CME 307: SENSOR NETWORK LOCALIZATION PROJECT

RILEY JUENEMANN, ISHANI KARMARKAR, AND ADRIENNE PROPP

1. Introduction

The Sensor Network Localization (SNL) Problem, which seeks to best estimate the position of a group of wireless sensors based on limited information, plays a key role in many engineering applications. Networks consisting of low cost and low power sensors can be deployed for a range of purposes, for example traffic monitoring, wireless communications, and environmental monitoring (e.g. water quality estimation and wildfire tracking). SNL methods can even be applied to determine the molecular conformation of proteins, important to unlocking mysteries about their functions and properties [4]. For essentially all of these applications, it is important to associate a location with the data collected from each sensor. While the sensors have the ability to communicate within a limited range, in practice, equipping each device with GPS capabilities is not feasible. For this reason, the ability to localize these sensors given only pairwise distances between sensors and the actual position of a few sensors (which we call anchors) is critical. The general problem can be stated as the following.

Problem 1. *Suppose we have n_s sensor locations $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_s} \in \mathbb{R}^d$; n_a anchors $\boldsymbol{a}_1, ..., \boldsymbol{a}_{n_a} \in \mathbb{R}^d$; and a fixed threshold radius $r \in \mathbb{R}$. Given the following information:*

- *The locations of the n_a anchors: $\boldsymbol{a}_1, ..., \boldsymbol{a}_{n_a} \in \mathbb{R}^d$;*
- *Sensor-sensor distance data within a radius r of each sensor: $d_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$ for all $(i,j) \in N_x := \{(i,j) \in [n_s] \times [n_s] : i < j, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leqslant r\}$; and*
- *Sensor-anchor distance data within a radius r of each sensor: $\widehat{d}_{ij} = \|\boldsymbol{x}_i - \boldsymbol{a}_j\|_2$ for all $(i,j) \in N_a := \{(i,j) \in [n_s] \times [n_a] : \|\boldsymbol{x}_i - \boldsymbol{a}_j\|_2 \leqslant r\}$;*

determine the locations of the n_s sensors $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_s} \in \mathbb{R}^d$

The Sensor Network Localization (SNL) Problem can be obtained by reformulating Problem 1 into a system of nonlinear equations whose solution corresponds to a set of sensor locations $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_s} \in \mathbb{R}^d$ which are consistent with the given sensor-sensor distance data and sensor-anchor distance data. Formally, the SNL problem is as follows.

Problem 2 (Sensor Network Localization (SNL))**.** *Given data as in Problem 1, find $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_s} \in \mathbb{R}^d$ such that*

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = d_{ij}^2, \qquad \forall (i,j) \in N_x, i < j,$$
$$\|\boldsymbol{x}_i - \boldsymbol{a}_j\|_2^2 = \widehat{d}_{ij}^2, \qquad \forall (i,j) \in N_a.$$

Throughout this report, we consider two metrics for error in the reconstructed sensor location, the two-norm error between the true and reconstructed sensor locations (the two norm of the vector of errors in all coordinates of the reconstructed sensor locations) and the ∞-norm error between the true and reconstructed sensor locations (the maximum error in any coordinate of the reconstructed sensor location). The two-norm error roughly measures the typical error in any coordinate of the reconstructed sensors, while the ∞-norm error measures the inaccuracy in the worst-approximated sensor location. Formally,

$$\textbf{2-norm error} = \sum_{i=1}^{n_s} \sum_{j=1}^{d} (\widetilde{\boldsymbol{z}}_{i_j} - \boldsymbol{x}_{i_j})^2, \qquad \textbf{∞-norm error} = \max_{i \in [n_s], j \in [d]} |\widetilde{\boldsymbol{z}}_{i_j} - \boldsymbol{x}_{i_j}|,$$

where $\widetilde{z}_i$ denotes the reconstructed location for the sensor $\widetilde{x}_i$.

2. Sensor Network Localization Relaxations

In class, we learned that a problem such as Problem 2 can be relaxed into a number of different formulations, each with their own properties, advantages, and disadvantages. In this section, we explore three different relaxations of the SNL problem: a second-order cone programming (SOCP) relaxation, a semi-definite programming (SDP) relaxation, and a nonlinear least squares formulation.

The Seconder Order Cone Program (SOCP) Relaxation of Problem 2 is as follows.

Date: March 14, 2022.

Problem 3 (Seconder Order Cone Program (SOCP) Relaxation of SNL)**.** *Given data as in Problem 1, find* $\boldsymbol{x}_1, ..., \boldsymbol{x}_{n_s} \in \mathbb{R}^d$ *to solve*

$$\begin{aligned} \underset{\{\boldsymbol{x}_i\}_{i=1}^{n_s}}{\text{minimize}} \quad & \sum_{i=1}^{n_s} \mathbf{0}^\top \boldsymbol{x}_i, \\ \text{subject to} \quad & \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \leqslant d_{ij}^2, \forall (i,j) \in N_x, i < j, \\ & \|\boldsymbol{x}_i - \boldsymbol{a}_j\|_2^2 \leqslant \widehat{d}_{ij}^2, \forall (i,j) \in N_a. \end{aligned}$$

The Semi-definite Program (SDP) Relaxation of Problem 2 is as follows.

Problem 4 (Semi-definite Program (SDP) Relaxation of SNL)**.** *Given data as in Problem 1, find a symmetric matrix* $\boldsymbol{Z} \in S^{d+n_s}$ *to solve*

$$\begin{aligned} \underset{\boldsymbol{Z}}{\text{minimize}} \quad & \mathbf{0} \bullet \boldsymbol{Z}, \\ \text{subject to} \quad & \boldsymbol{Z}[1:d, 1:d] = \boldsymbol{I}, \\ & (\mathbf{0}; \boldsymbol{e}_i - \boldsymbol{e}_j)(\mathbf{0}; \boldsymbol{e}_i - \boldsymbol{e}_j)^\top \bullet \boldsymbol{Z} = d_{ij}^2, \forall (i,j) \in N_x, i < j, \\ & (\boldsymbol{a}_j; -\boldsymbol{e}_i)(\boldsymbol{a}_j; -\boldsymbol{e}_i)^\top \bullet \boldsymbol{Z} = \widehat{d}_{ij}^2, \forall (i,j) \in N_a, \\ & \boldsymbol{Z} \succeq \mathbf{0}. \end{aligned}$$

The Nonlinear Least Squares approach to Problem 2 is as follows.

Problem 5 (Nonlinear Least Squares for SNL)**.** *Given data as in Problem 1, solve*

$$\underset{\{\boldsymbol{x}_i\}_{i=1}^{n_s}}{\text{minimize}} \quad \sum_{(i,j) \in N_x} \left(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 - d_{ij}^2 \right)^2 + \sum_{(i,j) \in N_a} \left(\|\boldsymbol{x}_i - \boldsymbol{a}_j\|_2^2 - \widehat{d}_{ij}^2 \right)^2.$$

Both the SOCP and SDP formulations were solved using a simple, user-friendly implementation in CVX [2, 3]. The ease of use of this software is a huge advantage when solving problems of this form. In contrast, the nonlinear least squares (NLS) required a steepest descent implementation. This was also relatively straightforward, but required explicit computation of the gradient of the objective function.

2.1. **Experimental results.** We performed the following experiments to compare the performance of the various approaches:

- 5 anchors, 10 sensors, radius of 0.5 for three different seeds for each of SOCP, SDP, and NLS (Figures 1-3),
- 5 anchors, 20 sensors, radius of 0.5 for three different seeds for each of SOCP, SDP, and NLS (Figures 4-6),
- 10 anchors, 20 sensors, radius of 0.5 for three different seeds for each of SOCP, SDP, and NLS (Figures 7-9),
- 5 anchors, 10 sensors, and varying radius for each of SOCP, SDP, and NLS (plots of error as a function of radius are provided in Figure 10),
- 5 anchors, 10 sensors, radius of 0.5 for 30 different seeds for each of SOCP, SDP, NLS, NLS with SOCP initialization, and NLS with SDP initialization (mean and variance of error for each method are provided in Table 1).

In this section and throughout the report, the coordinates of all sensors and anchors were sampled i.i.d. (independently and identically distributed) from a standard Gaussian distribution.

Of the three approaches, the SDP relaxation performed best, and a radius of 0.5 was able to almost perfectly localize all sensors in each case. The SOCP relaxation performed well for sensors lying inside the convex hull of the anchors, but the locations of the sensors lying outside of the convex hull were generally not estimated well. This inspired a series of additional experiments which are described later in this section. The nonlinear least squares (NLS) approach was typically either slightly better or slightly worse than SOCP in terms of overall error, but the error was more evenly distributed across sensors. It was consistently much worse than SDP.

Interestingly, the error in the SOCP estimated locations and NLS estimated locations presented fairly different patterns. Whereas the SOCP approach often predicted the sensors lying outside the convex hull as lying inside the convex hull (resulting in a star-like error pattern), the error from the NLS approach presented a less regular pattern. These patterns can be seen in Figures 1-9. The impressive performance of SDP in our experiments is consistent with results found by others, for example [5].

In our experiment varying radius, we found that SOCP and SDP both improved monotonically up to a certain point - SOCP to a low nonzero level of error, and SDP to essentially zero error. On the other hand, NLS improved at first but then worsened as the radius exceeded approximately 0.5 and approached 1. This is interesting as it

indicates that more information is not always better in the case of NLS, while more information only helps with SOCP and SDP.

We found that using the SOCP or SDP solution as an initial guess for nonlinear least squares via steepest descent was, in most cases, able to provide a better estimate than either method alone. It dramatically improved the results relative to nonlinear least squares with a random initial guess, and offered great improvement over the SOCP method and modest improvement over the SDP method. It is to be expected that nonlinear least squares starting from an SDP solution would be very accurate since the SDP solution is already very accurate. One hypothesis for why the nonlinear least squares starting from an SOCP solution offers such improvement over either method alone is that the error from SOCP and nonlinear least squares are qualitatively different (see Figures 1 through 9 in the appendix).

A summary of the mean and variance of the 2-norm and ∞-norm error for each of the three main approaches, as well as nonlinear least squares with an SOCP or SDP initial guess, can be found in Table 1 for a radius of 0.5 and in Table 2 for a radius of 2.5.

TABLE 1. Randomly placed sensors and anchors. Mean and variance of the 2-norm and ∞-norm error for each of the three approaches, for 30 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 0.5.

	Mean(2-norm)	**Var(2-norm)**	**Mean(∞-norm)**	**Var(∞-norm)**
SOCP	0.68	0.05	1.43	0.35
SDP	0.13	0.03	0.14	0.04
NLS	0.87	0.06	1.89	0.35
NLS w SOCP	0.64	0.09	1.13	0.40
NLS w SDP	0.14	0.03	0.16	0.05

TABLE 2. Randomly placed sensors and anchors. Mean and variance of the 2-norm and ∞-norm error for each of the three approaches, for 30 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 2.5.

	Mean(2-norm)	**Var(2-norm)**	**Mean(∞-norm)**	**Var(∞-norm)**
SOCP	0.74	0.05	1.50	0.38
SDP	4.95 e-12	2.11 e-23	1.20 e-11	1.21 e-22
NLS	0.34	0.40	0.76	1.93
NLS w SOCP	0.02	3.84 e-4	0.04	1.70 e-3
NLS w SDP	3.73e-12	1.07 e-23	8.98 e-12	6.13 e-23

As an extension experiment, we wanted to explore how the placement of sensors impacted the performance of SOCP, SDP, and NLS. Given the choice between placing anchors in the center or on the boundary of a region, one might wonder the optimal position. For this experiment we compared:

- Anchors positioned on the outside, so that all sensors are in the convex hull (Table 4 and Figure 11),
- Anchors positioned inside the convex hull of the sensors (Table 5 and Figure 12),
- Anchors and sensors positioned approximately along the perimeter of a circle (Table 6 and Figure 13).

Example results and summary statistics can be found the Appendix in the Tables and Figures indicated. It is unsurprising that placing the anchors such that sensors are in the convex hull would lead to smaller error, as this ensures the positions of the sensors can be related to the positions that are already known, and often means that all of the sensors will be relatively close to at least one anchor. SOCP performs notably better when the sensors are restricted to the convex hull of the anchors. This restriction also leads to slightly better performance of NLS, as does placing all points approximately along the perimeter of a circle. It is impressive that even when all sensors are positioned outside the convex hull of the anchors that the SDP method still gives an accurate estimation of the true sensor point positions.

3. Sensor Network Localization with Noisy Data

In practice, it may be impossible to obtain exact distance data between sensors and between anchors and sensors. To simulate this, we will attempt to solve the SNL problem with noisy distance data and an extra slack variable (or two) in our problem formulations to account for this. In our experiments, we perturb the distance data with i.i.d. Gaussian noise with mean 0 and variance σ, where σ varies according to the experiment.

3.1. **SOCP Relaxation with Noisy Data.** The SOCP relaxation with noise can be written as follows, where δ''_{ij} and $\hat{\delta''_{kj}}$ are slack variables that allow the inequality constraints to be satisfied even with error in the distances d_{ij} and $\hat{d_{kj}}$.

Problem 6 (Noisy SOCP)**.** *Given data as in Problem 1, solve*

$$\begin{aligned} \min_{\mathbf{x}_i, \delta'', \hat{\delta}''} \quad & \sum_{(i,j)\in N_x} (\delta''_{ij}) + \sum_{(k,j)\in N_a} (\hat{\delta}''_{kj}) \\ s.t. \quad & \|\mathbf{x}_i - \mathbf{x}_j\|^2 - \delta''_{ij} \leqslant d_{ij}^2, \ \forall\ (i,j) \in N_x,\ i<j, \\ & \|\mathbf{a}_k - \mathbf{x}_j\|^2 - \hat{\delta}''_{kj} \leqslant \hat{d}_{kj}^2, \ \forall\ (k,j) \in N_a. \end{aligned}$$

3.2. **SDP Relaxation with Noisy Data.** The SDP relaxation with noise can be written as follows, where δ'_{ij}, δ''_{ij}, $\hat{\delta'_{kj}}$, and $\hat{\delta''_{kj}}$ are slack variables that allow the equality constraints to be satisfied even with error in the distances d_{ij} and $\hat{d_{kj}}$. Note that here we need extra slack variables to correct in both directions due to the equality constraints, as noise in either direction can cause the original constraints to be violated.

Problem 7 (Noisy SDP)**.** *Given data as in Problem 1, solve*

$$\begin{aligned} \min_{Z, \delta', \delta'', \hat{\delta}', \hat{\delta}''} \quad & \sum_{(i,j)\in N_x} (\delta'_{ij} + \delta''_{ij}) + \sum_{(k,j)\in N_a} (\hat{\delta}'_{kj} + \hat{\delta}''_{kj}) \\ s.t. \quad & Z_{1:d,1:d} = I, \\ & (\mathbf{0}; \mathbf{e}_i - \mathbf{e}_j)(\mathbf{0}; \mathbf{e}_i - \mathbf{e}_j)^T \bullet Z + \delta'_{ij} - \delta''_{ij} = d_{ij}^2, \ \forall\ i,j \in N_x,\ i<j, \\ & (\mathbf{a}_k; -\mathbf{e}_j)(\mathbf{a}_k; -\mathbf{e}_j)^T \bullet Z + \hat{\delta}'_{kj} - \hat{\delta}''_{kj} = \hat{d}_{kj}^2, \ \forall\ k,j \in N_a, \\ & Z \succeq \mathbf{0} \\ & \delta', \delta'', \hat{\delta}', \hat{\delta}'' \geqslant 0. \end{aligned}$$

3.3. **Experimental results.** To investigate this version of the SNL problem, we explored solving 6 and 7, as well as the gradient-descent methods attempted in Section 2. In particular, promising results from [6] using an SDP estimate as the initialization for a gradient-descent method suggest that NLS with SOCP or SDP initialization are worth investigating. We therefore performed the following experiments to evaluate the performance of the various approaches with noisy data:

- All 5 methods with $\sigma = 0.05$, 5 anchors, 10 sensors, and a radius of 2.5 (the reconstructed localizations for each method are shown in Figure 14),
- All 5 methods with varying levels of noise, 5 anchors, 10 sensors, and a radius of 2.5 (plots of error as a function of noise level are provided in Figure 15).

Note that the 5 methods refer to: SOCP, SDP, NLS (initialized randomly), NLS initialized with SOCP, and NLS initialized with SDP. In these experiments, we opted for a larger radius to isolate the effect of noise.

Perhaps unsurprisingly, SDP performed better than SOCP and NLS for the first experiment. However, NLS with SOCP or SDP initializations performed even better than SDP, and demonstrated very similar performance to one another. This result was confirmed in the second experiment. While SDP has the lowest error when noise is very low or nonexistent, it is not robust to high levels of noise. SOCP actually performs better than SDP with high levels of noise (at least in the context evaluated), but NLS with SOCP and SDP initializations both performed well across the full spectrum of noise levels: they had extremely low error when noise was low (similar to SDP), but performed relatively well even with high levels of noise (similar to SOCP). It seems then, that in contexts where the distances are not perfectly known, that NLS with either an SOCP or SDP initialization is the preferred choice, as these approaches combine the advantages of each of SOCP, SDP, and NLS.

4. Steepest Descent Projection Method

4.1. **Overview of Method.** Recall the original SDP relaxation of SNL as outline in Problem 4. We can re-cast the SDP problem as a least squares problem as follows.

Problem 8 (Steepest Descent Projection Method Problem for SNL)**.** *Given data as in Problem 1, find a symmetric matrix $\boldsymbol{Z} \in S^{d+n_s}$ to solve*

$$\begin{aligned} & \underset{\boldsymbol{Z}}{minimize} && f(\boldsymbol{Z}) := \frac{1}{2}\|\mathcal{A}\boldsymbol{Z} - \boldsymbol{b}\|_2^2, \\ & subject\ to && \boldsymbol{Z} \succeq \mathbf{0}. \end{aligned}$$

where $\mathcal{A}\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{A_1} \bullet \boldsymbol{Z} \\ \vdots \\ \boldsymbol{A_q} \bullet \boldsymbol{Z} \end{pmatrix}$, *where we have one* $\boldsymbol{A}_i$ *corresponding to each of the* $q = |N_x| + |N_a| + d^2$ *linear constraints in Problem 4. That is, each pair* $(\boldsymbol{A}_i, \boldsymbol{b}_i)$ *corresponds to one of the constraints of the form* $\boldsymbol{A}_i \bullet \boldsymbol{Z} = \boldsymbol{b}_i$ *in Problem 4.*

The gradient of f can be expressed cleanly as $\nabla f(\boldsymbol{Z}) = \mathcal{A}^\top(\mathcal{A}\boldsymbol{Z} - \boldsymbol{b})$, so in order to solve Problem 8, it makes sense to employ the method of steepest descent. If we initialize our initial guess $\boldsymbol{Z}^0$ to be symmetric, then all the iterates $\boldsymbol{Z}^k$ of gradient descent will remain symmetric due to the symmetric nature of the gradient update. Unfortunately, when stepping in the direction of steepest descent of the objective function, the constraint $\boldsymbol{Z} \succeq \boldsymbol{0}$ may become violated even if $\boldsymbol{Z}^0 \succeq \boldsymbol{0}$. In order to remedy this, we use the SDP projection method.

Specifically, each step of the iterative optimization algorithm consists of two parts. Given an initial guess of $\boldsymbol{Z}^0$, we perform the following updates for $k = 1, 2, ...$ until convergence. The specific initialization and convergence details are discussed in more detail in the following subsection.

(1) Descent step: $\widehat{\boldsymbol{Z}}^{k+1} = \boldsymbol{Z}^k - \epsilon \nabla f(\boldsymbol{Z^k})$, where ϵ is a step-size;
(2) Projection step: $Z^{k+1} = \boldsymbol{V} \max(\boldsymbol{0}, \boldsymbol{\Lambda}) V^\top$, where $\widehat{\boldsymbol{Z}}^{k+1} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$ is the orthogonal eigen-decomposition of $\widehat{\boldsymbol{Z}}^{k+1}$.

The first step performs the gradient descent update to obtain a symmetric but not necessarily positive semi-definite matrix $\widehat{\boldsymbol{Z}}^{k+1}$, and the second step maps $\widehat{\boldsymbol{Z}}^{k+1}$ to a symmetric positive semi-definite matrix $\boldsymbol{Z}^{k+1}$ by replacing any of its negative eigenvalues with zero. Although the projection step seems elusive at first glance, we can analyze it theoretically using the following Lemma.

Lemma 1. *Let* $\boldsymbol{A} \in S^{n \times n}$ *be a symmetric matrix, and let* $\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$ *be its orthogonal eigen-decomposition. Let* $\boldsymbol{A}_{proj} := \boldsymbol{V} \max(\boldsymbol{0}, \boldsymbol{\Lambda})\boldsymbol{V}^\top$. *Then* $\boldsymbol{A}_{proj}$ *solves*

$$\begin{aligned} &\underset{\boldsymbol{B}}{\text{minimize}} \ \|\boldsymbol{A} - \boldsymbol{B}\|_F, \\ &\text{subject to } \boldsymbol{B} \succeq \boldsymbol{0}. \end{aligned}$$

Proof. Consider any matrix $\boldsymbol{B} \succeq 0$, and let $\boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{U}^\top$ be the orthogonal eigen-decomposition of $\boldsymbol{B}$. By orthogonality of $\boldsymbol{U}$ and the fact that the Frobenius norm of a matrix is invariant with respect to left or right multiplication of an orthogonal matrix, we have

$$\begin{aligned} \|\boldsymbol{A} - \boldsymbol{B}\|_F &= \|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{U}^\top\|_F \\ &= \|\boldsymbol{U}^\top\boldsymbol{A}\boldsymbol{U} - \boldsymbol{\Gamma}\|_F \\ &= \|\boldsymbol{C} - \boldsymbol{\Gamma}\|_F \\ &= \sum_{i \neq j} c_{ij}^2 + \sum_{i=1}^{n} (\gamma_i - c_{ii})^2 \\ &\geqslant 0 + \sum_{i=1}^{n} (\gamma_i - c_{ii})^2 \\ &\geqslant \sum_{i : c_{ii} < 0} c_{ii}^2, \end{aligned}$$

where we let $\boldsymbol{C} = \boldsymbol{U}^\top\boldsymbol{A}\boldsymbol{U}$, and in the last line we used the fact that $\gamma_i \geqslant 0$ because $\boldsymbol{B} \succeq \boldsymbol{0}$. Note that the inequalities are tight when $\gamma_i = \max(0, c_{ii})$ and $c_{ij} = 0$, which is true when $U = V$ and $\Gamma = \max(0, \boldsymbol{\Lambda})$ (because then $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top \implies \boldsymbol{C} = \boldsymbol{V}^\top\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\Lambda}$). □

This result justifies why the projection step (2) above makes sense. In step (1), we find the next iterate in steepest descent and in step (2), we project this iterate onto the SDP cone by finding the SDP matrix which is closest to it in the sense of the Frobenius norm. Unfortunately, in practice, computing the projection step (2) may be expensive. The expense of computing the eigen-decomposition can be reduced if we only compute the first $p < n_s$ largest eigenpairs of $\widehat{Z}^{k+1}$ at each iteration and simply set $Z^{k+1} = \boldsymbol{V}_p \max(\boldsymbol{0}, \boldsymbol{\Lambda}_p)\boldsymbol{V}_p^\top \approx \boldsymbol{V} \max(\boldsymbol{0}, \boldsymbol{\Lambda})\boldsymbol{V}^\top$ where $\boldsymbol{\Lambda}_p$ is the $p \times p$ diagonal matrix containing the p largest eigenvalues of $\widehat{Z}^{k+1}$ on the diagonal and $\boldsymbol{V}_p$ is the $(n_s + d) \times p$ matrix consisting of the columns of $\boldsymbol{V}$ corresponding to the eigenvectors associated with these eigenvalues. That is, Z^{k+1} is chosen as the best rank at-most-p approximation to the projection of $\widehat{Z}^{k+1}$ onto the SDP cone (by the Ekhart-Young Theorem).

4.2. **Experimental Results.** In order for the method to work well, $\boldsymbol{Z}^0$ should be a feasible initial guess, meaning that it must be symmetric and positive semi-definite. We considered two approaches for selecting the random initial guess $\boldsymbol{Z}^0$:

- Initialize random sensor locations $\boldsymbol{Z'} \in \mathbb{R}^{n_s \times d}$ sampled i.i.d. from a standard Gaussian distribution. (Consistent with our initialization for the true sensor and anchor locations in all experiments). Then define $\boldsymbol{Z}^0$ to be the block matrix
$$\boldsymbol{Z}^0 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{Z'}^\top \\ \boldsymbol{Z'} & \boldsymbol{Z'}\boldsymbol{Z'}^\top \end{pmatrix}.$$
This matrix $\boldsymbol{Z}^0$ is both positive semi-definite and symmetric.
- Initialize a random matrix $\widetilde{\boldsymbol{Z}}^0 \in \mathbb{S}^{d+n_s}$ with entries sampled i.i.d. from a standard Gaussian distribution. Then obtain $\boldsymbol{Z}^0$ by projecting $\widetilde{\boldsymbol{Z}}^0$ onto the positive semi-definite cone using the procedure outlined in Lemma 1.

In our experimentation, we found that the first method of initialization consistently served as a superior initial guess for the steepest descent projection method. This is perhaps unsurprising, as we know from lecture that the closed-form solution to the SDP relaxation problem is in the form of the first initialization (with $\boldsymbol{Z'} = \boldsymbol{X}$), so it is natural to expect that the first initialization approach would produce a solution which is closer to the optimal solution to Problem 8. Moreover, the first initialization approach automatically satisfies the constraint on the upper left $d \times d$ block in Problem 4, while the second initialization approach does not, which may make it a better initial guess.

In order to determine the convergence criteria, we simply stopped the iterations of projected steepest descent when the difference between the objective value across two iterations fell below a certain threshold t. In our experiments, we found that a convergence criteria of $\left|\|\mathcal{A}\boldsymbol{Z}^{k+1} - \boldsymbol{b}\|_2^2 - \|\mathcal{A}\boldsymbol{Z}^k - \boldsymbol{b}\|_2^2\right| < t = 1e-8$ worked well for detecting convergence.

Sample figures visualizing the reconstructed sensor locations can be found in the Appendix (Section 7). Figure 16 provides an example of the sensor location reconstructions of the projected steepest descent method when $n_s = 10$ sensors and $n_a = 5$ anchors are used. We find that the projected steepest descent method performs comparably to the SDP Relaxation solution, but runs much faster than the SDP relaxation program. Figure 17 illustrates how the error in the inf-norm (maximum error in sensor location across all sensors) compares to the error in the 2-norm (mean squared error in sensor location across all sensors). We find that a step size of roughly $\epsilon = .01$ works well for most examples, achieving a good trade-off between accuracy and speed of convergence, as supported by the figure. Finally, Figure 17 illustrates how the accuracy of sensor location reconstructions varies with p, the number of eigenpairs used for the projection onto the SDP cone. We find that the number of eigenpairs used does not significantly influence the accuracy, provided that at least 4 or 5 eigenpairs are used. We found this result to be robust to increasing the number of sensors and anchors.

5. ADMM Method for Sensor Network Localization

5.1. **Overview of Method.**

Problem 9 (Alternating Direction Method of Multipliers (ADMM) for SNL)**.** *Given data as in Problem 1, we can reformulate the nonlinear least squares method from Problem 5 to solve*

$$\begin{aligned} &\textit{minimize} && \sum_{(i,j)\in N_x} \left[(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{z}_i - \boldsymbol{z}_j) - d_{ij}^2\right]^2 + \sum_{(k,j)\in N_a} \left[(\boldsymbol{a}_k - \boldsymbol{x}_j)^T(\boldsymbol{a}_k - \boldsymbol{z}_j) - \hat{d}_{kj}^2\right]^2, \\ &\textit{subject to} && \boldsymbol{x_j} - \boldsymbol{z_j} = \boldsymbol{0}, \forall j. \end{aligned}$$

The augmented Lagrangian function for Problem 8 would be

$$\begin{aligned} L_a(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = &\sum_{(i,j)\in N_x} \left[(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{z}_i - \boldsymbol{z}_j) - d_{ij}^2\right]^2 + \sum_{(k,j)\in N_a} \left[(\boldsymbol{a}_k - \boldsymbol{x}_j)^T(\boldsymbol{a}_k - \boldsymbol{z}_j) - \hat{d}_{kj}^2\right]^2 \\ &- \sum_i \boldsymbol{y}_i^T(\boldsymbol{x}_i - \boldsymbol{z}_i) + \frac{\beta}{2}\sum_i \|\boldsymbol{x}_i - \boldsymbol{z}_i\|_2^2, \end{aligned}$$

where β can be viewed as the penalty constant for the discrepancy between $\boldsymbol{x}$ and $\boldsymbol{z}$. Then, we can treat the $\boldsymbol{x}_i$'s as the first block of variables and the $\boldsymbol{z}_i$'s as the second block of variables, and minimize $L_a(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y})$ in an alternative order:

$$\boldsymbol{x}_{k+1} := \arg\min_{\boldsymbol{x}} L_a(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{y}_k),$$

$$\boldsymbol{z}_{k+1} := \arg\min_{\boldsymbol{z}} L_a(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{y}_k),$$
$$\boldsymbol{y}_{k+1} := \boldsymbol{y}_k - \beta(\boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1}).$$

Note that each of the block optimization problems within each ADMM iteration is a convex quadratic minimization problem, so we can solve Problem 8 using a simple implementation in CVX [2, 3]. We initialize $\boldsymbol{x}_0$, $\boldsymbol{z}_0$, and $\boldsymbol{y}_0$ randomly, and output $(\boldsymbol{x} + \boldsymbol{z})/2$ at convergence, as in practice the final values of $\boldsymbol{x}$ and $\boldsymbol{z}$ may differ very slightly. These smaller block problems can be solved more efficiently and capitalize on benefits from applying the dual and Lagrangian framework to constrained optimization problems in a simple form.

5.2. **Experimental Results.** Figure 19 depicts an example with $n_a = 5$ anchors, $n_s = 10$ sensors, and a threshold radius of $r = 0.5$. Note that this is the same example as in Figure 16 for the steepest descent projection method. Compared to the steepest descent projection method, the error for the ADMM method on this example SNL data is similar, but ADMM leads to a smaller error the infinity norm, or the maximum error in any sensor, compared to the steepest descent projected method. Further, the ADMM method is able to achieve these very small error margins in much fewer iterations. The ADMM method is easier to tune, convexifies the problem, and is faster than SDP, but since the variables are coupled in the augmented Lagrangian, this cannot be implemented as a distributed method. Another interesting observation regarding ADMM is that compared to the steepest-descent projection method, ADMM is less sensitive to the choice of its tunable parameter β, as compared to the steepest-descent projection method, which we found to be extremely sensitive to the choice of its primary parameter ϵ. This motivates why ADMM may be easier to tune and more robust as compared to the steepest-descent projection method discussed in the previous section.

In Table 3, we compared the performance of these methods for 15 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 0.5. In addition, we analyze the impact the order of the block updates has on the performance of the ADMM method. For our initial implementation of ADMM, in each iteration we always updated $\boldsymbol{x}$, then $\boldsymbol{z}$, then $\boldsymbol{y}$. In a modified version of ADMM, we permute the order of the updates randomly for each iteration. In some sense, this leads to more fairness among the blocks, and in our numerical experiments leads to slightly better performance as seen by the decrease in mean and variance of the 2-norm. It is interesting that the mean error for these 15 runs using ADMM is so much greater than the example in Figure 19. In doing this experiment, we noticed that the error for the ADMM methods was usually dominated by a couple of sensors some distance outside of the convex hull of the anchors, where the predicted location remained inside the convex hull. This is similar to some of our observations for SOCP.

We saw that a penalty rate of $\beta = 1$ performed well for most examples. As seen in Figure 20, this selection for β leads to minimal error while also taking large enough steps to help reduce the number of iterations needed for convergence. In order to determine the convergence criteria, we simply stopped the iterations of projected steepest descent when the difference between the value of y across two iterations fell below a certain threshold. In our experiments, we found that a convergence criteria of $t = 1e - 8$ worked well for detecting convergence.

TABLE 3. Comparison of the steepest descent projection method (SDPM), the ADMM method with fixed block update order (ADMM), and the ADMM method with random permutation of blocks(Perm. ADMM). Mean and variance of the 2-norm and ∞-norm error for each of the three approaches, for 15 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 0.5. The ADMM methods were run for 500 iterations or until convergence, while the steepest descent projection method was run for 10,000 steps or until convergence.

	Mean(2-norm)	**Var(2-norm)**	**Mean(∞-norm)**	**Var(∞-norm)**
SDPM	0.69	0.12	1.30	0.207
ADMM	0.77	0.33	0.43	0.083
Perm. ADMM	0.69	0.26	0.44	0.083

6. Discussion

The formulation and solving of the Sensor Network Localization problem is a rich area of research. Our experimental inquiry in this project illuminated the relative strengths and weaknesses of several approaches in various contexts - for example that NLS with an SOCP or SDP initialization may be preferred, especially in contexts with noisy distance data. In particular, because the SOCP relaxation is relatively computationally inexpensive to solve, the SOCP solution is a good candidate for further refinement by the nonlinear least squares steepest descent method. Our experiments also revealed a trade-off between accuracy and reliability in the ADMM and steepest-descent-projection method (SDPM) for tackling versions of the more computationally expensive SDP relaxation.

Although SDPM runs much faster than the ADMM approach, the ADMM approach has advantages in that it produces more accurate results on average, is a convex problem, requires fewer iterations to converge, and is more robust to parameter choice.

A great deal remains unexplored, and there are several further methods one could apply to solve the SNL problem. For example, others (e.g. [7]) discuss various further adaptations of the problem formulation for specific applications. In addition, there are several further questions we could ask regarding the methods implemented in this report. For example, we might consider using a more sophisticated step size or adaptive step size to further refine the results of nonlinear least squares or the steepest descent projection method. Similarly, it would be interesting to consider an adapting penalty rate β in the ADMM approach to gradually push $\boldsymbol{x}_j$ and $\boldsymbol{z}_j$ closer to one another as the number of iterations increases. As another example of future experiments, throughout this paper we consider anchor and sensor locations sampled i.i.d. from a standard Gaussian distribution, with the exception of our experiments pertaining to the placement of anchors within the convex hull of the sensors. It would be interesting to see how the various methods perform under different underlying sensor and anchor location distributions, and specifically, it would be interesting to see which of these methods are most robust to extreme/pathological distributions of sensors and anchors.

For reference, the complete code used for the numerical experiments in this report can be found at `https://github.com/apropp/CME307`.

References

[1] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated. 2021.

[2] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, September 2013.

[3] Michael Grant and Stephen Boyd. "Graph implementations for nonsmooth convex programs," Recent Advances in Learning and Control (a tribute to M. Vidyasagar), pages 95-110, Lecture Notes in Control and Information Sciences, Springer, 2008.

[4] P. Biswas, K-C. Toh, and Y. Ye. "A Distributed SDP Approach for Large-Scale Noisy Anchor-Free Graph Realization with Applications to Molecular Conformation." *SIAM J. Scientific Computing*. 30. 1251-1277. 10.1137/05062754X, 2008.

[5] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," *Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, ACM Press, 2004, pp. 46–54.

[6] Pratik Biswas, T-C Liang, K-C Toh, Y. Ye, T-C Wang, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Transactions on Automation Science and Engineering*, 3(4), 2006, pp. 360-371.

[7] L. Doherty, L. E. Ghaoui, and K. S. J. Pister, "Convex position estimation in wireless sensor networks." in *Proceedings of IEEE Infocom*, Anchorage, Alaska, April 2001, pp. 1655 –1663.

[8] A. So. A Semidefinite Programming Approach to the graph realization problem: Theory, Applications and Extensions *PhD Thesis, Stanford University*, 2007.

7. Appendix

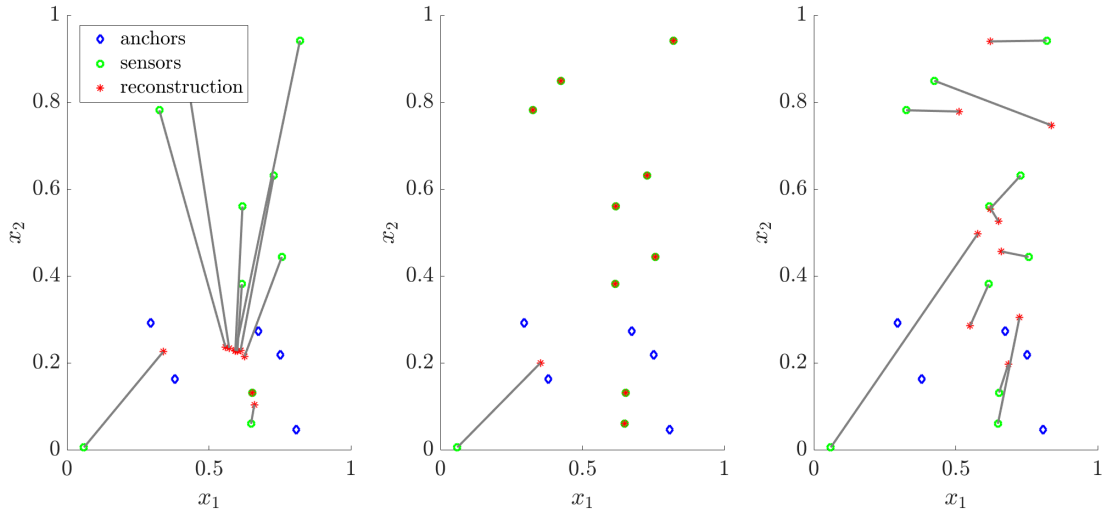

Figure 1. Results for each of the three methods with 5 anchors, 10 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

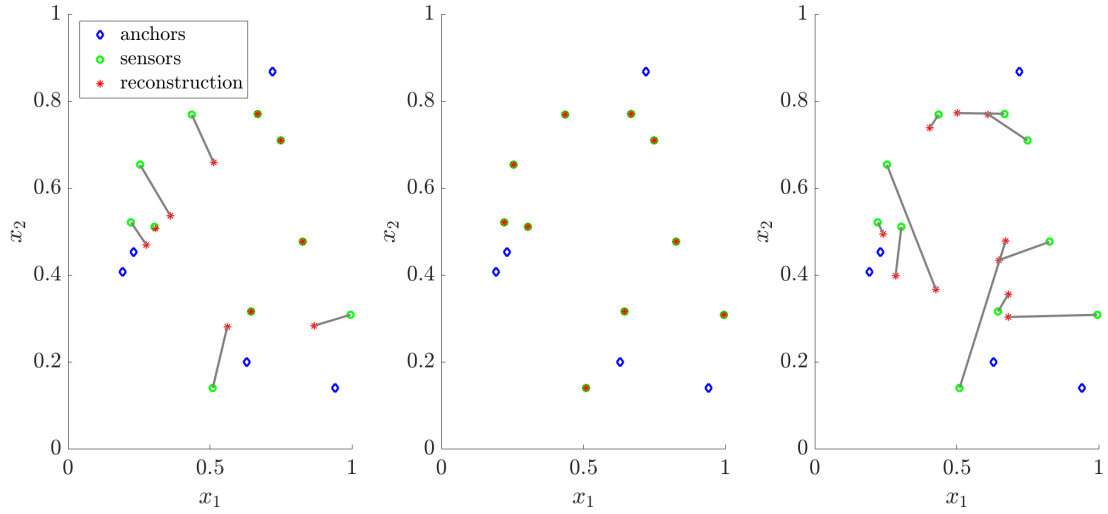

FIGURE 2. Results for each of the three methods with 5 anchors, 10 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

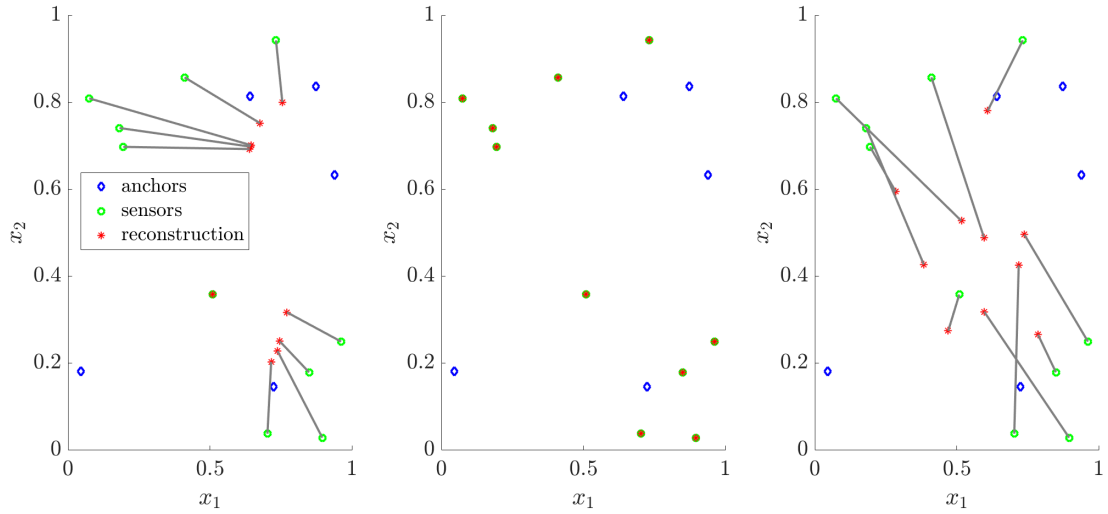

FIGURE 3. Results for each of the three methods with 5 anchors, 10 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

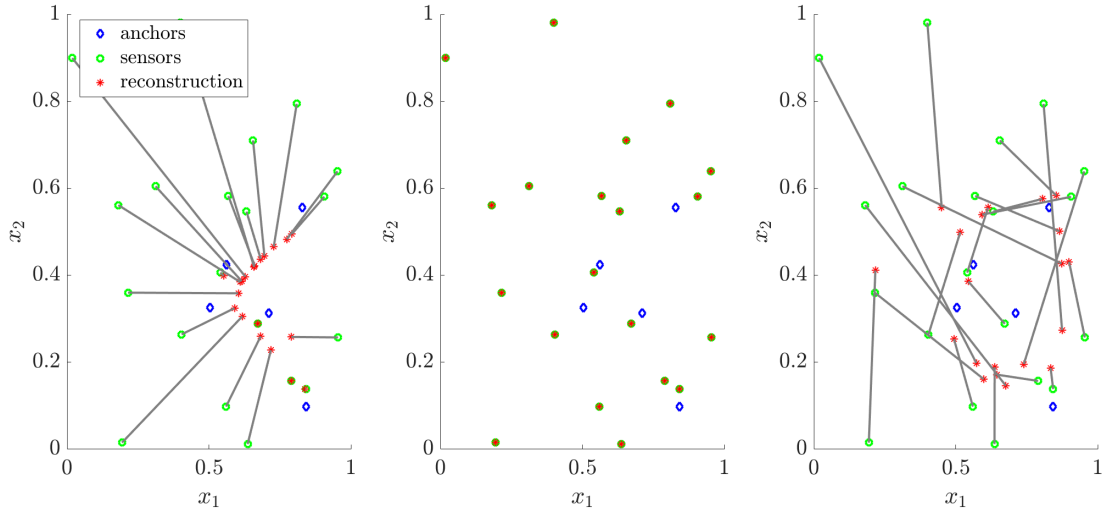

FIGURE 4. Results for each of the three methods with 5 anchors, 20 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

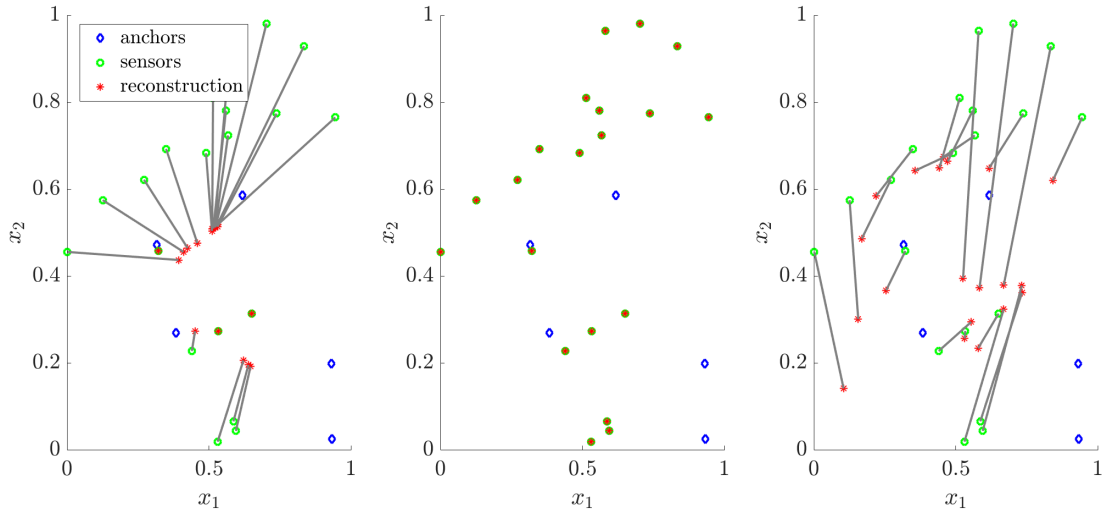

FIGURE 5. Results for each of the three methods with 5 anchors, 20 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

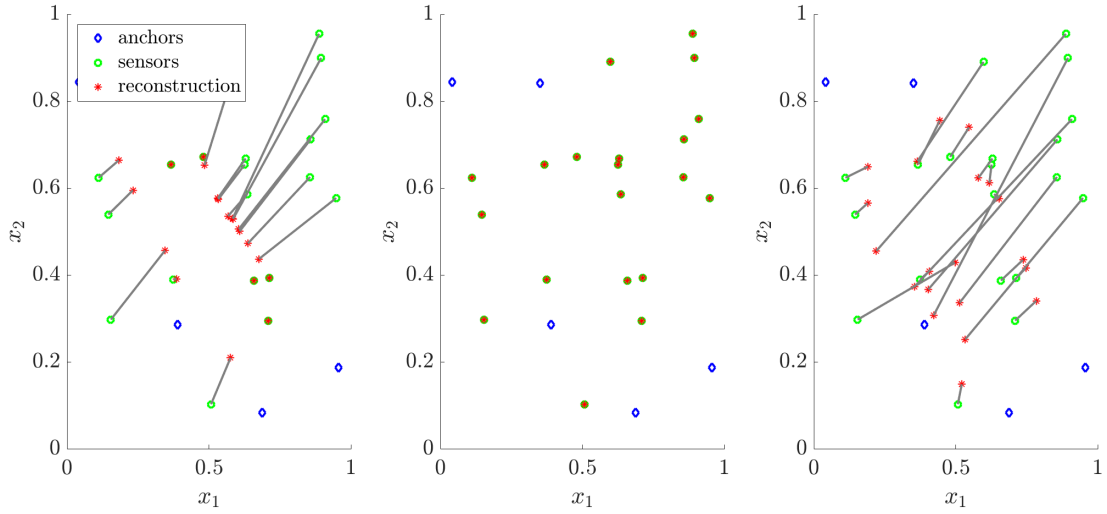

FIGURE 6. Results for each of the three methods with 5 anchors, 20 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

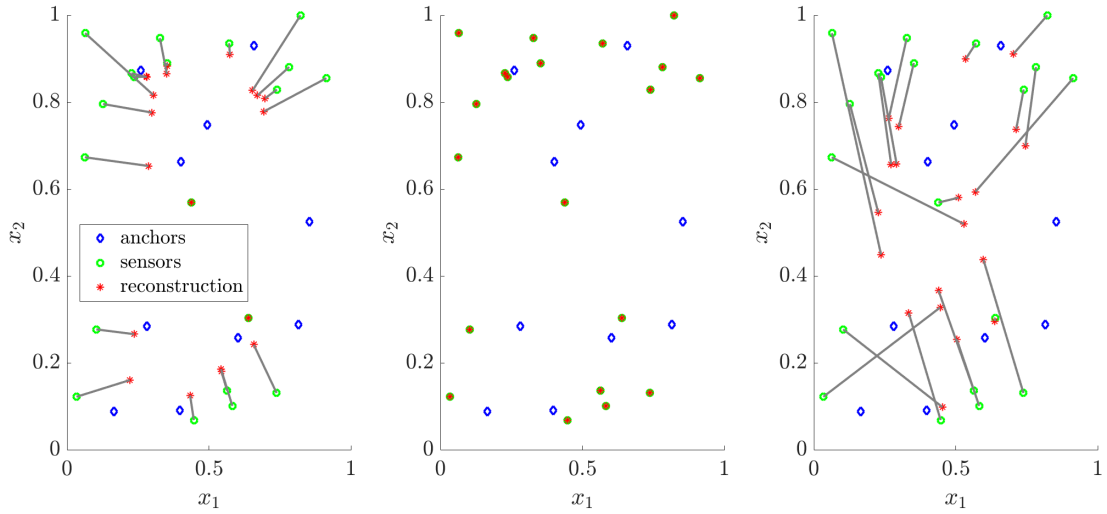

FIGURE 7. Results for each of the three methods with 10 anchors, 20 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

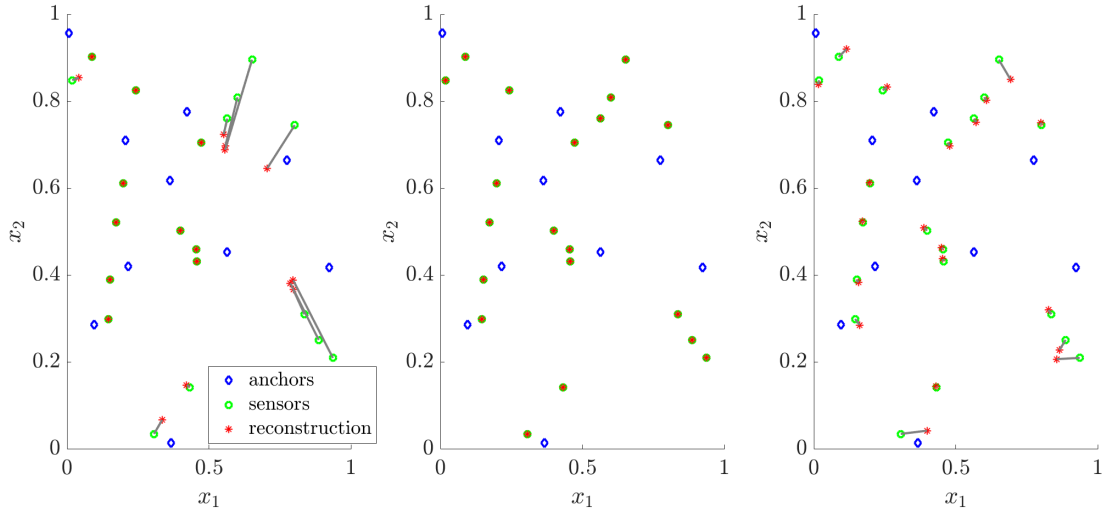

FIGURE 8. Results for each of the three methods with 10 anchors, 20 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

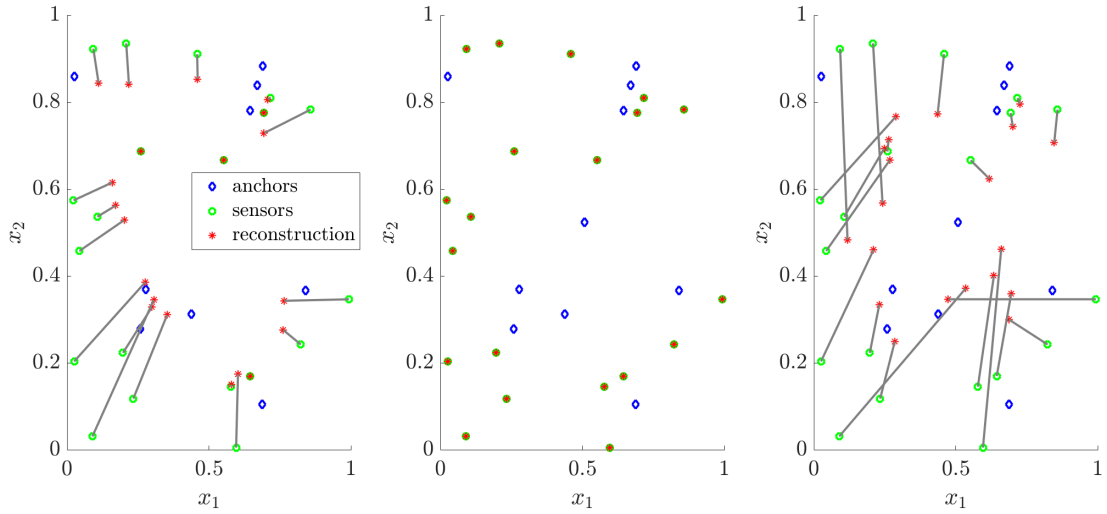

FIGURE 9. Results for each of the three methods with 10 anchors, 20 sensors, and a radius of 0.5. Left panel shows SOCP, middle panel shows SDP, and right panel shows NLS with a step size of 5*1e-3 and 100 iterations.

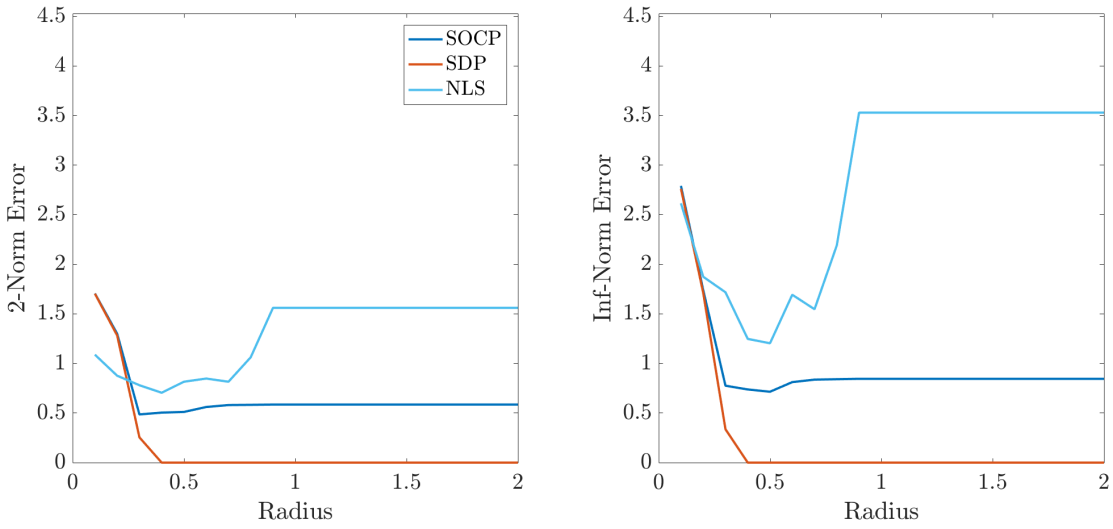

FIGURE 10. Error as a function of radius, for the three main methods with 5 anchors, 10 sensors, and a fixed seed. With a large enough radius, we see that error drops to essentially zero for SDP. However, for both SOCP and NLS, error cannot be completely eliminated even with a large enough radius to capture all distances. These results were robust to different seeds (i.e. different sensor and anchor locations).

TABLE 4. Anchors randomly placed near perimeter of unit circle, sensors randomly placed within the convex hull of those anchors. Mean and variance of the 2-norm and ∞-norm error for each of the three approaches, for 50 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 2.5.

	Mean(2-norm)	**Var(2-norm)**	**Mean(∞-norm)**	**Var(∞-norm)**
SOCP	0.0087	5.36e-4	0.0084	5.22 e-4
SDP	1.76 e-11	5.02 e-22	4.23 e-11	3.17 e-21
NLS	0.23	0.017	0.57	0.12

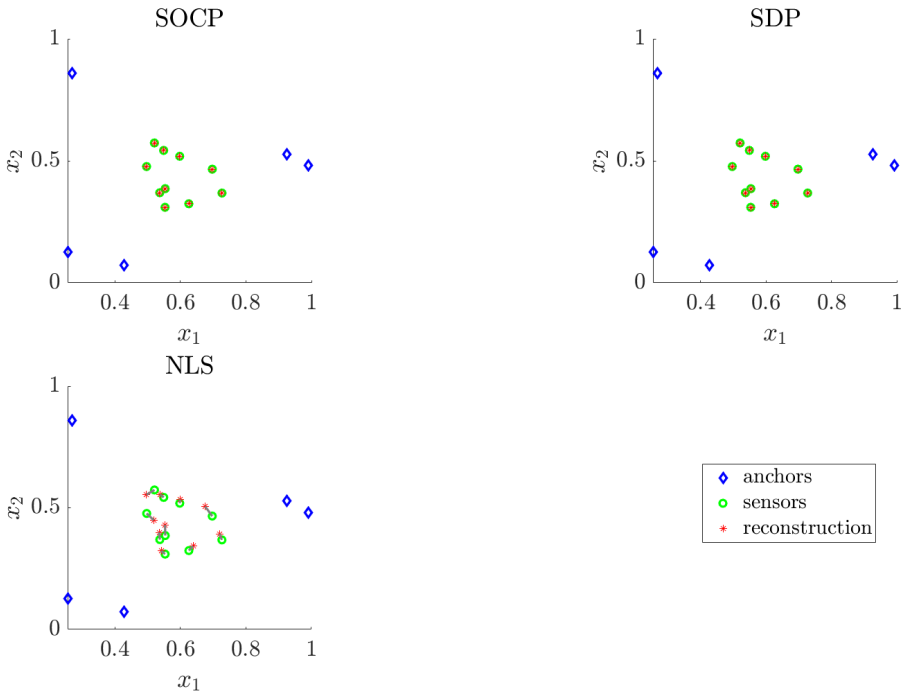

FIGURE 11. Results for each of the three methods with 10 anchors, 20 sensors, and a radius of 2.5, where anchors were randomly placed near perimeter of unit circle and sensors were randomly placed within the convex hull of those anchors. Left panel shows SOCP, upper right panel shows SDP, and lower left panel shows NLS with a step size of 5*1e-3 and 100 iterations.

TABLE 5. Anchors randomly placed on interior of unit circle, sensors randomly placed near perimeter of unit circle. Mean and variance of the 2-norm and ∞-norm error for each of the three approaches, for 50 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 2.5.

	Mean(2-norm)	**Var(2-norm)**	**Mean(∞-norm)**	**Var(∞-norm)**
SOCP	1.03	0.021	2.77	0.21
SDP	4.11 e-12	8.93 e-24	1.084 e-11	6.50 e-23
NLS	1.18	0.39	3.01	2.73

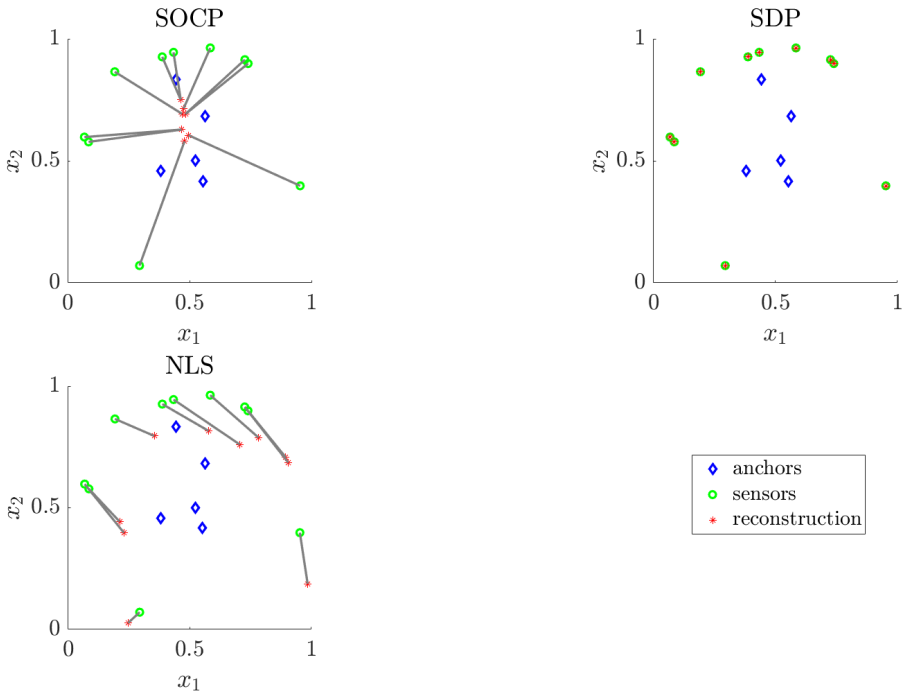

FIGURE 12. Results for each of the three methods with 10 anchors, 20 sensors, and a radius of 2.5, where anchors randomly placed on interior of unit circle and sensors randomly placed near perimeter of unit circle. Left panel shows SOCP, upper right panel shows SDP, and lower left panel shows NLS with a step size of 5*1e-3 and 100 iterations.

TABLE 6. Anchors and sensors randomly placed near perimeter of unit circle. Mean and variance of the 2-norm and ∞-norm error for each of the three approaches, for 50 randomly seeded runs with 5 anchors, 10 sensors, and a radius of 2.5.

	Mean(2-norm)	**Var(2-norm)**	**Mean(∞-norm)**	**Var(∞-norm)**
SOCP	0.75	0.10	1.70	0.63
SDP	4.04 e-12	6.47 e-24	1.03 e-11	4.26 e-23
NLS	0.52	0.40	1.27	2.29

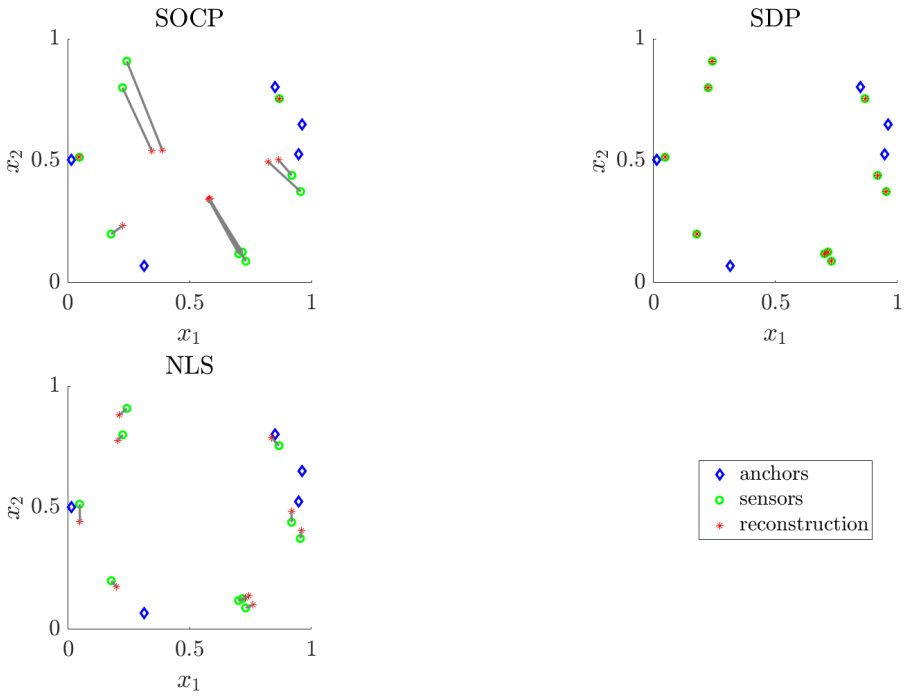

FIGURE 13. Results for each of the three methods with 10 anchors, 20 sensors, and a radius of 2.5, where anchors and sensors randomly placed near perimeter of unit circle. Left panel shows SOCP, upper right panel shows SDP, and lower left panel shows NLS with a step size of 5*1e-3 and 100 iterations.

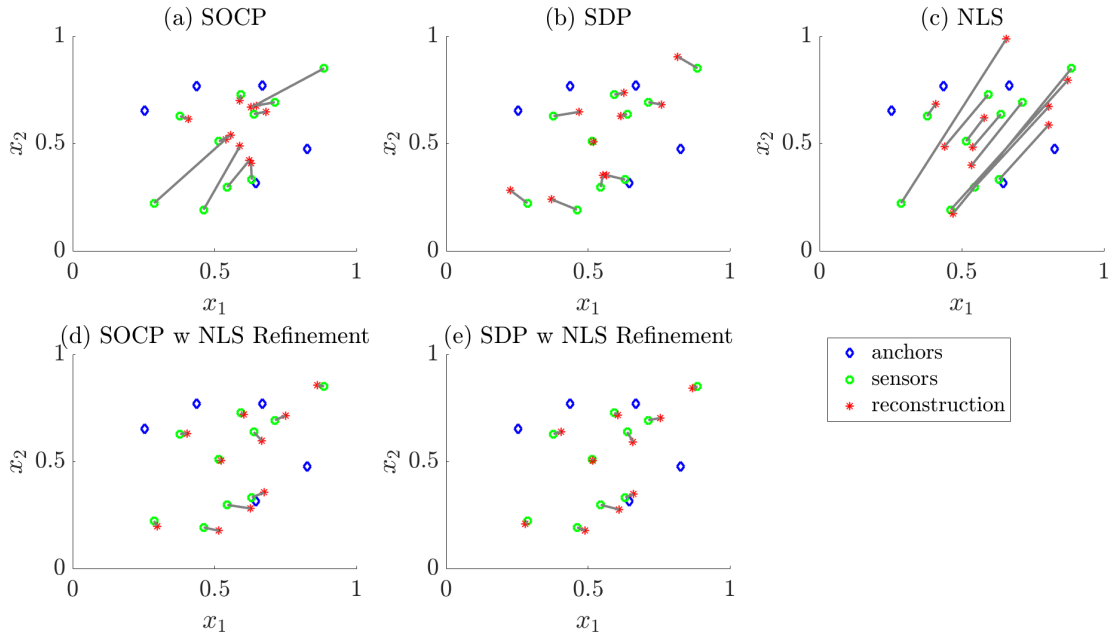

FIGURE 14. Results for each method with noisy distance data and 5 anchors, 10 sensors, a radius of 2.5. Normally distributed noise with a mean of 0 and standard deviation 0.05 was added to the matrix of pairwise distances to simulate the effect of imperfect data.

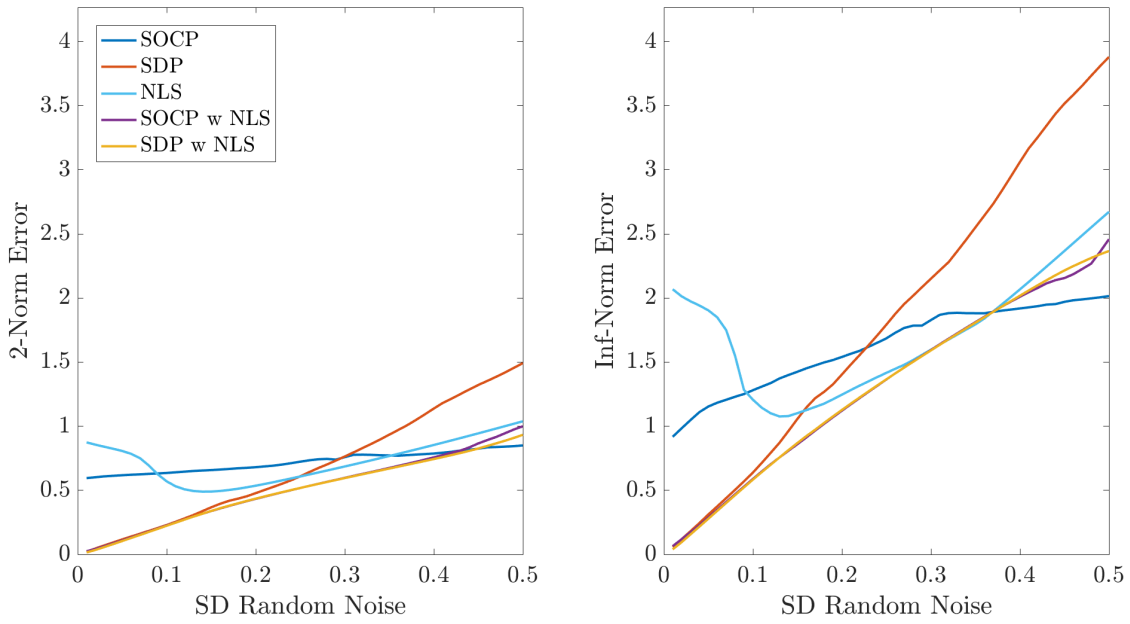

FIGURE 15. 2-norm and ∞-norm error for each method as a function of the standard deviation of the Gaussian noise distribution. Error increases sharply for the SDP method. While the SOCP method has a higher level of baseline error, its error does not increase as much with increasing noise. NLS with SOCP and SDP initializations have very low error for low levels of noise (similar to SDP on its own), but their error are more robust to increased noise than SDP on its own. NLS on its own actually performs better with a low level of noise than with no noise at all (this result was robust to different seeds), but error increases with increasing noise at a similar rate to NLS with SOCP or SDP initializations.

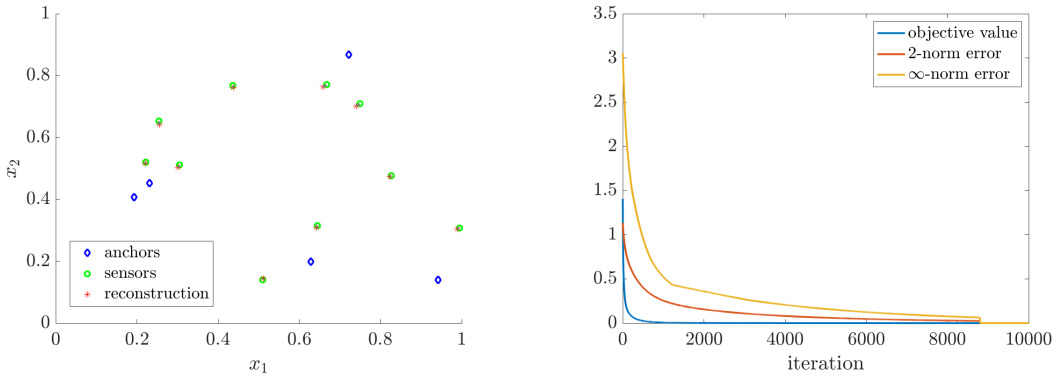

FIGURE 16. Example Projected Steepest Descent Sensor Reconstruction. The figure shows an example of the Projected Steepest Descent Method on a sample dataset of anchors and sensors. In this example, we used $n_a = 5$ anchors, $n_s = 10$ sensors, a threshold radius of $r = .5$, and a fixed step size of $\epsilon = .01$, which we found to work well for this example. The descent algorithm was run until convergence. The plot on the left illustrates the exact and approximate sensor locations; meanwhile, the plot on the right illustrates the decay in the objective function value, 2-norm reconstruction error, and inf-norm reconstruction error. Here, we used $p = 6$ eigenvalues when projecting onto the SDP cone after each descent step.

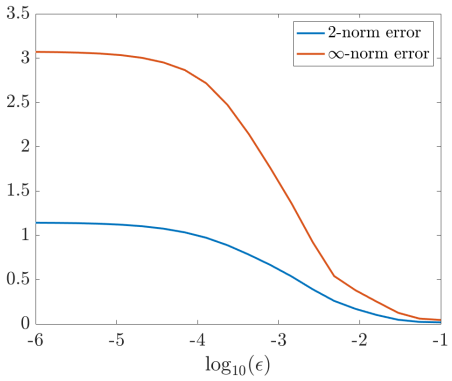

FIGURE 17. Plot of Error vs. Log-scale Step size of the Projected Steepest Descent Method. The figure shows how error decays when the fixed step size of the method is tuned between 10e-6 and 10e-1. In each case, the steepest descent method was run until convergence, or until a maximum iterations of 10,000 steps was reached. In this example, we use $n_a = 5$ anchors, $n_s = 10$ sensors, a threshold radius of $r = .5$, and $p = 6$ eigenvalues when projecting onto the SDP cone after each descent step.

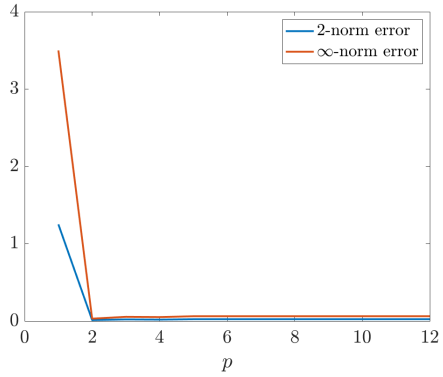

FIGURE 18. Plot of Error vs. p for the Projected Steepest Descent Method. The figure shows how error decays when the number of eigenvalues used in the projection (denoted p) is increased. In each case, the steepest descent method was run until convergence. In this example, we use $n_a = 5$ anchors, $n_s = 10$ sensors, a threshold radius of $r = .5$, and a step size of $\epsilon = .01$, which we found to work well in Figure 17

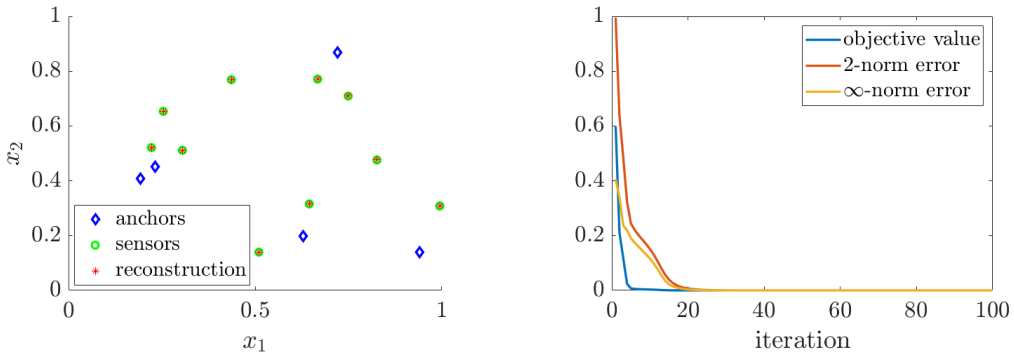

FIGURE 19. Example of ADMM Method on SNL. This is a depiction of the performance of the ADMM method with step size $\beta = 1$ on an example with $n_a = 5$ anchors, $n_s = 10$ sensors, and a threshold radius of $r = 0.5$. The method was run until convergence of y was achieved.

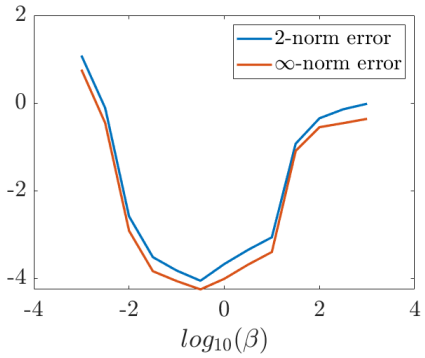

FIGURE 20. Plot of Error vs. Log-scale Penalty Rate of the ADMM Method. This figure shows the relationship between the error and the fixed penalty rate of the ADMM method as the penalty rate ranges from 10e-3 to 10e3. For each value of the penalty rate, ADMM was run until convergence of y or until a maximum of 250 iterations. The SNL example used for this numerical experiment has $n_a = 5$ anchors, $n_s = 10$ sensors, and a threshold radius of $r = 0.5$.