UNIVERSITY OF SOUTHAMPTON

FACULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES

Operational Research

Majorization-Projection Methods for Multidimensional Scaling via Euclidean Distance Matrix Optimization

by

Shenglong Zhou

Thesis submitted for the degree of Doctor of Philosophy

December 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES

Operational Research

Doctor of Philosophy

MAJORIZATION-PROJECTION METHODS FOR MULTIDIMENSIONAL SCALING VIA EUCLIDEAN DISTANCE MATRIX OPTIMIZATION

by Shenglong Zhou

This thesis aims to propose an efficient numerical method for a historically popular problem, multi-dimensional scaling (MDS), through the Euclidean distance matrix (EDM) optimization. The problem tries to locate a number of points in a low dimensional real space based on some inter-vector dissimilarities (i.e., noise contaminated Euclidean distances), which has been notoriously known to be non-smooth and non-convex.

When it comes to solving the problem, four classes of stress based minimizations have been investigated. They are stress minimization, squared stress minimization, robust MDS and robust Euclidean embedding, yielding numerous methods that can be summarized into three representative groups: coordinates descent minimization, semi-definite programming (SDP) relaxation and EDM optimization. Each of these methods was cast based on only one or two minimizations and difficult to process the rest. Especially, no efficient methods have been proposed to address the robust Euclidean embedding to the best of our knowledge.

In this thesis, we manage to formulate the problem into a general EDM optimization model with ability to possess four objective functions that respectively correspond to above mentioned four minimizations. Instead of concentrating on the primary model, we take its penalization into consideration but also reveal their relation later on. The appealing feature of the penalization allows its four objective functions to be economically majorized by convex functions provided that the penalty parameter is above certain threshold. Then the projection of the unique solution of the convex majorization onto a box set enjoys a closed form, leading to an extraordinarily efficient algorithm dubbed as `MPEDM`, an abbreviation for Majorization-Projection via EDM optimization. We prove that `MPEDM` involving four objective functions converges to a stationary point of the penalization and also an ϵ-KKT point of the primary problem. Therefore, we succeed in achieving a viable method that is able to solve all four stress based minimizations.

Finally, we conduct extensive numerical experiments to see the performance of `MPEDM` by carrying out self-comparison under four objective functions. What is more, when it is against with several state-of-the-art methods on a large number of test problems including wireless sensor network localization and molecular conformation, the superiorly fast computational speed and very desirable accuracy highlight that it will become a very competitive embedding method in high dimensional data setting.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Shenglong Zhou , declare that the thesis entitled *Majorization-Projection Methods for Multidimensional Scaling via 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;
- none of this work has been published before submission

Signed:..

Date:.....27/11/2018...

Acknowledgements

My deepest gratitude goes first and foremost to my supervisor, Professor Hou-Duo Qi, for his meticulous guidance and constant encouragement throughout all stages of my postgraduate study. His excellent mathematical knowledge and illuminating instructions contributed enormously to the accomplishment of this thesis.

I would also like to express my heartfelt gratitude to Professor Naihua Xiu from Beijing Jiaotong University for his generous support and invaluable advice. Without his help, it would not be such of comfort and convenience in my postgraduate life. The research group led by him offered me lots of help and care. Especially, I am greatly indebted to Professor Lingchen Kong and Professor Ziyan Luo who provided me thoughtful arrangements and shared with me various interesting research topics.

My thanks would also go to my examiners, Dr. Parpas Panos from Imperial College London and Dr. Stefano Coniglio, for their careful reading and valuable comments.

Finally, I would like to express my heartfelt gratitude to my beloved parents and brothers for their endless love and support all through my life. I also place my sense of gratitude to my friends and my fellow colleagues for their help and company over these years.

Nomenclature

$\mathbb{R}^n$	The n dimensional Euclidean real space. Particularly, $\mathbb{R} := \mathbb{R}^1$.
$\mathbb{R}^n_+$	The n dimensional Euclidean real space with all non-negative vectors.
$\mathbb{R}^{m\times n}$	The linear space of real $m \times n$ matrices.
$\mathbf{x}$	A vector with the i-th element x_i, similar to $\mathbf{y}, \mathbf{z}$ etc.
$\mathbf{e}$	A vector with all elements being 1.
X	A matrix with the j-th column $\mathbf{x}_j$ and the ij-th element X_{ij}.
I_n	The $n \times n$ order identity matrix I_n. Simply write as I if no ambiguity of its order in the context.
$\mathrm{tr}(A)$	The trace of A, i.e., $\mathrm{tr}(A) = \sum_i a_{ii}$
$\langle A, B\rangle$	The Frobenius inner product of $A, B \in \mathbb{R}^{m\times n}$, i.e., $\langle A, B\rangle = \mathrm{tr}(AB^\top)$.
$\mathbb{S}^n$	The space of $n \times n$ symmetric matrices, equipped with the inner product.
$\mathbb{S}^n_h$	The hollow space in $\mathbb{S}^n$, i.e., $\{A \in \mathbb{S}^n :\ A_{ii} = 0\}$.
$\mathbb{S}^n_+$	The cone of positive semi-definite matrices in $\mathbb{S}^n$, i.e. $\{A \in \mathbb{S}^n :\ A \succeq \mathbf{0}\}$.
$\mathbb{S}^n_+(r)$	Low rank matrices in $\mathbb{S}^n_+$, i.e., $\{A \in \mathbb{S}^n_+ :\ \mathrm{rank}(A) \le r\}$.
$\mathbb{A}$	A linear mapping, similar to $\mathbb{B}, \mathbb{P}, \mathbb{Q}$ etc.
$\mathbb{A}^*$	The adjoint linear mapping of $\mathbb{A}$, i.e., $\langle \mathbb{A}\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, \mathbb{A}^*\mathbf{y}\rangle$. Particularly, $A^\top$ is the transpose of matrix A. A self-adjoint linear mapping means $\mathbb{A} = \mathbb{A}^*$.
$\|\cdot\|$	The induced Frobenius norm for matrices and Euclidean norm for vectors.
$\lambda_i(A)$	The i-th largest eigenvalue of A.
J	The centring matrix with order n, i.e., $I_n - \mathbf{e}\mathbf{e}^\top/n$.
$\Pi^B_\Omega(X)$	The set of all projections of X onto a closed set Ω, i.e., $\mathrm{argmin}_{Y\in\Omega}\ \|Y - X\|$.
$\Pi_\Omega(X)$	The orthogonal projection of X onto a closed set Ω, i.e., $\Pi_\Omega(X) \in \Pi^B_\Omega(X)$. When Ω is convex, $\Pi_\Omega(X)$ is unique.
$X \circ Y$	The Hadamard product between X and Y, i.e., $(X \circ Y)_{ij} = X_{ij}Y_{ij}$.
$X^{(p)}$	$(X^{(p)})_{ij} = X^p_{ij}$ where $p > 0$, such as $(X^{(2)})_{ij} = X^2_{ij}$ and $(X^{(1/2)})_{ij} = \sqrt{X_{ij}}$.

Chapter 1

Introduction

Throughout this thesis, for the sake of clearness, definitions of some basic notation can be referred in Nomenclature on Page xvii if there is no extra explanations.

In this chapter, we first introduce the problem of interest of this thesis, Multidimensional scaling (MDS), which covers extensive applications in various research communities including Psychology, Statistics and Computer Science. Motivations of this topic are then presented through several specific applications, such as wireless sensor network localization, molecular conformation, fitting points on a sphere and dimensionality reduction.

1.1 Multidimensional Scaling (MDS)

Multidimensional scaling (MDS) as a data analysis technique aims at searching an embedding in a new (possibly low dimensional) vector space from some points/objects with hidden structures. Here for a given set of objects in $\mathbb{R}^d$, embedding them or searching an embedding in a new space $\mathbb{R}^r$ $(r \leq d)$ means finding their new coordinates in such space $\mathbb{R}^r$. Some inter-point distances of this embedding (i.e., new coordinates) are expected to approach a small portion of given pairwise dissimilarities as closely as possible. It is well known that MDS was originated from the field in psychology (Torgerson, 1952; Shepard, 1962; Kruskal, 1964) and covers extensively applications in various communities including both social and engineering sciences (e.g., visualization (Buja et al., 2008) and dimensionality reduction Tenenbaum et al. (2000)), which were well documented in the books (Cox and Cox, 2000) and (Borg and Groenen, 2005). Recently, it has been successfully applied into molecular conformation (Glunt et al., 1993; Zhou et al., 2018a) and wireless sensor network localization (SNL) in high dimensional data settings (see Biswas and Ye, 2004; Shang and Ruml, 2004; Biswas et al., 2006; Zhen, 2007; Costa et al., 2006; Karbasi and Oh, 2013; Bai and Qi, 2016). The problem can be briefly described as follows.

Suppose there are n points/objects $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ in $\mathbb{R}^r$, and (Euclidean) dissimilarities among some of the points can be observed:

$$\delta_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| + \epsilon_{ij}, \quad \textit{for some pairs } (\boldsymbol{x}_i\ \boldsymbol{x}_j), \tag{1.1}$$

where $\|\cdot\|$ is Euclidean norm, the ϵ_{ij} are noises/outliers and δ_{ij} are observed dissimilarities. The main task is to recover the n points $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ in $\mathbb{R}^r$ purely based on those available dissimilarities.

It is wroth mentioning that if the dissimilarity of one pair is not available, it is generally taken as 0. Therefore, a dissimilarity matrix $\Delta \in \mathbb{S}^n_h$ can be acquired by, for $i < j$,

$$\Delta_{ij} = \begin{cases} \delta_{ij}, & \text{for some pairs } (\mathbf{x}_i\ \mathbf{x}_j), \\ 0, & \text{otherwise.} \end{cases} \tag{1.2}$$

1.2 Motivations

The motivations for us to consider MDS are various important applications ranging from constrained multidimensional scaling in Psychology, spatial data representation in Statistics, machine learning and pattern recognition in Computer Science. Since the wide range is beyond our scope, we only introduce four specific examples: sensor network localization (SNL), molecular conformation (MC), embedding on a sphere (ES) and dimensionality reduction (DR).

1.2.1 Sensor Network Localization (SNL)

Wireless sensor network localization (SNL) plays an important role in real world such as health surveillance, battle field surveillance, environmental/earth or industrial monitoring, coverage, routing, location service, target tracking, rescue and so forth, 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.

The problem is to locate a number of points (known as sensors) based on some observed dissimilarities. See Figure 1.1 for example, there are five red squares, the given points (known as anchors), and seventy five sensors (blue circles) in $\mathbb{R}^2$. The difference between a sensor and a anchor is that the latter has fixed/known position. Actually, a sensor can be an anchor if its position is given before to locate other unknown sensors. The pink and green lines link two neighbour points, which indicates the dissimilarities between them can be observed in advance. The task is to find locations of all sensors in $\mathbb{R}^2$ based on those known dissimilarities. More details can be referred to Section 5.1.

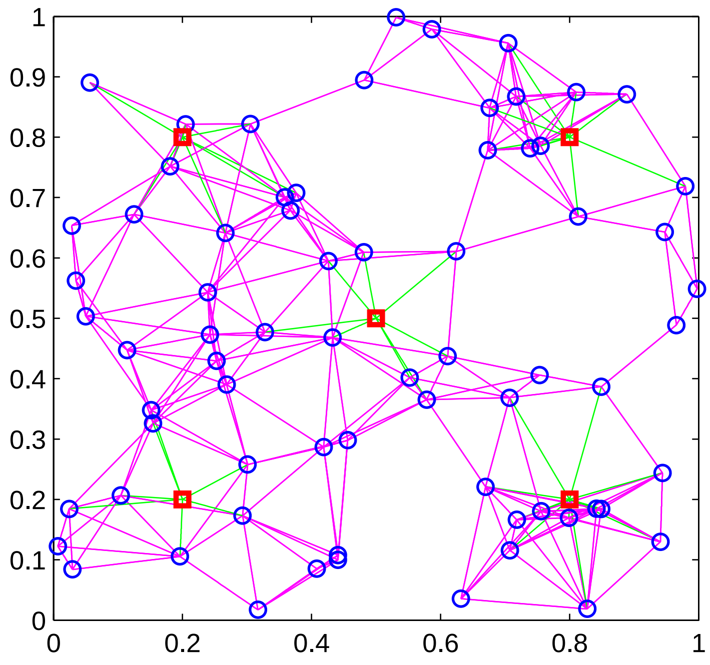


Figure 1.1: Sensor network localization of eighty nodes.

1.2.2 Molecular Conformation (MC)

Molecular conformation (MC) problem can be briefly described as follows. For a molecule with some atoms, the problem tries to determine the positions of these atoms in $\mathbb{R}^3$, given estimated some inter-atomic dissimilarities which could be derived from covalent bond lengths or measured by nuclear magnetic resonance (NMR) experiments.

As demonstrated by Figure 1.2, two molecules '1LFB' (containing 641 atoms) and '5BNA' (having 486 atoms) from Protein Data Bank were plotted. Since the maximal distance between two atoms that the NMR experiment can measure is nearly $6\mathring{A}(1\mathring{A} = 10^{-8}$cm), for each molecule, if the distance between two atoms is less than this threshold, then their dissimilarities is able to be gotten; otherwise no information about the pair is known. Therefore, the primary information is a set of dissimilarities and the task is to locate each atom in $\mathbb{R}^3$. Please refer to Section 5.2 for more details.

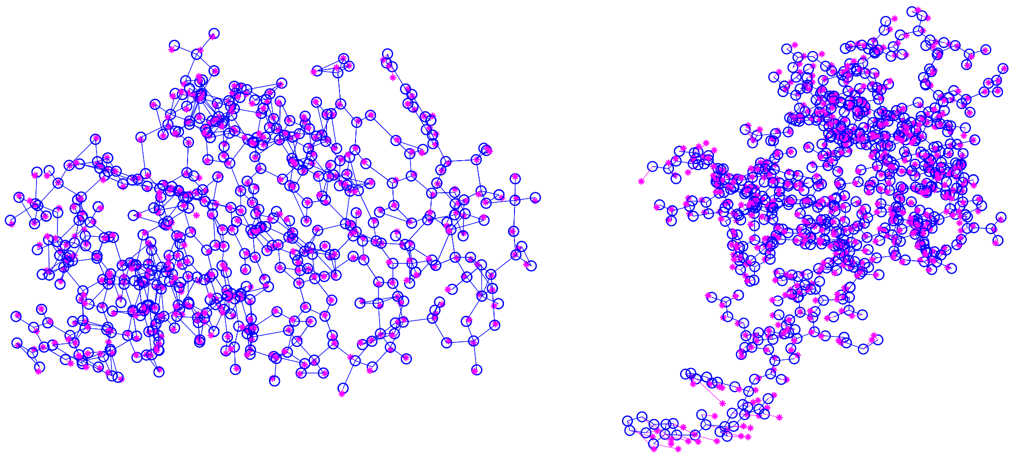

(a) 1AU6:$n = 506$ (b) 1LFB:$n = 641$

Figure 1.2: Molecular conformation of protein data.

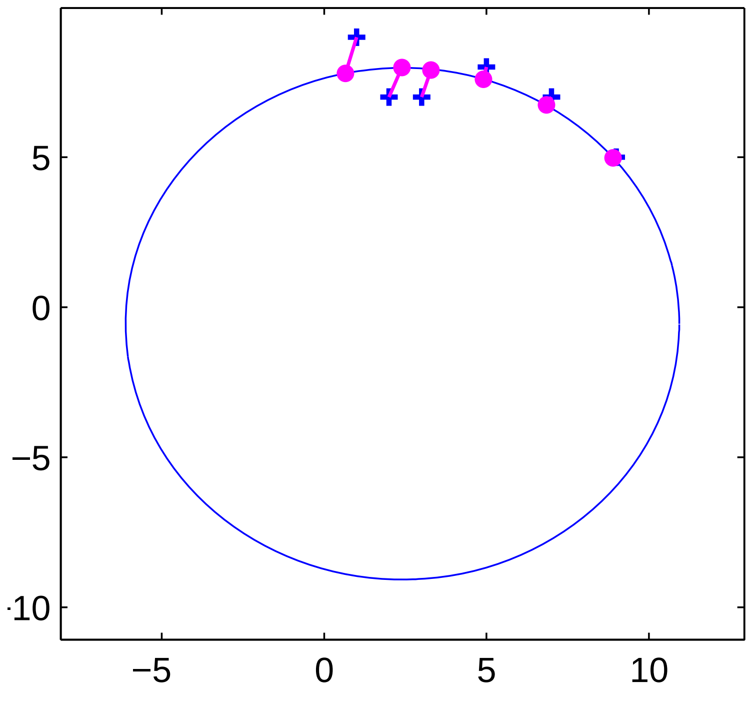


Figure 1.3: Circle fitting of six points.

1.2.3 Embedding on a Sphere (ES)

The goal of this problem is to place some points on a sphere in $\mathbb{R}^r$ in a best way, where $r = 2$ or 3. Particularly, when $r = 2$, the problem is known as circle fitting. The primary information utilized is inter-point distance between each two points. For example, as demonstrated in Figure 1.3, six ground truth points (marked by blue pluses) were given first. Then circle fitting managed to position their corresponding estimated points (marked by pink dots) that were used to find a proper circle, and the circle was able to fit these ground truth points well. Please see more details in Section 5.3.

1.2.4 Dimensionality Reduction (DR)

Dimensionality reduction (DR) problem is from the domain of manifold learning, attempting to reveal several major features from a lot of hidden features of a group of given objects. Let see a particular application in image sciences. The pixel data of a image can be regarded as a point/vector. Some dissimilarities between two images are obtained under a certain rule to form the primary information. The purpose is to find the major features (usually 2 or 3 features) of those images, and then visualize those features through presenting them on a graph.

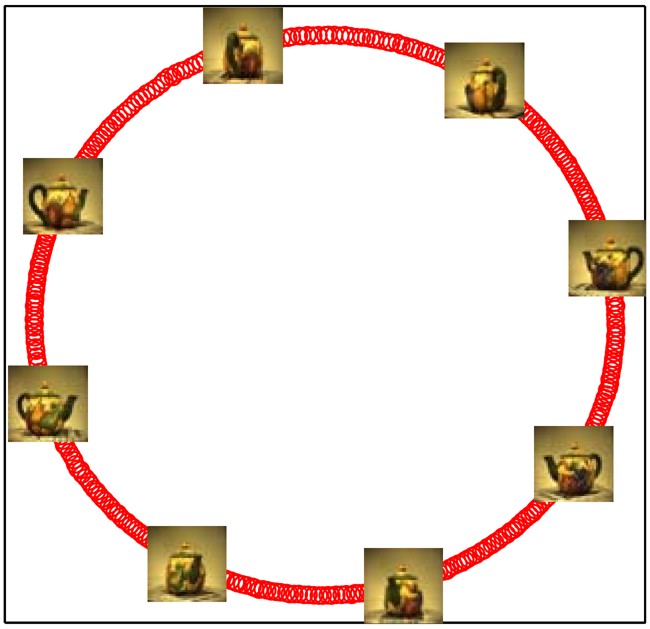


Figure 1.4: Dimensionality reduction of '`teapot`' Data.

For example, a camera is rotated 360 degree to take 400 images of a placed teapot. Each of '`teapot`' images has 76 × 101 pixels, with 3 byte color depth, giving rise to inputs of 23028 dimensions. As described by Weinberger and Saul (2006), though very high dimensional, these images are effectively parametrized by one degree of freedom: the angle of rotation, and two dimensional embedding is able to represent the rotating object as a circle. As presented in Figure 1.4, 400 small red circles form a big circle and each small red circle represents the location of one '`teapot`' image in this two dimensional space. We pick 8 small red circles whose corresponding images are presented in the graph. One can see that the handle of the teapot is rotated 360 degree, which coincides with the rotation of the camera. More examples can be seen in Section 5.4.

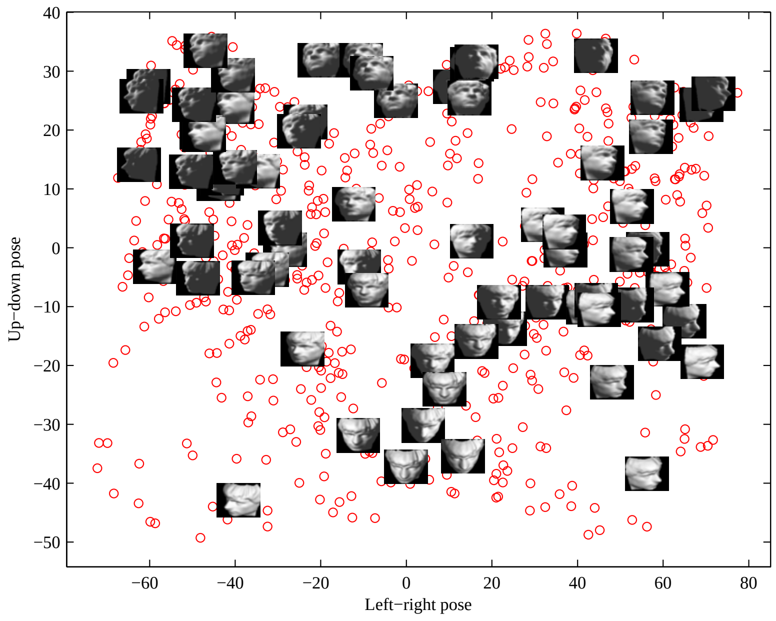


Figure 1.5: Dimensionality reduction of `Face698` Data.

Another example is the `Face698` dataset, which comprises 698 images (64 × 64 pixel) of faces with the different (up-down and left-right) face poses and different light directions. Each image is regarded as an input point/vector with high (64^2) dimension. For the purpose of highlighting the major features of those images: two face poses and light direction, it is natural to expect they lie in a low three dimensional space dominated by these three features. We presented two face poses features in Figure 1.5, from the left to the right side, the direction that the face in each image points to is gradually from the left to the right side as well. Then from the up and the down side, the direction that the face in each image points to is gradually from the up to the down side as well.

1.3 Preliminaries

Before the main part of this thesis ahead of us, we would like to introduce some elementary knowledge that will ease the reading of following contents.

1.3.1 Inner Product

Some useful properties of the Frobenius inner product are summarized below.

1) For $A, B \in \mathbb{R}^{m\times n}$, it holds

$$\langle A, B\rangle = \mathrm{tr}(AB^\top) = \sum_{i=1}^{m}\sum_{i=1}^{n} a_{ij}b_{ij};$$

Particularly, $\langle A, A\rangle = \|A\|^2 = \sum_{ij} a_{ij}^2$;

2) $\langle A, I\rangle = \mathrm{tr}(A) = \sum_i a_{ii} = \sum_i \lambda_i(A)$;

3) For $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{m\times m}$ and $Z \in \mathbb{R}^{n\times m}$, it holds

$$\langle Z^\top AZ, B\rangle = \langle A, ZBZ^\top\rangle \tag{1.3}$$

(since $\mathrm{tr}(Z^\top(AZB^\top)) = \mathrm{tr}((AZB^\top)Z^\top)$).

1.3.2 Principal Components Analysis

Suppose $A \in \mathbb{S}^n$ has the following Eigenvalue Decomposition:

$$A = \lambda_1\mathbf{p}_1\mathbf{p}_1^\top + \lambda_2\mathbf{p}_2\mathbf{p}_2^\top + \cdots + \lambda_n\mathbf{p}_n\mathbf{p}_n^\top, \tag{1.4}$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ are the eigenvalues of A in non-increasing order, and $\mathbf{p}_i$, $i = 1, \ldots, n$ are the corresponding orthonormal eigenvectors. We define a PCA-style matrix truncated at r ($r \leq n$):

$$\mathrm{PCA}_r^+(A) := \sum_{i=1}^{r} \max\{0, \lambda_i\}\mathbf{p}_i\mathbf{p}_i^\top. \tag{1.5}$$

One can verify that $\mathrm{PCA}_r^+(A) \in \mathrm{argmin}_{Y\in\mathbb{S}_+^n(r)}\ \|Y - A\|$.

1.3.3 Projections

We say $\Pi_\Omega(\mathbf{x})$ a projection of $\mathbf{x}$ onto a closed set Ω if it satisfies

$$\Pi_\Omega(\mathbf{x}) \in \underset{\mathbf{z}\in\Omega}{\mathrm{argmin}} \ \|\mathbf{z}-\mathbf{x}\|. \tag{1.6}$$

It is well known that when Ω is a convex set, then $\Pi_\Omega(\mathbf{x})$ is unique. But when Ω is non-convex, generally speaking, there are multiple solutions of (1.6). When such scenario happens, we denote $\Pi^B_\Omega(\mathbf{x})$ its all solutions. Several projections onto particular closed sets interested in this thesis are presented here.

- Projection onto a non-negative set:

$$\Pi_{\mathbb{R}^n_+}(\mathbf{x}) = \max\{\mathbf{x},\mathbf{0}\}; \tag{1.7}$$

  Hereafter, we write $\max\{\mathbf{x},\mathbf{a}\}$ to denote a vector with ith entry $\max\{x_i,a_i\}$

- Let $J = I - \frac{1}{n}\mathbf{e}\mathbf{e}^\top$ be the so-called centring matrix. Projection onto a box set:

$$\Pi_{[\mathbf{a},\mathbf{b}]}(\mathbf{x}) = \min\{\max\{\mathbf{x},\mathbf{a}\},\mathbf{b}\}; \tag{1.8}$$

- Projection onto a subspace $\Omega := \{\mathbf{x}\in\mathbb{R}^n : \mathbf{e}^\top\mathbf{x} = 0\}$:

$$\Pi_\Omega(\mathbf{x}) = J\mathbf{x}, \tag{1.9}$$

- Projection onto a positive semi-definite cone:

$$\Pi_{\mathbb{S}^n_+}(A) = \mathrm{PCA}^+_n(A); \tag{1.10}$$

  where $\mathrm{PCA}^+_n(A)$ is given by (1.5).

- Projection onto a positive semi-definite cone with rank-cut r:

$$\Pi^B_{\mathbb{S}^n_+(r)}(A) = \left\{\mathrm{PCA}^+_r(A)\right\} = \{\Pi_{\mathbb{S}^n_+(r)}(A)\}; \tag{1.11}$$

  $\Pi_{\mathbb{S}^n_+(r)}(A) = \mathrm{PCA}^+_r(A)$ is not unique since $\mathbb{S}^n_+(r)$ is just a special choice of $\Pi^B_{\mathbb{S}^n_+(r)}(A)$, see (1.5) or (Qi and Yuan, 2014, Lemma 2.2) or (Gao, 2010, Lemma 2.9).

1.3.4 Subdifferential

An extended-real-valued function $f : \mathbb{R}^n \to \mathbb{R} := (-\infty, \infty]$ is called *proper* if it is finite somewhere and never equals $-\infty$. The *domain* of f is denoted by *dom*f and is defined as *dom*$f := \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) < +\infty\}$. A function f is said to be *coercive* if $f(\mathbf{x}) \to +\infty$ when $\|\mathbf{x}\| \to \infty$. A function f is *lower semicontinuous* at the point $\mathbf{x}$ if

$$\liminf_{\mathbf{z}\to\mathbf{x}} f(\mathbf{z}) \ \geq f(\mathbf{x}).$$

If f is lower semicontinuous at every point of its domain, then it is called a *lower semi-continuous function*. Such a function is called *closed* if it is lower semicontinuous.

Given a proper function $f : \mathbb{R}^n \to \mathbb{R} := (-\infty, \infty]$, we use the symbol $\mathbf{z} \xrightarrow{f} \mathbf{x}$ to indicate $\mathbf{z} \to \mathbf{x}$ and $f(\mathbf{z}) \to f(\mathbf{x})$. Our basic *subdifferential* of f at $\mathbf{x} \in$ *dom*f (also known as the limiting subdifferential) is defined by

$$\begin{aligned} \partial f(\mathbf{x}) \quad := \quad \Big\{\mathbf{u} \in \mathbb{R}^n \ \Big| \ \exists\ \mathbf{x}^\ell \xrightarrow{f} \mathbf{x}, \mathbf{u}^\ell \to \mathbf{u} \text{ such that, for each } \ell, \\ \liminf_{\mathbf{z}\to\mathbf{x}^\ell} \frac{f(\mathbf{z}) - f(\mathbf{x}^\ell) - \langle \mathbf{u}^\ell, \mathbf{z} - \mathbf{x}^\ell\rangle}{\|\mathbf{z} - \mathbf{x}^\ell\|} \geq 0 \Big\}. \end{aligned} \tag{1.12}$$

where $\{\mathbf{x}^\ell\}$ is the sequence with $\ell = 1, 2, \ldots$. This is refereed as (Rockafellar and Wets, 2009, Definition 8.3) It follows immediately from the above definition that this subdifferential has the following robustness property:

$$\Big\{\mathbf{u} \in \mathbb{R}^n \ \Big| \ \exists\ \mathbf{x}^\ell \xrightarrow{f} \mathbf{x}, \mathbf{u}^\ell \to \mathbf{u}, \mathbf{u}^\ell \in \partial f(\mathbf{x}^\ell)\Big\} \subseteq \partial f(\mathbf{x}). \tag{1.13}$$

For a convex function f the subdifferential (1.12) reduces to the classical subdifferential in convex analysis, see, for example, (Rockafellar and Wets, 2009, Proposition 8.12):

$$\partial f(\mathbf{x}) = \Big\{\mathbf{u} \in \mathbb{R}^n \ \Big| \ f(\mathbf{z}) \geq f(\mathbf{x}) + \langle \mathbf{u}, \mathbf{z} - \mathbf{x}\rangle\Big\}. \tag{1.14}$$

Moreover, for a continuously differentiable function f, the subdifferential (1.12) reduces to the *derivative* of f denoted by $\bigtriangledown f$. For a function f with more than one group of variables, we use $\partial_{\mathbf{x}} f$ (resp., $\bigtriangledown_{\mathbf{x}} f$) to denote the subdifferential (resp., derivative) of f with respect to the variable $\mathbf{x}$. A *critical point* or *stationary point* of f is a point $\bar{\mathbf{x}}$ in the domain of f satisfying

$$\mathbf{0} \in \partial f(\bar{\mathbf{x}}).$$

Finally, a function f is *Lipschitz continuous* with a Lipschitz constant $L_f > 0$ if

$$|f(\mathbf{x}) - f(\mathbf{y})| \le L_f \|\mathbf{x} - \mathbf{y}\|.$$

A function is *gradient Lipschitz continuous* with a Lipschitz constant $L_f > 0$ if

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L_f \|\mathbf{x} - \mathbf{y}\|.$$

1.3.5 Majorization of functions

Let $f : \Omega \to \mathbb{R}$ be a proper function. We say f_M is a majorization of f on Ω if it satisfies

$$f(\mathbf{x}) \le f_M(\mathbf{x}; \mathbf{z}) \quad \text{and} \quad f(\mathbf{x}) = f_M(\mathbf{x}; \mathbf{x}) \tag{1.15}$$

for any $\mathbf{x}, \mathbf{z} \in \Omega$. For example, if f is a concave function (namely, -f is convex), then by (1.14), for any $\mathbf{u} \in \partial f(\mathbf{z})$, its majorization can be

$$f_M(\mathbf{x}; \mathbf{z}) := f(\mathbf{z}) + \langle \mathbf{u}, \mathbf{x} - \mathbf{z} \rangle,$$

Another example is the gradient Lipschitz continuous function f, and its majorization is able to be

$$f_M(\mathbf{x}; \mathbf{z}) := f(\mathbf{z}) + \langle \nabla f(\mathbf{z}), \mathbf{x} - \mathbf{z} \rangle + (L_f/2)\|\mathbf{x} - \mathbf{z}\|^2,$$

where $L_f > 0$ is the Lipschitz constant, see (Nesterov, 1998, Theorem 2.1.5).

The third example is the distance function, i.e., $f = \|\mathbf{x} - \Pi_\Omega(\mathbf{x})\|$, where Ω is a closed set, and its majorization allows to be

$$f_M(\mathbf{x}; \mathbf{z}) := \|\mathbf{x} - \Pi_\Omega(\mathbf{z})\|.$$

1.3.6 Roots of Depressed Cubic Equation

In our algorithm, we will encounter the positive root of a depressed cubic equation (Burton, 2011, Chp. 7), which arises from the optimality condition of the problem

$$\min_{x \ge 0} \quad s(x) := (1/2)(x - \omega)^2 + 2\nu\sqrt{x}, \tag{1.16}$$

where $\nu \neq 0$ and $\omega \in \mathbb{R}$ are given. A positive stationary point x must satisfy the optimality condition

$$0 = s'(x) = x - \omega + \nu/\sqrt{x}. \tag{1.17}$$

Let $z := \sqrt{x}$. The optimality condition above becomes

$$z^3 - \omega z + \nu = 0. \tag{1.18}$$

This is the classical form of the so-called depressed cubic equation (Burton, 2011, Chp. 7). Its roots (complex or real) and their computational formulae have a long history with fascinating and entertaining stories. A comprehensive revisit of this subject can be found in (Xing, 2003) and a successful application when $\nu > 0$ to the compressed sensing can be found in (Xu et al., 2012; Peng et al., 2017). The following lemma says that, under certain conditions, the equation (1.17) when $\nu > 0$ has two distinctive positive roots and its proof is a specialization of (Chen et al., 2014, Lem. 2.1(iii)) when $p = 1/2$.

Lemma 1.1. *(Chen et al., 2014, Lemma 2.1(iii)) For (1.16) with $\nu > 0$, let*

$$\bar{x} = (\nu/2)^{2/3} \qquad \textit{and} \qquad \bar{\omega} = 3\bar{x}.$$

When $\omega > \bar{\omega}$, $s(x)$ has two different positive stationary point x_1^* and x_2^* satisfying

$$s'(x) = 0 \qquad \textit{and} \qquad x_1^* < \bar{x} < x_2^*.$$

Next we focus on roots of (1.18) under several cases of interest in this thesis. Based on Candano' formula in (Burton, 2011, Chp. 7) and results in (Xing, 2003, Tables 3 and 4), we summarize associated properties as follows.

Lemma 1.2. *Consider the depressed cubic equation (1.18) with $\nu \neq 0$ and $\omega \in \mathbb{R}$. Let*

$$\tau := \nu^2/4 - \omega^3/27.$$

(i) If $\tau > 0$, then (1.18) has two conjugate complex roots and one real root. And the real root can be computed by

$$z = \left[-\nu/2 + \sqrt{\tau}\right]^{1/3} + \left[-\nu/2 - \sqrt{\tau}\right]^{1/3}. \tag{1.19}$$

For clarity, the real value of $a^{1/3}$ is calculated by

$$a^{1/3} = \begin{cases} a^{1/3}, & \textit{if} \quad a > 0, \\ 0, & \textit{if} \quad a = 0, \\ -(-a)^{1/3}, & \textit{if} \quad a < 0. \end{cases}$$

(ii) If $\tau < 0$, then (1.18) has three distinct real roots which can be computed by

$$z_1 = 2\sqrt{\frac{\omega}{3}}\cos\left[\frac{\theta}{3}\right], \quad z_2 = 2\sqrt{\frac{\omega}{3}}\cos\left[\frac{\theta+4\pi}{3}\right], \quad z_3 = 2\sqrt{\frac{\omega}{3}}\cos\left[\frac{\theta+2\pi}{3}\right],$$

where $\cos(\theta) = \frac{\nu}{2}(\frac{\omega}{3})^{-3/2}$ and $\theta \in (0, \pi/2) \cup (\pi/2, \pi)$. Moreover,

$$z_1 > \max\{z_2, 0\} > \min\{z_2, 0\} > z_3.$$

(iii) If $\tau = 0$, then (1.18) has three real roots which can be computed by

$$z_1 = z_2 = -\left[-\frac{\nu}{2}\right]^{1/3}, \qquad z_3 = -\frac{3\nu}{\omega}.$$

Moreover, $z_1 = z_2 > 0, z_3 < 0$ if $\nu > 0$ and $z_1 = z_2 < 0, z_3 > 0$ if $\nu < 0$.

1.3.7 Proximal Alternating Direction Methods of Multipliers

Let us consider the following model

$$\begin{aligned} \min \quad & f_1(\mathbf{x}) + f_2(\mathbf{y}) \\ \text{s.t.} \quad & \mathbb{A}\mathbf{x} + \mathbb{B}\mathbf{y} = \mathbf{b}, \\ & \mathbf{x} \in \mathcal{X}, \quad \mathbf{y} \in \mathcal{Y}, \end{aligned} \tag{1.20}$$

where $f_1 : \mathcal{X} \to \mathbb{R}$ and $f_2 : \mathcal{Y} \to \mathbb{R}$; $\mathbb{A} : \mathcal{X} \to \mathcal{Z}$ and $\mathbb{B} : \mathcal{Y} \to \mathcal{Z}$ are two linear operators, $\mathbf{b} \in \mathcal{Z}$; and $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ are real finite dimensional Euclidean spaces with inner product $\langle \cdot, \rangle$ and its induced norm $\| \cdot \|$. The augmented Lagrange function of (1.20) is

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{z}) := f_1(\mathbf{x}) + f_2(\mathbf{y}) - \langle \mathbf{z}, \mathbb{A}\mathbf{x} + \mathbb{B}\mathbf{y} \rangle + (\sigma/2)\|\mathbb{A}\mathbf{x} + \mathbb{B}\mathbf{y}\|^2, \tag{1.21}$$

for any $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$ and any given $\sigma > 0$, where $\mathbf{z}$ is the so-called Lagrange Multiplier. When (1.20) is a convex model, namely, $f_1 : \mathcal{X} \to \mathbb{R}$ and $f_2 : \mathcal{Y} \to \mathbb{R}$ are proper convex function on $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$ respectively, the Proximal Alternating Direction Methods of Multipliers (pADMM), is extensively used to tackle it. The algorithmic framework is described in Table 1.1.

Table 1.1: The framework of pADMM

Step 0 Let $\sigma, \tau > 0$ and $\mathbb{P}, \mathbb{Q}$ be given self-adjoint and positive semi-definite linear operators. Choose $(\mathbf{x}^0, \mathbf{y}^0, \mathbf{z}^0)$. Set $k := 0$.

Step 1 Perform the $(k+1)$-th iteration as follows

$$\mathbf{x}^{k+1} = \operatorname*{argmin}_{\mathbf{x}\in\mathcal{X}} \ \mathcal{L}(\mathbf{x}, \mathbf{y}^k, \mathbf{z}^k) + (1/2)\|\mathbf{x} - \mathbf{x}^k\|_{\mathbb{P}}^2, \tag{1.22}$$

$$\mathbf{y}^{k+1} = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{Y}} \ \mathcal{L}(\mathbf{x}^{k+1}, \mathbf{y}, \mathbf{z}^k) + (1/2)\|\mathbf{y} - \mathbf{y}^k\|_{\mathbb{Q}}^2, \tag{1.23}$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k - \tau\sigma(\mathbb{A}\mathbf{x}^{k+1} + \mathbb{B}\mathbf{y}^{k+1}), \tag{1.24}$$

Step 2 Set $k := k+1$ and go to Step 1 until convergence.

Here $\|\mathbf{x}\|_{\mathbb{P}}^2 := \langle \mathbf{x}, \mathbb{P}\mathbf{x}\rangle$. The purpose of adding the proximal terms $\|\mathbf{x}-\mathbf{x}^k\|_{\mathbb{P}}^2$ and $\|\mathbf{y}-\mathbf{y}^k\|_{\mathbb{Q}}^2$ basically is to enable the first two subproblems to be well defined (i.e., to have unique solution) on one side, and to be easily calculated on the other side. Notice that when $\mathbb{P} = \mathbf{0}$ and $\mathbb{Q} = \mathbf{0}$, pADMM reduces to the standard ADMM. For convex problem (1.20), its convergence property has been well established, seen (Fazel et al., 2013, Theorem B.1), which can be stated as follows.

Theorem 1.3. *Assume that the intersection of the relative interior of* (*dom* $f_1 \times$ *dom* f_2) *and the constraint set of (1.20) is non-empty. Let the sequence* $\{\boldsymbol{x}^k, \boldsymbol{y}^k, \boldsymbol{z}^k\}$ *be generated by pADMM in Table 1.1. Choose* $\mathbb{P}, \mathbb{Q}$ *such that* $\mathbb{P} + \sigma\mathbb{A}^*\mathbb{A}$ *and* $\mathbb{Q} + \sigma\mathbb{B}^*\mathbb{B}$ *are positive definite and* $\tau \in (0, (\sqrt{5}-1)/2)$*, then the sequence* $\{\boldsymbol{x}^k, \boldsymbol{y}^k\}$ *converges to an optimal solution to (1.20) and* $\{\boldsymbol{z}^k\}$*converges to an optimal solution to the dual problem of (1.20).*

1.4 Euclidean Distance Embedding

Euclidean Distance Embedding (EDE) turns out to be relevant to three elements: The definition of Euclidean Distance Matrix (EDM), characterizations of EDM and Euclidean Embedding associated with Procrustes analysis (Cox and Cox, 2000, Chap. 5).

1.4.1 Euclidean Distance Matrix (EDM)

A matrix $D \in \mathbb{S}^n$ is an EDM if there exist points $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ in $\mathbb{R}^r$ such that

$$D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2, \quad i, j = 1, \ldots, n.$$

Here $\mathbb{R}^r$ is often referred to as the *embedding space*, r is the *embedding dimension* when it is the smallest such r. The above equations mean that each element D_{ij} of D equals to the squared pairwise distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$, and because of this, D is symmetric obviously. For example, we give three points $\mathbf{x}_1 = (0,0)^\top, \mathbf{x}_2 = (1,0)^\top, \mathbf{x}_3 = (0,2)^\top$ in $\mathbb{R}^2$. Direct calculation derives an EDM $D = [0\ 1\ 4; 1\ 0\ 5; 4\ 5\ 0]$.

1.4.2 Characterizations of EDM

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

$$D \in \mathbb{S}^n_h, \quad J(-D)J \succeq \mathbf{0}. \tag{1.25}$$

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 (1.25). Moreover, the corresponding embedding dimension is

$$r = \mathrm{rank}(JDJ).$$

From (1.9), the centring matrix $J = I - \frac{1}{n}\mathbf{e}\mathbf{e}^\top$ satisfies $\mathbf{e}^\top J\mathbf{z} = 0$ for any $\mathbf{z} \in \mathbb{R}^n$. This combining with $J(-D)J \succeq \mathbf{0} \Leftrightarrow \mathbf{z}^\top J(-D)J\mathbf{z} \geq 0$ suffices to following equivalence :

$$D \in \mathbb{S}^n_h, \quad J(-D)J \succeq \mathbf{0} \quad \Longleftrightarrow \quad D \in \mathbb{S}^n_h, \quad -D \in \mathbb{K}^n_+, \tag{1.26}$$

where $\mathbb{K}^n_+$ is known as *conditional positive semi-definite cone*, defined by

$$\begin{aligned} \mathbb{K}^n_+ &:= \{A \in \mathbb{S}^n : \ \mathbf{x}^\top A\mathbf{x} \geq 0, \ \forall\ \mathbf{e}^\top\mathbf{x} = 0\} \\ &= \{A \in \mathbb{S}^n : \ JAJ \succeq \mathbf{0}\}. \end{aligned} \tag{1.27}$$

A nice property of the cone $\mathbb{K}^n_+$ is that the projection of any $A \in \mathbb{S}^n$ onto it can be derived through the orthogonal projection onto the positive semi-definite cone $\mathbb{S}^n_+$, seen

(Gaffke and Mathar, 1989) for more details,

$$\Pi_{\mathbb{K}^n_+}(A) = A + \Pi_{\mathbb{S}^n_+}(-JAJ). \tag{1.28}$$

Define a *conditional positive semi-definite cone with rank cut* r by

$$\mathbb{K}^n_+(r) \quad := \quad \mathbb{K}^n_+ \cap \{A \in \mathbb{S}^n : \ \mathrm{rank}(JAJ) \le r\}. \tag{1.29}$$

Overall, an EDM D with embedding dimension r is equivalent to

$$-D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+(r), \tag{1.30}$$

Regarding to $\mathbb{K}^n_+(r)$, the following proposition holds from (Qi and Yuan, 2014).

Proposition 1.4. *For given $A \in \mathbb{S}^n$ and an integer $r \le n$. The following results hold.*

(i) (Qi and Yuan, 2014, Eq. (26), Prop. 3.3) We have

$$\langle \Pi_{\mathbb{K}^n_+(r)}(A), \ A - \Pi_{\mathbb{K}^n_+(r)}(A) \rangle = 0.$$

(ii) (Qi and Yuan, 2014, Prop. 3.4) The function

$$h(A) := (1/2)\|\Pi_{\mathbb{K}^n_+(r)}(A)\|^2$$

is well defined and is convex. Moreover, denoting $\mathrm{conv}(S)$ *the convex hall of set S, we have*

$$\Pi_{\mathbb{K}^n_+(r)}(A) \in \partial h(A) = \mathrm{conv}\left(\Pi^B_{\mathbb{K}^n_+(r)}(A)\right).$$

(iii) (Qi and Yuan, 2014, Eq. (22), Prop. 3.3) One particular projection $\Pi_{\mathbb{K}^n_+(r)}(A)$ can be computed through

$$\Pi_{\mathbb{K}^n_+(r)}(A) = \mathrm{PCA}^+_r(JAJ) + (A - JAJ) \tag{1.31}$$

The benefit of having Proposition 1.4 is that the feasibility of a matrix to be a low-rank EDM can be represented by a well-behaved function as we see below:

$$-A \in \mathbb{K}^n_+(r) \iff A + \Pi_{\mathbb{K}^n_+(r)}(-A) = 0$$

This is equivalent to require $g(A) = 0$, where

$$g(A) := (1/2)\|A + \Pi_{\mathbb{K}^n_+(r)}(-A)\|^2. \tag{1.32}$$

Proposition 1.4 allows us to represent $g(A)$ in terms of $h(A)$. This relationship is so important that we include it in the proposition below.

Proposition 1.5. *For any $A \in \mathbb{S}^n$, we have following results.*

(i) $g(A) = \|A\|^2/2 - h(-A)$. *Hence, $g(A)$ is a difference of two convex functions;*

(ii) $g(A) \le (1/2)\|A + \Pi_{\mathbb{K}^n_+(r)}(-B)\|^2 =: g_M(A;B)$ *for any $B \in \mathbb{S}^n$; That is $g_M(A;B)$ can be a majorization of $g(A)$ based on (1.15);*

(iii) $\|\Pi_{\mathbb{K}^n_+(r)}(A)\| \le 2\|A\|$.

Proof (i) It follows from Proposition 1.4(i) that

$$\langle -A,\ \Pi_{\mathbb{K}^n_+(r)}(-A)\rangle = \|\Pi_{\mathbb{K}^n_+(r)}(-A)\|^2.$$

Substituting this into the first equation below to get

$$\begin{aligned} g(A) &= (1/2)\|A\|^2 + (1/2)\|\Pi_{\mathbb{K}^n_+(r)}(-A)\|^2 + \langle A,\ \Pi_{\mathbb{K}^n_+(r)}(-A)\rangle \\ &= (1/2)\|A\|^2 + (1/2)\|\Pi_{\mathbb{K}^n_+(r)}(-A)\|^2 - \|\Pi_{\mathbb{K}^n_+(r)}(-A)\|^2 \\ &= (1/2)\|A\|^2 - (1/2)\|\Pi_{\mathbb{K}^n_+(r)}(-A)\|^2 \\ &= (1/2)\|A\|^2 - h(-A). \end{aligned}$$

(ii) This is clear because of $\Pi_{\mathbb{K}^n_+(r)}(B) \in \mathbb{K}^n_+(r)$ and for any $Y \in \mathbb{K}^n_+(r)$

$$\|\Pi_{\mathbb{K}^n_+(r)}(A) - A\| = \min_{Y \in \mathbb{K}^n_+(r)} \|Y - A\| \le \|Y - A\|$$

(iii) Since $\mathbf{0} \in \mathbb{K}^n_+(r)$, by the definition of $\Pi_{\mathbb{K}^n_+(r)}(\cdot)$, we have

$$\|\Pi_{\mathbb{K}^n_+(r)}(A)\| - \|A\| \le \|\Pi_{\mathbb{K}^n_+(r)}(A) - A\| \le \|\mathbf{0} - A\| = \|A\|,$$

which yields the last claim. $\square$

1.4.3 Euclidean Embedding with Procrustes Analysis

If D is an EDM with embedding dimension r, then $J(-D)J \succeq \mathbf{0}$ by (1.30), and $J(-D)J$ can be decomposed from Table 2.1 as

$$-JDJ/2 = X^\top X, \tag{1.33}$$

where $X := [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n] \in \mathbb{R}^{r\times n}$. It is known that $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ are the embedding points of D in $\mathbb{R}^r$ such that $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. We also note that any rotation and shifting of $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ would give same D. In other words, there are infinitely many sets of embedding points. To find a desired set of embedding points that match positions of certain existing points, one needs to conduct the Procrustes analysis, which is a computational scheme and often has a closed-form formula, see (Cox and Cox, 2000, Chp. 5). The procedure is as follows.

Centralizing: Let $X = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]$ and $Z = [\mathbf{z}_1 \ \cdots \ \mathbf{z}_n]$ be the estimated and the ground truth embedding respectively. We first move X (resp. Z) along $\mathbf{x}_c = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i = \frac{1}{n}X\mathbf{e}$ (resp. $\mathbf{z}_c = \frac{1}{n}\sum_{i=1}^n \mathbf{z}_i = \frac{1}{n}Z\mathbf{e}$) to X_c (resp. Z_c) where the center of X_c and Z_c is the origin, namely,

$$X_c = X - \mathbf{x}_c\mathbf{e}^\top, \qquad Z_c = Z - \mathbf{z}_c\mathbf{e}^\top. \tag{1.34}$$

One can check the center of X_c is origin due to $X_c\mathbf{e} = X\mathbf{e} - \mathbf{x}_c\mathbf{e}^\top\mathbf{e} = \sum_{i=1}^n \mathbf{x}_i - n\mathbf{x}_c = \mathbf{0}$ and same as Z_c due to $Z_c\mathbf{e} = \mathbf{0}$.

Rotating The best rotational (including rotations and flips) embedding on X_c and Z_c can be done through solving the orthogonal Procrustes problem:

$$P^* = \mathrm{argmin}_{P\in\mathbb{R}^{r\times r}} \ \ \|PX_c - Z_c\|, \ \ \text{s.t. } P^\top P = I. \tag{1.35}$$

The matrix P enables the columns of X_c in a best way to match the corresponding columns of Z_c after the rotation. Problem (1.35) has a closed form solution $P^* = UV^\top$, where U and V are from the singular-value-decomposition of $Z_cX_c^\top = U\Lambda V^\top$ with the standard meaning of U, Λ and V. Then the points matching Z_c are

$$Z_P := P^*X_c = UV^\top X_c.$$

Matching: We finally move Z_P to Z_{new} that matches Z by

$$Z_{\text{new}} = Z_P + \mathbf{z}_c \mathbf{e}^\top. \tag{1.36}$$

Consider one example to illustrate this. Let

$$Z = \begin{bmatrix} -\frac{\sqrt{2}}{2} - 2 & \frac{\sqrt{2}}{2} - 2 & \frac{\sqrt{2}}{2} - 2 & -\frac{\sqrt{2}}{2} - 2 \\ \frac{\sqrt{2}}{2} - 2 & \frac{\sqrt{2}}{2} - 2 & -\frac{\sqrt{2}}{2} - 2 & -\frac{\sqrt{2}}{2} - 2 \end{bmatrix}, \quad X = \begin{bmatrix} 3 & 2 & 1 & 2 \\ 2 & 3 & 2 & 1 \end{bmatrix}.$$

It is easy to calculate $\mathbf{x}_c = \mathbf{z}_c = [2\ 2]^\top$. The singular-value-decomposition of $Z_c X_c^\top = U\Lambda V^\top$ yields that

$$U = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}, \quad V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then it is easy to verify that $Z_P = P^* X_c = UV^\top X_c = Z_c$ and $Z_{\text{new}} = Z_P + \mathbf{z}_c\mathbf{e}^\top = Z$. Actually, this procedure can be illustrated by Figure 1.6. It is worth mentioning that Z_P and Z_c are not exactly same in general after rotating, that is $Z_P \approx Z_c$ which thus indicates $Z_{\text{new}} = Z_P + \mathbf{z}_c\mathbf{e}^\top \approx Z$.

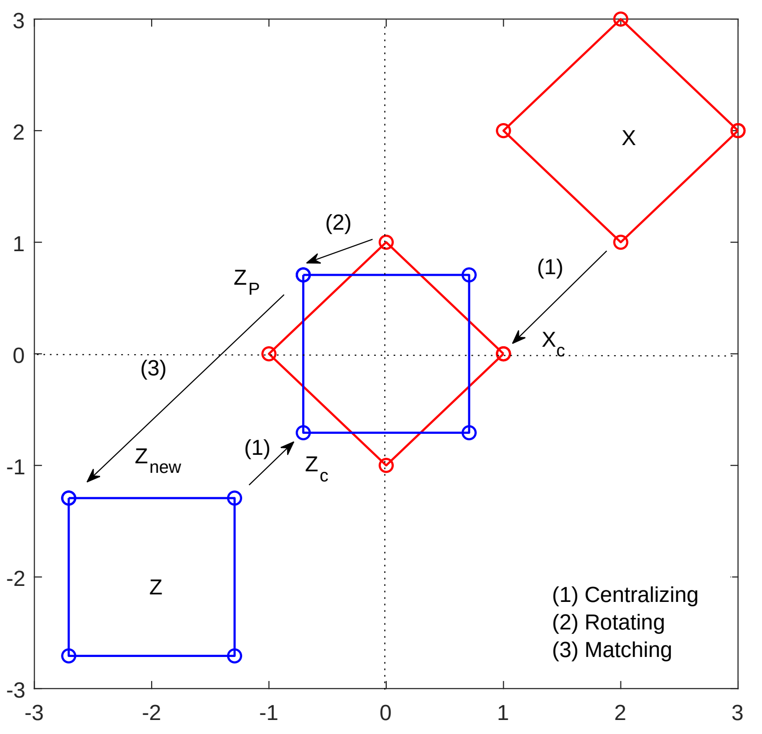


Figure 1.6: Procrustes analysis.

Chapter 2

Literature Review

A great deal of effort has been made to seek the best approximation for problem (1.1). This chapter starts with introducing a very traditional and powerful approach, the classical MDS (cMDS), followed then by summarizing four advanced alternatives to overcome the shortages of cMDS, and ends up with reviewing three groups of approaches that have been used to solve the four models.

2.1 Classical MDS

The scheme of the classical MDS method can be described in Table 2.1

Table 2.1: The procedure of cMDS.

Step 0	Give $\Delta \in \mathbb{S}^n_h$ by (1.2) and r.
Step 1	Spectral decomposition: $$-\frac{1}{2}J\Delta^{(2)}J = \lambda_1 \mathbf{p}_1\mathbf{p}_1^\top + \lambda_2 \mathbf{p}_2\mathbf{p}_2^\top + \cdots + \lambda_n \mathbf{p}_n\mathbf{p}_n^\top \qquad (2.1)$$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of $-J\Delta^{(2)}J/2$ and $\mathbf{p}_1, \cdots, \mathbf{p}_n$ are the corresponding orthonormal eigenvectors.
Step 2	Embedding points from the columns of $X := [\mathbf{x}_1 \cdots \mathbf{x}_n] \in \mathbb{R}^{r\times n}$, where $$\mathbf{x}_i = \left[\sqrt{\lambda_1}\mathbf{p}_{1i} \ \sqrt{\lambda_2}\mathbf{p}_{2i} \ \cdots \ \sqrt{\lambda_r}\mathbf{p}_{ri}\right]^\top, \quad i = 1, \ldots, n.$$

Actually, cMDS solves the following optimization problem:

$$Y^* \in \mathrm{argmin}_{Y \in \mathbb{S}^n_+(r)} \|Y - (-J\Delta^{(2)}J/2)\|. \tag{2.2}$$

The solution is $Y^* = X^\top X$, where $X \in \mathbb{R}^{r\times n}$ is given as in Table 2.1, namely,

$$X = \left[\sqrt{\lambda_1}\mathbf{p}_1 \ \sqrt{\lambda_2}\mathbf{p}_2 \ \cdots \ \sqrt{\lambda_r}\mathbf{p}_r\right]^\top.$$

The popularity of cMDS benefits from three main aspects:

- Simplicity of implementations. This is because of the simple scheme of cMDS;
- Low complexity of computations. The main computational step is the spectral decomposition of $J\Delta^{(2)}J$, whose complexity is $\mathcal{O}(n^3)$. And thus the complexity of whole procedure is also $\mathcal{O}(n^3)$;
- Desirable accuracy of embedding. When the given pairwise dissimilarities δ_{ij}^2 are close enough to the true inter-vector distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ of objects, it is capable of rendering an embedding with acceptable accuracy indeed.

However, cMDS takes advantage of the double-centring matrix $J\Delta^{(2)}J$, which limits its implementations to scenarios where large numbers of elements are available and the noises are quite small, i.e., small ϵ_{ij} in (1.2). However, under circumstances that some of elements (or even one single element) of Δ are contaminated by slightly large noise/outliers, not to mention when large number of elements of Δ are missing, it performs poorly because the double centring procedure $J(\cdot)J$ spreads out the error stemmed from the noise to the entire matrix $J\Delta^{(2)}J$. More detailed explanations can be found in (Spence and Lewandowsky, 1989; Cayton and Dasgupta, 2006).

From (1.9), the centring matrix $J = I_n - \mathbf{e}\mathbf{e}^\top/n =: J_n$ moves the center of every $\mathbf{x}$ to the origin (i.e., $\mathbf{e}^\top J\mathbf{x} = 0$). For a matrix $X = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n] \in \mathbb{R}^{n\times n}$, it holds

$$JXJ = X - \frac{1}{n}\mathbf{e}\mathbf{e}^\top X - \frac{1}{n}X\mathbf{e}\mathbf{e}^\top - \frac{\mathbf{e}^\top X\mathbf{e}}{n^2}\mathbf{e}\mathbf{e}^\top =: [\mathbf{y}_1 \ \cdots \ \mathbf{y}_n].$$

Then one can calculate that $\mathbf{e}^\top JXJ = JXJ\mathbf{e} = \mathbf{0}$, which means JXJ moves $[\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]$ to $[\mathbf{y}_1 \ \cdots \ \mathbf{y}_n]$ with their center being origin. This is why J is called centring matrix. Moreover, J is also plays an important role in statistics. For example, consider a sample matrix $A \in \mathbb{R}^{m\times n}$. $J_m A$ and AJ_n respectively remove the means from each of the n columns and m rows. Therefore double-centring $J_m AJ_n$ places the mean to be zero.

2.2 Stress-based Minimizations

To overcome the above mentioned drawbacks of cMDS, four advanced alternatives have been investigated for several decades. They are stress minimization, squared stress minimization, robust MDS and robust Euclidean embedding.

2.2.1 Stress Minimization

When the least-squares criterion and the dissimilarities were applied to (1.1), we have the popular model known as the Kruskal's stress (Kruskal, 1964) minimization:

$$\min_X \sum_{i,j=1}^{n} W_{ij}\left[\|\mathbf{x}_i - \mathbf{x}_j\| - \delta_{ij}\right]^2. \tag{2.3}$$

where $W_{ij} > 0$ if $\delta_{ij} > 0$ and $W_{ij} \geq 0$ otherwise (W_{ij} can be treated as a weight for the importance of δ_{ij}), and $X := [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ with each $\mathbf{x}_i$ being a column vector. To address this problem, a famous representative solver `SMACOF` (Scaling by Majorizing a Complicated Function) has been created (De et al., 1977; De Leeuw and Mair, 2011).

2.2.2 Squared Stress Minimization

When the least-squares criterion and the squared dissimilarities were applied to (1.1), we get the so-called squared stress minimization (Glunt et al., 1991; Kearsley et al., 1995; Borg and Groenen, 2005):

$$\min_X \sum_{i,j=1}^{n} W_{ij}\left[\|\mathbf{x}_i - \mathbf{x}_j\|^2 - \delta_{ij}^2\right]^2. \tag{2.4}$$

As stated by Kearsley et al. (1995), squared stress would make the computation more manageable than stress criterion because it is everywhere smooth. In addition, Borg and Groenen (2005) emphasized that this cost function tends to prefer large distances over the local distances.

2.2.3 Robust MDS

Another robust criterion, often known as Robust MDS, is defined by

$$\min_{X} \sum_{i,j=1}^{n} W_{ij} \left| \, \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \delta_{ij}^2 \, \right|. \tag{2.5}$$

The preceding problem is robust because of the robustness of the ℓ_1 norm used to quantify the errors (Mandanas and Kotropoulos, 2017, Sect. IV). This problem can be solved through SDP relaxation methods, such as (Biswas et al., 2006; Wang et al., 2008; Pong, 2012; Kim et al., 2012) in which the last one contributes to a comprehensive and famous Matlab package `SFSDP`.

2.2.4 Robust Euclidean Embedding

The most robust criterion to quantify the best approximation to (1.1) is the Robust Euclidean Embedding (REE) defined by

$$\min_{X} \sum_{i,j=1}^{n} W_{ij} \left| \, \|\mathbf{x}_i - \mathbf{x}_j\| - \delta_{ij} \, \right|. \tag{2.6}$$

In contrast to all other three problems mentioned above, there lacks efficient methods for the REE problem (2.6). One of the earliest computational papers that discussed this problem was Heiser (1988), which was followed up by Korkmaz and van der Veen (2009), where the Huber smoothing function was used to approximate the ℓ_1 norm near zero with a majorization technique. It was emphasized by Korkmaz and van der Veen (2009) that "*the function is not differentiable at its minimum and is hard to majorize, leading to a degeneracy that makes the problem numerically unstable*". The difficulty in solving (2.6) is well illustrated by a sophisticated Semi-definite Programming (SDP) approach in (Oguz-Ekim et al., 2011, Sect. IV) below.

2.3 Existing Methods

One can find a thorough review on all of the four problems by France and Carroll (2011), mainly from the perspective of applications, where the ℓ_1 norm and the ℓ_2 norm are respectively referred to as L_1-metric and L_2-metric. In particular, there contains a detailed and well-referenced discussion on the properties and use of the L_1 and L_2 metrics. One can also find valuable discussion on some of those problems in Introduction by An and Tao (2003). So the starting point of our review is that those problems have their own reasons to be studied and we are more concerned how they can be efficiently solved. Most of existing algorithms can be put in three groups: *alternating coordinates descent methods*, *Semi-Definite Programming* (SDP) and *Euclidean Distance Matrix optimization* (EDM). Below a more detailed review is given.

2.3.1 Alternating Coordinates Descent Approach

Those methods have main variables $\mathbf{x}_i$, $i = 1, \ldots, n$. A famous representative in this group is the method of `SMACOF` (Scaling by Majorizing a Complicated Function) for the stress minimization (2.3) (De et al., 1977; De Leeuw and Mair, 2011). The key idea is to alternatively minimize the objective function with respect to each $\mathbf{x}_i$, while keeping other points $\mathbf{x}_j$ $(j \neq i)$ unchanged, and each minimization problem is relatively easier to solve by employing the technique of *majorization*.

`SMACOF` is essentially a gradient based method, which has been proved that the sequence constructed by the majorization function is monotone decreasing and converges (to a local optimum), but also suffers from the typical slow convergence associated with first order optimization methods. To supplement on this, a single iteration of `SMACOF` requires the computation of the Euclidean pairwise distances between all points participating in the optimization at their current configuration, a time consuming task on its own, which limits its application to small size data. `SMACOF` has been widely used and the interested reader can refer to (Borg and Groenen, 2005) for more references and to (Zhang et al., 2010) for some critical comments on `SMACOF` when it is applied to the sensor network localization problem.

To overcome those drawbacks, authors in (Groenen, 1993; Trejos et al., 2000) constructed 'tunnels' in the `SMACOF`'s majorization function, aiming to find the global minima (but not guaranteeing to find it in practice). In (Rosman et al., 2008), vector extrapolation was utilized to accelerate the convergence rate of `SMACOF`.

Very recently, subspace methods have drawn much attention, including user-assisted method (Lawrence et al., 2011) in image processing and spectral `SMACOF` (Boyarski et al., 2017). Its crucial idea is to restrict the solution to lie within a carefully chosen subspace, making such kind of approaches feasible for large data sets. For example, Boyarski et al. (2017) built spectral `SMACOF` to restrict the embedding to lie within a low-dimensional subspace to reduce the dependence of `SMACOF` on the number of objects, which accelerated stress majorization by a significant amount.

Notice that all above methods were proposed to deal with stress minimization (2.3).

2.3.2 SDP Approach

We note that each of the four objective functions either involves the Euclidean distance $d_{ij} := \|\mathbf{x}_i - \mathbf{x}_j\|$ or its squared $D_{ij} = d_{ij}^2$. A crucial observation is that constraints on them often have SDP relaxations. For example, as $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$, it is easy to obtain

$$D_{ij} = d_{ij}^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \quad = \quad \|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_i^T\mathbf{x}_j = Y_{ii} + Y_{jj} - 2Y_{ij}, \qquad (2.7)$$

where $Y := X^TX \succeq 0$. Hence, the squared distance d_{ij}^2 is a linear function of the positive semi-definite matrix Y. Consequently, D being an EDM (i.e., $-D \in \mathbb{S}_h^n \cap \mathbb{K}_+^n$ from (1.26)) can be presented via linear transformations (2.7) of positive semi-definite matrices. One can further relax $Y = X^TX$ to $Y \succeq X^TX$. By the Schur-complement,

$$Z := \begin{bmatrix} Y & X^T \\ X & I_r \end{bmatrix} \succeq 0 \ \text{ has rank } r \quad \Longleftrightarrow \quad Y = X^TX. \qquad (2.8)$$

By dropping the rank constraint, the robust MDS problem (2.5) can be relaxed to a SDP, which was initiated by Biswas and Ye (2004).

For the Euclidean distance d_{ij}, we introduce a new variable $T_{ij} = d_{ij}$. One may relax this constraint to $T_{ij} \le d_{ij}$, which has a SDP representation:

$$T_{ij}^2 \le d_{ij}^2 = D_{ij} \quad \Longleftrightarrow \quad \begin{bmatrix} 1 & T_{ij} \\ T_{ij} & D_{ij} \end{bmatrix} \succeq 0. \qquad (2.9)$$

Combination of (2.7), (2.8) and (2.9) leads to a large number of SDP relaxations.

- For the stress problem (2.3), a typical SDP relaxation can be found in (Oguz-Ekim et al., 2011, Problem (8)).
- For the squared stress (2.4), one may refer to (Jiang et al., 2014; Drusvyatskiy et al., 2017).
- For the robust MDS (2.5), there are the SDP relaxation method (Biswas et al., 2006) and the edge-based SDP relaxation method (Wang et al., 2008; Pong, 2012) and (Kim et al., 2012) which leads to a comprehensive Matlab package `SFSDP`.

However, unlike the problems (2.3), (2.4) and (2.5), the REE problem (2.6) does not have a straightforward SDP relaxation. We use an attempt made by Oguz-Ekim et al. (2011) to illustrate this point below.

Firstly, it follows from $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ and (1.26) that problem (2.6) can be written in terms of EDM:

$$\begin{array}{ll} \min\limits_{D\in\mathbb{S}^n} & \sum_{i,j=1}^n W_{ij}|\sqrt{D_{ij}} - \delta_{ij}| \\ \text{s.t.} & -D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+(r). \end{array} \tag{2.10}$$

The term $|\sqrt{D_{ij}} - \delta_{ij}|$ is convex of $\delta_{ij} > \sqrt{D_{ij}}$ and is concave otherwise. A major obstacle is how to efficiently deal with the concavity in the objective.

Secondly, by dropping the rank constraint (namely, replacing $\mathbb{K}^n_+(r)$ by $\mathbb{K}^n_+$) and through certain linear approximation to the concave term, a SDP problem is proposed for (2.6) (see Eq. (20) in (Oguz-Ekim et al., 2011)):

$$\begin{array}{lll} \min\limits_{D,T\in\mathbb{S}^n} & \langle W,\ T\rangle & \\ \text{s.t.} & (\delta_{ij} - T_{ij})^2 \le D_{ij}, & (i,j) \in \mathcal{E} \\ & a_{ij}D_{ij} + b_{ij} \le T_{ij}, & (i,j) \in \mathcal{E} \\ & -D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+, & \end{array} \tag{2.11}$$

where the quantities a_{ij} and b_{ij} can be computed from δ_{ij}, and $\mathcal{E}$ is the set of the pairs (i,j), whose dissimilarities $\delta_{ij} > 0$ are known. We note that each quadratic constraint in (2.11) is equivalent to a positive semi-definite constraint on $\mathbb{S}^2_+$ and $-D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+$ is a semi-definite constraint on $\mathbb{S}^n_+$ by (1.26). Therefore, the total number of the semi-definite constraints is $|\mathcal{E}| + 1$, resulting in a very challenging SDP even for small n.

Finally, the optimal solution of (2.11) is then refined through a second-stage algorithm, see (Oguz-Ekim et al., 2011, Sect. IV(B)). Both stages of the algorithmic scheme above would need sophisticated implementation skills and its efficiency is yet to be confirmed for many problems tested in this thesis.

2.3.3 EDM Approach

A distinguishing feature from the EDM approach is that this approach treats D as the main variable, without having to rely on its SDP representations. This approach works because of the characterization (1.26) and that the orthogonal projection onto $\mathbb{K}^n_+$ has a closed-form formula (Glunt et al., 1990; Gaffke and Mathar, 1989). Several methods are based on this formula. The basic model for this approach is

$$\min_{D\in\mathbb{S}^n} \ \|\sqrt{W}\circ(D-\Delta^{(2)})\|^2 \qquad \text{s.t.} \quad -D\in\mathbb{S}^n_h\cap\mathbb{K}^n_+(r), \tag{2.12}$$

which is the convex relaxation of (2.4) if replacing $\mathbb{K}^n_+(r)$ by $\mathbb{K}^n_+$. Here the elements of the matrices $\Delta^{(2)}$ and $\sqrt{W}$ are defined by $\Delta^{(2)}_{ij} := \delta^2_{ij}$ and $(\sqrt{W})_{ij} := W^{1/2}_{ij}$ respectively. This model is NP-hard because of the usage of rank constraint. When the special choice $W_{ij}\equiv 1$ is used, model (2.12) is the so-called nearest EDM problem. The relaxation is obtained by replacing $\mathbb{K}^n_+(r)$ by $\mathbb{K}^n_+$. Since the constraints of the nearest EDM problem are the intersection of a subspace and a convex cone, the method of alternation projection was proposed in (Glunt et al., 1990; Gaffke and Mathar, 1989) with applications to the molecule conformation (Glunt et al., 1993). A Newton's method for (2.12) was developed by Qi (2013). Extensions of Newton's method for the model (2.12) with more constraints including general weights W_{ij}, the rank constraint $\mathrm{rank}(JDJ)\le r$ or the box constraints in (3.1) can be found in (Qi and Yuan, 2014; Ding and Qi, 2017; Bai and Qi, 2016). A recent application of the model (2.12) with a regularization term to Statistics was Zhang et al. (2016), where the problem was solved by an SDP, similar to that proposed by Toh (2008). It is worth to mentioning that Ding and Qi (2017) considered the problem as

$$D^* = \mathrm{argmin}_{D\in\mathbb{S}^n_h\cap\mathbb{S}^n_+} \ \nu\|\sqrt{W}\circ(D-\Delta^{(2)})\|^2 - \sum_{i=1}^{r}\lambda_i(D) + \sum_{i=r+1}^{n}\lambda_i(D),$$

where $\nu>0$ and $\lambda_1(D)\ge\cdots\ge\lambda_n(D)$ are eigenvalues of D. The usage of term $-\sum_{i=1}^r\lambda_i(D)+\sum_{i=r+1}^n\lambda_i(D)$ aims at pursuing a low-rank solution. They studied its statistical properties and proved that under some assumptions such model could guarantee the recovered solution D^* satisfying

$$\frac{\|D^* - \overline{D}\|^2}{n^2} = \mathcal{O}\left[\frac{rn\log(2n)}{m}\right],$$

with high probability, where $\overline{D}$ is the true EDM, m is the number of known elements of Δ. This indicates that EDM approaches work for MDS problems with high potential.

There are two common features in this class of methods. One is that they require the objective function to be convex, which is true for the problems (2.3), (2.4) and (2.5) when formulated in EDM. The second feature is that the nonconexity is only caused by the rank constraint. However, the REE problem (2.6) in terms of EDM has a non-convex objective coupled with the distance d_{ij} (not squared distances) being used, that is $\sqrt{D}$ will be involved when formulated in EDM, see (2.10). This has caused all existing EDM-based methods mentioned above invalid to solve (2.6).

Some latest researches by Zhou et al. (2018a) and Zhou et al. (2018b) managed to extend the EDM approaches to the stress minimization problem (2.3) and REE problem (2.6) respectively. Once again, we emphasize that the key difference between the problems (2.6) and (2.3) is non-convex objective vs convex objective and non-differentiability vs differentiability. Hence, the problem (2.6) is significantly more difficult to solve than (2.3). Nevertheless, we will show that both can be efficiently solved by the proposed EDM optimization.

Chapter 3

Theory of EDM Optimization

This chapter first throws a general EDM model of interest in this thesis whose objective function possesses the form being capable of covering four different variants. We then establish the relationship of the model and its penalization. The latter as our main model is able to be majorized efficiently by convex functions provided that the penalty parameter is large enough. Finally, derivation of the closed form solution of majorization problem under each objective function ends up this chapter.

3.1 EDM Optimization

In this thesis, we focus on the original constrained EDM optimization,

$$\begin{aligned} \min_{D\in\mathbb{S}^n} \quad & f(D) \\ \text{s.t.} \quad & g(D)=0, \quad L\leq D\leq U, \end{aligned} \tag{3.1}$$

where $L,U\in\mathbb{S}^n\cap\mathbb{R}^n_+$ are the given lower and upper bound matrices respectively and $f(\cdot):\mathbb{S}^n\to\mathbb{R}$ is the proper, closed and continuous function and $g:\mathbb{S}^n\to\mathbb{R}$ is defined as (1.32), namely,

$$g(D)=(1/2)\|D+\Pi_{\mathbb{K}^n_+(r)}(-D)\|^2. \tag{3.2}$$

Clearly, $g(D)$ measures the violation of the feasibility of a matrix D being to an EDM with embedding dimension r. Hereafter, we write $D\geq L$ to stand for D being no less than L elementwisely, that is $D_{ij}\geq L_{ij}$ for all i,j.

3.1.1 Objective Functions

Here f can be regarded as the loss function that measures the gap between the given dissimilarity matrix $\Delta^{(2)}$ and the estimated distance matrix D. We particularly interest in the following form,

$$f(D) := f_{p,q}(D) := \left\| W^{(1/q)} \circ \left(D^{(p/2)} - \Delta^{(p)} \right) \right\|_q^q, \tag{3.3}$$

where $W \in \mathbb{S}^n \cap \mathbb{R}^n_+$ is the given weighted matrix, $p, q = 1, 2$, $\|X\|_q^q = \sum_{ij} |X_{ij}|^q$. For notational convenience, we hereafter write

$$\begin{aligned} f &:= f_{pq} := f_{p,q}(D), \\ \sqrt{W} &:= W^{(1/2)}, \qquad W := W^{(1)}, \\ \|\cdot\|^2 &:= \|\cdot\|_2^2, \qquad \|\cdot\|_1 := \|\cdot\|_1^1, \end{aligned} \tag{3.4}$$

and similar rules are applied into D and Δ. Based on those notation, we have

$$f_{22} = \|\sqrt{W} \circ (D - \Delta^{(2)})\|^2 = \sum_{ij} W_{ij}(D_{ij} - \delta_{ij}^2)^2, \tag{3.5}$$

$$f_{21} = \|W \circ (D - \Delta^{(2)})\|_1 = \sum_{ij} W_{ij}|D_{ij} - \delta_{ij}^2| \tag{3.6}$$

$$f_{12} = \|\sqrt{W} \circ (\sqrt{D} - \Delta)\|^2 = \sum_{ij} W_{ij}(\sqrt{D_{ij}} - \delta_{ij})^2 \tag{3.7}$$

$$f_{11} = \|W \circ (\sqrt{D} - \Delta)\|_1 = \sum_{ij} W_{ij}|\sqrt{D_{ij}} - \delta_{ij}| \tag{3.8}$$

We now summarize the properties possessed by those f_{pq} in the following table, in which continuous, differentiable, gradient and Lipschtiz are abbreviated to cont., diff., grad. and Lip. respectively.

Table 3.1: Properties of four objective functions

f_{pq}	Convexity	Differentiability	Gradient Lipschtiz
f_{22}	Convex	Twice cont. diff.	Lip. & grad. Lip.
f_{21}	Convex	Cont. non-diff.	Lip. & non-grad. Lip.
f_{12}	Convex	Cont. non-diff.	Non-Lip. & non-grad. Lip.
f_{11}	Nonconvex	Cont. non-diff.	Non-Lip. & non-grad. Lip.

Remark 3.1. *Let us briefly explain some parts in above table.*

- f_{22} *is Lipschtiz continuous if* U *is bounded. In fact,*

$$\begin{aligned} |f_{22}(X) - f_{22}(Y)| &= \Big| \|\sqrt{W} \circ (X - \Delta^{(2)})\|^2 - \|\sqrt{W} \circ (Y - \Delta^{(2)})\|^2 \Big| \\ &\leq \|\sqrt{W} \circ (X - Y)\| \cdot \|\sqrt{W} \circ (X + Y - 2\Delta^{(2)})\| \\ &\leq 2 \max\ W_{ij}(\|U\| + \|\Delta^{(2)}\|)\|X - Y\|. \end{aligned}$$

  It is gradient Lipschtiz continuous because of

$$\begin{aligned} |\nabla f_{22}(X) - \nabla f_{22}(Y)| &= \|2W \circ (X - \Delta^{(2)}) - 2W \circ (Y - \Delta^{(2)})\| \\ &\leq 2 \max\ W_{ij}\|X - Y\|. \end{aligned}$$

- f_{21} *is non-differentiable at* $D = \Delta^{(2)}$, *but Lipschtiz continuous due to*

$$\begin{aligned} |f_{21}(X) - f_{21}(Y)| &= \Big| \|W \circ (X - \Delta^{(2)})\|_1 - \|W \circ (Y - \Delta^{(2)})\|_1 \Big| \\ &\leq \|W \circ (X - \Delta^{(2)}) - W \circ (Y - \Delta^{(2)})\|_1 \\ &= \|W \circ (X - Y)\|_1 \leq n \max\ W_{ij}\|X - Y\|_2. \end{aligned}$$

- f_{12} *is non-differentiable at* $D = \boldsymbol{0}$ *but convex on* $D \geq \boldsymbol{0}$ *due to*

$$f_{12} = \langle W, D\rangle - 2\langle W \circ \Delta, \sqrt{D}\rangle + \|\sqrt{W} \circ \Delta\|^2,$$

  It is non-Lipschtiz and non-gradient Lipschtiz continuous because of $\sqrt{D}$.

- f_{11} *is non-convex since* $|\sqrt{D_{ij}} - \delta_{ij}|$ *is concave when* $D_{ij} > \delta_{ij}^2$ *and convex if* $0 \leq D_{ij} \leq \delta_{ij}^2$. *It is also non-differentiable at* $D = \Delta^{(2)}$ *and* $D = \boldsymbol{0}$, *and non-Lipschtiz and non-gradient Lipschtiz continuous because of* $\sqrt{D}$.

Overall, one can discern those four objective functions from f_{22} *to* f_{11} *make the problem (3.1) more and more of difficulty. The most challenging one stems from* f_{11}. *According to the below stated relations among* f_{pq} *and stress-based minimizations in Section 2.2, the difficulty somewhat explains that why most existing viable methods have been proposed to deal with problems under the first three objective functions* f_{22}, f_{21} *ans* f_{12}, *and why few efficient methods were succeeded in processing REE problem.*

3.1.2 Relations among f_{pq} and Stress-based Minimizations

Since $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$, f_{pq} corresponds to stress-based minimizations in Section 2.2.

$$f_{22} \ \text{ coincides with } \ (2.4), \tag{3.9}$$

$$f_{21} \ \text{ coincides with } \ (2.3), \tag{3.10}$$

$$f_{12} \ \text{ coincides with } \ (2.5), \tag{3.11}$$

$$f_{11} \ \text{ coincides with } \ (2.6). \tag{3.12}$$

3.1.3 Generality of Constraints

Now we briefly explain that the constraints of proposed model (3.1) enable us to deal with a wide range of scenarios. In fact, it is obvious that

$$\begin{aligned} g(D) \quad &= \quad (1/2)\|D + \Pi_{\mathbb{K}^n_+(r)}(-D)\|^2 = 0 \\ &\Longleftrightarrow \quad D + \Pi_{\mathbb{K}^n_+(r)}(-D) = 0 \\ &\Longleftrightarrow \quad -D \in \mathbb{K}^n_+(r). \end{aligned} \tag{3.13}$$

Moreover, the box region $L \le D \le U$ is capable of covering several cases: $D \in \mathbb{S}^n_h$ or other linear equalities and inequalities. In fact, for any $L, U \in \mathbb{S}^n$, we always set

$$L_{ii} = U_{ii} = 0, \ i = 1, \ldots, n \quad \Longrightarrow \quad D \in \mathbb{S}^n_h. \tag{3.14}$$

If we set $L_{ij} = U_{ij}$ for some $(i, j) \in N$, then linear equalities constraints can be assured,

$$L \le D \le U \quad \Rightarrow \quad D_{ij} = L_{ij}, \ (i, j) \in N.$$

More constraints can be found in Chapter 5.

3.2 Penalization and Majorization

Let us take a close look at the constraints in model (3.1). The constraint $L \le D \le U$ is as simple as we can wish for. The difficult part is the nonlinear equation defined by $g(D)$, which measures the violation of the feasibility of a matrix D belonging to $-\mathbb{K}^n_+(r)$.

Previous studies tend to force D to be at least Euclidean (i.e, $-D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+$ by (1.26)), which often incurs heavy computational cost. On the other hand, it has long been known that cMDS works very well as long as the dissimilarity matrix is close to be Euclidean. This means that small violation of being Euclidean would not cause a major concern for the final embedding. To address difficulties stemmed from $g(D)$, we first shift it to the objective function as a penalized term. Then in order to let its computation tractable, we construct a majorization to approximate the penalty function.

3.2.1 Penalization — Main Model

We propose to penalize the function $g(D)$ to get the following optimization problem:

$$\begin{array}{rl} \min\limits_{D \in \mathbb{S}^n} & F_\rho(D) := f(D) + \rho g(D), \\ \text{s.t.} & L \le D \le U, \end{array} \tag{3.15}$$

where $\rho > 0$ is a penalty parameter. We will carry out our research based on model (3.15) in this thesis.

We note that the classical results on penalty methods (Nocedal and Wright, 2006) for the differentiable case (i.e., all functions involved are differentiable) are not applicable for some f_{pg} and g here. Our investigation on the penalty problem (3.15) is concerned on the quality of its optimal solution when the penalty parameter is large enough. Denote D^* and D_ρ the optimal solutions of (3.1) and (3.15) respectively. We first introduce the concept of ϵ-optimality.

Definition 3.2. *(ϵ-optimal solution) For a given error tolerance $\epsilon > 0$, a point D_ϵ is called an ϵ-optimal solution of (3.1) if it satisfies*

$$L \le D_\epsilon \le U, \quad g(D_\epsilon) \le \epsilon \quad \textit{and} \quad f(D_\epsilon) \le f(D^*).$$

Obviously, if $\epsilon = 0$, D_ϵ would be an optimal solution of (3.1). We will show that the optimal solution of (3.15) is ϵ-optimal provided that ρ is large enough. The following theorem is to establish the relation between (3.1) and (3.15), and also illustrate how changing of ρ would effect the solution of (3.15).

Theorem 3.3. *Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $(-JD_\rho J)$ and*

$$\overline{\lambda} := \max_{i=r+1,\ldots,n} |\lambda_i| = \max\{|\lambda_{r+1}|, |\lambda_n|\}.$$

For any given $\epsilon > 0$ if choose

$$\rho \geq f(D^*)/\epsilon,$$

then following results hold:

$$\overline{\lambda}^2 \leq 2\epsilon, \qquad f(D_\rho) \leq \left[1 - \frac{\overline{\lambda}^2}{2\epsilon}\right] f(D^*), \qquad g(D_\rho) \leq \min\left\{\frac{f(D^*)}{\rho}, \frac{n\overline{\lambda}^2}{2}\right\} \leq \epsilon.$$

Proof Firstly, it is easy to see that

$$\begin{aligned} g(D_\rho) &= (1/2)\|D_\rho + \Pi_{\mathbb{K}^n_+(r)}(-D_\rho)\|^2 \\ &\overset{(1.31)}{=} (1/2)\|JD_\rho J + \mathrm{PCA}^+_r(-JD_\rho J)\|^2 \\ &= (1/2)\sum_{i=1}^{r}(\lambda_i - \max\{\lambda_i, 0\})^2 + (1/2)\sum_{i=r+1}^{n}\lambda_i^2 \\ &\in \left[(1/2)\overline{\lambda}^2,\ (n/2)\overline{\lambda}^2\right]. \end{aligned} \tag{3.16}$$

where the last inequality is because of $\lambda_i^2 \leq \overline{\lambda}^2$ for any $i = r+1, \ldots, n$ and the fact that $\lambda_i \geq \lambda_{r+1}, \forall\ i = 1, \ldots, r$ suffices to

$$(\lambda_i - \max\{\lambda_i, 0\})^2 \leq \lambda_{r+1}^2 \leq \overline{\lambda}^2.$$

In fact, if $\lambda_i \geq 0$, $(\lambda_i - \max\{\lambda_i, 0\})^2 = 0 \leq \lambda_{r+1}^2$. If $\lambda_{r+1} \leq \lambda_i < 0$, then $(\lambda_i - \max\{\lambda_i, 0\})^2 = \lambda_i^2 \leq \lambda_{r+1}^2$. Moreover, D^* being the optimal solution of (3.15) yields $L \leq D^* \leq U$ and $g(D^*) = 0$. Overall, we have two conclusions:

$$\rho g(D_\rho) \leq f(D_\rho) + \rho g(D_\rho) \leq f(D^*) + \rho g(D^*) = f(D^*), \tag{3.17}$$

where the second inequality is due to D_ρ and D^* being the optimal and feasible solutions of (3.15) respectively, and

$$\rho g(D_\rho) \overset{(3.16)}{\geq} \rho\overline{\lambda}^2/2 \geq \overline{\lambda}^2 f(D^*)/(2\epsilon). \tag{3.18}$$

If $f(D^*) = 0$, then (3.17) results in $\overline{\lambda} = f(D_\rho) = g(D_\rho) = 0$ and thus conclusions hold immediately. Now consider $f(D^*) > 0$. Clearly, $\overline{\lambda}^2 \le 2\epsilon$ is a direct result of (3.18) and (3.17). Finally,

$$\begin{aligned} f(D_\rho) &\overset{(3.17)}{\le} f(D^*) - \rho g(D_\rho) \overset{(3.18)}{\le} \left[1 - \frac{\overline{\lambda}^2}{2\epsilon}\right] f(D^*) \\ g(D_\rho) &\overset{(3.16,3.17)}{\le} \min\left\{\frac{f(D^*)}{\rho}, \frac{n\overline{\lambda}^2}{2}\right\} \le \min\{\epsilon,\ n\epsilon\} = \epsilon. \end{aligned}$$

where the last inequality is due to $\rho \ge f(D^*)/\epsilon$ and $\overline{\lambda}^2 \le 2\epsilon$. The proof is finished. $\square$

Remark 3.4. *Regarding to Theorem 3.3, we have some comments.*

- *From Definition 3.2, the optimal solution D_ρ of (3.15) is an ϵ-optimal solution of the original problem (3.1) if we choose $\rho \ge f(D^*)/\epsilon$.*
- *If $f(D^*) = 0$ then $f(D_\rho) = g(D_\rho) = 0 = f(D^*)$, which means D_ρ solves (3.1) and D^* solves (3.15). Such case happens, for example, when no noise contaminates Δ, namely, $\delta_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ in (1.1);*
- *If $\overline{\lambda} = 0$ being equivalent to $g(D_\rho) = 0$ by (3.16), then $f(D_\rho) \le f(D^*)$. This together with $f(D^*) \le f(D)$ for any D such that $g(D) = 0, L \le D \le U$ yields $f(D_\rho) = f(D^*)$, which indicates D_ρ solves (3.1) and D^* solves (3.15); An extreme condition for such case is to set $\rho = +\infty$ and let $\epsilon = 0$.*
- *Clearly, $g(D_\rho) \le \epsilon$ means D_ρ is very close to $\mathbb{K}^n_+(r)$ when ϵ is sufficiently small (i.e., ρ is chosen sufficiently large). In other words, Theorem 3.3 enables us to control how far of D_ρ is from $\mathbb{K}^n_+(r)$.*
- *Since $f(D^*)$ is unknown and $f(D) \ge f(D^*)$ holds for any feasible solution D of problem (3.1), we could choose $\rho \ge f(D)/\epsilon$ to meet the condition of Theorem 3.3. For example, if $L = 0$, we can simply choose $D = \boldsymbol{0}$, namely $\rho \ge f(\boldsymbol{0})/\epsilon$.*

Theorem 3.3 states that a global solution of the penalized problem is also an ϵ-optimal one of the original problem provided that ρ is large enough. Theoretically, any sufficiently large ρ is fine, which means there is no upper bound for such ρ. However, when it come to the numerical computation, too large ρ would degrade the performance of proposed method since the heavy penalization on g might lead to large f, which is clearly not promising for preserving the local distances (namely, making $f(D)$ small).

The local version of this result is related to the so-called ϵ-approximate KKT point. Before introducing its definition, we need the Lagrangian function of (3.1) which is given by,

$$L(D,\ \beta) := f(D) + \beta g(D), \qquad \forall\ D \in \mathbb{S}^n,$$

where $\beta \in \mathbb{R}$ is the Lagrangian multiplier. Then a first order optimality condition of (3.1) is that there is $\overline{D} \in \mathbb{S}^n, \bar{\beta} \in \mathbb{R}$ and $\overline{\Xi} \in \partial_D L(\overline{D}, \bar{\beta})$ such that

$$\bar{\beta} > 0, \quad g(\overline{D}) = 0, \quad \langle \overline{\Xi},\ D - \overline{D}\rangle \geq 0,\ \forall\ L \leq D \leq U. \tag{3.19}$$

This condition is similar to the KKT system of (3.1), based on which we define an ϵ-approximate KKT point as below.

Definition 3.5. *(ϵ-approximate KKT point) For a given $\epsilon > 0$, we say $\overline{D} \in \mathbb{S}^n$ is an ϵ-approximate KKT point of (3.1) if there exists $\bar{\beta} \in \mathbb{R}$ and $\overline{\Xi} \in \partial_D L(\overline{D}, \bar{\beta})$ such that*

$$\bar{\beta} > 0, \quad g(\overline{D}) \leq \epsilon, \quad \langle \overline{\Xi},\ D - \overline{D}\rangle \geq 0,\ \forall\ L \leq D \leq U. \tag{3.20}$$

$\boxed{\text{Two crucial difficulties.}}$ Now let us take a look at two crucial difficulties that we are confronted with regarding to the penalization model (3.15):

- Nonconvexity of $g(D) = (1/2)\|D + \Pi_{\mathbb{K}^n_+(r)}(-D)\|^2$, resulting the non-convexity of the objective function in (3.15), despite the computation of $\Pi_{\mathbb{K}^n_+(r)}(-D)$ is efficient owing to (1.31) if D is provided; However, when treating D as unknown variable, $g(D)$ is very hard to process to the best of our knowledge.

- Some obstructions of the computation stemmed from f due to such as non-differentiability of f_{12} or non-differentiability and non-convexity of f_{11}. Especially, when $f = f_{11}$, the usage of ℓ_1 norm and the square root (i.e., $|\sqrt{D_{ij}} - \delta_{ij}|$) leads to a complicated theoretical analysis and challenging computation.

To eliminate the above mentioned difficulties, we will take advantage of the majorization scheme (seen Subsection 1.3.5) to get rid of the non-convexity of $g(D)$. In order to unravel the obstructions from f, we will make use of the famous roots of depressed cubic equations (seen Subsection 1.3.7). The following contents in this chapter are ready to achieve these targets.

3.2.2 Majorization

For a given $Z \in \mathbb{S}^n$, Proposition 1.5 (ii) suffices to

$$\begin{aligned} F_M(D,Z) &:= f(D) + \rho g_M(D,Z) \\ &= f(D) + (\rho/2)\|D + \Pi_{\mathbb{K}^n_+(r)}(-Z)\|^2 \\ &\geq f(D) + (\rho/2)\|D + \Pi_{\mathbb{K}^n_+(r)}(-D)\|^2 \\ &= f(D) + \rho g(D). \end{aligned}$$

This means $F_M(D,Z)$ is a majorization of $f(D) + \rho g(D)$ according to (1.15). Now for a given $Z \in \mathbb{S}^n$ and denoting $Z_K := -\Pi_{\mathbb{K}^n_+(r)}(-Z)$, let us consider the following model

$$\begin{aligned} \min_{D \in \mathbb{S}^n} \quad & F_M(D,Z) = f(D) + (\rho/2)\|D - Z_K\|^2, \\ \text{s.t.} \quad & L \leq D \leq U, \end{aligned} \tag{3.21}$$

Thanks to perfect separable property of $F_M(D,Z)$, the above optimization can be reduced to a number of one dimensional problems:

$$\begin{aligned} D^*_{ij} = \text{argmin}_{D_{ij} \in \mathbb{R}} \quad & W_{ij}\left|D_{ij}^{p/2} - \Delta_{ij}^p\right|^q + \frac{\rho}{2}\Big[D_{ij} - (Z_K)_{ij}\Big]^2, \\ \text{s.t.} \quad & L_{ij} \leq D_{ij} \leq U_{ij}. \end{aligned} \tag{3.22}$$

Clearly, if $W_{ij} = 0$, its optimal solution is

$$D^*_{ij} = \Pi_{[L_{ij}, U_{ij}]}\Big((Z_K)_{ij}\Big), \tag{3.23}$$

and the complicated cases are some (i,j) such that $W_{ij} > 0$. Fortunately, (3.22) has a closed form solution when $W_{ij} > 0$ for any f_{pq}, which will be studied in the next section. And this actually eliminates the second mentioned difficulty.

3.3 Derivation of Closed Form Solutions

In this section, we propose to solve (3.22) for each pair of p, q corresponding to four objective functions: f_{22}, f_{21}, f_{12} and f_{11}. Let start with concentrating on the following general one dimensional program,

$$\begin{aligned} \min \quad & (\rho/2)(x-z)^2 + w|x^{p/2} - \delta^p|^q, \\ \text{s.t.} \quad & a \le \ x \ \le b. \end{aligned} \tag{3.24}$$

where $p, q = 1, 2$, $b \ge a \ge 0$, $\rho > 0, \delta > 0, w > 0$ and $z \in \mathbb{R}$. If $\rho = 0$, one can verify the optimal solution of above program is $\Pi_{[a,b]}(\delta^2)$, a trivial case. That is one of reasons for us only to focus on $\rho > 0$. Now we are ready to solve the model for each pair of p, q one by one. And you will see that all derivations of closed form solutions are quite straightforward, but with different degree of complication.

3.3.1 Solution under f_{22}

When $p = q = 2$, (3.24) is specified as

$$\begin{aligned} \widehat{x} &= \operatorname*{arg\,min}_{a \le x \le b} \ (\rho/2)(x-z)^2 + w(x-\delta^2)^2 \\ &= \operatorname*{arg\,min}_{a \le x \le b} \ \left[x - \frac{\rho z + 2w\delta^2}{\rho + 2w} \right]^2 = \Pi_{[a,b]} \left[\frac{\rho z + 2w\delta^2}{\rho + 2w} \right]. \end{aligned} \tag{3.25}$$

3.3.2 Solution under f_{21}

When $p = 2, q = 1$, (3.24) is specified as

$$\widehat{x} = \operatorname*{arg\,min}_{a \le x \le b} \ (\rho/2)(x-z)^2 + w|x - \delta^2| \tag{3.26}$$

Let $y = x - \delta^2$. The above problem is equivalent to

$$\widehat{y} = \operatorname*{arg\,min}_{a-\delta^2 \le y \le b-\delta^2} \ (\rho/2)(y + \delta^2 - z)^2 + w|z| \tag{3.27}$$

$$= \Pi_{[a-\delta^2, b-\delta^2]} \Big(\operatorname{sign}(z-\delta^2) \max\{|z-\delta^2| - w/\rho, 0\} \Big), \tag{3.28}$$

where $\operatorname{sign}(x)$ is the sign of x defined by

$$\operatorname{sign}(x) \begin{cases} = 1, & \text{if} \quad x > 0, \\ \in [-1, 1], & \text{if} \quad x = 0, \\ = -1, & \text{if} \quad x < 0. \end{cases} \tag{3.29}$$

Minimizing the objective function of problem (3.27) without box constraints is actually the so-called soft thresholding mapping (Donoho, 1995) which has a closed form, i.e.,

$$\mathrm{sign}(\tau)\max\{|\tau|-\lambda,0\} = \mathrm{argmin}_t\ (1/2)(t-\tau)^2+\lambda|t|.$$

Then the convexity of the objective function yields solution (3.28). Overall,

$$\begin{aligned}\widehat{x}=\widehat{y}+\delta^2 &= \Pi_{[a-\delta^2,b-\delta^2]}\Big(\ \mathrm{sign}(z-\delta^2)\max\{\big|z-\delta^2\big|-w/\rho,0\}\ \Big)+\delta^2\\ &= \Pi_{[a,b]}\Big(\ \delta^2+\mathrm{sign}(z-\delta^2)\max\{\big|z-\delta^2\big|-w/\rho,0\}\ \Big). \qquad (3.30)\end{aligned}$$

3.3.3 Solution under f_{12}

When $p=1, q=2$, (3.24) is specified as

$$\begin{aligned}\widehat{x} &= \mathop{\arg\min}_{a\le x\le b}\ (\rho/2)(x-z)^2+w(\sqrt{x}-\delta)^2\\ &= \mathop{\arg\min}_{a\le x\le b}\ (\rho/2)(x-z+w/\rho)^2-2w\delta\sqrt{x}\\ &= \mathsf{dcrn}_{[a,b]}\Big[z-w/\rho,\ 2w\delta/\rho\Big], \qquad (3.31)\end{aligned}$$

where dcrn is to solve the following one-dimensional optimization problem

$$\mathsf{dcrn}_{[a,b]}[\omega,\alpha] := \mathop{\arg\min}_{a\le x\le b} q(x) := (1/2)(x-\omega)^2-\alpha\sqrt{x}. \qquad (3.32)$$

with $0\le a\le b$, $\alpha>0$ and $\omega\in\mathbb{R}$. Before addressing the above problem, we define

$$\begin{aligned} u &:= \alpha/4, \qquad v:=\omega/3, & (3.33)\\ \tau &:= u^2-v^3 & (3.34)\\ \gamma_{\omega,\alpha} &:= (\ \omega+\sqrt{\omega^2+2\alpha}\ )^2/4, & (3.35)\end{aligned}$$

for given $\alpha>0$ and $\omega\in\mathbb{R}$. Clearly, $\gamma_{\omega,\alpha}$ is increasing on $\omega\in\mathbb{R}$ since

$$\frac{\partial\gamma_{\omega,\alpha}}{\partial\omega}=\frac{(\ \omega+\sqrt{\omega^2+2\alpha}\ )^2}{2\sqrt{\omega^2+2\alpha}}>0,$$

which means if there exists $c>-\infty$ such that $\omega\ge c$, then

$$\gamma_{\omega,\alpha}\ge\gamma_{c,\alpha}>0. \qquad (3.36)$$

First, let us consider a simple version of (3.32) with non-negative constraint, namely setting $a = 0$ and $b = +\infty$.

Proposition 3.6. *Let $\alpha > 0$ and $\omega \in \mathbb{R}$ be given. Let*

$$x^-_{\omega,\alpha} \quad := \quad \arg\min_{x \geq 0} \, q(x) := (1/2)(x-\omega)^2 - \alpha\sqrt{x}. \tag{3.37}$$

Then the solution $x^-_{\omega,\alpha} > 0$ is unique with following closed form,

$$x^-_{\omega,\alpha} = \begin{cases} \left[(u+\sqrt{\tau})^{1/3} + (u-\sqrt{\tau})^{1/3}\right]^2, & \tau \geq 0, \\ \\ 4v\cos^2\left[\frac{1}{3}\arccos(uv^{-3/2})\right], & \tau < 0. \end{cases} \tag{3.38}$$

Proof For notational convenience, we denote $x^- := x^-_{\omega,\alpha}$. For $x > 0$, the objective function $q(x)$ in (3.37) is differentiable and the first and second derivatives are

$$q'(x) = x - \omega - \frac{\alpha}{2\sqrt{x}} \quad \text{and} \quad q''(x) = 1 + \frac{\alpha}{4}x^{-3/2}.$$

It follows that $q'(x) < 0$ when $x > 0$ is close to 0 and $q''(x) \geq 1$ for all $x > 0$. Hence, $q(x)$ is decreasing near 0 and it is strongly convex on the half line $(0, +\infty)$. Therefore, the problem (3.37) has a unique solution and $x^- > 0$. Moreover,

$$q'(x^-) = x^- - \omega - \frac{\alpha}{2\sqrt{x^-}} = 0. \tag{3.39}$$

Introducing $y := \sqrt{x^-}$, we get

$$y^3 - \omega y - \alpha/2 = 0, \tag{3.40}$$

which is known as the depressed cubic equation and has three roots (in the complex planes). However, we need to find the positive real root. Recall Lemma 1.2 in which set $\nu = -\alpha/2 = -2u$. we have following results.

If $\tau > 0$ coinciding with Lemma 1.2 (i), then the positive root of (3.40) is given by

$$y = (u+\sqrt{\tau})^{1/3} + (u-\sqrt{\tau})^{1/3}. \tag{3.41}$$

and hence $x^- = y^2$ gives the solution in Case (i).

If $\tau = 0$ coinciding with Lemma 1.2 (iii), then $\nu = -2u < 0$ implies the positive root of (3.40) is given by $y = -3\nu/\omega = 2u/v$ (the other two are negative). It is easy to verify that $y = 2u/v$ is same as (3.41) when $\tau = 0$, which also gives the solution in Case (i).

If $\tau < 0$ coinciding with Lemma 1.2 (ii), then $v^3 > u^2$ which yields $v > 0$ (hence $\omega > 0$). The three real roots are

$$y_1 = 2\sqrt{v}\cos\left[\frac{\theta}{3}\right], \quad y_2 = 2\sqrt{v}\cos\left[\frac{4\pi+\theta}{3}\right] \text{ and } y_3 = 2\sqrt{v}\cos\left[\frac{2\pi+\theta}{3}\right],$$

where $\cos(\theta) = uv^{-3/2} > 0$. Since Lemma 1.2 (ii) and $\cos(\theta) > 0$ implying $0 < \theta < \pi/2$, it is easy to see that $y_1 > 0 > y_2 > y_3$. Hence, y_1 is the only positive solution and $x^- = y_1^2$ gives the result in Case (ii). $\square$

The above result shows that $x^-_{\omega,\alpha} > 0$ whenever $\alpha > 0$. The next result states that it can be bounded away from 0 by a constant when ω satisfies certain bound.

Proposition 3.7. *Let $\alpha > 0$ and $\omega \in \mathbb{R}$ be given with $|\omega| \leq c$, where c is a given constant. Then there exists $\gamma > 0$ such that*

$$x^-_{\omega,\alpha} \geq \gamma.$$

Proof Suppose the result is not true. Then there exists a sequence $\{\omega_k\}_{k\geq 1}$ with $|\omega_k| \leq c$ such that

$$\lim_{k\to\infty} x^-_{\omega_k,\alpha} = 0.$$

By the proof in Proposition 3.6 (see (3.39)), $x^-_{\omega_k,\alpha} > 0$ must be the solution of the following equation:

$$x^-_{\omega_k,\alpha} - \omega_k - \frac{\alpha}{2\sqrt{x^-_{\omega_k,\alpha}}} = 0.$$

Multiplying $\sqrt{x^-_{\omega_k,\alpha}}$ on the both sides of the equation above and taking limits yield

$$0 = \lim_{k\to\infty}\left[\left(x^-_{\omega_k,\alpha}\right)^{3/2} - \omega_k\sqrt{x^-_{\omega_k,\alpha}}\right] = \lim_{k\to\infty}\frac{\alpha}{2} = \frac{\alpha}{2} > 0.$$

The contradiction establishes the result claimed. $\square$

Proposition 3.6 can be readily extended to the case where the constraint is an interval rather than being non-negative.

Proposition 3.8. *Let $0 \le a \le b$, $\alpha > 0$ and $\omega \in \mathbb{R}$ be given. Let*

$$\mathsf{dcrn}_{[a,b]}[\omega, \alpha] := \arg\min_{a \le x \le b} \ q(x) = (1/2)(x-\omega)^2 - \alpha\sqrt{x}. \tag{3.42}$$

Then $\mathsf{dcrn}_{[a,b]}[\omega, \alpha]$ is the unique solution with form

$$\mathsf{dcrn}_{[a,b]}[\omega, \alpha] = \Pi_{[a,b]}(x^-_{\omega,\alpha}) = \begin{cases} a, & \omega \le a - \frac{\alpha}{2\sqrt{a}} \\ x^-_{\omega,\alpha}, & a - \frac{\alpha}{2\sqrt{a}} < \omega < b - \frac{\alpha}{2\sqrt{b}} \\ b, & \omega \ge b - \frac{\alpha}{2\sqrt{b}} \end{cases} \tag{3.43}$$

where $x^-_{\omega,\alpha}$ is given by (3.37). Moreover it holds

$$\mathsf{dcrn}_{[a,b]}[\omega, \alpha] \ge \min\left\{b, 1, \gamma_{\omega,\alpha}\right\}. \tag{3.44}$$

Proof For convenience, we write $\mathsf{dcrn} := \mathsf{dcrn}_{[a,b]}[\omega, \alpha]$ and $\kappa := x^-_{\omega,\alpha}$. The first equality in (3.43) holds because $\kappa = \arg\min_{x \ge 0} \ q(x)$ and $q(x)$ is convex due to Proposition 3.6. Now we prove the second equality in (3.43). Two cases are considered: Case 1) $a > 0$ and Case 2) $a = 0$,

Case 1) $b \ge a > 0$. It follows that $q'(x) = x - \omega - \frac{\alpha}{2\sqrt{x}}$, which is increasing on $a \le x \le b$ and thus $q'(a) \le q'(x) \le q'(b)$. If $\omega \le a - \frac{\alpha}{2\sqrt{a}}$, then

$$q'(x) \ge q'(a) = a - \omega - \frac{\alpha}{2\sqrt{x}} \ge 0$$

which indicates $q(x)$ is increasing on $[a, b]$ and thus $\mathsf{dcrn} = a$. Similarly, we have $\mathsf{dcrn} = b$ if $\omega \ge b - \frac{\alpha}{2\sqrt{b}}$. If $a - \frac{\alpha}{2\sqrt{a}} < \omega < b - \frac{\alpha}{2\sqrt{b}}$, then $q'(a) \le q'(x) \le q'(b)$ with $q'(a) < 0$ and $q'(b) > 0$. This implies there is a unique $x^* \in (a, b)$ such that

$$q'(x^*) = x^* - \omega - \frac{\alpha}{2\sqrt{x^*}} = 0.$$

By Propositions 3.6, it proved that κ is the unique point in $(0, +\infty)$ such that $q'(\kappa) = 0$, which indicates $x^* = \kappa$. Overall dcrn the unique optimal solution of (3.42).

Case 2) $b \ge a = 0$. Two scenarios: $b = a = 0$ and $b > a = 0$ are taken into consideration. i) $b = a = 0$. Clearly, the unique optimal solution of (3.42) is 0, which coincides $\mathsf{dcrn} = 0 = b$ since the condition $\omega > b - \frac{\alpha}{2\sqrt{b}} = -\infty$ in (3.43) always holds.

ii) If $b > a = 0$, we conclude the optimal solution of (3.42) is

$$\mathsf{dcrn}_{[a,b]}[\omega,\alpha] = \begin{cases} x^-_{\omega,\alpha}, & a - \frac{\alpha}{2\sqrt{a}} < \omega < b - \frac{\alpha}{2\sqrt{b}} \\ b, & \omega \geq b - \frac{\alpha}{2\sqrt{b}} \end{cases} \tag{3.45}$$

which exactly is (3.43) since $\omega > a - \frac{\alpha}{2\sqrt{a}} = -\infty$. In fact, if $a - \frac{\alpha}{2\sqrt{a}} < \omega < b - \frac{\alpha}{2\sqrt{b}}$, as proved above $\kappa = \arg\min_{x\in(0,b]}\ q(x)$ and $q'(\kappa) = \kappa - \omega - \frac{\alpha}{2\sqrt{\kappa}} = 0$, then

$$\begin{aligned} q(\kappa) - q(0) &= (1/2)\kappa^2 - \omega\kappa - \alpha\sqrt{\kappa} \\ &= -(1/2)\kappa^2 + \kappa q'(\kappa) - (\alpha/2)\sqrt{\kappa} \\ &= -(1/2)\kappa^2 - (\alpha/2)\sqrt{\kappa} < 0. \end{aligned}$$

Therefore, $\kappa = \arg\min_{x\in(0,b]}\ q(x) = \arg\min_{x\in[0,b]}\ q(x)$. If $\omega \geq b - \frac{\alpha}{2\sqrt{b}}$, then for any $x \in (0,b)$ it holds $0 \geq q'(b) > q'(x)$, which indicates $q(x)$ is strictly decreasing on $(0,b]$. Notice that $q(x)$ is continuous on $[0,b]$, we must have $q(b) < q(0)$ and thus $b = \arg\min_{x\in[0,b]}\ q(x)$. Overall, dcrn is the unique optimal solution of (3.42).

Finally, $q'(\kappa) = 0$ implies that $\kappa - \omega - \frac{\alpha}{2\sqrt{\kappa}} = 0$. If $\kappa \leq 1$ then $\sqrt{\kappa} - \omega - \frac{\alpha}{2\sqrt{\kappa}} \geq \kappa - \omega - \frac{\alpha}{2\sqrt{\kappa}} = 0$ which suffices to $\sqrt{\kappa} \geq (\gamma_{\omega,\alpha})^{1/2} > 0$. Thus we must have $\kappa \geq \min\{1, \gamma_{\omega,\alpha}\}$. This together with $\mathsf{dcrn} = \Pi_{[a,b]}(\kappa) = \min\{b, \max\{a,\kappa\}\} \geq \min\{b,\kappa\}$ finishes the proof. □

Remark 3.9. *Regarding to Proposition 3.8, we name the solution of (3.42) dcrn since it is related to the root of the so-called depressed cubic equation (3.40) with negative part $-\alpha$. By (3.43), solution $\mathsf{dcrn}_{[a,b]}[\omega,\alpha]$ is positive and away from zero if b is positive and ω is bounded from below.*

3.3.4 Solution under f_{11}

When $p = q = 1$, (3.24) is specified as

$$\widehat{x} \ = \ \arg\min_{a\leq x\leq b}\ (\rho/2)(x-z)^2 + w|\sqrt{x} - \delta| = \mathsf{dcr}_{[a,b]}\Big[z, w/\rho, \delta\Big], \tag{3.46}$$

where dcr is to solve the following one-dimensional optimization problem:

$$\mathsf{dcr}_{[a,b]}[\omega,\alpha,\delta] := \arg\min_{a\leq x\leq b}\ p(x) := (1/2)(x-\omega)^2 + \alpha|\sqrt{x} - \delta|. \tag{3.47}$$

with $0 \le a \le b$, $\omega \in \mathbb{R}, \delta > 0$ and $4\delta^3 > \alpha > 0$. Before solving above problem, related to the sign of $(\sqrt{x} - \delta)$, we denote

$$p_+(x) := (1/2)(x-\omega)^2 + \alpha\sqrt{x} - \alpha\delta. \tag{3.48}$$

$$p_-(x) := (1/2)(x-\omega)^2 - \alpha\sqrt{x} + \alpha\delta. \tag{3.49}$$

We first solve the problem when $p(x)$ is replaced by $p_+(x)$ in (3.47).

Proposition 3.10. *Let $0 < a \le b$, $0 < \alpha < 4\sqrt{a^3}$ and*

$$\mathsf{dcrp}_{[a,b]}[\omega, \alpha] := \underset{a \le x \le b}{\arg\min}\ p_+(x) = (1/2)(x-\omega)^2 + \alpha\sqrt{x} - \alpha\delta. \tag{3.50}$$

Then we have following results:

(i) $p_+(x)$ is strictly convex on $x > (\alpha/4)^{2/3}$.

(ii) $\mathsf{dcrp}_{[a,b]}[\omega, \alpha]$ is the unique optimal solution with form

$$\mathsf{dcrp}_{[a,b]}[\omega, \alpha] = \begin{cases} a, & \omega \le a + \frac{\alpha}{2\sqrt{a}} \\ x^+_{\omega,\alpha}, & a + \frac{\alpha}{2\sqrt{a}} < \omega < b + \frac{\alpha}{2\sqrt{b}} \\ b, & \omega \ge b + \frac{\alpha}{2\sqrt{b}} \end{cases} \tag{3.51}$$

where $x^+_{\omega,\alpha}$ is given by

$$x^+_{\omega,\alpha} = 4v\cos^2\left[\arccos(-uv^{-\frac{3}{2}})/3\right]. \tag{3.52}$$

Furthermore, $a < x^+_{\omega,\alpha} < b$ when $a + \frac{\alpha}{2\sqrt{a}} < \omega < b + \frac{\alpha}{2\sqrt{b}}$.

Proof (i) For $x > 0$, $p_+(x)$ is differentiable and the first and second derivatives are

$$p'_+(x) = x - \omega + \frac{\alpha}{2\sqrt{x}} \quad \text{and} \quad p''_+(x) = 1 - \frac{\alpha}{4\sqrt{x^3}}.$$

It is easy to verify that for any $x > u^{2/3}$, $p''_+(x) > p''_+(u^{2/3}) = 0$. Namely, $p_+(x)$ is strictly convex on $(u^{2/3}, +\infty)$. For simplicity, we write $\mathsf{dcrp}; = \mathsf{dcrp}_{[a,b]}[\omega, \alpha]$.

(ii) Since $0 < \alpha < 4\sqrt{a^3}$, it holds $a > u^{2/3}$ and $[a,b] \subseteq (u^{2/3}, +\infty)$. Then strict convexity indicates the optimal solution of (3.50) is unique. For any $x > u^{2/3}$, $p''_+(x) > 0$ means that $p'_+(x)$ is increasing on $a \le x \le b$ and thus $p'_+(a) \le p'_+(x) \le p'_+(b)$. If

$\omega \leq a + \frac{\alpha}{2\sqrt{a}}$, then $p'_+(x) \geq p'_+(a) \geq 0$ which indicates $p_+(x)$ is increasing on $[a,b]$ and thus $\mathsf{dcrp} = a$. Similarly, we have $\mathsf{dcrp} = b$ if $\omega \geq b + \frac{\alpha}{2\sqrt{b}}$. If $a + \frac{\alpha}{2\sqrt{a}} < \omega < b + \frac{\alpha}{2\sqrt{b}}$, then $p'_+(a) \leq p'_+(x) \leq p'_+(b)$ with $p'_+(a) < 0$ and $p'_+(b) > 0$. This implies there is a unique $x^+ \in (a,b)$ such that $p'_+(x^+) = 0$, namely

$$x^+ - \omega + \alpha/(2\sqrt{x^+}) = 0 \Longleftrightarrow x^+ = \operatorname*{argmin}_{a<x<b} \ (1/2)(x-\omega)^2 + \alpha\sqrt{x}. \tag{3.53}$$

By introducing $y = \sqrt{x^+}$, we obtain the depressed cubic equation

$$y^3 - \omega y + \alpha/2 = 0. \tag{3.54}$$

Recall Lemma 1.2 in which set $\nu = \alpha/2 = 2u$. We have $\tau = u^2 - v^3 < 0$ due to

$$v = \frac{\omega}{3} > \frac{1}{3}\left[a + \frac{\alpha}{2\sqrt{a}}\right] = \frac{1}{3}\left[a + \frac{\alpha}{4\sqrt{a}} + \frac{\alpha}{4\sqrt{a}}\right] \geq \left[a\frac{\alpha}{4\sqrt{a}}\frac{\alpha}{4\sqrt{a}}\right]^{1/3} = u^{\frac{2}{3}}, \tag{3.55}$$

which coincides with Lemma 1.2 (ii) that depressed cubic equation has three real roots

$$y_1 = 2\sqrt{v}\cos\left[\frac{\theta}{3}\right], \quad y_2 = 2\sqrt{v}\cos\left[\frac{4\pi+\theta}{3}\right], \quad y_3 = 2\sqrt{v}\cos\left[\frac{2\pi+\theta}{3}\right],$$

where $\cos(\theta) = -uv^{-3/2} \in (-1,0)$ since $v^{3/2} > u > 0$ implying $\pi/2 < \theta < \pi$. This and Lemma 1.2 (ii) derive $y_1 > y_2 > 0 > y_3$.

Finally, we need to decide which positive solution is what we want. From Lemma 1.1 in which take $\bar{x} = u^{2/3}$ and $\bar{\omega} = 3u^{2/3}$, we have when $\omega > \bar{\omega}$ (due to (3.55)), problem (3.53) under the constraint $x > 0$ has tow positive stationary points x_1^+ and x_2^+ such that

$$x_1^+ < \bar{x} = u^{2/3} < x_2^+$$

And one can check when $\pi/2 < \theta < \pi$ that

$$y_1^2 = 4v\cos^2(\theta/3) > v \overset{(3.55)}{>} u^{2/3}.$$

But one also need exclude y_2^2, namely need show $y_2^2 < u^{2/3}$. By $\cos(\theta) = -uv^{-3/2}$ implying $v = u^{2/3}/\cos^2(\theta)$, we rewrite

$$y_2^2 = \frac{4\cos^2(4\pi/3 + \theta/3)u^{2/3}}{\cos^2(\theta)} =: t(\theta)u^{2/3}.$$

Let $\gamma := \cos(2\pi/3 + 2\theta/3) \in (-1, -1/2)$ due to $\pi/2 < \theta < \pi$, we have

$$\begin{aligned} t(\theta) &= 4\frac{\cos^2(4\pi/3+\theta/3)}{\cos^2(\theta)} &&= 4\frac{\cos(8\pi/3+2\theta/3)+1}{\cos(2\theta)+1} \\ &= 4\frac{\cos(2\pi/3+2\theta/3)+1}{\cos(2\pi+2\theta)+1} &&= 4\frac{\gamma+1}{4\gamma^3-3\gamma+1} \\ &= \frac{\gamma+1}{(\gamma+1)(\gamma-1/2)^2} &&= \frac{1}{(\gamma-1/2)^2} \in (4/9,\ 1)\,, \end{aligned}$$

where the second and fourth equalities are respectively from facts $\cos(2\theta) = 2\cos^2(\theta) - 1$ and $\cos(3\theta) = 4\cos^3(\theta) - 3\cos(\theta)$. This proves $y_2^2 < u^{2/3}$. Therefore, we conclude that

$$x_2^+ = y_1^2 \quad \text{and} \quad x_1^+ = y_2^2$$

and thus the only solution of (3.53) is $x^+ = y_1^2$, deriving (3.52). $\square$

Remark 3.11. *Regarding to Proposition 3.10, we name the solution of (3.50)* dcrp *since it is related to the root of the depressed cubic equation (3.54) with a positive constant α. Moreover, the unique solution is away from zero provided that $a > u^{2/3} > 0$.*

Proposition 3.12. *Let $0 \le a < \delta^2 < b, 0 < \alpha < 4\delta^3, \omega \in \mathbb{R}$, and $x^-_{\omega,\alpha}, x^+_{\omega,\alpha}$ be defined by (3.38) and (3.52) respectively. Then the unique optimal solution of (3.47) is*

$$\mathsf{dcrb}_{[a,b]}[\omega,\alpha,\delta] = \begin{cases} a, & \omega \le a - \frac{\alpha}{2\sqrt{a}}, \\ x^-_{\omega,\alpha}, & a - \frac{\alpha}{2\sqrt{a}} < \omega < \delta^2 - \frac{\alpha}{2\delta}, \\ \delta^2, & \delta^2 - \frac{\alpha}{2\delta} \le \omega \le \delta^2 + \frac{\alpha}{2\delta}, \\ x^+_{\omega,\alpha}, & \delta^2 + \frac{\alpha}{2\delta} < \omega < b + \frac{\alpha}{2\sqrt{b}}, \\ b, & \omega \ge b + \frac{\alpha}{2\sqrt{b}}. \end{cases} \tag{3.56}$$

Proof We consider two cases: 1) $x \in [a, \delta^2]$ and 2) $x \in [\delta^2, b]$. For case 1), it follows

$$\begin{aligned} \mathsf{dcrn}_{[a,\delta^2]}[\omega,\alpha] &= \mathrm{argmin}_{x\in[a,\delta^2]} p(x) && (3.57) \\ &\overset{(3.49)}{=} \mathrm{argmin}_{x\in[a,\delta^2]} p_-(x) && (3.58) \\ &\overset{(3.32)}{=} \mathrm{argmin}_{x\in[a,\delta^2]} q(x) \\ &= \begin{cases} a, & \omega \le a - \frac{\alpha}{2\sqrt{a}} \\ x^-_{\omega,\alpha}, & a - \frac{\alpha}{2\sqrt{a}} < \omega < \delta^2 - \frac{\alpha}{2\delta} \\ \delta^2, & \omega \ge \delta^2 - \frac{\alpha}{2\delta} \end{cases} && (3.59) \end{aligned}$$

where the last equality is from Proposition 3.8. For case 2), it follows that

$$\begin{aligned}\mathsf{dcrp}_{[\delta^2,b]}[\omega,\alpha] &= \text{argmin}_{x\in[\delta^2,b]}p(x) \overset{(3.48)}{=} \text{argmin}_{x\in[\delta^2,b]}p_+(x)\\ &= \begin{cases}\delta^2, & \omega \le \delta^2+\frac{\alpha}{2\delta}\\ x^+_{\omega,\alpha}, & \delta^2+\frac{\alpha}{2\delta}<\omega<b+\frac{\alpha}{2\sqrt{b}}\\ b, & \omega \ge b-\frac{\alpha}{2\sqrt{b}}\end{cases}\end{aligned} \tag{3.60}$$

where the last equality is from Proposition 3.10. Now we only show the first two scenarios in (3.56) because the rest are similar. If $\omega \le a-\frac{\alpha}{2\sqrt{a}}(<\delta^2+\frac{\alpha}{2\delta})$, we have

$$a = \arg\min_{x\in[a,\delta^2]} p(x), \qquad \delta^2 = \arg\min_{x\in[\delta^2,b]} p(x),$$

which means $p(a)\le p(x)$ for any $x\in[a,\delta^2]$ and $p(a)\le p(\delta^2)\le p(x)$ for any $x\in[\delta^2,b]$. Thus $\mathsf{dcrb}_{[a,b]}[\omega,\alpha,\delta]=a$. If $a-\frac{\alpha}{2\sqrt{a}}<\omega<\delta^2-\frac{\alpha}{2\delta}(<\delta^2+\frac{\alpha}{2\delta})$, we have

$$x^-_{\omega,\alpha} = \arg\min_{x\in[a,\delta^2]} p(x), \qquad \delta^2 = \arg\min_{x\in[\delta^2,b]} p(x),$$

which means $p(x^-_{\omega,\alpha})\le p(x)$ for any $x\in[a,\delta^2]$ and $p(x^-_{\omega,\alpha})\le p(\delta^2)\le p(x)$ for any $x\in[\delta^2,b]$. Thus $\mathsf{dcrb}_{[a,b]}[\omega,\alpha,\delta]=x^-_{\omega,\alpha}$. □

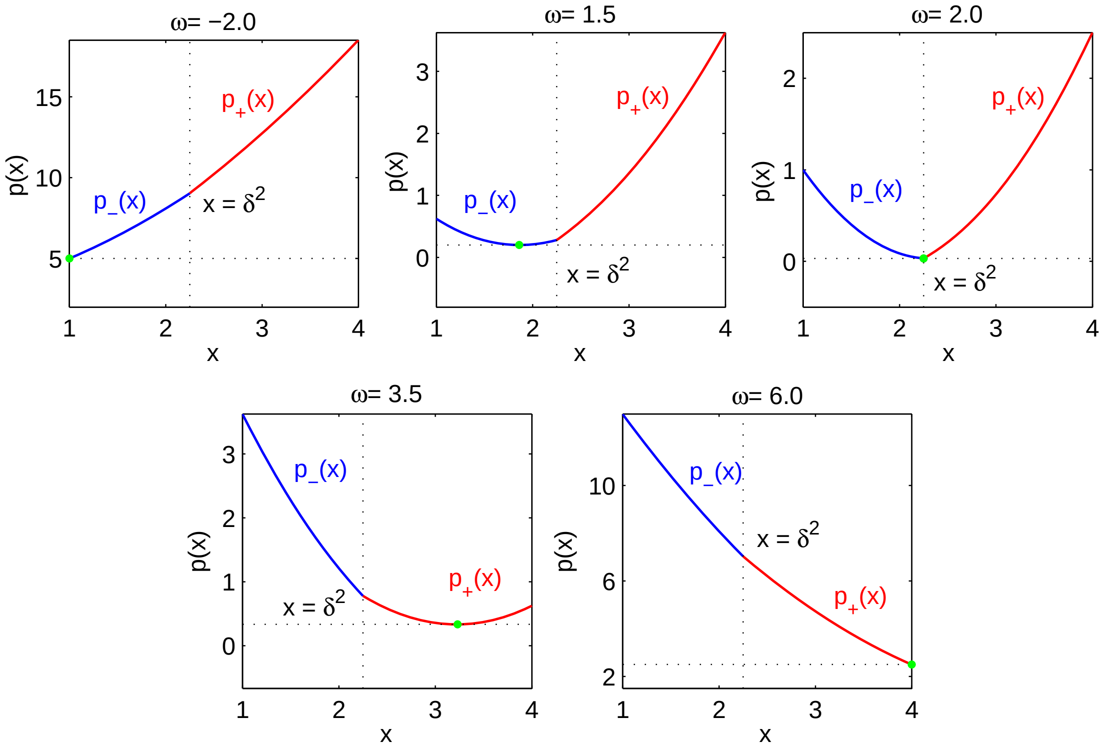


Figure 3.1: Optimal solutions of (3.61) under different cases.

Comment: The optimal solution $\mathsf{dcrb}_{[a,b]}[\omega, \alpha, \delta]$ is unique, whose location is depended on the magnitudes of the parameters (ω, α and δ) involved. Let see one simple example

$$\min_{1\leq x\leq 4} p(x) = 0.5(x-\omega)^2 + |\sqrt{x} - 1.5|. \tag{3.61}$$

with fixing $\alpha = 1, \delta = 1.5$. Its optimal solution (plotted by green dot) is illustrated in Figure 3.1, which matches (3.56) under different ω. For example, when $\omega = -2 < a - \alpha/(2\sqrt{a}) = 0.5$, it equals to $a = 1$. When $\omega = 1.5 \in (0.5, 1.917)$, it occurs on $p_-(x)$ within $[1, 2.25]$ whilst when $\omega = 3.5 \in (2.58, 4.25)$, it occurs on $p_+(x)$ within $[2.25, 4]$.

Recall $\mathrm{sign}(x)$ is the sign of x defined by (3.29). Then $|\sqrt{x} - \delta|$ is non-smooth on $x = \delta^2$ (which can be illustrated by Figure 3.1) and smooth for any $0 < x \neq \delta^2$. Its subdifferential can be calculated by

$$\partial(|\sqrt{x} - \delta|) = \left\{ \frac{\mathrm{sign}(\sqrt{x} - \delta)}{2\sqrt{x}} \right\} \quad \text{for} \quad x > 0. \tag{3.62}$$

Proposition 3.13. *Let $\delta > 0$ be given and $\phi(x) := |\sqrt{x} - \delta|$, then for any $x, y > 0$,*

$$\phi(x) - \phi(y) \leq \zeta(x-y) + \frac{(x-y)^2}{8\delta^3}, \qquad \forall\, \zeta \in \partial\phi(x). \tag{3.63}$$

Proof We prove it by considering three cases. Case 1: $0 < x < \delta^2$; Case 2: $x > \delta^2$ and Case 3: $x = \delta^2$. Let $\eta := \mathrm{sign}(\sqrt{x} - \delta)$ and $\zeta := \eta/(2\sqrt{x})$, then $\zeta \in \partial\phi(x)$.

Case 1: $0 < x < \delta^2$. For this case, $\eta = -1$. We note that $\phi(x) = \delta - \sqrt{x}$ is convex and differentiable at $0 < x < \delta^2$. Thus,

$$\phi(y) \geq \phi(x) - \frac{y-x}{2\sqrt{x}} \qquad \text{for any } 0 < y < \delta^2.$$

For $y \geq \delta^2$, we have the following chain of inequalities

$$\begin{aligned} \phi(x) - \frac{y-x}{2\sqrt{x}} \;&\leq\; \delta - \sqrt{x} - \frac{\delta^2 - x}{2\sqrt{x}} \;=\; \delta - \left[\frac{\sqrt{x}}{2} + \frac{\delta^2}{2\sqrt{x}}\right] \\ &\leq\; \delta - 2\left[\frac{\sqrt{x}}{2}\frac{\delta^2}{2\sqrt{x}}\right]^{1/2} \;=\; \delta - \delta = 0 \\ &\leq\; \sqrt{y} - \delta = \phi(y). \end{aligned}$$

Henceforth, we proved the conclusion for this case.

Case 2: $x > \delta^2$. For this case, $\eta = 1$. By defining

$$\varphi(u,v) := u(u^2 - v^2)^2 - 4\delta^3(u+v)^2 + 16u\delta^4$$

with $u > \delta$ and $0 < v < \delta$, we have

$$\frac{\partial \varphi(u,v)}{\partial v} = 2(u+v)(2uv(v-u) - 4\delta^3) \leq 0,$$

which indicates $\varphi(u,v)$ non-increasing with respect v and thus

$$\begin{aligned} \varphi(u,v) \geq \varphi(u,\delta) &= u(u^2-\delta^2)^2 - 4\delta^3(u+\delta)^2 + 16\delta^4 u \\ &= (u+\delta)^2(u(u-\delta)^2 - 4\delta^3) + 16\delta^4 u \\ &\geq (\delta+\delta)^2(\delta(\delta-\delta)^2 - 4\delta^3) + 16\delta^5 \\ &= 0. \end{aligned} \tag{3.64}$$

For $0 < y < \delta^2$, we have

$$\begin{aligned} \phi(x) - \phi(y) &= \sqrt{x} + \sqrt{y} - 2\delta \\ &= \frac{x-y}{2\sqrt{x}} + \frac{(\sqrt{x}+\sqrt{y})^2}{2\sqrt{x}} - 2\delta \\ &= \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{8\delta^3} - \left[\frac{(x-y)^2}{8\delta^3} - \frac{(\sqrt{x}+\sqrt{y})^2}{2\sqrt{x}} + 2\delta\right] \\ &= \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{8\delta^3} - \frac{\varphi(\sqrt{x},\sqrt{y})}{8\delta^3\sqrt{x}} \\ &\overset{(3.64)}{\leq} \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{8\delta^3} \end{aligned}$$

For $y \geq \delta^2$, we have the following chain of inequalities

$$\begin{aligned} \phi(x) - \phi(y) &= \sqrt{x} - \sqrt{y} = \frac{x-y}{2\sqrt{x}} + \frac{(\sqrt{x}-\sqrt{y})^2}{2\sqrt{x}} \\ &= \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{2\sqrt{x}(\sqrt{x}+\sqrt{y})^2} \\ &\leq \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{2\delta(\delta+\delta)^2} \\ &= \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{8\delta^3}. \end{aligned} \tag{3.65}$$

Hence, we proved the claim for this case.

Case 3: $x = \delta^2$. For this case, $-1 \le \eta \le 1$. Then for $0 < y < \delta^2$, we have

$$\begin{aligned} \phi(x) - \phi(y) &= \delta - \sqrt{x} - (\delta - \sqrt{y}) \\ &= \sqrt{y} - \sqrt{x} = \frac{y-x}{\sqrt{y}+\sqrt{x}} \le -\frac{x-y}{2\sqrt{x}} \le \frac{\eta(x-y)}{2\sqrt{x}}. \end{aligned}$$

where the first and last inequalities hold due to $y < \delta^2 = x$ and $|\eta| \le 1$. For $y \ge \delta^2$, similar to obtaining (3.65), it holds

$$\phi(x) - \phi(y) = \sqrt{x} - \sqrt{y} \le \frac{x-y}{2\sqrt{x}} + \frac{(x-y)^2}{8\delta^3} \le \frac{\eta(x-y)}{2\sqrt{x}} + \frac{(x-y)^2}{8\delta^3},$$

where the last inequality is due to $|\eta| \le 1$ and $x - y \le 0$, which concludes the claim of Case 3 and hence finishes the whole proof. □

Now we are ready to solve (3.47) based on propositions above.

Proposition 3.14. *Let* $0 \le a \le b$, $\omega \in \mathbb{R}$ *and* $0 < \alpha < 4\delta^3$. *Let*

$$\mathsf{dcr}_{[a,b]}[\omega, \alpha, \delta] := \arg \min_{a \le x \le b} \ p(x) := (1/2)(x-\omega)^2 + \alpha|\sqrt{x} - \delta|. \tag{3.66}$$

Then we have following results.

(i) $p(x)$ *is strictly convex on* $x > 0$.

(ii) $\mathsf{dcr}_{[a,b]}[\omega, \alpha, \delta]$ *is the unique optimal solution with*

$$\mathsf{dcr}_{[a,b]}[\omega, \alpha, \delta] = \begin{cases} \mathsf{dcrp}_{[a,b]}[\omega, \alpha], & \delta^2 \le a \\ \mathsf{dcrb}_{[a,b]}[\omega, \alpha, \delta], & a < \delta^2 < b \\ \mathsf{dcrn}_{[a,b]}[\omega, \alpha], & \delta^2 \ge b \end{cases} \tag{3.67}$$

where $\mathsf{dcrn}, \mathsf{dcrp}$ *and* dcrb *are defined by (3.43), (3.51) and (3.56) respectively.*

(iii) Let $\gamma_{\omega,\alpha}$ *be defined as (3.34), then*

$$\mathsf{dcr}_{[a,b]}[\omega, \alpha, \delta] \ge \min\{\delta^2, b, 1, \gamma_{\omega,\alpha}\};$$

Furthermore if $b > 0$ *and* ω *is bounded from below, then* $\min\{\delta^2, b, 1, \gamma_{\omega,\alpha}\} > 0$ *and there exists* $\zeta \in \partial p(\mathsf{dcr}_{[a,b]}[\omega, \alpha, \delta])$ *such that*

$$\zeta(x - \mathsf{dcr}_{[a,b]}[\omega, \alpha, \delta]) \ge 0 \quad \textit{for any } x \in B.$$

Proof (i) For any $x, y > 0$, it holds

$$\begin{aligned}
p(y) - p(x) &= \frac{1}{2}(y-\omega)^2 - \frac{1}{2}(x-\omega)^2 + \alpha(|\sqrt{y}-\delta| - |\sqrt{x}-\delta|) \\
&= (x-\omega)(y-x) + \frac{1}{2}(x-y)^2 + \alpha(|\sqrt{y}-\delta| - |\sqrt{x}-\delta|) \\
&\overset{(3.63)}{\geq} (x-\omega)(y-x) + \frac{1}{2}(x-y)^2 - \alpha\zeta_x(x-y) - \frac{\alpha}{8\delta^3}(x-y)^2 \\
&= [x-\omega+\alpha\zeta_x](y-x) + \frac{4\delta^3-\alpha}{8\delta^3}(x-y)^2,
\end{aligned}$$

for any $\zeta_x \in \partial\phi(x)$. Similarly, it holds

$$p(x) - p(y) \geq [y-\omega+\alpha\zeta_y](x-y) + \frac{4\delta^3-\alpha}{8\delta^3}(x-y)^2,$$

for any $\zeta_y \in \partial\phi(y)$. Adding above two equalities yields that

$$[(x-\omega+\alpha\zeta_x) - (y-\omega+\alpha\zeta_y)](x-y) \geq \frac{4\delta^3-\alpha}{4\delta^3}(x-y)^2.$$

Since $(x-\omega+\alpha\zeta_x) \in \partial p(x)$ and $(y-\omega+\alpha\zeta_y) \in \partial p(y)$, we conclude that $p(x)$ is strictly convex on $x > 0$ by $4\delta^3 > \alpha > 0$ and (Rockafellar and Wets, 2009, Theorem 12.17).

(ii) For convenience, denote $\mathsf{dcr} := \mathsf{dcr}_{[a,b]}[\omega,\alpha,\delta]$. If $\delta^2 \leq a$, then $0 < \alpha < 4\delta^3 \leq 4\sqrt{a^3}$ and $p(x) = p_+(x)$ for any $a \leq x \leq b$, which combining Proposition 3.10 derives $\mathsf{dcr} = \mathsf{dcrp}_{[a,b]}[\omega,\alpha]$. If $\delta^2 \geq b$, then Proposition 3.8 derives $\mathsf{dcr} = \mathsf{dcrn}_{[a,b]}[\omega,\alpha]$. If $a < \delta^2 < b$, it follows from Proposition 3.12 that $\mathsf{dcr} = \mathsf{dcrb}_{[a,b]}[\omega,\alpha,\delta]$.

(iii) If $\delta^2 \leq a$, $\mathsf{dcr} = \mathsf{dcrp}_{[a,b]}[\omega,\alpha] \geq a \geq \delta^2$ from Proposition 3.10 (ii); If $\delta^2 \geq b$, $\mathsf{dcr} = \mathsf{dcrn}_{[a,b]}[\omega,\alpha] \geq \min\{b, 1, \gamma_{\omega,\alpha}\}$ by Proposition 3.8 (iii); If $a < \delta^2 < b$, from the proof of Proposition 3.12 and (3.57 - 3.60), we have $\mathsf{dcr} = \mathsf{dcrb}_{[a,b]}[\omega,\alpha,\delta] \geq x^-_{[a,\delta^2]}[\omega,\alpha] \geq \min\{\delta^2, 1, \gamma_{\omega,\alpha}\}$ by Proposition 3.8 (iii). Overall, $\mathsf{dcr} \geq \min\{\delta^2, b, 1, \gamma_{\omega,\alpha}\}$. If ω is bounded from below, then $\gamma_{\omega,\alpha} > 0$ through (3.36). Further assume $b > 0$, then $\mathsf{dcr} \geq \min\{\delta^2, b, 1, \gamma_{\omega,\alpha}\} > 0$ due to $0 < \alpha < 4\delta^3$, which means $\partial p(\mathsf{dcr})$ is well defined. Finally, the optimality condition of a strictly convex function yields the last claim. □

Chapter 4

Majorization-Projection Method

This chapter centres on the algorithm to solve the proposed penalty model (3.15). We have already eliminated two major difficulties mentioned in Subsection 3.2.1 by using the ideas of majorization of g and closed form solution under each f_{pq}, seen Subsection 3.2.2 and Section 3.3. This naturally leads to the well known majorization minimization method which along with projection onto a box constraint results in our interested Majorization-Projection method dubbed as `MPEMD`, an abbreviation for Majorization-Projection method via EDM optimization.

The organization of this chapter is as follows. We first present the algorithmic framework of `MPEMD` and describe how to calculate each minimization sub-step by using the closed form solutions in Section 3.3. Then we prove the convergence property that the generated sequence converges to a stationary point in a general way under some reasonable assumptions. Finally, when convergence results are specified into `MPEMD` under each f_{pq}, relatively simpler assumptions/conditions are demanded.

4.1 Majorization-Projection Method

Recall the main proposed penalty model (3.15), namely,

$$\begin{aligned} \min_{D\in\mathbb{S}^n} \quad & F_\rho(D) = f(D) + \rho g(D), \\ \text{s.t.} \quad & L \le D \le U, \end{aligned} \tag{4.1}$$

where

$$\begin{aligned} f(D) &= f_{pq} = \left\| W^{(1/q)} \circ \left(D^{(p/2)} - \Delta^{(p)} \right) \right\|_q^q, && (4.2) \\ g(D) &= \frac{1}{2} \left\| D + \Pi_{\mathbb{K}_+^n(r)}(-D) \right\|^2. && (4.3) \end{aligned}$$

4.1.1 Algorithmic Framework

Based on the majorization minimization (3.21) of (4.1), if we start with a computed point D^k, then could update next iteration by

$$\begin{aligned} D^{k+1} &= \underset{L \le D \le U}{\arg\min}\ f(D) + (\rho/2)\|D - D_K^k\|^2 && (4.4) \\ &= \underset{L \le D \le U}{\arg\min}\ f(D) + \rho g_M(D, D^k) \end{aligned}$$

where $D_K^k := -\Pi_{\mathbb{K}_+^n(r)}(-D^k)$ and g_M is defined as Proposition 1.5 (ii). Below is the table summarizing the framework of `MPEMD`.

Table 4.1: Framework of Majorization-Projection method.

```
Algorithm 4.1: Majorization-Projection method via EDM

Step 1   (Input data) Dissimilarity matrix $\Delta$, weight matrix $W$, lower- and upper-
         bound matrices $L, U$, penalty parameter $\rho > 0$, and initial $D^0$. Set $k := 0$.
Step 2   (Update) Compute $D_K^k := -\Pi_{\mathbb{K}_+^n(r)}(-D^k)$ and
             $D^{k+1} = \arg\min_{L \le D \le U} f(D) + (\rho/2)\|D - D_K^k\|^2$
Step 3   (Convergence check) Set $k := k + 1$ and go to Step 2 until convergence.
```

Remark 4.1. *One may notice that Step 2, namely (4.4), has a closed form solution whose form will be provided in next subsection. Therefore, the computation for each iteration is dominated by $\Pi_{\mathbb{K}_+^n(r)}(-D^k)$ in the construction of the majorization function in (4.4). The calculation of $\Pi_{\mathbb{K}_+^n(r)}(-D^k)$ is revealed by (1.31), that is*

$$\Pi_{\mathbb{K}_+^n(r)}(-D^k) = \mathrm{PCA}_r^+(-JD^kJ) + (JD^kJ - D^k),$$

which will solve $\mathrm{PCA}_r^+(-JD^kJ)$ eventually. Advantage of solving PCA_r^+ is that there is *a MATLAB's built-in function `eigs.m` whose complexity of computing this is about $O(rn^2)$. Hence, our method `MPEMD` has a low computational complexity and is very fast due to a small number of iterations required to meet the stopping criteria. Its efficiency will be convincingly demonstrated in numerical experiments, see Chapter 6.*

4.1.2 Solving Subproblems

For each f_{pq}, we compute subproblem (4.4) in Algorithm 4.1 respectively based on closed form solutions in Subsection 3.3.

$\boxed{f = f_{22}.}$ By (4.4), we have

$$D^{k+1} \quad = \quad \mathop{\arg\min}_{L \le D \le U} \left\| \sqrt{W} \circ \left(D - \Delta^{(2)} \right) \right\|^2 + \frac{\rho}{2} \left\| D - D_K^k \right\|^2. \tag{4.5}$$

According to (3.25), it follows

$$D_{ij}^{k+1} = \Pi_{[L_{ij}, U_{ij}]} \left[\frac{\rho (D_K^k)_{ij} + 2W_{ij}\delta_{ij}^2}{\rho + 2W_{ij}} \right]. \tag{4.6}$$

This also covers the case of $W_{ij} = 0$ in (3.23).

$\boxed{f = f_{21}.}$ By (4.4), we have

$$D^{k+1} \quad = \quad \mathop{\arg\min}_{L \le D \le U} \left\| W \circ \left(D - \Delta^{(2)} \right) \right\|_1 + \frac{\rho}{2} \left\| D - D_K^k \right\|^2. \tag{4.7}$$

According to (3.30), it follows

$$D_{ij}^{k+1} = \Pi_{[L_{ij}, U_{ij}]} \left[\delta_{ij}^2 + \mathrm{sign}(\widetilde{D}^k) \max \left\{ |\widetilde{D}^k| - W_{ij}/\rho, 0 \right\} \right] \tag{4.8}$$

where $\widetilde{D}^k := (D_K^k)_{ij} - \delta_{ij}^2$. This formula is also able to cover the case of $W_{ij} = 0$ in (3.23) because of

$$\mathrm{sign}(\widetilde{D}^k) \max\{|\widetilde{D}^k| - W_{ij}/\rho, 0\} = \mathrm{sign}(\widetilde{D}^k)|\widetilde{D}^k| = \widetilde{D}^k.$$

$\boxed{f = f_{12}.}$ By (4.4), we have

$$D^{k+1} \quad = \quad \mathop{\arg\min}_{L \le D \le U} \left\| \sqrt{W} \circ \left(\sqrt{D} - \Delta \right) \right\|^2 + \frac{\rho}{2} \left\| D - D_K^k \right\|^2. \tag{4.9}$$

According to (3.31), it follows

$$D_{ij}^{k+1} = \mathsf{dcrn}_{[L_{ij},U_{ij}]}\left[(D_K^k)_{ij} - W_{ij}/\rho,\ 2W_{ij}\delta_{ij}/\rho\right], \tag{4.10}$$

where dcrn is defined by (3.43), which is able to cover the case of $W_{ij} = 0$ in (3.23). In fact, one can verify that $x_{\omega,0}^{-} = \omega$ according to (3.38). Then (3.43) implies that $\mathsf{dcrn}_{[a,b]}[\omega, 0] = \Pi_{[a,b]}(x_{\omega,0}^{-}) = \Pi_{[a,b]}(\omega)$. Overall, when $W_{ij} = 0$, it has

$$\mathsf{dcrn}_{[L_{ij},U_{ij}]}\left[(D_K^k)_{ij},\ 0\right] = \Pi_{[L_{ij},U_{ij}]}((D_K^k)_{ij}),$$

coinciding with (3.23).

$\boxed{f = f_{11}.}$ By (4.4), we have

$$D^{k+1} = \mathop{\arg\min}_{L \le D \le U} \left\|W \circ \left(\sqrt{D} - \Delta\right)\right\|_1 + \frac{\rho}{2}\left\|D - D_K^k\right\|^2. \tag{4.11}$$

According to (3.46), if $\rho > \rho_0$ where ρ_0 is defined as (4.25), it follows

$$D_{ij}^{k+1} = \begin{cases} \mathsf{dcr}_{[L_{ij},U_{ij}]}\left[(D_K^k)_{ij},\ W_{ij}/\rho,\ \delta_{ij}\right], & W_{ij} > 0 \qquad (4.12a) \\ \Pi_{[L_{ij},U_{ij}]}\left[(D_K^k)_{ij}\right], & W_{ij} = 0 \qquad (4.12b) \end{cases}$$

where dcr is defined by (3.67).

4.2 Convergence Analysis

A major obstacle in analysing the convergence for Algorithm 4.1 is the existence of sub-gradients of objective function f, since some of them involve $\sqrt{D}$. Therefore, we assume the following conditions, before which we denote

$$\partial_{ij} f(Z) := \frac{\partial f(D)}{\partial D_{ij}}\Big|_{D=Z}. \tag{4.13}$$

Assumption 4.2. *$\partial f(D^k)$ is well deafened for any $k \ge 1$, namely, there is a constant $0 < c_0 < +\infty$ such that*

$$\|\Xi^k\| \le c_0, \qquad \forall\ \Xi^k \in \partial f(D^k).$$

Assumption 4.2 is to avoid the cases of $D_{ij}^k = 0$ which result in non-differentiability of the sub-differential of $\sqrt{D_{ij}^k}$, since

$$\lim_{D_{ij}^k \downarrow 0} \nabla \sqrt{D_{ij}^k} = \lim_{D_{ij}^k \downarrow 0} \frac{1}{2\sqrt{D_{ij}^k}} = +\infty.$$

where $D_{ij}^k \downarrow 0$ means $D_{ij}^k > 0$ and $\lim_{k\to\infty} D_{ij}^k = 0$. Luckily, in our following analysis, all functions f_{pq} enable us to get rid of such cases.

Assumption 4.3. *For subproblem (4.4), there exists $\Xi^{k+1} \in \partial f(D^{k+1})$ such that,*

$$\left\langle \Xi^{k+1} + \rho D^{k+1} + \rho \Pi_{\mathbb{K}_+^n(r)}(-D^k),\ D - D^{k+1} \right\rangle \geq 0, \quad \forall\ L \leq D \leq U. \tag{4.14}$$

It is easy to see that this assumption is the first order optimality condition of subproblem (4.4). It can be verified when f is convex for example when $f = f_{22}, f_{12}$ and $f = f_{21}$. When $f = f_{11}$, (4.14) has to be proved carefully. The need for such assumption is to guarantee that the sub-problem (4.4) at least admits a global solution.

Assumption 4.4. *There exists a $\rho_o \geq 0$ such that for any $\Xi^{k+1} \in \partial f(D^{k+1})$*

$$f(D^k) \geq f(D^{k+1}) + \langle \Xi^{k+1},\ D^k - D^{k+1} \rangle - \frac{\rho_o}{2} \|D^{k+1} - D^k\|^2. \tag{4.15}$$

This assumption holds for any $\rho_o \geq 0$ when f is convex, e.g. $f = f_{22}, f_{21}$ or f_{12}. When $f = f_{11}$, we prove it through choosing ρ properly. Such assumption somehow establishes the relation between the $f(D^k) - f(D^{k+1})$ and $D^k - D^{k+1}$

Assumption 4.5. *The constrained box is bounded, i.e., U is bounded from above.*

Assumption 4.5 can be easily satisfied (e.g., setting the upper bound to be twice the largest δ_{ij}^2). The reason to require this assumption is that it constrains the generated sequence in a bounded area which thus makes the sequence bounded, Otherwise, the sequence might be unbounded since the f is not strongly convex, which leaves a hard issue in terms of doing convergence analysis.

Notice that all these assumptions will be verified in the next subsection. And we will see assumptions are actually very weak. Hereafter, let $\{D^k\}$ be the sequence generated by Algorithm 4.1. Based on above assumptions, we are ready to give a general proof of the convergence property.

Theorem 4.6. *Suppose Assumptions 4.2-4.5 hold and $\rho > \rho_o$.*

(i) Let $F_\rho(D)$ be defined in (4.1), then

$$F_\rho(D^{k+1}) - F_\rho(D^k) \le -\frac{\rho - \rho_o}{2}\|D^{k+1} - D^k\|^2 \quad \textit{for any} \quad k \ge 1. \tag{4.16}$$

Consequently, $\|D^{k+1} - D^k\| \to 0$.

(ii) Let $\widehat{D}$ be an accumulation point of $\{D^k\}$. Then there exists $\widehat{\Xi} \in \partial f(\widehat{D})$ such that

$$\langle \widehat{\Xi} + \rho \widehat{D} + \rho \Pi_{\mathbb{K}^n_+(r)}(-\widehat{D}),\ D - \widehat{D}\rangle \ge 0. \tag{4.17}$$

holds for any $L \le D \le U$. That is, $\widehat{D}$ is a stationary point of the problem (4.1).

(iii) If $\widehat{D}$ is an isolated accumulation point of the sequence $\{D^k\}$, then the whole sequence $\{D^k\}$ converges to $\widehat{D}$.

Proof (i) We are going to use the following facts that are stated on D^{k+1} and D^k. The first fact is the identity:

$$\|D^{k+1}\|^2 - \|D^k\|^2 = 2\langle D^{k+1} - D^k,\ D^{k+1}\rangle - \|D^{k+1} - D^k\|^2. \tag{4.18}$$

The second fact is due to the convexity of $h(D)$ (see Proposition 1.4 (ii)):

$$h(-D^{k+1}) - h(-D^k) \ge \langle \Pi_{\mathbb{K}^n_+(r)}(-D^k),\ -D^{k+1} + D^k\rangle. \tag{4.19}$$

The third fact is from Proposition 1.5 (i):

$$g(D^{k+1}) - g(D^k) = \|D^{k+1}\|^2 - \|D^k\|^2 - [h(-D^{k+1}) - h(-D^k)] \tag{4.20}$$

The last fact is that there is a $\Xi^{k+1} \in \partial f(D^{k+1})$ such that (4.14). Those facts yield the following chain of inequalities:

$$\begin{aligned}
& F_\rho(D^{k+1}) - F_\rho(D^k) \\
= \;& f(D^{k+1}) - f(D^k) + \rho g(D^{k+1}) - \rho g(D^k) \\
\overset{(4.15)}{\le} \;& \left\langle \Xi^{k+1},\ D^{k+1} - D^k \right\rangle + (\rho_o/2)\|D^{k+1} - D^k\|^2 + \rho g(D^{k+1}) - \rho g(D^k) \\
\overset{(4.20)}{=} \;& \left\langle \Xi^{k+1},\ D^{k+1} - D^k \right\rangle + (\rho_o/2)\|D^{k+1} - D^k\|^2
\end{aligned}$$

$$\begin{aligned}
&+ \quad (\rho/2)\left(\|D^{k+1}\|^2 - \|D^k\|^2\right) - \rho\left[h(-D^{k+1}) - h(-D^k)\right] \\
&\overset{(4.18)}{=} \quad \left\langle \Xi^{k+1} + \rho D^{k+1},\ D^{k+1} - D^k \right\rangle - (\rho/2 - \rho_o/2)\|D^{k+1} - D^k\|^2 \\
&- \quad \rho\left[h(-D^{k+1}) - h(-D^k)\right] \\
&\overset{(4.19)}{\leq} \quad \left\langle \Xi^{k+1} + \rho D^{k+1} + \rho\Pi_{\mathbb{K}^n_+(r)}(-D^k),\ D^{k+1} - D^k \right\rangle - \frac{\rho - \rho_o}{2}\|D^{k+1} - D^k\|^2 \\
&\overset{(4.14)}{\leq} \quad -\frac{\rho - \rho_o}{2}\|D^{k+1} - D^k\|^2.
\end{aligned}$$

This proves that the sequence $\{F_\rho(D^k)\}$ is non-increasing and it is also bounded below by 0. Taking the limits on both sides yields $\|D^{k+1} - D^k\| \to 0$.

(ii) The sequence $\{D^k\}$ is bounded because $L \leq D^k \leq U$ and U is bounded by Assumption 4.5. Suppose $\widehat{D}$ is the limit of a subsequence $\{D^{k_\ell}\}$, $\ell = 1, \ldots,$. Since we have established in (i) that $(D^{k_{\ell+1}} - D^{k_\ell}) \to 0$, the sequence $\{D^{k_\ell+1}\}$ also converges to $\widehat{D}$. Furthermore, there exists a sequence of $\Xi^{k_\ell+1} \in \partial f(D^{k_\ell+1})$ such that (4.14) holds. Assumption 4.2 ensures that there is a contact $c_0 > 0$ such that $\|\Xi^{k_\ell+1}\| \leq c_0$ for all k_ℓ. Hence, there exists a subsequence of $\{k_\ell\}$ (we still denote the subsequence by $\{k_\ell\}$ for simplicity) such that $\Xi^{k_\ell+1}$ converges to some $\widehat{\Xi} \in \partial f(\widehat{D})$. Now taking the limits on both sides of (4.14) on $\{k_\ell\}$, we reach the desired inequality (4.17).

(iii) We note that we have proved in (i) that $(D^{k+1} - D^k) \to 0$. The convergence of the whole sequence to $\widehat{D}$ follows from (Kanzow and Qi, 1999, Prop. 7). □

Theorem 4.7. *If $D^0 \in -\mathbb{K}^n_+(r)$, $L \leq D^0 \leq U$ and $\rho \geq \max\{\rho_o, f(D^0)/\epsilon\}$, then any accumulation point $\widehat{D}$ of $\{D^k\}$ is also an ϵ-approximate KKT point of (3.1), that is*

$$\rho > 0, \quad g(\widehat{D}) \leq \epsilon, \quad \langle \widehat{\Xi},\ D - \widehat{D}\rangle \geq 0, \ \forall\ L \leq D \leq U. \tag{4.21}$$

Proof Similar to the proof of Theorem 4.6 (ii), we have $\widehat{\Xi} \in \partial f(\widehat{D})$, i.e., $\widehat{\Xi} + \rho\widehat{D} + \rho\Pi_{\mathbb{K}^n_+(r)}(-\widehat{D}) \in \partial L(\widehat{D},\ \rho)$,

$$\langle \widehat{\Xi} + \rho\widehat{D} + \rho\Pi_{\mathbb{K}^n_+(r)}(-\widehat{D}),\ D - \widehat{D}\rangle \geq 0. \tag{4.22}$$

which is the condition (3.19) with $\bar{\beta} = \rho$. We only need to show $g(\widehat{D}) \leq \epsilon$. Since $D^0 \in -\mathbb{K}^n_+(r)$ and $L \leq D^0 \leq U$, we have

$$f(D^0) \quad = \quad f(D^0) + \rho g(D^0) \qquad\qquad (\text{because } g(D^0) = 0)$$

$$
\begin{aligned}
&= f(D^0) + \rho g_M(D^0, D^0) \\
&\overset{(4.4)}{\geq} f(D^1) + \rho g_M(D^1, D^0) && \text{(because } L \leq D^0 \leq U\text{)} \\
&\geq f(D^1) + \rho g(D^1) \geq \cdots && \text{(because of Proposition1.5 (ii))} \\
&\overset{(4.16)}{\geq} f(D^k) + \rho g(D^k).
\end{aligned}
$$

Taking the limit on the right-hand side yields

$$f(D^0) \geq f(\widehat{D}) + \rho g(\widehat{D}) \geq \rho g(\widehat{D}),$$

where we used $f(\widehat{D}) \geq 0$. Therefore, it has

$$g(\widehat{D}) \leq f(D^0)/\rho \leq \epsilon.$$

We proved that $\widehat{D}$ is an ϵ-approximate KKT point of (3.1). □

4.3 Assumptions Verification

In this section, we verify whether Assumptions 4.2-4.4 are easy to be satisfied when f are specified as f_{pq}, before which we assume the following conditions:

Assumption 4.8. *It holds* $U_{ij} > 0$ *if* $\delta_{ij} > 0$.

Assumption 4.8 manifests that if $\delta_{ij} > 0$ then we want the upper bound $U_{ij} > 0$; otherwise, $0 = U_{ij} \geq L_{ij} \geq 0$, the corresponding $D_{ij} = 0$ is forced to be away from δ_{ij}^2, a very poor approximation to positive δ_{ij}.

Assumption 4.9. *It holds* $W_{ij} = 0$ *if* $\delta_{ij} = 0$.

Assumption 4.9 means that if $\delta_{ij} = 0$ (e.g., value missing), the corresponding weight W_{ij} is suggested to be zero. This is a common practice in applications. One may discern that if there is a certain $\delta_{ij} = 0$ that means the true distance between object i and j actually being zero rather than being missing, then corresponding W_{ij} is supposed to be nonzero (e.g., a small positive constant) to guarantee the estimated D_{ij} to be zero. However, such case of $\delta_{ij} = 0$ is able to be put into the constraints by setting $L_{ij} = U_{ij} = 0$. Then we still set $W_{ij} = 0$.

4.3.1 Conditions under f_{22}

When $f = f_{22}$, namely, $f_{22} = \|\sqrt{W} \circ (D - \Delta^{(2)})\|^2$, we have

- From Table 3.1, f_{22} is twice continuous differentiable and thus ∇f_{22} is well defined, i.e., for any $D \in \mathbb{S}^n$, it has $\nabla f_{22} = 2W \circ (D - \Delta^{(2)})$ and

$$\|\nabla f_{22}\| \leq 2\Big[\max_{ij} W_{ij}\Big]\Big[\|U\| + \|\Delta^{(2)}\|\Big] =: c_0 < +\infty,$$

  where $c_0 < +\infty$ if Assumption 4.5 holds. Hence Assumption 4.2 holds.

- From Table 3.1, f_{22} is convex and thus subproblem (4.4) (i.e.,(4.5)) is also convex. Hence Assumption 4.3 holds.

- The convexity of f_{22} yields that $f_{22}(D^k) \geq f_{22}(D^{k+1}) + \langle \Xi^{k+1},\ D^k - D^{k+1}\rangle$, where $\Xi^{k+1} = 2W \circ (D^{k+1} - \Delta^{(2)})$, which means Assumption 4.4 holds for any $\rho_o \geq 0$, particularly, we take $\rho_o = 0$.

Overall, we are able to weaken the assumptions Theorem 4.6 as

Theorem 4.10. *Let $\{D^k\}$ be the sequence generated by Algorithm 4.1 under f_{22} and $\rho > 0$. Suppose Assumption 4.5. Then (i), (ii) and (iii) in Theorem 4.6 hold.*

4.3.2 Conditions under f_{21}

When $f = f_{21}$, namely, $f_{21} = \|W \circ (D - \Delta^{(2)})\|_1$, we have

- From Table 3.1, f_{21} is non-differentiable but continuous, and its subdifferential (see (1.13)) is well defined as $\partial f_{21} = W \circ \text{sign}(D - \Delta^{(2)})$. Then for any $D \in \mathbb{S}^n$,

$$\|\Xi\| \leq n \max_{ij} W_{ij} =: c_0 < +\infty, \qquad \forall \Xi \in \partial f_{21}.$$

  Hence Assumption 4.2 holds.

- From Table 3.1, f_{21} is convex and thus subproblem (4.4) (i.e.,(4.7)) is also convex. Hence Assumption 4.3 holds.

- The convexity of f_{21} yields that $f_{21}(D^k) \geq f_{21}(D^{k+1}) + \langle \Xi^{k+1}, \ D^k - D^{k+1}\rangle$ for any $\Xi^{k+1} = W \circ \mathrm{sign}(D - \Delta^{(2)})$, which means Assumption 4.4 holds for any $\rho_o \geq 0$, particularly, we take $\rho_o = 0$.

Overall, we are able to weaken the assumptions Theorem 4.6 as

Theorem 4.11. *Let $\{D^k\}$ be the sequence generated by Algorithm 4.1 under f_{21} and $\rho > 0$. Suppose Assumption 4.5. Then (i), (ii) and (iii) in Theorem 4.6 hold.*

4.3.3 Conditions under f_{12}

When $f = f_{12}$, namely, $f_{12} = \|\sqrt{W} \circ (\sqrt{D} - \Delta)\|^2$, to establish the existence of ∂f_{12}, we need following lemmas.

Lemma 4.12. *Suppose Assumptions 4.5 and 4.8 hold. Let $\{D^k\}$ be the sequence generated by Algorithm 4.1 under f_{12} with $\rho > 0$. Then we have*

(i) For any (i, j) satisfying $W_{ij} > 0$, there exists $c_1 > 0$ such that

$$D_{ij}^k \geq c_1, \qquad k = 1, 2, \ldots.$$

And hence f_{12} is continuously differentiable at D^k for any $k \geq 1$;

(ii) For any $\Xi^k \in \partial f_{12}(D^k)$, $\{\Xi^k\}$ and its any accumulated points are bounded. And hence f_{12} is continuously differentiable at any of limits of the sequence $\{D^k\}$.

Proof (i) We write f_{12} in terms of D_{ij}:

$$f_{12} = \sum_{i,j} W_{ij} D_{ij} - 2 \sum_{i,j} W_{ij}\delta_{ij}\sqrt{D_{ij}} + \sum_{i,j} W_{ij}\delta_{ij}^2.$$

We will prove for any given pair (i, j), $\partial f(D)/\partial D_{ij}$ exists and is continuous at any point D^k. We consider two cases. Case 1: $W_{ij}\delta_{ij} = 0$. This implies $f(D)$ is a linear function of D_{ij} and $\partial f(D)/\partial D_{ij} = 2W_{ij}$ is constant and hence is continuous.

Case 2: $W_{ij}\delta_{ij} > 0$ which implies $W_{ij} > 0$. It follows from (4.10) that

$$D_{ij}^{k+1} = \mathsf{dcrn}_{[L_{ij}, U_{ij}]}\left[(D_K^k)_{ij} - \frac{W_{ij}}{\rho}, \frac{2W_{ij}\delta_{ij}}{\rho}\right]. \tag{4.23}$$

where $D_K^k := -\Pi_{\mathbb{K}_+^n(r)}(-D^k)$. Let $\omega_{ij}^k := (D_K^k)_{ij} - \rho^{-1}W_{ij}$ and $\alpha_{ij} := 2\rho^{-1}W_{ij}\delta_{ij} > 0$. One can verify that

$$\begin{aligned}
\omega_{ij}^k &\geq -|(D_K^k)_{ij}| - \rho^{-1}W_{ij} \geq -\|D_K^k\| - \rho^{-1}W_{ij} \\
&\geq -2\|D^k\| - \rho^{-1}W_{ij} \quad \geq -2\|U\| - \rho^{-1}\max_{ij} W_{ij} \\
&=: c > -\infty,
\end{aligned}$$

where the third inequality results from Proposition 1.5 (iii) and the last inequality is due to boundedness of U by Assumption 4.5. By (3.36), it suffices to $\gamma_{\omega_{ij}^k,\alpha_{ij}} \geq \gamma_{c,\alpha_{ij}} > 0$. Finally (3.44) in Proposition 3.8 (iii) yields that

$$\begin{aligned}
D_{ij}^{k+1} &\geq \min\left\{U_{ij}, 1, \gamma_{\omega_{ij}^k,\alpha_{ij}}\right\} \quad \geq \min\left\{U_{ij}, 1, \gamma_{c,\alpha_{ij}}\right\} \\
&\geq \min_{(i,j):W_{ij}>0} \left\{U_{ij}, 1, \gamma_{c,\alpha_{ij}}\right\} =: c_1 > 0,
\end{aligned} \tag{4.24}$$

where the last inequality benefits from $\gamma_{c,\alpha_{ij}} > 0$ and $U_{ij} > 0$ implied by $\delta_{ij} > 0$ via Assumption 4.8. Since $D_{ij}^{k+1} \geq c_1 > 0$ for any $k \geq 1$, we have

$$\nabla_{ij} f_{12}(D^{k+1}) = W_{ij}\left[1 - \delta_{ij}/\sqrt{D_{ij}^{k+1}}\right].$$

which is continuous. This proved (i).

(ii) It follows from the above two formulas that

$$|\nabla_{ij} f_{12}(D^{k+1})| \leq W_{ij}\left[1 + \delta_{ij}/\sqrt{c_1}\right],$$

which suffices to

$$\|\nabla f_{12}(D^{k+1})\| \leq n\left[\max_{ij} W_{ij}\right]\left[1 + \max_{ij} \delta_{ij}/\sqrt{c_1}\right] := c_0 < +\infty,$$

Since U is bounded (Assumption 4.5) and $L \leq D^k \leq U$, the sequence $\{D^k\}$ is bounded. Let $\widehat{D}$ be one of its limits. Without loss of any generality, let us assume $D^k \to \widehat{D}$. The proof below is the continuation in (i). For a given pair (i,j), if $W_{ij}\delta_{ij} = 0$, we have seen in (i) that $\partial f_{12}/\partial D_{ij}$ is a constant (independent of D^k). We only need to consider the case $W_{ij}\delta_{ij} > 0$, which implies $\delta_{ij} > 0$ and $U_{ij} > 0$ by Assumption 4.8. Taking limit on the left-hand side of (4.24), we get $\widehat{D}_{ij} \geq c_1 > 0$. Hence, $\partial f_{12}/\partial D_{ij}$ exists and is continuous at $\widehat{D}_{ij}$. This proved (ii) and complete the whole proof. $\square$

Based above lemma, we have

- Assumptions 4.5 and 4.8 ensure f_{12} is continuously differentiable at D^k, which thus makes Assumption 4.2 hold.
- From Table 3.1, f_{12} is convex and thus subproblem (4.4) (i.e.,(4.9)) is also convex. This together with Lemma 4.12 (i) derives Assumption 4.3.
- The convexity of f_{12} yields that $f_{12}(D^k) \geq f_{12}(D^{k+1}) + \langle \Xi^{k+1},\ D^k - D^{k+1}\rangle$, where $(\Xi^{k+1})_{ij} = W_{ij}(1-\delta_{ij}/\sqrt{D_{ij}^{k+1}})$, which means Assumption 4.4 holds for any $\rho_o \geq 0$, particularly, we take $\rho_o = 0$.

Overall, we are able to weaken the assumptions in Theorem 4.6 as

Theorem 4.13. *Let $\{D^k\}$ be the sequence generated by Algorithm 4.1 under f_{12} with $\rho > 0$. Suppose Assumptions 4.5 and 4.8. Then (i), (ii) and (iii) in Theorem 4.6 hold.*

4.3.4 Conditions under f_{11}

When $f = f_{11}$, namely, $f_{11} = \|W \circ (\sqrt{D} - \Delta)\|_1$, we first define a constant

$$\rho_o := \rho_o(W, \Delta) := \max_{(i,j):W_{ij}>0} \frac{W_{ij}}{4\delta_{ij}^3}. \tag{4.25}$$

This constant is well defined under Assumption 4.9, since $W_{ij} > 0$ implies $\delta_{ij} > 0$. To establish the existence of ∂f_{11}, we need following properties.

Lemma 4.14. *Let $\{D^k\}$ be the sequence generated by Algorithm 4.1 under f_{11} and $\rho > \rho_o$ with ρ_o being defined by (4.25). Suppose Assumptions 4.5, 4.8 and 4.9. Then*

(i) For any (i,j) satisfying $W_{ij} > 0$, there exists $c_1 > 0$ such that

$$D_{ij}^k \geq c_1, \qquad k = 1, 2, \ldots.$$

(ii) For any $\Xi^k \in \partial f_{11}(D^k)$, $\{\Xi^k\}$ and its any accumulated points are bounded.

Proof (i) We write f_{11} in terms of D_{ij}:

$$f_{11} = \sum_{i,j} W_{ij}|\sqrt{D_{ij}} - \delta_{ij}|.$$

We will prove for any given pair (i,j), $\partial f(D)/\partial D_{ij}$ exists and is continuous at any point D^k. We consider two cases. Case 1: $W_{ij}=0$. This implies f_{11} is a constant function of D_{ij} and $\partial f(D)/\partial D_{ij}=0$ is constant and hence is continuous.

Case 2: $W_{ij}>0$ which implies $\delta_{ij}>0$ (Assumption 4.9). It follows from (4.12) that

$$D_{ij}^{k+1} = \mathsf{dcr}_{[L_{ij},U_{ij}]}\left[(D_K^k)_{ij}, \frac{W_{ij}}{\rho}, \delta_{ij}\right].$$

where $D_K^k := -\Pi_{\mathbb{K}_+^n(r)}(-D^k)$. Let $\omega_{ij}^k := (D_K^k)_{ij}$ and $\alpha_{ij} := W_{ij}/\rho > 0$. One can verify

$$\omega_{ij}^k \geq -|(D_K^k)_{ij}| \geq -\|D_K^k\| \geq -2\|D^k\| \geq -2\|U\| =: c > -\infty,$$

where the third inequality results from Proposition 1.5 (iii) and the last inequality is due to boundedness of U by Assumption 4.5. In addition,

$$\rho > \rho_o = \max_{(i,j):W_{ij}>0} W_{ij}/(4\delta_{ij}^3) \quad \text{indicates} \quad 0 < \alpha_{ij} = W_{ij}/\rho < 4\delta_{ij}^3.$$

By (3.36), it suffices to $\gamma_{\omega_{ij}^k,\alpha_{ij}} \geq \gamma_{c,\alpha_{ij}} > 0$. Those enable Proposition 3.14 (iii) to yield

$$D_{ij}^{k+1} \geq \min\left\{\delta_{ij}^2, U_{ij}, 1, \gamma_{\omega_{ij}^k,\alpha_{ij}}\right\} \geq \min\left\{\delta_{ij}^2, U_{ij}, 1, \gamma_{c,\alpha_{ij}}\right\} =: c_1 > 0, \tag{4.26}$$

where the last inequality benefits from $\gamma_{c,\alpha_{ij}} > 0$ and $U_{ij} > 0$ implied by $\delta_{ij} > 0$ via Assumption 4.8.

(ii) Clearly, we have

$$\partial_{ij} f_{11}(D^k) = \left\{W_{ij}\text{sign}\left(\sqrt{D_{ij}^k} - \delta_{ij}\right) \Big/ \left(2\sqrt{D_{ij}^k}\right)\right\}.$$

which combining (i) suffices to

$$|\xi_{ij}^k| \leq W_{ij}/\sqrt{4c_1}, \quad \forall\, \xi_{ij}^k \in \partial_{ij} f_{11}(D^k).$$

In other words, $\partial_{ij} f_{11}(D^k)$ is bounded by $W_{ij}/\sqrt{4c_1}$, which is independent of the index k. It follows directly from the definition of subdifferential (Rockafellar and Wets, 2009, Chp. 8.3) that

$$\partial f_{11}(D^k) \subseteq \bigotimes \partial_{ij} f_{11}(D^k)$$

in the sense that for any $\Xi^k \in \partial f_{11}(D^k)$, there exist $\xi^k_{ij} \in \partial_{ij} f_{11}(D^k)$ such that

$$\Xi^k_{ij} = \xi^k_{ij}, \qquad i,j = 1,\ldots,n.$$

Consequently, we have for all $k = 1,2,\ldots,$

$$\|\Xi^k\| \le n \max_{i,j} |\xi^k_{ij}| \le n \max_{i,j} W_{ij}/(2\sqrt{c_1}) := c_0 > 0.$$

Since $L \le D^k \le U$ which is a bounded region by Assumption 4.5, it has a convergent subsequence. Let $\widehat{D}$ be one of its accumulated points. Without loss of any generality, let us assume $D^k \to \widehat{D}$. The proof below is the continuation of (i).

For a given pair (i,j), if $W_{ij} = 0$, we have seen in (i) that $\partial f_{11}/\partial D_{ij} = 0$ is a constant (independent of D^k). We only need to consider the case $W_{ij} > 0$. Similar reasons allow us to prove that $D^k_{ij} \ge c_1 > 0$. Taking limit on the left-hand side, we get $\widehat{D}_{ij} \ge c_1 > 0$. Hence, for any $\widehat{\Xi} \in \partial f_{11}(\widehat{D})$,we have $|\widehat{\Xi}| \le c_0$. This completes the whole proof. □

Lemma 4.15. *Suppose assumptions of Lemma 4.14 hold. Then (4.11) is convex, and thus there exists $\Xi^{k+1} \in \partial f_{11}(D^{k+1})$ such that*

$$\left\langle \Xi^{k+1} + \rho D^{k+1} + \rho \Pi_{\mathbb{K}^n_+(r)}(-D^k),\ D - D^{k+1} \right\rangle \ge 0, \tag{4.27}$$

hols for any $L \le D \le U$.

Proof Since (4.11) is separable, it can be written as

$$D^{k+1}_{ij} \;=\; \operatorname*{arg\,min}_{L_{ij} \le D_{ij} \le U_{ij}} \ W_{ij}|\sqrt{D_{ij}} - \delta_{ij}| + (\rho/2)(D_{ij} - (D^k_K)_{ij})^2. \tag{4.28}$$

When $\rho > \rho_o = \max_{(i,j):W_{ij}>0}\ W_{ij}/(4\delta^3_{ij})$, it has $0 < \alpha := W_{ij}/\rho < 4\delta^3_{ij}$. By Proposition 3.14 (i), above problem (4.28) is strictly convex. In addition, Lemma 4.14 (i) proved $\partial f_{11}(D^{k+1})$ is well-defined. Those allow us to claim (4.27) immediately. □

Lemma 4.16. *Suppose assumptions of Lemma 4.14 hold. Then*

$$f_{11}(D^k) \ge \langle f_{11}(D^{k+1}) + \langle \Xi^{k+1},\ D^k - D^{k+1}\rangle - \frac{\rho_o}{2}\|D^{k+1} - D^k\|^2 \tag{4.29}$$

holds for any $\Xi^{k+1} \in \partial f_{11}(D^{k+1})$

Proof Direct calculation yields the following chain of inequalities

$$\begin{aligned} f_{11}(D^k) - f_{11}(D^{k+1}) &= \sum_{ij} W_{ij}\left[|\sqrt{D^k_{ij}} - \delta_{ij}| - |\sqrt{D^{k+1}_{ij}} - \delta_{ij}|\right] \\ &\geq \sum_{ij} W_{ij}\left[\zeta^{k+1}_{ij}(D^{k+1}_{ij} - D^k_{ij}) - (D^k_{ij} - D^{k+1}_{ij})^2/(8\delta^3)\right] \\ &\geq \sum_{ij} \left[W_{ij}\zeta^{k+1}_{ij}(D^{k+1}_{ij} - D^k_{ij}) - \frac{\rho_o}{2}(D^k_{ij} - D^{k+1}_{ij})^2\right] \\ &= \langle \Xi^{k+1},\ D^k - D^{k+1}\rangle - \frac{\rho_o}{2}\|D^{k+1} - D^k\|^2 \end{aligned}$$

where $\zeta^{k+1}_{ij} \in \partial(|\sqrt{D^{k+1}_{ij}} - \delta_{ij}|)$ and $(\Xi^{k+1})_{ij} = W_{ij}\zeta^{k+1}_{ij}$, the first and the last inequalities are due to Proposition 3.13 and $\rho_o = \max_{(i,j):W_{ij}>0}\ W_{ij}/(4\delta^3_{ij})$ respectively. $\square$

Based above lemmas, we have conditions of Lemma 4.14 enable us to prove Lemmas 4.14, 4.15 and 4.16, which implies Assumptions 4.2, 4.3 and 4.4 respectively. Overall we are able to alter the assumptions in Theorem 4.6 as

Theorem 4.17. *Let $\{D^k\}$ be the sequence generated by Algorithm 4.1 under f_{11} and $\rho > \rho_o$ with ρ_o being defined by (4.25). Suppose Assumptions 4.5, 4.8 and 4.9. Then (i), (ii) and (iii) in Theorem 4.6 hold.*

Table 4.2: Conditions assumed under each objective function

p, q	Assumptions	Parameter $\rho > \rho_o$
$f_{22} = \|\sqrt{W} \circ (D - \Delta^{(2)})\|^2$	Ass. 4.5	$\rho > 0$
$f_{21} = \|W \circ (D - \Delta^{(2)})\|_1$	Ass. 4.5	$\rho > 0$
$f_{12} = \|\sqrt{W} \circ (\sqrt{D} - \Delta)\|^2$	Ass. 4.5, 4.8	$\rho > 0$
$f_{11} = \|W \circ (\sqrt{D} - \Delta)\|_1$	Ass. 4.5, 4.8, 4.9	$\rho > \max\limits_{(i,j):W_{ij}>0} \frac{W_{ij}}{4\delta^3_{ij}}$

To end this section, we summarize conditions to derive the convergence properties under each f_{pq} in Table 4.2. It is worth to mentioning that all conditions are assumed on the known data (i.e., U, W, ρ) and as we mentioned above, all assumptions are reasonable and very easy to be satisfied. For example, for Assumption 4.9, if $\delta_{ij} = 0$ we actually hope $W_{ij} = 0$. This is because in many applications, $\delta_{ij} = 0 (i \neq j)$ means the the ground truth value $d_{ij} > 0$ is missing rather than d_{ij} being zero. If we set $W_{ij} > 0$ then $W_{ij}(\sqrt{D_{ij}} - \delta_{ij}) = W_{ij}(\sqrt{D_{ij}} - 0)$ may result in $D_{ij} = 0$ which might be away from d_{ij}, hence leading a poor estimation.

Chapter 5

Applications via EDM Optimization

In this chapter, we focus on the four previously mentioned applications in Section 1.2. For each application, mathematical formula will be cast by using EDM theory in Section 1.4. One may discern that in this way it is capable of dealing with various of constraints, such as linear equations or bounded constraints. When it comes to the numerical implementations, their data generations will be explained.

5.1 Wireless Sensor Network Localization

Wireless Sensor Networks (WSNs) can be applied in many applications, such as natural resources investigation, targets tracking, unapproachable places monitoring and so forth. In these applications, the information is collected and transferred by the sensor nodes. Various applications request these sensor nodes location information.

The Global Positioning System (GPS) is the most promising and accurate positioning technologies. Although it is widely accessible, the limitation of high cost and energy consuming of GPS system makes it impractical to install in every sensor node where the lifetime of a sensor node is very crucial. In order to reduce the energy consumption and cost, only a few number of nodes called anchors contain the GPS modules. The other nodes could obtain their position information through localization method. Wireless sensor network is composed of a large number of inexpensive nodes that are densely deployed in a region of interests to measure certain phenomenon.

Therefore, localization algorithms become one of the most important issues in WSNs researches, and have been intensively studied in recent years, with most of these studies relying on the condition that only a small proportion of sensor nodes (anchors) know exact positions through GPS devices or manual configuration. Other sensor nodes only collect their distances to the neighbour nodes and calculate positions by localizing techniques later on.

5.1.1 Problematic Interpretation

The general setting of wireless SNL is as follows. Assume a sensor network in $\mathbb{R}^r$ ($r = 2$ or 3) has n nodes in total, with m known anchor nodes and $s(:= n - m)$ sensor nodes to be located. Let $\mathbf{x}_i = \mathbf{a}_i, i = 1, \dots, m$ denote the location of ith anchor node, and $\mathbf{x}_j, j = m+1, \dots, n$ denote the location of jth sensor node. The maximum communication range is R, which determines two index sets N_{xx} and N_{ax} that indicates the connectivity states of nodes. For any $(j, k) \in N_{xx}$, the Euclidean distance between sensor nodes $\mathbf{x}_j$ and $\mathbf{x}_k$ is not greater than R. Hence, the two sensor nodes are able to transmit signal between each other. Similarly, for any $(i, j) \in N_{ax}$, the Euclidean distance between an anchor $\mathbf{x}_i$ and a sensor $\mathbf{x}_j$ is smaller R, making them able to communicate. Denote

$$\begin{aligned}
N_{aa} &= \{(i,j) : \ i < j = 1, \dots, m\}, \\
N_{ax} &= \{(i,j) : \ \|\mathbf{x}_i - \mathbf{x}_j\| \le R, \ i = 1, \dots, m, \ j = m+1, \dots, n\}, \\
N_{xx} &= \{(j,k) : \ \|\mathbf{x}_j - \mathbf{x}_k\| \le R, \ j < k = m+1, \dots, n\}, \qquad (5.1) \\
\overline{N}_{ax} &= \{(i,j) : \ \|\mathbf{x}_i - \mathbf{x}_j\| > R, \ i = 1, \dots, m, \ j = m+1, \cdots, n\}, \\
\overline{N}_{xx} &= \{(j,k) : \ \|\mathbf{x}_j - \mathbf{x}_k\| > R, \ j < k = m+1, \dots, n\}.
\end{aligned}$$

where $\overline{N}_{ax}$ and $\overline{N}_{xx}$ indicate a pair of nodes that are too far away to communicate. If two nodes can transmit signal, then their distance can be measured, namely, for any nodes $\mathbf{x}_j$ and $\mathbf{x}_k$, where $(j, k) \in N_{ax} \cup N_{xx}$, a range measurement but with being contaminated by noise due to the reality environment (i.e., the dissimilarity) can be obtained,

$$\delta_{jk} = \|\mathbf{x}_j - \mathbf{x}_k\| + \epsilon_{jk}, \quad (j, k) \in N_{ax} \cup N_{xx}, \qquad (5.2)$$

where ϵ_{ij}s are noises. We always assume that the distance estimations are symmetric, i.e., $\delta_{jk} = \delta_{kj}$. Overall, the range based sensor network localization problem can be described as to recover $\{\mathbf{x}_{m+1}, \ldots, \mathbf{x}_n\}$ in $\mathbb{R}^r$ such that

$$\begin{aligned} \|\mathbf{x}_j - \mathbf{x}_k\| &\approx \delta_{jk}, &\forall\ (j,k) \in N_{xx}, \quad (5.3)\\ \|\mathbf{x}_j - \mathbf{x}_k\| &> R, &\forall\ (j,k) \in \overline{N}_{xx}, \quad (5.4)\\ \|\mathbf{x}_i - \mathbf{a}_j\| &\approx \delta_{ij}, &\forall\ (i,j) \in N_{ax}, \quad (5.5)\\ \|\mathbf{x}_i - \mathbf{a}_j\| &> R, &\forall\ (i,j) \in \overline{N}_{ax}, \quad (5.6)\end{aligned}$$

Constraints (5.3) and (5.5) come from the incomplete distance information, and (5.4) and (5.6) 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 (5.4) and (5.6). 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.

From the point of view of EDM theory in Section 1.4, it allows us to derive a given matrix $\Delta \in \mathbb{S}^n$ in advance, for $i \leq j$,

$$\Delta_{ij} = \begin{cases} 0, & (i,j) \in N_{aa}, \\ \delta_{ij}, & (i,j) \in N_{ax} \cup N_{xx}, \\ 0, & \text{otherwise.} \end{cases} \quad (5.7)$$

In a nutshell, SNL problem is to find an EDM D with embedding dimension r that is nearest to $\Delta^{(2)}$ and satisfies constraints (5.3-5.6). By these constraints and (1.30), in other words, it aims at approximating $\Delta^{(2)}$ by D such that

$$\begin{aligned} -D &\in \mathbb{S}^n_h \cap \mathbb{K}^n_+(2) & (5.8)\\ D_{ij} &= \|\mathbf{a}_i - \mathbf{a}_j\|^2, \quad (i,j) \in N_{aa}, & (5.9)\\ D_{ij} &\leq R, \quad (i,j) \in N_{xx} \cup N_{ax}, & (5.10)\\ D_{ij} &\geq R, \quad (i,j) \in \overline{N}_{xx} \cup \overline{N}_{ax}. & (5.11)\end{aligned}$$

Here, we put anchors information (5.9) into constraints since we treat the whole D as a variable, which explains no information in Δ when $(i,j) \in N_{aa}$ are provided in (5.7). To derive the box constraints $L \le D \le U$ in (3.1) and $L, U \in \mathbb{S}^n$, for any $i \le j$, we set

$$L_{ij} = \begin{cases} 0, & i = j & (5.12a) \\ \|\mathbf{a}_i - \mathbf{a}_j\|^2, & (i,j) \in N_{aa} & (5.12b) \\ R, & (i,j) \in \overline{N}_{xx} \cup \overline{N}_{ax} & (5.12c) \end{cases}$$

$$U_{ij} = \begin{cases} 0, & i = j & (5.13a) \\ \|\mathbf{a}_i - \mathbf{a}_j\|^2, & (i,j) \in N_{aa} & (5.13b) \\ R, & (i,j) \in N_{xx} \cup N_{ax} & (5.13c) \end{cases}$$

5.1.2 Data Generation

This subsection describes data generation of SNL examples. Some of them are either direct versions or slightly modified versions of those in existing literature, such as (Bai and Qi, 2016; Biswas et al., 2006; Qi and Yuan, 2014; Tseng, 2007).

Example 5.1. (Biswas et al., 2006; Qi and Yuan, 2014; Tseng, 2007) *This example is widely tested since it was detailed studied by Biswas et al. (2006). First, $m = 4$ anchors are placed at four inner corners $(\pm 0.2, \pm 0.2)$. Then $(n - m)$ points are randomly generated in the unit square $[-0.5, 0.5] \times [-0.5, 0.5]$ via the MATLAB command:*

$$X = -0.5 + \texttt{rand}(2, n - m).$$

Example 5.2. (Biswas et al., 2006; Qi and Yuan, 2014; Tseng, 2007) *First, $m = 4$ anchors are placed at four outer corners $(\pm 0.45, \pm 0.45)$. Then the generation of rest $(n - m)$ points are similar to Example 5.1.*

Example 5.3. (Tseng, 2007) *The n points are randomly generated in the square $[-0.5, 0.5]$ $\times[-0.5, 0.5]$ via the MATLAB command: $X = -0.5 + \texttt{rand}(2, n)$. Then, the first m columns of X are chosen to be anchors and the rest $n - m$ columns are sensors.*

Example 5.4. (EDM word network, Bai and Qi (2016)) *This problem has a non-regular topology and is first used in Bai and Qi (2016) to challenge existing localization methods. In this example, n points are randomly generated in a region whose shape is similar to the letters "E", "D" and "M". The ground truth network is depicted in Figure 5.1. We choose the first m points to be the anchors and the rest $n - m$ columns are sensors.*

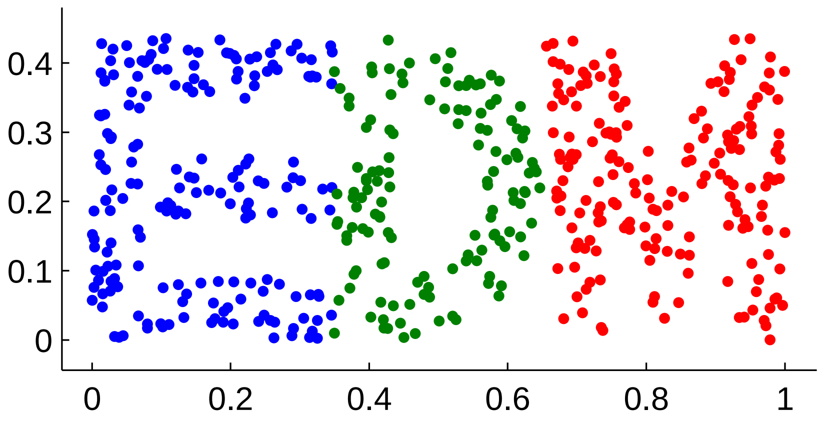


Figure 5.1: Ground truth EDM network with 500 nodes.

Let $[\mathbf{x}_1 \cdots \mathbf{x}_n] =: X$, namely ground truth point $\mathbf{x}_i$ is the ith column of X. Based on Subsection 5.1.1, we are next to generate N_{ax} and N_{xx} through (5.1) decided by the maximum communication range R (e.g., $R = 0.2$). Then similar to (5.2) noise contaminated distances will be observed, that is

$$\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \cdot |1 + \epsilon_{ij} \cdot \mathtt{nf}|, \quad (i,j) \in N_{ax} \cup N_{xx}, \tag{5.14}$$

where $\mathtt{nf}$ is the noise factor (e.g., $\mathtt{nf} = 0.1$ corresponds 10% of noise level); and ϵ_{ij} are independent standard normal random variables. This type of perturbation in δ_{ij} is known to be multiplicative and follows the unit-ball rule in defining N_{xx} and N_{ax} (see (Bai and Qi, 2016, Sect. 3.1) for more detail).

The last issue confronting us is to set those parameters: $W, \Delta, L, U \in \mathbb{S}^n$ which are generated as Table 5.1, where Δ is taken from (5.7); L, U are set relied on (5.12) and (5.13); W is given to satisfy the Assumption 4.9; Moreover, to meet Assumption 4.5, M is a positive bounded constant, e.g., $M := n \max_{ij} \Delta_{ij}$.

Table 5.1: Parameter generation of SNL.

(i,j)	W_{ij}	Δ_{ij}	L_{ij}	U_{ij}
$i = j$	0	0	0	0
$(i,j) \in N_{aa}$	0	0	$\|\mathbf{a}_i - \mathbf{a}_j\|^2$	$\|\mathbf{a}_i - \mathbf{a}_j\|^2$
$(i,j) \in N_{ax} \cup N_{xx}$	1	δ_{ij}	0	R^2
$(i,j) \in \overline{N}_{ax} \cup \overline{N}_{xx}$	0	0	R^2	M^2

5.1.3 Impact Factors

For SNL problems, for each fixed n, a network will be affected by three factors: **radio range** R**, anchors number** m, and **noise factor** $\texttt{nf}$. Clearly, the radio range R decides the amount of missing dissimilarities among all elements of Δ. The smaller R is, the more numbers of δ_{ij} are unavailable, yielding problems more challenging to be solved. As what we expect, more anchors given means more information provided, and because of this, more easier the problems would be. Finally, when a noise with large factor $\texttt{nf}$ contaminates the distance, then the dissimilarity will get far away from the truth distance, which apparently leads to a tough network to be localized. Therefore, we will test our method to see its sensitivity to these three factors through fixing two factors and altering the third one from a proper range.

For each example, since it is generated randomly, we will test 20 times for each instance $(n, m, R, \texttt{nf})$, and record average results over 20 times. For example, if we aim to see the performance along with the changing of R, we will fix $n = 200, m = 4, \texttt{nf} = 0.1$ and alter $R \in \{0.2, 0.4, \ldots, 1.4\}$ for Example 5.1. Then for each instance $(n, m, R, \texttt{nf})$ $= (200, 4, R, 0.1)$, we run our method 20 times and $20 \times 7 = 140$ times in total.

5.2 Molecular Conformation

An important area of research in computational biochemistry is the design of molecules for specific applications. Examples of these types of applications occur in the development of enzymes for the removal of toxic wastes, the development of new catalysts for material processing and the design of new anti-cancer agents. The design of these drugs depends on the accurate determination of the structure of biological macro-molecules. This problem is known as the molecular conformation problem, and has long been an important application of EDM optimization (Glunt et al., 1993; Moré and Wu, 1997).

5.2.1 Problematic Interpretation

The setting of MC problem is as follows. For a given molecule with n atoms $\{\mathbf{x}_1, \ldots \mathbf{x}_n\}$ in $\mathbb{R}^3$, if the Euclidean distance between two atoms is less than R (where R is the maximal distance that some equipments can measure), then the distance is chosen; otherwise no distance information about this pair is known. For example, $R = 6\mathring{A}$ ($1\mathring{A} = 10^{-8}$cm) is nearly the maximal distance that the nuclear magnetic resonance(NMR) experiment can measure between two atoms. For realistic molecular conformation problems, not all the distances below R are known from NMR experiments, so one may obtain $c\%$ (e.g., $c = 30\%$) of all the distances below R. Similar to (5.1), denote N_{xx} a set formed by indices of those measured distances. Moreover, the exact distances in N_{xx} actually can not be measured and only the noisy contaminated lower bounds a_{ij} and upper bounds b_{ij} on distances are provided, that is for $(i, j) \in N_{xx}$,

$$a_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| + \epsilon_{ij}, \qquad b_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| + \varepsilon_{ij}. \tag{5.15}$$

where ϵ_{ij} and ε_{ij} are noises. A typical noise rule used by Jiang et al. (2013) is

$$a_{ij} = \max\left\{1, (1 - |\epsilon_{ij}|)\|\mathbf{x}_i - \mathbf{x}_j\|\right\}, \quad b_{ij} = (1 + |\varepsilon_{ij}|)\|\mathbf{x}_i - \mathbf{x}_j\|. \tag{5.16}$$

where ϵ_{ij} and ε_{ij} are independent normal or uniform random variables. Therefore the task of MC problem is to find $\{\mathbf{x}_1, \ldots \mathbf{x}_n\}$ in $\mathbb{R}^3$ such that

$$a_{ij} \le \|\mathbf{x}_i - \mathbf{x}_j\| \le b_{ij} \quad \text{for any} \quad (i, j) \in N_{xx} \tag{5.17}$$

From definition of EDM in Subsection 1.4.1,, an information matrix $\Delta \in \mathbb{S}^n$ can be derived first, for $i \le j$,

$$\Delta_{ij} = \begin{cases} (a_{ij} + b_{ij})/2, & (i, j) \in N_{xx}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.18}$$

Overall, MC problem is to find an EDM D with embedding dimension 3 that is nearest to $\Delta^{(2)}$ and satisfies (5.17) and (1.30), namely,

$$-D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+(3) \tag{5.19}$$

$$a^2_{ij} \le D_{ij} \le b^2_{ij}, \quad (i, j) \in N_{xx}. \tag{5.20}$$

To derive the box constraints $L \le D \le U$ in (3.1) and $L, U \in \mathbb{S}^n$, for any $i \le j$, we set

$$L_{ij} = \begin{cases} 0, & i = j, \\ a^2_{ij}, & (i, j) \in N_{xx}, \end{cases} \qquad U_{ij} = \begin{cases} 0, & i = j, \\ b^2_{ij}, & (i, j) \in N_{xx}. \end{cases} \tag{5.21}$$

5.2.2 Data Generation

Two MC examples with artificial data and real data from Protein Data Bank (PDB) Berman et al. (2002) respectively will be studied in this part. For the former, we adopt the rule of generating data from (Moré and Wu, 1997; An and Tao, 2003). For the latter, we collected real data of 12 molecules derived from 12 structures of proteins from PDB. They are `1GM2, 304D, 1PBM, 2MSJ, 1AU6, 1LFB, 104D, 1PHT, 1POA, 1AX8, 1RGS, 2CLJ`. They provide a good set of test problems in terms of the size n, which ranges from a few hundreds to a few thousands (the smallest $n = 166$ for `1GM` and the largest $n = 4189$ for `2CLJ`). The distance information was obtained in a realistic way as done by Jiang et al. (2013).

Example 5.5. (Moré and Wu, 1997; An and Tao, 2003) *The artificial molecule has $n = s^3$ atoms $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ located in the three-dimensional lattice*

$$\{(i_1, i_2, i_3) : i_1, i_2, i_3 = 0, 1, \ldots, s-1\}$$

for some integer $s \geq 1$, i.e., $\boldsymbol{x}_i = (i_1, i_2, i_3)^\top$.

Since for MC problem no atoms are known in advance, it follows $m = 0$, i.e., $N_{ax} = \emptyset$. Similar to (Moré and Wu, 1997; An and Tao, 2003), we adapt two rules to define N_{xx} which determines the index set on which δ_{ij} are available as:

$$\begin{aligned} \text{Rule 1:} \quad N_{xx} &:= \{(i,j) : ||\mathbf{x}_i - \mathbf{x}_j|| \leq R\} && (5.22) \\ \text{Rule 2:} \quad N_{xx} &:= \{(i,j) : |\chi(\mathbf{x}_i) - \chi(\mathbf{x}_j)| \leq \sigma\}, && (5.23) \end{aligned}$$

where $R \geq 1, \sigma \geq 0$ and

$$\chi(\mathbf{x}_i) := 1 + (1, s, s^2)\mathbf{x}_i = 1 + i_1 + si_2 + s^2 i_3.$$

Clearly Rule 1 is same as (5.1). As indicated by Moré and Wu (1997), a difference between these definitions of N_{xx} is that (5.23) includes all nearby atoms, while (5.22) includes some of nearby atoms and some relatively distant atoms.

Then similar to (5.14), noise contaminated distances will be observed, that is

$$\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\| \cdot |1 + \epsilon_{ij} \cdot \mathtt{nf}|, \quad (i,j) \in N_{xx}, \tag{5.24}$$

where nf is the noise factor and ϵ_{ij} are independent standard normal random variables. Finally, the generation of $W, \Delta, L, U \in \mathbb{S}^n$ are taken as in Table 5.2 where M is a positive bounded constant to meet Assumption 4.5 , e.g., $M := n \max_{ij} \Delta_{ij}$ for Rule 1 and $M := \sqrt{3}(s-1)$ for Rule 2.

Table 5.2: Parameter generation of MC problem with artifical data.

	Rule 1			Rule 2		
(i,j)	$i=j$	$(i,j) \in N_{xx}$	otherwise	$i=j$	$(i,j) \in N_{xx}$	otherwise
W_{ij}	0	1	0	0	1	0
Δ_{ij}	0	δ_{ij}	0	0	δ_{ij}	0
L_{ij}	0	1	R^2	0	1	1
U_{ij}	0	R^2	M^2	0	$\max_{(i,j)\in N_{xx}} \|\|\mathbf{x}_i - \mathbf{x}_j\|\|^2$	M^2

Example 5.6. (Real PDB data) *We collect 12 molecules derived from 12 structures of proteins from PDB. Each molecule comprises n atoms $\{\boldsymbol{x}_1, \ldots \boldsymbol{x}_n\}$ in $\mathbb{R}^3$.*

Table 5.3: Parameter generation of MC problem with PDB data.

(i,j)	W_{ij}	Δ_{ij}	L_{ij}	U_{ij}
$i=j$	0	0	0	0
$(i,j) \in N_{xx}$	1	$(a_{ij}+b_{ij})/2$	a_{ij}^2	b_{ij}^2
$(i,j) \in \overline{N}_{xx}$	0	0	0	M^2

As described in Subsection 5.2.1, we first generate N_{xx}, and then the noise contaminated lower and upper bounds of distances on N_{xx}, namely (5.16) where we take the noise from the normal distribution as,

$$\epsilon_{ij}, \varepsilon_{ij} \sim N(0, \texttt{nf}^2 \times \pi/2).$$

Finally, parameters $W, \Delta, L, U \in \mathbb{S}^n$ are given as in Table 5.3, where Δ is from (5.18), L, U are decided by (5.21) and $M := n \max_{ij} \Delta_{ij}$.

5.2.3 Impact Factors

For MC problem, for each fixed n, a molecular will be affected by two factors: **range** R **or** σ, and **noise factor** $\texttt{nf}$. Similar to the test on SNL problems, for each example, we test 20 times for each instance $(s, R, \texttt{nf})$ or $(s, \sigma, \texttt{nf})$, and record average results over 20 times. For example, if we aim to see the performance along with the changing of R, we will fix $s = 6(n = s^3), \texttt{nf} = 0.1$ and alter $R \in \{2, 3, \ldots, 8\}$ for Example 5.5 under Rule 1. Then for each instance $(s, R, \texttt{nf}) = (6, R, 0.1)$, we run our method 20 times, which means for such example `MPEDM` will be run $20 \times 7 = 140$ times in total.

5.3 Embedding on A Sphere

Embedding given objects on a sphere to be estimated arises from various disciplines such as Statistics (spatial data representation), Psychology (constrained multidimensional scaling), and Computer Science (machine learning and pattern recognition).

5.3.1 Problematic Interpretation

The purpose of this problem is to find a sphere in $\mathbb{R}^r$ fits a group of given points $\{\mathbf{x}_1, \ldots \mathbf{x}_{n-1}\}$ in $\mathbb{R}^r$ ($r = 2$ or 3) in a best way. Generally, the center and radius of sphere are unknown. If we introduce an extra unknown point $\mathbf{x}_n$ to denote the center and an unknown variable R to denote the radius, then the problem is able to be describe as finding $\mathbf{x}_n$ and R such that

$$\|\mathbf{x}_i - \mathbf{x}_n\| \approx R, \quad i = 1, \ldots, n-1. \tag{5.25}$$

For more details, one can refer to (Bai et al., 2015; Beck and Pan, 2012). From definition of EDM in Subsection 1.4.1, a dissimilarity matrix $\Delta \in \mathbb{S}^n$ can be derived first, namely, for $i \leq j$,

$$\Delta_{ij} = \begin{cases} \|\mathbf{x}_i - \mathbf{x}_j\|, & i, j = 1, \ldots, n-1, \\ R_o, & i = 1, \ldots, n-1, j = n. \end{cases} \tag{5.26}$$

where R_o can be estimated straightforwardly such as $R_o := \max_{ij} \|\mathbf{x}_i - \mathbf{x}_n\|/2$.

Overall, this problem is to find an EDM D with embedding dimension r that is nearest to $\Delta^{(2)}$. By (1.30), in other words, it aims at approximating $\Delta^{(2)}$ by D such that

$$-D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+(r) \tag{5.27}$$

To derive the box constraints $L \leq D \leq U$ in (3.1) and $L, U \in \mathbb{S}^n$, for any $i \leq j$, we set

$$L_{ij} = \begin{cases} 0, & i = j, \\ 0, & i < j, \end{cases} \qquad U_{ij} = \begin{cases} 0, & i = j, \\ M^2, & i < j. \end{cases} \tag{5.28}$$

where $M > 0$ is a large constance.

5.3.2 Data Generation

Three examples are introduced in this subsection, comprising data in $\mathbb{R}^r$ with $r = 2$ or 3. When $r = 2$, the problem is the so-called circle fitting problem that has recently been studied by Beck and Pan (2012) where more references on the topic can be found. Two circle fitting problems including the one considered by Beck and Pan (2012) and one with randomly generated data will be tested.

Example 5.7. (`HA30`, Bai et al. (2015)) *This dataset comprises spherical distances among $n = 30$ global cities $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, measured in hundreds of miles and selected by Hartigan (1975) from the World Almanac, 1966. It also provides XYZ coordinates of those cities, implying $r = 3$. Euclidean dissimilarities among those cities can be calculated through the formula:*

$$\delta_{ij} = 2R_a \sin(s_{ij}/(2R_a)), \tag{5.29}$$

where s_{ij} is the spherical distance between city i and city j, $R_a = 39.59$ (hundreds miles) is the Earth radius. We need emphasize here, the spherical distance s_{ij} is actually contaminated by noise, that is $\delta_{ij} \approx \|\boldsymbol{x}_i - \boldsymbol{x}_n\|$. To make such test example reasonable, we use the spherical distance s_{ij} to derive δ_{ij} (5.29) rather than using XYZ coordinates since the latter are accurate.

Example 5.8. (Circle fitting, Beck and Pan (2012)) *Let points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}\} \in \mathbb{R}^2$ be given. The problem is to find a circle with center $\boldsymbol{x}_n \in \mathbb{R}^2$ and radius R such that the* *points stay as close to the circle as possible. One criterion was considered by Beck and Pan (2012):*

$$\min_{\boldsymbol{x}_n, R} \sum_{i=1}^{n-1} (\|\boldsymbol{x}_i - \boldsymbol{x}_n\| - R)^2. \tag{5.30}$$

Beck gave a specific example (Beck and Pan, 2012, Example 5.3) with:

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

Then we just let $\delta_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$.

Example 5.9. *(Circle fitting with random data) We generate* $n-1$ *points* $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n-1}\}$ *on a circle with radius* 1 *and center in origin by:*

$$\boldsymbol{x}_i = \begin{bmatrix} \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}^\top \in \mathbb{R}^2$$

with $\theta_i, i = 1, \ldots, n-1$ *is generated from a uniform distribution on* $[0, 2\pi]$*. Then we add noise on the distance between each two points to make the problem more difficult:*

$$\delta_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \cdot |1 + \epsilon_{ij} \cdot \boldsymbol{nf}|, \quad i, j = 1, \ldots, n, \tag{5.31}$$

where $\boldsymbol{nf}$ *is the noise factor and* ϵ_{ij} *are independent standard normal random variables.*

Bases on Subsection 5.3.1, parameters $W, \Delta, L, U \in \mathbb{S}^n$ are given as in Table 5.4, where Δ is from (5.26), L, U are decided by (5.28) and M is a positive bounded constant, e.g., $M := n \max_{ij} \Delta_{ij}$ and R_o is the estimated radius, e.g., $R_o := \max_{ij} \|\mathbf{x}_i - \mathbf{x}_n\|/2$ for Examples 5.7 and 5.9, and $R_o := \max_{ij} \|\mathbf{x}_i - \mathbf{x}_n\|$ for Example 5.8.

Table 5.4: Parameter generation of ES problem.

(i,j)	W_{ij}	Δ_{ij}	L_{ij}	U_{ij}
$i = j$	0	0	0	0
$i, j = 1, ..., n-1$	1	δ_{ij}	0	M^2
$i = 1, ..., n-1, j = n$	1	R_o	0	M^2

5.4 Dimensionality Reduction

Nowadays, there are more and more large volumes of high-dimensional data including global climate patterns, stellar spectra, or human gene distributions regularly confronted us. To find meaningful low-dimensional structures hidden in their high-dimensional observations, known as dimensionality reduction (DR), becomes much more of importance, for the sake of easy visual perception or understanding.

5.4.1 Problematic Interpretation

Suppose there is a group of n images each of which has an $m_1 \times m_2 =: d$ pixel matrix. Denote $\mathbf{z}_i \in \mathbb{R}^d, i = 1, \ldots, n$ the vector formed by all columns of each pixel matrix. Clearly those vectors are in a space with a high dimension d, which seems to be impossible to visualize them by a graph. Fortunately, since these images are taken from one group, they potentially possess several common features or they are dominated by a few common features. One can refer to (Tenenbaum et al., 2000; Weinberger and Saul, 2006) for more details. Recall `Face698` data (see Subsection 1.2.4) where images of faces are categorized by three features: the different (up-down and left-right) face poses and different light directions, seen Subsection 1.2.4. In order to capture r features ($r = 2$ or 3) and thus to visualize images, a proper way is to find $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^r$ such that

$$\|\mathbf{x}_i - \mathbf{x}_j\| \approx \|\mathbf{z}_i - \mathbf{z}_j\|, \quad i = 1, \ldots, n. \tag{5.32}$$

These aim at preserving the local information among objects. For example, there are three images $\mathbf{z}_i, \mathbf{z}_j$ and $\mathbf{z}_k$ in which the first two are quite similar and the last two differ with each other a lot. This means $\|\mathbf{z}_i - \mathbf{z}_j\|$ is very small while $\|\mathbf{z}_j - \mathbf{z}_k\|$ is relatively large. Then $\mathbf{x}_i, \mathbf{x}_j$ and $\mathbf{x}_k$ such that (5.32), $\|\mathbf{x}_i - \mathbf{x}_j\| \approx \|\mathbf{z}_i - \mathbf{z}_j\|$ and $\|\mathbf{x}_j - \mathbf{x}_k\| \approx \|\mathbf{z}_j - \mathbf{z}_k\|$, are able to preserve the local information among these three images.

However in practice, not all pairwise distances $\|\mathbf{z}_i - \mathbf{z}_j\|$ are used. A common way to obtain pairwise distances is the **k-Nearest Neighbour rule** (k-NNR). More detailed, for each node $\mathbf{z}_i$, only k smallest distances among $\{\|\mathbf{z}_i - \mathbf{z}_j\|, j \neq i\}$ are kept. Denote N_{xx} a set formed by (i, j) if $\|\mathbf{z}_i - \mathbf{z}_j\|$ are kept by k-NNR. In order to guarantee the graph whose nodes are $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ and edges are $(i, j) \in N_{xx}$ being connected, k should be chosen carefully (not able to be too small). Then constraints (5.32) is altered as

$$\|\mathbf{x}_i - \mathbf{x}_j\| \approx \|\mathbf{z}_i - \mathbf{z}_j\|, \quad (i,j) \in N_{xx}. \tag{5.33}$$

From definition of EDM in Subsection 1.4.1, an information matrix $\Delta \in \mathbb{S}^n$ can be derived first, for $i \leq j$,

$$\Delta_{ij} = \begin{cases} \|\mathbf{z}_i - \mathbf{z}_n\|, & (i,j) \in N_{xx}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.34}$$

Overall, such problem is to find an EDM matrix D with embedding dimension r that is nearest to $\Delta^{(2)}$ and satisfies (5.33). By these constraints and (1.30), in other words, it aims at approximating $\Delta^{(2)}$ by D such that

$$-D \in \mathbb{S}^n_h \cap \mathbb{K}^n_+(r) \tag{5.35}$$

To derive the box constraints $L \leq D \leq U$ in (3.1) and $L, U \in \mathbb{S}^n$, for any $i \leq j$, we set

$$L_{ij} = \begin{cases} 0, & i = j, \\ 0, & i < j, \end{cases} \qquad U_{ij} = \begin{cases} 0, & i = j, \\ M^2, & i < j. \end{cases} \tag{5.36}$$

where $M > 0$ is a large constance.

5.4.2 Data Generation

Three real datasets which have been widely used in manifold learning (Tenenbaum et al., 2000; Weinberger and Saul, 2006) will be considered here.

Example 5.10. (`Teapot`) *This dataset comprises $n = 400$ images of a teapot taken from different angles by rotating the teapot 360 degrees. Each image has 76×101 pixels with 3 byte color depth i.e., $d = 76 \times 101 \times 3$. Two dimensional ($r = 2$) embedding will be considered in such example.*

Example 5.11. (`Face698`) *This dataset comprises $n = 698$ images (64×64 pixels) of faces with the different poses (up-down and left-right) and different light directions. Therefore, the embedding is naturally expected to lie in the two or three dimensional ($r = 2$ or 3) space parameterized by these major features.*

Example 5.12. (`Digit1`) *This dataset comprises $n = 1135$ images (28×28 pixels, i.e., $d = 28^2$) of digits "1" with the two important features: the slant and the line thickness. Therefore, the embedding is naturally expected to lie in the two dimensional ($r = 2$) space parameterized by these major features.*

Based on Subsection 5.4.1, let $\mathbf{z}_i \in \mathbb{R}^p, i = 1, \ldots, n$ be the vector generated from the pixel matrix of each image. Then by using k-NNR, we acquire N_{xx}. To make problems more of difficulty, we add some noise on the distances as

$$\delta_{ij} = \|\mathbf{z}_i - \mathbf{z}_n\| \cdot |1 + \epsilon_{ij} \cdot \mathtt{nf}|, \quad (i,j) \in N_{xx},$$

where `nf` is the noise factor and ϵ_{ij} are independent standard normal random variables. Finally, parameters $W, \Delta, L, U \in \mathbb{S}^n$ are given as in Table 5.5, where L, U are decided by (5.36) and M is a positive bounded constant, e.g., $M := n \max_{ij} \Delta_{ij}$.

Table 5.5: Parameter generation of DR problem.

(i,j)	W_{ij}	Δ_{ij}	L_{ij}	U_{ij}
$i = j$	0	0	0	0
$(i,j) \in N_{xx}$	1	δ_{ij}	0	M^2
$(i,j) \in \overline{N}_{xx}$	0	0	0	M^2

Chapter 6

Numerical Experiments

In this chapter, we illustrate how to implement Algorithm 4.1 proposed in Table 4.1. To emphasize ideas of majorization-Projection and EDM optimization, we name it `MPEDM`. We first design its stopping criteria and initialization. When it is introduced to process each application described in Chapter 5, the specific procedure is then summarized. Finally, we do self-comparison of `MPEDM` under each f_{pq} to see the effectiveness of each objective function, and compare `MPEDM` under f_{11} with other existing state-of-the-art methods to highlight its exceptional performance. All numerical experiments of our algorithm `MPEDM` is conducted by MATLAB (R2014a) on a desktop of 8GB memory and Inter(R) Core(TM) i5-4570 3.2Ghz CPU. Part of Matlab packages can be downloaded at *https://www.researchgate.net/profile/Shenglong_Zhou/publications* or *https://github.com/ShenglongZhou*.

6.1 Implementation

We first design the stopping criteria and initialization of `MPEDM`. Then performance measurement of method on testing examples from Chapter 5 is introduced, and finally the whole procedure to implement the algorithm on each example is summarized.

6.1.1 Stopping Criteria

We now consider the stopping criteria used in Step 3 to terminate Algorithm 4.1.

The `MPEDM` is easy to implement. We monitor two quantities. One is on how close of the current iterate D^k is to be Euclidean (belonging to $-\mathbb{K}^n_+(r)$). This can be computed by using (1.31) as follows.

$$\begin{aligned}
\texttt{Kprog}_k \quad &:= \quad \frac{2g(D^k)}{\|JD^kJ\|^2} = \frac{\|D^k + \Pi_{\mathbb{K}^n_+(r)}(-D^k)\|^2}{\|JD^kJ\|^2} \\
&= \quad \frac{\|\mathrm{PCA}_r^+(-JD^kJ) + (JD^kJ)\|^2}{\|JD^kJ\|^2} \\
&= \quad 1 - \frac{\sum_{i=1}^r \left[\lambda_i^2 - (\lambda_i - \max\{\lambda_i, 0\})^2\right]}{\lambda_1^2 + \ldots + \lambda_n^2} \le 1,
\end{aligned}$$

where $\lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_n$ are the eigenvalues of $(-JD^kJ)$. The smaller $\texttt{Kprog}_k$ is, the closer $-D^k$ is to $\mathbb{K}^n_+(r)$. The benefit of using `Kprog` over $g(D)$ is that the former is independent of any scaling of D.

The other quantity is to measure the progress in the functional values F_ρ by the current iterate D^k. In theory (see Thm. 4.6), we should require $\rho > \rho_o$, which can be found in Table 4.2 and is potentially large if one δ_{ij} is very small and $f = f_{11}$. As with the most penalty methods (Nocedal and Wright, 2006, Chp. 17), starting with a very large penalty parameter may degrade the performance of the method (e.g., causing air-conditionness). Therefore, for all f_{pq}, we uniformly adopt a dynamic updating rule for ρ. Let κ counts the number of non-zero elements of Δ. We choose $\rho_0 = \kappa n^{-3/2} \max \delta_{ij}$ and update it as

$$\rho_{k+1} = \begin{cases} 1.25\rho_k, & \text{if } \texttt{Kprog}_k > \texttt{Ktol}, \texttt{Fprog}_k \le 0.2\texttt{Ftol}, \\ 0.75\rho_k, & \text{if } \texttt{Fprog}_k > \texttt{Ftol}, \texttt{Kprog}_k \le 0.2\texttt{Ktol}, \\ \rho_k, & \text{otherwise}, \end{cases} \tag{6.1}$$

where

$$\texttt{Fprog}_k := \frac{F_{\rho_{k-1}}(D^{k-1}) - F_{\rho_{k-1}}(D^k)}{1 + \rho_{k-1} + F_{\rho_{k-1}}(D^{k-1})} \tag{6.2}$$

and `Ftol` and `Ktol` are chosen as

$$\texttt{Ftol} = \ln(\kappa) \times 10^{-4}, \qquad \texttt{Ktol} = \begin{cases} 10^{-2} & \text{if } n \ge 100, \\ 10^{-4} & \text{if } n < 100. \end{cases} \tag{6.3}$$

The rule for updating ρ_k seems to be complicated but works well for numerical experiments. Let us simply explain why we choose to update ρ_k as (6.1). We know the role of

ρ is to balance the $f(D)$ and $g(D)$. Therefore, if in a step $f(D^k)$ decreases sufficiently such as $\texttt{Fprog}_k \le 0.2\texttt{Ftol}$ while $g(D^k)$ is still not to be Euclidean, then ρ is suggested to be increase for next iteration. Or in a step $g(D^k)$ is to be almost Euclidean sufficiently such as $\texttt{Kprog}_k \le 0.2\texttt{Ktol}$ while $f(D^k)$ still violates the stopping criterion, then ρ is suggested to be reduced for next iteration. For other cases, there is no need to vary ρ.

Taking two quantities into consideration, we terminate `MPEDM` when

$$(\texttt{Fprog}_k \le \texttt{Ftol} \ \text{ and } \ \texttt{Kprog}_k \le \texttt{Ktol}) \quad \text{or} \quad k > 2000.$$

6.1.2 Initialization

Since the main problem (4.1) is non-convex, a good starting point D^0 would benefit for Algorithm 4.1. As mentioned in Chapter 5, each application renders an information matrix Δ with either some elements being unapproachable or all elements being obtained. A potential starting point one can utilize is $D^0 := \Delta^{(2)}$, because certain elements in Δ keep some useful information that we want to use. However, numerical experiments have demonstrated that when large amounts (e.g., over 80%) of elements of Δ are unavailable (and this phenomenon is quite common in practice), such choice of starting point would lead to a very poor performance of `MPEDM`. A possible reason is that when large amounts of elements of Δ are unavailable, Δ is far from a EDM which leads to $\Pi_{\mathbb{K}^n_+(r)}(\Delta^{(2)})$ a very bad initial point that approximates the true EDM.

An alternative is to keep using these known elements in Δ but replacing those missing dissimilarities by its shortest path distances. Namely, consider a graph with each vertex being each point/object and an edge being the known dissimilarity between two points. Since some of dissimilarities are missing, the graph has many edges unknown. Then we take advantage of the shortest path method to complete all missing edges by using shortest path distances. A MATLAB build-in function `graphallshortestpaths` to calculate the shortest path distances can be called to complete those missing distances. More detailed, the pseudo-Matlab code to initialize D^0 is as follows

$$D^0 = \begin{cases} \left(\texttt{graphallshortestpaths}(\texttt{sparse}(\Delta))\right)^{(2)}, & \kappa/n^2 \le 80\%, \quad (6.4a) \\ \Delta^{(2)}, & \text{otherwise}, \quad (6.4b) \end{cases}$$

where $\texttt{sparse}(\Delta)$ is the sparse version of Δ.

6.1.3 Measurements and Procedures

$\boxed{\text{SNL problems.}}$ This problem contains four examples: Examples 5.1-5.4. To see accuracy of embedding results of `MPEDM`, we adopt a widely used measure `RMSD` (Root of the Mean Squared Deviation) defined by

$$\texttt{RMSD} := \left[\frac{1}{n-m}\sum_{i=m+1}^{n}\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|^2\right]^{1/2},$$

where $\mathbf{x}_i$'s are the true positions of the sensors or atoms in our test problems and $\widehat{\mathbf{x}}_i$'s are their corresponding estimates. The $\widehat{\mathbf{x}}_i$'s were obtained by applying the classical MDS (see Table 2.1) method to the final output of the distance matrix, followed by aligning them to the existing anchors through the well-known Procrustes procedure (see Zhang et al. (2010), (Borg and Groenen, 2005, Chp. 20) or (Qi et al., 2013, Prop. 4.1) for more details). Furthermore, upon obtaining $\widehat{\mathbf{x}}_i$'s, a heuristic gradient method can be applied to improve their accuracy to further get $\widehat{\mathbf{x}}_i^{\text{ref}}$'s and it is called the refinement step in Biswas et al. (2006). We report `rRMSD` to highlight its contribution'

$$\texttt{rRMSD} := \left[\frac{1}{n-m}\sum_{i=m+1}^{n}\|\widehat{\mathbf{x}}_i^{\text{ref}} - \mathbf{x}_i\|^2\right]^{1/2}.$$

As we will see, if the ground truth $\mathbf{x}_i$s are known, our method benefits from this step for the most of problems because it could improve the finally embedding accuracy but may occur computational expense especially when n is very large. In addition, we record `rTime` (the time of refinement step) and the total cup time `Time` (including `rTime`) consumed by our proposed method to demonstrate its computational speed. Hereafter, the unit of all recorded time is second. Thus four indicators will be reported for this example, that is,

$$(\texttt{RMSD},\ \texttt{rRMSD},\ \texttt{Time},\ \texttt{rTime}).$$

The whole procedure for `MPEDM` to solve SNL problems is summarized in Table 6.1. More detailed about step three: **Sensors Recovery**, we only apply the Procrustes analysis on the known anchors $[\mathbf{a}_1 \cdots \mathbf{a}_m] =: Z$ and the recovered anchors $[\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_m] =: X$. Recall Subsection 1.4.3, we can derive $\mathbf{z}_c, \mathbf{x}_c$ and P^* which further get

$$[\widehat{\mathbf{x}}_{m+1}, \cdots, \widehat{\mathbf{x}}_n] = P^*[\overline{\mathbf{x}}_{m+1} - \mathbf{x}_c \cdots \overline{\mathbf{x}}_n - \mathbf{x}_c] + \mathbf{z}_c.$$

Table 6.1: `MPEDM` for SNL problems

Initialization	Set the dissimilarities matrix Δ, initialize D^0 by (6.4);
EDM Reconstruction	Solve `MPEDM` in Algorithm 4.1 to get a closed EDM $\overline{D}$.
Sensors Recovery	Apply cMDS in Table 2.1 on $\overline{D}$ to get embedding points $X := [\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_n]$ in $\mathbb{R}^2$. Then apply Procrustes analysis on the embedding points $[\overline{\mathbf{x}}_{m+1} \cdots \overline{\mathbf{x}}_n]$ by only using the known anchors $[\mathbf{a}_1 \cdots \mathbf{a}_m]$ to get new sensors $[\widehat{\mathbf{x}}_{m+1}, \cdots, \widehat{\mathbf{x}}_n]$.
Refinement	Apply the gradient descent method on $[\widehat{\mathbf{x}}_{m+1}, \cdots, \widehat{\mathbf{x}}_n]$ to further get refined sensors $[\widehat{\mathbf{x}}^{\text{ref}}_{m+1}, \cdots, \widehat{\mathbf{x}}^{\text{ref}}_n]$.

Table 6.2: `MPEDM` for MC problems

Initialization	Set the dissimilarities matrix Δ, initialize D^0 by (6.4);
EDM Reconstruction	Solve `MPEDM` in Algorithm 4.1 to get a closed EDM $\overline{D}$.
Atoms Recovery	Apply cMDS in Table 2.1 on $\overline{D}$ to get embedding points $X := [\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_n]$ in $\mathbb{R}^3$. Then apply Procrustes analysis on $[\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_n]$ by using the ground truth atoms $[\mathbf{x}_1 \cdots \mathbf{x}_n]$ to get new atoms $[\widehat{\mathbf{x}}_1, \cdots, \widehat{\mathbf{x}}_n]$.
Refinement	Apply the gradient descent method on $[\widehat{\mathbf{x}}_1, \cdots, \widehat{\mathbf{x}}_n]$ to further get refined atoms $[\widehat{\mathbf{x}}^{\text{ref}}_1, \cdots, \widehat{\mathbf{x}}^{\text{ref}}_n]$.

$\boxed{\text{MC problems.}}$ This problem contains two examples: Examples 5.5 and 5.6, in which no atoms are given in advance. We still utilize four indicators (`RMSD`, `rRMSD`, `Time`, `rTime`) to highlight the performance of `MPEDM`. The whole procedure for `MPEDM` to solve MC problems is summarized in Table 6.2.

More detailed about step three: **Atoms Recovery**, since no atoms are given, we apply the Procrustes analysis on the ground truth atoms $[\mathbf{x}_1 \cdots \mathbf{x}_n] =: Z$ and the recovered anchors $[\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_n] =: X$. Recall Subsection 1.4.3, we can derive $\mathbf{z}_c, \mathbf{x}_c$ and P^* which further get

$$[\widehat{\mathbf{x}}_1, \cdots, \widehat{\mathbf{x}}_n] = P^*[\overline{\mathbf{x}}_1 - \mathbf{x}_c \cdots \overline{\mathbf{x}}_n - \mathbf{x}_c] + \mathbf{z}_c.$$

$\boxed{\text{ES problems.}}$ This problem contains three examples: Examples 5.7-5.9. To highlight the goodness of a found sphere fitting the given points, we define the fitness of embedding to a sphere (FES) as

$$\texttt{FES} := \sum_{i=1}^{n-1} (\|\mathbf{x}_i - \mathbf{x}_c\| - R_{\text{est}})^2,$$

where R_{est} and $\mathbf{x}_c$ are the estimated radius and center of the found sphere. The smaller FES is, the better the estimated sphere fits the given points. This is actually the optimal objective function value of (5.30). Therefore, we would like to report (RMSD, FES, R_{est}) to demonstrate the performance of MPEDM.

Table 6.3: MPEDM for ES problems

Initialization	Set the dissimilarities matrix Δ, initialize D^0 by (6.4);
EDM Reconstruction	Solve MPEDM in Algorithm 4.1 to get a closed EDM $\overline{D}$.
Points Recovery	Apply cMDS in Table 2.1 on $\overline{D}$ to get embedding points $X := [\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_n]$ in $\mathbb{R}^r$. Then apply Procrustes analysis on $[\overline{\mathbf{x}}_1 \cdots \overline{\mathbf{x}}_{n-1}]$ by using the ground truth points $[\mathbf{x}_1 \cdots \mathbf{x}_{n-1}]$ to get new points $[\widehat{\mathbf{x}}_1 \cdots \widehat{\mathbf{x}}_{n-1}]$.
Sphere fitting	Find center $\mathbf{x}_c$ and radius R_{est} of a fitted sphere for points $[\widehat{\mathbf{x}}_1 \cdots \widehat{\mathbf{x}}_{n-1}]$.

The whole procedure for MPEDM to solve ES problems is summarized in Table 6.3, in which in the last step Sphere fitting, there are two methods to find a sphere, which are stated as below. We will take advantage of first way to find the sphere since it would render us more accurate results.

- Use MATLAB solver `sphereFit`[1] to find a sphere by pseudo MATLAB code:

$$[\mathbf{x}_c, R_{\text{est}}] = \texttt{sphereFit}\Big([\widehat{\mathbf{x}}_1 \cdots \widehat{\mathbf{x}}_{n-1}]\Big).$$

[1] `sphereFit` is available at: *https://uk.mathworks.com/matlabcentral/fileexchange/34129-sphere-fit–least-squared-*

and `circfit`[2] to find a circle by pseudo MATLAB code:

$$\Big[\mathbf{x}_{c1}, \mathbf{x}_{c2}, R_{\text{est}}\Big] = \texttt{circfit}\Big([\widehat{\mathbf{x}}_{11} \cdots \widehat{\mathbf{x}}_{n-1,1}], [\widehat{\mathbf{x}}_{12} \cdots \widehat{\mathbf{x}}_{n-1,2}]\Big),$$

where $\mathbf{x}_c = [\mathbf{x}_{c1}\ \mathbf{x}_{c2}]^\top$, $\widehat{\mathbf{x}}_i = [\widehat{\mathbf{x}}_{i1}\ \widehat{\mathbf{x}}_{i2}]^\top, i = 1, \ldots, n-1$.

- Compute $\mathbf{x}_c := \overline{\mathbf{x}}_n$ and
$$R_{\text{est}} = \frac{1}{n-1}\sum_{i=1}^{n-1} \overline{D}_{in}.$$

$\boxed{\text{DR problems.}}$ This problem contains three examples: Examples 5.10-5.12. The whole procedure for `MPEDM` to solve such problems is summarized in Table 6.4.

Table 6.4: `MPEDM` for DR problems

Initialization	Set the dissimilarities matrix Δ, initialize D^0 by (6.4);
EDM Reconstruction	Solve `MPEDM` in Algorithm 4.1 to get a closed EDM $\overline{D}$.
Points Recovery	Apply cMDS in Table 2.1 on $\overline{D}$ to get embedding points $X := [\widehat{\mathbf{x}}_1 \cdots \widehat{\mathbf{x}}_n]$ in $\mathbb{R}^r$.

To emphasize the quality of reduction of dimensionality, we will compute `EigScore`(r) associated with the eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ of $(-J\overline{D}^{(2)}J/2)$ as

$$\texttt{EigScore}(r) := \frac{\lambda_1 + \cdots + \lambda_r}{|\lambda_1| + \cdots + |\lambda_n|}.$$

Clearly, $0 \leq \texttt{EigScore}(r) \leq 1$. The closer to 1 `EigScore`$(r)$ is, the better the dimensionality is reduced to r. We also calculate the relative error that is able to measure the preservation of local distance (`PRE`) as

$$\texttt{PRE} := \frac{\sum_{(i,j)\in N_{xx}} \left(\overline{D}_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|\right)^2}{\sum_{(i,j)\in N_{xx}} \|\mathbf{z}_i - \mathbf{z}_j\|^2}.$$

[2] `circfit` is available at: *https://uk.mathworks.com/matlabcentral/fileexchange/5557-circle-fit*

6.2 Numerical Comparison among f_{pq}

In this section, we will conduct extensive numerical simulations of our proposed method `MPEDM` which is associated with four objective functions. For simplicity, we write $\texttt{MPEDM}_{pq}$ $(p, q = 1, 2)$ to denote `MPEDM` under each f_{pq}.

6.2.1 Test on SNL

$\boxed{\text{Test on Example 5.1.}}$ We first demonstrate the performance of `MPEDM` under each f to the radio range R by fixing $n = 200, m = 4$, $\texttt{nf}= 0.1$ and altering R among $\{0.2, 0.4, \cdots, 1.4\}$. Average results were demonstrated in Figure 6.1 in which there was no big difference of `rRMSD`. Clearly, $\texttt{MPEDM}_{22}$ got the worst `RMSD` in most cases.

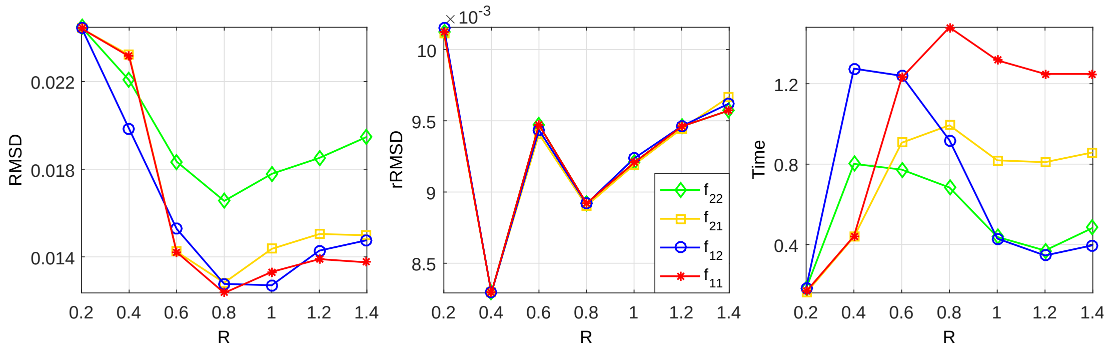


Figure 6.1: Example 5.1 with $n = 200, m = 4, \texttt{nf} = 0.1$.

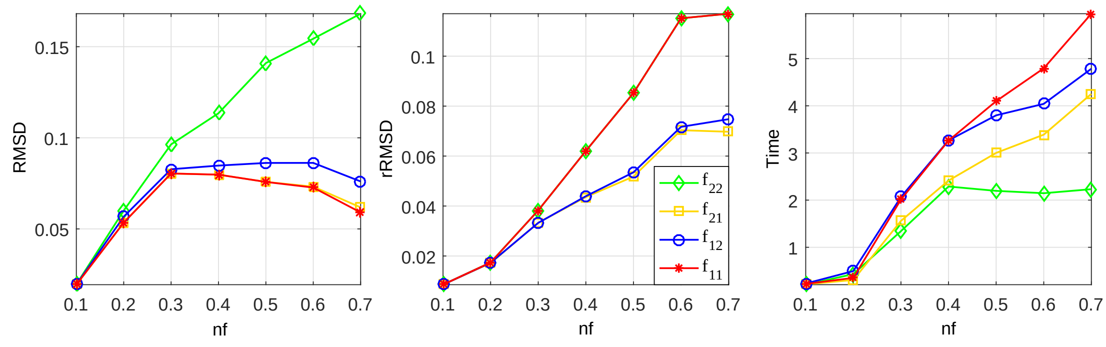


Figure 6.2: Example 5.1 with $n = 200, m = 4, R = 0.3$.

We then demonstrate the performance of $\texttt{MPEDM}_{pq}$ to the noise factor `nf` by fixing $n = 200, m = 4, R = 0.3$, and altering `nf` among $\{0.1, 0.2, \cdots, 0.7\}$. Average results were presented in Figure 6.2. Clearly, $\texttt{MPEDM}_{11}$ got the best `RMSD`, followed by $\texttt{MPEDM}_{21}$, which means they two were more robust to the noise factor due to the use of ℓ_1 norm. By contrast, $\texttt{MPEDM}_{22}$ rendered the worst `RMSD` but ran the fastest. Interestingly, one may notice that `RMSD` generated by $\texttt{MPEDM}_{11}$ was bigger than `rRMSD` when $\texttt{nf} \geq 0.5$, indicating the refinement step made the localization accuracy of $\texttt{MPEDM}_{11}$ worse.

Table 6.5: Example 5.1 with $m = 4, R = 0.2, \texttt{nf} = 0.1$.

n		1000	2000	3000	4000	5000
`RMSD`	$\texttt{MPEDM}_{22}$	1.23e-2	1.06e-2	1.01e-2	9.61e-3	9.73e-3
	$\texttt{MPEDM}_{21}$	1.22e-2	1.07e-2	1.02e-2	9.72e-3	9.82e-3
	$\texttt{MPEDM}_{12}$	1.23e-2	1.06e-2	1.01e-2	9.59e-3	9.71e-3
	$\texttt{MPEDM}_{11}$	1.23e-2	1.07e-2	1.04e-2	9.87e-3	9.93e-3
`rRMSD`	$\texttt{MPEDM}_{22}$	3.39e-3	3.57e-3	4.71e-3	4.21e-3	2.99e-3
	$\texttt{MPEDM}_{21}$	3.40e-3	3.58e-3	4.79e-3	4.29e-3	3.02e-3
	$\texttt{MPEDM}_{12}$	3.40e-3	3.52e-3	4.51e-3	4.22e-3	2.99e-3
	$\texttt{MPEDM}_{11}$	3.39e-3	3.57e-3	4.71e-3	4.21e-3	2.99e-3
`Time`	$\texttt{MPEDM}_{22}$	4.93	17.46	52.42	104.61	211.16
	$\texttt{MPEDM}_{21}$	5.20	17.06	51.55	103.05	215.97
	$\texttt{MPEDM}_{12}$	5.78	18.77	59.38	111.46	227.30
	$\texttt{MPEDM}_{11}$	4.91	17.93	50.91	104.34	212.65
`rTime`	$\texttt{MPEDM}_{22}$	3.12	3.74	11.22	8.78	47.14
	$\texttt{MPEDM}_{21}$	5.65	2.88	10.49	13.09	43.53
	$\texttt{MPEDM}_{12}$	5.86	3.77	11.58	8.39	44.68
	$\texttt{MPEDM}_{11}$	3.20	3.27	10.58	8.52	43.68

Finally, we test this example with much larger sizes $n \in \{1000, 2000, \ldots, 5000\}$ and fixing $m = 4, R = 0.2, \texttt{nf} = 0.1$. Average results were recorded in Table 6.5. It can be clearly observed that four objective functions made $\texttt{MPEDM}_{pq}$ generated similar results, which was probably because the small noise factor $\texttt{nf} = 0.1$ added. In addition, along with ascending of n, `RMSD` tended to be better. The reason of such phenomenon was that the network became much denser when n increasing since all points were generated in a unit region.

$\boxed{\text{Test on Example 5.2.}}$ We first demonstrate the performance of `MPEDM` under each f to the radio range R by fixing $n = 200, m = 4$, $\texttt{nf} = 0.1$ and altering R among $\{0.2, 0.4, \cdots, 1.4\}$. Average results were demonstrated in Figure 6.3. It can be seen that there was no big different of `rRMSD`. Obviously, $\texttt{MPEDM}_{22}$ still got the worse `RMSD` but ran the fastest in most cases.

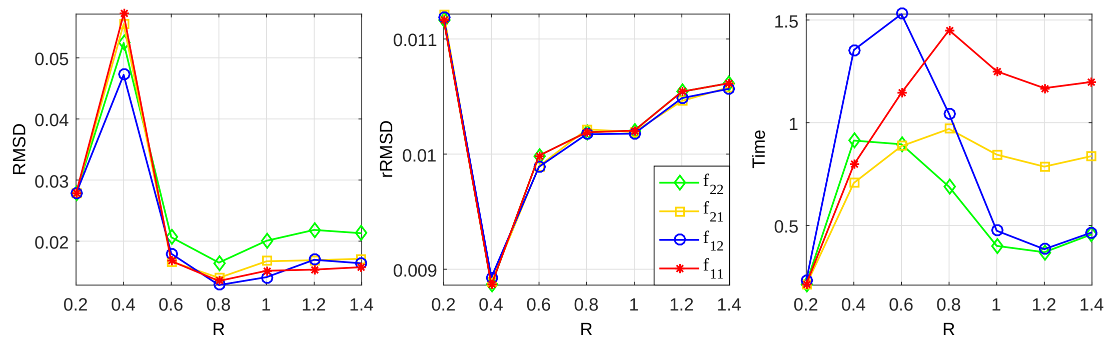


Figure 6.3: Example 5.2 with $n = 200, m = 4, \texttt{nf} = 0.1$.

We then demonstrate the performance of our method under each f to the noise factor `nf` by fixing $n = 200, m = 4, R = 0.3$, and altering `nf` among $\{0.1, 0.2, \cdots, 0.7\}$. Average results were demonstrated in Figure 6.4. Clearly, $\texttt{MPEDM}_{11}$ got the best `RMSD`, followed by $\texttt{MPEDM}_{21}$, again indicating ℓ_1 norm were more robust to the noise factor. By contrast, $\texttt{MPEDM}_{22}$ rendered the worst `RMSD`. After refinement, all of them produced similar `rRMSD`. In terms of computational speed, $\texttt{MPEDM}_{22}$ ran the fastest indeed, followed by $\texttt{MPEDM}_{21}$ and $\texttt{MPEDM}_{12}$, and $\texttt{MPEDM}_{11}$ came the last.

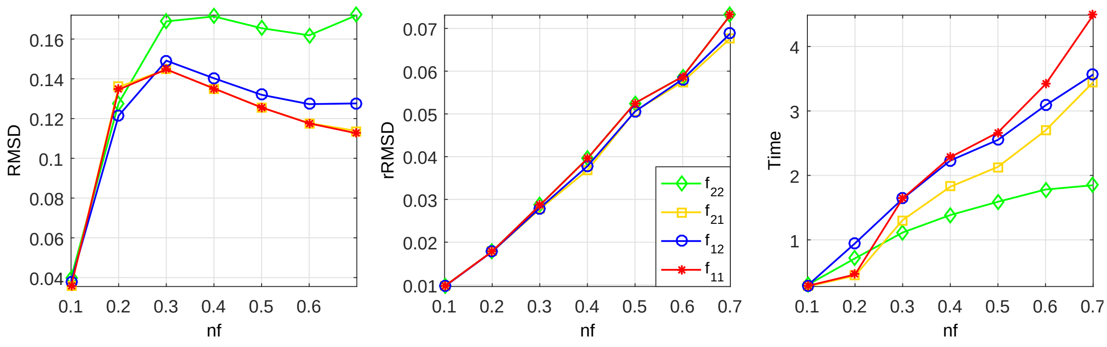


Figure 6.4: Example 5.2 with $n = 200, m = 4, R = 0.3$.

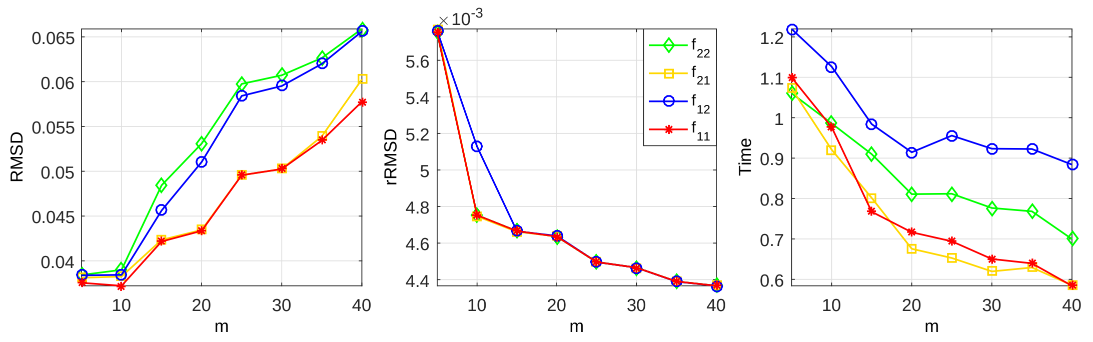


Figure 6.5: Example 5.3 with $n = 200, m = 4, R = 0.2$.

$\boxed{\text{Test on Example 5.3.}}$ This example has randomly size anchors, thus we demonstrate the performance of `MPEDM` under each f to the anchors number m by fixing $n = 200, R = 0.2$, `nf`$= 0.1$ and altering m among $\{5, 10, \cdots, 40\}$. Average results were shown in Figure 6.5. It can be seen that $\texttt{MPEDM}_{11}$ and $\texttt{MPEDM}_{21}$ out performed the other two both in terms of `RMSD` and `Time`. But after refinement, there was no big difference of `rRMSD`.

We then plot the embedding of $\texttt{MPEDM}_{pq}$ associated with the noise factor by choosing `nf` from $\{0.3, 0.5, 0.7, 0.9\}$ and fixing $n = 200, m = 10, R = 0.3$ in Figure 6.6, where 10 anchors were plotted in green squares and $\widehat{\mathbf{x}}_i^{\text{ref}}$ in pink points were jointed to its ground truth locations (blue circles). No big difference when `nf`≤ 0.5. However, when `nf` got bigger, $\texttt{MPEDM}_{11}$ and $\texttt{MPEDM}_{21}$ achieved the best `rRMSD`, followed by $\texttt{MPEDM}_{12}$. And apparently $\texttt{MPEDM}_{22}$ failed to locate when `nf`$= 0.9$.

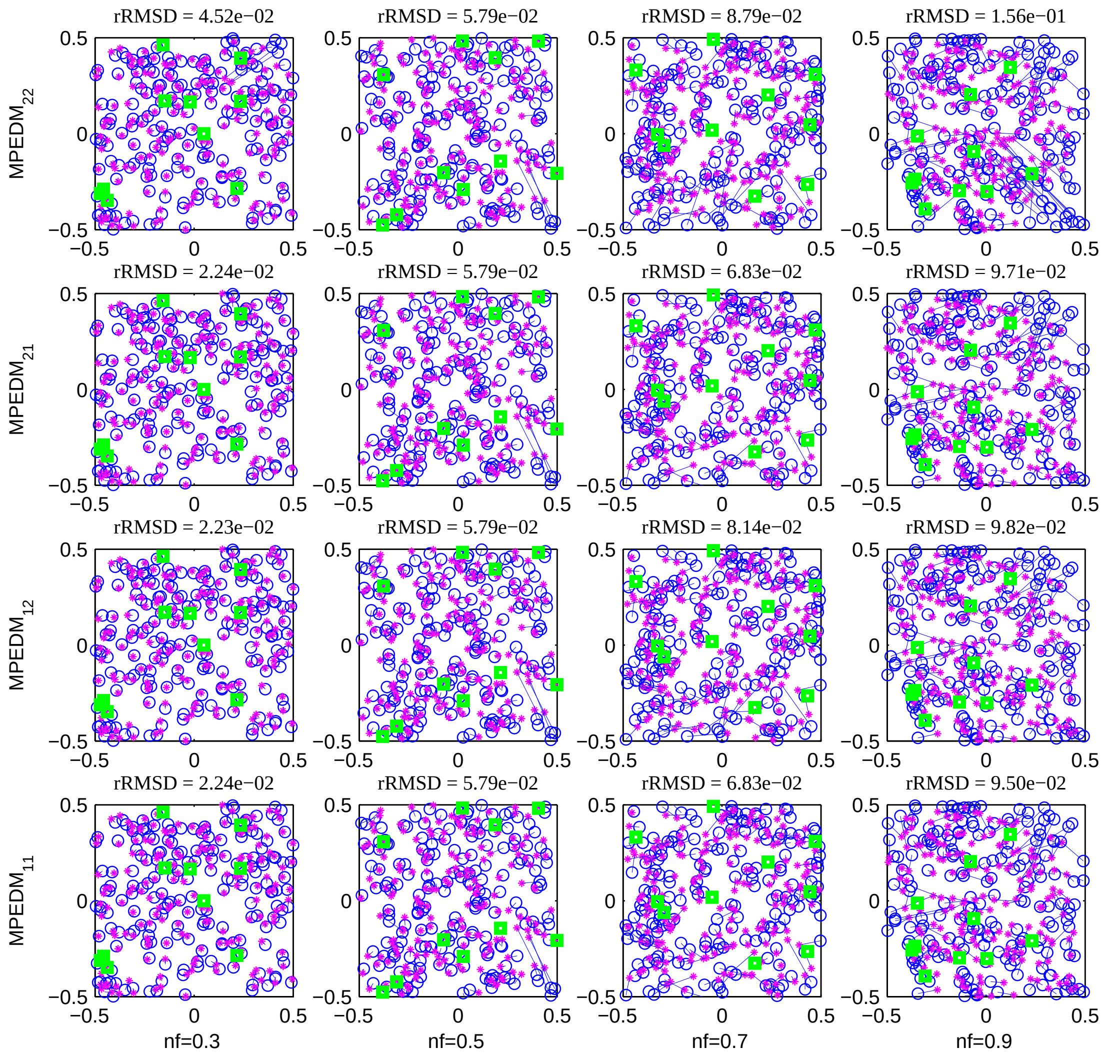


Figure 6.6: Example 5.3 with $n = 200, m = 10, R = 0.3$.

$\boxed{\text{Test on Example 5.4.}}$ We first demonstrate the performance of `MPEDM`$_{pq}$ to the noise factor `nf` by fixing $n = 200, m = 10, R = 0.3$, and altering `nf` among $\{0.1, 0.2, \cdots, 0.7\}$. Average results were demonstrated in Figure 6.7. Clearly, `MPEDM`$_{11}$ got the best `RMSD`, followed by `MPEDM`$_{21}$. By contrast, `MPEDM`$_{22}$ rendered the worst `RMSD` but ran the fastest. Moreover, `MPEDM`$_{12}$ and `MPEDM`$_{21}$ benefited a lot from the refinement since they two rendered the best `rRMSD`. And `MPEDM`$_{11}$ benefited less from the refinement along with `nf` increasing. Interestingly, when `nf` got bigger, `RMSD` was smaller. For example, `MPEDM`$_{11}$ yielded better `RMSD` when `nf`≥ 0.5 than those when $0.2 \leq$ `nf` < 0.5.

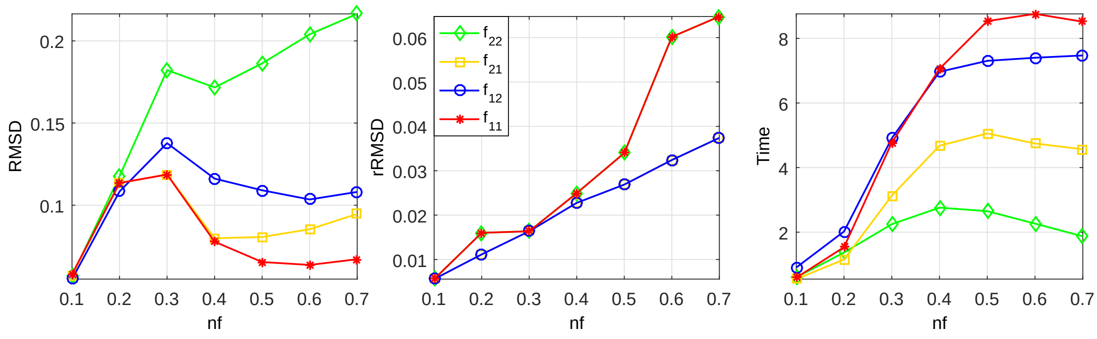


Figure 6.7: Example 5.4 with $n = 200, m = 10, R = 0.3$.

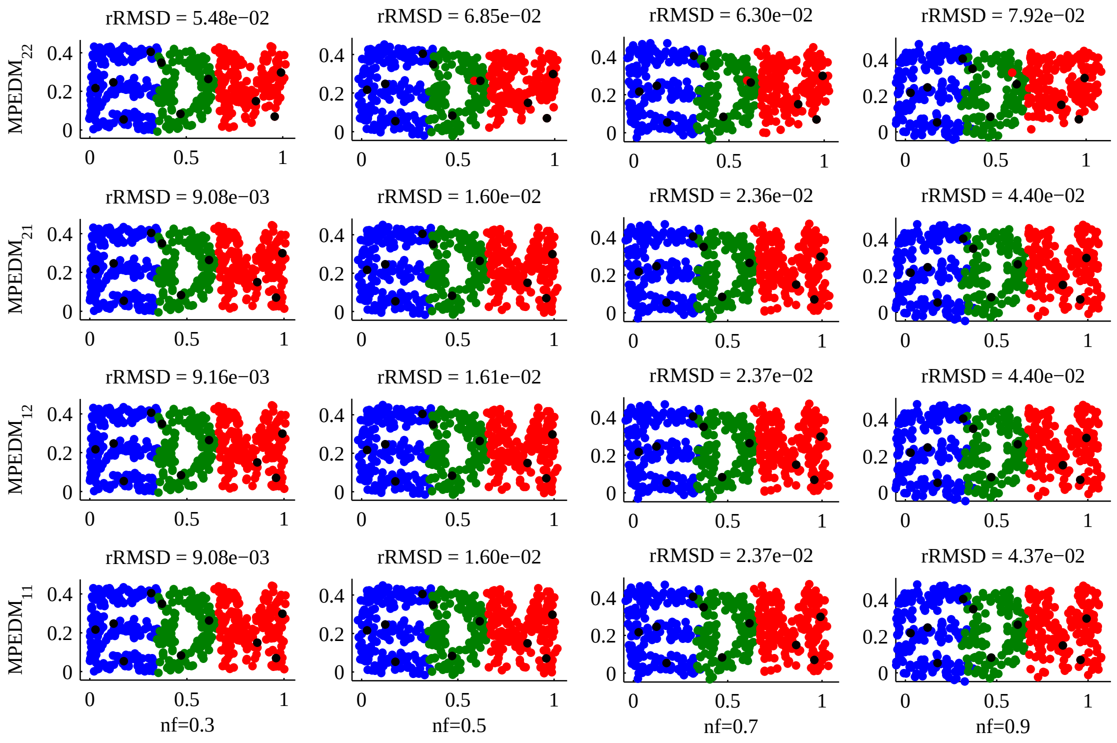


Figure 6.8: Example 5.4 with $n = 500, m = 10, R = 0.3$.

We then plot the embedding of $\texttt{MPEDM}_{pq}$ with fixing $n = 500, m = 10, R = 0.3$ but varying `nf` from $\{0.3, 0.5, 0.7, 0.9\}$ in Figure 6.8. Clearly, all methods except for $\texttt{MPEDM}_{22}$ were capable of capturing the shape of three letters. By contrast, the shape of letter 'M' obtained by $\texttt{MPEDM}_{22}$ was deformed more heavily as the rising of `nf`.

Finally, we test this example with much larger sizes $n \in \{1000, 2000, \ldots, 5000\}$ and fixing $m = 10, R = 0.1, \texttt{nf} = 0.1$. Average results were recoded in Table 6.6. Obviously, four objective functions made `MPEDM` generated similar results. One may notice that the refinement almost took over half of the total `Time`, which implies it is of computational inefficiency for such example.

Table 6.6: Example 5.4 with $m = 10, R = 0.1, \texttt{nf} = 0.1$.

n		1000	2000	3000	4000	5000
`RMSD`	$\texttt{MPEDM}_{22}$	5.01e-2	5.65e-2	8.00e-2	6.44e-2	8.50e-2
	$\texttt{MPEDM}_{21}$	5.01e-2	5.63e-2	8.00e-2	6.45e-2	8.48e-2
	$\texttt{MPEDM}_{12}$	5.01e-2	5.64e-2	8.00e-2	6.45e-2	8.48e-2
	$\texttt{MPEDM}_{11}$	4.99e-2	5.61e-2	8.00e-2	6.45e-2	8.48e-2
`rRMSD`	$\texttt{MPEDM}_{22}$	2.43e-3	5.02e-3	1.72e-2	6.31e-3	1.32e-2
	$\texttt{MPEDM}_{21}$	2.44e-3	4.87e-3	1.54e-2	6.28e-3	1.12e-2
	$\texttt{MPEDM}_{12}$	2.50e-3	5.81e-3	1.55e-2	5.96e-3	1.53e-2
	$\texttt{MPEDM}_{11}$	2.43e-3	5.02e-3	1.72e-2	6.31e-3	1.32e-2
`Time`	$\texttt{MPEDM}_{22}$	5.30	32.31	89.22	177.64	278.38
	$\texttt{MPEDM}_{21}$	5.27	32.17	91.31	171.68	277.09
	$\texttt{MPEDM}_{12}$	5.53	32.99	95.27	181.76	266.44
	$\texttt{MPEDM}_{11}$	5.24	29.48	90.25	174.19	279.84
`rTime`	$\texttt{MPEDM}_{22}$	3.67	20.89	53.19	98.57	129.62
	$\texttt{MPEDM}_{21}$	3.63	21.25	57.67	98.47	139.78
	$\texttt{MPEDM}_{12}$	3.57	20.28	57.56	101.41	117.78
	$\texttt{MPEDM}_{11}$	3.69	18.55	56.77	100.94	142.37

6.2.2 Test on MC

$\boxed{\text{Test on Example 5.5.}}$ This example has two rules (5.22) and (5.23) to define N_{xx}. We will demonstrate the performance of $\texttt{MPEDM}_{pq}$ to them respectively.

Under Rule 1 To see the effect of range R, we fix $s = 6$, `nf`$= 0.1$ and alter R among $\{2, 3, \ldots, 8\}$. Average results were shown in Figure 6.9 in which there was no big difference of `rRMSD`. Basically, $\texttt{MPEDM}_{11}$ got the best `RMSD`, followed by $\texttt{MPEDM}_{12}$, $\texttt{MPEDM}_{21}$ and $\texttt{MPEDM}_{22}$. What is more, $\texttt{MPEDM}_{22}$ and $\texttt{MPEDM}_{21}$ ran faster than the rest two.

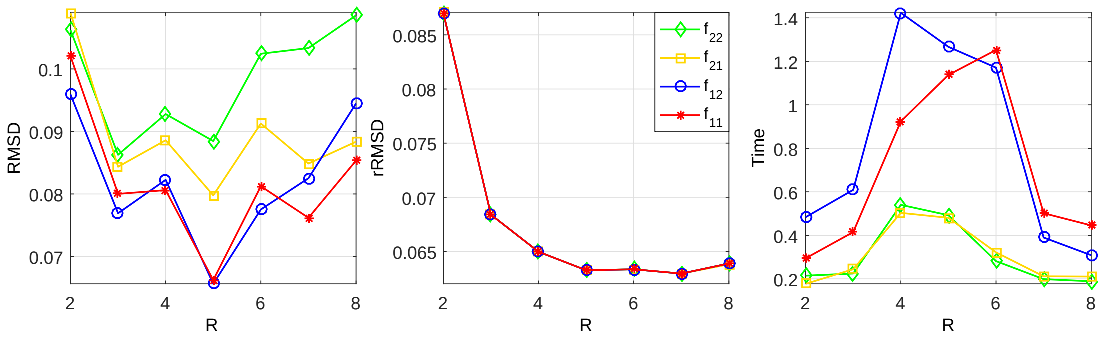


Figure 6.9: Example 5.5 under Rule 1 with $s = 6, \texttt{nf} = 0.1$.

To see the effect of noise factor `nf`, we fix $s = 6$, $R = 3$ and alter `nf` among $\in \{0.1, 0.2, \ldots, 0.5\}$. Results were demonstrated in Figure 6.10 in which there was no big difference of `rRMSD`. Apparently, $\texttt{MPEDM}_{12}$ got the best `RMSD`, followed by $\texttt{MPEDM}_{11}$, $\texttt{MPEDM}_{21}$ and $\texttt{MPEDM}_{22}$. Obviously, $\texttt{MPEDM}_{22}$ and $\texttt{MPEDM}_{21}$ ran faster than the rest two.

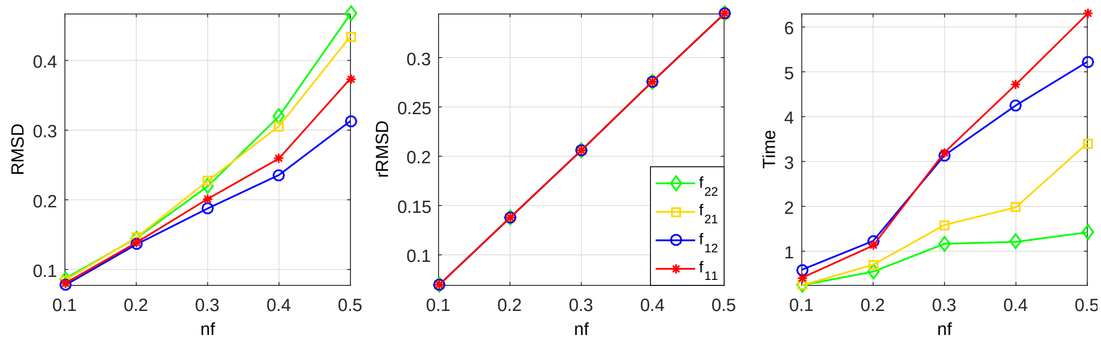


Figure 6.10: Example 5.5 under Rule 1 with $s = 6, R = 3$.

To see the effect of larger size problems, we fix $R = 3, \texttt{nf} = 0.1$ and change s from $\{10, 12, \ldots, 18\}$ corresponding to $n \in \{1000, 1728, \ldots, 5832\}$. Clearly, as reported in Table 6.7, results under each f were not big of difference. Most importantly, our proposed method `MPEDM` ran very fast even for large relatively large size problem, e.g., consuming about 100 seconds when $n = 18^3 = 5832$. What is more, `RMSD` and `rRMSD` got bigger along with the ascending of s, this was because more and more dissimilarities in Δ were unavailable since $R = 3$ fixed, yielding such example more of difficulty.

Table 6.7: Example 5.5 under Rule 1 with $R = 3, \texttt{nf} = 0.1$.

s		10	12	14	16	18
RMSD	$\texttt{MPEDM}_{22}$	1.22e-1	1.33e-1	1.45e-1	1.54e-1	1.62e-1
	$\texttt{MPEDM}_{21}$	1.25e-1	1.36e-1	1.48e-1	1.56e-1	1.64e-1
	$\texttt{MPEDM}_{12}$	1.23e-1	1.35e-1	1.48e-1	1.56e-1	1.64e-1
	$\texttt{MPEDM}_{11}$	1.25e-1	1.36e-1	1.48e-1	1.56e-1	1.64e-1
rRMSD	$\texttt{MPEDM}_{22}$	6.06e-2	5.88e-2	5.82e-2	5.67e-2	5.61e-2
	$\texttt{MPEDM}_{21}$	6.06e-2	5.88e-2	5.81e-2	5.67e-2	5.62e-2
	$\texttt{MPEDM}_{12}$	6.06e-2	5.88e-2	5.81e-2	5.67e-2	5.62e-2
	$\texttt{MPEDM}_{11}$	6.06e-2	5.88e-2	5.82e-2	5.67e-2	5.61e-2
Time	$\texttt{MPEDM}_{22}$	2.30	7.22	18.49	40.40	119.81
	$\texttt{MPEDM}_{21}$	2.03	6.54	16.72	37.05	96.50
	$\texttt{MPEDM}_{12}$	2.42	7.39	17.67	39.37	103.33
	$\texttt{MPEDM}_{11}$	2.13	6.70	16.85	37.92	97.42
rTime	$\texttt{MPEDM}_{22}$	0.34	1.12	1.88	4.27	14.59
	$\texttt{MPEDM}_{21}$	0.32	0.96	2.10	4.25	6.24
	$\texttt{MPEDM}_{12}$	0.33	0.97	2.11	4.22	6.32
	$\texttt{MPEDM}_{11}$	0.34	0.97	2.07	4.24	6.68

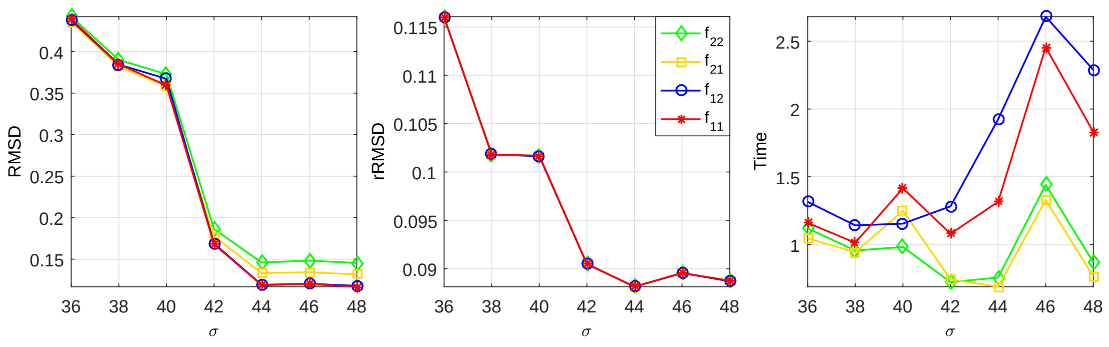


Figure 6.11: Example 5.5 under Rule 2 with $s = 6, \texttt{nf} = 0.1$.

Under Rule 2 To see the effect of σ, we fix $s = 6$, $\texttt{nf} = 0.1$ and vary σ among $\{36, 38, \cdots, 48\}$, similar observations to Rule 1 can be seen in Figure 6.11. Namely, $\texttt{MPEDM}_{11}$ and $\texttt{MPEDM}_{12}$ got the best RMSD, followed by $\texttt{MPEDM}_{21}$ and $\texttt{MPEDM}_{22}$. In terms of computational speed, $\texttt{MPEDM}_{22}$ and $\texttt{MPEDM}_{21}$ ran faster than the rest two. Notice that the percentage of available dissimilarities over all elements of Δ ascended from 32.47% to 39.87% along with increasing σ from 36 to 48, making problems more and more 'easier'. This would explain the generated RMSD became smaller as σ increased.

To see the effect of noise factor `nf`, we fix $s = 6$, $\sigma = 36$ and choose $\texttt{nf} \in \{0.1, 0.2, \cdots, 0.5\}$. Average results were presented in Figure 6.12. Apparently, $\texttt{MPEDM}_{11}$ got the best `RMSD`, followed by $\texttt{MPEDM}_{12}$ and $\texttt{MPEDM}_{21}$. Again, $\texttt{MPEDM}_{22}$ rendered the worst `RMSD`. For computational time, $\texttt{MPEDM}_{22}$ ran the fastest whilst $\texttt{MPEDM}_{11}$ came the last.

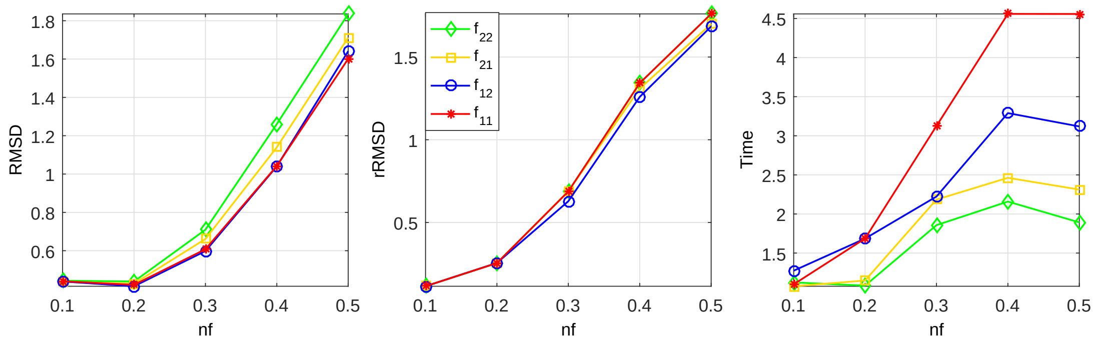


Figure 6.12: Example 5.5 under Rule 2 with $s = 6, \sigma = 36$.

Table 6.8: Example 5.5 under Rule 2 with $\sigma = s^2, \texttt{nf} = 0.1$.

s		10	12	14	16	18
RMSD	$\texttt{MPEDM}_{22}$	7.71e-1	9.24e-1	1.07e+0	1.21e+0	1.35e+0
	$\texttt{MPEDM}_{21}$	7.65e-1	9.21e-1	1.07e+0	1.21e+0	1.34e+0
	$\texttt{MPEDM}_{12}$	7.69e-1	9.25e-1	1.07e+0	1.22e+0	1.35e+0
	$\texttt{MPEDM}_{11}$	7.70e-1	9.23e-1	1.07e+0	1.21e+0	1.34e+0
rRMSD	$\texttt{MPEDM}_{22}$	2.90e-1	3.90e-1	4.78e-1	5.41e-1	6.23e-1
	$\texttt{MPEDM}_{21}$	2.77e-1	4.00e-1	4.75e-1	5.53e-1	6.08e-1
	$\texttt{MPEDM}_{12}$	3.38e-1	3.56e-1	4.87e-1	5.57e-1	6.67e-1
	$\texttt{MPEDM}_{11}$	2.90e-1	3.90e-1	4.78e-1	5.41e-1	6.23e-1
Time	$\texttt{MPEDM}_{22}$	9.48	37.06	92.96	284.15	527.80
	$\texttt{MPEDM}_{21}$	8.79	34.64	90.96	265.07	504.29
	$\texttt{MPEDM}_{12}$	10.17	52.49	104.39	299.92	563.25
	$\texttt{MPEDM}_{11}$	10.47	42.54	96.82	285.88	534.52
rTime	$\texttt{MPEDM}_{22}$	4.64	20.38	37.39	134.76	137.45
	$\texttt{MPEDM}_{21}$	4.36	18.75	38.64	123.17	150.21
	$\texttt{MPEDM}_{12}$	2.79	29.81	36.40	124.78	134.18
	$\texttt{MPEDM}_{11}$	5.37	24.77	39.60	134.57	152.16

Finally, we test this example with much larger sizes $s \in \{10, 12, \ldots, 18\}$. By fixing $\sigma = s^2, \texttt{nf} = 0.1$, we recoded average results in Table 6.8. Still, four objective functions made $\texttt{MPEDM}_{pq}$ generated similar results. What is more, `RMSD` and `rRMSD` got bigger along with the ascending of s, this was because the percentage of available dissimilarities over all elements of Δ declined from 8% to 2% when σ was increased from 10 to 18, yielding such example more challenging.

$\boxed{\text{Test on Example 5.6.}}$ In this test, we fixed $R = 6, c = 50\%$ and $\texttt{nf} = 0.1$. The complete numerical results for the 12 problems were reported in Table 6.9. It can be clearly seen that results of `MPEDM` under four objective functions have no big difference. For each data, `MPEDM` benefited from the refinement step but with different degree. One may notice for data `304D`, `MPEDM` basically failed to conform the molecule. Most importantly, for a very large size problem `2CLJ` with $n = 4189$, our proposed method run very fast, for instance $\texttt{MPEDM}_{11}$ only consumed less than 50 seconds.

Table 6.9: Self-comparisons of `MPEDM` for Example 5.6.

		$\texttt{MPEDM}_{22}$	$\texttt{MPEDM}_{21}$	$\texttt{MPEDM}_{12}$	$\texttt{MPEDM}_{11}$
	RMSD	0.887	0.886	0.886	0.886
`1GM2`	rRMSD	0.238	0.238	0.238	0.238
$n = 166$	rTime	0.159	0.153	0.154	0.153
	Time	0.276	0.237	0.248	0.262
	RMSD	3.497	3.497	3.497	3.497
`304D`	rRMSD	2.601	2.601	2.601	2.601
$n = 237$	rTime	0.147	0.149	0.147	0.147
	Time	0.244	0.252	0.256	0.266
	RMSD	1.040	1.040	1.040	1.040
`1PBM`	rRMSD	0.223	0.222	0.224	0.223
$n = 388$	rTime	0.370	0.339	0.385	0.339
	Time	0.773	0.763	0.831	0.768
	RMSD	0.918	0.918	0.918	0.918
`2MSJ`	rRMSD	0.255	0.255	0.255	0.255
$n = 480$	rTime	0.324	0.319	0.324	0.321
	Time	0.779	0.795	0.823	0.797
	RMSD	0.687	0.687	0.687	0.687
`1AU6`	rRMSD	0.173	0.172	0.172	0.173
$n = 506$	rTime	0.261	0.321	0.320	0.332
	Time	0.941	1.046	1.099	1.061
	RMSD	1.516	1.516	1.516	1.516
`1LFB`	rRMSD	0.545	0.545	0.545	0.545
$n = 641$	rTime	0.420	0.404	0.413	0.402
	Time	1.246	1.267	1.294	1.267

	RMSD	3.087	3.086	3.086	3.086
`104D`	rRMSD	1.226	1.226	1.226	1.226
$n = 766$	rTime	0.665	0.658	0.679	0.652
	Time	2.281	2.342	2.398	2.348
	RMSD	1.596	1.596	1.596	1.596
`1PHT`	rRMSD	1.032	1.032	1.032	1.032
$n = 814$	rTime	0.554	0.558	0.565	0.569
	Time	2.163	2.216	2.278	2.219
	RMSD	1.505	1.505	1.505	1.505
`1POA`	rRMSD	0.404	0.404	0.404	0.404
$n = 914$	rTime	0.536	0.542	0.532	0.538
	Time	2.550	2.614	2.672	2.585
	RMSD	1.292	1.292	1.292	1.292
`1AX8`	rRMSD	0.607	0.607	0.607	0.607
$n = 1003$	rTime	0.240	0.241	0.246	0.244
	Time	2.411	2.468	2.526	2.439
	RMSD	1.975	1.975	1.975	1.975
`1RGS`	rRMSD	0.555	0.555	0.555	0.555
$n = 2015$	rTime	1.190	1.183	1.176	1.201
	Time	10.25	10.50	10.51	10.24
	RMSD	1.561	1.561	1.561	1.561
`2CLJ`	rRMSD	0.626	0.626	0.626	0.626
$n = 4189$	rTime	2.889	2.907	2.879	2.912
	Time	44.30	45.03	45.80	44.40

6.2.3 Test on ES

$\boxed{\text{Test on Example 5.7.}}$ As described in Example 5.7, the initial dissimilarities matrix Δ can be obtained by $\Delta_{ij} = 2R_a \sin(s_{ij}/(2R_a))$. It is observed that the matrix $(J\Delta^{(2)}J)$ 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, we apply `MPEDM` to correct those errors. We plotted the resulting coordinates of the 30 cities in Figure 6.13, where the true coordinates $\mathbf{x}_i$ and estimated coordinates $\widehat{\mathbf{x}}_i$ of 30 cities were presented as blue circles and pink dots respectively. One of the remarkable features was that `MPEDM` waws able to recover the Earth radius with high accuracy $R_{\rm est} \approx 39.59 = R_a$. It seemed that $\texttt{MPEDM}_{12}$ slightly outperformed others due to the smallest `FES` and closest $R_{\rm est}$ to R_a.

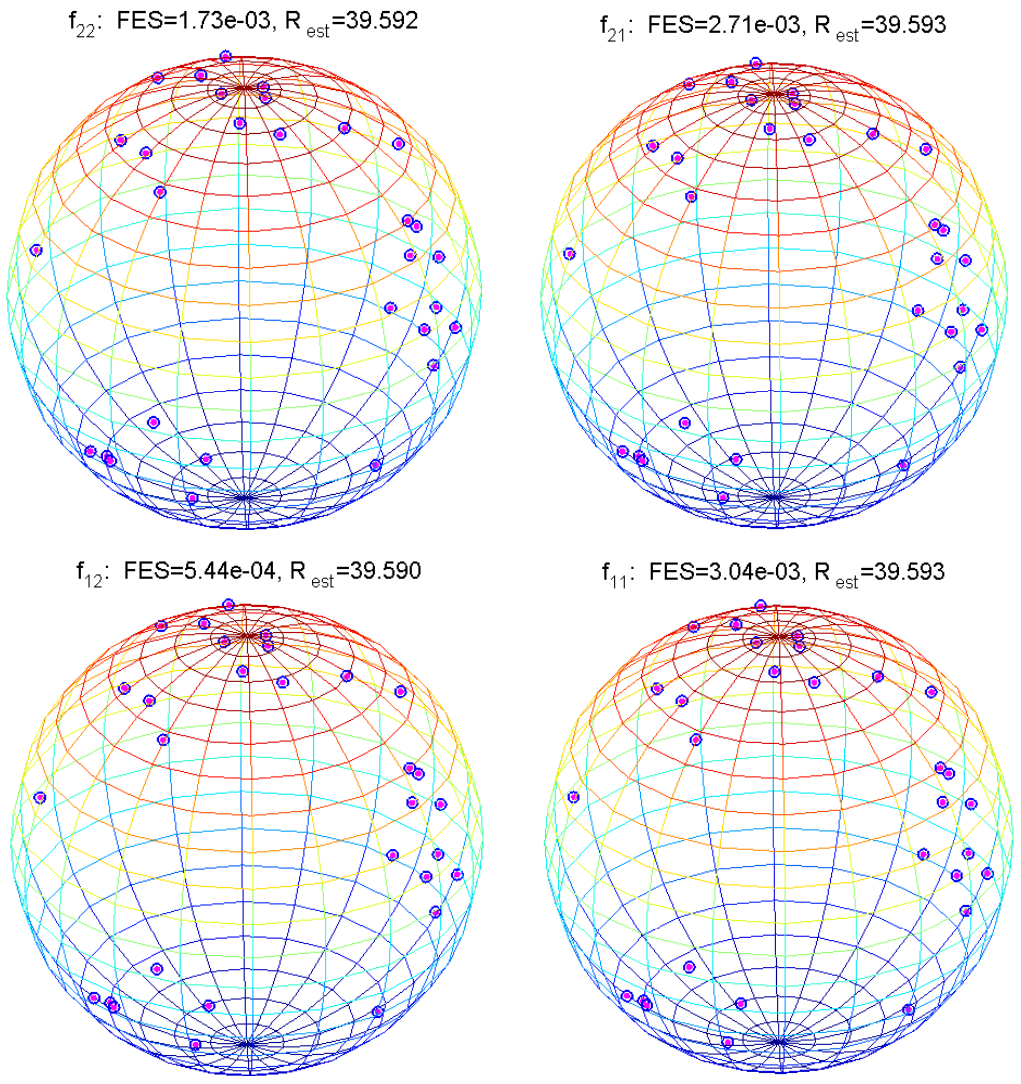


Figure 6.13: Example 5.7: embedding 30 cities on earth for data `HA30`.

$\boxed{\text{Test on Example 5.8.}}$ We apply `MPEDM` to first relocate $[\mathbf{x}_1 \cdots \mathbf{x}_6]$ to derive $[\widehat{\mathbf{x}}_1 \cdots \widehat{\mathbf{x}}_6]$ and then find a circle by `circlefit` base on new points. The new results were depicted in Figure 6.14, where the coordinates $\mathbf{x}_i$ and estimated coordinates $\widehat{\mathbf{x}}_i$ of 6 points were presented as blue circles and pink dots respectively. Apparently, the new found circles fitted much better than the circle in Figure 6.15. It is worth mentioning that four `FES` achieved by `MPEDM`$_{pq}$ were smaller than 3.6789 reported by Bai and Qi (2016) and close to 3.1724 reported by Beck and Pan (2012). One may ask if `circlefit` is able to find the circle of the 6 original points $[\mathbf{x}_1 \cdots \mathbf{x}_6]$ directly. We now plotted its found circle in Figure 6.15. Clearly, the found circle was not fitted the original points well, which also indicates our proposed method `MPEDM` making sense to relocate $[\mathbf{x}_1 \cdots \mathbf{x}_6]$ first and then find the circle.

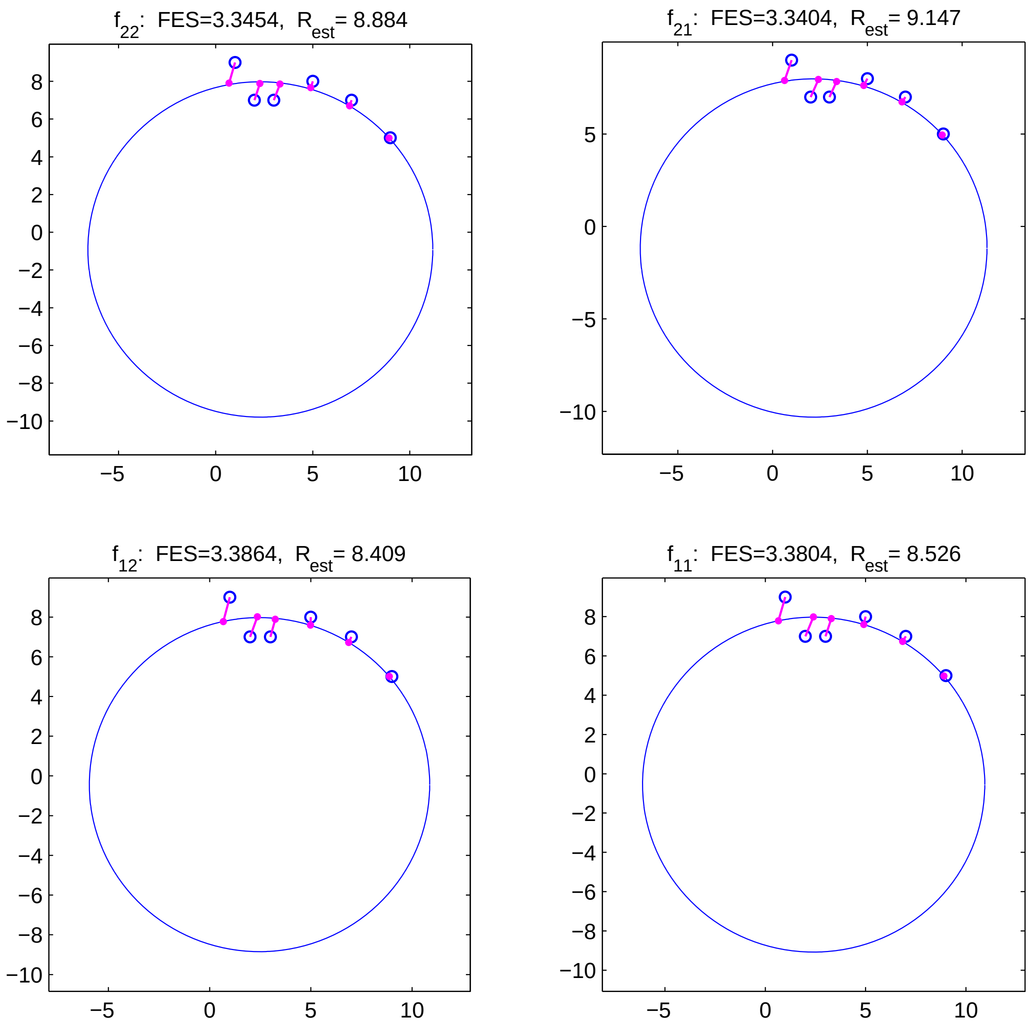


Figure 6.14: Example 5.8: fitting 6 points on a circle.

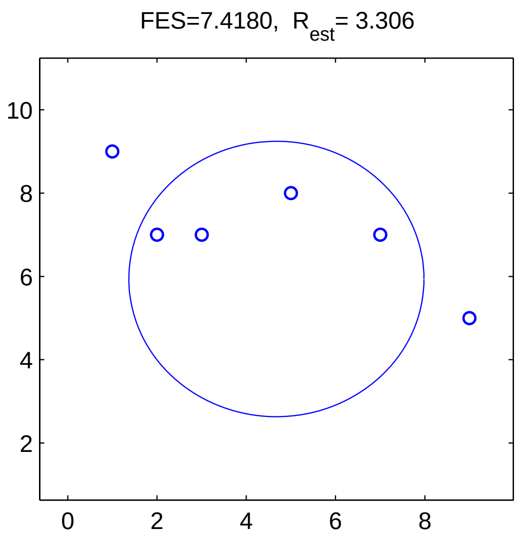


Figure 6.15: Example 5.8: fitting 6 points on a circle by `circlefit`.

$\boxed{\text{Test on Example 5.9.}}$ We first demonstrate the performance of `MPEDM` under each f to n by fixing `nf`$= 0.1$ and altering n among $\{10, 20, \cdots, 100\}$. Clearly, as increasing of n, more information will be obtained, leading to problems being easier. This phenomenon can be explained by Figure 6.16, in which we plotted results by $\texttt{MPEDM}_{11}$. The coordinates $\mathbf{x}_i$ and estimated coordinates $\widehat{\mathbf{x}}_i$ were presented as blue circles and pink dots respectively. It can be clearly seen that `FES` declined and $R_{\rm est}$ got closer to 1 when n got bigger, namely, more and more information were provided.

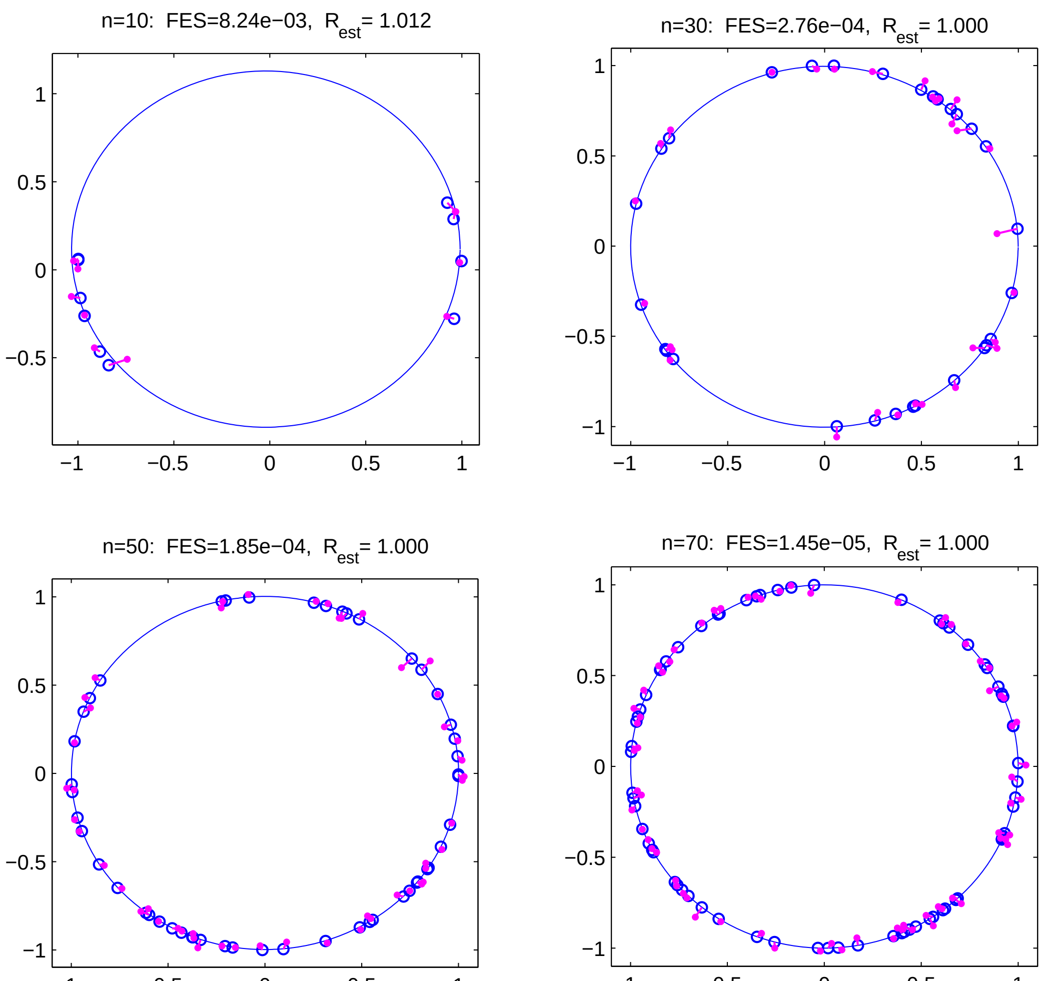


Figure 6.16: Example 5.9: circle fitting with `nf`$= 0.1$ by $\texttt{MPEDM}_{11}$.

Average results were plotted in Figure 6.17 in which $\texttt{MPEDM}_{11}$ got the best `FES` and `RMSD` in most cases, followed by $\texttt{MPEDM}_{12}$, $\texttt{MPEDM}_{21}$ and $\texttt{MPEDM}_{22}$. In terms of cup time, $\texttt{MPEDM}_{12}$ ran the fastest and $\texttt{MPEDM}_{11}$ came last.

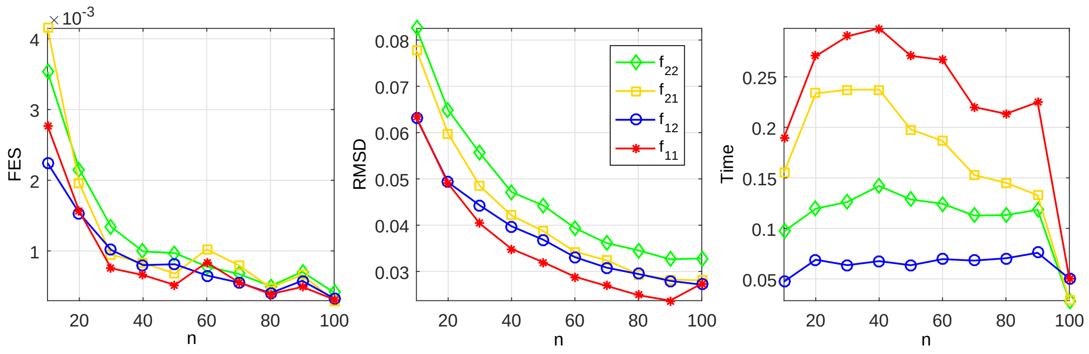
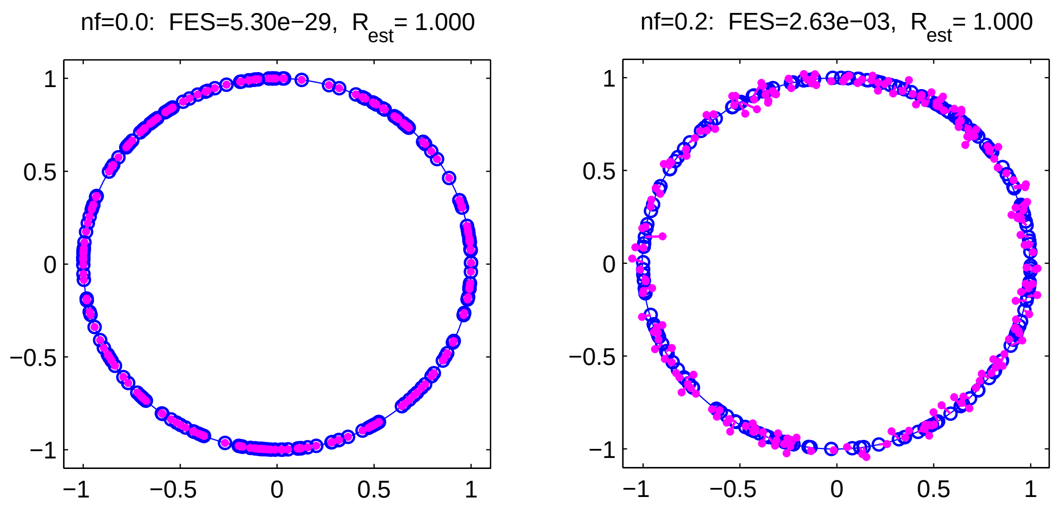


Figure 6.17: Example 5.9 with `nf` $= 0.1$.



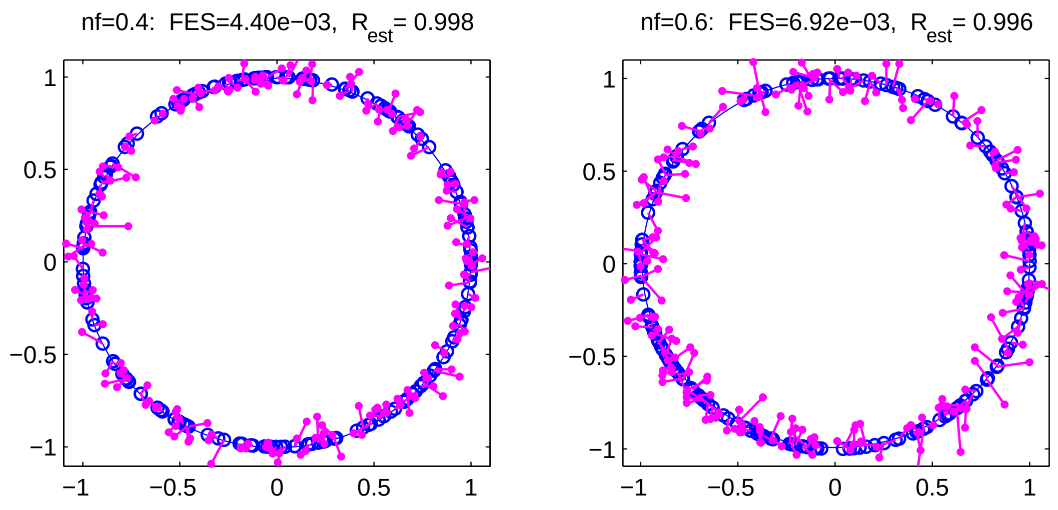


Figure 6.18: Example 5.9: circle fitting with $n = 200$ by $\texttt{MPEDM}_{11}$.

We then demonstrate the performance of $\texttt{MPEDM}_{pq}$ to the noise factor `nf` by fixing $n = 200$, and altering `nf` among $\{0.1, 0.2, \cdots, 0.7\}$. Similarly, we presented results by $\texttt{MPEDM}_{11}$

in Figure 6.18. Apparently, `FES` became bigger and $R_{\rm est}$ got more far away from 1 when `nf` increased. However, even with very large noise (e.g., `nf` $= 0.7$) being contaminated, `MPEDM` was still able to find a circle that fitted the estimated data slightly bad but fitted the original data very well.

Average results were demonstrated in Figure 6.19. Clearly, $\texttt{MPEDM}_{11}$ got the best `FES` and `RMSD`, followed by $\texttt{MPEDM}_{12}$, and $\texttt{MPEDM}_{21}$. $\texttt{MPEDM}_{22}$ came last. In terms of cup time, $\texttt{MPEDM}_{22}$ and $\texttt{MPEDM}_{21}$ ran faster than the other two. Moreover, all `FES`, `RMSD` and `Time` were ascending with `nf` rising. However, since all `FES` were quite small (in order of 10^{-2}), `MPEDM` was capable of find a proper circle to well fit original data.

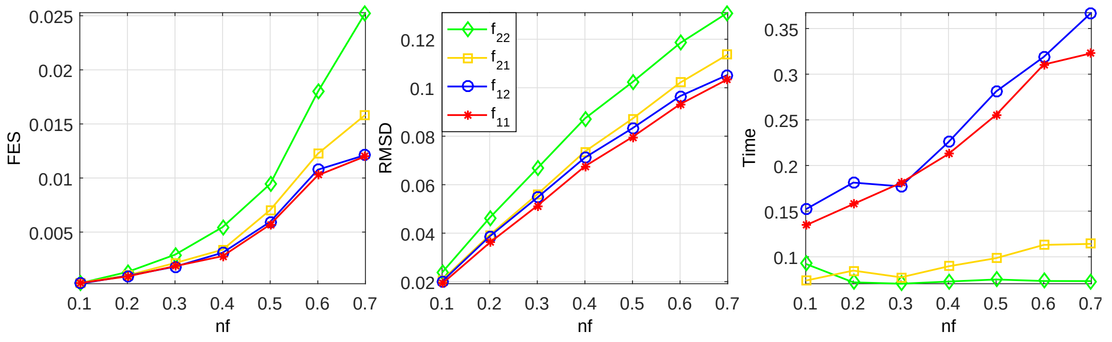
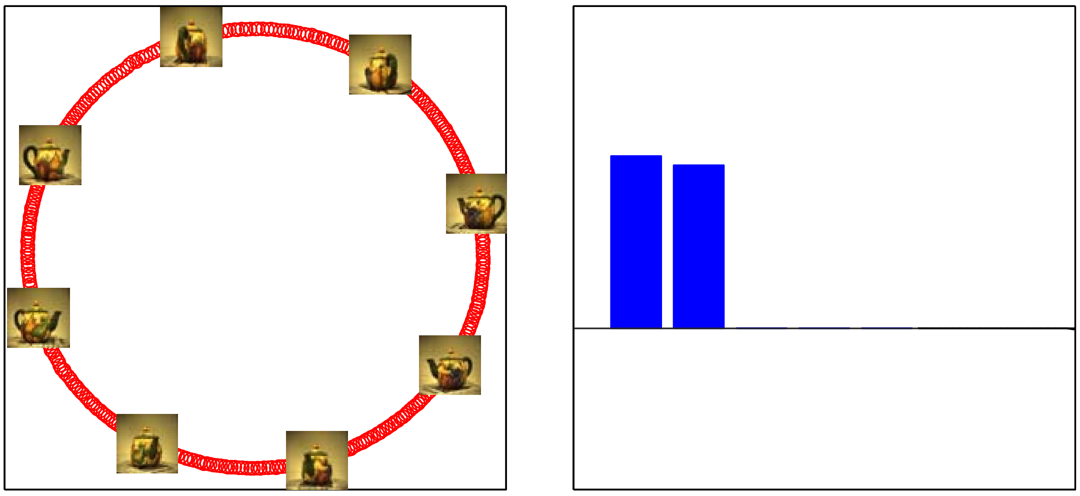


Figure 6.19: Example 5.9 with $n = 200$.

6.2.4 Test on DR

For simplicity, 5-NNR (i.e., $k = 5$) is used to generate N_{xx} and `nf` $= 0.1$. To reduce the dimensionality sufficiently, we set `Ktol`$= 10^{-5}$ in (6.3). Since below numerical results have shown that each $\texttt{MPEDM}_{pq}$ has similar results for DR problems, we only visualize results of $\texttt{MPEDM}_{11}$ on graphs.

$\boxed{\text{Test on Example 5.10.}}$ The ‘`teapot`’ images have 76×101 pixels, with 3 byte color depth, giving rise to inputs of 23028 dimensions. As described by Weinberger and Saul (2006), though very high dimensional, the images in this data set are effectively parameterized by one degree of freedom: the angle of rotation, and two dimensional ($r = 2$) embedding is able to represent the rotating object as a circle. As presented in Figure 6.20, $\texttt{MPEDM}_{11}$ generated embedding which formed a proper circle as expected, and obtained two large eigenvalues of $(-J\overline{D}^{(2)}J/2)$. It is worth mentioning that ISOMAP (Tenenbaum et al., 2000) returned more than two nonzero eigenvalues, which leaded to the artificial wave in the third dimension, seen comments by Ding and Qi (2017).



(a) Visualization (b) Eigenvalues

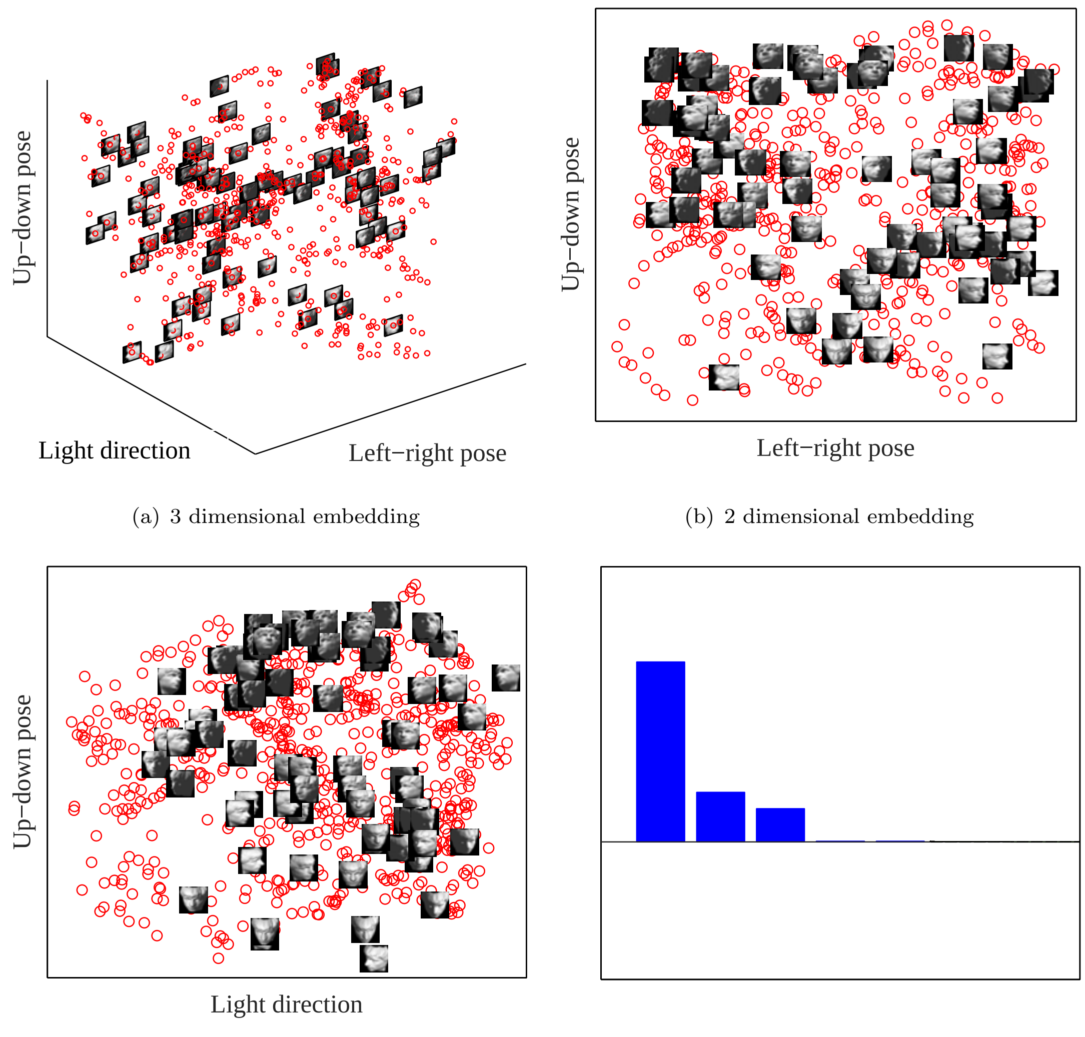
Figure 6.20: Example 5.10: dimensionality reduction by `MPEDM`.

The whole results were recorded in Table 6.10, where EigScore(2) was very close to 1, indicating `MPEDM` was capable of capturing the two main features of '`teapot`' images data. Moreover, our method also preserved the local distances well since PRE was quite small, and apparently, ran very fast.

Table 6.10: Results of $\texttt{MPEDM}_{pq}$ on Example 5.10.

	$\texttt{MPEDM}_{22}$	$\texttt{MPEDM}_{21}$	$\texttt{MPEDM}_{12}$	$\texttt{MPEDM}_{11}$
EigScore(2)	0.9848	0.9848	0.9848	0.9848
PRE	0.0764	0.0704	0.0704	0.0704
Time	0.4085	0.2409	0.2614	0.3057

Test on Example 5.11. The '`face698`' images have three features, naturally leading to 3 dimensional embedding. As demonstrated in Figure 6.21, $\texttt{MPEDM}_{11}$ generated embedding well capturing the three features. More detailed, in subfigure 6.21(b), from the horizontal axis, faces in images pointed to left side and gradually to right side, then from the vertical axis, faces in images looked down and gradually looked up. In subfigure 6.21(c), the light shot faces from left side and gradually shot faces from right side. And it can clearly seen that $\texttt{MPEDM}_{11}$ rendered three large eigenvalues of $(-J\overline{D}^{(2)}J/2)$ in subfigure 6.21(d).



(a) 3 dimensional embedding (b) 2 dimensional embedding

(c) 2 dimensional embedding (d) 3 main eigenvalues

Figure 6.21: Example 5.11: dimensionality reduction by `MPEDM`.

The whole results were recorded in Table 6.11, where EigScore(3) was very close to 1, indicating `MPEDM` was capable of capturing the three major features of '`face698`' images data. Moreover, our method also preserved the local distances well since PRE was quite small, and apparently, ran very fast.

Table 6.11: Results of $\texttt{MPEDM}_{pq}$ on Example 5.11.

	$\texttt{MPEDM}_{22}$	$\texttt{MPEDM}_{21}$	$\texttt{MPEDM}_{12}$	$\texttt{MPEDM}_{11}$
EigScore(3)	0.9616	0.9612	0.9612	0.9612
PRE	0.0704	0.0681	0.0681	0.0681
Time	1.6568	1.5571	1.8546	1.6493

Test on Example 5.12. The '`digit1`' images have two features, naturally leading to 2 dimensional embedding. As demonstrated in Figure 6.22, two features 'Line thickness' and 'Slant' generated by $\texttt{MPEDM}_{11}$ were properly posed in left subfigure, and thus two expected relatively large eigenvalues were got (see in right subfigure).

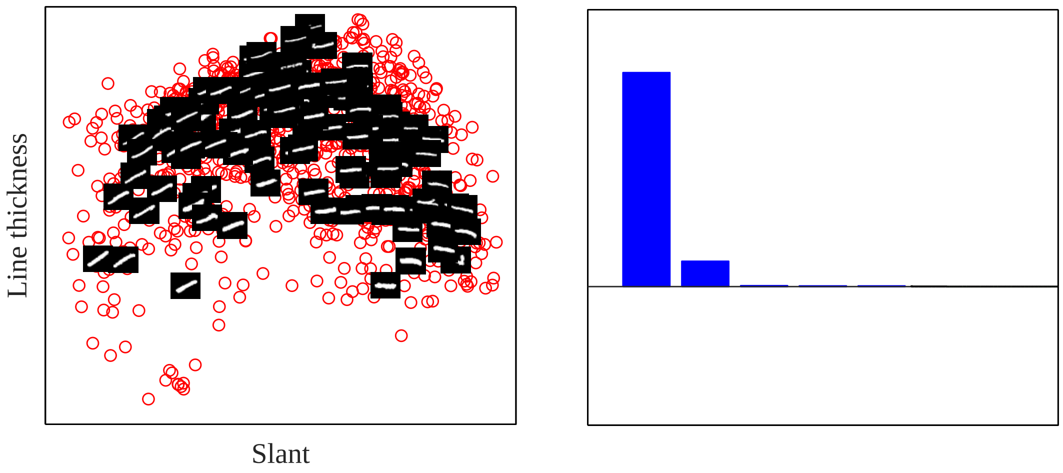


Figure 6.22: Example 5.12: dimensionality reduction by `MPEDM`.

The whole results were recorded in Table 6.12, where EigScore(2) was very close to 1, indicating `MPEDM` was capable of capturing the two main features of '`digit1`' images data. Moreover, our method also preserved the local distances well since PRE was quite small, and apparently, ran fast.

Table 6.12: Results of $\texttt{MPEDM}_{pq}$ on Example 5.12.

	$\texttt{MPEDM}_{22}$	$\texttt{MPEDM}_{21}$	$\texttt{MPEDM}_{12}$	$\texttt{MPEDM}_{11}$
EigScore(2)	0.9375	0.9376	0.9376	0.9376
PRE	0.0773	0.0702	0.0702	0.0702
Time	13.274	14.209	15.891	14.664

6.3 Numerical Comparison with Existing Methods

In this section, we will compare our proposed method `MPEDM` under objective function f_{11} (i.e., $\texttt{MPEDM}_{11}$, for simplicity, we still write it as `MPEDM`) with six representative state-of-the-art methods: `ADMMSNL` (Piovesan and Erseghe, 2016), `ARAP` (Zhang et al., 2010), EVEDM (Drusvyatskiy et al., 2017, short for EepVecEDM), PC (Agarwal et al., 2010), PPAS (Jiang et al., 2013, short for PPA Semismooth) and SFSDP (Kim et al., 2012). In following comparison, results of some methods will be omitted either it made our desktop ran out of memory (such as ADMMSNL when $n \geq 500, R = \sqrt{2}$ for Example 5.1) or it consumed too much time being longer than 10^4 seconds (such as ARAP when $n \geq 1000, R = \sqrt{2}$ for Example 5.1). We use '$--$' to denote the omitted results.

6.3.1 Benchmark methods

The above mentioned six methods have been shown to be capable of returning satisfactory localization/embedding in many applications. We will compare our method MPEDM with ADMMSNL, ARAP, EVEDM, PC and SFSDP for SNL problems and with EVEDM, PC, PPAS and SFSDP for MC problems since the current implementations of ADMMSNL, ARAP do not support the embedding for $r \geq 3$.

We note that ADMMSNL is motivated by Soares et al. (2015) and aims to enhance the package diskRelax of Soares et al. (2015) for the SNL problems ($r = 2$). Both methods are based on the stress minimization (2.3). As we mentioned before that SMACOF (De et al., 1977; De Leeuw and Mair, 2011) has been a very popular method to tackle stress minimization (2.3) though, we are not to compare it with other methods here since its performance demonstrated in (Zhang et al., 2010; Zhou et al., 2018a) was poorly both for SNL and MC problems. PC was proposed to deal with the model with same objective function of (2.6). We select SFSDP because it solves problem with objective function (2.5). The rest of methods take advantage of "squared" distances and least square loss function, namely, (2.4). Overall, each model will be addressed by these methods.

In our tests, we used all of their default parameters except one or two in order to achieve the best results. In particular, for PC, we terminate it when $|f(D^{k-1}) - f(D^k)| < 10^{-4} \times f(D^k)$ and set its initial point the embedding by cMDS on Δ. For SFSDP which is a high-level MATLAB implementation of the SDP approach initiated in Wang et al. (2008), we set pars.SDPsolver = "sedumi" because it returns the best overall performance. In addition, as suggested when we solve problems with noise, we set pars.objSW = 1 when $m > r + 1$ and $= 3$ when $m = 0$. For ARAP, in order to speed up the termination, we let tol $= 0.05$ and IterNum $= 20$ to compute its local neighbour patches. Numerical performance demonstrated that ARAP could yield satisfactory embedding, but took long time for examples with large n.

6.3.2 Comparison on SNL

$\boxed{\text{Effect to Radio range } R.}$ It is easy to see that the radio range R decides the amount of missing dissimilarities among all elements of Δ. The smaller R is, the more numbers of δ_{ij} are unavailable, yielding more difficult problems. Therefore, we first demonstrate the performance of each method to the radio range R. For Example 5.1, we fix $n = 200, m = 4$, nf$= 0.1$, and alter the radio range R among $\{0.2, 0.4, \cdots, 1.4\}$. Average results were demonstrated in Figure 6.23. It can be seen that ARAP and MPEDM were joint winners in terms of both RMSD and rRMSD. However, the time used by ARAP was the longest. When R got bigger than 0.6, ADMMSNL, SFSDP and EVEDM produced similar rRMSD as ARAP and MPEDM, while the time consumed by ADMMSNL was significantly larger than that by SFSDP, EVEDM and MPEDM. By contrast, PC only worked well when $R \geq 1$.

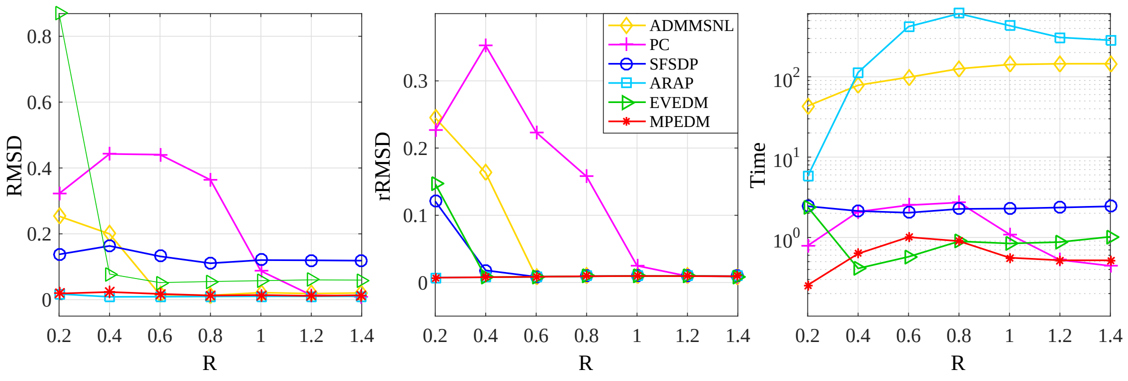


Figure 6.23: Average results for Example 5.1 with $n = 200, m = 4$, nf$= 0.1$.

Next we test a number of instances with larger size $n \in \{300, 500, 1000, 2000\}$. For Example 5.1, average results were recorded in Table 6.13. When $R = \sqrt{2}$ under which no dissimilarities were missing because Example 5.1 was generated in a unit region, PC, ARAP and MPEDM produced the better RMSD (almost in order of 10^{-3}) but after refinement all methods got similar rRMSD. This meant SFSDP and EVEDM benefited a lot from refinement. For computational speed, MPEDM outperformed others, followed by PC, EVEDM and SFSDP. By contrast, ARAP consumed too much time even when $n = 500$.

Table 6.13: Comparison for Example 5.1 with $m = 4, R = \sqrt{2}, \texttt{nf} = 0.1$.

n		`ADMMSNL`	`PC`	`SFSDP`	`ARAP`	`EVEDM`	`MPEDM`
300	RMSD	2.07e-2	**8.31e-3**	1.21e-1	1.01e-2	5.95e-2	1.11e-2
	rRMSD	7.82e-3	7.86e-3	7.89e-3	7.96e-3	7.93e-3	**7.80e-3**
	rTime	3.63	0.66	3.87	0.94	3.35	1.06
	Time	348.13	1.36	6.79	503.86	3.84	**1.36**
500	RMSD	−−	**6.11e-3**	1.19e-1	7.51e-3	5.87e-2	8.46e-3
	rRMSD	−−	**5.94e-3**	5.96e-3	6.04e-3	6.70e-3	6.11e-3
	rTime	−−	1.37	14.79	3.26	13.35	3.92
	Time	−−	**3.83**	20.22	2479.8	14.44	4.41
1000	RMSD	−−	**4.46e-3**	1.25e-1	−−	5.81e-2	6.59e-3
	rRMSD	−−	**4.15e-3**	7.34e-3	−−	6.53e-3	4.59e-3
	rTime	−−	3.51	83.96	−−	68.06	9.75
	Time	−−	23.05	103.29	−−	71.52	**10.85**
2000	RMSD	−−	**3.30e-3**	1.20e-1	−−	5.92e-2	4.57e-3
	rRMSD	−−	**3.10e-3**	7.82e-3	−−	1.24e-2	3.37e-3
	rTime	−−	12.74	282.88	−−	258.97	13.04
	Time	−−	143.41	398.87	−−	271.91	**18.49**

Table 6.14: Comparison for Example 5.1 with $m = 4, R = 0.2, \texttt{nf} = 0.1$.

n		`ADMMSNL`	`PC`	`SFSDP`	`ARAP`	`EVEDM`	`MPEDM`
300	RMSD	3.48e-1	4.42e-1	1.93e-1	4.02e-2	6.81e+1	**1.88e-2**
	rRMSD	3.33e-1	3.12e-1	1.73e-1	6.83e-3	1.72e-1	**6.82e-3**
	rTime	0.50	0.44	0.41	0.36	0.48	0.36
	Time	84.19	2.37	3.45	24.11	0.56	**0.47**
500	RMSD	3.53e-1	4.30e-1	2.02e-1	1.95e-2	1.52e-1	**1.77e-2**
	rRMSD	3.35e-1	3.11e-1	1.80e-1	5.57e-3	5.59e-2	**5.51e-3**
	rTime	1.11	1.15	1.06	0.80	1.11	0.92
	Time	156.76	5.50	6.90	161.04	1.30	**1.23**
1000	RMSD	3.62e-1	4.54e-1	1.79e-1	**9.96e-3**	7.21e-2	1.46e-2
	rRMSD	3.44e-1	3.16e-1	1.28e-1	3.57e-3	**4.06e-3**	3.83e-3
	rTime	5.58	5.58	5.25	1.69	5.16	3.76
	Time	450.03	24.82	19.90	2833.5	6.00	**5.86**
2000	RMSD	3.71e-1	4.35e-1	1.80e-1	−−	5.92e-2	**1.37e-2**
	rRMSD	3.51e-1	3.63e-1	8.29e-2	−−	3.53e-3	**3.29e-3**
	rTime	40.40	40.65	37.94	−−	24.72	4.58
	Time	1255.1	171.01	77.03	−−	32.31	**17.51**

When $R = 0.2$, average results were reported in Table 6.14. The picture was significantly different since there were large amounts of unavailable dissimilarities in Δ. Basically, `ADMMSNL`, `PC` and `SFSDP` failed to localize even with refinement due to undesirable `RMSD` and `rRMSD` (both in order of 10^{-1}). Clearly, `ARAP` and `MPEDM` produced the best `RMSD` and `rRMSD`, and `EVEDM` got comparable `rRMSD` but inaccurate `RMSD`. In terms of computational speed, `EVEDM` and `MPEDM` were very fast, consuming about 30 seconds to solve problem with $n = 2000$ nodes. By contrast, `ARAP` and `ADMMSNL` still were the slowest.

For Example 5.4, average results were recorded in Table 6.15. One can discern that no and large numbers of dissimilarities were missing when $R = \sqrt{1.25}$ and $R = 0.1$ respectively because this example was generated in region $[0, 1] \times [0, 0.5]$ as presented in Fig. 5.1. When $R = \sqrt{1.25}$, it can be clearly seen that `SFSDP` and `EVEDM` basically failed to recover before refinement owing to large `RMSD` (in order of 10^{-1}), whilst the rest four methods succeeded in localizing. However, they all achieved similar `rRMSD` after refinement except for `EVEDM` under the case $n = 500$. Still, `MPEDM` ran the fastest and `ARAP` came the last, (5.13 vs. 2556.3 when $n = 500$).

Table 6.15: Comparisons for Example 5.4 with $m = 10, R = \sqrt{1.25}, \texttt{nf} = 0.1$.

n		`ADMMSNL`	`PC`	`SFSDP`	`ARAP`	`EVEDM`	`MPEDM`
300	RMSD	4.02e-2	**5.33e-3**	1.45e-1	1.27e-2	1.62e-1	9.26e-3
	rRMSD	5.12e-3	5.14e-3	5.11e-3	5.12e-3	**5.09e-3**	5.15e-3
	rTime	3.28	0.66	3.71	1.69	3.94	1.44
	Time	346.98	2.00	6.74	553.87	4.42	**1.87**
500	RMSD	−−	**4.09e-3**	1.07e-1	8.50e-3	1.63e-1	7.15e-3
	rRMSD	−−	**4.03e-3**	4.04e-3	4.05e-3	1.02e-1	4.15e-3
	rTime	−−	2.68	17.28	7.07	17.39	3.12
	Time	−−	7.24	23.44	2556.3	18.89	**5.13**
1000	RMSD	−−	**3.07e-3**	1.12e-1	−−	1.28e-1	5.05e-3
	rRMSD	−−	2.98e-3	3.50e-3	−−	4.15e-3	**3.15e-3**
	rTime	−−	10.35	119.79	−−	122.12	15.73
	Time	−−	43.69	140.66	−−	125.46	**20.11**
2000	RMSD	−−	**2.36e-3**	1.15e-1	−−	1.03e-1	3.75e-3
	rRMSD	−−	2.28e-3	7.34e-3	−−	7.78e-3	**2.26e-3**
	rTime	−−	13.43	537.70	−−	489.30	10.59
	Time	−−	238.31	659.71	−−	500.72	**20.25**

Now take a look at the results of $R = 0.1$ in Table 6.16. `MPEDM` generated the most accurate `RMSD` and `rRMSD` (in order of 10^{-3}) whilst results of rest methods were only in order of 10^{-2}. Obviously, `ADMMSNL`, `PC` and `EVEDM` failed to localize. Compared with other four methods, `EVEDM` and `MPEDM` were joint winners in terms of computational speed, only with 30 seconds to address problems with $n = 2000$, a large scale network. But we should mention here `EVEDM` failed to localize.

Table 6.16: Comparisons for Example 5.4 with $m = 10, R = 0.1, \texttt{nf} = 0.1$.

n		`ADMMSNL`	`PC`	`SFSDP`	`ARAP`	`EVEDM`	`MPEDM`
300	RMSD	1.81e-1	3.77e-1	8.64e-2	8.19e-2	4.06e-1	**3.97e-2**
	rRMSD	1.43e-1	1.24e-1	6.69e-2	5.38e-2	1.17e-1	**8.21e-3**
	rTime	0.27	0.22	0.21	0.21	0.22	0.21
	Time	76.57	1.21	3.24	7.24	3.41	**0.32**
500	RMSD	9.73e-2	3.30e-1	5.08e-2	5.77e-2	2.16e-1	**3.63e-2**
	rRMSD	7.82e-2	1.15e-1	3.48e-2	3.08e-2	9.78e-2	**3.63e-3**
	rTime	0.67	0.63	0.60	0.58	0.61	0.50
	Time	148.06	3.63	6.41	50.81	2.07	**1.85**
1000	RMSD	2.26e-1	3.29e-1	**4.80e-2**	8.75e-2	2.22e-1	5.01e-2
	rRMSD	1.01e-1	1.21e-1	9.15e-3	4.55e-2	1.02e-1	**2.95e-3**
	rTime	2.74	2.66	2.67	2.58	2.61	2.60
	Time	353.07	18.01	17.10	842.43	**3.22**	4.24
2000	RMSD	1.66e-1	3.29e-1	8.21e-2	−−	1.02e-1	**5.73e-2**
	rRMSD	1.22e-1	1.53e-1	7.10e-2	−−	3.64e-2	**4.97e-3**
	rTime	23.22	23.30	23.06	−−	23.12	17.99
	Time	887.30	108.81	62.65	−−	**26.12**	29.89

$\boxed{\text{Effect to anchors' number } m.}$ As what we expect, more anchors are given, and more easily the problem is to be solved since more information are provided. We thus then demonstrate how anchors' number m would effect the performance of each method. For Example 5.2, we fix $n = 200, R = 0.2$, `nf`$= 0.1$, and alter anchors' number m among $\{5, 10, \cdots, 40\}$. As presented in Figure 6.24, it can be seen that `ARAP` and `MPEDM` were again joint winners in terms of both `RMSD` and `rRMSD`. And `rRMSD` produced by the rest methods declined along with more anchors being given. What is more, `MPEDM` was the fastest, followed by `EVEDM`, `PC` and `SFSDP`, whilst `ADMMSNL` and `ARAP` ran quite slowly.

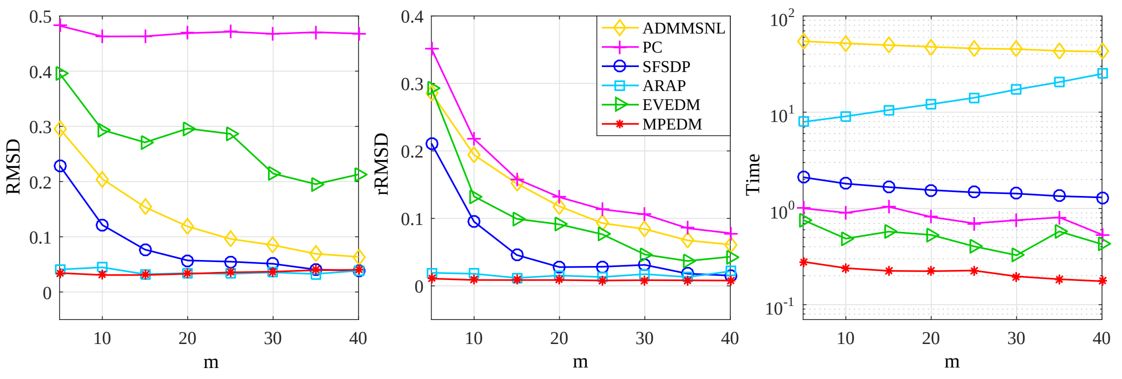


Figure 6.24: Average results for Example 5.2 with $n = 200, R = 0.2$, nf$= 0.1$.

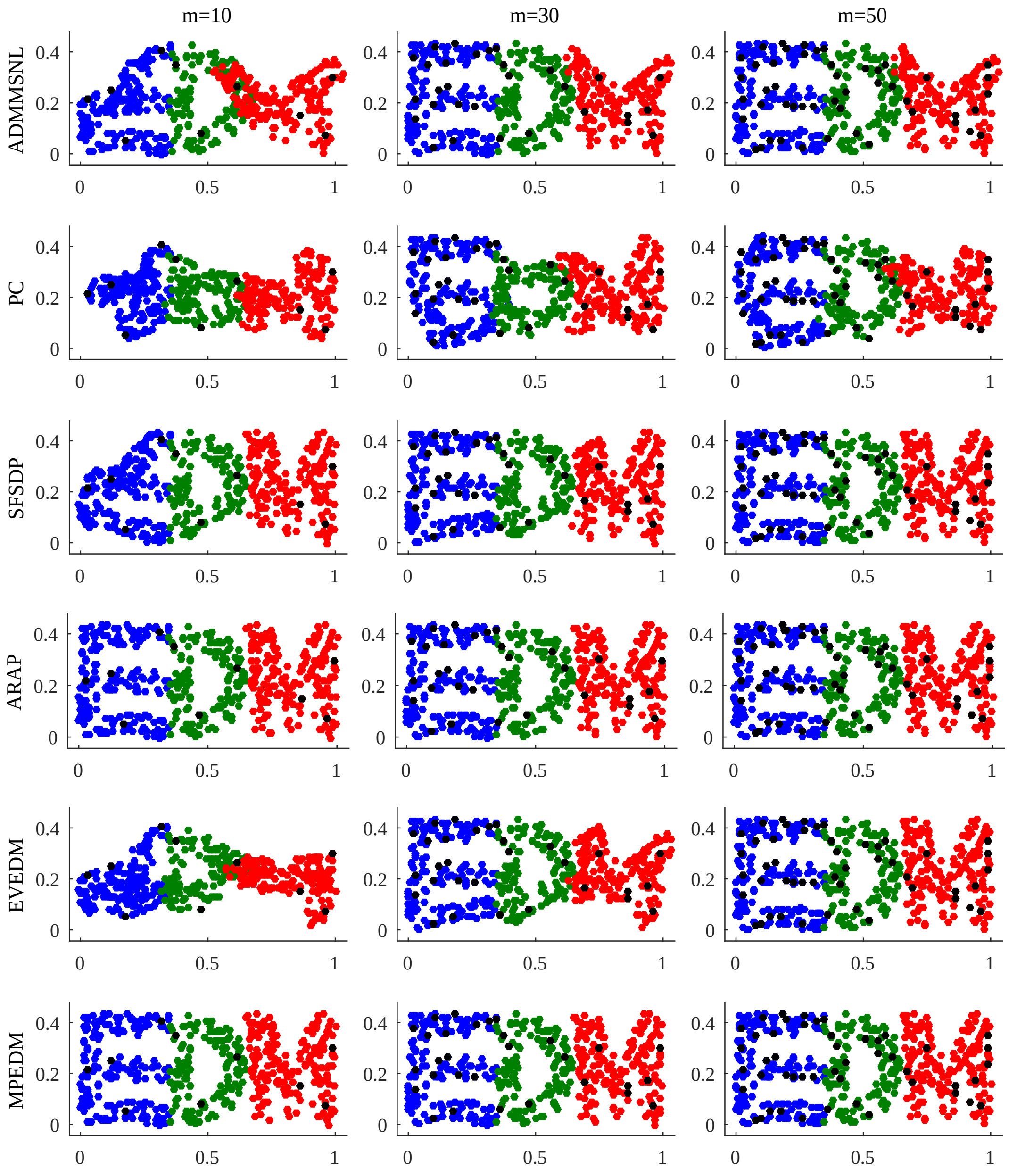


Figure 6.25: Localization for Example 5.4 with $n = 500, R = 0.1$, nf$= 0.1$.

Next for Example 5.4 with fixed $n = 500, R = 0.1$, $\texttt{nf}= 0.1$, we test it under $m \in \{10, 30, 50\}$. As depicted in Figure 6.25, `ARAP` and `MPEDM` were always capable of capturing the shape of letters 'E', 'D' and 'M' that was similar to Figure 5.1. By contrast, `SFSDP` and `EVEDM` derived desirable outline of three letters only when $m = 50$, and `ADMMSNL` and `PC` got better results along with increasing of m but still with deformed shape of letter 'M'.

Finally we test a number of instances with sizes $n \in \{300, 500, 1000, 2000\}$. For Example 5.2 with $m = 10$, its average results were recorded in Table 6.17. `ADMMSNL` and `PC` got undesirable `RMSD` and `rRMSD` (both in order of 10^{-1}). `SFSDP` benefited greatly from the refinement because it generated relatively inaccurate `RMSD`. By contrast the rest three methods enjoyed the successful recovering except for `EVEDM` under the case $n = 300$. Regarding to computational speed, `EVEDM` and `MPEDM` were the fastest, followed by `SFSDP`, `PC`, `ADMMSNL` and `ARAP`.

Table 6.17: Comparisons for Example 5.2 with $m = 10, R = 0.2, \texttt{nf} = 0.1$.

n		ADMMSNL	PC	SFSDP	ARAP	EVEDM	MPEDM
300	RMSD	2.56e-1	4.59e-1	1.34e-1	**2.60e-2**	2.72e-1	3.99e-2
	rRMSD	2.49e-1	2.43e-1	7.19e-2	6.71e-3	1.44e-1	**6.69e-3**
	rTime	0.40	0.43	0.36	0.26	0.39	0.28
	Time	81.62	2.02	3.18	24.92	0.47	**0.40**
500	RMSD	1.86e-1	4.41e-1	9.70e-2	**2.42e-2**	8.62e-2	3.29e-2
	rRMSD	1.82e-1	2.07e-1	4.99e-2	5.07e-3	5.05e-3	**5.02e-3**
	rTime	0.81	1.30	0.93	0.69	0.84	0.64
	Time	163.55	4.70	6.67	170.82	1.04	**1.02**
1000	RMSD	1.82e-1	4.39e-1	9.93e-2	**2.71e-2**	6.88e-2	3.95e-2
	rRMSD	1.60e-1	1.96e-1	2.92e-2	3.21e-3	**3.20e-3**	3.63e-3
	rTime	4.79	5.53	4.38	3.90	4.66	3.71
	Time	441.08	24.70	18.64	2861.9	5.47	**5.18**
2000	RMSD	2.17e-1	4.39e-1	1.30e-1	$--$	6.08e-2	**5.03e-2**
	rRMSD	1.87e-1	2.54e-1	6.88e-2	$--$	**2.64e-3**	2.82e-3
	rTime	39.22	39.32	36.29	$--$	33.85	14.43
	Time	1251.07	170.55	75.29	$--$	37.33	**28.95**

When $m = 50$, its average results were recorded in Table 6.18. Under such case, more information known, results were better than before, especially for methods `ADMMSNL` and

PC. But PC still solved problems unsuccessfully before refinement. The rest five methods basically managed to embed all problems but with different degrees. For example, MPEDM produced the most accurate rRMSD for all cases. The comparison of computational speed is similar to the case of $m = 10$.

Table 6.18: Comparisons for Example 5.2 with $m = 50, R = 0.2, \texttt{nf} = 0.1$.

n		ADMMSNL	PC	SFSDP	ARAP	EVEDM	MPEDM
300	RMSD	3.19e-2	4.49e-1	**3.09e-2**	5.30e-2	1.09e-1	5.07e-2
	rRMSD	3.10e-2	4.39e-2	1.13e-2	1.26e-2	1.84e-2	**5.78e-3**
	rTime	0.12	0.20	0.09	0.09	0.11	0.09
	Time	74.71	1.44	2.41	48.83	**0.22**	0.25
500	RMSD	2.80e-2	4.60e-1	**3.54e-2**	4.39e-2	5.10e-2	6.09e-2
	rRMSD	2.68e-2	4.93e-2	6.77e-3	**4.42e-3**	5.61e-3	**4.42e-3**
	rTime	0.24	0.50	0.21	0.21	0.19	0.19
	Time	144.93	4.25	4.67	232.14	**0.46**	0.72
1000	RMSD	1.91e-2	4.57e-1	3.21e-2	**2.27e-2**	5.06e-2	5.99e-2
	rRMSD	1.27e-2	3.75e-2	4.76e-3	2.94e-3	**2.94e-3**	**2.94e-3**
	rTime	1.05	2.52	1.10	1.05	1.01	1.12
	Time	406.88	20.29	12.48	3150.6	**1.86**	4.02
2000	RMSD	2.17e-2	4.47e-1	**3.63e-2**	−−	5.16e-2	4.72e-2
	rRMSD	6.13e-3	2.78e-2	3.52e-3	−−	**2.06e-3**	**2.06e-3**
	rTime	11.89	25.95	10.43	−−	8.80	7.71
	Time	1171.22	156.45	40.45	−−	**11.15**	22.53

$\boxed{\text{Effect to noise factor } \texttt{nf}.}$ To see the performance of each method to the noise factor, we first test Example 5.4 with fixing $n = 200, m = 10, R = 0.3$ and varying the noise factor $\texttt{nf} \in \{0.1, 0.2, \cdots, 0.7\}$. As shown in Figure 6.26, in terms of RMSD it can be seen that ARAP got the smallest ones, whilst EVEDM and PC obtained the worst ones. The line of ADMMSNL dropped down from $0.1 \leq \texttt{nf} \leq 0.3$ and then ascended, by contrast the line of MPEDM reached the peak at $\texttt{nf} = 0.3$ but declined afterwards and gradually approached to RMSD of ARAP. However, after refinement, ARAP, SFSDP and MPEDM derived similar rRMSD while the other three methods produced undesirable ones. Apparently, EVEDM was indeed the fastest but basically failed to locate when $\texttt{nf} \geq 0.3$, followed by PC, SFSDP and MPEDM. Again, ARAP and ADMMSNL were always the slowest.

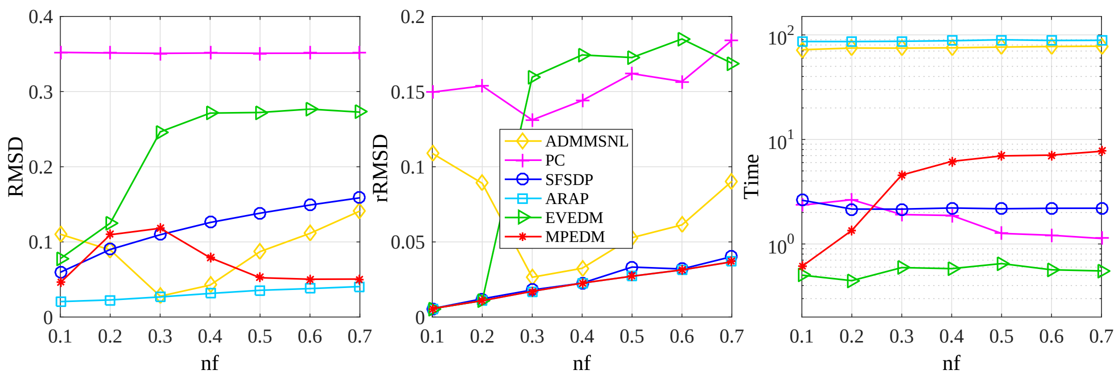


Figure 6.26: Average results for Example 5.4 with $n = 200, m = 10, R = 0.3$.

Next, we test Example 5.2 with a moderate size $n = 200, m = 4$ and $R = 0.3$ but with altering $\texttt{nf} \in \{0.1, 0.3, 0.5\}$. The actual embedding by each method was shown in Figure 6.27, where the four anchors were plotted in green square and $\widehat{\mathbf{x}}_i$ in pink points were jointed to its ground truth location (blue circle). It can be clearly seen that ARAP and MPEDM were quite robust to the noise factor since their localization matched the ground truth well. EVEDM failed to locate when $\texttt{nf} = 0.5$. By contrast, SFSDP generated worse results when nf got bigger, and ADMMSNL and PC failed to localize for all cases.

Table 6.19: Comparisons for Example 5.1 with $m = 4, R = 0.3, \texttt{nf} = 0.1$.

n		ADMMSNL	PC	SFSDP	ARAP	EVEDM	MPEDM
300	RMSD	3.16e-1	4.46e-1	1.74e-1	**1.03e-2**	6.58e-2	1.64e-2
	rRMSD	2.84e-1	3.10e-1	9.63e-2	6.62e-3	**6.55e-3**	6.57e-3
	rTime	0.75	0.71	0.62	0.31	0.43	0.34
	Time	101.07	3.09	4.39	117.33	**0.55**	0.57
500	RMSD	2.96e-1	4.02e-1	1.59e-1	**6.73e-3**	5.25e-2	1.25e-2
	rRMSD	2.14e-1	2.81e-1	6.05e-2	**4.59e-3**	4.64e-3	4.73e-3
	rTime	1.68	1.74	1.50	0.50	1.03	0.81
	Time	182.09	9.10	6.16	769.39	**1.32**	1.48
1000	RMSD	3.47e-1	4.77e-1	1.83e-1	**5.35e-3**	5.57e-2	1.13e-2
	rRMSD	2.71e-1	2.52e-1	5.52e-2	3.63e-3	3.65e-3	**3.49e-3**
	rTime	14.89	15.11	12.00	1.97	10.32	5.22
	Time	601.92	56.65	24.49	15686.4	11.63	**10.03**
2000	RMSD	−−	4.47e-1	1.81e-1	−−	5.53e-2	**1.16e-2**
	rRMSD	−−	4.25e-1	2.21e-2	−−	3.32e-3	**3.12e-3**
	rTime	−−	82.17	82.35	−−	45.12	5.85
	Time	−−	470.32	122.45	−−	49.18	**34.68**

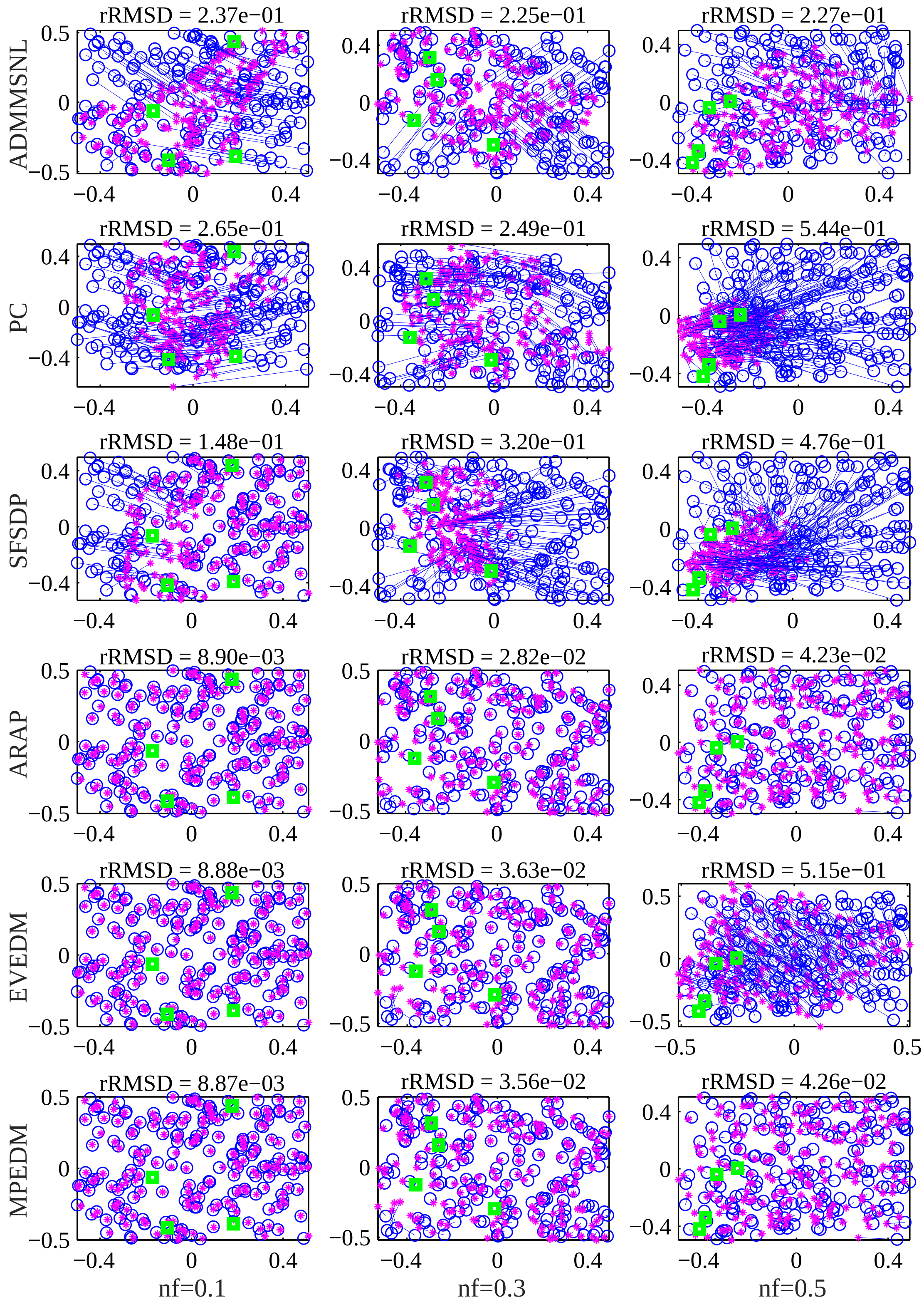


Figure 6.27: Localization for Example 5.2 with $n = 200, m = 4, R = 0.3$.

Finally, we test Example 5.1 with larger sizes $n \in \{300, 500, 1000, 2000\}$ and fixed $m = 4, R = 0.3$. Its average results were recorded in Table 6.19. When `nf` $= 0.1$, `ADMMSNL` and `PC` failed to render accurate embedding. Compared with `ARAP`, `EVEDM` and `MPEDM`, `SFSDP` generated lager `RMSD` and `rRMSD`. Again, `EVEDM` and `MPEDM` ran far faster than `ARAP`. When `nf` $= 0.7$, results recorded in Table 6.20 were different. `ARAP` and `MPEDM` still were able to produce accurate `RMSD` and `rRMSD`, while the former took extremely long time (16617 vs. 83 seconds). By contrast, `ADMMSNL` and `PC` again failed to solve problems. In addition, `EVEDM` got large `RMSD` but comparable `rRMSD` when $n \leq 1000$, and failed when $n = 2000$.

Table 6.20: Comparisons for Example 5.1 with $m = 4, R = 0.3, \texttt{nf} = 0.7$.

n		`ADMMSNL`	`PC`	`SFSDP`	`ARAP`	`EVEDM`	`MPEDM`
300	RMSD	2.80e-1	4.36e-1	3.27e-1	6.70e-2	2.08e-1	**5.04e-2**
	rRMSD	2.31e-1	3.60e-1	2.47e-1	5.48e-2	6.10e-2	**4.92e-2**
	rTime	0.75	0.83	0.83	0.29	0.47	0.38
	Time	107.48	1.74	83.73	123.18	**0.59**	7.49
500	RMSD	2.64e-1	4.53e-1	−−	4.24e-2	1.76e-1	**3.73e-2**
	rRMSD	1.94e-1	3.59e-1	−−	3.52e-2	3.47e-2	**3.23e-2**
	rTime	1.66	1.88	−−	0.47	0.87	0.67
	Time	177.24	5.13	−−	844.74	**1.31**	20.15
1000	RMSD	2.21e-1	4.52e-1	−−	2.84e-2	1.45e-1	**2.79e-2**
	rRMSD	9.69e-2	3.26e-1	−−	2.47e-2	2.93e-2	**2.40e-2**
	rTime	9.83	15.69	−−	1.41	7.78	2.54
	Time	599.30	41.55	−−	16617.1	**9.16**	83.64
2000	RMSD	−−	4.51e-1	−−	−−	2.26e-1	**2.13e-2**
	rRMSD	−−	3.35e-1	−−	−−	1.23e-1	**1.52e-2**
	rTime	−−	92.45	−−	−−	58.25	3.79
	Time	−−	274.90	−−	−−	**62.52**	303.43

6.3.3 Comparison on MC

As we mentioned before, the current implementations of `ADMMSNL` and `ARAP` do not support the embedding for $r \geq 3$ and thus are removed in the following comparison where another method `PPAS` will be added.

$\boxed{\text{Test on Example 5.5 under Rule 2.}}$ To see the performance of each method to this problem, we first test it with fixing $s = 6$ (i.e., $n = 6^3 = 216$), `nf` $= 0.1$ but altering $\sigma \in \{36, 38, \cdots, 48\}$. Notice that the percentage of available dissimilarities over all elements of Δ ascends from 32.47% to 39.87% along with increasing σ from 36 to 48, making problems more and more 'easier'.

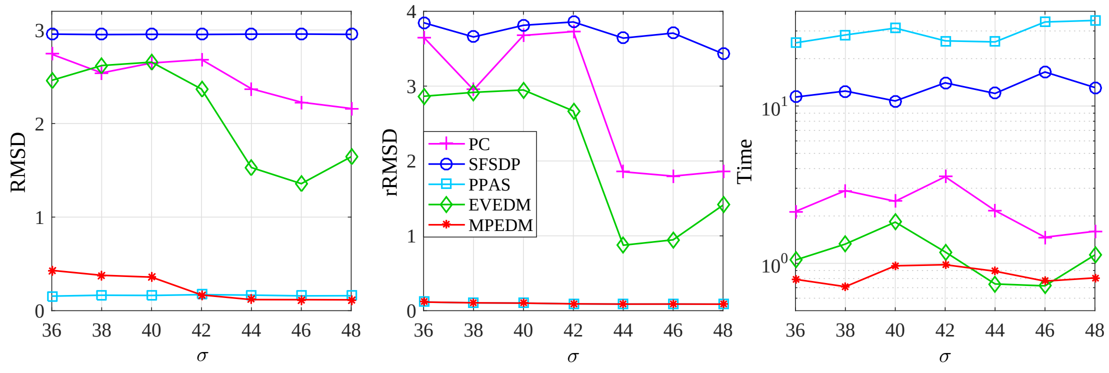


Figure 6.28: Average results for Example 5.5 with $s = 6, \texttt{nf} = 0.1$.

Average results were recoded in Figure 6.28. Clearly, `MPEDM` and `PPAS` outperformed the other three methods in terms of `RMSD` and `rRMSD`. The former generated the best `RMSD` when $\sigma \geq 42$ while the latter got the best `RMSD` when $\sigma \leq 42$, but they both obtained similar `rRMSD`. As for computational speed, `MPEDM` ran far faster than `PPAS`. By contrast, the other three methods failed to produce accurate embeddings due to worse `RMSD` and `rRMSD` obtained. Notice that refinement would not always make final results better, for example, `rRMSD` yielded by `SFSDP` was bigger than `RMSD` for each s.

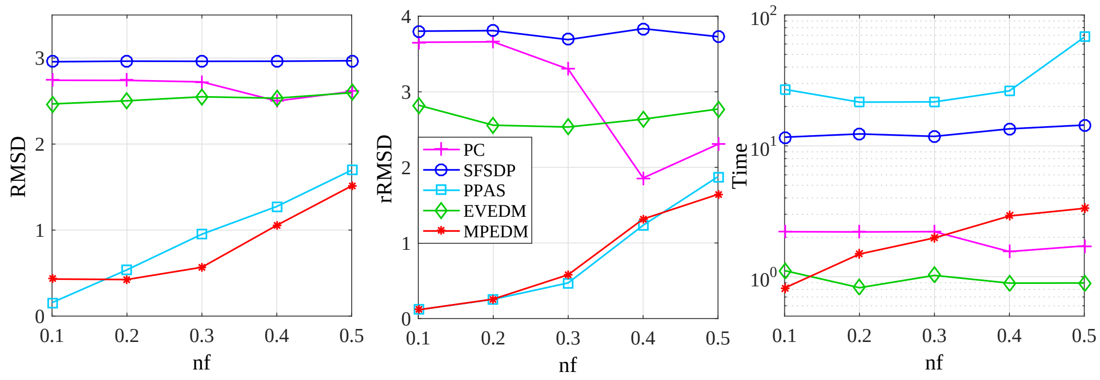


Figure 6.29: Average results for Example 5.5 with $s = 6, \sigma = s^2$.

We then test it with fixing $s = 6(n = 6^3), \sigma = s^2$ and varying the noise factor $\texttt{nf} \in \{0.1, 0.2, \cdots, 0.5\}$. As illustrated in Figure 6.29, in terms of `RMSD` and `rRMSD`, it can be clearly seen that `MPEDM` and `PPAS` were the joint winners. More detailed, our method rendered the best `RMSD` when $\texttt{nf} \geq 0.2$ and also ran much faster than `PPAS`. Obviously, the other three methods again failed to obtain desirable `RMSD` and `rRMSD` no matter how fast or slow they were. What is more, when `nf` ≥ 0.4, most of methods somewhat suffered from the refinement. For example, refinement made `rRMSD` of `MPEDM` worse than `RMSD` when `nf` ≥ 0.4.

Finally, for larger size problems with $n = s^3$ and $s \in \{7, 8, \dots, 13\}$, average results were presented in Figure 6.30, where we omitted results of `PPAS` when $s > 10$ since it cost too much time. It is worth mentioning that the rate of available dissimilarities over all elements of Δ declines from 26.78% to 14.83% when increasing s from 7 to 13, making problems more and more of 'difficulty'. Clearly `PC`, `SFSDP` and `EVEDM` failed to locate all atoms in $\mathbb{R}^3$. `PPAS` rendered the most accurate `RMSD` when $s \leq 10$ whilst `MPEDM` achieved the most accurate `RMSD` when $s > 10$ and the most accurate `rRMSD` for all cases. Most importantly, `PPAS` ran relatively slowly, consuming over 2000 seconds when $s \geq 10$. By contrast, `MPEDM` spent less than 50 seconds for all cases.

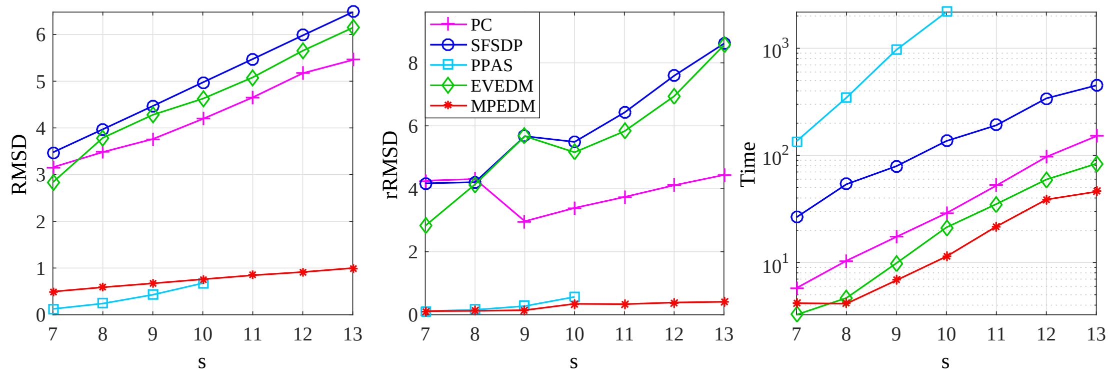


Figure 6.30: Average results for Example 5.5 with $n = s^3, \sigma = s^2, \texttt{nf} = 0.1$.

$\boxed{\text{Test on Example 5.6.}}$ In this test, we fixed $R = 6, c = 50\%$ and `nf` $= 0.1$. The generated embeddings by five methods for the three molecules `1GM2`, `1AU6` and `1LFB` were shown in Figure 6.31, where the true and estimated positions of the atoms were plotted by blue circles and pink stars respectively. Each pink star was linked to its corresponding blue circle by a pink line. For these three data, `MPEDM` and `PPAS` almost conformed the shape of the original data. Clearly, the other three methods failed to conform. The complete numerical results for the 12 problems were reported in Table 6.21. It can be clearly seen that `MPEDM` and `PPAS` performed significantly better regarding to `RMSD` and `rRMSD`. But importantly, the time used by `MPEDM` was just a small fraction of `PPAS` who spent relatively long time. For example, `MPEDM` only used 32.64 seconds for `2CLJ`, which is a very large data set with $n = 4189$.

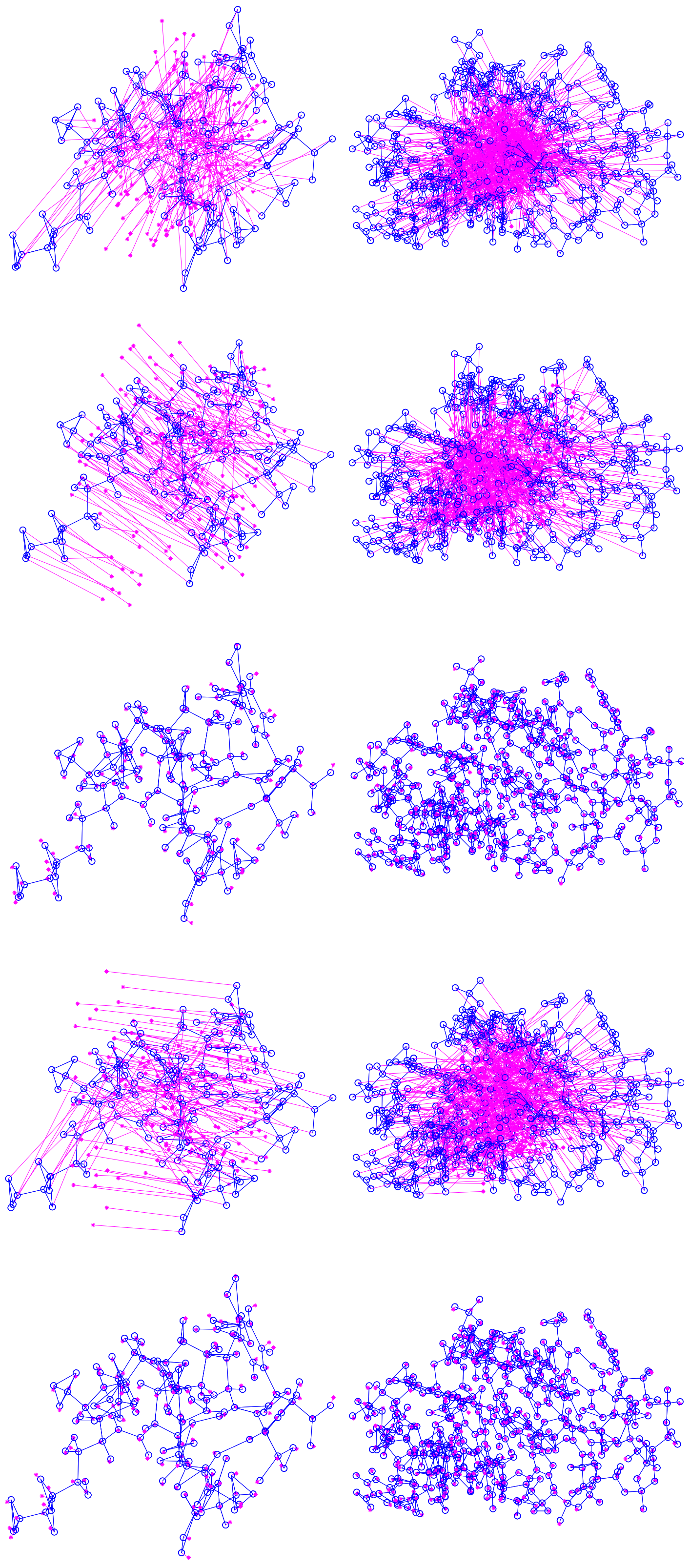


Figure 6.31: Molecular conformation. From top to bottom, the method is `PC`, `SFSDP`, `PPAS`, `EVEDM`, `MPEDM`. From left to right, the data is `1GM2`, `1AU6`, `1LFB`.

Table 6.21: Comparisons of five methods for Example 5.6.

		PC	SFSDP	PPAS	EVEDM	MPEDM
	RMSD	6.60	6.65	**0.41**	6.51	0.91
1GM2	rRMSD	7.07	6.92	**0.27**	7.41	0.35
$n = 166$	rTime	0.17	0.18	0.22	0.18	0.16
	Time	0.98	4.84	15.43	0.98	**0.27**
	RMSD	10.30	10.30	**2.89**	10.20	3.61
304D	rRMSD	10.70	10.80	**1.43**	10.80	2.50
$n = 237$	rTime	0.16	0.16	0.55	0.16	0.15
	Time	1.07	7.76	36.44	1.36	**0.23**
	RMSD	8.45	8.47	**0.53**	8.35	1.23
1PBM	rRMSD	9.13	8.91	**0.20**	9.28	0.21
$n = 388$	rTime	0.51	0.49	0.53	0.49	0.32
	Time	2.84	28.64	112.82	1.45	**0.54**
	RMSD	10.60	10.60	**0.54**	10.50	0.92
2MSJ	rRMSD	11.20	11.10	**0.30**	11.00	0.33
$n = 480$	rTime	0.40	0.39	0.54	0.39	0.32
	Time	2.32	118.60	196.12	1.47	**0.59**
	RMSD	9.30	9.31	**0.40**	9.20	0.67
1AU6	rRMSD	9.99	9.83	0.17	9.69	**0.16**
$n = 506$	rTime	0.70	0.68	0.30	0.69	0.35
	Time	4.12	47.68	262.28	1.47	**0.70**
	RMSD	13.40	13.40	1.56	13.30	**1.55**
1LFB	rRMSD	13.90	13.50	**0.54**	13.70	0.74
$n = 641$	rTime	0.49	0.49	1.63	0.48	0.37
	Time	2.93	132.96	956.44	1.64	**0.79**
	RMSD	12.30	12.30	4.30	12.20	**3.27**
104D	rRMSD	12.70	12.70	2.02	12.60	**1.26**
$n = 766$	rTime	0.89	0.86	3.40	0.87	**0.61**
	Time	5.04	72.16	2024.51	1.47	1.40
	RMSD	12.30	12.30	1.70	12.30	**1.58**
1PHT	rRMSD	12.90	12.60	**0.92**	12.60	0.99
$n = 814$	rTime	0.74	0.74	2.57	0.74	0.48
	Time	4.86	411.14	4726.96	1.71	**1.25**
	RMSD	14.20	14.20	**1.39**	14.10	1.48
1POA	rRMSD	14.50	14.60	**0.33**	14.60	0.45
$n = 914$	rTime	0.58	0.55	1.34	0.55	0.52
	Time	5.03	587.14	1623.43	1.99	**1.45**

	RMSD	14.30	14.30	−−	14.30	**1.23**
`1AX8`	rRMSD	14.70	14.50	−−	14.40	**0.50**
$n = 1003$	rTime	0.62	0.58	−−	0.59	0.34
	Time	5.78	1404.53	−−	1.54	**1.49**
	RMSD	20.20	−−	−−	20.20	**1.99**
`1RGS`	rRMSD	20.50	−−	−−	20.60	**0.68**
$n = 2015$	rTime	1.33	−−	−−	1.25	0.94
	Time	16.08	−−	−−	3.69	**5.71**
	RMSD	22.70	−−	−−	22.70	**1.54**
`2CLJ`	rRMSD	23.00	−−	−−	22.90	**0.65**
$n = 4189$	rTime	4.46	−−	−−	3.82	2.35
	Time	43.10	−−	−−	378.35	**32.64**

6.3.4 A Summary of Benchmark Methods

To end this section, we would like to summarize the performance of each method: `MPEDM`, `ADMMSNL`, `PC`, `SFSDP`, `ARAP` and `EVEDM`. The first summary is given to `MPEDM` under different f_{pq}. Clearly, `MPEDM`$_{11}$ is most robust to the noise but runs the lowest for most examples. By contrast, `MPEDM`$_{22}$ runs the fastest but with providing worst accuracy. `MPEDM` under f_{12} and f_{21} renders mediate performance. For the benchmark methods, we summarize their advantages and disadvantages as follows.

`ADMMSNL` works well when the network Δ has large numbers of known dissimilarities (i.e. R is large) but rans very slow when n is large (e.g. $n > 1000$). In addition, it performs poorly when R is small (e.g., $R < 0.3$). Notice that its current package version can not process three dimensional embedding like MC problems.

`PC` only works well when the network Δ has large numbers of known dissimilarities and rans very fast such as the sceneries $R \geq 1$. However, its performance is degraded significantly with more and more dissimilarities are missing in Δ. Moreover, it is not robust to the noise. For three dimensional embedding such as MC problems, it behaves very poorly.

`SFSDP` could achieve accurate embedding when R is large and anchors' number m is large. Its computational speed is heavily relied on the range R. When R is small, it could solve the problem with acceptable CPU time when n is not too large (e.g.

$n \leq 2000$) but with undesirable accuracy. When R is large, the CPU time its costs shoot up dramatically with the increasing of n. For three dimensional embedding, it performs very unappealing.

`ARAP` has ability to achieve embedding with desirable accuracy no matter how small the R is and also very robust to the noise. But it runs extremely slow particularly when $n > 1000$. Also its current package version can not processes the 3 dimensional embedding like MC problems.

`EVEDM` performs better when n is being larger. Its computational speed tends to be slower along with the ascending of R, and it is also very sensitive to the noise. What is more, it is not suitable for three dimensional embedding such as MC problems.

`PPAS` is designed for three dimensional embedding such as MC problems, with ability to render accurate embedding. However, it runs relatively slow and is difficult to process problems with $n > 1000$ objects.

Chapter 7

Two Extensions

This chapter introduces two extensions of the topic in this thesis. The first one is to take more general constraints of model (3.1) into consideration based on some applications. The second one is to solve the original problem (3.1) instead of its penalty model (3.15) through a popular approach, Proximal Alternating Direction Methods of Multipliers (pADMM) described in Subsection 1.3.7.

7.1 More General Model

The main EDM optimization of interest is (3.1), which brought us the penalized model (3.15) with only box constraints $L \leq D \leq U$. Now by adding an extra constraint in (3.1), we investigate a more general EDM optimization,

$$\begin{aligned} \min \quad & f(D) \\ s.t. \quad & g(D) = 0, \\ & D \in \Omega := \{D \in \mathbb{S}^n \mid L \leq D \leq U, \mathbb{A}(D) = \mathbf{0}\} \end{aligned} \tag{7.1}$$

where $\mathbb{A} : \mathbb{S}^n \to \mathbb{R}^d$ is a liner mapping. Hereafter, we always assume that $\Omega \neq \emptyset$. The above problem naturally leads to the following penalized model,

$$\begin{aligned} \min \quad & f(D) + \rho g(D) \\ s.t. \quad & D \in \Omega := \{D \in \mathbb{S}^n \mid L \leq D \leq U, \mathbb{A}(D) = \mathbf{0}\} \end{aligned} \tag{7.2}$$

7.1.1 Algorithmic Framework

Similar to Algorithm 4.1 in Table 4.1, we have the following algorithmic framework,

Table 7.1: Framework of Majorization-Projection method.

```
Algorithm 7.1: Majorization-Projection method via EDM

Step 1  (Input data) Dissimilarity matrix $\Delta$, weight matrix $W$, lower- and upper-
        bound matrix $L, U$, penalty parameter $\rho > 0$, and initial $D^0$. Set $k := 0$.
Step 2  (Update) Compute $D^k_K := -\Pi_{\mathbb{K}^n_+(r)}(-D^k)$ and update
            $D^{k+1} = \arg\min_{D\in\Omega} f(D) + (\rho/2)\|D - D^k_K\|^2.$     (7.3)
Step 3  (Convergence check) Set $k := k + 1$ and go to Step 2 until convergence.
```

Remark 7.1. *We have some comments regarding to Algorithm 7.1.*

- *One may notice that subproblem (7.3) is the major obstruction for the method. In Section 3.3, we know that if ρ is chosen properly (seen Table 4.2), the objective function of (7.3) is able to be convex for each f_{pq} where $p, q = 1, 2$. Then the linearity of $\mathbb{A}$ indicates Ω is convex and hence (7.3) has a unique solution,*

$$D^{k+1} = \Pi_\Omega\left(\widehat{D}^{k+1}\right), \tag{7.4}$$

where $\widehat{D}^{k+1}$ can be calculated explicitly (seen Section 3.3) by

$$\widehat{D}^{k+1} := \arg\min_{D\in\mathbb{S}^n} f(D) + (\rho/2)\|D - D^k_K\|^2.$$

- *All convergence results in Section 4.2 still hold under same assumptions since subproblem (7.3) is convex if σ is chosen properly;*

Therefore, we need an assumption on $\mathbb{A}$ as below.

Assumption 7.2. *$\mathbb{A}$ has the ability to make $\Pi_\Omega(\cdot)$ easily computed.*

7.1.2 One Application

Recall the problem of ES in Subsection 5.3.1. The problem is to find a matrix D that is nearest to $\Delta^{(2)}$ and satisfies

$$-D \in \mathbb{S}_h^n \cap \mathbb{K}_+^n(r)$$

Clearly, such constraints did not take the radius and center into account, which might render an less accurate embedding. Since all distances between each point $\{\mathbf{x}_1, \dots \mathbf{x}_{n-1}\}$ and the center $\mathbf{x}_n$ should be equal, a natural constraint for this problem is

$$D_{in} = D_{1n}, \quad i = 1, \dots, n-1, \tag{7.5}$$

which can be covered if $\mathbb{A}$ is defined by

$$\mathbb{A}(D) = (D_{2n} - D_{1n}, \cdots, D_{n-1,n} - D_{1n})^T. \tag{7.6}$$

In order to calculate $\Pi_\Omega(\cdot)$, a proposition below is required.

Proposition 7.3. *Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{a} \leq \boldsymbol{b}$. Assume that*

$$[\boldsymbol{a}, \boldsymbol{b}] \cap \{\boldsymbol{z} \in \mathbb{R}^n : z_1 = \cdots = z_n\} \neq \emptyset.$$

Denote $a_o := \max a_i, b_o := \min b_i$, then

$$\Pi_\Omega(\boldsymbol{x}) = \left[\Pi_{[a_o, b_o]}(\boldsymbol{x}^\top \boldsymbol{e}/n)\right] \boldsymbol{e}. \tag{7.7}$$

Proof Denote $\Gamma := \{\mathbf{z} \in \mathbb{R}^n : z_1 = \cdots = z_n\}$. It obviously has

$$[a_o\mathbf{e}, b_o\mathbf{e}] \cap \Gamma = [\mathbf{a}, \mathbf{b}] \cap \Gamma \neq \emptyset.$$

Then we have

$$\begin{aligned} \Pi_{[\mathbf{a},\mathbf{b}]\cap\Gamma}(\mathbf{x}) &= \Pi_{[a_o\mathbf{e}, b_o\mathbf{e}]\cap\Gamma}(\mathbf{x}) = \text{argmin}_{\mathbf{z}\in[a_o\mathbf{e}, b_o\mathbf{e}]\cap\Gamma} \|\mathbf{z} - \mathbf{x}\|^2 \\ &= \left[\text{argmin}_{z_1 \in [a_o, b_o]} \|z_1\mathbf{e} - \mathbf{x}\|^2\right] \mathbf{e} = \left[\Pi_{[a_o,b_o]}(\mathbf{x}^\top \mathbf{e}/n)\right] \mathbf{e}, \end{aligned}$$

which finishes the proof immediately. □

7.2 Solving the Original Problem

This section aims to solve the original model (7.1) instead of its penalty relaxation (7.2). By introducing an auxiliary variable $Z = -D$, since

$$g(D) = 0 \overset{(3.13)}{\Longleftrightarrow} -D \in \mathbb{K}^n_+(r) \Longleftrightarrow Z \in \mathbb{K}^n_+(r),$$

we equivalently rewrite (7.1) as

$$\begin{array}{ll} \min & f(D) \\ s.t. & D + Z = 0, \quad D \in \Omega, \quad Z \in \mathbb{K}^n_+(r). \end{array} \tag{7.8}$$

7.2.1 pADMM

Recall Subsection 1.3.7, the augmented Lagrange function of model (7.8) is

$$\mathcal{L}(D, Z, W) := f(D) - \langle W, D + Z\rangle + (\sigma/2)\|D + Z\|^2, \quad D \in \Omega, \ Z \in \mathbb{K}^n_+(r). \tag{7.9}$$

Then based on the procedure in Table 1.1, for already computed (D^k, Z^k, W^k), we update next iteration as

$$\begin{aligned} D^{k+1} &= \underset{D\in\Omega}{\text{argmin}} \ \mathcal{L}(D, Z^k, W^k), \\ &= \underset{D\in\Omega}{\text{argmin}} \ f(D) + (\sigma/2)\|D - (W^k/\sigma - Z^k)\|^2 && (7.10) \\ Z^{k+1} &= \underset{Z\in\mathbb{K}^n_+(r)}{\text{argmin}} \ \mathcal{L}(D^{k+1}, Z, W^k) \\ &= \underset{Z\in\mathbb{K}^n_+(r)}{\text{argmin}} \ \|Z - (W^k/\sigma - D^{k+1})\| \\ &= \Pi_{\mathbb{K}^n_+(r)}(W^k/\sigma - D^{k+1}) && (7.11) \\ W^{k+1} &= W^k - \tau\sigma(D^{k+1} + Z^{k+1}) && (7.12) \end{aligned}$$

Clearly, how to solve subproblem (7.10) is now an issue confronted us.

- $\Omega := \{D \in \mathbb{S}^n \mid L \le D \le U\}$. Similar to Subsection 4.1.2, we have closed form solutions for each $f = f_{pq}$. Denote $D^k_\sigma := W^k/\sigma - Z^k$, then its solution under each $f = f_{pq}$ can be calculated by replacing D^k_K by D^k_σ and replacing ρ by σ in (4.6-4.12) respectively.

- $\Omega := \{D \in \mathbb{S}^n \mid L \leq D \leq U, \mathbb{A}(D) = \mathbf{0}\}$. In Section 3.3, we know that if σ is chosen properly (being larger than a threshold $\sigma > \rho_0$ where $\rho_0 = \max_{\{(i,j):W_{ij}>0\}} \frac{W_{ij}}{4\delta_{ij}^3}$ if $f = f_{11}$ and $\rho_0 = 0$ otherwise), the objective function of (7.10) is able to be convex for each f_{pq} where $p, q = 1, 2$. Then the linearity of $\mathbb{A}$ indicates Ω is convex and hence (7.10) has a unique solution as well,

$$D^{k+1} = \Pi_{\Omega}\ (\overline{D}^{k+1}), \tag{7.13}$$

where $\overline{D}^{k+1}$ can be calculated explicitly (seen Section 3.3) by

$$\overline{D}^{k+1} := \arg\min_{D\in\mathbb{S}^n}\ f(D) + (\sigma/2)\|D - D_{\sigma}^k\|^2.$$

Overall, similar to Table 1.1, the framework of pADMM solving model (7.8) can be summarized in Table 7.2.

Table 7.2: Framework of pADMM for (7.8)

```
Algorithm 7.2: pADMM via EDM

Step 1   (Input data) Let $\sigma, \tau > 0$. Choose $(Z^0, W^0)$. Set $k \Leftarrow 0$.
Step 2   (Update) Compute $(D^{k+1}, Z^{k+1}, W^{k+1})$ by
             $D^{k+1} = \text{argmin}_{D\in\Omega}\ f(D) + (\sigma/2)\|D - (W^k/\sigma - Z^k)\|^2,$
             $Z^{k+1} = \Pi_{\mathbb{K}_+^n(r)}(W^k/\sigma - D^{k+1}),$
             $W^{k+1} = W^k - \tau\sigma(D^{k+1} + Z^{k+1}),$
Step 3   (Convergence check) Set $k := k + 1$ and go to Step 2 until convergence.
```

7.2.2 Current Convergence Results of Nonconvex pADMM

We have stated the global convergence results, namely, Theorem 1.3, of pADMM solving the convex model (1.20) in Subsection 1.3.7. However, our proposed model (7.8) is obviously non-convex due to the non-convex set $\mathbb{K}_+^n(r)$ or the non-convex objective function f_{11}. And thus Theorem 1.3 can not applied to derive the convergence results of pADMM solving model (7.8).

In order to ease reading, we recall the model (1.20) again

$$\begin{aligned} \min \quad & f_1(\mathbf{x}) + f_2(\mathbf{y}) \\ \text{s.t.} \quad & \mathbb{A}\mathbf{x} + \mathbb{B}\mathbf{y} = \mathbf{b}, \\ & \mathbf{x} \in \mathcal{X}, \quad \mathbf{y} \in \mathcal{Y}, \end{aligned} \tag{7.14}$$

To achieve the convergence results that the cluster point of the sequence generated by pADMM is the stationary point of above model in non-convex scenarios, a large number of scholars have proposed sorts of conditions in recent decade. Here we list some of them which are not relied on the iterates themselves.

Li and Pong (2015) proposed proximal ADMM and established the convergence property under following scenarios:

- f_1 is not necessarily convex, but twice continuously differentiable with a bounded Hessian matrix.
- f_2 is proper, closed, and not necessary convex.
- $\mathbb{A}$ has full row rank, $\mathbb{B}$ is the identity matrix.
- Either $\mathbb{A}$ is invertible and f_2 is coercive, or f_1 is coercive and f_2 is lower bounded.

Wang et al. (2015a) proposed the so-called Bregman ADMM and includes the standard ADMM as a special case. By setting all the auxiliary functions in their algorithm to zero, their assumptions for the standard ADMM reduce to the following scenarios:

- f_2 is strongly convex.
- f_1 is not necessarily convex, but is gradient Lipschitz differentiable and lower bounded. Moreover, There exists $c > 0$ such that $f_1 - c\| \bigtriangledown f_1\|^2$ is lower bounded.
- $\mathbb{A}$ is full row rank.
- Either $\mathbb{A}$ is square (which means it is invertible), or $f_1 - c\| \bigtriangledown f_1\|^2$ is coercive.

Hong et al. (2016) studied ADMM for non-convex consensus and sharing problem. Their assumptions for the sharing problem are

- f_1 is gradient Lipschitz continuous, either smooth non-convex or convex (possibly non-smooth).
- f_2 is gradient Lipschitz continuous.
- $f_1 + f_2$ is lower bounded over $\mathcal{X}$.
- $\mathbb{A}$ has full column rank, $\mathbb{B}$ is the identity matrix.
- $\mathbf{x} \in \Omega$ with Ω being a closed convex set.

Wang et al. (2015b) considered following non-convex case

- f_1 is proper and continuous, possibly non-smooth and not necessarily convex, but either is the so-called restricted prox-regular or continuous and piecewise linear.
- f_2 is proper and not necessarily convex, but is gradient Lipschitz continuous.
- $f_1 + f_2$ is coercive over the feasible set $\Omega := \{(\mathbf{x}, \mathbf{y}) \mid \mathbb{A}\mathbf{x} + \mathbb{B}\mathbf{y} = \mathbf{b}\}$, that is, $f_1(\mathbf{x}) + f_2(\mathbf{y}) \to \infty$ when $\|(\mathbf{x}, \mathbf{y})\| \to \infty$ and $(\mathbf{x}, \mathbf{y}) \in \Omega$.
- $\mathrm{Im}(\mathbb{A}) \subseteq \mathrm{Im}(\mathbb{B})$ where $\mathrm{Im}(\mathbb{A})$ is the image of $\mathbb{A}$.

Yang et al. (2017) tried to solve the non-convex problem under following settings

- $f_1(\mathbf{x}) := \phi(\mathbf{x}_1) + \psi(\mathbf{x}_2)$ with $\mathbf{x} = [\mathbf{x}_1^\top \ \mathbf{x}_2^\top]^\top$ and ϕ is proper closed nonnegative functions and convex. ψ is proper closed nonnegative functions, but possibly non-convex, non-smooth, and non-Lipschitz.
- f_2 is a quadratic function.
- $\mathbb{A} = [\mathbb{A}_1 \mathbb{A}_2]$ with $\mathbb{A}_1$ and $\mathbb{A}_2$ being injective, i.e. $\mathbb{A}_1^* \mathbb{A}_1 \succ \mathbf{0}$ and $\mathbb{A}_2^* \mathbb{A}_2 \succ \mathbf{0}$. Moreover $\mathbb{B}$ is the identity matrix.

Remark 7.4. *Wang et al. (2015b) claimed that their proposed conditions are weaker than those of (Hong et al., 2016; Wang et al., 2015a; Li and Pong, 2015), which means they got the best results for convergence property of ADMM when it is applied in non-convex scenarios. Let rewrite the model (7.8) as*

$$\begin{aligned} \mathit{min} \quad & \underbrace{f(D) + I_\Omega(D)}_{:=f_1} + \underbrace{I_{\mathbb{K}^n_+(r)}(Z)}_{:=f_2} \\ \mathit{s.t.} \quad & D + Z = 0, \end{aligned}$$

where $I_\Omega(D)$ is the indicator function, defined by $I_\Omega(D) = 0$ if $D \in \Omega$, $I_\Omega(D) = +\infty$ otherwise. Then, the the above model we consider does not meet all requirements of Wang et al. (2015b). Clearly, $f_2 = I_{\mathbb{K}^n_+(r)}(Z)$ is not gradient Lipschitz continuous.

In addition, above model does not meet all requirements of Wang et al. (2015b) since $f_2 = I_{\mathbb{K}^n_+(r)}(Z)$ is not quadratic and $f(D)$ (corresponding ϕ) is not convex when $f = f_{11}$.

7.2.3 Numerical Experiments

a) Implementation. The starting points for D^0 is same as that in Subsection 6.1.2, and then let $Z^0 = -D^0$ and $W^0 = \mathbf{0}$. Similar to Subsection 6.3, we monitor two quantities:

$$\texttt{Kprog}_k(\sigma) \quad := \quad \frac{\|D^k + Z^k\|}{1 + \sigma + \|D^k\| + \|Z^k\|}. \tag{7.15}$$

where σ is the parameter in(7.9), and

$$\texttt{Fprog}_k(\sigma) := \frac{f_\sigma(D^{k-1}) - f_\sigma(D^k)}{1 + \sigma + f_\sigma(D^{k-1})} \tag{7.16}$$

Next we would like to explain how to choose the parameter σ, similar to update ρ_{k+1} as (6.1), we initial $\sigma_0 = (1 + \max\ \delta_{ij})e^\kappa \log(n)$, where κ counts the number of non-zero elements of Δ, and update it as

$$\sigma_{k+1} = \begin{cases} 1.5\sigma_k, & \text{if } \texttt{Kprog}_k(\sigma_{k-1}) > \texttt{Ktol}, \texttt{Fprog}_k(\sigma_{k-1}) < \texttt{Ftol}, \\ 0.5\sigma_k, & \text{if } \texttt{Fprog}_k(\sigma_{k-1}) > \texttt{Ftol}, \texttt{Kprog}_k(\sigma_{k-1}) \le 0.2\texttt{Ktol}, \\ \sigma_k, & \text{otherwise}, \end{cases}$$

where Ftol and Ktol are chosen as

$$\texttt{Ftol} = \ln(\kappa) \times 10^{-4}, \qquad \texttt{Ktol} = \begin{cases} 10^{-2} & \text{if } n \ge 100, \\ 10^{-4} & \text{if } n < 100. \end{cases}$$

Taking two quantities into consideration, we terminate pADMM when

$$(\texttt{Fprog}_k(\sigma_k) \le \texttt{Ftol} \ \text{ and } \ \texttt{Kprog}_k(\sigma_k) \le \texttt{Ktol}) \quad \text{or} \quad k > 2000.$$

b) Self-Comparisons. To see the performance of Algorithm 7.2, we apply it under four objective functions f_{pq} into solving Examples 5.1, 5.4 and 5.6. As shown in Table

7.3, it was slightly slower and less accurate than MPEDM (see Table 6.5). Moreover, results under different f_{pq} varied a lot, while MPEDM produce more stable ones.

Table 7.3: ADMM on solving Example 5.1 with $m = 4, R = 0.2, \texttt{nf} = 0.1$.

n		1000	2000	3000	4000	5000
RMSD	f_{22}	8.23e-2	9.86e-2	6.23e-2	6.12e-2	5.18e-2
	f_{21}	7.88e-2	3.97e-2	4.33e-2	4.80e-2	4.33e-2
	f_{12}	7.88e-2	9.40e-2	8.56e-2	7.09e-2	5.41e-2
	f_{11}	5.05e-2	5.21e-2	4.50e-2	3.91e-2	3.43e-2
rRMSD	f_{22}	4.07e-3	7.92e-3	6.67e-3	5.77e-3	5.14e-3
	f_{21}	4.31e-3	6.59e-3	4.57e-3	5.98e-3	4.03e-3
	f_{12}	3.71e-3	9.65e-3	5.26e-3	5.49e-3	5.39e-3
	f_{11}	3.07e-3	7.92e-3	6.67e-3	5.77e-3	3.21e-3
Time	f_{22}	8.13	25.55	76.47	162.83	285.13
	f_{21}	7.62	22.17	64.14	131.66	239.51
	f_{12}	8.49	35.29	92.57	168.46	278.72
	f_{11}	8.40	38.38	99.26	187.58	340.13
rTime	f_{22}	3.23	4.67	17.80	17.72	26.64
	f_{21}	3.42	5.20	13.76	14.39	25.78
	f_{12}	3.36	6.83	18.46	17.23	22.62
	f_{11}	2.31	6.81	13.94	16.34	24.71

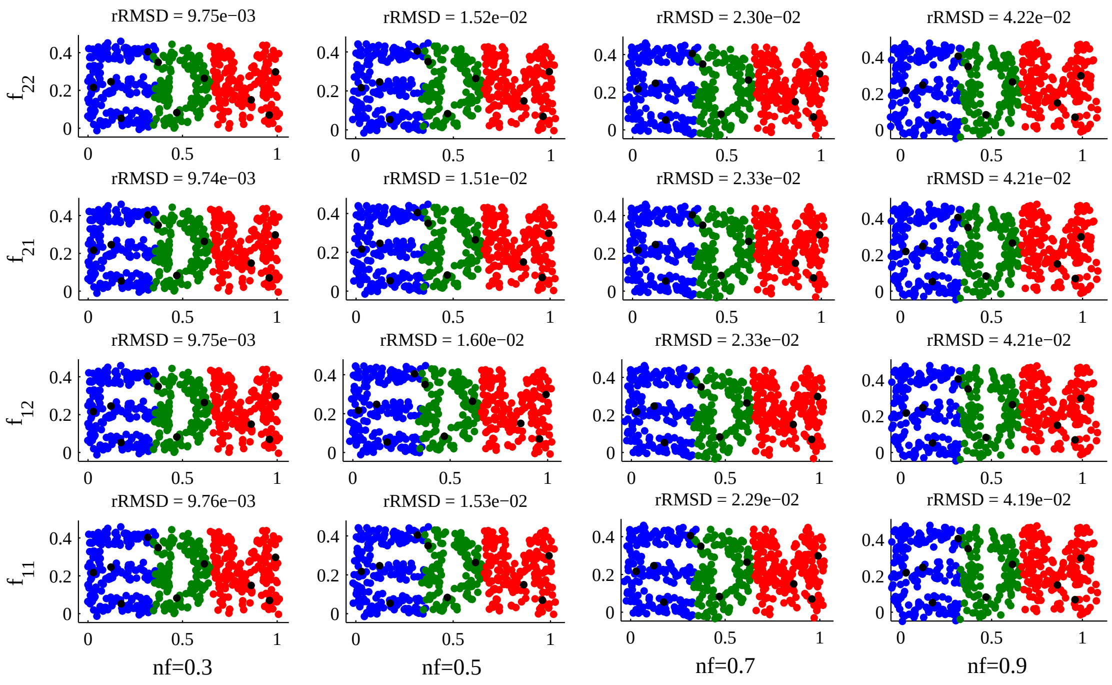


Figure 7.1: ADMM on solving Example 5.4 with $n = 500, m = 10, R = 0.3$.

We then see how noise ratio `nf` would affect on the performance of `ADMM` under each f_{pq}. Results were presented Figure 7.1, it can be clearly seen that their performance had no big difference. Compared with `MPEDM` solving such example (see Figure 6.8), it seemed that `ADMM` got the better results when `nf`≥ 0.3.

Finally, we solve Example 5.6 by using `ADMM` under each f_{pq}. As demonstrated in Table 7.4, f_{21} enabled `ADMM` to produced the most accurate results for most data, while f_{22} made `ADMM` run the fastest. There was no big difference of rRMSD after refinement.

Table 7.4: `ADMM` on solving Example 5.6.

		f_{22}	f_{21}	f_{12}	f_{11}
	RMSD	0.882	**0.734**	0.881	0.745
`1GM2`	rRMSD	0.238	0.238	0.238	0.238
$n = 166$	rTime	0.116	0.100	0.100	0.099
	Time	0.343	0.234	**0.158**	0.299
	RMSD	3.477	**3.354**	3.476	3.468
`304D`	rRMSD	2.522	2.522	2.522	2.522
$n = 237$	rTime	0.092	0.091	0.091	0.091
	Time	**0.152**	0.206	**0.152**	0.157
	RMSD	1.024	1.018	1.023	**0.930**
`1PBM`	rRMSD	0.223	0.223	0.223	0.223
$n = 388$	rTime	0.256	0.245	0.253	0.252
	Time	0.450	**0.433**	0.458	0.739
	RMSD	0.897	**0.766**	0.897	1.166
`2MSJ`	rRMSD	0.255	0.255	0.255	0.255
$n = 480$	rTime	0.216	0.240	0.241	0.241
	Time	**0.469**	0.800	0.532	1.511
	RMSD	0.678	**0.569**	0.666	0.830
`1AU6`	rRMSD	0.172	0.173	0.172	0.172
$n = 506$	rTime	0.206	0.160	0.197	0.171
	Time	**0.531**	0.680	0.595	1.249
	RMSD	1.468	**1.425**	1.468	1.465
`1LFB`	rRMSD	0.531	0.523	0.531	0.531
$n = 641$	rTime	0.277	0.277	0.276	0.278
	Time	**0.732**	0.958	0.771	0.753
	RMSD	2.959	2.909	2.956	**2.861**
`104D`	rRMSD	1.172	1.172	1.173	1.172
$n = 766$	rTime	0.499	0.505	0.495	0.501
	Time	**1.225**	2.195	1.291	2.751

	RMSD	1.588	**1.517**	1.587	1.568
1PHT	rRMSD	1.001	1.042	1.002	1.001
$n = 814$	rTime	0.427	0.436	0.429	0.425
	Time	**1.201**	2.699	1.276	1.404
	RMSD	1.459	1.402	1.458	**1.394**
1POA	rRMSD	0.401	0.391	0.401	0.401
$n = 914$	rTime	0.407	0.411	0.410	0.409
	Time	**1.362**	2.189	1.443	2.425
	RMSD	1.288	1.290	1.288	**1.286**
1AX8	rRMSD	0.728	0.488	0.468	0.728
$n = 1003$	rTime	0.094	0.370	0.404	0.375
	Time	**1.279**	1.671	1.717	2.171
	RMSD	1.892	**1.683**	1.891	1.790
1RGS	rRMSD	0.604	0.493	0.604	0.604
$n = 2015$	rTime	0.897	0.984	0.648	0.781
	Time	**5.951**	20.40	5.994	7.700
	RMSD	1.552	**1.479**	1.552	1.550
2CLJ	rRMSD	0.724	0.579	0.724	0.724
$n = 4189$	rTime	1.830	3.099	1.730	2.624
	Time	**24.25**	89.64	25.28	35.51

7.2.4 Future Proposal

According to above literature review, there are two main difficulties are confronted us.

- When it comes to establishing the convergence results, one is supposed to investigate the optimality condition of subproblem (7.11). Namely, for a given Z_0, consider the problem $\min_{Z\in\mathbb{K}^n_+(r)}\|Z - Z_0\|^2$. The difficulty arises from the rank constraint $\text{rank}(JZJ) \le r$ in $\mathbb{K}^n_+(r)$.
- How to establish the convergence property of ADMM described in Algorithm 7.2? Apparently, one of difficulties also stems from the constraints $Z \in \mathbb{K}^n_+(r)$.

Chapter 8

Conclusion

This thesis cast a general model (3.1) containing four objective functions from the domain of EDM optimization, with ability to cover four kinds of existing topics in MDS researches: stress minimization, squared stress minimization, robust MDS and robust Euclidean embedding described in Section 2.2. The model has been notoriously known to be non-smooth, non-convex and one of its objective functions is also non-Lipschtiz continuous. Its extra difficulty arose from the constraints that the variable being an EDM with low rank embedding dimension, but could be eliminated when such constraints were penalized. The relationship between the general model and its penalization (3.15) was then established, yielding our first contribution of this thesis, seen Theorem 3.3.

The main contents of this thesis were designing an efficient numerical method dubbed as `MPEDM` to tackle the penalty model whose four objective functions were able to be majorized by strictly convex functions provided that the penalty parameter was above a certain threshold. Then every step of `MPEDM` projected a unique solution of each convex majorized function onto a box set, which brought us the second contribution of this thesis that all projections turned out to enjoy closed forms (seen Section 3.3), despite some of the majorized functions seemed to be very complicated. Finally, traditional convergence analysis of majorization minimization can not be applied into `MPEDM` due to unappealing properties of the penalty model. However, we proved that the generated sequence converged to a stationary point in a general way (seen Section 4.2), and all involved assumptions were very easy to be satisfied when each objective function was specified (seen Section 4.3). This was the third contribution.

A large number of numerical experiments were then implemented to demonstrate the performance of MPEDM. Through comparing it among itself under four objective functions, general speaking, squared stress allowed MPEDM to run the fastest but with lowest accuracy, whilst robust Euclidean embedding enabled MPEDM to render the most accurate embedding but consume the longest time. When MPEDM under objective function in the sense of robust Euclidean embedding was against with other state-of-the-art methods, the desirable accuracy and fast computational speed manifested that it was very competitive, especially in big data settings. For example, it only consumed 42 seconds to conform a molecule with 4189 atoms in Table 6.21, a very large size.

Finally, the proposed model (3.1) was relatively easy to be extended to process more complicated problems, such as ones with extra constraints, as long as the projection onto those constraints were computable, seen Section 7.1. Upon the derivation of closed form solution under each objective function, we could solve the original model (3.1) straightforwardly by taking advantage of ADMM rather than dealing with its penalization. However, the less-than-ideal properties of the original model highlight the difficulty of proving the algorithmic convergence, which could leave as promising future research.

References

Agarwal, A., Phillips, J. M., and Venkatasubramanian, S. (2010). Universal multi-dimensional scaling. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1149–1158. ACM.

An, L. T. H. and Tao, P. D. (2003). Large-scale molecular optimization from distance matrices by a dc optimization approach. *SIAM Journal on Optimization*, 14(1):77–114.

Bai, S. and Qi, H. (2016). Tackling the flip ambiguity in wireless sensor network localization and beyond. *Digital Signal Processing*, 55:85–97.

Bai, S., Qi, H.-D., and Xiu, N. (2015). Constrained best euclidean distance embedding on a sphere: a matrix optimization approach. *SIAM Journal on Optimization*, 25(1):439–467.

Beck, A. and Pan, D. (2012). On the solution of the gps localization and circle fitting problems. *SIAM Journal on Optimization*, 22(1):108–134.

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., et al. (2002). The protein data bank. *Acta Crystallographica Section D: Biological Crystallography*, 58(6):899–907.

Biswas, P., Liang, T.-C., Toh, K.-C., Ye, Y., and Wang, T.-C. (2006). Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE transactions on automation science and engineering*, 3(4):360–371.

Biswas, P. and Ye, Y. (2004). Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 46–54. ACM.

Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Boyarski, A., Bronstein, A. M., and Bronstein, M. M. (2017). Subspace least squares multidimensional scaling. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 681–693. Springer.

Buja, A., Swayne, D. F., Littman, M. L., Dean, N., Hofmann, H., and Chen, L. (2008). Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 17(2):444–472.

Burton, D. (2011). *The history of mathematics: An introduction*. McGraw-Hill Companies.

Cayton, L. and Dasgupta, S. (2006). Robust euclidean embedding. In *Proceedings of the 23rd international conference on machine learning*, pages 169–176. ACM.

Chen, Y., Xiu, N., and Peng, D. (2014). Global solutions of non-lipschitz $s_2 - s_p$ minimization over the positive semidefinite cone. *Optimization Letters*, 8(7):2053–2064.

Costa, J. A., Patwari, N., and Hero III, A. O. (2006). Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):39–64.

Cox, T. F. and Cox, M. A. (2000). *Multidimensional scaling*. CRC press.

De, L. J., Barra, J. R., Brodeau, F., Romier, G., and Van, C. B. (1977). Applications of convex analysis to multidimensional scaling. *Recent Developments in Statistics*.

De Leeuw, J. and Mair, P. (2011). Multidimensional scaling using majorization: Smacof in r.

Ding, C. and Qi, H.-D. (2017). Convex optimization learning of faithful euclidean distance representations in nonlinear dimensionality reduction. *Mathematical Programming*, 164(1-2):341–381.

Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627.

Drusvyatskiy, D., Krislock, N., Voronin, Y.-L., and Wolkowicz, H. (2017). Noisy euclidean distance realization: robust facial reduction and the pareto frontier. *SIAM Journal on Optimization*, 27(4):2301–2331.

Fazel, M., Pong, T. K., Sun, D., and Tseng, P. (2013). Hankel matrix rank minimization with applications to system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34(3):946–977.

France, S. L. and Carroll, J. D. (2011). Two-way multidimensional scaling: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(5):644–661.

Gaffke, N. and Mathar, R. (1989). A cyclic projection algorithm via duality. *Metrika*, 36(1):29–54.

Gao, Y. (2010). *Structured low rank matrix optimization problems: a penalty approach.* PhD thesis.

Glunt, W., Hayden, T., and Liu, W.-M. (1991). The embedding problem for predistance matrices. *Bulletin of Mathematical Biology*, 53(5):769–796.

Glunt, W., Hayden, T. L., Hong, S., and Wells, J. (1990). An alternating projection algorithm for computing the nearest euclidean distance matrix. *SIAM Journal on Matrix Analysis and Applications*, 11(4):589–600.

Glunt, W., Hayden, T. L., and Raydan, M. (1993). Molecular conformations from distance matrices. *Journal of Computational Chemistry*, 14(1):114–120.

Gower, J. C. (1985). Properties of euclidean and non-euclidean distance matrices. *Linear Algebra and its Applications*, 67:81–97.

Groenen, P. (1993). A comparison of two methods for global optimization in multidimensional scaling. pages 145–155.

Hartigan, J. A. (1975). Clustering algorithms. *Wiley.*

Heiser, W. J. (1988). Multidimensional scaling with least absolute residuals. *Classification and related methods of data analysis*, pages 455–462.

Hong, M., Luo, Z.-Q., and Razaviyayn, M. (2016). Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364.

Jiang, K., Sun, D., and Toh, K.-C. (2013). Solving nuclear norm regularized and semidefinite matrix least squares problems with linear equality constraints. In *Discrete Geometry and Optimization*, pages 133–162. Springer.

Jiang, K., Sun, D., and Toh, K.-C. (2014). A partial proximal point algorithm for nuclear norm regularized matrix least squares problems. *Mathematical Programming Computation*, 6(3):281–325.

Kanzow, C. and Qi, H.-D. (1999). A qp-free constrained newton-type method for variational inequality problems. *Mathematical Programming*, 85(1):81–106.

Karbasi, A. and Oh, S. (2013). Robust localization from incomplete local information. *IEEE/ACM Transactions on Networking*, 21(4):1131–1144.

Kearsley, A. J., Tapia, R. A., and Trosset, M. W. (1995). The solution of the metric stress and sstress problems in multidimensional scaling using newton's method. Technical report, RICE UNIV HOUSTON TX DEPT OF COMPUTATIONAL AND APPLIED MATHEMATICS.

Kim, S., Kojima, M., Waki, H., and Yamashita, M. (2012). Algorithm 920: Sfsdp: a sparse version of full semidefinite programming relaxation for sensor network localization problems. *ACM Transactions on Mathematical Software (TOMS)*, 38(4):27.

Korkmaz, S. and van der Veen, A.-J. (2009). Robust localization in sensor networkswith iterative majorization techniques. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 2049–2052. IEEE.

Kruskal, J. B. (1964). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129.

Lawrence, J., Arietta, S., Kazhdan, M., Lepage, D., and O'Hagan, C. (2011). A user-assisted approach to visualizing multidimensional images. *IEEE transactions on visualization and computer graphics*, 17(10):1487–1498.

Li, G. and Pong, T. K. (2015). Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization*, 25(4):2434–2460.

Mandanas, F. D. and Kotropoulos, C. L. (2017). Robust multidimensional scaling using a maximum correntropy criterion. *IEEE Transactions on Signal Processing*, 65(4):919–932.

Moré, J. J. and Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7(3):814–836.

Nesterov, Y. (1998). Introductory lectures on convex programming volume i: Basic course. *Lecture notes*.

Nocedal, J. and Wright, S. J. (2006). *Sequential quadratic programming*. Springer.

Oguz-Ekim, P., Gomes, J. P., Xavier, J., and Oliveira, P. (2011). Robust localization of nodes and time-recursive tracking in sensor networks using noisy range measurements. *IEEE Transactions on Signal Processing*, 59(8):3930–3942.

Peng, D., Xiu, N., and Yu, J. (2017). $s_{1/2}$ regularization methods and fixed point algorithms for affine rank minimization problems. *Computational Optimization and Applications*, 67(3):543–569.

Piovesan, N. and Erseghe, T. (2016). Cooperative localization in wsns: a hybrid convex/non-convex solution. *IEEE Transactions on Signal and Information Processing over Networks*.

Pong, T. K. (2012). Edge-based semidefinite programming relaxation of sensor network localization with lower bound constraints. *Computational Optimization and Applications*, 53(1):23–44.

Qi, H. and Yuan, X. (2014). Computing the nearest euclidean distance matrix with low embedding dimensions. *Mathematical programming*, 147(1-2):351–389.

Qi, H.-D. (2013). A semismooth newton method for the nearest euclidean distance matrix problem. *SIAM Journal on Matrix Analysis and Applications*, 34(1):67–93.

Qi, H.-D., Xiu, N., and Yuan, X. (2013). A lagrangian dual approach to the single-source localization problem. *IEEE Transactions on Signal Processing*, 61(15):3815–3826.

Rockafellar, R. T. and Wets, R. J.-B. (2009). *Variational analysis*, volume 317. Springer Science & Business Media.

Rosman, G., Bronstein, A. M., Bronstein, M. M., Sidi, A., and Kimmel, R. (2008). Fast multidimensional scaling using vector extrapolation. *SIAM J. Sci. Comput*, 2.

Schoenberg, I. J. (1935). 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*, pages 724–732.

Shang, Y. and Ruml, W. (2004). Improved mds-based localization. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2640–2651. IEEE.

Shepard, R. N. (1962). The analysis of proximities: multidimensional scaling with an unknown distance function. i. *Psychometrika*, 27(2):125–140.

Soares, C., Xavier, J., and Gomes, J. (2015). Simple and fast convex relaxation method for cooperative localization in sensor networks using range measurements. *IEEE Transactions on Signal Processing*, 63(17):4532–4543.

Spence, I. and Lewandowsky, S. (1989). Robust multidimensional scaling. *Psychometrika*, 54(3):501–513.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Toh, K.-C. (2008). An inexact primal–dual path following algorithm for convex quadratic sdp. *Mathematical programming*, 112(1):221–254.

Torgerson, W. S. (1952). Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419.

Trejos, J., Castillo, W., González, J., and Villalobos, M. (2000). Application of simulated annealing in some multidimensional scaling problems. In *Data Analysis, Classification, and Related Methods*, pages 297–302. Springer.

Tseng, P. (2007). Second-order cone programming relaxation of sensor network localization. *SIAM Journal on Optimization*, 18(1):156–185.

Wang, F., Cao, W., and Xu, Z. (2015a). Convergence of multi-block bregman admm for nonconvex composite problems. *arXiv preprint arXiv:1505.03063*.

Wang, Y., Yin, W., and Zeng, J. (2015b). Global convergence of admm in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*.

Wang, Z., Zheng, S., Ye, Y., and Boyd, S. (2008). Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM Journal on Optimization*, 19(2):655–673.

Weinberger, K. Q. and Saul, L. K. (2006). Unsupervised learning of image manifolds by semidefinite programming. *International journal of computer vision*, 70(1):77–90.

Xing, F. (2003). Investigation on solutions of cubic equations with one unknown. *J. Central Univ. Nat.(Natural Sci. Ed.)*, 12(3):207–218.

Xu, Z., Chang, X., Xu, F., and Zhang, H. (2012). $s_{1/2}$ regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on neural networks and learning systems*, 23(7):1013–1027.

Yang, L., Pong, T. K., and Chen, X. (2017). Alternating direction method of multipliers for a class of nonconvex and nonsmooth problems with applications to background/-foreground extraction. *SIAM Journal on Imaging Sciences*, 10(1):74–110.

Young, G. and Householder, A. S. (1938). Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22.

Zhang, L., Liu, L., Gotsman, C., and Gortler, S. J. (2010). An as-rigid-as-possible approach to sensor network localization. *ACM Transactions on Sensor Networks (TOSN)*, 6(4):35.

Zhang, L., Wahba, G., and Yuan, M. (2016). Distance shrinkage and euclidean embedding via regularized kernel estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(4):849–867.

Zhen, W. (2007). Large scale sensor network localization. *Department of Statistics, Stanford University.*

Zhou, S., Xiu, N., and Qi, H.-D. (2018a). A fast matrix majorization-projection method for penalized stress minimization with box constraints. *IEEE Transactions on Signal Processing*, 66(16):4331– 4346.

Zhou, S., Xiu, N., and Qi, H.-D. (2018b). Robust euclidean embedding via edm optimization. *https://www.researchgate.net/publication/323945500.*