{\mathcal{S}_+^n}(-JDJ))$. cMDS then uses this projection in replace of $(-JDJ)$ in (1.12) to get the embedding points in X. This method is also known as principal coordinate analysis by Gower (1966). We summarize the cMDS algorithm as Algorithm 1. We need to point out here that cMDS works well when D is close to a true EDM. Otherwise it may perform poorly in terms of embedding quality due to the rank of Gram matrix being too high.

As introduced in Section 1.1.1, any rigid transformation of X would yield the same distance matrix D. In order to find a desired set of points that match positions of certain existing points, one needs to conduct the Procrustes analysis, which is

Algorithm 1 Classical MDS

1: **Input:** Distance matrix D and embedding dimension r.
2: Compute Gram matrix $G = -\frac{1}{2}JDJ$.
3: Conduct eigenvalue decomposition on G: $G = P\Lambda P^T$, where $\Lambda = \mathrm{Diag}(\boldsymbol{\lambda})$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of G being arranged in the non-increasing order. $P := [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n] \in \mathcal{O}^n$ with $\mathcal{O}^n$ being the set of all $n \times n$ orthogonal real matrices.
4: **Output:** $X = \left[\mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r}\right), \mathbf{0}_{r\times(n-r)}\right] P^T$.

a computational scheme and often has a closed-form formula. In the following part of this section, we briefly introduce the method, more details can be found in Gower (1975).

Suppose the coordinates of the first m points are known in advance, denoted as $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$, and this kind of known points is always referred to as *anchor* node in the research of sensor network localization. Let $\overline{X}$ be the result obtained by Algorithm 1. Denote

$$\overline{X} = [\overline{X}_1, \overline{X}_2], \text{ with } \overline{X}_1 \in \mathbb{R}^{r\times m}, \ \overline{X}_2 \in \mathbb{R}^{r\times(n-m)},$$

where $\overline{X}_1$ contains m points corresponding to the m points with known positions, whose true positions are collected in $A = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$. Our first task is to match $\overline{X}_1$ to A. To do this, we first centralize the points in $\overline{X}_1$ and A by

$$\overline{X}_1^c := \overline{X}_1 J_m \quad \text{and} \quad A^c := AJ_m.$$

Denote

$$\overline{\mathbf{a}} := \frac{1}{m}\sum_{i=1}^{m} \mathbf{a}_i \qquad \text{and} \qquad \overline{\mathbf{x}} := \frac{1}{m}\sum_{i=1}^{m} \overline{\mathbf{x}}_i.$$

We then find the best rotation matrix Q by solving

$$\min_{Q\in\mathbb{R}^{r\times r}} f = \|Q\overline{X}_1^c - A^c\|, \quad \text{s.t. } Q^TQ = I_r. \tag{1.14}$$

Problem (1.14) has a closed form solution $Q = U_\Sigma V_\Sigma^T$, where U_Σ and V_Σ are from

the singular-value-decomposition of $A^c(\overline{X}_1^c)^T = U_\Sigma \Sigma V_\Sigma^T$. Then the remaining $n - m$ points will be mapped to

$$\mathbf{x}_i = Q(\overline{\mathbf{x}}_i - \overline{\mathbf{x}}) + \overline{\mathbf{a}}, \quad i = m+1, \ldots, n, \tag{1.15}$$

which are the positions of points in the same coordinate system as anchors.

1.1.4 Related methods

Over the past decade, Euclidean Distance Matrix has been receiving increased attention from researchers for two main reasons. The first one is that it covers many application areas such as Multidimensional Scaling (MDS) (Cox and Cox, 2000; Borg and Groenen, 2005), Sensor Network Localization (SNL) (Biswas and Ye, 2004; Biswas et al., 2006; Kim et al., 2012; Wang et al., 2008; Zhang et al., 2010; Alfakih et al., 2011; Ding et al., 2010), and Dimensionality Reduction (DR) (Weinberger and Saul, 2006; Weinberger et al., 2006; Liberti et al., 2014) in feature extraction and machine learning. More applications can be found in (Al-Homidan and Wolkowicz, 2005; Alfakih and Wolkowicz, 1998; Krislock and Wolkowicz, 2010), see also (Krislock and Wolkowicz, 2012) for detailed summary. The second reason is that EDM has a close relation with semidefinite matrix, and by reformulating EDM-based optimization as Semidefinite Programming (SDP), using the relation in (1.10) and (1.11), one can solve some of the problems very efficiently using powerful SDP solvers such as SDPT3 (Toh et al., 1999), SDPNAL (Zhao et al., 2010), and PENNON (Kočvara and Stingl, 2003).

As discussed in the above section, Algorithm 1 requires the distance matrix D to be close to a true EDM to work well, which narrows down the applications in which it can be used, since in many circumstances the matrix D is noisy and incomplete. Lots of works have been done to tackle such problem. A well-known

approach is called SMACOF (De Leeuw and Mair, 2009), which minimizes the sum of squares, commonly called *stress*, which is defined as

$$\sigma(X) := \sum_{i,j=1}^{n} h_{ij}(d_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|)^2. \tag{1.16}$$

Here, H is a known $n \times n$ matrix of *weights* h_{ij}. For example H can be used to indicate whether a distance between two points is acquired or not by defining $h_{ij} = 1$ for known distance and $h_{ij} = 0$ for missing distance. Since $\sigma(X)$ is a nonconvex function, De Leeuw and Mair (2009) proposed a majorization function to linearize the concave part in $\sigma(X)$. Moreover, SMACOF is capable to recover points on quadratic surfaces such as ellipse, hyperbola, parabola in $\mathbb{R}^2$ and ellipsoids, hyperboloids, paraboloids, and cylinders in $\mathbb{R}^3$.

Another approach is to first find an EDM that is as close as possible to the given matrix D, which involves a matrix completion problem, then use Algorithm 1 to recover the points' positions. Matrix completion is the task of filling in the missing entries of a partially observed matrix. In specific, Euclidean distance matrix completion problem is to find the nearest Euclidean distance matrix to the given data matrix D. Define $\mathcal{E}^n := \mathcal{S}_h^n \cap (-\mathcal{K}_+^n)$ as the cone of Euclidean distance matrices, as introduced by Krislock and Wolkowicz (2012), the Euclidean distance matrix completion problem can be posed as

$$\begin{aligned} \text{find} \quad & \hat{D} \in \mathcal{E}^n \\ \text{s.t.} \quad & H \circ \hat{D} = H \circ D, \end{aligned} \tag{1.17}$$

where H is defined in (1.16) indicating whether the distance information is acquired and $\circ$ denotes the Hadamard product among matrices. Several methods have been proposed to solve (1.17), one of them is via Semidefinite Programming (SDP) proposed by Alfakih et al. (1999), which introduced a linear mapping

$\mathcal{K}_V : \mathcal{S}_+^{n-1} \to \mathcal{E}^n$ defined by

$$\mathcal{K}_V(Y) = \mathrm{diag}(VYV^T)e^T + e\mathrm{diag}(VYV^T)^T - 2VYV^T,$$

where $V \in \mathbb{R}^{n\times(n-1)}$ satisfies $V^TV = I_{n-1}$ and $V^Te = 0$. $\mathcal{K}_V$ is a bond connecting semidefinite matrices and EDM, and formulated the EDM completion problem as the following SDP problem

$$\min_Y \|H \circ (\mathcal{K}_V(Y) - D)\|^2/2 \quad \text{s.t.} \quad Y \in \mathcal{S}_+^{n-1}, \tag{1.18}$$

where $\circ$ denotes the Hadamard product among matrices. Alfakih *et al.* designed a primal-dual interior point algorithm based on Gauss-Newton direction to solve problem (1.18), whose number of variables is $\mathcal{O}(n^2)$. This method can only deal with problems with size up to a hundred as when the scale of the problem goes large, much computational effort and storage capacity are required. For convex quadratic SDP problem like (1.18), Toh (2008) proposed an inexact primal-dual path following algorithm and the problem scale can go to a couple of thousands.

Reformulating EDM completion problem as SDP can avoid dealing with the $\mathcal{K}_+^n$ cone, but the objective function becomes more complicated. A direct approach is to model the problem based on $\mathcal{K}_+^n$ cone:

$$\min_Y \|H \circ (D - Y)\|^2/2 \quad \text{s.t.} \quad Y \in \mathcal{S}_h^n \cap (-\mathcal{K}_+^n). \tag{1.19}$$

Different methods are proposed to solve problem (1.19), such as alternating projection methods of Dykstra-Han type (Dykstra, 1983; Han, 1988), the Modified Alternating Projection (MAP) by Glunt et al. (1990), and an independent work by Gaffke and Mathar (1989). Qi (2013) first proposed a semismooth Newton-CG

method for its dual problem

$$\min_{y\in\mathbb{R}^n} \|\Pi_{\mathcal{K}_+^n}(D+\mathrm{Diag}(y))\|^2/2 \tag{1.20}$$

for the case $H = E$, then extend it to deal with problem with a general H using a majorization approach. Based on this work, by putting it in the framework of Majorized Penalty Approach (MPA) proposed by Gao and Sun (2010), Qi and Yuan (2014) successively extended the Newton-type method for the EDM completion problem with rank constraint

$$\begin{array}{ll} \min_Y & \frac{1}{2}\|H\circ(Y-D)\|^2 \\ \text{s.t.} & Y\in\mathcal{S}_h^n\cap(-\mathcal{K}_+^n), \\ & \mathrm{rank}(-JYJ)\le r. \end{array} \tag{1.21}$$

Some equality constraints may also be added to (1.19) and (1.21), but as pointed out in Qi (2013), if we have too many extra constraints in (1.19), we may lose the property of constraint nondegeneracy, which may destroy the quadratic convergence of the Newton method.

1.2 Background on semismooth Newton method

Consider the following unconstraint optimization problem:

$$\min_x f(x), \tag{1.22}$$

where $x\in\mathbb{R}^n$ is a real vector with $n\ge 1$ components and $f:\mathbb{R}^n\to\mathbb{R}$ is (at least) once continuously differentiable. The first-order optimality condition for x^* to be

a local minimizer is that x^* is a solution of the system of nonlinear equations:

$$F(x) := \nabla f(x) = 0, \tag{1.23}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is the gradient of f. To design Newton-type algorithm to solve (1.23), one needs to make assumptions on the function F.

Assume F is (at least) once continuously differentiable, the traditional approach for designing Newton's method is based on replacing the complicated nonlinear map F by its first order Taylor expansion about a given estimation $x_k \in \mathbb{R}^n$:

$$F(x_k + d) \approx F(x_k) + F'(x_k)d =: m_k(d), \tag{1.24}$$

where $F' : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the Jacobian matrix of F. then the Newton direction $d_k \in \mathbb{R}^n$ for the kth iteration can be obtained by solving

$$m_k(d) = 0.$$

Obviously this step is only well-defined if $F'(x_k)$ is non-singular. We summarize the classical Newton's Method, see e.g. Bertsekas (1999), in Algorithm 2. Theorem 1.1 gives the local rate-of-convergence properties of Newton's method.

Algorithm 2 Newton's Method

1: Given $f : \mathbb{R}^n \to \mathbb{R}$ twice continuously differentiable and $x_0 \in \mathbb{R}^n$, $k = 0$.
2: Unless a stopping criteria is satisfied, solve

$$\nabla^2 f(x_k)d = -\nabla f(x_k)$$

to get Newton direction d_k.
3: set $x_{k+1} = x_k + d_k$ and go to 2.

Theorem 1.1. *(Nocedal and Wright, 2006, Thm. 3.5). Suppose f is twice differentiable and that the Hessian $\nabla^2 f(x)$ is Lipschitz continuous in a neighbourhood of a local minimizer x^* at which $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*) \succ 0$. Consider the sequence $\{x_k\}$ generated by Algorithm 2. Then we have the following results.*

(i) If the starting point x_0 is sufficiently close to x^*, then the sequence of iterates converges to x^*.

(ii) The rate of convergence of $\{x_k\}$ is quadratic.

(iii) The sequence of gradient norms $\{\|\nabla f_k\|\}$ converges quadratically to zero.

Newton's method enjoys a fast convergence rate but it also has some drawbacks. Unless the initial point is sufficiently close to the local minimizer, the Hessian may not be invertible and may not be positive definite, which makes the Newton direction not be a decreasing direction. One of the methods to compensate this drawback is to conduct a line search method to choose a step length α_k at each iteration k instead of using 1 as the step length throughout the algorithm and update x_{k+1} as

$$x_{k+1} = x_k + \alpha_k d_k.$$

By combining the *Armijo condition* which stipulates that α_k should give sufficient decrease in the objective function f, a backtracking line search method, see e.g. Bertsekas (1999), is summarized in Algorithm 3

Algorithm 3 Backtracking Line Search

1: Choose $\bar{\alpha} > 0$, $\rho \in (0,1)$, $c \in (0,1)$; Set $\alpha \leftarrow \bar{\alpha}$;
2: **while** $f(x_k + \alpha d_k) > f(x_k) + c\alpha \nabla f_k^T d_k$ **do**
3: $\quad \alpha \leftarrow \rho\alpha$
4: **end while**
5: Terminate with $\alpha_k = \alpha$.

Algorithm 2 and 3 form a framework of Newton's method to solve unconstraint optimization problem with objective function being twice continuously differentiable. However, very often the nonlinear mapping $\nabla f(x) : \mathbb{R}^n \to \mathbb{R}^n$ is not necessarily differentiable in the classical (Fréchet) sense. This in fact coins the name "nonsmooth" in connection with analytical (nonsmooth analysis) as well as numerical issues (nonsmooth or generalized Newton's method). In the following

part of this section, we introduce some background of a special kind of nonsmooth function which is called *semismooth* function and its properties.

Definition 1.2. Let $F : \mathbb{R}^n \to \mathbb{R}^m$ and $x, h \in \mathbb{R}^n$. The directional derivative of F at x along h is the following limit (if it exists at all):

$$F'(x; h) = \lim_{t \to 0^+} \frac{F(x + th) - F(x)}{t}.$$

F is said to be directionally differentiable at x if $F'(x; h)$ exists for all h.

Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be a Lipschitz continuous function. By Rademacher's theorem (Rockafellar and Wets, 2009, Sect. 9.J), F is Fréchet differentiable almost everywhere. Let

$$D_F := \{x \in \mathbb{R}^n | F \text{ is differentiable at } x\}.$$

be the set of points in $\mathbb{R}^n$ where F is differentiable. Let $F'(x)$ denote the Jacobian of F at $x \in D_F$. The the Bouligand subdifferential of F at $x \in \mathbb{R}^n$ is defined by

$$\partial_B F(x) := \{V \in \mathbb{R}^{m \times n} | V \text{ is an accumulation point of } F'(x^k), x^k \to x, x^k \in D_F\}.$$

The Clark's generalized Jacobian (Clarke, 1990) is the convex hull of $\partial_B F(x)$, i.e.,

$$\partial F(x) = \text{conv}\{\partial_B F(x)\}.$$

Then the generalized Newton's method, see e.g. Bertsekas (1999), for nonsmooth systems is give in Algorithm 4.

Without requiring further properties of F, one can only expect linear convergence of Algorithm 4 at best if it converges at all, presumed that the iteration is well-defined. However, if the gradient F enjoys nice properties such as semismoothness

Algorithm 4 Generalized Newton's Method

1: Given $f : \mathbb{R}^n \to \mathbb{R}$ continuously differentiable and its gradient $F : \mathbb{R}^n \to \mathbb{R}^n$ locally Lipschitz continuous, $x_0 \in \mathbb{R}^n$, $k = 0$.
2: Unless a stopping criteria is satisfied, solve

$$V_k d = -F(x_k)$$

to get Newton direction d_k, where V_k is an arbitrary element of $\partial F(x_k)$.
3: set $x_{k+1} = x_k + d_k$ and go to 2.

or strong semismoothness, Algorithm 4 will have superlinear or even quadratic convergence rate under some conditions.

First introduced by Mifflin (1977) for functionals, the following concept of semismoothness was extended by Qi and Sun (1993) for vector-valued functions which are not differentiable, but locally Lipschitz continuous. Moreover, by the work from Sun and Sun (2002), the concept of semismoothness was pushed further to the case when a function is matrix-valued, which contains a very important type of function that is the function of semidefinite projection, which plays an important role in developing semismooth Newton methods and smoothing methods for semidefinite programming. Let $\mathcal{X}$ and $\mathcal{Y}$ be finite dimensional real Euclidean spaces each equipped with an inner product $\langle \cdot, \cdot \rangle$ and its norm $\| \cdot \|$.

Definition 1.3. Let $F : \mathcal{U} \subset \mathcal{X} \to \mathcal{Y}$ be a locally Lipschitz continuous function on the open set $\mathcal{U}$. F is said to be semismooth at a point $x \in \mathcal{U}$ if

1. F is directionally differentiable at x; and
2. for any $V \in \partial F(x + h)$ and $h \to 0$,

$$F(x + h) - F(x) - Vh = o(\|h\|), \quad h \in \mathcal{U}.$$

Furthermore, F is said to be strongly semismooth at a point $x \in \mathcal{U}$ if

$$F(x + h) - F(x) - Vh = o(\|h\|^2), \quad h \in \mathcal{U}.$$

With F being semismooth, the Algorithm 4 is called semismooth Newton's method. Qi and Sun (1993) showed that the iterate sequence generated by Algorithm 4 converges superlinearly under nonsingular condition. The following convergence result is essential.

Theorem 1.4. *((Qi and Sun, 1993, Thm. 3.2)). Let x^* be a solution of the equation $F(x) = 0$ and let F be a locally Lipschitz function which is semismooth at x^*. Assume that all $V \in \partial F(x^*)$ are nonsingular matrices. Then every sequence generated by Algorithm 4 is superlinearly convergent to $x*$, provided that the starting point x^0 is sufficiently close to x^*. Moreover, if F is strongly semismooth at x^*, the convergence rate is quadratic.*

In fact, many functions such as convex functions and smooth functions are semismooth everywhere. Moreover, showed by Sun and Sun (2002) in their paper, the function of semidefinite projection $\Pi_{\mathcal{S}^n_+}(\cdot)$ is strongly semismooth everywhere in $\mathcal{S}^n$. This result together with the projection relation (1.5) opens the gate to design semismooth Newton's method for EDM based optimization problems.

One key issue to implement Algorithm 4 is to compute the element in $\partial F(x_k)$. Given $X \in \mathcal{S}^n$, assume X has the following eigenvalue decomposition

$$X = P\Lambda P^T,$$

where Λ is the diagonal matrix with diagonal element being the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0 \geq \lambda_{k+1} \geq \cdots \geq \lambda_n$ of X and $P \in \mathcal{O}$ is the corresponding orthogonal matrix of eigenvectors. Then the semidefinite projection function is given by

$$\Pi_{\mathcal{S}^n_+}(X) = P\Lambda_+ P^T,$$

where $\Lambda_+ = \max\{\Lambda, 0\}$. Define the operator $\mathcal{W} : \mathcal{S}^n \to \mathcal{S}^n$, by

$$\mathcal{W}H = P(\Omega \circ (P^T H P))P^T, \quad H \in \mathcal{S}^n, \tag{1.25}$$

where

$$\Omega = \begin{pmatrix} E_k & \overline{\Omega} \\ \overline{\Omega}^T & 0 \end{pmatrix}, \overline{\Omega}_{ij} = \frac{\lambda_i}{\lambda_i - \lambda_j}, i \in \{1, 2, \ldots, k\}, j \in \{k+1, \ldots, n\}.$$

where E_k is the matrix of ones with dimensional k. It has been proved by Pang et al. (2003) that $\mathcal{W}H$ is actually an element of the set $\partial\Pi_{\mathcal{S}_+^n}(X)H$

1.3 Background on alternating direction method of multipliers

In this section, we provide background to a kind of first-order methods called alternating direction methods of multipliers (ADMM), which are often more suitable, and sometimes the only practical choice for solving large-scale problems. Furthermore, ADMM are capable to find a solution of low to medium accuracy and makes it efficient to deal with large amount of equality and inequality constraints, which brings difficulty to the semismooth Newton's method since the dual problem is no longer unconstraint.

Let $f : \mathcal{X} \to (-\infty, +\infty]$ and $g : \mathcal{Y} \to (-\infty, +\infty]$ be closed proper convex functions, and $\mathcal{F} : \mathcal{X} \to \mathcal{Z}$ and $\mathcal{G} : \mathcal{Y} \to \mathcal{Z}$ are linear operators, where $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be real finite dimensional Euclidean spaces with inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. Let ∂f and ∂g be the subdifferential mappings of f and g, respectively. Since the subdifferential mappings of the closed proper convex functions are maximal monotone (Rockafellar and Wets, 2009, Theorem 12.17), there exist two self-adjoint and positive semidefinite operators Σ_f and Σ_g such that for all $x, \hat{x} \in \mathrm{dom}(f)$, $u \in \partial f(x)$, and $\hat{u} \in \partial f(\hat{x})$,

$$f(x) > f(\hat{x}) + \langle \hat{u}, x - \hat{x} \rangle + \frac{1}{2}\|x - \hat{x}\|_{\Sigma_f}^2 \quad \text{and} \quad \langle u - \hat{u}, x - \hat{x} \rangle \geq \|x - \hat{x}\|_{\Sigma_f}^2, \quad (1.26)$$

and for all $y, \hat{y} \in \mathrm{dom}(g)$, $v \in \partial g(y)$, and $\hat{v} \in \partial g(\hat{y})$,

$$g(y) > g(\hat{y}) + \langle \hat{v}, y - \hat{y} \rangle + \frac{1}{2}\|y - \hat{y}\|^2_{\Sigma_g} \quad \text{and} \quad \langle v - \hat{v}, y - \hat{y} \rangle \geq \|y - \hat{y}\|^2_{\Sigma_g}. \tag{1.27}$$

1.3.1 2-block semi-proximal ADMM

Consider the 2-block convex optimization problem with the following separable structure:

$$\begin{array}{ll} \min\limits_{x,y} & f(x) + g(y) \\ \text{s.t.} & \mathcal{F}^*(x) + \mathcal{G}^*(y) = c. \end{array} \tag{1.28}$$

And its dual problem is given by

$$\min \left\{ \langle c, z \rangle + f^*(s) + g^*(t) \mid \mathcal{F}(z) + s = 0,\ \mathcal{G}(z) + t = 0 \right\}. \tag{1.29}$$

Let $\sigma > 0$ be a parameter. The augmented Lagrangian function for problem (1.28) is given by (e.g., see Rockafellar and Wets (2009, Sec. 11K))

$$\mathcal{L}_\sigma(x, y; z) = f(x) + g(y) + \langle \mathcal{F}(x) + \mathcal{G}(y) - c, z \rangle + \frac{\sigma}{2}\|\mathcal{F}(x) + \mathcal{G}(y) - c\|^2. \tag{1.30}$$

The classical augmented Lagrangian method (Hestenes, 1969; Powell, 1967; Rockafellar, 1976) consists of the following two steps on the kth iteration:

$$\left(x^{k+1}, y^{k+1}\right) = \arg\min_{x,y} \mathcal{L}_\sigma(x, y; z^k) \tag{1.31}$$

$$z^{k+1} = z^k + \tau\sigma\left(\mathcal{F}(x^{k+1}) + \mathcal{G}(y^{k+1}) - c\right), \tag{1.32}$$

where $\tau > 0$, e.g., $\tau \in (0, \frac{1+\sqrt{5}}{2})$, is a positive constant that controls the step length in (1.32). To solve subproblem (1.31) exactly or approximately with high accuracy can be a challenging task in many situations. To tackle this challenge, one may try to solve (1.32) in terms of x and y alternately other than simultaneously.

In the following part of this section, we mainly discuss a semi-proximal ADMM proposed by Fazel et al. (2013), which is more general and applicable than most of the previous works on ADMM. We summarize the semi-proximal ADMM in the following Algorithm 5. Since the proximal terms that be added can be positive semidefinite other than positive definite, the corresponding method is called semi-proximal ADMM instead of proximal ADMM.

Algorithm 5 Semi-Proximal ADMM

1: Set initial points $(x^0, y^0, z^0) \in \mathrm{dom} f \times \mathrm{dom} g \times \mathcal{Z}$. Choose $\sigma > 0$ as the penalty parameter, $\tau \in (0, \frac{1+\sqrt{5}}{2})$ is the step length, and S and T are two self-adjoint positive semidefinite, not necessarily positive definite, operators on $\mathcal{X}$ and $\mathcal{Y}$, respectively.

2: Compute

$$x^{k+1} = \arg\min_{x \in \mathcal{X}} \mathcal{L}_\sigma(x, y^k, z^k) + \frac{\sigma}{2}\|x - x^k\|_S^2, \tag{1.33}$$

3: Compute

$$y^{k+1} = \arg\min_{y \in \mathcal{Y}} \mathcal{L}_\sigma(x^{k+1}, y, z^k) + \frac{\sigma}{2}\|y - y^k\|_T^2, \tag{1.34}$$

4: Compute

$$z^{k+1} = z^k + \tau\sigma\left(\mathcal{F}(x^{k+1}) + \mathcal{G}(y^{k+1}) - c\right). \tag{1.35}$$

5: If a termination criterion is not met, go to step 2.

Algorithm 5 will reduce to the classical ADMM introduced by Glowinski and Marroco (1975) and Gabay and Mercier (1976) when $S = 0$ and $T = 0$. The presence of S and T can help to guarantee the existence of solutions for the subproblem (1.33) and (1.34). Moreover, they can ensure the boundedness of the two generated sequences $\{x^{k+1}\}$ and $\{y^{k+1}\}$. The choices of S and T depends very much on the problem, the basic principle is that S and T should be as small as possible while $\{x^{k+1}\}$ and $\{y^{k+1}\}$ are still easy to compute.

For the convergence of Algorithm 5, we need the following assumptions on the constraint qualification (CQ).

Assumption 1.5. *There exists* $(x_0, y_0) \in ri(dom\ f \times dom\ g)$ *that* $\mathcal{F}(x_0)+\mathcal{G}(y_0) = c$.

Based on the result in Fazel et al. (2013), we have the convergence result below, see also Sun et al. (2014) for details.

Theorem 1.6. *Let* S *and* T *be the self-adjoint and positive semidefinite operators defined by (1.26) and (1.27), respectively. Suppose that the solution set of the problem (1.28) is nonempty and Assumption (1.5) holds. Assume that* S *and* T *are chosen such that the sequence* $\{(x_k, y_k, z_k)\}$ *generated by Algorithm 5 is well defined. Then under the condition either (a)* $\tau \in (0, \frac{1+\sqrt{5}}{2})$ *or (b)* $\tau \geq \frac{1+\sqrt{5}}{2}$ *but* $\sum_{k=0}^{\infty} \left(\|\mathcal{G}^*(y^{k+1} - y^k)\|^2 + \tau^{-1}\|\mathcal{F}^*(x^{k+1}) + \mathcal{G}^*(y^{k+1}) - c\|^2\right) < \infty$, *the following results hold:*

(i) If $(x^\infty, y^\infty, z^\infty)$ *is an accumulation point of* $\{(x_k, y_k, z_k)\}$, *then* (x^∞, y^∞) *solves problem (1.28) and* z^∞ *solves problem (1.29).*

(ii) If both $\sigma^{-1}\Sigma_f + S + \mathcal{F}\mathcal{F}^*$ *and* $\sigma^{-1}\Sigma_g + T + \mathcal{G}\mathcal{G}^*$ *are positive definite, then the sequence* $\{(x_k, y_k, z_k)\}$, *which is automatically well defined, converges to a unique limit, say,* $(x^\infty, y^\infty, z^\infty)$ *with* (x^∞, y^∞) *solves problem (1.28) and* z^∞ *solves problem (1.29).*

(iii) When the y-part disappears, the corresponding results in parts (i) and (ii) hold under the condition either $\tau \in (0, 2)$ *or* $\tau \geq 2$ *but* $\sum_{k=0}^{\infty} \|\mathcal{G}^* y^{k+1} - c\|^2 < \infty$.

1.3.2 Schur complement based semi-proximal ADMM

The problem will be more complicated when it comes to be multi-block. The motivation for reviewing ADMM for multi-block problems comes from interests in the sensor network localization problem which is thoroughly investigated in

Chapter 3. The sensor network localization problem is an important special case of the following convex quadratic conic programming

$$\begin{array}{ll} \min & \frac{1}{2}\langle X, \mathcal{Q}X\rangle + \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}_E X = b_E, \mathcal{A}_I X \geq b_I, X \in \mathcal{K}_+^n \cap \mathcal{K}, \end{array} \tag{1.36}$$

where $\mathcal{K}_+^n$ is the almost positive semidefinite cone introduced in Section 1.1.2, $\mathcal{Q}$ is a self-adjoint positive semidefinite linear operator from $\mathcal{S}^n$ to $\mathcal{S}^n$, $\mathcal{A}_E : \mathcal{S}^n \to \mathbb{R}^{m_E}$ and $\mathcal{A}_I : \mathcal{S}^n \to \mathbb{R}^{m_I}$ are two linear maps, $C \in \mathcal{S}^n$, $b_E \in \mathbb{R}^{m_E}$, $b_I \in \mathbb{R}^{m_I}$ are given data, $\mathcal{K}$ is nonempty simple closed convex set, e.g., $\mathcal{K} = \{W \in \mathcal{S}^n : L \leq W \leq U\}$ with $L, U \in \mathcal{S}^n$ being given matrices. The dual of problem (1.36) is given by

$$\begin{array}{ll} \max & -\delta_{\mathcal{K}}^*(-Z) - \frac{1}{2}\langle X', \mathcal{Q}X'\rangle + \langle b_E, y_E\rangle + \langle b_I, y_I\rangle \\ \text{s.t.} & Z - \mathcal{Q}X' + S + \mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I = C, \\ & X' \in \mathcal{S}^n,\ y_I \geq 0,\ S \in (\mathcal{K}_+^n)^*, \end{array} \tag{1.37}$$

where $(\mathcal{K}_+^n)^*$ is the dual cone of $\mathcal{K}_+^n$ and $\delta_{\mathcal{K}}^*(\cdot)$ is the conjugate function of $\delta_{\mathcal{K}}(\cdot)$ given by

$$\delta_{\mathcal{K}}^*(-Z) = \sup_{W \in \mathcal{K}} \langle -Z, W\rangle = -\inf_{W \in \mathcal{K}} \langle Z, W\rangle\,. \tag{1.38}$$

By introducing a slack variable $u \in \mathbb{R}^{m_I}$ and indicator function, one can always reformulate problem (1.37) as

$$\begin{array}{ll} \min & (\delta_{\mathcal{K}}^*(-Z) + \delta_{\mathbb{R}_+^{m_I}}(u)) + \frac{1}{2}\langle X', \mathcal{Q}X'\rangle + \delta_{(\mathcal{K}_+^n)^*}(S) - \langle b_E, y_E\rangle - \langle b_I, y_I\rangle \\ \text{s.t.} & Z - \mathcal{Q}X' + S + \mathcal{A}_E^* y_E + \mathcal{A}_I^* y_I = C, \\ & u - y_I = 0,\ X' \in \mathcal{S}^n, \end{array} \tag{1.39}$$

where $\delta_{\mathbb{R}_+^{m_I}}(\cdot)$ is the indicator function over $\mathbb{R}_+^{m_I}$, i.e., $\delta_{\mathbb{R}_+^{m_I}}(u) = 0$ if $u \in \mathbb{R}_+^{m_I}$ and $\delta_{\mathbb{R}_+^{m_I}}(u) = \infty$ otherwise. It is not difficult to verify that problem (1.39) fits

the following general convex composite quadratic optimization model:

$$\begin{array}{ll} \min & f(u) + \sum_{i=1}^{p} \theta_i(y_i) + g(v) + \sum_{j=1}^{q} \varphi_j(z_j) \\ \text{s.t.} & \mathcal{F}^* u + \sum_{i=1}^{p} \mathcal{A}_i^* y_i + \mathcal{G}^* v + \sum_{j=1}^{q} \mathcal{B}_j^* z_j = c, \end{array} \tag{1.40}$$

where p and q are given nonnegative integers, $f : \mathcal{U} \to (-\infty, +\infty]$, $g : \mathcal{V} \to (-\infty, +\infty]$ are closed proper convex functions, $\theta_i : \mathcal{Y}_i \to (-\infty, +\infty]$, $i = 1, 2, \ldots, p$, and $\varphi_j : \mathcal{Z}_j \to (-\infty, +\infty]$, $j = 1, 2, \ldots, q$ are convex quadratic functions, $\mathcal{F} : \mathcal{X} \to \mathcal{U}$, $\mathcal{G} : \mathcal{X} \to \mathcal{V}$, $\mathcal{A}_i : \mathcal{X} \to \mathcal{Y}_i$, $i = 1, 2, \ldots, p$, and $\mathcal{B}_j : \mathcal{X} \to \mathcal{Z}_j$, $j = 1, 2, \ldots, q$ are linear maps, $\mathcal{U}, \mathcal{V}, \mathcal{Y}_1, \ldots, \mathcal{Y}_p, \mathcal{Z}_1, \ldots, \mathcal{Z}_q$ and $\mathcal{X}$ are all real finite dimensional Euclidean spaces each equipped with an inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. Without loss of generality, we use the adjoint form of the linear maps in the constraint of model (1.40) since in general we are more interested in solving the dual problem of a matrix optimization model with linear constraints and the dual problem often involves the adjoint of linear maps. Model (1.40) covers a wide range of applications involving matrix completion such as EDM completion problem (1.18). We must point out that model (1.40) is a very general model and the dimension of the Euclidean spaces $\mathcal{U}, \mathcal{V}, \mathcal{Y}_1, \ldots, \mathcal{Y}_p, \mathcal{Z}_1, \ldots, \mathcal{Z}_q$ and $\mathcal{X}$ depend on the special structure of the models in different application areas. For example, in our case described in Chapter 3, we only need the 3-block version of the model (1.40) resulting in $p = 1$, $q = 0$, $\mathcal{U}, \mathcal{V}$ are the space of $n \times n$ symmetric matrices $\mathcal{S}^n$ and y_1 in the vector space $\mathcal{Y}_1$ with dimension depending on the context.

An efficient algorithm for (1.40) is essential to many large scale problems with large amount of equality and inequality constraints. In this section, we mainly review a Schur complement based semi-proximal ADMM proposed by Li et al. (2014), which is an efficient and convergent ADMM that provides a solution of low to medium accuracy to problem (1.40). We will integrate this algorithm in our sensor network localization scheme. Details can be found in Chapter 3.

To write problem (1.40) in a compact form, define $\mathcal{Y} := \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_p$, $\mathcal{Z} := \mathcal{Z}_1 \times \mathcal{Z}_2 \times \cdots \times \mathcal{Z}_q$. Denote $y \equiv (y_1, y_2, \ldots, y_p) \in \mathcal{Y}$ and $z \equiv (z_1, z_2, \ldots, z_q) \in \mathcal{Z}$. Define the linear map $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ such that its adjoint is given by

$$\mathcal{A}^* y = \sum_{i=1}^{p} \mathcal{A}_i^* y_i \quad \forall y \in \mathcal{Y},$$

where $\mathcal{A}_i^*$ is the adjoint of $\mathcal{A}_i$. For example, if $\mathcal{A}_i : \mathcal{S}^n \to \mathbb{R}^m$ is a linear operator, we call the adjoint of $\mathcal{A}_i$, the linear operator $\mathcal{A}_i^* : \mathbb{R}^m \to \mathcal{S}^n$ such that

$$\langle \mathcal{A}_i(X), y \rangle = \langle X, \mathcal{A}_i^* y \rangle, \quad \text{for all } X \in \mathcal{S}^n,\ y \in \mathbb{R}^m.$$

Similarly define the linear map $\mathcal{B} : \mathcal{X} \to \mathcal{Z}$ such that its adjoint is given by

$$\mathcal{B}^* z = \sum_{i=1}^{q} \mathcal{B}_j^* z_i \quad \forall z \in \mathcal{Z}.$$

Let $\theta(y) := \sum_{i=1}^{p} \theta_i(y_i)$, $y \in \mathcal{Y}$ and $\varphi(z) := \sum_{j=1}^{q} \varphi_j(z_j)$, $z \in \mathcal{Z}$. Then problem (1.40) is equivalent to the following form:

$$\begin{array}{ll} \min & f(u) + \theta(y) + g(v) + \varphi(z) \\ \text{s.t.} & \mathcal{F}^* u + \mathcal{A}^* y + \mathcal{G}^* v + \mathcal{B}^* z = c, \end{array} \tag{1.41}$$

which is a special case of the following block-separable convex optimization problem:

$$\min \left\{ \sum_{i=1}^{n} \phi_i(w_i) \mid \sum_{i=1}^{n} \mathcal{H}_i^* w_i = c \right\}, \tag{1.42}$$

where for each $i \in \{1, \ldots, n\}$, $\mathcal{W}_i$ is a finite dimensional real Euclidean space equipped with an inner product $\langle \cdot, \cdot \rangle$, $\phi_i : \mathcal{W}_i \to (-\infty, +\infty]$ is a closed proper convex function, $\mathcal{H}_i : \mathcal{X} \to \mathcal{W}_i$ is a linear map and $c \in \mathcal{X}$ is given. The augmented

Lagrangian function for (1.42) is

$$\mathcal{L}_\sigma(w_1, w_2, \ldots, w_n; x) = \sum_{i=1}^{n} \phi_i(w_i) + \langle x, \sum_{i=1}^{n} \mathcal{H}_i^* w_i - c\rangle + \frac{\sigma}{2}\| \sum_{i=1}^{n} \mathcal{H}_i^* w_i - c\|^2$$

for $w_i \in \mathcal{W}_i$, $i = 1, \ldots, n$, $x \in \mathcal{X}$ and $\sigma > 0$ being a given parameter. A direct extension of 2-block semi-proximal ADMM in Algorithm 5 has the following updating process:

$$\begin{aligned}
w_1^{k+1} &= \arg\min \mathcal{L}_\sigma(w_1, w_2^k, \ldots, w_n^k; x^k), \\
&\vdots \\
w_i^{k+1} &= \arg\min \mathcal{L}_\sigma(w_1^{k+1}, \ldots, w_{i-1}^{k+1}, w_i, w_{i+1}^k, \ldots, w_n^k; x^k), \\
&\vdots \\
w_n^{k+1} &= \arg\min \mathcal{L}_\sigma(w_1^{k+1}, \ldots, w_{n-1}^{k+1}, w_n; x^k), \\
x^{k+1} &= x^k + \tau\sigma\left(\sum_{i=1}^{n} \mathcal{H}_i^* w_i^{k+1} - c\right).
\end{aligned} \tag{1.43}$$

There has no clear results on the convergence of the n-block ADMM (1.43) above. Wen et al. (2010) designed an efficient software for solving some large scale SDP problems using the 3-block ADMM with $\tau = 1.618$. However, Chen et al. (2014) showed that the direct extension of the ADMM to the 3-block convex problem is not necessarily convergent. By thoroughly exploiting the special quadratic structure in the convex quadratic problem (1.41), Li et al. (2014) proposed a convergent Schur complement based semi-proximal ADMM called SCB-SPADMM with a competitive numerical efficiency to the direct extended ADMM (1.43).

In the following part of this section, we focus on the problem (1.40), with all θ_i and φ_j being assumed to be convex quadratic functions:

$$\begin{aligned}
\theta_i(y_i) &= \frac{1}{2}\langle y_i, \mathcal{P}_i y_i\rangle - \langle b_i, y_i\rangle, i = 1, \ldots, p, \\
\varphi_j(z_j) &= \frac{1}{2}\langle z_j, \mathcal{Q}_j z_j\rangle - \langle d_j, z_j\rangle, j = 1, \ldots, q,
\end{aligned}$$

where $\mathcal{P}_i$ and $\mathcal{Q}_j$ are given self-adjoint positive semidefinite linear operators. The dual problem of (1.40) can be written as

$$\begin{aligned} \min \quad & \langle c, x\rangle + f^*(s) + \textstyle\sum_{i=1}^{p} \theta_i^*(r_i) + g^*(t) + \sum_{j=1}^{q} \varphi_j^*(w_j) \\ \text{s.t.} \quad & \mathcal{F}x + s = 0, \quad \mathcal{A}_i x + r_i = 0, \quad i = 1, \ldots, p, \\ & \mathcal{G}x + t = 0, \quad \mathcal{B}_j x + w_i = 0, \quad j = 1, \ldots, q. \end{aligned} \tag{1.44}$$

Define

$$\mathcal{T}_{\theta_i} := \mathcal{E}_{\theta_i} - \sigma^{-1}\mathcal{P}_i - \mathcal{A}_i\mathcal{A}_i^* \succeq 0, \quad i = 1, \ldots, p, \tag{1.45}$$

where $\mathcal{E}_{\theta_i}$ is a self-adjoint positive definite linear operator on $\mathcal{Y}_i$ such that it is a majorization of $\sigma^{-1}\mathcal{P}_i + \mathcal{A}_i\mathcal{A}_i^*$, i.e.,

$$\mathcal{E}_{\theta_i} \succeq \sigma^{-1}\mathcal{P}_i + \mathcal{A}_i\mathcal{A}_i^*.$$

Similarly, denote

$$\mathcal{T}_{\varphi_j} := \mathcal{E}_{\varphi_j} - \sigma^{-1}\mathcal{Q}_j - \mathcal{B}_j\mathcal{B}_j^* \succeq 0, \quad j = 1, \ldots, q, \tag{1.46}$$

where $\mathcal{E}_{\varphi_j}$ is the majorization of $\sigma^{-1}\mathcal{Q}_j + \mathcal{B}_j\mathcal{B}_j^*$. For computational efficiency, $\mathcal{T}_{\theta_i}$ and $\mathcal{T}_{\varphi_j}$ need to be as small as possible for each i and j, respectively. $\mathcal{E}_{\theta_i}$ and $\mathcal{E}_{\varphi_j}$ need to be chosen such that the inverse can be computed at a moderate cost.

For notational convenience, we also define

$$y_{\le i} := (y_1, y_2, \ldots, y_i), \quad y_{\ge i} := (y_i, y_{i+1}, \ldots, y_p), \quad i = 0, \ldots, p+1$$

with the convention that $y_0 = y_{p+1} = y_{\leq 0} = y_{\geq p+1} = \emptyset$. For $i = 1, \ldots, p$, define the linear operator $\mathcal{A}_{\leq i} : \mathcal{X} \to \mathcal{Y}$ by

$$\begin{pmatrix} \mathcal{A}_1 x \\ \mathcal{A}_2 x \\ \vdots \\ \mathcal{A}_i x \end{pmatrix} \equiv \mathcal{A}_{\leq i} x := \mathcal{A}_1 x \times \mathcal{A}_2 x \times \cdots \times \mathcal{A}_i x \quad \forall x \in \mathcal{X}.$$

Similarly, we can define $z_{\leq j}$, $z_{\geq j}$ for $j = 0, \ldots, q+1$ and linear operator $\mathcal{B}_{\leq j}$ for $j = 1, \ldots, q$. To simplify the augmented Lagrangian function of (1.41), we need define the following affine function $\Gamma : \mathcal{U} \times \mathcal{Y} \times \mathcal{V} \times \mathcal{Z} \to \mathcal{X}$ by

$$\Gamma(u, y, v, z) := \mathcal{F}^* u + \mathcal{A}^* y + \mathcal{G}^* v + \mathcal{B}^* s - c \quad \forall (u, y, v, z) \in \mathcal{U} \times \mathcal{Y} \times \mathcal{V} \times \mathcal{Z}. \tag{1.47}$$

Then the augmented Lagrangian function of (1.41) is

$$\mathcal{L}_\sigma(u, y, v, z; x) = f(u) + \theta(y) + g(v) + \varphi(z) + \langle x, \Gamma(u, y, v, z)\rangle + \frac{\sigma}{2}\|\Gamma(u, y, v, z)\|^2. \tag{1.48}$$

By the definitions above, we can summarize the SCB-SPADMM in Algorithm 6.

Compared to the direct extension of 2-block semi-proximal ADMM, SCB-SPADMM requires two extra updating processes, which are step 2 and step 4 that solve the proximal augmented Lagrangian function backwards with respect to y and z. By doing this, the convergence of algorithm SCB-SPADMM is guaranteed. Genuinely, SCB-SPADMM firstly splits problem (1.41) into a 2-block framework with the (u, y)-block and (v, z)-block. For each block, the semi-proximal augmented Lagrangian method is applied to get the solution of the subproblem simultaneously. That is, for example, the step 2-3 are actually solving the problem

$$(u^{k+1}, y^{k+1}) = \arg\min_{u,y} \mathcal{L}_\sigma(u, y, v^k, z^k; x^k) + \frac{\sigma}{2}\|(u, y_{\leq p-1}) - (u^k, y^k_{\leq p-1})\|^2_{\hat{\mathcal{T}}_{f_p}} + \frac{\sigma}{2}\|y_p - y^k_p\|^2_{\mathcal{T}_{\theta_p}},$$

Algorithm 6 SCB-SPADMM

1: Set initial points $(u^0, y^0, v^0, z^0, x^0) \in \mathrm{dom} f \times \mathcal{Y} \times \mathrm{dom} g \times \mathcal{Z} \times \mathcal{X}$. Choose $\sigma > 0$ as the penalty parameter, $\tau \in (0, \infty)$ is the step length, and $\mathcal{T}_f$ and $\mathcal{T}_g$ are two self-adjoint positive semidefinite operators defined on $\mathcal{U}$ and $\mathcal{V}$ respectively. For $k = 0, 1, \ldots$, generate $(u^{k+1}, y^{k+1}, v^{k+1}, z^{k+1})$ and x^{k+1} according to the following iteration.

2: Compute for $i = p, \ldots, 1$,

$$\overline{y}_i^k = \arg\min_{y_i} \mathcal{L}_\sigma(u^k, (y_{\le i-1}^k, y_i, \overline{y}_{\ge i+1}^k), v^k, z^k; x^k) + \frac{\sigma}{2}\|y_i - y_i^k\|^2_{\mathcal{T}_{\theta_i}}, \tag{1.49}$$

where $\mathcal{T}_{\theta_i}$ is defined in (1.45). Then compute

$$u^{k+1} = \arg\min_{u} \mathcal{L}_\sigma(u, \overline{y}^k, v^k, z^k; x^k) + \frac{\sigma}{2}\|u - u^k\|^2_{\mathcal{T}_f}. \tag{1.50}$$

3: Compute for $i = 1, \ldots, p$,

$$y_i^{k+1} = \arg\min_{y_i} \mathcal{L}_\sigma(u^{k+1}, (y_{\le i-1}^{k+1}, y_i, \overline{y}_{\ge i+1}^k), v^k, z^k; x^k) + \frac{\sigma}{2}\|y_i - y_i^k\|^2_{\mathcal{T}_{\theta_i}}. \tag{1.51}$$

4: Compute for $j = q, \ldots, 1$,

$$\overline{z}_j^k = \arg\min_{z_j} \mathcal{L}_\sigma(u^{k+1}, y^{k+1}, v^k, (z_{\le j-1}^k, z_j, \overline{z}_{\ge j+1}^k); x^k) + \frac{\sigma}{2}\|z_j - z_j^k\|^2_{\mathcal{T}_{\varphi_j}}, \tag{1.52}$$

where $\mathcal{T}_{\varphi_i}$ is defined in (1.46). Then compute

$$v^{k+1} = \arg\min_{v} \mathcal{L}_\sigma(u^{k+1}, y^{k+1}, v, \overline{z}^k; x^k) + \frac{\sigma}{2}\|v - v^k\|^2_{\mathcal{T}_g}. \tag{1.53}$$

5: Compute for $j = 1, \ldots, q$,

$$z_i^{k+1} = \arg\min_{z_j} \mathcal{L}_\sigma(u^{k+1}, y^{k+1}, v^{k+1}, (z_{\le j-1}^{k+1}, z_j, \overline{z}_{\ge j+1}^k); x^k) + \frac{\sigma}{2}\|z_j - z_j^k\|^2_{\mathcal{T}_{\varphi_j}}. \tag{1.54}$$

6: Compute

$$x^{k+1} = x^k + \tau\sigma(\mathcal{F}^* u^{k+1} + \mathcal{A}^* y^{k+1} + \mathcal{G}^* v^{k+1} + \mathcal{B}^* z^{k+1} - c). \tag{1.55}$$

7: If a termination criterion is not met, go to step 2.

where $\hat{\mathcal{T}}_{f_1} := \mathcal{T}_f + \mathcal{F}_1\mathcal{A}_1^*\mathcal{E}_{\theta_1}^{-1}\mathcal{A}_1\mathcal{F}_1^*$,

$$\hat{\mathcal{T}}_{f_i} := \begin{pmatrix} \hat{\mathcal{T}}_{f_{i-1}} & \\ & \mathcal{T}_{\theta_{i-1}} \end{pmatrix} + \mathcal{F}_i\mathcal{A}_i^*\mathcal{E}_{\theta_i}^{-1}\mathcal{A}_i\mathcal{F}_i^*, \quad i = 2, \ldots, p. \tag{1.56}$$

Moreover, the existence of the proximal terms eliminates the quadratic terms containing linear transformations and makes it easier to use the function of projection onto cone to update in each step. We would like to point out that Algorithm 6 is to solve multi-block problem other than just 4-block problem, even though we use $\theta(y)$ and $\varphi(z)$ to denote the summation of θ_i and φ_j respectively, we treat each θ_i and φ_j as independent blocks.

Similar to the 2-block semi-proximal ADMM, SCB-SPADMM requires the following CQ.

Assumption 1.7. *There exists* $(\hat{u}, \hat{y}, \hat{v}, \hat{z}) \in ri(domf) \times \mathcal{Y} \times ri(domg) \times \mathcal{Z}$ *such that* $\mathcal{F}^*\hat{u} + \mathcal{A}^*\hat{y} + \mathcal{G}^*\hat{v} + \mathcal{B}^*\hat{z} = c$.

The convergence result is given in the following theorem, see more detail in Li et al. (2014).

Theorem 1.8. *Let* Σ_f *and* Σ_g *be the two self-adjoint and positive semidefinite operators defined by (1.26) and (1.27), respectively. Suppose that the solution set of problem (1.40) is nonempty and that Assumption 1.7 holds. Assume that* $\mathcal{T}_f$ *and* $\mathcal{T}_g$ *are chosen such that the the sequence* $\{(u^k, y^k, v^k, z^k, x^k)\}$ *generated by Algorithm SCB-SPADMM is well defined. Recall that* $\mathcal{T}_{\theta_i}$ *is defined in (1.45) for* $1 \le i \le p$ *and* $\mathcal{T}_{\varphi_j}$ *is defined in (1.46) for* $1 \le j \le q$. *Then, under the condition either (a)* $\tau \in (0, (1+\sqrt{5})/2)$ *or (b)* $\tau \ge (1+\sqrt{5})/2$ *but* $\sum_{k=0}^{\infty}(\|\mathcal{G}^*(v^{k+1} - v^k) + \mathcal{B}^*(z^{k+1} - z^k)\|^2 + \tau^{-1}\|\mathcal{F}^*u^{k+1} + \mathcal{A}^*y^{k+1} + \mathcal{G}^*v^{v+1} + \mathcal{B}^*z^{k+1} - c\|^2) < \infty$, *the following results hold:*

(i) *If* $(u^\infty, y^\infty, v^\infty, z^\infty, x^\infty)$ *is an accumulation point of* $\{(u^k, y^k, v^k, z^k, x^k)\}$, *then* $(u^\infty, y^\infty, v^\infty, z^\infty)$ *solves problem (1.41) and* x^∞ *solves (1.44), respectively.*

(ii) *If both* $\sigma^{-1}\Sigma_f + \mathcal{T}_f + \mathcal{F}\mathcal{F}^*$ *and* $\sigma^{-1}\Sigma_g + \mathcal{T}_g + \mathcal{G}\mathcal{G}^*$ *are positive definite, then the sequence* $\{(u^k, y^k, v^k, z^k, x^k)\}$, *which is automatically well defined, converges to a unique limit, say,* $(u^\infty, y^\infty, v^\infty, z^\infty, x^\infty)$ *with* $(u^\infty, y^\infty, v^\infty, z^\infty)$ *solving problem (1.41) and* x^∞ *solving (1.44), respectively.*

(iii) *When the* u, y*-part disappears, the corresponding results in parts (i)-(ii) hold under the condition either* $\tau \in (0, 2)$ *or* $\tau \geq 2$ *but* $\sum_{k=0}^{\infty} \|\mathcal{G}^* v^{k+1} + \mathcal{B}^* z^{k+1} - c\|^2 < \infty$.

SCB-SPADMM provides a potentially efficient approach to handle large scale and dense linear constraints. In this thesis, we use it for solving 3-blcok EDM-based optimization model with equality and inequality constraint in sensor network localization. We integrate the inequality constraints into a closed convex set constraint which allows us to use projection in the algorithm. Details can be found in Section 3.3. Together with other techniques in modelling and algorithm design, we propose a new framework for sensor network localization that achieves both robustness and efficiency.

Chapter 2

Best Euclidean distance embedding on a sphere

In this chapter, we mainly discuss a class of EDM-based optimization problem with spherical constraints for data representation on a sphere of unknown radius. This problem arises from various disciplines such as Statistic (spatial data representation), Psychology (constrained multidimensional scaling), and Computer Science (machine learning and pattern recognition). The best representation often needs to minimize a distance function of the data on a sphere as well as to satisfy some Euclidean distance constraints. As discussed in Section 1.1.4, those spherical and Euclidean distance constraints will present an enormous challenge to the existing algorithms. In this chapter, we introduce a reformulation of the problem as an EDM-based optimization problem with a low rank constraint. We then propose an iterative algorithm that uses a quadratically convergent Newton-CG method at its each step. We study fundamental issues including constraint nondegeneracy and the nonsingularity of generalized Jacobian that ensure the quadratic convergence of the Newton method. We use some classic examples from the spherical multidimensional scaling to demonstrate the flexibility of the algorithm in incorporating various constraints.

The section is organized as follows. In Section 2.1, we give a background and literature review for spherical data representation problem. In Section 2.2, We first argue that when the EDM is used to formulate the problem, it is necessary to introduce a new point to represent the center of the sphere. This is due to a special property arising from embedding an EDM. The algorithmic framework that we use for the obtained non-convex matrix optimization problem is closely related to the majorized penalty method of Gao and Sun (2010) for the nearest low-rank correlation matrix problem. One of the key elements in this type of method is that the subproblems are convex. Those convex problems are structurally similar to a convex relaxation of the original matrix optimization problem and they all can be solved by a quadratically convergent Newton-CG method. We establish that this is the case for our problem by studying the challenging issue of constraint nondegeneracy, which further ensures the nonsingularity of generalized Jacobian used by the Newton-CG method. Those results can be found in Section 2.3 and ensure that the extension of the majorization method of Gao and Sun (2010) to our problem is complete. The algorithm is presented in Section 2.4 and its key convergent results are stated without detailed proofs as they can be proved similarly as in Gao and Sun (2010). Section 2.5 aims to demonstrate a variety of applications from classical MDS to the circle fitting problem. The numerical performance is highly satisfactory with those applications.

2.1 Introduction to spherical data representation

The problem that we are mainly concerned with is placing n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in a best way on a sphere in $\mathbb{R}^r$. The primary information that we use is an incomplete/complete set of pairwise Euclidean distances (often with noises) among the n points. In such a setting, $\mathbb{R}^r$ is often a low-dimensional space (e.g., r takes

2 or 3 for data visualization) and is known as the embedding space. The center of the sphere is unknown. For some applications, the center can be put at origin in $\mathbb{R}^r$. Furthermore, the radius of the sphere is also unknown. In our matrix optimization formulation of the problem, we treat both the center and the radius as unknown variables. We develop a fast numerical method for this problem and present a few of interesting applications taken from existing literature.

The problem described above has long appeared in the constrained Multi-Dimensional Scaling (MDS) when $r \leq 3$, which is mainly for the purpose of data visualization, see Cox and Cox (2000, Sect. 4.6) and Borg and Groenen (2005, Sect. 10.3) for more details. In particular, it is known as the spherical MDS when $r = 3$ and the circular MDS when $r = 2$. Most numerical methods in this part took advantages of r being 2 or 3. For example, two of the earliest circular MDS were by Borg and Lingoes (1980) and Lee and Bentler (1980), where they introduced a new point $\mathbf{x}_0 \in \mathbb{R}^r$ as the center of the sphere (i.e., circles in their case) and further forced the following constraints to hold:

$$D_{01} = D_{02} = \cdots = D_{0n}.$$

Here $D_{0j} = \|\mathbf{x}_0 - \mathbf{x}_j\|$, $j = 1, \ldots, n$ are the Euclidean distances between the center $\mathbf{x}_0$ and the other n points. In their models, the variables are the coordinates of the $(n + 1)$ points in $\mathbb{R}^r$. In Borg and Lingoes (1980), the optimal criterion was a **stress** function widely used in MDS literature (see Borg and Groenen (2005, Chp. 3)), whereas Lee and Bentler (1980) used a least square loss function as its optimal criterion.

In the spherical MDS of Cox and Cox (1991), Cox and Cox placed the center of the sphere at origin and represented the n points by their spherical coordinates. Moreover, they also argued for the Euclidean distance to be used over the seemingly more appropriate geodesic distance on the sphere. This is particularly the

case when the order of the distances among the n points are more important than the magnitude of their actual distances. For the accurate relationship between Euclidean distance and the geodesic distance on a sphere, see Pękalska and Duin (2005, Thm. 3.23), which is credited to Schoenberg (1937). A recent method known as MDS on a quadratic surface (MDS-Q) was proposed by De Leeuw and Mair (2009), where geodesic distances were used. As noted in De Leeuw and Mair (2009, p. 12), "geodesic MDS-Q, however, seems limited for now to spheres in any dimension, with the possible exception of ellipses and parabolas in $\mathbb{R}^2$". For the spherical case, MDS-Q places the center at origin and the variables are the radius and the coordinates of the n points on the sphere. The Euclidean distances were then converted to the corresponding geodesic distances. The optimal criterion is a weighted least square loss function.

When the center of the sphere is placed at origin, any point on the sphere satisfies the spherical constraint of the type $\|\mathbf{x}\| = R$, where $\mathbf{x} \in \mathbb{R}^r$ and R is the radius. Optimization with spherical constraints has recently attracted much attention of researchers, see, e.g., Malick (2007); Ling et al. (2010); Gao (2010); Gao and Sun (2010); Li and Qi (2011); Zhou et al. (2012) and the references therein. Such a problem can be cast as a more general optimization problem over the Stiefel manifold (Wen and Yin, 2013; Jiang and Dai, 2014). One important example is the nearest low-rank correlation matrix problem, where the unit diagonals of the correlation matrix yields the spherical constraints (Gao and Sun, 2010; Li and Qi, 2011; Wen and Yin, 2013; Jiang and Dai, 2014). It is noted that the sequential second-order methods in Gao and Sun (2010); Li and Qi (2011) as well as the feasibility-preserving methods in Wen and Yin (2013); Jiang and Dai (2014) all rely on the fact that the radius is known (e.g., $R = 1$). This is in contrast to our problem where R is a variable.

2.2 EDM-based optimization formulation

The available information for us to find n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ embedded on a sphere in $\mathbb{R}^r$ is the set of approximate (squared) Euclidean distances among the n points:

$$D^0_{ij} \approx \|\mathbf{x}_i - \mathbf{x}_j\|^2, \qquad i, j = 1, \ldots, n.$$

Denote the center of the sphere by $\mathbf{x}_{n+1}$ (the $(n+1)$th point) and its radius by R. Since the n points are placed on the sphere, we must have

$$\|\mathbf{x}_j - \mathbf{x}_{n+1}\| = R, \qquad j = 1, \ldots, n.$$

Although we do not know the exact magnitude of R, we can be sure that twice the radius cannot be bigger than the diameter of the data set:

$$2R \le d_{\max} := \max_{i,j} \sqrt{D^0_{ij}}.$$

We therefore define the approximate distance matrix $D \in \mathcal{S}^{n+1}$ by (only upper part of D is defined)

$$D_{ij} = \begin{cases} \frac{1}{4} d^2_{\max} & i = 1, \ldots, n, \ j = n+1 \\ D^0_{ij} & i < j = 2, \ldots, n \\ 0 & i = j, \end{cases} \tag{2.1}$$

The elements in D are approximate Euclidean distances among the $(n+1)$ points $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}\}$. But D may not be a true EDM. Our purpose is to find the nearest EDM Y to D such that the embedding dimension of Y is r and its embedding

points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ are on a sphere centered at $\mathbf{x}_{n+1}$. The resulting matrix optimization model is then given by

$$\begin{array}{ll} \min_{Y \in \mathcal{S}^{n+1}} & \frac{1}{2}\|Y - D\|^2 \\ \text{s.t.} & Y \in \mathcal{S}_h^{n+1},\ -Y \in \mathcal{K}_+^{n+1},\ \text{rank}(JYJ) \le r \\ & Y_{1(n+1)} = Y_{j(n+1)}, \quad j = 2, \ldots, n. \end{array} \tag{2.2}$$

Once we find the nearest EDM $\overline{Y}$ from which the total deviation of D is the smallest, combined with the classical MDS Algorithm 1, we will get the positions of n embedding points.

Problem (2.2) is always feasible (e.g., the zero matrix is feasible). The feasible region is closed and the objective function is coercive. Let $\overline{Y}$ be its optimal solution. The first group of constraints in (2.2) implies that $\overline{Y}$ is an EDM with an embedding dimension not greater than r. If $r < n$ (i.e., $\text{rank}(JYJ) < n$), the problem is nonconvex. If $r = n$, then we can drop the rank constraint so that the problem is convex. This is due to the fact that any EDM of size $(n+1) \times (n+1)$ has an embedding dimension not greater than $(n + 1 - 1) = n$. One can easily check that 0 is always an eigenvalue of $J\overline{Y}J$ and e is the corresponding eigenvector. Therefore, the rank constraint is automatically satisfied if $r = n$. The second group of constraints in (2.2) means that the distances from $\mathbf{x}_i$, $i = 1, \ldots, n$ to $\mathbf{x}_{n+1}$ are equal. Hence, $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ lie on a sphere centered at $\mathbf{x}_{n+1}$. We call the constraints $Y_{1(n+1)} = Y_{j(n+1)}, j = 2, \ldots, n$ spherical constraints and we note that they are linear. This is in contrast to the nonlinear formulation of the spherical constraints in the previous studies (Borg and Lingoes, 1980; Lee and Bentler, 1980; Cox and Cox, 1991; De Leeuw and Mair, 2009).

Regarding to model (2.2), we have the following two remarks.

Remark 2.1. The idea of introducing a variable representing the center (i.e., one more dimension in our formulation) is similar to that of Borg and Lingoes (1980);

Lee and Bentler (1980), whose main purpose was for the case $r = 2$ and the variables of the optimization problems are the coordinates of the points concerned. Our model is more general for arbitrary r and is conducive to (second-order) algorithmic development because the spherical constraints are linear. Furthermore, as introduce in Section 1.1.3, the actual embedding is left out as a separate issue, which can be done by Algorithm 1, possibly through Procrustes analysis.

Remark 2.2. The following reasoning further justifies why it is necessary to introduce a new point for the center of the sphere. Let D^0 denote the true squared Euclidean distance matrix among n points on a sphere. From Gower (1982), the decomposition

$$-\frac{1}{2}JD^0J = X^TX \qquad \text{with} \quad X \in \mathbb{R}^{r\times n}, \tag{2.3}$$

would provide a set of points $\{\mathbf{x}_i : \ i = 1, \ldots, n\}$ such that the distances in D^0 are recovered through $D^0_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. In order for those points to lie on a sphere centered at origin, it is necessary and sufficient to enforce the constraints

$$\|\mathbf{x}_1\| = \|\mathbf{x}_2\| = \cdots = \|\mathbf{x}_n\|. \tag{2.4}$$

We note that

$$\begin{aligned} \|\mathbf{x}_i\|^2 &= e_i^T(X^TX)e_i = -\frac{1}{2}e_i^TJD^0Je_i \\ &= D^0_{ii} + \frac{1}{2n}\left(e_iD^0e + e^TD^0e_i\right) - \frac{e^TD^0e}{2n^2} \\ &= \frac{1}{2n}\langle D^0,\ A^i\rangle - \frac{e^TD^0e}{2n^2}, \end{aligned}$$

where $A^i := e_ie^T + ee_i^T$. The spherical constraints are then equivalent to

$$\langle D^0,\ A^1 - A^i\rangle = 0, \quad i = 2, \cdots, n,$$

which are linear in the Euclidean distance matrix D^0. It seems that there is no need to introduce a new point to represent the center of the sphere. However, there

is a potential conflict in this seemingly correct argument. We note that there is an implicit constraint we ignored. In (2.3), the embedding points in X have to satisfy the centralization condition (because of the projection matrix J)

$$Xe = 0. \tag{2.5}$$

A potential conflict is that the constraints (2.4) and (2.5) may be contradicting to each other. Such possible contradiction can be verified through the following example: Let D^0 be from the tree points on the unit circle centered at origin:

$$\mathbf{x}_1 = (1,0)^T, \qquad \mathbf{x}_2 = (-1,0)^T, \qquad \mathbf{x}_3 = (0,1)^T.$$

There exists no $X \in \mathbb{R}^{2\times 3}$ that satisfies (2.3) (hence (2.5)) and (2.4). Now we define D by (2.1) and solves problem (2.2), we obtain the following 4 embedding points:

$$\mathbf{z}_1 = (-1, 0.25)^T, \quad \mathbf{z}_2 = (1, 0.25)^T, \quad \mathbf{z}_3 = (0, -0.75)^T, \quad \mathbf{z}_4 = (0, 0.25)^T.$$

The first three points are on the unit circle centered at $\mathbf{z}_4$. The original three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ can be obtained through the simple shift $\mathbf{x}_i = \mathbf{z}_i - \mathbf{z}_4$ (the simplest Procrustes analysis). This example shows that it is necessary to introduce a new point to represent the center in order to remove the potential confliction in representing the spherical constraints as linear equations.

We now reformulate (2.2) in a more conventional format. By replacing Y by $(-Y)$ (in order to get rid of the minus sign before $\mathcal{K}_+^{n+1}$), we obtain

$$\begin{array}{ll} \min_{Y\in\mathcal{S}^{n+1}} & \frac{1}{2}\|Y + D\|^2 \\ \text{s.t.} & Y \in \mathcal{S}_h^{n+1},\ Y \in \mathcal{K}_+^{n+1},\ \operatorname{rank}(JYJ) \le r \\ & Y_{1(n+1)} = Y_{j(n+1)}, \quad j = 2, \ldots, n. \end{array}$$

Define three linear mappings $\mathcal{A}_1 : \mathcal{S}^{n+1} \to \mathbb{R}^{n+1}$, $\mathcal{A}_2 : \mathcal{S}^{n+1} \to \mathbb{R}^{n-1}$ and $\mathcal{A} : \mathcal{S}^{n+1} \to \mathbb{R}^{2n}$ respectively by

$$\mathcal{A}_1(Y) := \mathrm{diag}(Y),\ \mathcal{A}_2(Y) := \left(Y_{1(n+1)} - Y_{j(n+1)}\right)_{j=2}^{n} \text{ and } \mathcal{A}(Y) := \begin{pmatrix} \mathcal{A}_1(Y) \\ \mathcal{A}_2(Y) \end{pmatrix}.$$

It is therefore that solving (2.2) is equivalent to solving the following problem

$$\begin{array}{ll} \min_{Y \in \mathcal{S}^{n+1}} & \frac{1}{2}\|Y + D\|^2 \\ \text{s.t.} & \mathcal{A}(Y) = 0, \quad Y \in \mathcal{K}_+^{n+1} \\ & \mathrm{rank}(JYJ) \le r. \end{array} \tag{2.6}$$

We note that without the spherical constraints $\mathcal{A}_2(Y) = 0$, the problem reduces to the problem (1.21) studied in Qi and Yuan (2014). However, with the spherical constraints, the analysis in Qi and Yuan (2014), especially for the semismooth Newton-CG method developed in Qi (2013); Qi and Yuan (2014) is not valid any more because it heavily depends on the simple structure of the diagonal constraints $\mathcal{A}_1(Y) = 0$. One of our main tasks in this section is to develop more general analysis that covers the spherical constraints.

2.3 Convex relaxation

The framework of solving our problem (2.6) is based on the majorized penalty method of Gao and Sun (2010) which involves solving a sequence of convex relaxation of (2.6) without the rank constraint. So before we introduce the main framework, it is important to analyse how to solve the convex relaxation efficiently and accurately. The convex relaxation is obtained by dropping the rank constraint

from (2.6).

$$\begin{array}{ll} \min_{Y \in \mathcal{S}^{n+1}} & \frac{1}{2}\|Y + D\|^2 \\ \text{s.t.} & \mathcal{A}(Y) = 0, \quad Y \in \mathcal{K}_+^{n+1}. \end{array} \tag{2.7}$$

The convex relaxation is not only important on its own right but also plays a vital role in our algorithm because a sequence of such convex problems will be solved. This section has two parts. The first part is about two constraint qualifications that the convex relaxation may enjoy. The second part is about the semismooth Newton-CG method that solves the convex relaxation and it is proved to be quadratically convergent under the qualification of constraint nondegeneracy.

2.3.1 Constraint qualifications

Constraints qualifications are essential properties in deriving optimality conditions and effective algorithms for optimization problems, see, e.g., Bonnans and Shapiro (2013). We only study two of them, which are pertinent to our numerical method to be developed later on. The first is the generalized Slater condition and the second is constraint nondegeneracy.

It is easy to see that the linear equations in $\mathcal{A}(Y) = 0$ are linearly independent. Together with the characterization of EDM in (1.8), we further have

Proposition 2.3. *The generalized Slater condition hold for the convex relaxation (2.7). That is, there exists $Y \in \mathcal{S}^{n+1}$ such that*

$$\mathcal{A}(Y) = 0 \qquad \textit{and} \qquad Y \in \operatorname{int} \mathcal{K}_+^{n+1},$$

where $\operatorname{int} \mathcal{K}_+^{n+1}$ denotes the interior of $\mathcal{K}_+^{n+1}$.

Proof. Let $\mathbf{x}_i = (\sqrt{2}/2)(e_i - (1/(n+1))e)$, $i = 1, \ldots, n+1$, where e_i is the ith unit vector in $\mathbb{R}^{n+1}$. Define $Y \in \mathcal{S}^{n+1}$ by

$$Y_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2 = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

It follows from (1.12) that

$$-\frac{1}{2}JYJ = \begin{bmatrix} (\mathbf{x}_1)^T \\ \vdots \\ (\mathbf{x}_{n+1})^T \end{bmatrix} [\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}] \qquad \text{and} \qquad \operatorname{rank}(JYJ) = n.$$

Moreover, $(-Y) \in \mathcal{K}_+^{n+1}$. By formula (1.8), there exist $Z \in \mathcal{S}_+^n$, $z \in \mathbb{R}^n$ and $z_0 \in \mathbb{R}$ such that

$$-Y = Q \begin{bmatrix} Z & z \\ z^T & z_0 \end{bmatrix} Q.$$

By using the facts in (1.3), we obtain that

$$JYJ = Q \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} QYQ \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} Q = -Q \begin{bmatrix} Z & 0 \\ 0 & 0 \end{bmatrix} Q.$$

Since the rank of JYJ is n and $Z \in \mathcal{S}_+^n$, Z must be positive definite. This proves that $Y \in \operatorname{int} \mathcal{K}_+^{n+1}$. Apparently, $\mathcal{A}(Y) = 0$ by the definition of Y. Hence, the generalized Slater condition holds. □

The concept of constraint nondegeneracy was first studied by Robinson (1987, 2003) for abstract optimization problems and has been extensively used in Bonnans and Shapiro (2013) and Shapiro (2003) for sensitivity analysis in optimization and variational analysis. It plays a vital role in the characterizations of strong regularity (via Clark's generalized Jacobian) in nonlinear semidefinite programming

(SDP) by Sun (2006). For linear SDP, it reduces to the primal (dual) nondegeneracy of Alizadeh et al. (1997), see also Chan and Sun (2008) for further deep implications in SDP. It has been shown fundamental in many optimization problems, see Qi and Sun (2006); Miao et al. (2012); Qi (2013); Mian (2013); Laurent and Varvitsiotis (2014). Our main result is that constraint nondegeneracy holds for the convex problem (2.7) under a very weak condition and it further ensures that the Newton-CG method is quadratically convergent. For problem (2.7), constraint nondegeneracy is defined as follows (note that the problem has $2n$ linear constraints).

Definition 2.4. We say that constraint nondegeneracy holds at a feasible point $\overline{A}$ of (2.7) if

$$\mathcal{A}\left(\mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A}))\right) = \mathbb{R}^{2n}, \tag{2.8}$$

where $\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A})$ is the tangent cone of $\mathcal{K}_+^{n+1}$ at $\overline{A}$ and $\mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A}))$ is the largest subspace contained in $\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A})$.

Let $\overline{A} \in \mathcal{K}_+^{n+1}$ and denote

$$\overline{A} = Q \begin{bmatrix} Z & z \\ z^T & z_0 \end{bmatrix} Q, \qquad Z \in \mathcal{S}_+^n. \tag{2.9}$$

We assume that $\mathrm{rank}(Z) = r$ and let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r > 0$ be the r positive eigenvalues of Z in nonincreasing order. Let $\Lambda := \mathrm{Diag}(\lambda_1, \ldots, \lambda_r)$. We assume that Z takes the following spectral decomposition

$$Z = U \begin{bmatrix} \Lambda & \\ & 0 \end{bmatrix} U^T, \tag{2.10}$$

where $U \in \mathbb{R}^{n\times n}$ and $U^TU = I_n$. Let

$$\overline{U} := \begin{bmatrix} U & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{2.11}$$

Then $\overline{U}^T\overline{U} = I$. It has been depicted by Qi (2013, Eq. (24)) that

$$\mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^n}(\overline{A})) = \left\{ Q\overline{U} \begin{bmatrix} \begin{bmatrix} \Sigma_1 & \Sigma_{12} \\ \Sigma_{12}^T & 0 \end{bmatrix} & a \\ a^T & a_0 \end{bmatrix} \overline{U}^T Q : \begin{array}{l} \Sigma_1 \in \mathcal{S}^r \\ \Sigma_{12} \in \mathbb{R}^{r\times(n-r)} \\ a \in \mathbb{R}^n, a_0 \in \mathbb{R} \end{array} \right\} \tag{2.12}$$

which can be proved as follows. Since $\mathcal{K}_+^n$ is convex, the tangent cone $\mathcal{T}_{\mathcal{K}_+^n}(\overline{A})$ of $\mathcal{K}_+^n$ at $\overline{A} \in \mathcal{K}_+^n$ can be defined as the polar cone of $\mathcal{N}_{\mathcal{K}_+^n}(\overline{A})$:

$$\mathcal{T}_{\mathcal{K}_+^n}(\overline{A})) := (\mathcal{N}_{\mathcal{K}_+^n}(\overline{A}))^\circ, \tag{2.13}$$

where $\mathcal{N}_{\mathcal{K}_+^n}(\overline{A})$ is the normal cone of $\mathcal{K}_+^n$ at $\overline{A} \in \mathcal{K}_+^n$ defined by

$$\mathcal{N}_{\mathcal{K}_+^n}(\overline{A}) := \{X \in \mathcal{S}^n : \langle X, A - \overline{A}\rangle \le 0 \ \forall\ A \in \mathcal{K}_+^n\}.$$

The normal cone $\mathcal{N}_{\mathcal{K}_+^n}(\overline{A})$ is given by Glunt et al. (1990) that

$$\mathcal{N}_{\mathcal{K}_+^n}(\overline{A}) = \left\{ Q \begin{bmatrix} U \begin{bmatrix} 0 & 0 \\ 0 & M \end{bmatrix} U^T & 0 \\ 0 & 0 \end{bmatrix} Q : -M \in \mathcal{S}_+^{n-r} \right\}. \tag{2.14}$$

According to (2.13), the tangent cone can be written as

$$\mathcal{T}_{\mathcal{K}_+^n}(\overline{A}) = \left\{ Q\overline{U} \begin{bmatrix} \begin{bmatrix} \Sigma_1 & \Sigma_{12} \\ \Sigma_{12}^T & \Sigma_2 \end{bmatrix} & a \\ a^T & a_0 \end{bmatrix} \overline{U}^T Q : \begin{array}{l} \Sigma_1 \in \mathcal{S}^r, \Sigma_2 \in \mathcal{S}_+^{n-r} \\ \Sigma_{12} \in \mathbb{R}^{r\times(n-r)} \\ a \in \mathbb{R}^n, a_0 \in \mathbb{R} \end{array} \right\} \tag{2.15}$$

Then that the largest subspace contained in $\mathcal{T}_{\mathcal{K}_+^n}(\overline{A})$ is (2.12) is proved.

Consider matrix X of the following form:

$$X := Q\overline{U}\begin{bmatrix} \Gamma & -\Gamma q + \sqrt{n+1}a \\ (-\Gamma q + \sqrt{n+1}a)^T & q^T\Gamma q \end{bmatrix}\overline{U}^T Q, \tag{2.16}$$

where

$$\Gamma := \begin{bmatrix} \Sigma_1 & \Sigma_{12} \\ \Sigma_{12}^T & 0 \end{bmatrix} \in \mathcal{S}^n, \qquad q := U^T e, \qquad a \in \mathbb{R}^n. \tag{2.17}$$

Obviously, $X \in \mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^n}(\overline{A}))$. Define two linear mappings $\widetilde{\mathcal{A}}_i : \mathcal{S}^n \mapsto \mathbb{R}^n$ for $i = 1, 2$ respectively by

$$\widetilde{\mathcal{A}}_1(Y) = (Y_{11}, Y_{22}, \ldots, Y_{nn})^T$$

and

$$\widetilde{\mathcal{A}}_2(Y) = \left(Y_{1(n+1)} - Y_{2(n+1)}, \ldots, Y_{1(n+1)} - Y_{n(n+1)}, Y_{(n+1)(n+1)}\right)^T.$$

Note that different from $\mathcal{A}_1 : \mathcal{S}^{n+1} \to \mathbb{R}^{n+1}$ and $\mathcal{A}_2 : \mathcal{S}^{n+1} \to \mathbb{R}^{n-1}$, by rearranging the linear equations, $\widetilde{\mathcal{A}}_1$ and $\widetilde{\mathcal{A}}_2$ are both maps to $\mathbb{R}^n$.

By using the following two lemmas, we can prove the constraint nondegeneracy result in Proposition 2.7.

Lemma 2.5. *For any given $y \in \mathbb{R}^n$, there exists $a \in \mathbb{R}^n$, independent of the choice of Γ in X of (2.16), such that*

$$\widetilde{\mathcal{A}}_2(X) = y. \tag{2.18}$$

Proof. Simple calculation can verify that

$$Qe_{n+1} = -\frac{1}{\sqrt{n+1}}e \qquad \text{and} \qquad Q(e_1 - e_j) = e_1 - e_j \qquad \text{for } j = 1, \ldots, n.$$

We now calculate the elements of $\widetilde{\mathcal{A}}_2(X)$. For $j = 1, \ldots, n-1$, we have

$$\begin{aligned}
\left(\widetilde{\mathcal{A}}_2(X)\right)_j &= (e_1 - e_{j+1})^T Q\overline{U} \begin{bmatrix} \Gamma & -\Gamma q + \sqrt{n+1}a \\ (-\Gamma q + \sqrt{n+1}a)^T & q^T\Gamma q \end{bmatrix} \overline{U}^T Q e_{n+1} \\
&= (e_1 - e_{j+1})^T \overline{U} \begin{bmatrix} \Gamma & -\Gamma q + \sqrt{n+1}a \\ (-\Gamma q + \sqrt{n+1}a)^T & q^T\Gamma q \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{n+1}} q \\ -\frac{1}{\sqrt{n+1}} \end{bmatrix} \\
&= (e_1 - e_{j+1})^T \overline{U} \begin{bmatrix} -a \\ -q^T a \end{bmatrix} \\
&= (u^{j+1} - u^1)^T a, \qquad \text{(using (2.11))}
\end{aligned}$$

where u^j denotes the jth column of U^T. Similarly, we can calculate the last element of $\widetilde{\mathcal{A}}_2(X)$:

$$\begin{aligned}
\left(\widetilde{\mathcal{A}}_2(X)\right)_n &= X_{(n+1)(n+1)} \\
&= \frac{1}{\sqrt{n+1}} [q^T, 1] \begin{bmatrix} a \\ q^T a \end{bmatrix} \\
&= \frac{2}{\sqrt{n+1}} q^T a.
\end{aligned}$$

Then, equation (2.18) becomes the following simultaneous equations

$$\begin{cases} \langle u^{j+1} - u^1,\ a \rangle &= y_j,\ j = 1, \ldots, n-1 \\ \langle U^T e,\ a \rangle &= \frac{\sqrt{n+1}}{2} y_n. \end{cases} \tag{2.19}$$

It is easy to verify that the vectors $\{u^2 - u^1, \ldots, u^n - u^1, U^T e\}$ are linearly independent. Hence, there exists a unique solution $a \in \mathbb{R}^n$ to (2.19) for any given $y \in \mathbb{R}^n$. We also note that the solution of a is independent of Γ in X. □

Lemma 2.6. *Let $\overline{A}$ be decomposed as in (2.9). Suppose that there exists an eigenvector $u \in \mathbb{R}^n$ of Z corresponding to one of its positive eigenvalues such that*

$$\tau_i := u_i + \frac{1}{\sqrt{n+1}+1}\rho \neq 0 \qquad \forall\ i = 1, \ldots, n \qquad \text{with} \qquad \rho := \sum_{j=1}^{n} u_j. \quad (2.20)$$

Then for any given $z \in \mathbb{R}^n$ and $a \in \mathbb{R}^n$, there exists Γ of the type in (2.17) such that

$$\widetilde{\mathcal{A}}_1(X) = z, \quad (2.21)$$

where X is defined by (2.16).

Proof. Let $a \in \mathbb{R}^n$ and $z \in \mathbb{R}^n$ be given. Define

$$\overline{X} := X_1 + X_2 + X_3 + X_4,$$

with

$$X_1 = \begin{bmatrix} \Gamma & 0 \\ 0 & 0 \end{bmatrix},\ X_2 = \begin{bmatrix} 0 & -\Gamma q \\ -(\Gamma q)^T & 0 \end{bmatrix},\ X_3 = \begin{bmatrix} 0 & 0 \\ 0 & q^T\Gamma q \end{bmatrix},\ X_4 = \sqrt{n+1}\begin{bmatrix} 0 & a \\ a^T & 0 \end{bmatrix}.$$

We calculate the first n diagonal elements of X. For $i = 1, \ldots, n$, we have

$$\begin{aligned} X_{ii} &= e_i^T Q\overline{U}\begin{bmatrix} \Gamma & -\Gamma q + \sqrt{n+1}a \\ (-\Gamma q + \sqrt{n+1}a)^T & q^T\Gamma q \end{bmatrix}\overline{U}^T Q e_i \\ &= \left\langle \overline{U}^T Q e_i e_i^T Q\overline{U},\ \overline{X} \right\rangle = \langle W_i,\ \overline{X}\rangle, \end{aligned}$$

where $W_i := \overline{U}^T Q e_i e_i^T Q\overline{U}$. Then equation (2.21) becomes

$$\langle W_i,\ X_1 + X_2 + X_3\rangle = z_i - \langle W_i,\ X_4\rangle, \qquad i = 1, \ldots, n. \quad (2.22)$$

We would like to determine what Γ satisfies (2.22).

Note that for $i = 1, \ldots, n$,

$$Qe_i = e_i - \frac{1}{n+1+\sqrt{n+1}}v$$

and

$$\begin{aligned} W_i e_{n+1} &= \overline{U}^T Q e_i e_i^T Q \overline{U} e_{n+1} \\ &= \overline{U}^T Q e_i e_i^T Q e_{n+1} = -\frac{1}{\sqrt{n+1}} \overline{U}^T Q e_i \\ &= -\frac{1}{\sqrt{n+1}} \begin{bmatrix} U^T e_i \\ 0 \end{bmatrix} + \frac{1}{(n+1)(\sqrt{n+1}+1)} \begin{bmatrix} U^T e \\ 1+\sqrt{n+1} \end{bmatrix}. \end{aligned}$$

We derive the following identities

$$\begin{aligned} \langle W_i, X_1 \rangle &= \operatorname{Tr}\left([U^T, 0] Q e_i e_i Q^T \begin{bmatrix} U \\ 0 \end{bmatrix} \Gamma \right) \\ &= \left\langle U^T (e_i - \frac{1}{n+1+\sqrt{n+1}} e)(e_i - \frac{1}{n+1+\sqrt{n+1}} e)^T U,\ \Gamma \right\rangle. \end{aligned}$$

$$\begin{aligned} \langle W_i, X_2 \rangle &= -2 \left\langle W_i e_{n+1},\ [q^T \Gamma,\ 0]^T \right\rangle \\ &= \frac{2}{\sqrt{n+1}} (q^T \Gamma U^T e_i) - \frac{2}{(n+1)(\sqrt{n+1}+1)} (q^T \Gamma U^T e) \\ &= \frac{1}{\sqrt{n+1}} \left\langle U^T (e e_i^T + e_i e^T) U,\ \Gamma \right\rangle - \frac{2}{(n+1)(\sqrt{n+1}+1)} \left\langle U^T e e^T U,\ \Gamma \right\rangle. \end{aligned}$$

$$\langle W_i,\ X_3 \rangle = q^T \Gamma q (e_{n+1}^T Q e_i e_i^T Q e_{n+1}) = \frac{1}{n+1} q^T \Gamma q = \frac{1}{n+1} \langle U^T e e^T U,\ \Gamma \rangle.$$

The fact $q = U^T e$ was used above. We add together the identities above and simplify to get

$$\langle W_i,\ X_1 + X_2 + X_3 \rangle = \left\langle U^T \overline{W}_i U,\ \Gamma \right\rangle, \tag{2.23}$$

with

$$\overline{W}_i := \left(e_i + \frac{1}{\sqrt{n+1}+1}e\right)\left(e_i + \frac{1}{\sqrt{n+1}+1}e\right)^T.$$

Now we assume that condition (2.20) holds. Without loss of generality, we assume that u is the leading eigenvector of Z corresponding to the largest eigenvalue λ_1. Let $\gamma \in \mathbb{R}^n$ and define $\Gamma \in \mathcal{S}^n$ by

$$\Gamma_{ij} := \begin{cases} \gamma_1 & \text{if } i = j = 1 \\ \gamma_j/2 & \text{if } i = 1, j \geq 2 \\ \gamma_i/2 & \text{if } j = 1, i \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Such Γ is consistent with the structure in (2.17). It follows from (2.23) that

$$\langle W_i,\ X_1 + X_2 + X_3\rangle = \langle U^T\overline{W}_iUe_1,\ \gamma\rangle = \langle \overline{W}_iUe_1,\ U\gamma\rangle = \langle \overline{W}_iu,\ \overline{\gamma}\rangle,$$

where $\overline{\gamma} := U\gamma$ and the fact $u = Ue_1$ was used. Then the linear equations in (2.22) become

$$\langle \overline{W}_iu,\ \overline{\gamma}\rangle = z_i - \langle W_i,\ X_4\rangle, \qquad i = 1, \ldots, n \tag{2.24}$$

with $\overline{\gamma}$ being unknown. To ensure the existence of $\overline{\gamma}$ that satisfies (2.24), it is enough to prove the linear independence of the vectors $\{\overline{W}_iu\}_{i=1}^n$. Assume that there exist $\mu_i \in \mathbb{R}$, $i = 1 \ldots, n$ such that

$$\sum_{i=1}^n \mu_i\overline{W}_iu = 0,$$

which implies (by using the structure of $\overline{W}_i$)

$$\begin{cases} \mu_1\tau_1 + \frac{1}{\sqrt{n+1}+1}\sum_{i=1}^n \mu_i\tau_i &= 0 \\ \vdots & \vdots \ \ \vdots \\ \mu_n\tau_n + \frac{1}{\sqrt{n+1}+1}\sum_{i=1}^n \mu_i\tau_i &= 0. \end{cases}$$

We must have from the above equations that

$$\mu_1\tau_1 = \mu_2\tau_2 = \cdots = \mu_n\tau_n = 0.$$

Under the assumption of (2.20), we have $\mu_i = 0$ for $i = 1, \ldots, n$. Hence, the vectors $\{\overline{W}_i u\}_{i=1}^n$ are linearly independent. There is a unique $\overline{\gamma}$ satisfying (2.24). Therefore, $\gamma = U^T\overline{\gamma}$. Γ is well defined and the resulting X defined in (2.16) satisfies (2.21). This proves the result. □

Combining the two lemmas together gives our constraint nondegeneracy result.

Proposition 2.7. *Let $\overline{A}$ given by (2.9) be a feasible point of (2.7). Suppose condition (2.20) is satisfied for $\overline{A}$. Then constraint nondegeneracy holds at $\overline{A}$.*

Proof. By the definition of constraint nondegeneracy, it is sufficient to prove that for any given $x \in \mathbb{R}^{2n}$ there exists $X \in \mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A}))$ such that $\mathcal{A}(X) = x$. This is equivalent to existence of $X \in \mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A}))$ such that both (2.18) and (2.21) hold simultaneously for any given $y, z \in \mathbb{R}^n$. Lemmas 2.5 and 2.6 just ensured this is the case. We can choose $a \in \mathbb{R}^n$ first in Lemma 2.5 and then choose Γ in Lemma 2.6 to generate the matrix X of (2.16), which satisfies (2.18) and (2.21) simultaneously for any given $y, z \in \mathbb{R}^n$. Hence, constraint nondegeneracy holds at $\overline{A}$ under assumption (2.20). □

Proposition 2.7 means that the feasible points of (2.7) need to satisfy one more condition (2.20) to enjoy the constraint nondegeneracy property. For example, we know that $\overline{A} = 0$ is feasible and constraint nondegeneracy does not hold at 0

(can be verified directly through definition). Condition (2.20) serves the purpose of removing such points from the consideration.The following example shows that condition (2.20) holds everywhere but one point.

Example 2.1. *Consider the (squared) Euclidean distance matrix*

$$\overline{A} = \begin{bmatrix} 0 & 4 & 2(1-t) & 1 \\ 4 & 0 & 2(1+t) & 1 \\ 2(1-t) & 2(1+t) & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad -1 \le t \le 1.$$

It corresponds to a triangular embedding on a unit circle with the length of one edge equal the diameter of 2. *The remaining point of the triangle moves around the circle. Hence* $\operatorname{rank}(J\overline{A}J) = 2$ *(i.e.,* $r = 2$*). The corresponding matrix* Z *is*

$$Z = \frac{1}{18} \begin{bmatrix} 37-12t & -35 & -5+30t \\ -35 & 37+12t & -5-30t \\ -5+30t & -5-30t & 25 \end{bmatrix}.$$

It can be verified that condition (2.20) is satisfied for all t *except* $t = 0$. *When* $t = 0$, *the eigenvectors corresponding to the positive eigenvalues of the corresponding matrix* Z *are* $[-0.1925, -0.1925, 0.9623]^T$ *and* $[-0.7071, 0.7071, 0]^T$, *which all violate condition (2.20).*

2.3.2 Semismooth Newton method for convex relaxation

In this subsection, we develop the semismooth Newton method for the convex relaxation problem (2.7). The method is shown to be quadratically convergent under constraint nondegeneracy at the optimal solution. We will use this method to solve a sequence of subproblems that will appear in solving the nonconvex problem (2.6).

(a) Semismooth Newton Method. The Newton method is actually designed for the Lagrangian dual problem of (2.7), which is an unconstraint minimization problem.

For any given set Ω, $\delta_\Omega(\cdot)$ is the indicator function over Ω such that $\delta_\Omega(u) = 0$ if $u \in \Omega$ and ∞ otherwise. Then problem (2.7) is equivalent to

$$\begin{array}{ll} \min_{Y \in \mathcal{S}^{n+1}} & \frac{1}{2}\|Y + D\|^2 + \delta_{\mathcal{K}_+^{n+1}}(Y) \\ \text{s.t.} & \mathcal{A}(Y) = 0. \end{array} \tag{2.25}$$

The Lagrangian function of (2.25) is define as

$$L(Y; y) = \frac{1}{2}\|Y + D\|^2 + \delta_{\mathcal{K}_+^{n+1}}(Y) - \langle Y, \mathcal{A}^*(y)\rangle. \tag{2.26}$$

where $y \in \mathbb{R}^{2n}$ is the Lagrangian multiplier corresponding to the equality constraint. The Lagrangian dual problem is

$$\max_{y \in \mathbb{R}^{2n}} \left\{ \min_{Y \in \mathcal{S}^{n+1}} L(Y; y) \right\} \tag{2.27}$$

To solve the inner optimization problem, we need to use the following Moreau's theorem (Moreau, 1962).

Theorem 2.8. *(Moreau, 1962). Let $\mathcal{K}$ be a closed convex cone in a Hilbert space $(\mathcal{H}, \langle\cdot,\cdot\rangle)$, and $\mathcal{K}^\circ$ be its polar cone; that is the closed convex cone defined by $\mathcal{K}^\circ := \{a \in \mathcal{H} \mid \langle a, b\rangle \le 0, \forall b \in \mathcal{K}\}$. For $x, y, z \in \mathcal{H}$, the following statements are equivalent:*

1. *$z = x + y$, $x \in \mathcal{K}$, $y \in \mathcal{K}^\circ$, and $\langle x, y\rangle = 0$.*
2. *$x = \Pi_{\mathcal{K}}(z)$ and $y = \Pi_{\mathcal{K}^\circ}(z)$.*

Let

$$\begin{aligned}
\phi(y) &:= \min_{Y\in\mathcal{S}^{n+1}} L(Y;y) \\
&= \min_{Y\in\mathcal{S}^{n+1}} \frac{1}{2}\|Y+D\|^2 + \delta_{\mathcal{K}_+^{n+1}}(Y) - \langle Y, \mathcal{A}^*(y)\rangle \\
&= \min_{Y\in\mathcal{S}^{n+1}} \frac{1}{2}\|Y+D-\mathcal{A}^*(y)\|^2 + \delta_{\mathcal{K}_+^{n+1}}(Y) - \frac{1}{2}\|\mathcal{A}^*(y)\|^2 + \langle D, \mathcal{A}^*(y)\rangle \\
&= \frac{1}{2}\|\Pi_{\mathcal{K}_+^{n+1}}(\mathcal{A}^*(y)-D)+D-\mathcal{A}^*(y)\|^2 - \frac{1}{2}\|\mathcal{A}^*(y)\|^2 + \langle D, \mathcal{A}^*(y)\rangle \\
&= \frac{1}{2}\|\Pi_{\mathcal{K}_+^{n+1}}(\mathcal{A}^*(y)-D)+D-\mathcal{A}^*(y)\|^2 - \frac{1}{2}\|\mathcal{A}^*(y)-D\|^2 + \frac{1}{2}\|D\|^2,
\end{aligned}$$

by using Theorem 2.8, we have

$$\mathcal{A}^*(y) - D = \Pi_{\mathcal{K}_+^{n+1}}(\mathcal{A}^*(y)-D) + \Pi_{(\mathcal{K}_+^{n+1})^\circ}(\mathcal{A}^*(y)-D), \tag{2.28}$$

substitute (2.28) into the last equation of $\phi(y)$ we have

$$\begin{aligned}
\phi(y) &= \frac{1}{2}\|\Pi_{(\mathcal{K}_+^{n+1})^\circ}(\mathcal{A}^*(y)-D)\|^2 - \frac{1}{2}\|\Pi_{(\mathcal{K}_+^{n+1})^\circ}(\mathcal{A}^*(y)-D)\|^2 \\
&\quad -\frac{1}{2}\|\Pi_{\mathcal{K}_+^{n+1}}(\mathcal{A}^*(y)-D)\|^2 + \frac{1}{2}\|D\|^2 \\
&= -\frac{1}{2}\|\Pi_{\mathcal{K}_+^{n+1}}(\mathcal{A}^*(y)-D)\|^2 + \frac{1}{2}\|D\|^2.
\end{aligned}$$

Then the Lagrangian dual problem of (2.7) (in the form of minimization) is

$$\min_{y\in\mathbb{R}^{2n}} \ \theta(y) := \frac{1}{2}\|\Pi_{\mathcal{K}_+^{n+1}}(-D+\mathcal{A}^*(y))\|^2 - \frac{1}{2}\|D\|^2, \tag{2.29}$$

where $\mathcal{A}^* : \mathbb{R}^{2n} \mapsto \mathcal{S}^{n+1}$ is the adjoint operator of $\mathcal{A}$.

As discussed in Section 2.3.1, the linear transformations in $\mathcal{A}$ are linearly independent and that the generalized Slater condition holds for problem (2.7). It follows from the general results (Gao, 2010, Prop. 2.20, Prop. 4.11) that the dual function $\theta(\cdot)$ is coercive (i.e., $\theta(y) \to \infty$ as $\|y\| \to \infty$). Furthermore, because $\mathcal{K}_+^{n+1}$ is a closed and convex cone, $\theta(\cdot)$ is convex and continuously differentiable

(see Hiriart-Urruty and Lemaréchal (2013, Chp. IV, Example 2.1.4)). Therefore, the dual problem (2.29) must admit an optimal solution. To obtain the first-order optimality condition, we need to calculate the gradient of function $\theta(y)$. Let $\eta(y) = -D + \mathcal{A}^*(y)$, we temporarily treat $\theta(y)$ as a function of $\eta(y)$ and have

$$\theta(y) = \frac{1}{2}\|\Pi_{\mathcal{K}_+^{n+1}}(\eta(y))\|^2 - \frac{1}{2}\|D\|^2.$$

According to Theorem 2.8, we know

$$\begin{aligned}
\theta(y) &= \frac{1}{2}\|\eta(y) - \Pi_{(\mathcal{K}_+^{n+1})^\circ}(\eta(y))\|^2 - \frac{1}{2}\|D\|^2 \\
&= \min_{Z\in(\mathcal{K}_+^{n+1})^\circ}\{\frac{1}{2}\|\eta(y) - Z\|^2 - \frac{1}{2}\|D\|^2\} \\
&= \min_{Z\in\mathcal{S}^n}\{\delta_{(\mathcal{K}_+^{n+1})^\circ}(Z) + \frac{1}{2}\|\eta(y) - Z\|^2 - \frac{1}{2}\|D\|^2\},
\end{aligned}$$

where $\delta_{(\mathcal{K}_+^{n+1})^\circ}(\cdot)$ is the indicator function over $(\mathcal{K}_+^{n+1})^\circ$ such that $\delta_{(\mathcal{K}_+^{n+1})^\circ}(u) = 0$ if $u \in (\mathcal{K}_+^{n+1})^\circ$ and ∞ otherwise. Thus, $\theta(y)$ can be interpreted as the Moreau-Yosida regularization of $\delta_{(\mathcal{K}_+^{n+1})^\circ}(\cdot)$. Directly from the Theorem 4.1.4 by Hiriart-Urruty and Lemaréchal (1993), we have

$$\nabla_{\eta(y)}\theta(y) = \eta(y) - \Pi_{(\mathcal{K}_+^{n+1})^\circ}(\eta(y)) = \Pi_{\mathcal{K}_+^{n+1}}(\eta(y)). \tag{2.30}$$

Thus by (2.30) and the chain rule for composition functions, the first-order optimality condition is

$$F(y) := \nabla_y\theta(y) = \mathcal{A}\left(\Pi_{\mathcal{K}_+^{n+1}}(-D + \mathcal{A}^*(y))\right) = 0. \tag{2.31}$$

It follows from the introduction of semismooth Newton's method in Section 1.2 that $F(y)$ is *strongly semismooth* because it is a composition of linear mappings and $\Pi_{\mathcal{S}_+^{n+1}}(\cdot)$ (due to the relationship between $\Pi_{\mathcal{S}_+^{n+1}}(\cdot)$ and $\Pi_{\mathcal{K}_+^{n+1}}(\cdot)$ in (1.5)), which is known to be strongly semismooth. Then the main step of semismooth

Newton's method summarized in Algorithm 4 is that given $y^0 \in \mathbb{R}^{2n}$ and let $k := 0$, compute $V_k \in \partial F(y^k)$ and

$$y^{k+1} = y^k - V_k^{-1} F(y^k), \qquad k = 0, 1, 2, \ldots. \tag{2.32}$$

Since F is the gradient of θ and is nondifferentiable, ∂F is often called the generalized Hessian of θ, and $\theta(y)$ belongs to the class of all $C^{1,1}$ functions, that is, the class of all functions which are continuously differentiable and whose gradient mapping is locally Lipschitz, see (Hiriart-Urruty et al., 1984).

According to Theorem 1.4 for the convergence result of semismooth Newton's method, a key condition for it to be quadratically convergent is that the generalized Jacobian $\partial F(y^*)$ is nonsingular, where y^* denotes the optimal solution of (2.29). The optimal solution Y^* for the original convex problem (2.7) can be computed by

$$Y^* = \Pi_{\mathcal{K}_+^{n+1}}(-D + \mathcal{A}^*(y^*)). \tag{2.33}$$

To practically implement the semismooth Newton method (2.32), we have to address two key issues. One is how to compute a particular matrix $V \in \partial F(y)$. This has led us to use $\widehat{\partial} F(y)$ instead (to be developed in part (c) in this subsection and also see (2.44)). The other issue is how to solve the linear equation in (2.32). Direct evaluation of V would need $O(n^4)$ flops and hence direct methods are very expensive. We choose to use the well-developed conjugate gradient (CG) method to solve the Newton equation (see Qi and Sun (2006) and Zhao et al. (2010)). This results in the Newton-CG method that does not need to explicitly form the matrix V. The main task below is to show that the nonsingularity of $\widehat{\partial} F(y^*)$ (hence of $\partial F(y)$) under constraint nondegeneracy at Y^*.

(b) Characterization of Constraint Nondegeneracy. Let $\overline{A} \in \mathcal{K}_+^{n+1}$ be decomposed as in (2.9) and let $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_r > 0$ be the positive eigenvalues of Z in nonincreasing order. Let $\alpha := \{1, 2, \ldots, r\}$. We have the following characterization of constraint nondegeneracy at $\overline{A}$.

Lemma 2.9. *Let $h \in \mathbb{R}^{2n}$ be given. Denote*

$$\underline{H} = \begin{bmatrix} \underline{H}_1 & \underline{h} \\ \underline{h}^T & \underline{h}_0 \end{bmatrix} := Q(\mathcal{A}^*(h))Q \qquad \text{with } \underline{H}_1 \in \mathcal{S}^n,\ \underline{h} \in \mathbb{R}^n \text{ and } \underline{h}_0 \in \mathbb{R}. \quad (2.34)$$

Let $\overline{A} \in \mathcal{K}_+^{n+1}$ be decomposed as in (2.9) and the resulting Z has the spectral decomposition (2.10). Constraint nondegeneracy holds at $\overline{A}$ if and only if the following implication holds

$$\left.\begin{aligned} U_\alpha^T \underline{H}_1 &= 0 \\ \underline{h} &= 0 \\ \underline{h}_0 &= 0 \end{aligned}\right\} \quad \Longrightarrow \quad h = 0. \qquad (2.35)$$

Proof. By (2.8), constraint nondegeneracy holds at $\overline{A}$ if and only if

$$h \in \left\{ \mathcal{A}\left(\mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A})) \right) \right\}^\perp \quad \Longrightarrow \quad h = 0. \qquad (2.36)$$

It follows from (2.12) that

$$\left\{ QBQ :\ B \in \mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A})) \right\} = \left\{ \begin{bmatrix} U \begin{bmatrix} \Sigma_1 & \Sigma_{12} \\ \Sigma_{12}^T & 0 \end{bmatrix} U^T & a \\ a^T & a_0 \end{bmatrix} : \begin{array}{l} \Sigma_1 \in \mathcal{S}^r \\ \Sigma_{12} \in \mathbb{R}^{r\times(n-r)} \\ a \in \mathbb{R}^n, a_0 \in \mathbb{R} \end{array} \right\}.$$

The left-hand side of (2.36) is equivalent to, for any $B \in \mathrm{lin}(\mathcal{T}_{\mathcal{K}_+^{n+1}}(\overline{A}))$,

$$\begin{aligned} 0 &= \langle h, \mathcal{A}(B)\rangle = \langle \mathcal{A}^*(h), B\rangle \\ &= \langle Q\mathcal{A}^*(h)Q,\ QBQ\rangle \qquad\qquad (\text{because } Q^2 = I) \\ &= 2\langle \underline{h}, a\rangle + \underline{h}_0 a_0 + \mathrm{Tr}\left(U^T \underline{H}_1 U \begin{bmatrix} \Sigma_1 & \Sigma_{12} \\ \Sigma_{12}^T & 0 \end{bmatrix} \right). \end{aligned}$$

The above identities are for any $a \in \mathbb{R}^n$, $a_0 \in \mathbb{R}$, $\Sigma_1 \in \mathcal{S}^r$ and $\Sigma_{12} \in \mathbb{R}^{r\times(n-r)}$. Hence, we must have (recall $\alpha = \{1, 2, \ldots, r\}$)

$$\underline{h} = 0, \quad \underline{h}_0 = 0 \quad \text{and} \quad U_\alpha^T \underline{H}_1 U = 0.$$

Because of the nonsingularity of U, the above condition is equivalent to

$$\underline{h} = 0, \quad \underline{h}_0 = 0 \quad \text{and} \quad U_\alpha^T \underline{H}_1 = 0.$$

Therefore, (2.36) holds if and only if (2.35) holds. □

(c) Structure of $\widehat{\partial} F(y)$. For a given $y \in \mathbb{R}^{2n}$, we let

$$Y := -J(-D + \mathcal{A}^*(y))J \qquad \text{and} \qquad \overline{A} := \Pi_{\mathcal{K}_+^{n+1}}(-D + \mathcal{A}^*(y)).$$

Denote

$$\begin{bmatrix} \underline{Z} & \underline{z} \\ \underline{z}^T & \underline{z}_0 \end{bmatrix} := -Q(-D + \mathcal{A}^*(y))Q \qquad \text{with } \ \underline{Z} \in \mathcal{S}^n,\ \underline{z} \in \mathbb{R}^n,\ \underline{z}_0 \in \mathbb{R}.$$

We then have from (1.3) that

$$Y = Q \begin{bmatrix} \underline{Z} & 0 \\ 0 & 0 \end{bmatrix} Q \quad \text{and} \quad \Pi_{\mathcal{S}_+^{n+1}}(Y) = Q \begin{bmatrix} \Pi_{\mathcal{S}_+^n}(\underline{Z}) & 0 \\ 0 & 0 \end{bmatrix} Q. \tag{2.37}$$

We further have

$$\begin{aligned} Q\overline{A}Q &= Q\Pi_{\mathcal{K}_+^{n+1}}(-D + \mathcal{A}^*(y))Q \\ &= Q(-D + \mathcal{A}^*(y))Q + Q\Pi_{\mathcal{S}_+^{n+1}}(Y)Q \qquad \text{(by (1.5))} \\ &= -\begin{bmatrix} \underline{Z} & \underline{z} \\ \underline{z}^T & \underline{z}_0 \end{bmatrix} + \begin{bmatrix} \Pi_{\mathcal{S}_+^n}(\underline{Z}) & 0 \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \Pi_{\mathcal{S}_+^n}(-\underline{Z}) & -\underline{z} \\ -\underline{z}^T & -\underline{z}_0 \end{bmatrix}. \end{aligned}$$

We write $\overline{A}$ as in (2.9). It follows that

$$Z = \Pi_{\mathcal{S}_+^n}(-\underline{Z}), \quad z = -\underline{z} \quad \text{and} \quad z_0 = -\underline{z}_0. \tag{2.38}$$

Let $\underline{Z}$ admit the following spectral decomposition

$$\underline{Z} = W\underline{\Lambda}W^T,$$

with $\underline{\Lambda} := \mathrm{Diag}(\underline{\lambda}_1, \ldots, \underline{\lambda}_n)$ and $\underline{\lambda}_1 \geq \ldots \geq \underline{\lambda}_n$ being the eigenvalues of $\underline{Z}$ and $WW^T = I_n$.

Define

$$\underline{\alpha} := \{i : \underline{\lambda}_i > 0\}, \quad \underline{\beta} := \{i : \underline{\lambda}_i = 0\} \quad \text{and} \quad \underline{\gamma} := \{i : \underline{\lambda}_i < 0\}.$$

The relationship between Y and $\underline{Z}$ in (2.37) means that $\{\underline{\lambda}_1, \ldots, \underline{\lambda}_n\}$ are the

eigenvalues of Y. Moreover, Y has just one more eigenvalue, which is *zero*, than $\underline{Z}$. For those eigenvalues, define the corresponding symmetric matrix $\Omega \in \mathcal{S}^n$ with entries

$$\Omega_{ij} := \frac{\max\{\underline{\lambda}_i, 0\} + \max\{\underline{\lambda}_j, 0\}}{|\underline{\lambda}_i| + |\underline{\lambda}_j|}, \qquad i, j = 1, \ldots, n \tag{2.39}$$

where $0/0$ is defined to be 1. Let

$$W = [W_{\underline{\alpha}},\ W_{\underline{\beta}},\ W_{\underline{\gamma}}] \quad \text{and} \quad \overline{W} := \begin{bmatrix} W_{\underline{\alpha}} & W_{\underline{\beta}} & 0 & W_{\underline{\gamma}} \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.40}$$

It follows from (2.31) and (1.5) that

$$F(y) = \mathcal{A}(-D + \mathcal{A}^*(y)) + \mathcal{A}\left(\Pi_{\mathcal{S}^{n+1}_+}(-J(-D + \mathcal{A}^*(y))J)\right).$$

The key part in $F(y)$ is the composite function between $\Pi_{\mathcal{S}^{n+1}_+}(\cdot)$ and the linear operator $J\mathcal{A}^*(\cdot)J$. Because of this feature, it is hard to express $\partial F(y)$ exactly. We therefore define the following alternative:

$$\widehat{\partial} F(y)(\cdot) := \mathcal{A}\mathcal{A}^*(\cdot) - \mathcal{A}\left(\partial \Pi_{\mathcal{S}^{n+1}_+}(Y)(J\mathcal{A}^*(\cdot)J)\right). \tag{2.41}$$

Although we do not know whether $\partial F(y)$ is contained in $\widehat{\partial} F(y)$, their images of vectors coincide:

$$\partial F(y)h = \widehat{\partial} F(y)h, \qquad \forall\ h \in \mathbb{R}^{2n} \tag{2.42}$$

which implies that if all elements in $\widehat{\partial} F(y)$ are positive definite, so are those in $\partial F(y)$. Thus, by analysing the elements in $\widehat{\partial} F(y)$ using (2.41), we can proof the positive definiteness of the elements in $\partial F(y)$ to proof the convergent result of Newton's method. This is the motivation of (2.41). To explain why the relationship holds in (2.42), we need the following corollary.

Corollary 2.10. *(Clarke, 1990, p75). Let $H : \mathbb{R}^n \to \mathbb{R}^m$ be Lipschitz near x and $G : \mathbb{R}^m \to \mathbb{R}^k$ be Lipschitz near $H(x)$. Then, for any v in $\mathbb{R}^n$, one has*

$$\partial(G \circ H)(x)v \subset co\{\partial G(H(x))\partial H(x)v\}.$$

If G is continuously differentiable near $H(x)$, then equality holds (and co is superfluous).

$F(\cdot)$ is the gradient of a convex function and the projection on positive semidefinite cone is Lipschitz continuous. Thus, Corollary 2.10 implies the relationship in (2.42) holds for all $h \in \mathbb{R}^{2n}$, see also (Hiriart-Urruty et al., 1984).

The valuable benefit in using $\widehat{\partial} F(y)$ is that the set can be completely characterized because the generalized Jacobian $\partial \Pi_{\mathcal{S}_+^{n+1}}(\cdot)$ has a full characterization, take (1.25) as an example, also see Sun (2006, Prop. 2.2). We describe $\widehat{\partial} F(y)$ in the following result by making use of Sun (2006, Prop. 2.2). Its proof can be patterned after those in Sect. 3.1 to Sect. 3.3 of Qi (2013) for Qi (2013, Prop. 3.2). We note that only the submatrix $\Omega_{\underline{\alpha}\underline{\gamma}}$ is used in the description.

Proposition 2.11. *For every matrix $M \in \widehat{\partial} F(y)$, there exists $\widetilde{V} \in \partial \Pi_{\mathcal{S}_+^{|\underline{\beta}|+1}}(0)$ such that*

$$Mh = \mathcal{A}(\mathcal{A}^*(h)) - \mathcal{A}(P\mathcal{W}_h P^T), \qquad \forall\ h \in \mathbb{R}^{2n} \tag{2.43}$$

where $P := Q\overline{W}$,

$$\mathcal{W}_h := \begin{bmatrix} W_{\underline{\alpha}}^T \underline{H}_1 W_{\underline{\alpha}} & \begin{bmatrix} W_{\underline{\alpha}}^T \underline{H}_1 W_{\underline{\beta}} & 0 \end{bmatrix} & \Omega_{\underline{\alpha}\underline{\gamma}} \circ W_{\underline{\alpha}}^T \underline{H}_1 W_{\underline{\gamma}} \\ \begin{bmatrix} W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\alpha}} \\ 0 \end{bmatrix} & \widetilde{V}\left(\begin{bmatrix} W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\beta}} & 0 \\ 0 & 0 \end{bmatrix}\right) & 0 \\ \Omega_{\underline{\alpha}\underline{\gamma}}^T \circ W_{\underline{\gamma}}^T \underline{H}_1 W_{\underline{\alpha}} & 0 & 0 \end{bmatrix}$$

and $\underline{H}_1$ is from the partition in (2.34).

An implementable version of the semismooth Newton method (2.32) takes the following form

$$y^{k+1} = y^k - M_k^{-1} F(y^k), \qquad M_k \in \widehat{\partial} F(y^k), \quad k = 0, 1, 2, \ldots. \tag{2.44}$$

To implement the above method, we need to choose an explicit element $M_k \in \widehat{\partial} F(y^k)$. The matrix M (subscript k is omitted) used in our implementation is given by (2.43) with $\widetilde{V} = 0$. This can be proved by using Proposition 2.11 and Pang et al. (2003, Lemma 11).

(d) Nonsingularity of $\widehat{\partial} F(y)$. Recall the matrices $\underline{H}$ and $\overline{W}$ are respectively defined in (2.34) and (2.40). It is easy to verify that

$$\overline{W}^T \underline{H} \overline{W} = \begin{bmatrix} W_{\underline{\alpha}}^T \underline{H}_1 W_{\underline{\alpha}} & W_{\underline{\alpha}}^T \underline{H}_1 W_{\underline{\beta}} & W_{\underline{\alpha}}^T \underline{h} & W_{\underline{\alpha}}^T \underline{H}_1 W_{\underline{\gamma}} \\ W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\alpha}} & W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\beta}} & W_{\underline{\beta}}^T \underline{h} & W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\gamma}} \\ \underline{h}^T W_{\underline{\alpha}} & \underline{h}^T W_{\underline{\beta}} & \underline{h}_0 & \underline{h}^T W_{\underline{\gamma}} \\ W_{\underline{\gamma}}^T \underline{H}_1 W_{\underline{\alpha}} & W_{\underline{\gamma}}^T \underline{H}_1 W_{\underline{\beta}} & W_{\underline{\gamma}}^T \underline{h} & W_{\underline{\gamma}}^T \underline{H}_1 W_{\underline{\gamma}} \end{bmatrix}. \tag{2.45}$$

We further denote

$$G_1 := \begin{bmatrix} W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\beta}} & W_{\underline{\beta}}^T \underline{h} \\ \underline{h}^T W_{\underline{\beta}} & \underline{h}_0 \end{bmatrix}, \qquad G_2 := \begin{bmatrix} W_{\underline{\beta}}^T \underline{H}_1 W_{\underline{\beta}} & 0 \\ 0 & 0 \end{bmatrix}.$$

It is easy to see that $\|G_2\| \le \|G_1\|$ and

$$(\|G_1\| + \|G_2\|)(\|G_1\| - \|G_2\|) = \|G_1\|^2 - \|G_2\|^2 = 2\|W_{\underline{\beta}}^T \underline{h}\|^2 + \underline{h}_0^2.$$

Hence we have

$$\|G_1\|(\|G_1\| - \|G_2\|) \ge \|W_{\underline{\beta}}^T \underline{h}\|^2 + \frac{1}{2}\underline{h}_0^2. \tag{2.46}$$

It follows from Chan and Sun (2008, Eq (17)) that

$$\langle Z_1,\ \widetilde{V}(Z_2)\rangle \le \|Z_1\|\|Z_2\| \qquad \forall\ \widetilde{V} \in \partial\Pi_{\mathcal{S}_+^{|\underline{\beta}|+1|}}(0),\quad Z_1,\ Z_2 \in \mathcal{S}^{|\underline{\beta}|+1}. \tag{2.47}$$

From (2.38), the positive eigenvalues of Z are just the opposite of those negative eigenvalues in $\underline{\gamma}$. Let Γ be the permutation matrix which maps the sequence $\{1, 2, \ldots, |\underline{\gamma}|\}$ to its reverse order. We have,

$$Z = W_{\underline{\gamma}}(-\Lambda_{\underline{\gamma}})W_{\underline{\gamma}} = (W_{\underline{\gamma}}\Gamma)(\Gamma(-\Lambda_{\underline{\gamma}})\Gamma)(W_{\underline{\gamma}}\Gamma)^T.$$

Hence, U_α, which consists of the eigenvectors of positive eigenvalues in the spectral decomposition (2.10), can be chosen to be

$$U_\alpha = W_{\underline{\gamma}}\Gamma. \tag{2.48}$$

Theorem 2.12. *Let y be the optimal solution of the dual problem (2.29). Let $\overline{A} := \Pi_{\mathcal{K}_+^{n+1}}(-D + \mathcal{A}^*(y))$. We assume that constraint nondegeneracy holds at $\overline{A}$. Then every matrix $M \in \widehat{\partial}F(y)$ is positive definite.*

Proof. We continue to use the notation developed so far. Let $M \in \widehat{\partial}F(y)$. Our purpose is to prove $\langle h,\ Mh\rangle > 0$ for all $0 \ne h \in \mathbb{R}^{2n}$. It follows from (2.43)

$$Mh = \mathcal{A}(\mathcal{A}^*(h)) - \mathcal{A}\Big(P\mathcal{W}_h P^T\Big),$$

where $\mathcal{W}_h$ is given in Proposition 2.11.

We now calculate $\langle h,\ Mh\rangle$.

$$\begin{aligned}
\langle h,\ Mh\rangle &= \|\mathcal{A}^*(h)\|^2 - \langle \mathcal{A}^*(h),,\ P\mathcal{W}_h P^T\rangle = \|Q\mathcal{A}^*(h)Q\|^2 - \langle P^T\mathcal{A}^*(h)P,\ \mathcal{W}_h\rangle \\
&= \|\underline{H}\|^2 - \langle \overline{W}^T\underline{H}\overline{W},\ \mathcal{W}_h\rangle \qquad \text{(by (2.34) and } P = Q\overline{W}) \\
&= \|\overline{W}^T\underline{H}\overline{W}\|^2 - \langle \overline{W}^T\underline{H}\overline{W},\ \mathcal{W}_h\rangle \qquad \text{(by } \overline{W}\ \overline{W}^T = I_{n+1}) \\
&= 2\left\{\|W_{\underline{\alpha}}^T\underline{h}\|^2 + \|W_{\underline{\alpha}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2 - \langle W_{\underline{\alpha}}^T\underline{H}_1 W_{\underline{\gamma}},\ \Omega_{\underline{\alpha}\underline{\gamma}}\circ(W_{\underline{\alpha}}^T\underline{H}_1 W_{\underline{\gamma}})\rangle\right\} \\
&\quad +2\left\{\|W_{\underline{\beta}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2 + \|W_{\underline{\gamma}}^T\underline{h}\|^2 + \|W_{\underline{\gamma}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2/2\right\} \\
&\quad +\|G_1\|^2 - \langle G_1,\ \widetilde{V}(G_2)\rangle.
\end{aligned}$$

The last equality made use of the structure of $\mathcal{W}_h$ and (2.45).

Define $\tau_{\max} := \max_{i\in\underline{\alpha},j\in\underline{\gamma}}\ \Omega_{ij}$. By (2.39), $0 < \tau_{\max} < 1$. We continue to simplify $\langle h, Mh\rangle$.

$$\begin{aligned}
\langle h,\ Mh\rangle &\geq 2\left\{\|W_{\underline{\alpha}}^T\underline{h}\|^2 + \|W_{\underline{\gamma}}^T\underline{h}\|^2 + \|W_{\underline{\beta}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2 + (1-\tau_{\max})\|W_{\underline{\alpha}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2\right\} \\
&\quad +\|W_{\underline{\gamma}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2 + \|G_1\|^2 - \|G_1\|\|G_2\| \qquad \text{(by (2.47))} \\
&\geq 2\left\{\|W_{\underline{\alpha}}^T\underline{h}\|^2 + \|W_{\underline{\gamma}}^T\underline{h}\|^2 + \frac{1}{2}\|W_{\underline{\beta}}^T\underline{h}\|^2\right\} + \|W_{\underline{\gamma}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2 \\
&\quad +2\left\{(1-\tau_{\max})\|W_{\underline{\alpha}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2 + \|W_{\underline{\beta}}^T\underline{H}_1 W_{\underline{\gamma}}\|^2\right\} + \frac{1}{2}\underline{h}_0^2 \qquad \text{(by (2.46))} \\
&\geq 0. \qquad (2.49)
\end{aligned}$$

Hence, the assumption $\langle h,\ Mh\rangle = 0$ would imply

$$W_{\underline{\alpha}}^T\underline{h} = 0, \quad W_{\underline{\beta}}^T\underline{h} = 0, \quad W_{\underline{\gamma}}^T\underline{h} = 0, \quad \text{and} \quad \underline{h}_0 = 0,$$

and

$$W_{\underline{\alpha}}^T\underline{H}_1 W_{\underline{\gamma}} = 0, \quad W_{\underline{\beta}}^T\underline{H}_1 W_{\underline{\gamma}} = 0, \quad W_{\underline{\gamma}}^T\underline{H}_1 W_{\underline{\gamma}} = 0.$$

Because of (2.40) and nonsingularity of W, the two equations above yield:

$$\underline{h} = 0, \quad \underline{h}_0 = 0 \quad \text{and} \quad \underline{H}_1 W_{\underline{\gamma}} = 0,$$

which by (2.48) and the nonsingularity of Γ (Γ is a permutation matrix) leads to

$$\underline{h} = 0, \quad \underline{h}_0 = 0 \quad \text{and} \quad \underline{H}_1 U_\alpha = 0. \tag{2.50}$$

By assumption, constraint nondegeneracy holds at $\overline{A}$. Lemma 2.9 forces $h = 0$. The inequality (2.49) implies that $\langle h,\ Mh\rangle > 0$ for any $h \neq 0$. Therefore, any matrix in $\widehat{\partial} F(y)$ is positive definite under constraint nondegeneracy. □

(e) Quadratic convergence. The direct consequence of Theorem 2.12 is the quadratic convergence of the Newton method (2.44). Let y^* be an optimal solution of the Lagrangian dual problem (2.29).

Theorem 2.13. *The Newton method (2.44) is quadratically convergent provided that y^0 is sufficiently close to the optimal solution y^* of (2.29) and constraint nondegeneracy holds at Y^* that is defined by (2.33).*

Proof. In the general quadratic convergence-rate Theorem 1.4 for semismooth Newton methods, there are three conditions: (i) The function F is strongly semismooth, which is true for our case because it is a composition of linear mappings and the strongly semismooth mapping $\Pi_{\mathcal{S}_+^{n+1}}(\cdot)$. (ii) Every matrix in the generalized Jacobian of $\widehat{\partial} F(y^*)$ is nonsingular, which has been proved in Theorem 2.12 under constraint nondegeneracy assumption. Furthermore, $\widehat{\partial} F(\cdot)$ is compact and upper semicontinuous. The last condition is that the initial point y^0 stays close to y^*. This proves our result. □

Since (2.29) is convex, globalization of the Newton method (2.44) is straightforward. We simply use one of the well-developed globalization method (Newton-CG method) studied by Qi and Sun (2006) in our numerical experiment.

2.4 Majorized penalty method

In this section, we extend the majorized penalty method of Gao and Sun (2010) to our problem (2.6). The method has previously been used to compute the nearest EDM of low embedding dimensions in Qi and Yuan (2014). The situation here is that we have spherical constraints to deal with. The structure of the extension is similar to that in Qi and Yuan (2014).

(a) The penalty problem. It has been shown that without the rank constraint $\text{rank}(JYJ) \le r$, the convex relaxation problem (2.7) can be solved by the Newton-CG method (2.44). Problem (2.7) implicitly implies a very important fact that the matrix (JYJ) is positive semidefinite for any feasible point Y. Define

$$\begin{aligned} p(Y) &:= \sum_{i=r+1}^{n} \lambda_i(JYJ) = \langle I_{n+1},\ JYJ\rangle - \sum_{i=1}^{r} \lambda_i(JYJ) \\ &= \langle J,\ Y\rangle - \sum_{i=1}^{r} \lambda_i(JYJ), \qquad \text{(because } J^2 = J\text{)} \end{aligned}$$

where $\lambda_1(JYJ) \ge \ldots \ge \lambda_{n+1}(JYJ)$ are the eigenvalues of (JYJ). The equivalent relationship below is obvious.

$$\text{rank}(JYJ) \le r \quad \text{and} \quad JYJ \succeq 0 \quad \Longleftrightarrow \quad p(Y) = 0 \quad \text{and} \quad JYJ \succeq 0.$$

Moreover, $p(Y) \ge 0$ for any Y satisfying $JYJ \succeq 0$. Therefore, the function $p(Y)$ can be used as a penalty function for the rank constraint over the feasible region of (2.6). A similar fact has been used by Gao and Sun (2010) in their majorized penalty method for computing the nearest low-rank correlation matrix, which is

necessarily positive semidefinite. The resulting penalty problem in our case is

$$\begin{aligned} \min \quad & f_c(Y) := f(Y) + cp(Y) \\ \text{s.t.} \quad & \mathcal{A}(Y) = 0, \quad Y \in \mathcal{K}_+^{n+1}, \end{aligned} \tag{2.51}$$

where $c > 0$ is the penalty parameter and $f(Y) := \|Y + D\|^2/2$.

Similar to the result in Gao and Sun (2010, Prop. 3.1 and Prop. 3.2), we have the following result on the relationship between the original problem (2.6) and its penalty counterpart (2.51)

Proposition 2.14. *Let Y_c^* denote a global optimal solution of (2.51), Y_r be a feasible solution of (2.6), and Y^* be an optimal solution of the convex problem (2.7).*

(i) If $\text{rank}(JY_c^*J) \le r$, then Y_c^* already solves (2.6).

(ii) If the penalty parameter c is chosen to satisfy $c \ge (f(Y_r) - f(Y^*))/\epsilon$, for some given $\epsilon > 0$, then we have

$$p(Y_c^*) \le \epsilon \qquad \text{and} \qquad f(Y_c^*) \le \nu^* - cp(Y_c^*),$$

where ν^* denotes the optimal objective vale of (2.6).

Proof. (i) If $\text{rank}(JY_c^*J) \le r$, then Y_c^* is a feasible solution to (2.6) and $p(Y_c^*) = 0$. Since Y_r is also a feasible solution of (2.6), by noting that $p(Y_r) = 0$, we have

$$f(Y_c^*) = f(Y_c^*) + cp(Y_c^*) \le f(Y_r) + cp(Y_r) = f(Y_r).$$

This shows that the conclusion of (i) holds.

(ii) Since Y_r is also a feasible solution of (2.6) and $p(Y_r) = 0$, we have

$$f(Y_r) + cp(Y_r) = f_c(Y_r) \ge f_c(Y_c^*) = f(Y_c^*) + cp(Y_c^*) \ge f(Y^*) + cp(Y_c^*),$$

which implies

$$p(Y_c^*) \le (f(Y_r) - f(Y^*))/c \le \epsilon.$$

Let $\overline{X}$ be a global optimal solution to problem (2.6). Then from

$$f(\overline{X}) + cp(\overline{X}) = f_c(\overline{X}) \ge f_c(Y_c^*) = f(Y_c^*) + cp(Y_c^*)$$

and the fact that $p(\overline{X}) = 0$, we obtain that $f(Y_c^*) \le f(\overline{X}) - cp(Y_c^*) = \nu^* - cp(Y_c^*)$. □

The result in (ii) means that when the rank error measured by $p(\cdot)$ at Y_c^* is less than ϵ, the corresponding objective value comes very close to the optimal value ν^*. Such a solution is referred to as an ϵ-optimal solution in Gao and Sun (2010).

(b) Majorized Penalty Approach. The focus now is on solving the penalty problem (2.51). Since $p(Y)$ is concave (i.e., the sum of the first r largest eigenvalues of a symmetric matrix is a convex function of the matrix), it can be majorized by the linear function defined by its subgradient: For given $Y^k \in \mathcal{S}^{n+1}$ (the current iterate) and $U^k \in \partial p(Y^k)$, we have

$$p(Y) \le m_k^p(Y) := p(Y^k) + \langle U^k,\ Y - Y^k \rangle \qquad \forall\ Y. \tag{2.52}$$

The function $m_k^p(Y)$ is called a majorization of $p(Y)$ at Y^k because of (2.52) and $p(Y^k) = m_k^p(Y^k)$. The majorized (convex) subproblem to be solved is

$$\min\ f(Y) + cm_k^p(Y), \qquad \text{s.t.}\quad \mathcal{A}(Y) = 0,\quad Y \in \mathcal{K}_+^{n+1}. \tag{2.53}$$

We now extend the majorized penalty algorithm of Gao and Sun (2010) to our problem (2.6).

Algorithm 7 Majorized Penalty Algorithm (MPA)

1: Choose a feasible point Y^0 of (2.7). Set $k := 0$.
2: Solve subproblem (2.53) to get $Y^{k+1} = \arg\min_Y \{f(Y) + cm_k^p(Y) \mid \mathcal{A}(Y) = 0,\ Y \in \mathcal{K}_+^{n+1}\}$.
3: If $Y^{k+1} = Y^k$, stop; otherwise, set $k := k + 1$ and go to step 2.

We note that in the step 2 in Algorithm 7, subproblem (2.53) can be solved by the Newton-CG method (2.44). Note that the objective function in (2.53) is

$$\begin{aligned} f(Y) + cm_k^p(Y) &= f(Y) + c(p(Y^k) + \langle U^k,\ Y - Y^k\rangle) \\ &= f(Y) + c\langle U^k,\ Y\rangle + c(p(Y^k) - \langle U^k,\ Y^k\rangle) \\ &= \|Y + D\|^2/2 + c\langle U^k,\ Y\rangle + c(p(Y^k) - \langle U^k,\ Y^k\rangle), \end{aligned}$$

then subproblem (2.53) is equivalent to the following problem, which is the type of the convex problem (2.7) but with different input D:

$$\min\ \frac{1}{2}\|Y + \overline{D}\|^2, \quad \text{s.t.}\ \ \mathcal{A}(Y) = 0,\ \ Y \in \mathcal{K}_+^{n+1}, \tag{2.54}$$

where $\overline{D} := D + cU^k$. We note that the feasible region remains unchanged. Hence, the generalized Slater condition and the constraint nondegeneracy results studied before hold for those subproblems.

We have the following remarks about the algorithm.

(R1) There are a few choices for the starting (feasible) point Y^0 in (S.1). One of them is the optimal solution of (2.7) by the Newton-CG method (2.44).

(R2) Algorithm 7 generates a sequence of decreasing objective values $\{f_c(Y^k)\}$ for fixed c. This is because of the majorization property (2.52). In our practical implementation, the penalty parameter is updated according to some rules.

(R3) The algorithm converges to a B-stationary point of (2.6), which is stated below and whose proof can be patterned after Gao and Sun (2010, Thm. 3.4).

For the definition of B-stationary point, see Gao and Sun (2010). Roughly speaking, as problem (2.6) is nonconvex, converging to a B-stationary point is one kind of global convergence that the algorithm can best achieve. We omit the details. We also note that if Y^f is the final iterate of Algorithm 7, then $(-Y^f)$ should be the final output as it is a true Euclidean distance matrix (put the minus back because we have introduced the minus sign in the formulation process that led to problem (2.6)).

Proposition 2.15. *Let $\{Y^k\}$ be the sequence generated by Algorithm 7. Then $\{f_c(Y^k)\}$ is a monotonically decreasing sequence. If $Y^{k+1} = Y^k$ for some Y^k, then Y^k is an optimal solution of (2.51). Otherwise, the infinite sequence $\{Y^k\}$ satisfies*

$$\frac{1}{2}\|Y^{k+1} - Y^k\|^2 \leq f_c(Y^k) - f_c(Y^{k+1}), \quad k = 0, 1, \ldots.$$

Moreover, the sequence $\{Y^k\}$ is bounded and any accumulation point is a B-stationary point of (2.51).

2.5 Numerical examples by FITS

In this section, we first briefly describe the Matlab implementation of Algorithm 7. For ease of reference, we call the resulting code `FITS`, standing for "FIT data on a Sphere". We then test a few well-known examples that have spherical constraints. Through those examples, it is demonstrated that `FITS` can provide data visualization of high quality and is capable of including extra (linear) constraints such as the "pole constraints" in Ekman's color example (Ekman, 1954), which results in a wheel representation of 14 colors. We are not aware any existing methods that can deal with such pole constraints. It also provides an alternative method for the circle fitting problem, recently studied by Beck and Pan (2012).

(a) Termination Criterion. We terminate Algorithm 7 when the following two conditions are met. The first condition is on the objective function value:

$$f_{\text{frog}} := \frac{|\sqrt{f(Y^k)} - \sqrt{f(Y^{k-1})}|}{\max\{100, \sqrt{f(Y^{k-1})}\}} \leq \texttt{tol}, \tag{2.55}$$

where $f(Y) = 0.5\|Y + D\|^2$ and `tol` is a small tolerance level (e.g., 1.0×10^{-4}). In other words, whenever there is lack of the relative progress on the successive objective function values, we believe that the current iterate is a good candidate subject to the second condition below. This stopping criterion was suggested by Gao and Sun (2010) for the low-rank nearest correlation matrix problem.

The second condition is on the rank of the current iterate Y^k. There are two ways to monitor the rank. One is to compute the absolute value of the eigenvalue residue:

$$\texttt{rankerror} := \sum_{i=r+1}^{n} \lambda_i(JY^kJ) \leq \texttt{ranktol}, \tag{2.56}$$

where `ranktol` is a small tolerance (e.g., 10^{-2}) and $\lambda_1 \geq \ldots \geq \lambda_n$ are the eigenvalues of (JY^kJ), which is positive semidefinite. This quantity does not scale well with the magnitude of (JY^kJ). To rectify this drawback, we also calculate the percentage of the first r eigenvalues of (JY^kJ) out of all the eigenvalues:

$$\texttt{Eigenratio} := \sum_{i=1}^{r} \lambda_i(JY^kJ) / \sum_{i=1}^{n} \lambda_i(JY^kJ) \leq \texttt{Eigentol}, \tag{2.57}$$

where `Eigentol` is a high percentage (e.g., 90%). We terminate the algorithm when (2.55) and either of (2.56) and (2.57) are satisfied.

(b) Initial Point and Updating the Penalty Parameter. The initial point is computed by the Semismooth Newton-CG method for the convex problem (2.7). We note that all the subproblems of (2.54) are solved by the same Newton-CG

method. Algorithm 7 solves the penalty problem (2.51) for a fixed penalty parameter c. In practical implementation, we may start from c_0 and increase c a few times before we can find a good solution. The initial $c_0 = 10$ in our implementation. We update c_k $(k \geq 1)$ as follows

$$c_k := \begin{cases} c_{k-1}, & \text{if } \operatorname{rank}(JY^{k-1}J) \leq r \\ 4c_{k-1}, & \text{otherwise.} \end{cases}$$

That is, we keep the penalty parameter unchanged if the current iterate has the desired embedding dimension. Otherwise, it is increased by 4 times.

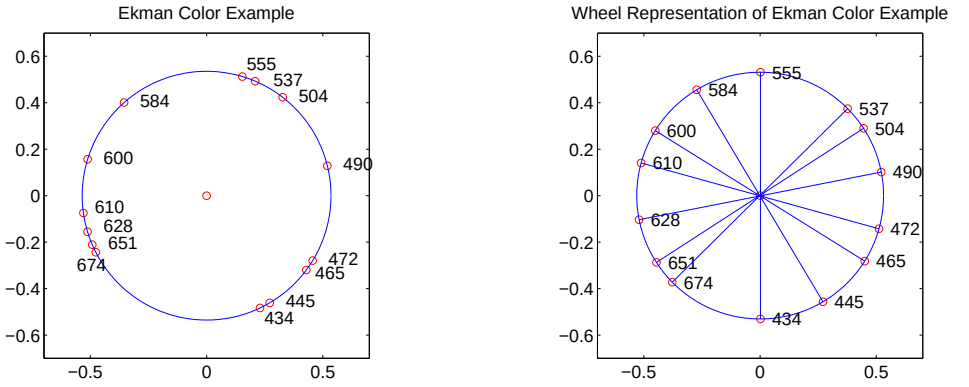
(c) Numerical Examples. Five examples were tested. They are (E1) Ekman's color example (Ekman, 1954), (E2) Trading globe in Cox and Cox (1991), (E3) 3D Map of global cities (HA30 data set[1]) in Hartigan (1975), and (E4) Circle fitting problem in Beck and Pan (2012). (E5) Points randomly distributed on a sphere and on a circle. We also show how a Procrustes procedure can be devised to assess the quality of the final embedding in (E3) or to help to get the final embedding to match the existing points in (E4).

(E1) Ekman color example. This is a classical example in MDS where data can be represented on a circle, called circular fitting. Ekman (1954) presents similarities for 14 colors (wavelengths from 434 to 674 nm). The similarities are based on a rating by 31 subjects where each pair of colors was rated on a 5-point scale (0 means no similarity up to 4 meaning identical). After averaging, the similarities were divided by 4 such that they are within the unit interval. The resulting similarity matrix (shown in Table 2.1) is denoted by Δ. The initial distance matrix is obtained from $(D^0)_{ij} := (1 - \Delta_{ij})^2$.

[1]Data available from http://people.sc.fsu.edu/~jburkardt/m_src/distance_to_position_sphere.html

Table 2.1: Similarities of colors with wavelengths from 434 nm to 674 nm (Ekman, 1954)

	434	445	465	472	490	504	537	555	584	600	610	628	651	674
434	0.00	0.86	0.42	0.42	0.18	0.06	0.07	0.04	0.02	0.07	0.09	0.12	0.13	0.16
445	0.86	0.00	0.50	0.44	0.22	0.09	0.07	0.07	0.02	0.04	0.07	0.11	0.13	0.14
465	0.42	0.50	0.00	0.81	0.47	0.17	0.10	0.08	0.02	0.01	0.02	0.01	0.05	0.03
472	0.42	0.44	0.81	0.00	0.54	0.25	0.10	0.09	0.02	0.01	0.00	0.01	0.02	0.04
490	0.18	0.22	0.47	0.54	0.00	0.61	0.31	0.26	0.07	0.02	0.02	0.01	0.02	0.00
504	0.06	0.09	0.17	0.25	0.61	0.00	0.62	0.45	0.14	0.08	0.02	0.02	0.02	0.01
537	0.07	0.07	0.10	0.10	0.31	0.62	0.00	0.73	0.22	0.14	0.05	0.02	0.02	0.00
555	0.04	0.07	0.08	0.09	0.26	0.45	0.73	0.00	0.33	0.19	0.04	0.03	0.02	0.02
584	0.02	0.02	0.02	0.02	0.07	0.14	0.22	0.33	0.00	0.58	0.37	0.27	0.20	0.23
600	0.07	0.04	0.01	0.01	0.02	0.08	0.14	0.19	0.58	0.00	0.74	0.50	0.41	0.28
610	0.09	0.07	0.02	0.00	0.02	0.02	0.05	0.04	0.37	0.74	0.00	0.76	0.62	0.55
628	0.12	0.11	0.01	0.01	0.01	0.02	0.02	0.03	0.27	0.50	0.76	0.00	0.85	0.68
651	0.13	0.13	0.05	0.02	0.02	0.02	0.02	0.02	0.20	0.41	0.62	0.85	0.00	0.76
674	0.16	0.14	0.03	0.04	0.00	0.01	0.00	0.02	0.23	0.28	0.55	0.68	0.76	0.00



(a) Circular fitting without any constraints

(b) Circuclar fitting with pole constraints

Figure 2.1: Comparison between the two circular fitting of Ekman's 14 color problem with and without pole constraints.

Figure 2.1 (a) (the radius is $R = 0.5354$) is the resulting circular representation by `FITS` with colors appearing on the circle one by one in order of their wavelength. This figure is similar to De Leeuw and Mair (2009, Fig. 2), where more comments on this example can be found. A pair of colors (i, j) are said opposing to each other if their distance equals the diameter of the circle. That is

$$Y_{ij} = 4Y_{1(n+1)}, \tag{2.58}$$

which means that the squared distance between opposing colors is fourfold of the radius squared. This type of constraints is called "pole constraint". An interesting feature is that we assume that the first 7 colors are set to oppose the remaining 7 colors, the resulting circular representation appears as a nice wheel, without having changed the order of the colors, see Figure 2.1(b) (the radius is 0.5310). Practitioners in Psychology may have new interpretation of such nice representation. We emphasize that our method can easily include the pole constraints and other linear constraints without any technical difficulties. We are not aware any existing methods that can directly handle those extra constraints.

(E2) Trading globe. The data in this example was first mapped to a sphere ($r =$

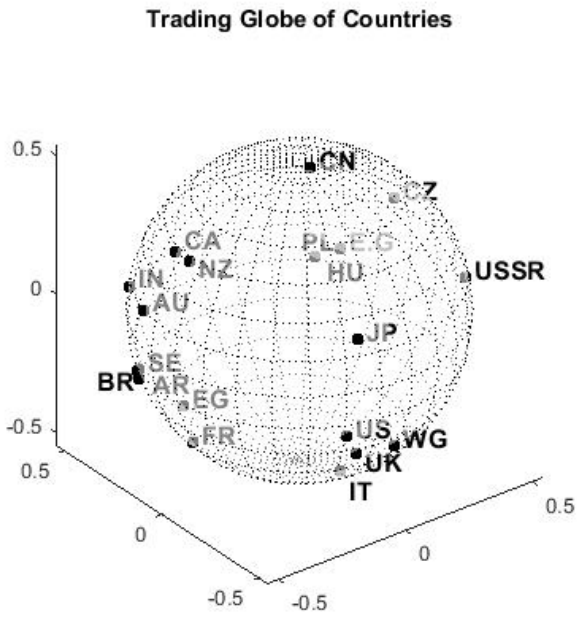


Figure 2.2: Spherical representation for trading data in 1986 between countries {Argentina, Australia, Brazil, Canada, China, Czechoslovakia, East Germany, Egypt, France, Hungary, India, Italy, Japan, New Zealand, Poland, Sweden, UK, USA, USSR, West Germany}.

3) in Cox and Cox (1991) and was recently tested in De Leeuw and Mair (2009). The data was originally taken from the New Geographical Digest (1986) on which countries traded with other countries. For 20 countries the main trading partners are dichotomously scored (1 means trade performed, 0 trade not performed) as shown in Table 2.2. Based on this dichotomous matrix X the distance matrix D^0 is computed using the squared Jaccard coefficient (computed by the Matlab build-in function `pdist`(X, 'jaccard'). The most intuitive MDS approach is to project the resulting distances to a sphere which gives a "trading globe".

In Figure 2.2 ($R = 0.5428$), the counties were projected on to a globe with the shaded points being on the other side of the sphere. The figure is from the default viewpoint of Matlab. It is interesting to point out that obvious clusters of countries can be observed. For example, on the top left is the cluster of Commonwealth nations (Australia, Canada, India, and New Zealand). On the bottom right is the cluster of western allies (UK, US, and West Germany) with Japan

Table 2.2: Nations' trading data from New Geographical Digest (1986)

Arge	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1
Aust	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	1
Braz	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1
Cana	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
Chin	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
Czec	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0
Egyp	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1
E.Ge	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0	1
Fran	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1
Hung	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1
Indi	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1
Ital	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1
Japa	1	1	1	1	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0
N.Ze	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
Pola	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1
Swed	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
USA	1	1	1	1	0	0	1	0	1	0	1	0	1	1	0	1	0	0	1	1
USSR	0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0
U.K	0	1	0	1	0	0	1	0	1	0	1	0	0	1	0	1	1	0	0	1
W.Ge	1	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	1	0	1	0

not far on above of them. On the north pole is China, which reflects its isolated trading situation back in 1986. On the backside is the cluster of countries headed by USSR. On the left backside is the cluster of Brazil, Argentina, Egypt. We note that this figure appears different from those in Cox and Cox (1991); De Leeuw and Mair (2009) mainly because that each used a different method on a different (nonconvex) model of the spherical embedding of the data.

(E3) 3D Map of global cities in HA30 data set. HA30 is a dataset of spherical distances among 30 global cities, measured in hundreds of miles and selected by Hartigan (1975) from the World Almanac, 1966. It also provides XYZ coordinates of those cities. In order to use `FITS`, we first convert the spherical distances to Euclidean distances through the formula: $d_{ij} := 2R\sin(s_{ij}/(2R))$ where s_{ij} is the spherical distance between city i and city j and $R = 39.59$ (hundreds miles) is the Earth radius (see Pękalska and Duin (2005, Thm. 3.23)). The initial matrix D^0 consists of the squared distances d_{ij}^2. It is observed that the matrix $(-JD^0J)$ has 15 positive eigenvalues and 14 negative eigenvalues and 1 zero eigenvalue. Therefore, the original spherical distances are not accurate and contain large errors. Therefore, `FITS` is needed to correct those errors. We plot the resulting coordinates of the 30 cities in Figure 2.3. One of the remarkable features is that `FITS` is able to recover the Earth radius with high accuracy $R = 39.5916$.

We now assess the quality of the spherical embedding in Figure 2.3 through a Procrustes analysis introduced in Section 1.1.3. The optimal objective f in (1.14) is $f = 0.2782$. This small error is probably due to the fact that the radius used in HA30 is 39.59 in contrast to ours 39.5916. This small value also confirms the good quality of the embedding from `FITS` when compared to the solution in HA30.

(E4) Circle fitting. The problem of circle fitting has recently been studied in

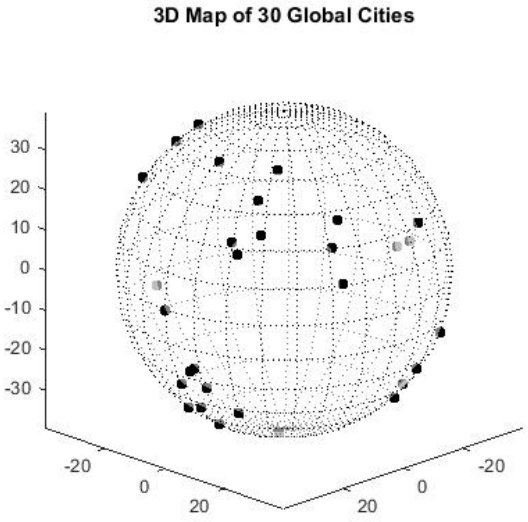


Figure 2.3: Spherical embedding of HA30 data set with radius $R = 39.5916$.

Beck and Pan (2012), where more references on the topic can be found. Let points $\{\mathbf{a}_i\}_{i=1}^n$ with $\mathbf{a}_i \in \mathbb{R}^r$ be given. The problem is to find a circle with center $\mathbf{x} \in \mathbb{R}^r$ and radius R such that the points stay as close to the circle as possible. Two criteria were considered in Beck and Pan (2012):

$$\min_{\mathbf{x},\ R} \ f_1 = \sum_{i=1}^{n} \left(\|\mathbf{a}_i - \mathbf{x}\| - R\right)^2 \tag{2.59}$$

and

$$\min_{\mathbf{x},\ R} \ f_2 = \sum_{i=1}^{n} \left(\|\mathbf{a}_i - \mathbf{x}\|^2 - R^2\right)^2. \tag{2.60}$$

Problem (2.60) is much easier to solve than (2.59). But the key numerical message in Beck and Pan (2012) is that (2.59) may produce far better geometric fitting than (2.60). This was demonstrated through the following example Beck and Pan (2012, Example 5.3):

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 9 \end{bmatrix}, \ \mathbf{a}_2 = \begin{bmatrix} 2 \\ 7 \end{bmatrix}, \ \mathbf{a}_3 = \begin{bmatrix} 5 \\ 8 \end{bmatrix}, \ \mathbf{a}_4 = \begin{bmatrix} 7 \\ 7 \end{bmatrix}, \ \mathbf{a}_5 = \begin{bmatrix} 9 \\ 5 \end{bmatrix}, \ \mathbf{a}_6 = \begin{bmatrix} 3 \\ 7 \end{bmatrix}.$$

Model (2.60) produces a very small circle, not truly reflecting the geometric layout of the data.

The Euclidean distance embedding studied in this paper provides an alternative model. Let $D^0_{ij} = \|\mathbf{a}_i - \mathbf{a}_j\|^2$ for $i = 1, \ldots, n$ and $n = 6$, $r = 2$ in this example. Let $\overline{Y}$ be the final distance matrix from `FITS` and the embedding points in X be obtained from (1.12). The first 6 columns $\{\mathbf{x}_i\}_{i=1}^6$ of X correspond to the known points $\{\mathbf{a}_i\}_{i=1}^6$. The last column $\mathbf{x}_7$ is the center. The points $\{\mathbf{x}_i\}_{i=1}^6$ are on the circle centered at $\mathbf{x}_7$ with radius R ($R = \sqrt{\overline{Y}_{1(n+1)}}$). We need to match $\{\mathbf{x}_i\}_{i=1}^6$ to $\{\mathbf{a}_i\}_{i=1}^6$ so that the known points stay as close to the circle as possible. This can be done through the orthogonal Procrustes problem (1.14).

We first centralize both sets of points. Let

$$\mathbf{a}_0 := \frac{1}{n}\sum_{i=1}^n \mathbf{a}_i, \quad \overline{\mathbf{a}}_i := \mathbf{a}_i - \mathbf{a}_0 \quad \text{and} \quad \mathbf{x}_0 := \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i, \quad \overline{\mathbf{x}}_i := \mathbf{x}_i - \mathbf{x}_0, \quad i = 1, \ldots, n.$$

Let A be the matrix whose columns are $\overline{\mathbf{a}}_i$ and Z whose columns are $\overline{\mathbf{x}}_i$ for $i = 1, \ldots, n$. Solve the orthogonal Procrustes problem (1.14) to get $P = UV^T$. The resulting points are

$$\mathbf{z}_i := P\overline{\mathbf{x}}_i + \mathbf{a}_0, \qquad i = 1, \ldots, n$$

and the new center, denoted by $\mathbf{z}_{n+1}$, is

$$\mathbf{z}_{n+1} := P(\mathbf{x}_{n+1} - \mathbf{x}_0) + \mathbf{a}_0.$$

It can be verified that the points $\{\mathbf{z}_i\}_{i=1}^n$ are on the circle centered at $\mathbf{z}_{n+1}$ with radius R. That is

$$\|\mathbf{z}_i - \mathbf{z}_{n+1}\|^2 = \|P(\mathbf{x}_i - \mathbf{x}_{n+1})\|^2 = \|\mathbf{x}_i - \mathbf{x}_{n+1}\|^2 = R^2.$$

This circle is the best circle from model (2.2) and is plotted in Figure 2.4 with the

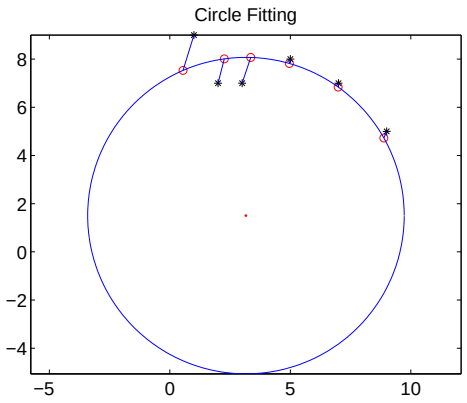
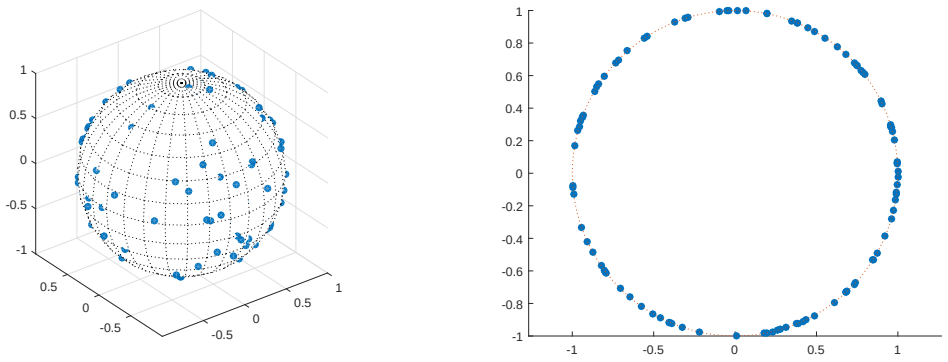


Figure 2.4: Circle fitting of 6 points with $R = 6.5673$. The known points and their corresponding points on the circle by `FITS` are linked by a line.

pair of points $\{\mathbf{a}_i,\ \mathbf{z}_i\}$ being linked by a line. When the obtained center $\mathbf{x} = \mathbf{z}_{n+1}$ and R are substituted to (2.59), we get $f_1 = 3.6789$, not far from the reported value $f_1 = 3.1724$ in Beck and Pan (2012). The circle fits the original data reasonably well. The model used by Beck and Pan (2012) is nonconvex and the resulting f highly depends on a good starting point while our algorithm solves a sequence of convex relaxations that do not count on a good starting point. We complete this example by noting a common feature between our model (2.2) and the squared least square model (2.60) in that the squared distances are used in both models. But the key difference is that (2.2) used all available pairwise squared distances among $\mathbf{a}_i$ rather than just those from $\mathbf{a}_i$ to the center $\mathbf{x}$ as is in (2.60).

(E5) Synthetic data. In this part we generate random data to test the performance of Algorithm 7 with growing dimension. Two types of data are used as shown in Figure 2.5. The first one contains points randomly distributed on a sphere and the second one contains points on a circle, both of them have the radius 1, note that we do not actually use the radius as a given information in our algorithm. The noise in the distance information is generated following a standard framework as follows:

$$\hat{D}_{ij} = D_{ij} \times |1 + nf \times randn|,\ i = 1, \ldots, n, j = 1, \ldots, n, \tag{2.61}$$



(a) Points randomly distributed on a sphere

(b) Points randomly distributed on a circle

Figure 2.5: Synthetic data with $n = 200$ points randomly distributed

where D_{ij} is the true Euclidean distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$, $0 \leqslant nf \leqslant 1$ is the noise factor, *randn* is the standard normal random variable. The accuracy measurement of the estimated positions is the root mean square distance (RMSD)

$$\text{RMSD} := \frac{1}{\sqrt{n}} \left(\sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 \right)^{\frac{1}{2}}, \tag{2.62}$$

where $\hat{\mathbf{x}}_i$ is the estimated position and $\mathbf{x}_i$ is the ground-truth position. Note that without knowing some points position in advance, only relative positions can be found. To calculate RMSD, we apply the Procrustes (1.14) to all estimated points to get the global coordinates.

To test the computation efficiency of Algorithm 7, data with various number of points from 100 to 1000 are used. The noise factor nf is set to 1. The time and accuracy results of sphere data are listed in Table 2.3.

The first column is the number of data points that are generated. The second column is the number of convex subproblems that are solved in the step (2) of Algorithm 7. The third column contains the total number of iteration in Newton method among all subproblems, i.e., the iteration number of the semismooth Newton method (2.44). We can see that as the scale of problem going large, the RMSD decreases since the radius of the sphere remains 1 and the density of points

Table 2.3: Execution Time and Quality Results on Sphere Data

n	Subproblem	Total iteration	RMSD	Time
100	5	25	4.05E-02	5.80
200	5	26	2.89E-02	11.90
300	5	31	2.39E-02	23.05
400	5	34	2.09E-02	41.14
500	5	35	1.91E-02	57.75
600	5	38	1.75E-02	86.01
700	5	40	1.64E-02	103.57
800	5	41	1.56E-02	144.18
900	5	43	1.47E-02	170.94
1000	5	44	1.41E-02	209.07

is increasing. For problem with small scale, our method only takes seconds to get a result with RMSD around 10^{-2}. For problem with 1000 points, it takes Algorithm 7 around 3 minutes to solve it. Similar observation can be obtained for circle data shown in Table 2.4.

Table 2.4: Execution Time and Quality Results on Circle Data

n	Subproblem	Total iteration	RMSD	Time
100	5	24	2.55E-02	5.13
200	5	28	2.06E-02	11.97
300	5	33	1.69E-02	22.15
400	5	35	1.45E-02	38.11
500	5	37	1.32E-02	58.39
600	5	40	1.20E-02	77.61
700	5	40	1.13E-02	100.70
800	5	40	1.07E-02	124.75
900	5	41	1.04E-02	144.65
1000	5	44	1.01E-02	186.20

To test the influence of noise factor nf on the accuracy of **FITS**, we vary nf from 0.1 to 0.5, the resulting RMSD on sphere and circle data are depicted in Figure 2.6. we can see that our algorithm achieved better RMSD on circle data than on sphere data, and the RMSD on both data sets are less than 20% of the radius even when the noise factor is as large as 0.5.

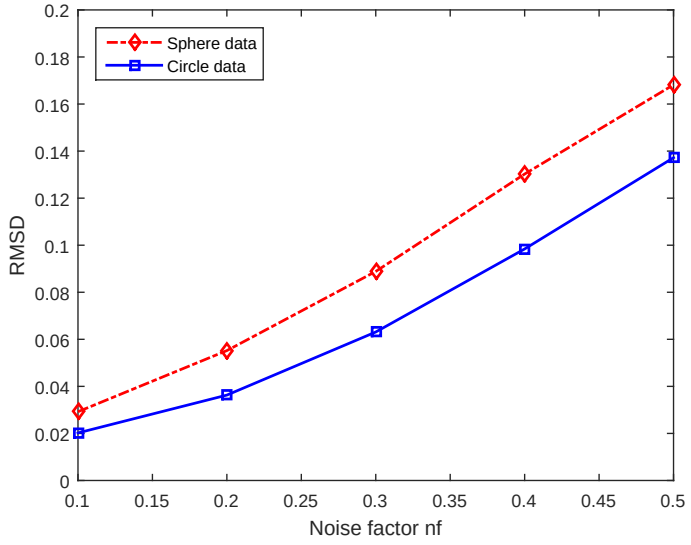


Figure 2.6: Variation of RMSD with varying number of noise factor nf

2.6 Summary

In this section, we proposed a matrix optimization approach to the problem of Euclidean distance embedding on a sphere. We applied the majorized penalty method of Gao and Sun (2010) to the resulting matrix problem. A key feature we exploited is that all subproblems to be solved share a common set of Euclidean distance constraints with a simple distance objective function. We showed that such problems can be efficiently solved by the Newton-CG method, which is proved to be quadratically convergent under constraint nondegeneracy.

Constraint nondegeneracy is a difficult constraint qualification to analyze. We proved it under a weak condition for our problem. We illustrated in Example 2.1 that this condition holds everywhere but one point ($t = 0$). This means that constraint nondegeneracy is satisfied for $t \neq 0$. For the case $t = 0$, we can verify (through verifying Lemma 2.9) that constraint nondegeneracy also holds. This motivates our open question whether constraint nondegeneracy should hold under a weaker condition.

In the numerical part, we used 4 existing embedding problems on a sphere to demonstrate a variety of applications that the developed algorithm can be applied to. The first two examples are from classical MDS and new features (wheel representation for E1 and new clusters for E2) are revealed. For E3, despite the large noises in the initial distance matrix, our method is remarkably able to recover the Earth radius and to project accurate mapping of the 30 global cities on the sphere. The last example is different from the others in that its inputs are the coordinates of known points (rather than a distance matrix). Finding the best circle to fit those points requires localization of its center and radius. The resulting visualizations are very satisfactory for all the examples. Since those examples are of small scale, our method took less than 1 second to find the optimal embedding. Hence, we omitted reporting such information.

Chapter 3

EDM-based optimization approach for sensor network localization

In this chapter, we introduce an EDM-based optimization model with bound constraints to develop a new scheme for sensor network localization. Compared to the previous work on problem (1.21) and results in the last chapter, the model to be discussed in the chapter contains a large number of inequality constraints, which will cause difficulties for previous methods to work efficiently. And efficiency is a fundamental requirement for any sensor network localization scheme. Moreover, A robust and fault-tolerant localization system is also essentially required in many monitoring applications such as environment monitoring and industrial control. There have been significant advances in range-based numerical methods for sensor network localizations over the past decade. However, there remain a few challenges to be resolved to satisfaction. Those issues include, for example, the flip ambiguity, high level of noises in distance measurements, and irregular topology of the concerning network. Each or a combination of them often severely degrades the otherwise good performance of existing methods. Integrating the connectivity

constraints is an effective way to deal with those issues. However, there are too many of such constraints, especially in a large and sparse network. In our model, the connectivity constraints can be simply represented as lower and upper bounds on the elements of EDM, resulting in a standard 3-block quadratic conic programming, which can be efficiently solved by the 3-block alternating direction method of multipliers, which is a special case of Algorithm 5. Numerical experiments show that the EDM model effectively eliminates the flip ambiguity and retains robustness in terms of being resistance to irregular wireless sensor network topology and high noise levels.

The rest of the chapter is organized as follows. In the first section, we give a brief introduction to sensor network localization and basic modelling technique using EDM. Our EDM-based localization scheme is described in Section 3.2, where we describe the convex optimization model, deal with the rank constraint as well as discuss the global coordinate recovery. In Section 3.3, we introduce a convergent 3-block ADMM algorithm for our convex optimization model to retrieve missing distance information. Numerical comparison with existing state-of-art methods is reported in Section 3.4, which demonstrates the robustness of our model in handling the issues reviewed in Section 3.1.

3.1 Introduction to sensor network localization

Wireless Sensor Networks (WSNs) consist of a collection of spatially distributed autonomous sensor nodes, which can sense, measure, gather and transmit information from a geographical area (Patwari et al., 2005). WSNs play an important role in a variety of applications such as environmental/earth sensing and industrial monitoring, in which an accurate realization of sensor positions with respect to a global coordinate system is highly desirable for the data gathered to be geographically meaningful. One way to locate sensor nodes is by equipping each of

them with a global positioning system (GPS) device, which could be very costly and significantly energy consuming for networks with numerous sensors. Thus, a common approach is to firstly acquire only a small portion of sensor positions by manual deployment or by GPS, and they are known as *anchor* nodes. It is then to locate the rest of nodes in the network using the connectivity or pairwise distance information from typical range measurements such as time-of-arrival (ToA). We refer to the book by Akyildiz and Vuran (2010) for the details of hardware technologies. Localization with only connectivity information is categorized as range-free localization. It often provides less accurate positions than the range-based localization, in which both connectivity and pairwise distance information are used. The main focus of this section lies in the latter category as we assume that partial information on pairwise distances is provided.

Suppose there are n sensors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^r$ ($r = 2$ or 3). Some of them may be anchors. Assume that Euclidean distance among some of the sensors can be observed:

$$\widehat{d}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| + \epsilon_{ij}, \tag{3.1}$$

where $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^r$, ϵ_{ij} are errors, and $\widehat{d}_{ij}$ are observed distances. The sensor network localization (SNL) problem is to recover the sensor positions through those distances. The most popular method may be the classical Multidimensional Scaling (cMDS) summarized in Algorithm 1. It works well when a large number of $\widehat{d}_{ij}$ are close to their true distances (i.e., ϵ_{ij}s are relatively small). Otherwise, one has to opt for many of its variants. An alternative method called SMACOF (De Leeuw and Mair, 2009) is proposed to recover node positions from incomplete distance information, which minimizes a nonconvex *stress* function iteratively using majorization. Based on SMACOF, a distributed weighted-MDS is developed by Costa et al. (2006). Another well known example of metric MDS is MDS-MAP proposed by Shang et al. (2003), which tries to estimate all missing distance information by computing the shortest paths between all pairs of nodes.

Theoretical analysis on the performance of MDS-MAP can be found in Karbasi and Oh (2013); Drineas et al. (2006). MDS-MAP often works when nodes are positioned relatively uniformly in the space, but does not perform well on networks with irregular topology, where the shortest path distance does not correlate well with the true Euclidean distance. To compensate for this drawback, several "patching" algorithms, such as MDS-MAP(P) (Shang and Ruml, 2004), PATCHWORK (Koren et al., 2005) and As-Rigid-As-Possible (ARAP) (Zhang et al., 2010), are proposed, which starts by first localizing small patches and then stitching them together to recover the global coordinates. Gepshtein *et al.* proposed an anchor-based sensor networks localization scheme called ADESR in Gepshtein and Keller (2015) that utilizes a dual spectral embedding. It is experimentally shown that ADESR outperforms ARAP in terms of robustness to noise and localization accuracy, but it may suffer from the increasing computational complexity.

Another important class of methods is the Semi-Definite Programming (SDP) relaxation-based, initially introduced by Biswas and Ye (2004) to SNL, which draws much attention in the WSN society due to their global convergence property. The major modelling procedure in many papers in this class of methods is first to formulate the SNL problem as a quadratic optimization problem and then to relax it as SDP, which can be efficiently solved by interior point methods such as SDPT3 in Toh et al. (1999). Important techniques have been introduced in order to improve the quality and/or computational efficiency of various relaxations. For example, techniques of regularization and refinement have been employed in Biswas et al. (2006). Edge-based SDP (ESDP) and node-based SDP (NSDP) relaxations of the full SDP (FSDP) (Biswas and Ye, 2004) are studied in Wang et al. (2008). These further relaxation approaches are weaker than the original SDP relaxation in theory, but computational results show that the quality of solutions may not be influenced. A sparse version of FSDP has been implemented as SFSDP by Kim et al. (2012) and it requires much less computational time than most of

the SDP relaxation methods. Besides SDP techniques, second-order cone programming (SOCP) relaxation is also studied by Tseng (2007) to handle thousands of sensors, however it is less inaccurate compared to SDP-based algorithms (Kim et al., 2012). SDP relaxation based methods work well on most types of networks. However, numerical simulation shows that they are very sensitive to networks with randomly deployed anchor nodes, and often suffer from flip ambiguity, which may induce large errors to the final localization.

As lightly touched above and already reported in existing literature, the issues such as high level of noises (i.e., ϵ_{ij} in (3.1) are relatively large), irregular topology (e.g., sensors are not uniformly distributed over a convex region) and flip ambiguity can severely degrade the performance of existing methods. In particular, flip ambiguity is one of the major challenges brought up by ranging errors. It arises when the neighbours of a node lie almost in a line such that the node can be mirror reflected across the line while still satisfying the distance constraints from its neighbours (Kannan et al., 2010), as illustrated in Figure 3.1. In this small network, node A could be mirror reflected to the position of A' when there exists error in the distance measurements, while the distances under the same connectivity states, i.e. $A'B$, $A'C$, $A'D$, can still remain unchanged, thereby causing a large localization error. Moreover, the error may transmit and a large part of the sensors could be folded over. For a larger network example, see Figure 3.4(b).

Several heuristic methods have been purposefully proposed to tackle the issue of flip ambiguity such as the two-phase simulated annealing algorithm by Kannan et al. (2006) and the two-step tabu search algrithm by Shekofteh et al. (2010). These methods can be robust and efficient if the parameters are well controlled. However, tuning parameters for networks of various sizes and topologies is very time consuming with no guarantee of success. Most of the patching and stitching methods (Shang and Ruml, 2004; Koren et al., 2005; Zhang et al., 2010) can resist

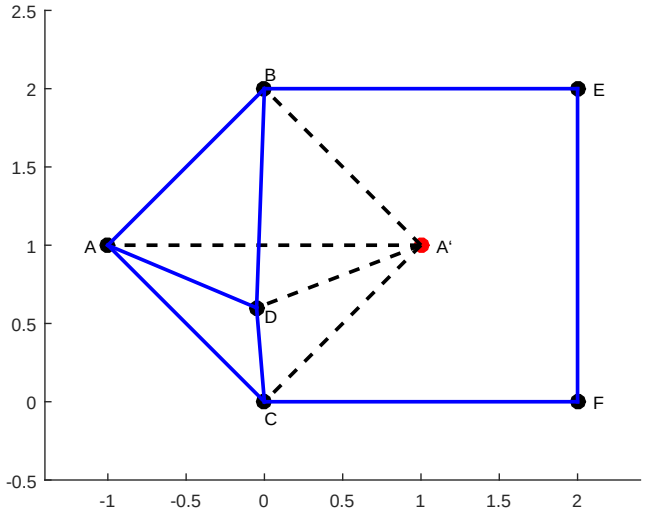


Figure 3.1: Illustration of flip ambiguity, a small network with 6 nodes, communication radius $R = 2.1$, blue lines indicate the existence of communication between two nodes.

flip ambiguity quite well, however, as discussed above, they are either encountered with inaccurate positions or inefficient calculation.

The strategy we propose here is more direct and is geometric. Take the case in Figure 3.1 for example, we note that point A has no connection with both points E and F. If we enforce the (dis)connectivity constraints

$$\|\mathbf{x}_A - \mathbf{x}_E\| > R \quad \text{and} \quad \|\mathbf{x}_A - \mathbf{x}_F\| > R, \tag{3.2}$$

then the possibility of point A flipping to point A' would be removed because A' would fall within the communication range of points E and F. For this strategy to be effective, the following three elements have to be taken into consideration. (i) There are a large number of such connectivity constraints as in (3.2). The order is $O(n^2)$. The more sparse the network is , the more of such constraints there are. (ii) The connectivity constraints such as (3.2) should be satisfied in the embedding space $\mathbb{R}^r$, which is a low-dimensional space. This essentially introduces nonconvexity to the problem. (iii) The computational model of localization, as

done in SDP relaxation methods reviewed above, should be of convex optimization in order to develop fast algorithms. The consideration of the three elements all together will result in robust localization and this is where our research departs from most of the existing ones.

By employing the concept of EDM, we can effectively address the three elements above. Namely, the connectivity constraints will be simply represented as lower and/upper bounds constraints. It is those bound constraints that play an important role in mitigating flip ambiguity and obtaining robustness under presence of large noises in the distance measurements. The nonconvexity concerning the embedding space is formulated as a rank constraint, which is further approximated by a (heuristic) linear function to induce a low rank solution. The final computational model belongs to the class of convex EDM optimization model (1.21) with additional bound constraints. Both primal and dual problem of our model has 3 separate block-variables. It is this nice and tidy structure that makes it possible to implement an efficient algorithm of Alternating Direction Method of Multiplier (ADMM). The algorithm is derived based on Algorithm 6 introduced in Section 1.3.2, which allows us to handle millions of inequality constraints efficiently when the sensor number grows to thousands. Numerical comparison will demonstrate that, with less computational complexity, our approach outperforms the existing state-of-art methods in terms of flip ambiguity elimination, being resistance to irregular WSN topology and presence of large noises in ranging measurements.

3.2 EDM-based localization scheme

In this section, we first state the SNL problem, especially on what types of constraints we intend to address. We then describe how we reformulate the problem as a convex EDM optimization. The final part is about placing the recovered

sensors to the coordinate systems used by the existing anchors. If there exist no anchors, this part will not be necessary.

3.2.1 SNL problem statement

Assume a sensor network in $\mathbb{R}^r$ ($r = 2$ or 3) has n nodes in total, with m known anchor nodes and $n - m$ sensor nodes whose locations are unknown (m may take 0). Let $\mathbf{x}_k = \mathbf{a}_k \in \mathbb{R}^r$, $k = 1, 2, \ldots, m$ denote the location of kth anchor node, and $\mathbf{x}_i \in \mathbb{R}^r$, $i = m + 1, \ldots, n$ denote the location of ith sensor node. The maximum communication range is R, which determines two index sets $\mathcal{N}_x$ and $\mathcal{N}_a$ that indicates the connectivity states of nodes. For any $(i, j) \in \mathcal{N}_x$, the Euclidean distance d_{ij} between sensor nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ is not greater than R. Hence, the two sensor nodes are able to transmit signal between each other. Similarly, for any $(i, k) \in \mathcal{N}_a$, a sensor $\mathbf{x}_i$ and an anchor node $\mathbf{a}_k$ can communicate with each other and their Euclidean distance $d_{ik} \leq R$. Therefore, we have

$$\begin{aligned} \mathcal{N}_x &:= \{(i, j) : m + 1 \leq i < j \leq n, \|\mathbf{x}_i - \mathbf{x}_j\| \leq R\}, \\ \mathcal{N}_a &:= \{(i, k) : m + 1 \leq i \leq n, 1 \leq k \leq m, \|\mathbf{x}_i - \mathbf{a}_k\| \leq R\}. \end{aligned}$$

To indicate a pair of nodes that are too far away to communicate with each other, we define

$$\begin{aligned} \overline{\mathcal{N}}_x &:= \{(i, j) : m + 1 \leq i < j \leq n, \|\mathbf{x}_i - \mathbf{x}_j\| > R\}, \\ \overline{\mathcal{N}}_a &:= \{(i, k) : m + 1 \leq i \leq n, 1 \leq k \leq m, \|\mathbf{x}_i - \mathbf{a}_k\| > R\}. \end{aligned}$$

For any sensor nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, $(i, j) \in \mathcal{N}_x \cup \mathcal{N}_a$, a noisy range measurement $\widehat{d}_{ij}$ is taken. We assume that the distance estimations are symmetric, i.e., $\widehat{d}_{ij} = \widehat{d}_{ji}$. Then the range-based sensor network localization problem can be described as to

recover $\mathbf{x}_{m+1}, \mathbf{x}_{m+2}, \ldots, \mathbf{x}_n$ such that

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 \approx \widehat{d}_{ij}^2, \quad \forall (i,j) \in \mathcal{N}_x, \tag{3.3a}$$

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 > R^2, \quad \forall (i,j) \in \overline{\mathcal{N}}_x, \tag{3.3b}$$

$$\|\mathbf{x}_i - \mathbf{a}_k\|^2 \approx \widehat{d}_{ik}^2, \quad \forall (i,k) \in \mathcal{N}_a, \tag{3.3c}$$

$$\|\mathbf{x}_i - \mathbf{a}_k\|^2 > R^2, \quad \forall (i,k) \in \overline{\mathcal{N}}_a. \tag{3.3d}$$

(3.3a) and (3.3c) come from the incomplete distance information, and (3.3b) and (3.3d) come from the connectivity information due to the limitation of radio range. That is, if there is no distance information measured between two nodes, then their Euclidean distance is greater than R. Many existing localization schemes neglect all the inequality constraints (3.3b) and (3.3d). However, as some of the existing research demonstrated, those bound constraints can actually improve the robustness and accuracy of localization. In particular, Biswas and Ye (2004) suggest to select some of these lower bound constraints based on an iterative active-constraint generation technique. We note that the total number of the constraints is order of $O(n^2)$.

3.2.2 EDM-based optimization reformulation

Let Y be such an EDM that it satisfies

$$Y_{ij} \approx \widehat{d}_{ij}^2, \quad \forall (i,j) \in \mathcal{N}_x \text{ or } \mathcal{N}_a, \tag{3.4a}$$

$$Y_{ij} > R^2, \quad \forall (i,j) \in \overline{\mathcal{N}}_x \text{ or } \overline{\mathcal{N}}_a, \tag{3.4b}$$

$$Y_{ij} = \|\mathbf{a}_i - \mathbf{a}_j\|^2, \quad \forall i,j \in \{1,2,\ldots,m\}, \tag{3.4c}$$

$$\operatorname{rank}(-JYJ) = r. \tag{3.4d}$$

The first two constraints in (3.4) are to accommodate the constraints in (3.3). The third constraint includes the fixed distances among the m anchors and the last constraint enforces that the embedding points from Y should be in the low-dimensional space $\mathbb{R}^r$. Our EDM optimization seeks for a best Y from all matrices of satisfying those constraints in (3.4).

For the constraints in (3.4a), we use the principle of least-squares to get our objective. For other constraints, we keep them. This leads to the following optimization problem:

$$\begin{array}{ll} \min & \sum_{(i,j)\in\mathcal{N}_x\cup\mathcal{N}_a} \left(Y_{ij} - \widehat{d}_{ij}^2\right)^2 \\ \text{s.t.} & Y_{ij} \geq R^2,\ \forall (i,j) \in \overline{\mathcal{N}}_x \cup \overline{\mathcal{N}}_a \\ & Y_{ij} = \|\mathbf{a}_i - \mathbf{a}_j\|^2,\ \forall\, i,j \in \{1,2,\ldots,m\} \\ & \operatorname{rank}(-JYJ) = r,\ Y \in (-\mathcal{K}_+^n) \cap \mathcal{S}_h^n. \end{array} \tag{3.5}$$

Compared to problem (1.21), (3.5) has additional equality and extra inequality constraints. For this problem to be well defined, we consider the graph $\mathcal{G}$ formed by n nodes with its edges being $(i,j) \in \mathcal{N}_x \cup \mathcal{N}_a$. We assume that $\mathcal{G}$ is connected. Otherwise, problem (3.5) can be separated into smaller subproblems each corresponding to a connected subgraph in $\mathcal{G}$. The following result ensures that our problem is well defined and its effective region which contains all optimal solutions can be bounded.

Proposition 3.1. *Assume that $\mathcal{G}$ is connected. Then the optimal solution of (3.5) is attained. Moreover, the effective feasible region that contains all optimal solutions is bounded.*

Proof. Consider the following embedding points $\mathbf{x}_i = \mathbf{a}_i$, for $i = 1,\ldots,m$ and $\mathbf{x}_i = i(R+c)e$, for $i = m+1,\ldots,n$, where $e \in \mathbb{R}^r$ is the vector of all ones and $c := \max\{\|\mathbf{a}_i\|\}$. Let D be the corresponding EDM generated by those points in $\mathbb{R}^r$. It is easy to verify that all constraints in (3.5) are satisfied. For example, for

$(i,j) \in \overline{\mathcal{N}}_x$, we have

$$\|\mathbf{x}_i - \mathbf{x}_j\| = (R+c)\sqrt{r} \geq R,$$

and for $(i,k) \in \overline{\mathcal{N}}_a$, we have by triangle inequality

$$\|\mathbf{x}_i - \mathbf{x}_k\| \geq \|\mathbf{x}_i\| - \|\mathbf{x}_k\| \geq i\sqrt{r}(R+c) - c \geq R.$$

The rank of $(-JDJ)$ obviously is r. Hence, the feasible region is nonempty.

Let f_{opt} be the optimal objective value of (3.5). Then

$$f_{\text{opt}} \leq f_{\text{bound}} := \sum_{(i,j)\in\mathcal{N}_x\cup\mathcal{N}_a} \left(D_{ij} - \widehat{d}_{ij}^2\right)^2,$$

where D is any (fixed) feasible point. Let

$$\Omega := \left\{ D \in \mathcal{S}^n \mid D \geq 0,\ D_{ij} \leq \widehat{d}_{ij}^2 + c, \forall (i,j) \in \mathcal{N}_x \cup \mathcal{N}_a \right\},$$

where $c \geq \sqrt{f_{\text{bound}}}$ is a sufficiently large number so that it intersects the feasible region of (3.5). It is easy to see that any optimal solution (if exists) would have to be in Ω. Otherwise the optimal objective will be larger than f_{bound}. Hence, the effective region $\mathcal{F} \cap \Omega$ contains all the optimal solutions, where $\mathcal{F}$ denotes the feasible region of (3.5). Now consider any matrix D in this effective region. For any pair of nodes (i,j), the distance d_{ij} is bounded by the shortest path in the graph $\mathcal{G}$ since $\mathcal{G}$ is connected. This means that D is bounded by the elements in Ω. Hence, the effective region is bounded and closed. The optimal solution of (3.5) is attained. □

Let us temporarily ignore the rank constraint (3.4d). Then problem (3.5) becomes convex. We put it in a more general and compact form. Define symmetric matrices

$\widehat{D}, H \in \mathcal{S}^n$ respectively by

$$\widehat{D}_{ij} := \begin{cases} \widehat{d}_{ij}^2, & \text{if } (i,j) \in \mathcal{N}_x \cup \mathcal{N}_a, \\ 0, & \text{otherwise}, \end{cases}$$
$$H_{ij} := \begin{cases} 1, & \text{if } (i,j) \in \mathcal{N}_x \cup \mathcal{N}_a, \\ 0, & \text{otherwise}, \end{cases} \tag{3.6}$$

and two matrices $L, U \in \mathcal{S}^n$ being lower bound and upper bound for D respectively as

$$L_{ij} := \begin{cases} \|\mathbf{a}_i - \mathbf{a}_j\|^2, & \text{if } i,j \in \{1,2,\ldots,m\} \\ R^2, & \text{if } (i,j) \in \overline{\mathcal{N}}_x \cup \overline{\mathcal{N}}_a, \\ 0, & \text{otherwise}, \end{cases}$$
$$U_{ij} := \begin{cases} \|\mathbf{a}_i - \mathbf{a}_j\|^2, & \text{if } i,j \in \{1,2,\ldots,m\} \\ R^2, & \text{if } (i,j) \in \mathcal{N}_x \cup \mathcal{N}_a, \\ M^2, & \text{otherwise}, \end{cases} \tag{3.7}$$

where M is a large number (e.g., $M = n\max\{\widehat{d}_{ij}, R, \|\mathbf{a}_i\|, c\}$, where c is the constant used in the proof of Proposition 3.1). Let "$\circ$" denote the Hadamard (componentwise) product of two matrices of same size, the convex relaxation of problem (3.5) takes the form:

$$\begin{array}{ll} \min_{Y \in \mathcal{S}^n} & \frac{1}{2}\|H \circ (Y - \widehat{D})\|^2 \\ \text{s.t.} & L \le Y \le U, \\ & Y \in \mathcal{S}_h^n, \ -Y \in \mathcal{K}_+^n. \end{array} \tag{3.8}$$

Compared to problem (1.21), (3.8) has extra lower and upper bounds constraints and the number of them is of the order $O(n^2)$. These inequality constraint will bring the semismooth Newton's method used in the last chapter into difficulty

since the Lagrangian dual problem of (3.8) is no longer unconstrained.

3.2.3 Regularization to dealing with the rank constraint

Another potential issue for the convex model (3.8) is from dropping the rank constraint. In order for those distances in Y to be "closer" to the distances in $\widehat{D}$, the points would have to be embedded in a higher dimensional space. The found positions can then be seen as the projections from this higher dimensional embedding, resulting in a network in which the sensors always tend to be crowded around the center. Weinberger and Saul (2006) proposed a strategy for the embedding points $\mathbf{x}_i$ to be pushed away from each other. In fact, the term $\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|^2$ (called the variance in Weinberger and Saul (2006)) is maximized. Biswas *et al.* also used the similar idea in Biswas et al. (2006) to add a regularization term in their SDP model.

Let us explain this term a bit more. In the SDP models in both Weinberger and Saul (2006); Biswas et al. (2006), the positive semidefinite (Gram) matrix $G \in \mathcal{S}^n_+$ takes the form, as defined in (1.11):

$$G = X^T X, \qquad \text{with } X := [\mathbf{x}_1, \ldots, \mathbf{x}_n],$$

under the condition that those points are centralized

$$\mathbf{x}_1 + \ldots + \mathbf{x}_n = 0. \tag{3.9}$$

Then it is easy to verify that

$$\sum_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|^2 = 2n \sum_{i=1}^{n} \|\mathbf{x}_i\|^2 = 2n\langle I, G\rangle = 2n\mathrm{Tr}(G).$$

Using the above fact, the minimization SDP models in Weinberger and Saul (2006); Biswas et al. (2006) add the term $-\mathrm{Tr}(Y)$ to their objective. In other words, they try to maximize the trace. In convex optimization, it is widely known (Candès and Recht, 2009) that maximizing trace intends to maximize the rank since trace can be used in the nuclear norm minimization in positive semidefinite programming, which is a heuristic introduced by Fazel (2002) to minimize the rank. Thus it is a contradiction to the original purpose of minimizing the rank.

Now coming back to the distances, we know from (1.12) that the embedding points in X satisfy

$$2\mathrm{Tr}(G) = 2\mathrm{Tr}(X^TX) = \mathrm{Tr}(-JYJ) = -\langle J,\ Y\rangle, \tag{3.10}$$

where we used the fact $J^2 = J$. We also note that it follows from (1.12) that the centralization condition (3.9) is met for those points in X.

Based on the reasoning above, maximizing the variance (trace) can only be interpreted as pushing the embedding points far from each other and cannot be used to minimize rank. We introduce below a new technique to reduce the rank. Our intention here, under the condition that the variance $\mathrm{Tr}(-JYJ)$ is made as large as possible, is to decrease the rank of $(-JYJ)$. To describe the technique, we need the following Fan's inequality.

Theorem 3.2. *(Borwein and Lewis (2000, Thm. 1.2.1)) Any matrices X and Y in $\mathcal{S}^n$ satisfy the inequality*

$$\mathrm{Tr}(XY) \le \lambda(X)^T\lambda(Y). \tag{3.11}$$

Equality holds if and only if X and Y have a simultaneous ordered spectral decomposition: there is a matrix $U \in \mathcal{O}^n$ with

$$X = U^T(\mathrm{Diag}\lambda(X))U \quad \textit{and} \quad Y = U^T(\mathrm{Diag}\lambda(Y))U. \tag{3.12}$$

It follows from Theorem 3.2 that

$$\langle V,\ -JYJ\rangle \le \sum_{i=1}^{r} \mu_i(V)\lambda_i(-JYJ),$$

where $\mu_i(V) \ge \ldots \ge \mu_r(V)$ and $\lambda_1(-JYJ) \ge \ldots \ge \lambda_n(-JYJ)$ are respectively the eigenvalues of V and $(-JYJ)$ in nonincreasing order. And the equality holds if and only if both V and $(-JYJ)$ achieve simultaneous ordered spectral decomposition. If we maximize the term $\langle V,\ -JYJ\rangle$ under the constraints in (3.5) (recall its feasible region is bounded from Proposition 3.1), then the matrix $(-JYJ)$ is likely to be a low-rank matrix.

Let us consider the possible choice of V. Suppose we have an initial solution $\widetilde{Y}$, which, for example, may be obtained by the shortest path distances in $\mathcal{G}$, as used in MDS-MAP. Let

$$V = P_1 P_1^T \qquad \text{with} \quad P_1 = [\mathbf{p}_1, \ldots, \mathbf{p}_r],$$

where $\mathbf{p}_i$ is the orthogonal eigenvector of $(-J\widetilde{Y}J)$ corresponding to its ith largest positive eigenvalue. Since e is an eigenvector of $(-J\widetilde{Y}J)$ due to $Je = 0$, we have

$$JVJ = \Big(I - \frac{1}{n}ee^T\Big)P_1P_1^T\Big(I - \frac{1}{n}ee\Big) = P_1P_1^T = V.$$

Hence,

$$\langle V,\ -JYJ\rangle = \langle JVJ,\ -Y\rangle = \langle V,\ -Y\rangle. \tag{3.13}$$

By subtracting the regularization term (3.10) (to achieve maximal variance) and the term (3.13) (to reduce the rank) from the objective function of (3.8), we have

our final convex optimization model:

$$\begin{array}{ll} \min_{Y\in\mathcal{S}^n} & \frac{1}{2}\|H\circ(Y-\widehat{D})\|^2+\nu\langle J,Y\rangle+\langle V,Y\rangle \\ \text{s.t.} & L\le Y\le U, \\ & Y\in\mathcal{S}_h^n,\ -Y\in\mathcal{K}_+^n, \end{array} \tag{3.14}$$

where $\nu \geq 1$ is the parameter that balances the trade-off between preserving the local distances, maximizing the variance, and reducing the rank. Our numerical experiment will show that the solution compared to (3.8) always enjoys the low rank property.

3.2.4 Global coordinates recovery and EDM-SNL scheme

After obtaining a solution $\overline{Y}$ of (3.14), cMDS in Algorithm 1 is then applied to obtain the coordinates of the sensors. If there are anchors ($m > 0$), then the matching procedure introduced in Section 1.1.3 must be carried out to place the found coordinates to the coordinate system used by the anchors.

Integrated all the procedures, we summarize our EDM-based localization scheme as algorithm EDM-SNL below.

Algorithm 8 EDM-SNL

1: **Distance Reconstruction:** Set $\widehat{D}$ and H by (3.6), L and U by (3.7) as input data, solve (3.14) to get estimated full distance matrix $\overline{Y}$.
2: **cMDS:** Compute the sensor positions $\overline{X}$ by cMDS Algorithm 1
3: **if** Anchors exist **then**
4: Solve (1.14) to get rotation matrix Q and compute sensor positions by (1.15). **Return** X_2 as recovered sensor coordinates.
5: **end if**

In the next section, we will show that problem (3.14) in the first step of the above algorithm can be efficiently solved by an alternating minimization method.

3.3 A convergent 3-Block ADMM algorithm

The key difference of the problem (3.14) from (1.21) is that (3.14) has a large number of lower and upper bounds constraints. Coupled with the conic constraints $Y \in (-\mathcal{K}_+^n) \cap \mathcal{S}_h^n$, those key differences make the existing methods (Qi, 2013; Qi and Yuan, 2014) for (1.21) not applicable anymore since the dual problem is no longer unconstrained and we may lose the property of constraint nondegeneracy. Here, we choose to apply a recent ADMM (alternating direction method of multipliers) of Sun *et al.* in Algorithm 6 for general convex quadratic programming. To describe this method, we need to reformulate (3.14).

3.3.1 Reformulation and Lagrangian dual problem

Let $\mathcal{N} := \mathcal{N}_x \cup \mathcal{N}_a$. Note that $\mathcal{N}_x \cap \mathcal{N}_a = \emptyset$, so we have $|\mathcal{N}| = |\mathcal{N}_x| + |\mathcal{N}_a|$, where $|\mathcal{N}|$ is the cardinality of set $\mathcal{N}$. To simplify our notation, we denote $|\mathcal{N}| = N$ and

$$\mathcal{N} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_N, j_N)\}.$$

Define the linear operator $\mathcal{A}_1 : \mathcal{S}^n \to \mathbb{R}^N$ by

$$\mathcal{A}_1(X) := (\langle A_1, X\rangle, \langle A_2, X\rangle, \ldots, \langle A_N, X\rangle)^T, \tag{3.15}$$

where A_k is the symmetric selection matrix defined as

$$A_k := \frac{1}{2}(e_{i_k} e_{j_k}^T + e_{j_k} e_{i_k}^T), \ (i_k, j_k) \in \mathcal{N},$$

and e_i is the ith column vector of the identity matrix. Apparently, we have $\langle A_k, Y\rangle = Y_{i_k j_k}$ (hence the name of the selection matrix). Furthermore, let

$\mathbf{b}_1 = \mathcal{A}_1(\widehat{D})$. We then have

$$\|H \circ (Y - \widehat{D})\|^2 = \|\mathcal{A}_1(Y) - \mathbf{b}_1\|^2.$$

We further define linear operator $\mathcal{A}_2 : \mathcal{S}^n \to \mathbb{R}^n$, the matrix $C \in \mathcal{S}^n$, and closed convex set $\mathcal{C}$ respectively as

$$\mathcal{A}_2(X) := \mathrm{diag}(X), \quad C := \nu J + V,$$

and

$$\mathcal{C} := \{W \in \mathcal{S}^n : L \leq W \leq U\},$$

and the problem (3.14) can be equivalently written as

$$\begin{array}{ll} \min_{Y \in \mathcal{S}^n} & \frac{1}{2}\|\mathcal{A}_1(Y) - \mathbf{b}_1\|^2 + \langle C,\ Y\rangle \\ \text{s.t.} & \mathcal{A}_2(Y) = 0, \\ & Y \in \mathcal{C}, \ \ -Y \in \mathcal{K}_+^n. \end{array} \tag{3.16}$$

To derive the Lagrangian dual of (3.16), we introduce a new variable $\mathbf{t} := \mathcal{A}_1(Y) - \mathbf{b}_1 \in \mathbb{R}^N$ and a slack variable $W \in \mathcal{S}^n$, then (3.16) is equivalent to

$$\begin{array}{ll} \min\limits_{Y \in \mathcal{S}^n, \mathbf{t} \in \mathbb{R}^N, W \in \mathcal{S}^n} & \frac{1}{2}\|\mathbf{t}\|^2 + \langle C, Y\rangle + \delta_{\mathcal{C}}(W) + \delta_{\mathcal{K}_+^n}(-Y) \\ \text{s.t.} & \mathcal{A}(Y) - B\mathbf{t} = \mathbf{b}, \\ & Y = W, \end{array} \tag{3.17}$$

where $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^N \times \mathbb{R}^n$ is the linear mapping defined by

$$\mathcal{A}(Y) := (\mathcal{A}_1(Y); \mathcal{A}_2(Y)), \quad \mathbf{b} := (\mathbf{b}_1; \mathbf{0}^n) \in \mathbb{R}^{N+n},$$

and $B := [I_N; \mathbf{0}^{n \times N}] \in \mathbb{R}^{(N+n) \times N}$, I_N is the identity matrix with order N, $\mathbf{0}^{n \times N}$ is the matrix of all zeroes with order $n \times N$. For any given set Ω, $\delta_\Omega(\cdot)$ is the

indicator function over Ω such that $\delta_\Omega(u) = 0$ if $u \in \Omega$ and ∞ otherwise. We note here that directly solving problem (3.17) by ADMM is applicable since it fits the general convex quadratic conic programming (1.40), however, it has been discovered to be much less efficient than working on its dual problem.

Let the Lagrangian function of the above problem be defined by

$$\begin{aligned} L(Y, \mathbf{t}, W; \mathbf{y}, Z) \quad &:= \tfrac{1}{2}\|\mathbf{t}\|^2 + \langle C, Y\rangle + \delta_{\mathcal{C}}(W) + \delta_{\mathcal{K}_+^n}(-Y) \\ &\quad - \langle \mathcal{A}(Y) - B\mathbf{t} - \mathbf{b}, \mathbf{y}\rangle - \langle Y - W, Z\rangle\,, \end{aligned}$$

where $\mathbf{y} \in \mathbb{R}^{N+n}$ and $Z \in \mathcal{S}^n$ are the Lagrangian multipliers corresponding to the two constraints. The Lagrangian dual problem is

$$\max_{\mathbf{y},Z} \left\{ \min_{Y,\mathbf{t},W} L(Y, \mathbf{t}, W; \mathbf{y}, Z) \right\}. \tag{3.18}$$

The inner optimization problem can be solved with respect to t, Y, W separately. Let

$$\mathbf{t}^* = \arg\min_{\mathbf{t}} L(Y, \mathbf{t}, W; \mathbf{y}, Z),$$

then

$$\begin{aligned} \mathbf{t}^* &= \arg\min_{\mathbf{t}} \frac{1}{2}\|\mathbf{t}\|^2 + \langle B^T\mathbf{y}, \mathbf{t}\rangle \\ &= \arg\min_{\mathbf{t}} \frac{1}{2}\|\mathbf{t} + B^T\mathbf{y}\|^2 - \frac{1}{2}\|B^T\mathbf{y}\|^2 \\ &= -B^T\mathbf{y}. \end{aligned} \tag{3.19}$$

To minimize $L(Y, \mathbf{t}, W; \mathbf{y}, Z)$ regrading Y, note that

$$\min_{Y} L(Y, \mathbf{t}, W; \mathbf{y}, Z)$$

is equivalent to

$$\min_{Y} \delta_{\mathcal{K}_+^n}(-Y) + \langle C - \mathcal{A}^*\mathbf{y} - Z, Y\rangle$$

which implies that

$$\mathcal{A}^*\mathbf{y} + Z - C \in (\mathcal{K}_+^n)^*, \tag{3.20}$$

and the objective function is 0.

To minimize $L(Y, \mathbf{t}, W; \mathbf{y}, Z)$ regarding W is equivalent to

$$\min_{W} \delta_{\mathcal{C}}(W) + \langle W, Z\rangle. \tag{3.21}$$

Using (3.19), (3.20) and (3.21), the Lagrangian dual problem can be derived as

$$\begin{array}{ll} \max_{\mathbf{y},Z,S} & -\frac{1}{2}\|B^T\mathbf{y}\|^2 + \langle \mathbf{b}, \mathbf{y}\rangle - \delta_{\mathcal{C}}^*(-Z) \\ \text{s.t.} & Z + \mathcal{A}^*\mathbf{y} - S - C = 0, \\ & S \in (\mathcal{K}_+^n)^*, \end{array} \tag{3.22}$$

where $(\mathcal{K}_+^n)^*$ is the dual cone of $\mathcal{K}_+^n$ and $\delta_{\mathcal{C}}^*(\cdot)$ is the conjugate function of $\delta_{\mathcal{C}}(\cdot)$ given by

$$\delta_{\mathcal{C}}^*(-Z) = \sup_{W\in\mathcal{C}} \langle -Z, W\rangle = -\inf_{W\in\mathcal{C}} \langle Z, W\rangle\,. \tag{3.23}$$

We note that the insider optimization problem in (3.18) also gives the relationship between $\mathbf{t}$ and $\mathbf{y}$: $\mathbf{t} = -B^T\mathbf{y}$.

3.3.2 Implementation of 3-block ADMM

Problem (3.22) is well structured with three separable block variables. The SCB-SPADMM Algorithm 6 in Li et al. (2014) can be directly applied to get the optimal solution. The algorithm actually alternatively minimizes the associated Augmented Lagrangian defined by (note: here we cast (3.22) as a minimization

problem)

$$\begin{aligned} L_\sigma(\mathbf{y}, Z, S; Y) \;&:= \tfrac{1}{2}\|B^T\mathbf{y}\|^2 - \langle \mathbf{b}, \mathbf{y}\rangle + \delta^*_{\mathcal{C}}(-Z) \\ &\quad + \langle Z + \mathcal{A}^*\mathbf{y} - S - C, Y\rangle \\ &\quad + \tfrac{\sigma}{2}\|Z + \mathcal{A}^*\mathbf{y} - S - C\|^2, \end{aligned}$$

where $\sigma > 0$ is a given parameter and $Y \in \mathcal{S}^n$ is the Lagrangian multiplier corresponding to the equality constraint in (3.22). The proposed ADMM algorithm is described below.

Let $\sigma > 0$ and $\tau \in (0, \infty)$, set $(\mathbf{y}^0, Z^0, S^0, Y^0) \in \mathbb{R}^{N+n} \times \mathcal{S}^n \times \mathcal{S}^n \times \mathcal{S}^n$ as initial point. For iteration $k = 1, 2, \ldots$ perform the kth iteration as follows

$$\begin{cases} \bar{\mathbf{y}}^k = \arg\min_{\mathbf{y}\in\mathbb{R}^{N+n}} L_\sigma(y, Z^k, S^k; Y^k), \\ Z^{k+1} = \arg\min_{Z\in\mathcal{S}^n} L_\sigma(\bar{\mathbf{y}}^k, Z, S^k; Y^k), \\ \mathbf{y}^{k+1} = \arg\min_{y\in\mathbb{R}^{N+n}} L_\sigma(\mathbf{y}, Z^{k+1}, S^k; Y^k), \\ S^{k+1} = \arg\min_{S\in(\mathcal{K}^n_+)^*} L_\sigma(\mathbf{y}^{k+1}, Z^{k+1}, S; Y^k), \\ Y^{k+1} = Y^k + \tau\sigma\left(Z^{k+1} + \mathcal{A}^*\mathbf{y}^{k+1} - S^{k+1} - C\right), \\ \mathbf{t}^{k+1} = -B^T\mathbf{y}^{k+1}. \end{cases} \tag{3.24}$$

The convergence analysis of the algorithm above as well as the necessity of having the first update of $\bar{\mathbf{y}}$ have been already discussed in Section 1.3.2.

For any given set $\Omega \in \mathcal{S}^n$, we let $\Pi_\Omega(A)$ denote the orthogonal projection of $A \in \mathcal{S}^n$ onto Ω. The subproblems in (3.24) all have closed-form solutions involving orthogonal projections.

To update $\bar{\mathbf{y}}^k$, compute the gradient of Lagrangian function of $\mathbf{y}$ while other variables are fixed, we have

$$\nabla_{\mathbf{y}} L_\sigma(\mathbf{y}, Z^k, S^k; Y^k) = (BB^T + \sigma\mathcal{A}\mathcal{A}^*)\mathbf{y} - (b - \sigma\mathcal{A}(Z^k + \sigma^{-1}Y^k - S^k - C)).$$

By taking $\nabla_{\mathbf{y}} L_\sigma(\mathbf{y}, Z^k, S^k; Y^k) = 0$ we have

$$\bar{\mathbf{y}}^k = \left(BB^T + \sigma\mathcal{A}\mathcal{A}^*\right)^{-1}\left(\mathbf{b} - \sigma\mathcal{A}(Z^k + \sigma^{-1}Y^k - S^k - C)\right).$$

To update Z^{k+1}, we need to solve the following optimization problem:

$$\min_{Z\in\mathcal{S}^n} \delta_{\mathcal{C}}^*(-Z) + \frac{\sigma}{2}\|Z + \mathcal{A}^*\bar{\mathbf{y}}^k - S^k + \sigma^{-1}Y^k - C\|. \tag{3.25}$$

We need the following useful results from Moreau-Yosida regularization to derive the solution of the above optimization problem.

Proposition 3.3. *(Moreau, 1965; Yosida, 1995). Let $\mathcal{K}$ be a closed convex set and $\varphi(\bar{x}) := \min \delta_{\mathcal{K}}^*(-x) + \frac{\sigma}{2}\|x - \bar{x}\|^2$, the following results hold:*

(i) $x^+ = \arg\min \delta_{\mathcal{K}}^*(-x) + \frac{\sigma}{2}\|x - \bar{x}\|^2 = \bar{x} + \frac{1}{\sigma}\Pi_{\mathcal{K}}(-\sigma\bar{x})$.

(ii) $\nabla\varphi(\bar{x}) = \sigma(\bar{x} - x^+) = -\Pi_{\mathcal{K}}(-\sigma\bar{x})$.

(iii) $\varphi(\bar{x}) = \langle -x^+, \Pi_{\mathcal{K}}(-\sigma\bar{x})\rangle + \frac{1}{2\sigma}\|\Pi_{\mathcal{K}}(-\sigma\bar{x})\|^2 = -\langle\bar{x}, \Pi_{\mathcal{K}}(-\sigma\bar{x})\rangle - \frac{1}{2\sigma}\|\Pi_{\mathcal{K}}(-\sigma\bar{x})\|^2$.

Note that the objective function in problem (3.25) has exactly the same structure with the objective function of $\varphi(\bar{x})$ in Proposition 3.3. By using Proposition 3.3 *(i)*, we can immediately have

$$Z^{k+1} = S^k - \mathcal{A}^*\bar{\mathbf{y}}^k - \sigma^{-1}Y^k + C + \sigma^{-1}\Pi_{\mathcal{C}}(-\sigma(S^k - \mathcal{A}^*\bar{\mathbf{y}}^k - \sigma^{-1}Y^k + C)).$$

The update of $\mathbf{y}^{k+1}$ is similar to $\bar{\mathbf{y}}^k$ and the update of S^{K+1} is directly from the definition of projection function. We summarize the update process as follows:

$$\begin{aligned}
\bar{\mathbf{y}}^k &= \left(BB^T + \sigma\mathcal{A}\mathcal{A}^*\right)^{-1}\left(\mathbf{b} - \sigma\mathcal{A}(Z^k + \widehat{W}^k)\right) \\
Z^{k+1} &= \widehat{Z}^k + \sigma^{-1}\Pi_{\mathcal{C}}(-\sigma\widehat{Z}^k) \\
\mathbf{y}^{k+1} &= \left(BB^T + \sigma\mathcal{A}\mathcal{A}^*\right)^{-1}\left(\mathbf{b} - \sigma\mathcal{A}(Z^{k+1} + \widehat{W}^k)\right) \\
S^{k+1} &= \Pi_{(\mathcal{K}_+^n)^*}\left(Z^{k+1} + \mathcal{A}^*\mathbf{y}^{k+1} - C + \sigma^{-1}Y^k\right)
\end{aligned}$$

where $\widehat{W}^k := \sigma^{-1}Y^k - S^k - C$, and $\widehat{Z}^k := S^k - \mathcal{A}^*\bar{\mathbf{y}}^k - \sigma^{-1}Y^k + C$. Because $\mathcal{C}$ is a box-type constraints (lower and upper bounds), the projection $\Pi_{\mathcal{C}}(A)$ is easy to compute. For the projection $\Pi_{(\mathcal{K}_+^n)^*}(A)$, we use the Moreau decomposition formula:

$$\Pi_{(\mathcal{K}_+^n)^*}(A) = A + \Pi_{\mathcal{K}_+^n}(-A),$$

and the Gaffke-Mather formula (1.5). Hence, $\Pi_{(\mathcal{K}_+^n)^*}(A) = \Pi_{\mathcal{S}_+^n}(JAJ)$, which can be calculated through the eigen-decomposition of JAJ. We note that the matrix $(BB^T + \sigma\mathcal{A}\mathcal{A}^*)$ is a positive definite diagonal matrix in $\mathcal{S}^{N+n}$. The computation complexity of JAJ is in the order $O(n^2)$ because of the structure in J. Hence, the major computational complexity in (3.24) is dominated by the eigen-decomposition of JAJ in computing S^{k+1}.

We directly adopt the measurement in Li et al. (2014) on the accuracy of an approximate optimal solution for (3.16) and its dual (3.22) by using the following relative residual obtained from their general optimality conditions (KKT conditions):

$$\eta_e(\mathbf{t}, \mathbf{y}, Y, Z, S) = \max\left\{\eta_P, \eta_Y, \eta_Z, \eta_{C_1}, \eta_{C_2}\right\}, \tag{3.26}$$

where

$$\eta_P = \frac{\|\mathcal{A}(Y) - B\mathbf{t} - \mathbf{b}\|}{1 + \|\mathbf{b}\|}, \quad \eta_D = \frac{\|Z + \mathcal{A}^*\mathbf{y} - S - C\|}{1 + \|C\|},$$
$$\eta_Z = \frac{\|Y - \Pi_{\mathcal{C}}(Y - Z)\|}{1 + \|Y\| + \|Z\|}, \quad \eta_{C_1} = \frac{|\langle S, Y\rangle|}{1 + \|S\| + \|Y\|},$$
$$\eta_{C_2} = \frac{\|Y + \Pi_{\mathcal{K}_+^n}(-Y)\|}{1 + \|Y\|}.$$

We note that η_P measures the violation of the first equation constraint in the primal problem (3.17); η_D for the violation of the equation constraint in the dual problem (3.22); η_Z for the violation of Y belonging to $\mathcal{C}$; η_{C_1} measures the complementarity condition between S and D; and finally η_{C_2} measures the violation of $-Y$ belonging to $\mathcal{K}_+^n$. We terminate the algorithm if the maximum of the violations is below certain level, i.e.,

$$\eta_e(\mathbf{t}^k, \mathbf{y}^k, Y^k, Z^k, S^k) \leq \text{tol}, \tag{3.27}$$

where tol is a given tolerance.

3.4 Experimental results by EDM-SNL

In this section, we conducted numerical experiments using MATLAB (R2015a) on a desktop of 4GB memory and Intel(R) Core(TM) i5-2500 3.3GHz CPU to evaluate the performance of the proposed EDM-SNL, whose parameters are set as

$$\nu = 10, \ \text{tol} = 10^{-3},$$

and the initial points $\mathbf{y}^0, Z^0, S^0, Y^0$ are all zeros in the corresponding space. Below we first briefly discuss the numerical methods that we plan to compare with and why we choose them. We then generate three classes of test problems having

regular and irregular topologies. We finally present the comparison results, which show that EDM-SNL is overall more robust and faster on the tested problems.

3.4.1 Benchmark methods

We select two representative state-of-the-art methods: ARAP in Zhang et al. (2010) and SFSDP in Kim et al. (2012) as benchmark methods. ARAP is a MDS-based localization scheme that "stitches" together local structures of sensor nodes. Shown by the authors, ARAP outperforms other MDS-based methods such as SMACOF and MDS-MAP(P, R). SFSDP is a sparse version of the full SDP localization scheme proposed by Biswas and Ye (2004) and it achieves results with similar accuracy but using much less computation time. There are other methods that also produce comparable results, but their codes are not publicly available.

For ARAP, we use its default parameters. However, ARAP is mainly for the anchor-free localization problem (i.e., $m = 0$ in our case). But it can be used to deal with the case $m \neq 0$. One would have to do an extra step to place the generated localizations by ARAP to the coordinate system used by the anchors. The step is very similar to the one described in Section 3.2.4. For SFSDP, we keep all default parameters except the SDP solver option and the objective function option. We set *pars.SDPsolver* = 'sedumi' because by our observation, SeDuMi always gives us a solution with better accuracy than SDPA (a sover used by SFSDP for SDPs). We set $pars.objSW = 3$ to add the regularization term (the trace term) to the SDP objective function. It is reported in Biswas et al. (2006) that the use of regularization in SDP provides notable improvement for the networks with random anchor distribution. We believe in these settings, SFSDP would have the best performance in terms of localization accuracy.

Another important feature of SFSDP is its refinement step, which was previously used by Biswas et al. (2006). The refinement is a heuristic step that uses the

steepest gradient method to improve the quality of the final localization. The general view on the refinement step is that it more often than not improves the quality, but not always. For more detail on this, see Biswas et al. (2006). We will report both the results with and without the refinement step for all algorithms. However, it will be seen in Section 3.4.4 that when the network is in a square region, ARAP actually benefits little from the refinement step.

3.4.2 Test examples

We follow the standard framework in generating the noisy distances:

$$\widehat{d}_{ij} = d_{ij} \times |1 + nf \times randn|, \quad \forall\ (i,j) \in \mathcal{N}_x \cup \mathcal{N}_a,$$

where d_{ij} is the true Euclidean distance between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ which will be generated soon, $0 \leq nf \leq 1$ is the noise factor, and *randn* is the standard normal random variable. We generate $\mathbf{x}_i$ in the following three examples, the first having square region layout and the rest two having irregular topologies.

Example 3.1. *(Square Network) In a square region of* $[0, 100]^2$*, we randomly generate* n *points* $\mathbf{x}_i$ *following the uniform distribution. Two types of anchor positions are considered: placed at corners or randomly.*

Example 3.2. *(Corridor Network) We generate* n *points* $\mathbf{x}_i$ *following a uniform distribution in a region that looks like a corridor, as shown in Figure 3.2. The corridors are formed by creating two rectangular gaps inside the square* $[0, 100]^2$*.*

Example 3.3. *(EDM Network) We randomly generated* n *points* $\mathbf{x}_i$ *in a region whose shape is similar to the letter "E", "D" and "M". The ground truth network is depicted in Figure 3.3.*

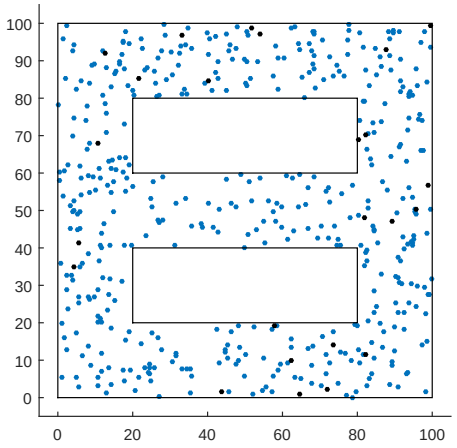


Figure 3.2: Ground truth *Corridor*network with $n = 494$ nodes in total, among which are $m = 24$ anchor nodes randomly distributed (colored in black).

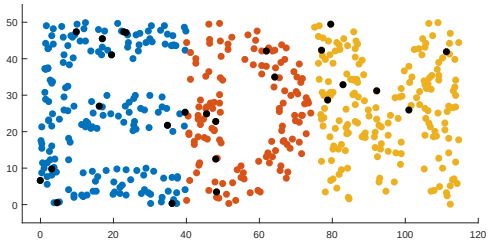


Figure 3.3: Ground truth *EDM*network with $n = 511$ nodes in total, among which are $m = 25$ anchor nodes randomly distributed (colored in black).

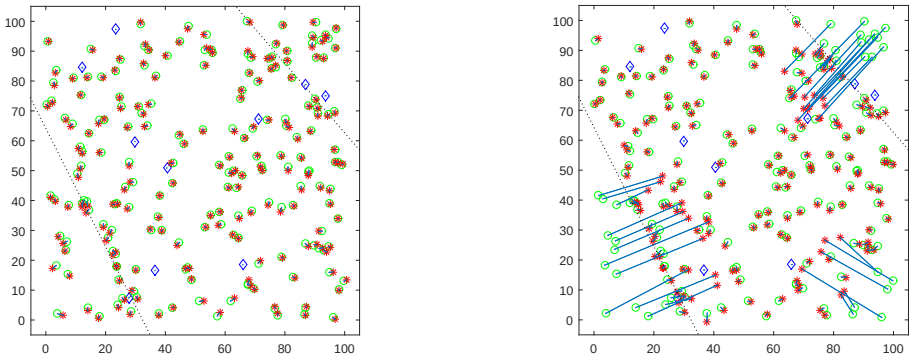
To measure the accuracy of the estimated positions, we simply adopt the commonly used root mean square distance (RMSD)

$$\text{RMSD} := \frac{1}{\sqrt{n-m}} \left(\sum_{i=m+1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 \right)^{\frac{1}{2}}, \tag{3.28}$$

where $\hat{\mathbf{x}}_i$ is the estimated position and $\mathbf{x}_i$ is the ground-truth sensor position.

3.4.3 Performance comparison on quality of localizations

Before we report more detailed numerical comparison, we would like to emphasize one important fact that the flip ambiguity does not always occur. However, when



(a) (b)

Figure 3.4: Network generated in Example 3.1 with $n = 200$ and $m = 10$ anchors randomly distributed. Noise factor $nf = 0.1$. Communication radius $R = 20$. (a) EDM-SNL: Localization Error $= 3.93\%R$. (b) SFSDP: Localization Error $= 52.62\%R$. (Blue diamond: anchor position. Green circle: original sensor position. Red star: estimate sensor position. Blue line: error offset between original and estimate sensor position.)

it occurs, it would severely degrade the localization of the whole network. A simulation result from randomly generated network is depicted in Figure 3.4 to illustrate the importance of flip ambiguity mitigation. As shown in Figure 3.4(b), the lower left and upper right parts of the sensors are completely folded over across the dotted line towards the center of the region, and causes a large localization error by SFSDP, while EDM-SNL recovers the sensor positions correctly in Figure 3.4(a).

Now we report our observation on the general performance of all methods using the three test problems. The results are the average over 50 randomly generated instances for each problem. Figure 3.5 depicts the variation of localization error (i.e. RMSD normalized by communication range R) from each method for Example 3.1 (with $n = 200$ and various anchor positions) when the noise in the distance measurements increases.

In Figure 3.5(a), four anchors are placed at four corners of the square region, i.e., $[0, 0]$, $[1, 100]$, $[100, 0]$ and $[100, 100]$. It can be observed that when the noise

factor is less than 0.3, all three methods performed similarly well. However, as the noise keeps increasing, EDM-SNL outperforms both ARAP and SFSDP in terms of localization accuracy, and controls the localization error around $50\%R$ even when the noise factor is as large as 0.9. When the anchor distribution type is changed to random, as shown in Figure 3.5(b), SFSDP got a large impact on its localization accuracy, while EDM-SNL and ARAP are steadily getting worse when the noise level gets bigger. Overall, EDM-SNL performed best. As expected, with more anchors being used, all three methods obtained better quality of localization and this can be observed from Figure 3.5(c) and Figure 3.5(d).

Now we test all the methods on Example 3.2 and Example 3.3, both of which have irregular topology. It is always important for a localization system to be resistance to irregular wireless sensor networks, because many applications require sensor network to be restricted in a certain region. The main challenge of this problem comes from its multiscale structure and nonconvex region. Such characteristics of irregular layout would cause large localization error for ARAP in its patching and stitching procedures when the distance measurements contain large noise. For example, for the corridor network in Example 3.2 (it contains total 494 nodes randomly (uniformly) distributed in a nonconvex region and $m = 24$ of them are anchors, and the communication radius of sensor/anchor nodes is $R = 12$), the localization error by three methods under different noise factors are compared in Figure 3.6. It can be observed that when the noise factor is less than 0.3, EDM-SNL and ARAP both work well and ARAP is slightly better. However, when the noise factor goes beyond 0.3, the performance of ARAP degrades dramatically, whereas EDM-SNL is still stable and kept the localization error under $30\%R$.

We further tested Example 3.3 with $n = 511$ and $m = 25$. The communication radius of sensor/anchor nodes is $R = 10$. Localization error comparison is shown in Figure 3.7. Like the results for Example 3.2, when the noise factor is less than 20%, all three methods perform well and ARAP is slightly better. When the

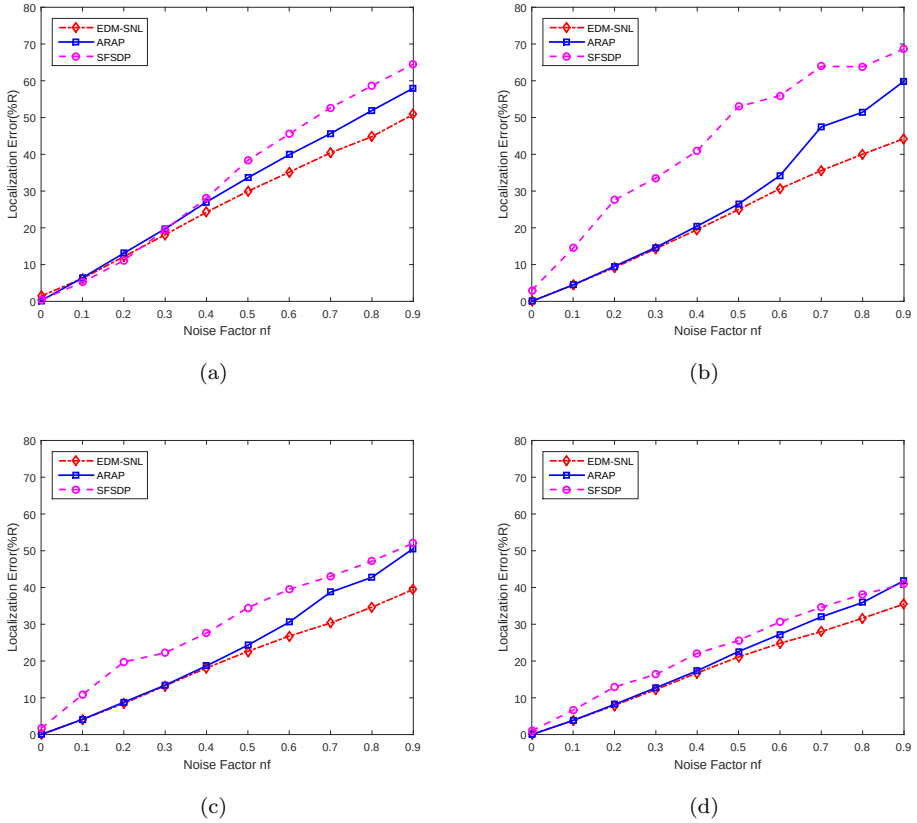


Figure 3.5: Variation of localization error with varying noise factor and anchor distribution type. Networks of totally 200 nodes with communication radius $R = 20$. (a) 4 anchor nodes distributed at corners. (b) 10 anchor nodes randomly (uniformly) distributed. (c) 20 anchor nodes randomly (uniformly) distributed. (d) 40 anchor nodes randomly (uniformly) distributed.

noise is greater than 0.5, our method still controls the error under $20\%R$, while ARAP and SDP recovered sensor positions with relatively large error. Qualitative results are shown in Figure 3.8. When the noise factor goes to as large as 0.6, EDM-SNL is still capable of generating sensor nodes that form the shape of the *EDM* network, especially for the rightmost letter ('M'), leading to the superior localization results. One of the reasons that make EDM-SNL robust to large noise measurement is that the upper bound constraint in (3.8) makes sure the distance between two sensor nodes that have communication with each other is less than the communication radius R, no matter how large the input distance is detected.

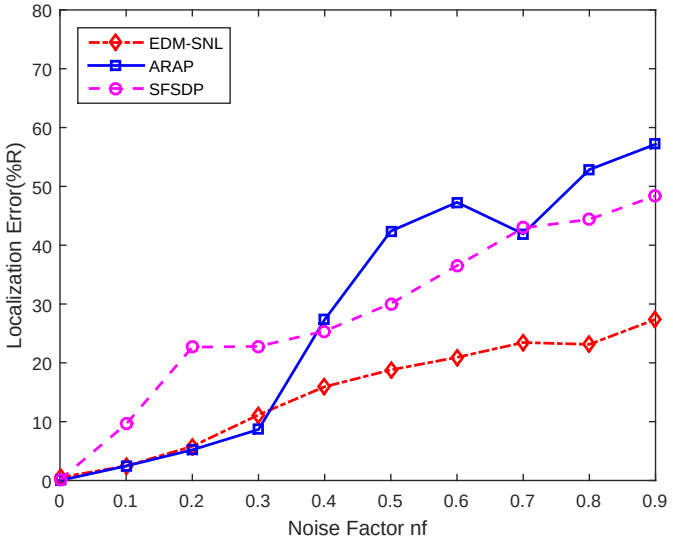


Figure 3.6: Variation of localization error with varying noise factor. Networks of totally 494 nodes, among which are 24 anchor nodes distributed randomly. Communication radius $R = 12$.

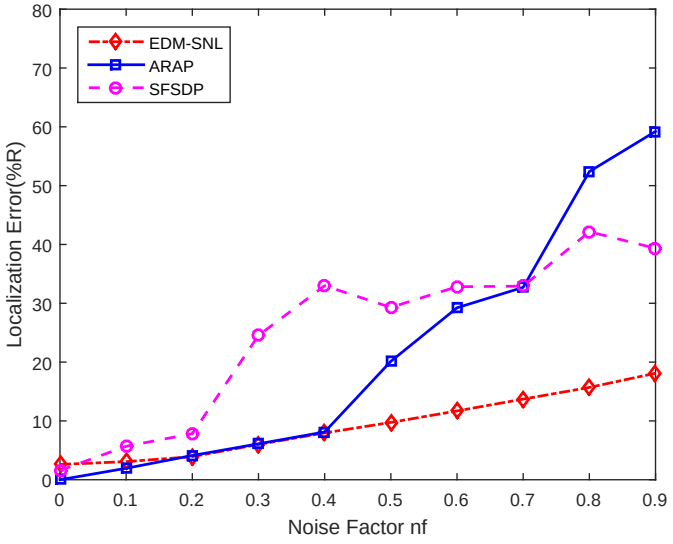


Figure 3.7: Variation of localization error with varying noise factor. Networks of totally 511 nodes, among which are 25 anchor nodes distributed randomly. Communication radius $R = 10$.

3.4.4 Performance comparison on computation time

We use Example 3.1 and Example 3.2 to test the computation efficiency of our EDM-based localization approach for networks with regular and irregular topology respectively, we run simulations on networks with various number of sensor

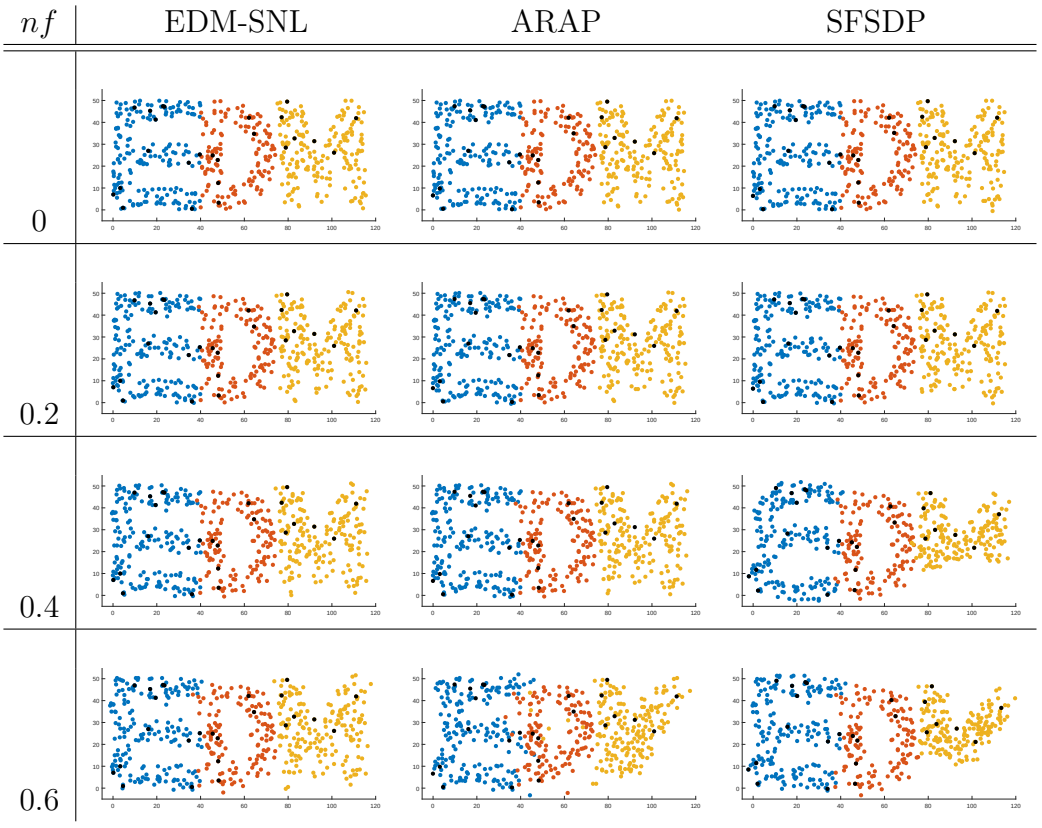


Figure 3.8: Qualitative localization results of *EDM* network, comparing EDM-SNL, ARAP, SFSDP with different noise level

and anchor nodes randomly (uniformly) distributed. The total number of (sensor/anchor) nodes n varies as $n = 200, 500$ and 1000, and the number of anchor nodes $m = 5\%, 10\%$ and 20% of n. The noise factor nf is fixed at $nf = 0.4$ and the communication radius R varies in order to control the average degree of nodes. For each specific network, we randomly generated 50 different inputs (sparse and noisy distance matrices) and the average results are reported in Table 3.1 for square networks and Table 3.2 for corridor networks, which contain the execution time and the corresponding localization error for all three methods before and after refinement step.

From both Table 3.1 and Table 3.2, it can be observed that the refinement step usually reduces the localization error by some extent, except for the EDM-SNL when $n = 500$ and $m = 5\%n,\ 10\%n$ in Table 3.1, and the additional computation time is only around 1 second. ARAP benefits the least from refinement among

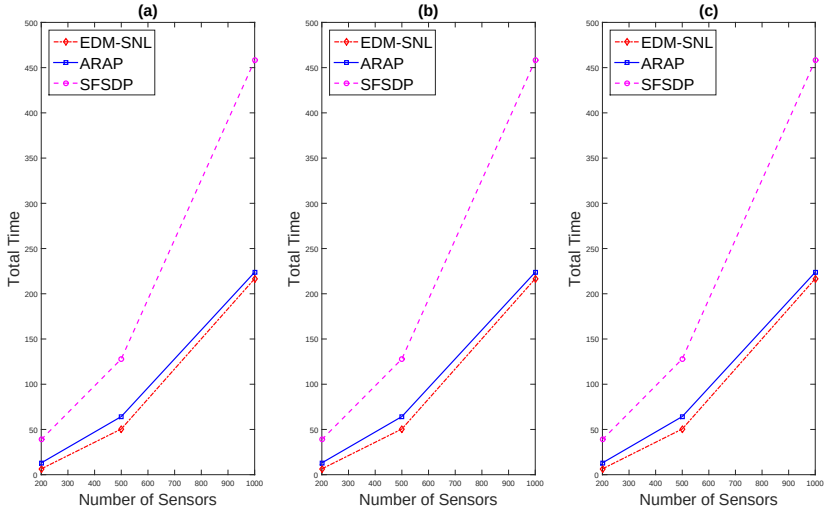


Figure 3.9: Computation time comparison in Square Network. (a) The number of anchors is 5% of the number of sensors. (b) The number of anchors is 10% of the number of sensors. (c) The number of anchors is 20% of the number of sensors.

three methods since it actually uses a stress majorization step, which is similar to the refinement we used here, in localizing each small patch to improve the local embedding in its whole process.

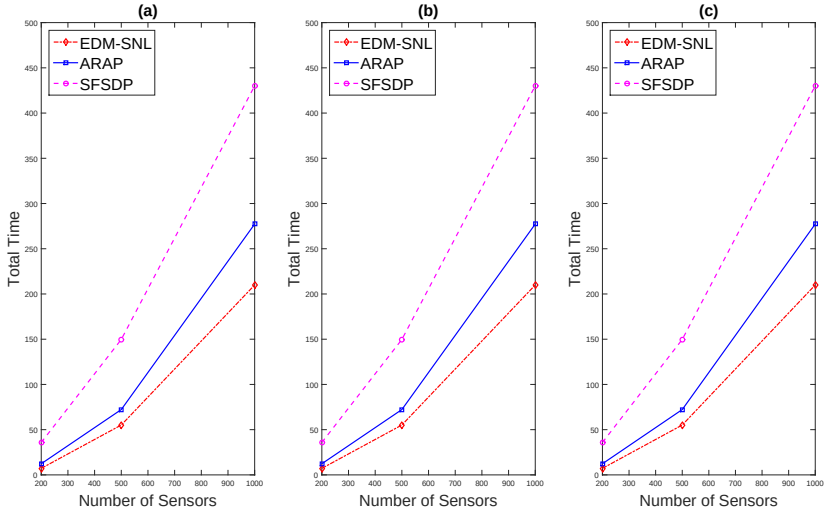


Figure 3.10: Computation time comparison in Corridor Networks. (a) The number of anchors is 5% of the number of sensors. (b) The number of anchors is 10% of the number of sensors. (c) The number of anchors is 20% of the number of sensors.

The computation time of both the EDM-SNL and SFSDP will benefit from the increasing number of anchor nodes since more prior knowledge is used in the localization scheme. It can be observed from Table 3.1 and Table 3.2 that our approach is the fastest among three methods with comparable values of RMSD for all cases tested. Figure 3.9 and Figure 3.10 depict the computation time comparison clearly. For each scenario our method (showed in red line) outperforms the other two methods on time consuming. For small networks, e.g. $n = 200$, our approach only took around 7 seconds to achieve a localization result with the lowest error among three methods. When the number of nodes goes to thousand, e.g. $n = 1000$ in Table 3.2, EDM-SNL used approximate 100 seconds less time than ARAP for the case $m = 20\%n$, and the difference of localization error from two schemes is only as small as 0.07.

Table 3.1: Execution Time Results of Square Network

Test Problems		Localization					
$n(R)$	m	Scheme	RMSD	RMSD.Re	Time	Time.Re	Total Time
200(20)	5%n	EDM-SNL	4.18E+00	3.74E+00	6.34	0.10	6.44
		ARAP	4.98E+00	3.90E+00	12.90	0.12	13.02
		SFSDP	1.47E+01	7.11E+00	38.71	0.14	38.85
	10%n	EDM-SNL	3.76E+00	3.44E+00	4.97	0.08	5.05
		ARAP	4.96E+00	3.57E+00	12.91	0.07	12.97
		SFSDP	1.30E+01	5.81E+00	34.28	0.08	34.36
	20%n	EDM-SNL	3.15E+00	3.12E+00	2.91	0.04	2.94
		ARAP	4.79E+00	3.35E+00	12.97	0.06	13.02
		SFSDP	1.12E+01	4.01E+00	24.40	0.04	24.44
500(15)	5%n	EDM-SNL	1.89E+00	1.98E+00	50.29	0.20	50.49
		ARAP	2.16E+00	1.99E+00	64.09	0.15	64.24
		SFSDP	1.19E+01	3.04E+00	127.47	0.43	127.91
	10%n	EDM-SNL	1.92E+00	1.89E+00	32.28	0.12	32.40
		ARAP	2.14E+00	1.90E+00	64.76	0.13	64.89
		SFSDP	9.81E+00	3.09E+00	125.10	0.22	125.32
	20%n	EDM-SNL	1.48E+00	1.81E+00	29.68	0.07	29.75
		ARAP	2.13E+00	1.81E+00	64.60	0.09	64.68
		SFSDP	8.85E+00	2.03E+00	88.08	0.14	88.22
1000(12)	5%n	EDM-SNL	3.24E+00	2.15E+00	216.08	0.56	216.64
		ARAP	1.48E+00	1.34E+00	223.56	0.26	223.82
		SFSDP	9.61E+00	2.44E+00	457.10	0.88	457.98
	10%n	EDM-SNL	3.03E+00	1.76E+00	168.87	0.49	169.36
		ARAP	1.47E+00	1.28E+00	223.36	0.24	223.60
		SFSDP	1.10E+01	1.92E+00	345.86	0.76	346.62
	20%n	EDM-SNL	3.72E+00	1.22E+00	146.34	0.21	146.55
		ARAP	1.47E+00	1.22E+00	223.30	0.16	223.46
		SFSDP	1.04E+01	1.56E+00	275.52	0.27	275.79

Table 3.2: Execution Time Results of Corridor Network

Test Problems		Localization					
$n(R)$	m	Scheme	RMSD	RMSD.Re	Time	Time.Re	Total Time
200(20)	5%n	EDM-SNL	4.41E+00	4.35E+00	7.09	0.12	7.21
		ARAP	1.25E+01	9.20E+00	12.18	0.14	12.32
		SFSDP	1.66E+01	9.29E+00	35.64	0.21	35.85
	10%n	EDM-SNL	3.76E+00	3.74E+00	4.85	0.08	4.93
		ARAP	1.21E+01	5.68E+00	12.13	0.11	12.24
		SFSDP	1.40E+01	7.24E+00	31.32	0.10	31.41
	20%n	EDM-SNL	3.50E+00	3.33E+00	3.35	0.06	3.41
		ARAP	1.20E+01	4.63E+00	12.15	0.06	12.22
		SFSDP	1.57E+01	4.86E+00	26.70	0.08	26.78
500(15)	5%n	EDM-SNL	3.11E+00	2.02E+00	54.70	0.42	55.12
		ARAP	2.63E+00	2.00E+00	71.86	0.35	72.21
		SFSDP	1.39E+01	4.91E+00	149.00	0.71	149.71
	10%n	EDM-SNL	3.56E+00	1.78E+00	36.36	0.28	36.65
		ARAP	2.59E+00	1.78E+00	71.55	0.28	71.82
		SFSDP	1.09E+01	2.51E+00	150.56	0.53	151.09
	20%n	EDM-SNL	2.15E+00	1.72E+00	32.01	0.13	32.13
		ARAP	2.57E+00	1.70E+00	71.71	0.13	71.84
		SFSDP	8.31E+00	2.01E+00	105.00	0.23	105.22
1000(12)	5%n	EDM-SNL	3.69E+00	1.44E+00	209.28	0.68	209.96
		ARAP	1.58E+00	1.49E+00	276.90	0.42	277.32
		SFSDP	9.81E+00	1.90E+00	428.95	1.13	430.08
	10%n	EDM-SNL	3.62E+00	1.23E+00	147.77	0.41	148.18
		ARAP	1.58E+00	1.45E+00	276.83	0.28	277.12
		SFSDP	7.77E+00	1.52E+00	421.06	0.54	421.60
	20%n	EDM-SNL	5.92E+00	1.18E+00	161.07	0.29	161.36
		ARAP	1.57E+00	1.11E+00	276.84	0.19	277.03
		SFSDP	1.09E+01	1.42E+00	312.34	0.38	312.73

3.5 Summary

In this section, we introduced a new scheme for wireless sensor network localization problem based on the Euclidean distance matrix. A conic programming built upon the $\mathcal{K}_+^n$ cone is solved by a recently developed alternating direction method of multipliers to retrieve missing distances between nodes. Then classical multidimensional and Procrustes analysis are applied to recover the node positions. By modelling the problem based on EDM, inequality constraints are integrated in a simple closed convex set, resulting in a localization scheme that acquires both efficiency and robustness towards the existence of large noise. Numerical experiments showed the EDM-based Localization scheme outperforms existing state-of-the-art ARAP and SFSDP schemes.

The major computation in EDM-SNL is the spectral decomposition of a symmetric matrix, which could be fatal to the scheme when the number of nodes in the network goes beyond thousands. For the large scale networks consisting tens of thousands nodes, following the paradigm of "think globally, fit locally", the Euclidean distance matrix could be completed part by part, thus a robust EDM-based localization scheme for large-scale networks could be developed.

Chapter 4

Implication for large-scale network and future work

When dealing with large-scale distance geometry problems of, for example, 10000 points or more, we may face a big challenge in algorithm design due to the limitation of the capacity of PC hardware such as CPU and memory. For instance, a single execution of eigenvalue decomposition of a 10000×10000 symmetric matrix would take around 200 seconds by MATLAB on a normal computer with 3.30GHz CPU and 4GB memory. This almost closes the door to use the alternating direction method of multipliers which always involves projection operations in matrix conic programming. Large-scale networks exist in many application areas such as molecular conformation (Crippen et al., 1988; Havel and Wüthrich, 1985), sensor network localization (Biswas and Ye, 2004; Biswas et al., 2006; Wang et al., 2008) and dimensionality reduction (Weinberger et al., 2006, 2004), and large-scale data analysis is the new trend in future research. In this chapter, we focus on the large-scale network localization problem and the framework can be extended to other application areas with minor modification. We first introduce background on a class of patching methods for large-scale network localization problem, then we

describe our proposal for dealing with such problems and some primary results. Finally we summarize the issues to be investigated in the future.

4.1 Backgroud on patching method

Given a large-scale network which can always be represented as an undirected graph with vertices V and edges E, as described in Chapter 3, a localization problem is to find the (relative or global) coordinates of vertices based on (complete or incomplete) pairwise distance information, which is always noisy. We already know for network of medium size, EDM-SNL can recover the positions of vertices robustly and efficiently. When comes to the large-scale network, a direct idea is to divide the network into small subnetworks, called *patch* in some literatures, then localize each patch separately, finally "stitch" them together to get the final network localization.

The idea comes from the distributed sensor network localization, in which each node calculates its own position based on the local data gathered from its neighbours. Savarese et al. (2001) and Čapkun et al. (2002) treat every point and its R-hop neighbours (R usually takes 1 or 2) as a subnetwork, then use triangulation or geometry to localize each subnetwork. By using anchor information, Savarese et al. (2001) managed to get global coordinates of each point. Čapkun et al. (2002) only provided a method to retrieve relative positions in GPS-free mobile ad hoc networks. Using triangulation and geometry, methods in Savarese et al. (2001); Čapkun et al. (2002) will result in positions of points with large error when the distance information is highly noisy. Moreover, such distributed framework requires that each point has computation capacity, which brings higher requirement to the sensor hardware.

Based on the work in Savarese et al. (2001); Čapkun et al. (2002), Shang and Ruml (2004) proposed a MDS-based localization scheme called MDS-MAP(P), which uses classical MDS to localize each patch instead of triangulation. For a network with n points, MDS-MAP(P) first generates local map for each node by its R-hop neighbours, then computes the local map position for each individual point. To localize the patches, shortest paths between all pairs of nodes in the patch are computed, and classical MDS is conducted to get relative coordinates of each point. After getting all relative positions of patches, the next work is to merge the patches to get the whole network. To do that, they random choose a patch as the core network, then merge its neighbour patches one by one using Procrustes analysis introduced in Section 1.1.3, until the whole network is recovered. A least square minimization is also used to refine the localization for both the patches and whole network. And the global coordinates are retrieved if sufficient anchor nodes exist. The main purpose of MDS-MAP(P) is to compensate the drawbacks of MDS-MAP (Shang et al., 2003) for networks with irregular topology, in which the shortest path can not estimate the true distance between far away points very well. By dividing the whole network into patches, the authors believe that each small patch is relatively regular and shortest path can serve well as an estimation of missing distances. Stitching the patches incrementally to the global coordinate system seems to be sound, however, unfortunately, it may accumulate error indefinitely, especially when the pairwise distances are highly noisy.

Several stitching strategies are proposed to reduce the error that incremental algorithms could bring, resulting in different localization schemes. One of which is PATCHWORK introduced by Koren et al. (2005). Unlike MDS-MAP(P), for each patch PATCHWORK estimates the missing distances by the upper bound and lower bound. Denote D_{ij} as the distance between node $\mathbf{x}_i$ and $\mathbf{x}_j$, for every

$(i,j) \notin E$, from the triangle inequality, the upper bound $\overline{D}_{ij}$ is obtained by

$$\overline{D}_{ij} = \min_{k:(i,k)\in E,(j,k)\in E} \{D_{ik} + D_{jk}\}.$$

Moreover, for the disk graph, the lower bound $\underline{D}_{ij}$ is given by

$$\underline{D}_{ij} = \max\{\max_{k:(i,k)\in E} \{D_{ik}\}, \max_{k:(j,k)\in E} \{D_{jk}\}\}.$$

Then D_{ij} is estimated as $(\overline{D}_{ij} + \underline{D}_{ij})/2$. After retrieving all pairwise distance information, classical MDS is used to get relative coordinates of the points in each patch. To stitch all patches together, they propose to work with affine transformations that not only include rigid transformations, but also has additional operations such as scaling and shear. Using affine transformations allows compensating for errors introduced by noise.

Let $(V(s), E(s))$ be a patch that centered at point s, $\overline{X}(s) \in \mathbb{R}^{n_s \times (r+1)}$ be the collection of locations of the points in the patch. The ith row of $\overline{X}(s)$ represents the ith point, where the first r entries are its r coordinates and the last entry is always 1. Denote $X(s)$ as the corresponding coordinates in the global coordinate system, then the affine transformation $Q(s) \in \mathbb{R}^{(r+1)\times(r+1)}$ gives the relation as $\overline{X}(s)Q(s) \approx X(s)$. The best least square solution is

$$Q(s) = \overline{X}(s)^{+} X(s),$$

where $\overline{X}(s)^{+}$ is the pseudo-inverse. Then the global coordinates can be recovered by solving the minimization problem

$$\min_{X(s)} \|X(s) - \overline{X}(s)Q(s)\|^2,$$

which is identical to

$$\min_{X(s)} \|X(s) - (\overline{X}(s)\overline{X}(s)^{+})X(s)\|^2. \tag{4.1}$$

Koren et al. (2005) proposed to solve the above optimization problem, which is equivalent to solving a linear equation system. They also introduced how to use anchor position information to avoid degenerate solutions of the linear equation system.

Instead of using affine transformation, Zhang et al. (2010) used rigid transformation, which is efficient to preserve better local relationships between patches, but has the disadvantage of resulting in a nonlinear system of equations. The stitching process is modelled to solve the following optimization problem simultaneously:

$$(X^*, Q_i^*, \ldots, Q_N^*) = \arg \min_{X, Q_i, \ldots, Q_N} \{\sum_{i=1}^{N} \|X_i - Q_i \overline{X}_i\|_F^2 : Q_i^T Q_i = I\}, \tag{4.2}$$

where N is the number of patch, X is the collection of global coordinates of points in each patch, $Q_i, i = 1, \ldots, N$ are the rigid transformations that transform patch i into the global coordinate system, and $\overline{X}_i$ is the local (relative) coordinates of points in patch i. To solve (4.2), Zhang *et al.* proposed to use a two-phase Alternating Least-Square (ALS) method. In the first phase, X is fixed and they solve (4.2) for each Q_i. In the second phase, all Q_i are assumed to be fixed, and they solve (4.2) for X. Based on ALS, they developed a MATLAB code called ARAP, however, numerical experiment shows that ARAP will provide large error solution when the distance information is very noisy, especially for the networks with irregular topology.

To compensate the error that occurs in the stitching process, a convex relaxation is proposed by Chaudhury et al. (2013, 2015), in which a semidefinite programming is built up to merge patches together. In their paper, the merge process is call

global registration. Let $\mathbf{x}_{k,i}$ be the relative coordinate of sensor $\mathbf{x}_k$ in the ith patch, the registration is to find some orthogonal transform Q_i and translation $\mathbf{t}_i$ such that

$$\mathbf{x}_k = Q_i \mathbf{x}_{k,i} + \mathbf{t}_i.$$

For network with anchors, an additional equation should be satisfied as follows:

$$\mathbf{a}_l = Q_i \mathbf{a}_l + \mathbf{t}_i,$$

where $\mathbf{a}_l$ is the lth anchor's position. Then the loss function is defined as

$$\phi = \sum_{i=1}^{N} \{ \sum_{k \in \mathcal{P}_i} \|\mathbf{x}_k - Q_i \mathbf{x}_{k,i} - \mathbf{t}_i\|^2 + \lambda \sum_{l \in \mathcal{P}_i} \|Q_{N+1} \mathbf{a}_l - Q_i \mathbf{a}_l - \mathbf{t}_i\|^2 \},$$

where N is the number of patch, $\mathcal{P}_i$ is the collection of indices of points in patch i, Q_{N+1} is added to make ϕ homogeneous with respect to the variables. The registration process is to minimize ϕ with respect to x_k, Q_i and t_i. By simplification and substitution, the original problem is reduced to minimize

$$\mathrm{Tr}(CQ^TQ) = \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} \mathrm{Tr}(C_{ij} Q_i^T Q_j),$$

where the variables are the orthogonal matrices $Q_1, \ldots, Q_{N+1}$, and C_{ij} is constant matrices. Let $G = Q^TQ$, the above optimization problem is equivalent to

$$\begin{aligned} \min \quad & \mathrm{Tr}(CG) \\ \text{s.t.} \quad & G \succeq 0, \mathrm{rank}(G) = r, G_{ii} = I_r (i = 1, \ldots, N+1). \end{aligned} \tag{4.3}$$

By dropping the rank constraint, the resulting convex optimization model is given as

$$\begin{aligned} \min \quad & \mathrm{Tr}(CG) \\ \text{s.t.} \quad & G \succeq 0,\ G_{ii} = I_r\ (i = 1, \ldots, N+1). \end{aligned} \tag{4.4}$$

Problem (4.4) is a linear positive semidefinite programming that can be efficiently solved by state-of-the-art SDP solvers such as SDPT3 (Toh et al., 1999).

For localization of each patch, Chaudhury *et al.* suggest to use SDP-based localization schemes such as SNLSDP (Biswas et al., 2006) and ESDP (Wang et al., 2008) as they offer a good tradeoff between accuracy and run-time for small problems. The numerical experiment shows that the localization scheme proposed in Chaudhury et al. (2013) can localize a network with 8000 points in 30 minutes on a standard PC with a four-core 2.83 GHz Linux workstation with a 3.6 GB memory.

All the localization schemes in Savarese et al. (2001); Čapkun et al. (2002); Shang and Ruml (2004); Koren et al. (2005); Chaudhury et al. (2013) follow the idea of patching-stitching process, however, they all divide the whole network into subnetworks by taking each point and its neighbours as one patch, resulting in a large amount of patches, which may slow down the localization process and accumulate localization error. Other research groups provided different dividing and localization approaches, especially in the research area of gene network and molecular conformation.

Tzeng et al. (2008) proposed a split-and-combine MDS (SC-MDS) for large genomic data sets. They divided the set of points into K groups randomly with N_g points in each group and N_I overlapping points in each intersection region. For each patch, they used cMDS to get the relative position of points. To stitch patches together, they chose a patch as central reference and combined the patch around it one by one, until all patches are stitched to the main network. They claimed that for this method to work, the number of points in the intersection region should be greater than the real data dimension so that one can find the proper affine transformation. Their experimental result also showed that the points in each group should be chosen randomly, or the rotation effect will hamper the performance

of the low dimensional location recovery. This method is based on the full and well estimated information of pairwise distances between all points, otherwise the random picking would result in a patch with unconnected graph.

Biswas *et al.* developed a distributed SDP approach for large-scale noisy anchor-free graph realization for molecular conformation in Biswas et al. (2008). To break the limitation of interior point method for large-scale SDP problem, they used a symmetric reverse Cuthill-McKee permutation (George and Liu, 1981) to divide the whole molecular network into subnetworks. Then they used SDP relaxation with several techniques on rank reduction to localize each subnetwork. Finally they stitched the patches one by one to get the whole final molecular conformation.

4.2 EDM-SNL based localization scheme for large-scale network

Following patching-stitching scheme for large-scale network, we introduce our EDM-SNL localization method for networks with large amount of points. As discussed in the previous section, three key issues should be addressed for developing such method:

1. How to divide the whole network into patches?
2. How to localize points in each patch in the local coordinate system?
3. How to stitch patches together to get the global coordinates of all points?

In the following of this section, we will try to answer above three questions and propose a EDM-SNL based localization scheme for large-scale networks.

4.2.1 Grouping strategy to divide network into patches

Recall the model (3.8) for EDM-SNL in Section 3.2.2, the most important input for localization of a network is the distance matrix D. To divide the whole network into several patches, we need to decide how much distance information we will need for each patch's localization and the merging process. We want the distance information in each patch to be as much as possible, then we can manage to localize each patch efficiently and accurately. We also want the intersection of each patch is reasonably large, then the stitching process will provide more accurate global network. Due to the limitation of radio range of each sensor, only the distance information between points in a certain neighbourhood is available, resulting in a sparse symmetric distance matrix D. Assume D has the partition as

$$D = \begin{bmatrix} D_{aa} & D_{as} \\ D_{sa} & D_{ss} \end{bmatrix} \in \mathcal{S}^n, \tag{4.5}$$

where $D_{aa} \in \mathcal{S}^m$ contains the distance between anchors, $D_{as} \in \mathbb{R}^{m\times(n-m)}$ and $D_{sa} \in \mathbb{R}^{(n-m)\times m}$ contain the distance between anchors and sensors, $D_{ss} \in \mathcal{S}^{n-m}$ is the distance matrix containing distance information between sensors. Note that m can be 0 for the case that there is no anchor position available.

To decide the patch where a certain sensor should go to, we perform the symmetric inverse Cuthill-McKee permutation (Cuthill and McKee, 1969; Liu and Sherman, 1976) on the distance matrix of sensors D_{ss}. The Cuthill-McKee permutation was fist developed in 1969 to reduce the bandwidth of sparse symmetric matrices by E. Cuthill and J. McKee. It seeks a permutation that moves the non-zero element of a matrix to the diagonal as much as possible, resulting in a quasi-diagonal block matrix. An example is showed in Figure 4.1, which is the sparsity pattern of a distance matrix $D \in \mathcal{S}^{2000}$ after Cuthill-McKee permutation.

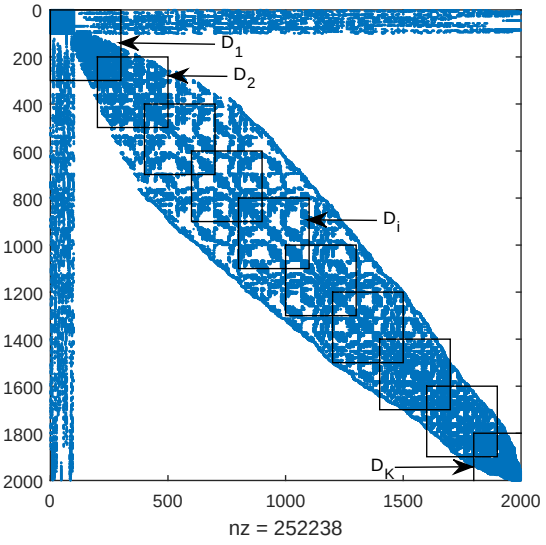


Figure 4.1: Sparsity pattern of distance matrix D after Cuthill-McKee permutation

The network used in Figure 4.1 contains $n = 2000$ points in total. The number of anchors $m = 100$. After the permutation, the distance information are gathered around the diagonal of the matrix as close as possible. Then it is straight forward for us to partition the points into different patches by breaking the whole matrix D into several small distance matrices $D_1, D_2, \ldots, D_K$. By now, we successively separate the whole network into K small patches. Every rectangular in Figure 4.1 represents a distance matrix for a certain patch. We will not waste the distance between anchors and sensors as well. During the localization process of each patch, the anchor-sensor distance (the blue band on the top of the graph in Figure 4.1) is added to the patch distance matrix D_i, resulting in a more accurate localization. In the stitching process, the points that in the intersection of two patches (the intersection of two black rectangulars in Figure 4.1) are used to stitch the two patches together.

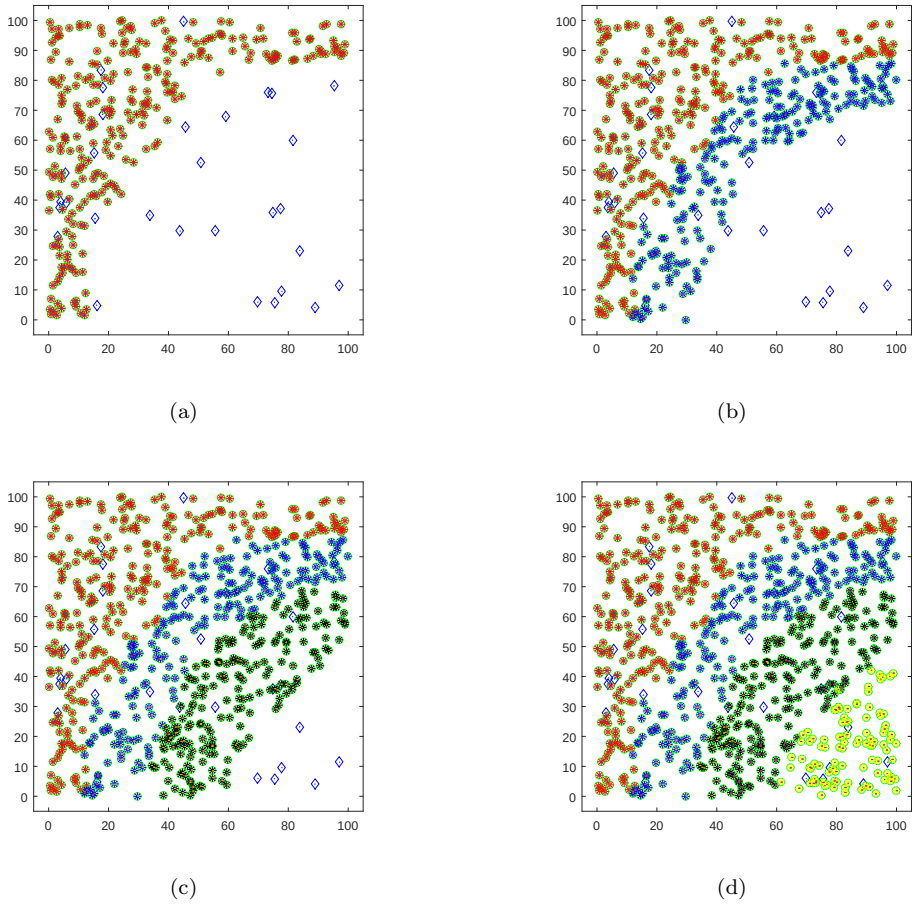
4.2.2 Localization and stitching methods for patches

Once we have the distance matrices $D_1, \ldots, D_K$ for K patches, EDM-SNL introduced in Chapter 3 is used for localizing each patch. We will use the example in Figure 4.1 to introduce the stitching process. At this stage of research, we choose to stitch each patch one by one form D_1 to D_K. Notice that the only information to stitch the points in D_1 to the global coordinate system is the anchor distance information (upper left blue rectangular in D_1). Then the points in D_1 can be successively stitched to the global coordinate system by Procrustes analysis. For the patch D_i, $i > 1$, we not only use the anchors, we also use the points in the intersection region of D_i and D_{i-1} as new anchors for stitching D_i, resulting in a much robust merging process to retrieve global positions of points. We summarize the above process in Algorithm LSEDM-SNL (Large-Scale EDM-based Sensor Network Localization).

Algorithm 9 LSEDM-SNL

1: Set np and mp as the number of points in each patch except for the last one and number of points in the intersection region between two patches correspondingly. Use symmetric inverse Cuthill-McKee permutation to get distance matrices $D_1, D_2, \ldots, D_K$.
2: Localize the points in patch D_1 using EDM-SNL, transform the points into global coordinate system by anchor positions through Procrustes analysis.
3: **for** $i = 2$ **to** K **do**
4: Localize the points in D_i using EDM-SNL, use the anchor position and the points in the intersection region of D_i and D_{i-1} to transform the points into global coordinate system through Procrustes analysis.
5: **end for**

The localization process by LSEDM-SNL is illustrated in Figure 4.2. The network contains $n = 800$ points in total, which are randomly distributed in the square region of $[0, 100]^2$. Among n points, there are $m = 30$ anchor nodes randomly distributed in the same region. The green circles represent the ground true locations of the generated sensor nodes and the blue diamonds represent the anchor nodes positions. The radio range is set to be $R = 15$. Following the process illustrated in Figure 4.1, setting the number of points in each patch $np = 300$ and the number



(a) (b)

(c) (d)

Figure 4.2: Localization process for network with $n = 800$ points in total, number of anchors $m = 30$, the radio range $R = 15$. Number of points in each patch $np = 300$, number of points in the intersection region of two patches $mp = 100$. Total CPU time $t = 27.43$s.

of points in the intersection region of two patches $mp = 100$, we get four distance matrices D_i, $i = 1, 2, 3, 4$. Localizing and stitching each patch one by one, we have the process showed in Figure 4.2, the points in four patches are coloured in red, blue, black and yellow respectively. The points in D_1 is localized first as illustrated in the Figure 4.2(a). After that, we localize and stitch the blue points in patch D_2 showed in Figure 4.2(b), so on and so forth, we have all four patches localized and stitched to the global coordinate system. We must point out that the randomly distributed anchors play an important role for patch localization. If there are only a few anchors available or no anchor at all, the idea that finds the global coordinate of each patch one by one would not work well. It is an interesting question and in

that case, one can find local coordinates instead of global ones of each patch and use the intersection of patches to stitch them together. Without anchors, the resulting coordinates could only be relative. Figure 4.3 depicts the relation between the number of anchors and localization quality. When the anchors get fewer, the localization error gets larger since the patch localization struggles resulting in the error propagation.

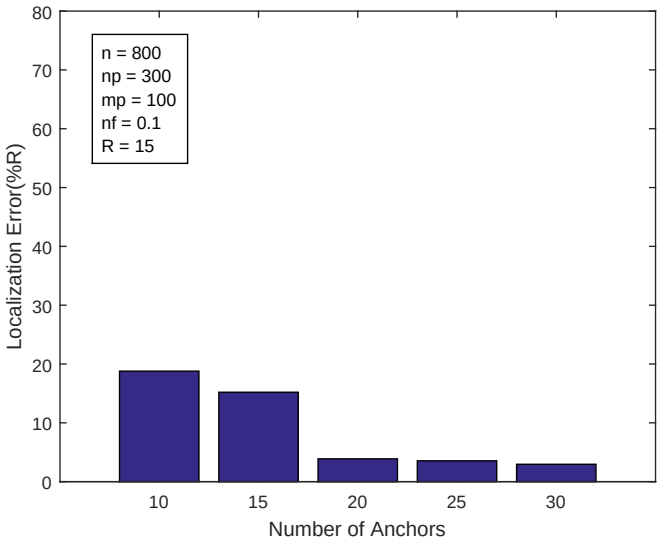


Figure 4.3: Variation of localization error with varying number of anchors. Networks of totally 800 nodes. Noise factor $nf = 0.1$. Communication radius $R = 15$.

Indeed, the stitching method has been in the literature for a long time and the quality of it highly depends on the localization quality of each patch and the intersection of two patches, i.e., the intersection of two squares in Figure 4.1.

4.2.3 Primary experimental results by LSEDM-SNL

In this section, we show the primary numerical experiment results using MATLAB (R2015a) on a desktop of Intel(R) Core(TM) i5-2500 3.3GHz CPU and 4GB memory to evaluate the efficiency of the proposed LSEDM-SNL algorithm.

The test networks are generated in a square region of $[0, 100]^2$, in which the (sensor and anchor) points are randomly distributed following the uniform distribution. The noise in the distance information is generated following a standard framework as follows:

$$\hat{d}_{ij} = d_{ij} \times |1 + nf \times randn|, \ \forall (i, j) \in \mathcal{E}, \tag{4.6}$$

where d_{ij} is the true Euclidean distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$, $0 \leqslant nf \leqslant 1$ is the noise factor, $randn$ is the standard normal random variable, and $\mathcal{E}$ is the index set containing the existing edges in the network. The accuracy measurement of the estimated positions follows directly from that of Chapter 3, which is the root mean square distance (RMSD)

$$\text{RMSD} := \frac{1}{\sqrt{n-m}} \left(\sum_{i=m+1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 \right)^{\frac{1}{2}}, \tag{4.7}$$

where $\hat{\mathbf{x}}_i$ is the estimated position and $\mathbf{x}_i$ is the ground-truth sensor position.

To test both accuracy and scalability of LSEDM-SNL, we generate networks with the total number of points n varying from 2000 to 10000. The radio range R varies from 15 to 10 to control the number of available edges. The noise factor is 0 and 0.1. Two importance parameters for the algorithm to work are the number of points in each patch np and the number of points in the intersection region of two patches mp. At this stage of research, we set $np = 300$ and $mp = 100$ fixed for all patches. We would like to point out that there may be wiser ways to choose different np and mp for each patch to make the algorithm more efficient. For each test example, 10 random inputs are generated and the average results are reported.

Figure 4.4 shows the change of localization quality while the noise factor increases from 0 to 0.9. The network used contains 2000 points in total. The number of anchor nodes is 100. The radio range is set to be 15. We can see that LSEDM-SNL

controls the localization error within $20\%R$ even when the noise factor is as large as 0.9.

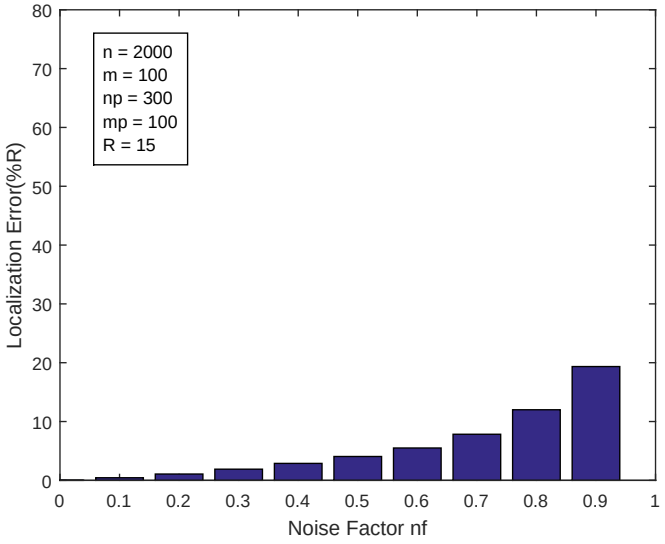


Figure 4.4: Variation of localization error with varying noise factor. Networks of totally 2000 nodes, among which are 100 anchor nodes distributed randomly. Communication radius $R = 15$.

The execution time and localization quality results by LSEDM-SNL are shown in Table 4.1. The number of sensors n varies from 2000 to 10000, and the number of anchors is fixed to 100. Compared to the results in Section 3.4.4, we know that patching-stitching localization scheme LSEDM-SNL is much faster than the direct localization EDM-SNL. LSEDM-SNL solves a problem of 2000 points in only 100 seconds while EDM-SNL uses 200 seconds to solve a problem of only 1000 points. Even for the problem of 10000 points, it only takes LSEDM-SNL less than 14 minutes to give an acceptable solution with RMSD ≈ 3.9.

However, we would like to point out that there are space for LSEDM-SNL to be improved. As shown in Figure 4.5, with the number of points in the network increasing, the localization error gets bigger.

One reason for that may be that we localize and stitch each patch one by one, using the position of the points in the last patch as anchor nodes for the next

Table 4.1: Execution Time and Quality Results by LSEDM-SNL

n	nf	R	RMSD	CPU time (secs)
2000	0	15	2.10E-05	97.79
	0.1	15	6.69E-02	97.20
4000	0	14	5.95E-04	200.50
	0.1	14	5.77E-02	199.73
6000	0	13	3.67E-02	316.86
	0.1	13	9.81E-01	320.83
8000	0	12	1.06E+00	471.79
	0.1	12	1.08E+00	454.33
10000	0	10	3.93E+00	771.08
	0.1	10	3.95E+00	767.16

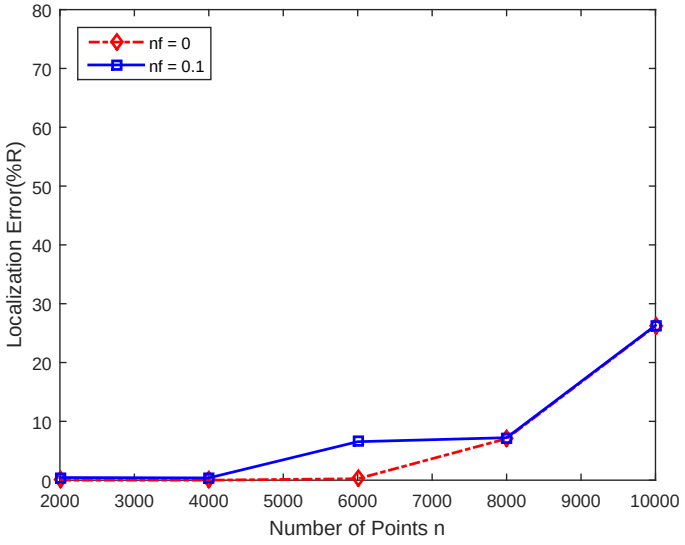
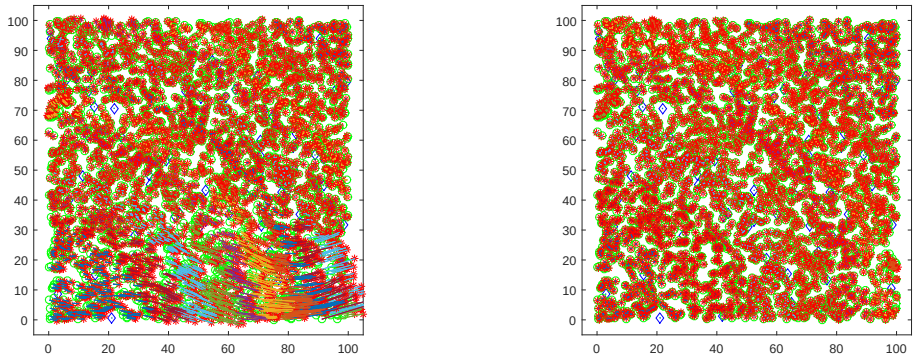


Figure 4.5: Variation of localization error with varying number of points in the network.

patch's localization, which may propagate the error through the whole process and built up the final localization error. An example is shown in Figure 4.6(a). The points of the lower part in the network are all shifted to the right since one patch in the lower right corner is miss localized.

Another possible reason is that we fix the number of each patch $np = 300$ and the number of points in the intersection region of two patches $mp = 100$ for all $n = 2000$ to 10000. For networks with points as many as 10000, it would drop so much useful distance information, resulting in a solution with bigger error. As



(a) (b)

Figure 4.6: Localization result of Network with $n = 3000$ and $m = 100$ anchors randomly distributed. Noise factor $nf = 0.1$. (a) $np = 300$, $mp = 100$, localization error RMSD $= 2.61\text{E}+00$, CPU time $t = 185.80$ seconds. (b) $np = 500$, $mp = 100$, localization error RMSD $= 2.48\text{E}-01$, CPU time $t = 239.17$ seconds. (Blue diamond: anchor position. Green circle: original sensor position. Red star: estimate sensor position. Blue line: error offset between original and estimate sensor position.)

illustrated in Figure 4.6(a), when $np = 300$, we got a localization with large error RMSD $= 2.61\text{E}+00$, and all the points in the lower part shifted to the right. When we change mp to 500, we got a very well localized network showed in Figure 4.6(b), but more time was taken. It can be explained clearly using Figure 4.1. If the black rectangulars are too small, and the bandwidth (the horizon length of the blue band on the diagonal) is very large, the rectangulars can not cover enough space of the blue band, resulting in the lose of useful distance information. To fix the above problems, several interesting and important issues, which are discussed in the next section, should be investigated.

4.3 Discussion and future work

In this Chapter, following the patching-stitching idea, based on the EDM-SNL, we introduced a large-scale network localization scheme called LSEDM-SNL, which

opens a gate to large-scale network localization using convex optimization approach. Numerical results showed that LSEDM-SNL is able to handle large data even with 10000 points in less than 14 minutes, while the solution still enjoys an acceptable accuracy. In the following part of this section, we would like to point out some interesting and important problems that are worth further discussion and research.

The number of points in each patch np and the number of points in the intersection region of two patches mp are the key parameters that influence the efficiency and accuracy of the LSEDM-SNL. We can not choose np to be too large since the eigenvalue decomposition for large matrix is not affordable and it may slow down the EDM-SNL for each patch's localization. The parameter mp can not be too small as it would degrade the quality of the stitching process. By now, we fix np and mp all the same for each patch. To make the most of EDM-SNL, np and mp should be chosen wisely to generate each patch individually with different size and intersection region. One possible way to do that is to use the bandwidth showed in Figure 4.1 and set the lower bound of distance information percentage in each patch.

Another possible upgrade of LSEDM-SNL is from the distributed localization scheme. Recall that LSEDM-SNL localizes and stitches patches one by one, the stitching process of each patch is depend on the localization quality of the previous patch, especially the localization quality of points in the intersection region since they are used as anchor nodes in the localization of next patch. This kind of way would propagate error through the whole process and build up the localization error. Moreover, it requires there exist enough anchor nodes in the first patch to stitch it to the global coordinate system. A more robust way is to localize each patch independently without using the points in the intersection region as anchor nodes. The role of points in the intersection region of two patches is only for gluing the two patches together, and the anchor nodes are used only to register

the whole network into the global coordinate system. However, this kind of way may degrade the localization quality in each patches, which may also influence the stitching and registration process.

One more interesting strategy is about parallel computing, in which many calculations are carried out simultaneously. It is possible to conduct parallel computing due to the development of computer architecture, mainly in the form of multi-core processor. If each patch is localized independently without exchanging data with other patches, as discussed in the above paragraph, then parallel computing would make the algorithm several times faster, depending on the number of cores in processor of the PC hardware. We believe that parallel computing is the new trend in the research area on big data analysis.

Chapter 5

Conclusions

In this thesis, we designed algorithms for solving two kinds of models in EDM-based optimization problem, both with rank constraint. The first model consists of spherical constraint, the second model has a large amount of equality and inequality constraints which can be integrated into bound constraints on the EDM.

For the EDM-based optimization problem with spherical constraint, we applied the framework of majorized penalty method to the resulting matrix problem in order to deal with the rank constraint iteratively. A key feature we exploited is that all the subproblems can be efficiently solved by the semismooth Newton method. Even though the resulting equation system is nonsmooth, we proved that the Newton method is quadratically convergent under the constraint nondegeneracy condition. Constraint nondegeneracy is a difficult constraint qualification to analyse. We proved it under a weak condition for our spherical constrained problem. A complete set of MATLAB code called **FITS** is written for test. We used 4 classic examples from the spherical multidimensional scaling to demonstrate the flexibility of the algorithm in incorporating various constraints.

For the EDM optimization with bound constraints, a heuristic linear function is added to induce low rank solution, resulting in a sensor network localization scheme

that mitigates flip ambiguity and obtains robustness under presence of large noises in the distance measurements. A novel alternating direction method of multipliers is applied to solve the resulting 3-block convex quadratic optimization model. We implemented the integrated localization scheme in the MATLAB code **EDM-SNL** for public test. Numerical experiments showed the EDM-SNL outperforms the existing state-of-the-art localization schemes.

Both Newton-type algorithm and ADMM algorithm have advantages against each other, but they also have their own drawbacks. Newton-type algorithm usually converges fast and could provide accurate solution efficiently, however the convergence may be influenced by the presence of different types of constraint. ADMM algorithm could deal with different types of constraints, however, it requires more iteration to get a high accuracy solution. For EDM-based optimization, the major computation is the spectral decomposition of a symmetric matrix, which could be fatal to the algorithm efficiency when the scale of matrix goes beyond thousands. To tackle this problem, we introduced an EDM-SNL based patching-stitching localization approach called LSEDM-SNL for large-scale network localization. Breaking the whole network into small patches allows us to use limited computation resource and less time to deal with localization problems of large-scale. Numerical results showed that the proposed LSEDM-SNL algorithm is promising.

References

I. F. Akyildiz and M. C. Vuran. *Wireless sensor networks*, volume 4. John Wiley & Sons, 2010.

S. Al-Homidan and H. Wolkowicz. Approximate and exact completion problems for Euclidean distance matrices using semidefinite programming. *Linear algebra and its applications*, 406:109–141, 2005.

A. Alfakih, A. Khandani, and H. Wolkowicz. Solving Euclidean distance matrix completion problems via semidefinite programming. *Computational Optimization and Applications*, 12(1-3):13–30, 1999.

A. Y. Alfakih and H. Wolkowicz. *On the embeddability of weighted graphs in Euclidean spaces*. Citeseer, 1998.

A. Y. Alfakih, M. F. Anjos, V. Piccialli, and H. Wolkowicz. Euclidean distance matrices, semidefinite programming and sensor network localization. *Portugaliae Mathematica*, 68(1):53, 2011.

F. Alizadeh, J.-P. Haeberly, and M. Overton. Complementarity and nondegeneracy in semidefinite programming. *Mathematical Programming*, 77(1):111–128, 1997.

S. Bai and H.-D. Qi. Tackling the flip ambiguity in wireless sensor network localization and beyond. *Digital Signal Processing*, 55:85–97, 2016.

S. Bai, H.-D. Qi, and N. Xiu. Constrained best euclidean distance embedding on a sphere: A matrix optimization approach. *SIAM Journal on Optimization*, 25 (1):439–467, 2015.

A. Beck and D. Pan. On the solution of the GPS localization and circle fitting problems. *SIAM Journal on Optimization*, 22(1):108–134, 2012.

D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, pages 46–54, New York, NY, USA, 2004. ACM.

P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *Automation Science and Engineering, IEEE Transactions on*, 3(4): 360–371, Oct 2006.

P. Biswas, K.-C. Toh, and Y. Ye. A distributed SDP approach for large-scale noisy anchor-free graph realization with applications to molecular conformation. *SIAM Journal on Scientific Computing*, 30(3):1251–1277, 2008.

J. F. Bonnans and A. Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

I. Borg and P. J. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

I. Borg and J. Lingoes. A model and algorithm for multidimensional scaling with external constraints on the distances. *Psychometrika*, 45(1):25–38, 1980.

J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer., 2000.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772, 2009.

S. Čapkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad hoc networks. *Cluster Computing*, 5(2):157–167, 2002.

Z. X. Chan and D. Sun. Constraint nondegeneracy, strong regularity, and nonsingularity in semidefinite programming. *SIAM Journal on Optimization*, 19(1): 370–396, 2008.

K. N. Chaudhury, Y. Khoo, and A. Singer. Large-scale sensor network localization via rigid subnetwork registration. *arXiv preprint arXiv:1310.8135*, 2013.

K. N. Chaudhury, Y. Khoo, and A. Singer. Global registration of multiple point clouds using semidefinite programming. *SIAM Journal on Optimization*, 25(1): 468–501, 2015.

C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, pages 1–23, 2014.

F. H. Clarke. *Optimization and nonsmooth analysis*, volume 5. Siam, 1990.

J. A. Costa, N. Patwari, and A. O. Hero, III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sen. Netw.*, 2(1):39–64, Feb. 2006.

T. F. Cox and M. A. Cox. Multidimensional scaling on a sphere. *Communications in Statistics - Theory and Methods*, 20(9):2943–2953, 1991.

T. F. Cox and M. A. Cox. *Multidimensional scaling*. CRC Press, 2000.

G. M. Crippen, T. F. Havel, et al. *Distance geometry and molecular conformation*, volume 74. Research Studies Press Somerset, England, 1988.

E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172, New York, NY, USA, 1969. ACM.

J. De Leeuw and P. Mair. Multidimensional scaling using majorization: SMACOF in R. *Journal of Statistical Software*, 31(3):1–30, 2009.

Y. Ding, N. Krislock, J. Qian, and H. Wolkowicz. Sensor network localization, euclidean distance matrix completions, and graph realization. *Optimization and Engineering*, 11(1):45–66, 2010.

P. Drineas, A. Javed, M. Magdon-Ismail, G. Pandurangan, R. Virrankoski, and A. Savvides. Distance matrix reconstruction from incomplete distance information for sensor network localization. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, volume 2, pages 536–544, Sept 2006.

R. L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

G. Ekman. Dimensions of color vision. *The Journal of Psychology*, 38(2):467–474, 1954.

M. Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

M. Fazel, T. K. Pong, D. Sun, and P. Tseng. Hankel matrix rank minimization with applications to system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34(3):946–977, 2013.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17 – 40, 1976.

N. Gaffke and R. Mathar. A cyclic projection algorithm via duality. *Metrika*, 36 (1):29–54, 1989.

Y. Gao. *Structured Low Rank Matrix Optimization Problems: a Penalty Approach*. PhD thesis, National University of Singapore, 2010.

Y. Gao and D. Sun. A majorized penalty approach for calibrating rank constrained correlation matrix problems. Technical report, Department of Mathematics, National University of Singapore, March 2010.

A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.

S. Gepshtein and Y. Keller. Sensor network localization by augmented dual embedding. *Signal Processing, IEEE Transactions on*, 63(9):2420–2431, May 2015.

R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(2):41–76, 1975.

W. Glunt, T.-L. Hayden, S. Hong, and J. Wells. An alternating projection algorithm for computing the nearest euclidean distance matrice. *SIAM J. Matrix Anal. Appl.*, 11:589–600, 1990.

J. C. Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53(3-4):325–338, 1966.

J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

J. C. Gower. Euclidean distance geometry. *Mathematical Scientist*, 7(1):1–14, 1982.

J. C. Gower. Properties of Euclidean and non-Euclidean distance matrices. *Linear Algebra and its Applications*, 67:81 – 97, 1985.

S.-P. Han. A successive projection method. *Mathematical Programming*, 40(1-3): 1–14, 1988.

J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975.

T. F. Havel and K. Wüthrich. An evaluation of the combined use of nuclear magnetic resonance and distance geometry for the determination of protein conformations in solution. *Journal of molecular biology*, 182(2):281–294, 1985.

T. Hayden and J. Wells. Approximation by matrices positive semidefinite on a subspace. *Linear Algebra and its Applications*, 109:115 – 130, 1988.

M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

J.-B. Hiriart-Urruty and C. Lemaréchal. Convex analysis and minimization algorithms ii: Advanced theory and bundle methods, vol. 306 of grundlehren der mathematischen wissenschaften, 1993.

J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms I: fundamentals*, volume 305. Springer Science & Business Media, 2013.

J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. Nguyen. Generalized Hessian matrix and second-order optimality conditions for problems with C 1,1 data. *Applied Mathematics and Optimization*, 11(1):43–56, 1984.

B. Jiang and Y.-H. Dai. A framework of constraint preserving update schemes for optimization on Stiefel manifold. *Mathematical Programming*, pages 1–41, 2014.

A. Kannan, G. Mao, and B. Vucetic. Simulated annealing based wireless sensor network localization with flip ambiguity mitigation. In *Vehicular Technology*

Conference, 2006. VTC 2006-Spring. IEEE 63rd, volume 2, pages 1022–1026, May 2006.

A. Kannan, B. Fidan, and G. Mao. Analysis of flip ambiguities for robust sensor network localization. *Vehicular Technology, IEEE Transactions on*, 59(4):2057–2070, May 2010.

A. Karbasi and S. Oh. Robust localization from incomplete local information. *Networking, IEEE/ACM Transactions on*, 21(4):1131–1144, Aug 2013.

S. Kim, M. Kojima, H. Waki, and M. Yamashita. Algorithm 920: SFSDP: A sparse version of full semidefinite programming relaxation for sensor network localization problems. *ACM Trans. Math. Softw.*, 38(4):27:1–27:19, Aug. 2012.

M. Kočvara and M. Stingl. PENNON: A code for convex nonlinear and semidefinite programming. *Optimization methods and software*, 18(3):317–333, 2003.

Y. Koren, C. Gotsman, and M. Ben-Chen. PATCHWORK: Efficient localization for sensor networks by distributed global optimization. Technical report, Citeseer, 2005.

N. Krislock and H. Wolkowicz. Explicit sensor network localization using semidefinite representations and facial reductions. *SIAM Journal on Optimization*, 20 (5):2679–2708, 2010.

N. Krislock and H. Wolkowicz. *Euclidean distance matrices and applications*. Springer, 2012.

M. Laurent and A. Varvitsiotis. Positive semidefinite matrix completion, universal rigidity and the strong Arnold property. *Linear Algebra and its Applications*, 452:292 – 317, 2014.

S.-Y. Lee and P. M. Bentler. Functional relations in multidimensional scaling. *British Journal of Mathematical and Statistical Psychology*, 33(2):142–150, 1980.

Q. Li and H.-D. Qi. A sequential semismooth Newton method for the nearest low-rank correlation matrix problem. *SIAM Journal on Optimization*, 21(4): 1641–1666, 2011.

X. Li, D. Sun, and K.-C. Toh. A Schur complement based semi-proximal admm for convex quadratic conic programming and extensions. *Mathematical Programming*, pages 1–41, 2014.

L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, 2014.

C. Ling, J. Nie, L. Qi, and Y. Ye. Biquadratic optimization over unit spheres and semidefinite programming relaxations. *SIAM Journal on Optimization*, 20(3): 1286–1310, 2010.

W.-H. Liu and A. H. Sherman. Comparative analysis of the CuthillMcKee and the reverse CuthillMcKee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

J. Malick. The spherical constraint in boolean quadratic programs. *Journal of Global Optimization*, 39(4):609–622, 2007.

W. Mian. *Matrix completion procedure with fixed basis coefficients and rank regularized problems with hard constraints*. PhD thesis, National University of Singapore, 2013.

W. Miao, S. Pan, and D. Sun. A rank-corrected procedure for matrix completion with fixed basis coefficients. *arXiv preprint arXiv:1210.3709*, 2012.

R. Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

J.-J. Moreau. Décomposition orthogonale dun espace hilbertien selon deux cônes mutuellement polaires. *CR Acad. Sci. Paris*, 255:238–240, 1962.

J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

J.-S. Pang, D. Sun, and J. Sun. Semismooth homeomorphisms and strong stability of semidefinite and Lorentz complementarity problems. *Mathematics of Operations Research*, 28(1):39–63, 2003.

N. Patwari, J. Ash, S. Kyperountas, A. Hero, R. Moses, and N. Correal. Locating the nodes: cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE*, 22(4):54–69, July 2005.

E. Pękalska and R. P. Duin. *The dissimilarity representation for pattern recognition: foundations and applications*. Number 64. World Scientific, 2005.

M. J. Powell. *A method for non-linear constraints in minimization problems*. UKAEA, 1967.

H.-D. Qi. A semismooth Newton method for the nearest euclidean distance matrix problem. *SIAM Journal on Matrix Analysis and Applications*, 34(1):67–93, 2013.

H.-D. Qi and D. Sun. A quadratically convergent Newton method for computing the nearest correlation matrix. *SIAM Journal on Matrix Analysis and Applications*, 28(2):360–385, 2006.

H.-D. Qi and X. Yuan. Computing the nearest Euclidean distance matrix with low embedding dimensions. *Mathematical Programming*, 147(1-2):351–389, 2014.

L. Qi and J. Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 58(1-3):353–367, 1993.

S. Robinson. Local structure of feasible sets in nonlinear programming, part iii: Stability and sensitivity. In B. Cornet, V. Nguyen, and J. Vial, editors, *Nonlinear Analysis and Optimization*, volume 30 of *Mathematical Programming Studies*, pages 45–66. Springer Berlin Heidelberg, 1987.

S. M. Robinson. Constraint nondegeneracy in variational analysis. *Mathematics of Operations Research*, 28(2):201–232, 2003.

R. T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2): 97–116, 1976.

R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

C. Savarese, J. Rabaey, and J. Beutel. Location in distributed ad-hoc wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 4, pages 2037–2040 vol.4, 2001.

I. J. Schoenberg. Remarks to Maurice Frechet's article "sur la definition axiomatique d'une classe d'espace distances vectoriellement applicable sur l'espace de hilbert. *Annals of Mathematics*, 36(3):pp. 724–732, 1935.

I. J. Schoenberg. On certain metric spaces arising from Euclidean spaces by a change of metric and their embedding in Hilbert space. *Annals of Mathematics*, 38(4):pp. 787–793, 1937.

Y. Shang and W. Ruml. Improved MDS-based localization. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2640–2651 vol.4, March 2004.

Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM International Symposium on Mobile*

Ad Hoc Networking &Amp; Computing, MobiHoc '03, pages 201–212, New York, NY, USA, 2003. ACM.

A. Shapiro. Sensitivity analysis of generalized equations. *Journal of Mathematical Sciences*, 115(4):2554–2565, 2003.

S. Shekofteh, M. Khalkhali, M. Yaghmaee, and H. Deldari. Localization in wireless sensor networks using tabu search and simulated annealing. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 2, pages 752–757, Feb 2010.

D. Sun. The strong second-order sufficient condition and constraint nondegeneracy in nonlinear semidefinite programming and their implications. *Mathematics of Operations Research*, 31(4):761–776, 2006.

D. Sun and J. Sun. Semismooth matrix-valued functions. *Mathematics of Operations Research*, 27(1):150–169, 2002.

D. Sun, K.-C. Toh, and L. Yang. A convergent 3-block semi-proximal alternating direction method of multipliers for conic programming with 4-type of constraints. *arXiv preprint arXiv:1404.5378*, 2014.

K.-C. Toh. An inexact primaldual path following algorithm for convex quadratic SDP. *Mathematical Programming*, 112(1):221–254, 2008.

K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3-a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.

P. Tseng. Second-order cone programming relaxation of sensor network localization. *SIAM Journal on Optimization*, 18(1):156–185, 2007.

J. Tzeng, H. H. Lu, and W.-H. Li. Multidimensional scaling for large genomic data sets. *BMC bioinformatics*, 9(1):179, 2008.

Z. Wang, S. Zheng, Y. Ye, and S. Boyd. Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM Journal on Optimization*, 19(2):655–673, 2008.

K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 106–, New York, NY, USA, 2004. ACM.

K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in neural information processing systems*, pages 1489–1496, 2006.

Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

Z. Wen, D. Goldfarb, and W. Yin. Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, 2010.

K. Yosida. Functional analysis. reprint of the sixth (1980) edition. classics in mathematics. *Springer-Verlag, Berlin*, 11:14, 1995.

G. Young and A. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22, 1938.

L. Zhang, L. Liu, C. Gotsman, and S. J. Gortler. An as-rigid-as-possible approach to sensor network localization. *ACM Trans. Sen. Netw.*, 6(4):35:1–35:21, July 2010.

X.-Y. Zhao, D. Sun, and K.-C. Toh. A Newton-CG augmented lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4):1737–1765, 2010.

G. Zhou, L. Caccetta, K. L. Teo, and S.-Y. Wu. Nonnegative polynomial optimization over unit spheres and convex programming relaxations. *SIAM Journal on Optimization*, 22(3):987–1008, 2012.