A Laplacian-based Error Minimization Algorithm for Distance Geometry Problem

Zhenli Sheng

szl@lsec.cc.ac.cn

Institute of Computational Mathematics and Scientific/Engineering Computing,
Academy of Mathematics and Systems Science,
Chinese Academy of Sciences

Oct 19, 2014
ORSC, Xuzhou

Outline

Outline

the Problem

Distance Geometry Problem (DGP)

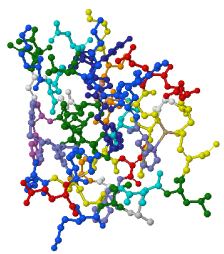
For a graph $G = (V, E)$, given a distance matrix D (L and U, respectively) on E and an integer d, find $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$, such that

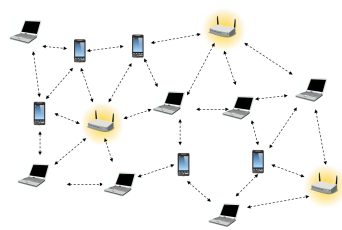
$$\|x_i - x_j\| = d_{ij}, \quad (i,j) \in E. \tag{1}$$

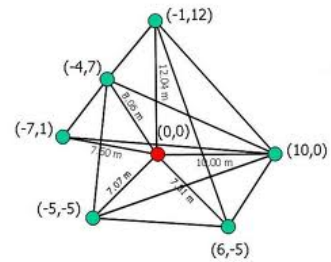
or

$$l_{ij} \leq \|x_i - x_j\| \leq u_{ij}, \quad (i,j) \in E. \tag{2}$$

- NP-hard in general
  - polynomial-time solvable instances: K-trilateration graph
- Applications: protein structure determination, sensor network localization, graph drawing, multidimensional scaling, etc







- In most instances, distances are **local**, **sparse** and **noisy**.

Solution Methods

- Matrix Decomposition Method (Blumenthal 1953, Torgerson 1958)
- The Embedding Algorithm (Crippen-Havel 1988)
- Global Smoothing Algorithm (Moré-Wu 1997)
- Geometric Buildup Method (Dong-Wu 2002, Sit-Wu-Yuan 2009, Sheng-Yuan 2014)
- SDP Relaxation Method (Biswas-Toh-Ye 2007)
- Nearest Euclidean Distance Matrix (Qi-Yuan 2013)
- The Branch-and-Prune Algorithm (Liberti-Lavor-Maculan-Mucherino 2014)
- and so on ...

Models: error functions

The problem is usually modeled as an unconstrained **global** optimization problem.

- Stress function

$$Stress(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in E} \omega_{ij}(\|x_i - x_j\| - d_{ij})^2 \tag{3}$$

  - ω_{ij} specify the degree of confidence or simply weights to balance all the terms
  - $\omega_{ij} = 1$: raw stress function
  - $\omega_{ij} = 1/d_{ij}^2$: relative errors

- Smoothed Stress function

$$SStress(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in E} \omega_{ij}(\|x_i - x_j\|^2 - d_{ij}^2)^2 \tag{4}$$

- Absolute Error function

$$AbsErr(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in E} \omega_{ij} \left|\|x_i - x_j\|^2 - d_{ij}^2\right| \tag{5}$$

- our proposed function

$$f(x_1, x_2, \ldots, x_n) = \sum_{(i,j)\in E} \omega_{ij} h(\frac{\|x_i - x_j\|}{d_{ij}}), \quad h(x) = \begin{cases} \frac{1}{2}(x-1)^2 & x \geq 1 \\ x - 1 - ln(x) & x < 1 \end{cases} \tag{6}$$

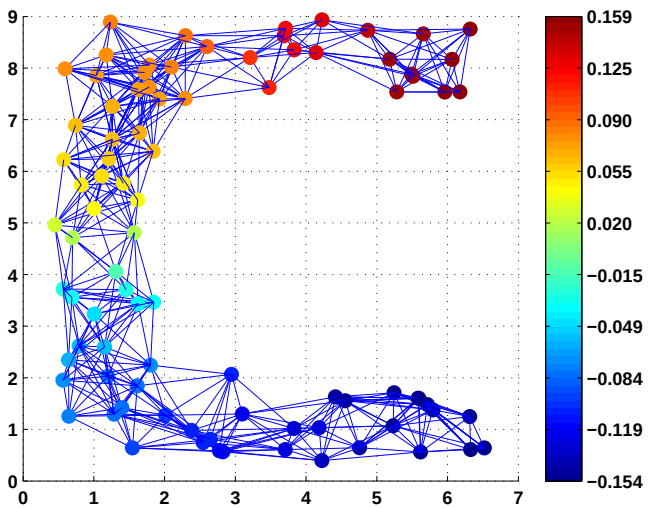
Motivation: a short story

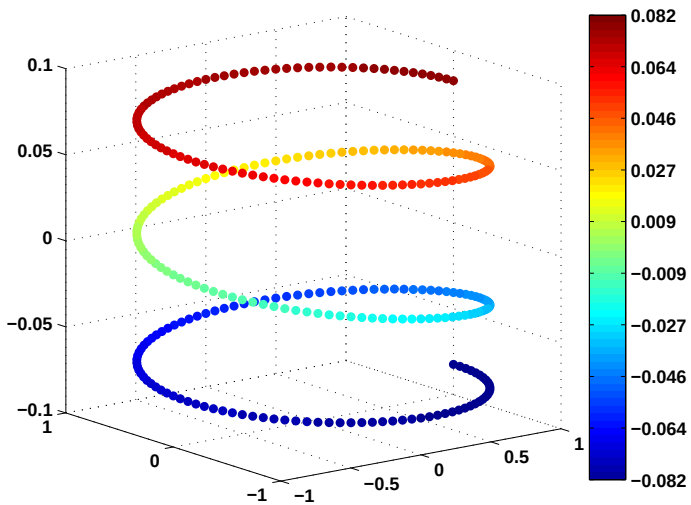
When I was an undergraduate student, I read the papers about Geometric Buildup Method and wrote `MATLAB` code to implement it, I found that

Error Accumulation is a DEVIL!

I came up with two ideas:

- design strategies to prevent error accumulation (linear/nonlinear least square)
- divide and conquer (Laplacian matrix)





Outline

Laplacian matrix

Definition: Laplacian matrix

Given a graph $G = (V, E)$ and the parameter t, the Laplacian matrix L is defined by $L = D - W$, where

$$\omega_{ij} = \begin{cases} exp(-\|x_i - x_j\|^2/t) & if\ (i,j) \in E \\ 0 & otherwise \end{cases} \tag{7}$$

and D is a diagonal matrix with

$$D_{ii} = \sum_{j \neq i} \omega_{ij}.$$

$$L_{ij} = \begin{cases} -\omega_{ij} & i \neq j \\ \sum_{k \neq i} \omega_{ik} & i = j \end{cases} \tag{8}$$

- Note that when $t = \infty$, W degenerates to the adjacency matrix.
- Research about Laplacian matrix is called **spectral graph theory**[1].
- Two normalized variations:
  - $L_{sym} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$
  - $L_{rw} := D^{-1} L = I - D^{-1} W$

[1] Chung, Fan RK. *Spectral graph theory*. Vol. 92. American Mathematical Soc., 1997.

Some properties of Laplacian matrix

For a graph G and its Laplacian matrix L with eigenvalues $\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$

1. For every vector $f \in \mathbb{R}^n$ we have

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} \omega_{ij} (f_i - f_j)^2 \tag{9}$$

2. L is symmetric and positive-semidefinite.
3. λ_0 is always 0 because every Laplacian matrix has an eigenvector $v_0 = [1, 1, \ldots, 1]^T$.
4. The number of times 0 appears as an eigenvalue in the Laplacian is the number of connected components in the graph.

Proof of (1):

$$f^T LF = f^T Df - f^T Wf = \sum_{i=1}^{n} d_{ii} f_i^2 - \sum_{i,j=1}^{n} f_i f_j \omega_{ij}$$

$$= \frac{1}{2} \Big(\sum_{i=1}^{n} d_{ii} f_i^2 - 2 \sum_{i,j=1}^{n} f_i f_j \omega_{ij} + \sum_{j=1}^{n} d_{jj} f_j^2 \Big) = \frac{1}{2} \sum_{i,j=1}^{n} \omega_{ij} (f_i - f_j)^2$$

Outline

Nonlinear dimensionality reduction

Dimensionality Reduction Problem

Given a set of n points $x_1, \ldots, x_n$ in $\mathbb{R}^l$, find a set of points $y_1, \ldots, y_n$ in $\mathbb{R}^m (m \ll l)$ such that y_i "represents" x_i.

Relation to DGP:

- In noisy case, the distances can not be **precisely** realized in low-dimensional space.
- If the distances do not contradict with each other, the points can be precisely realized in high-dimensional space.
- a two-dimensional example

Optimal embedding

Problem: Map the weighted graph G to a line such that the adjacent vertices stay as close as possible.

Let $y = (y_1, y_2, \ldots, y_n)^T$ be such a map, so we want to minimize

$$\sum_{(i,j)\in E} (y_i - y_j)^2 \omega_{ij} = y^T L y \tag{10}$$

Therefore,

$$\min_{y^T D y = 1, y^T D e = 0} y^T L y \tag{11}$$

gives us a reasonable solution.

General case: Embed the graph into m-dimensional Euclidean space.

Let $Y = [y_1 \ y_2 \ \ldots \ y_n]^T \in \mathbb{R}^{n\times m}$ that each row gives a coordinate. Similarity we need to minimize

$$\sum_{(i,j)\in E} \|y_i - y_j\|^2 \omega_{ij} = tr(Y^T L Y) \tag{12}$$

Courant-Fischer Theorem

Courant-Fischer Theorem

Let A be a symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and corresponding eigenvectors $v_1, v_2, \ldots, v_n$. For $1 \leq k \leq n$, let S_k denote the span of $v_1, \ldots, v_k$ (with $S_0 = \{0\}$), and let $S_k^{\perp}$ denote the orthogonal complement of S_k. Then

$$\lambda_k = \min_{\|x\|=1, x \in S_{k-1}^{\perp}} x^T A x = \min_{x \in S_{k-1}^{\perp}} \frac{x^T A x}{x^T x}$$

Nonlinear dimensionality reduction algorithm

Nonlinear dimensionality reduction algorithm framework (Belkin-Niyogi, 2002)

Step 1 Constructing the adjacency graph
(a) ϵ-neighborhood (b) k-nearest neighbors

Step 2 Choosing the weights
(a) Heat kernel (b) Simple-minded

Step 3 Eigenmaps: $x_i \to (f_1(i), \ldots, f_m(i))$, where $Lf_i = \lambda_i Df_i$, $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{n-1}$.

Grid points example 1

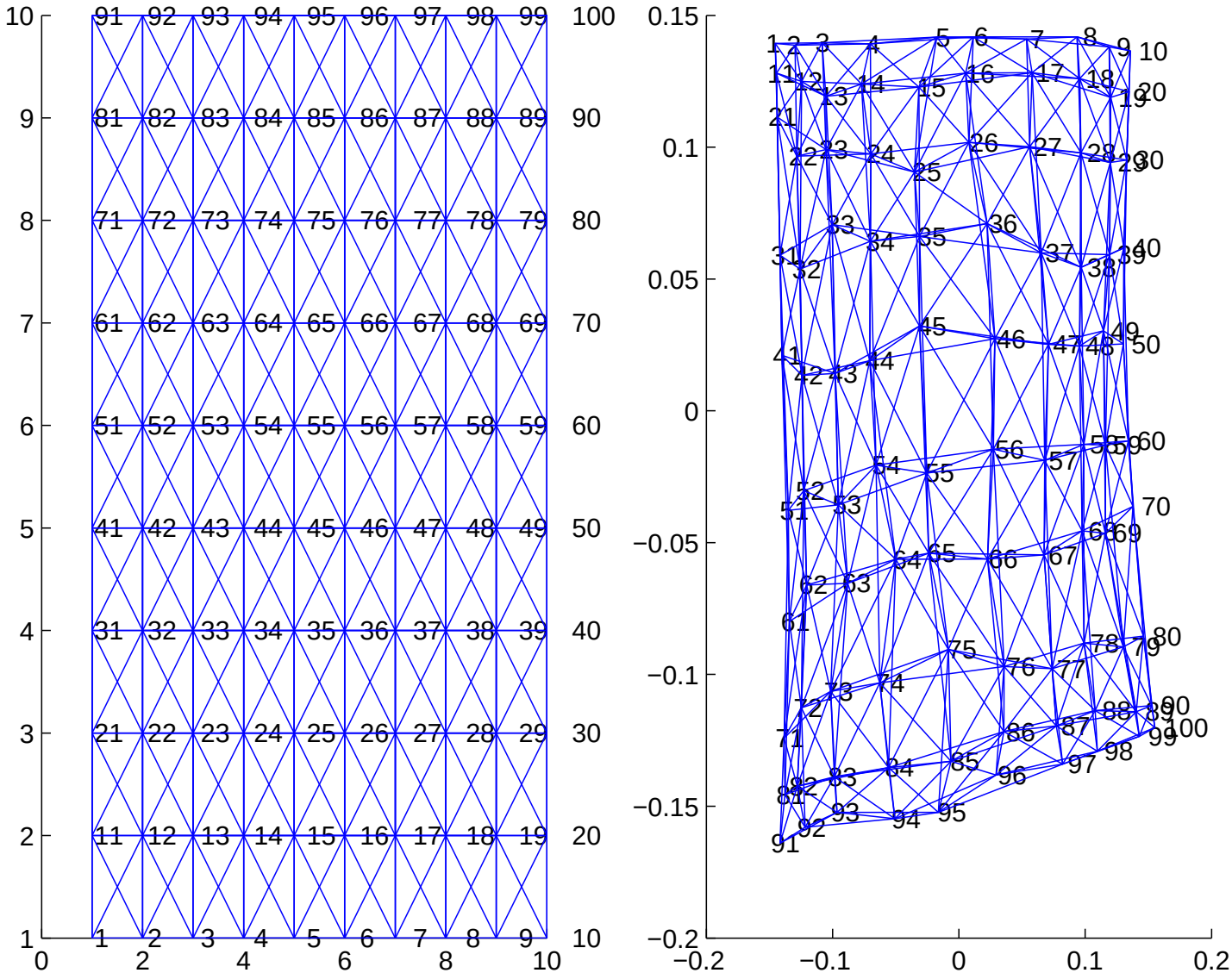


Figure 1 : cutoff=2, degree: [12,5,10], noise = 20%

Grid points example 2

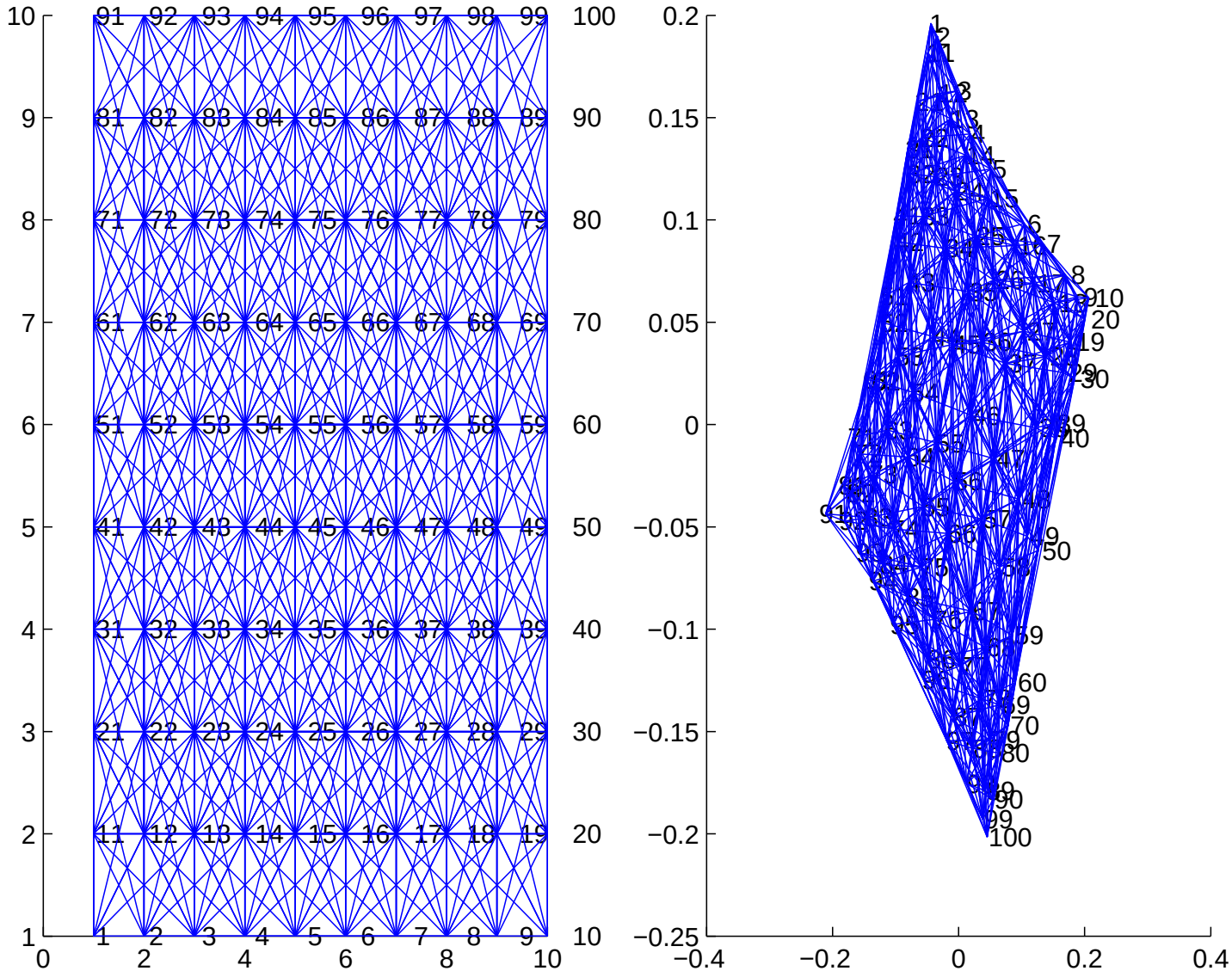
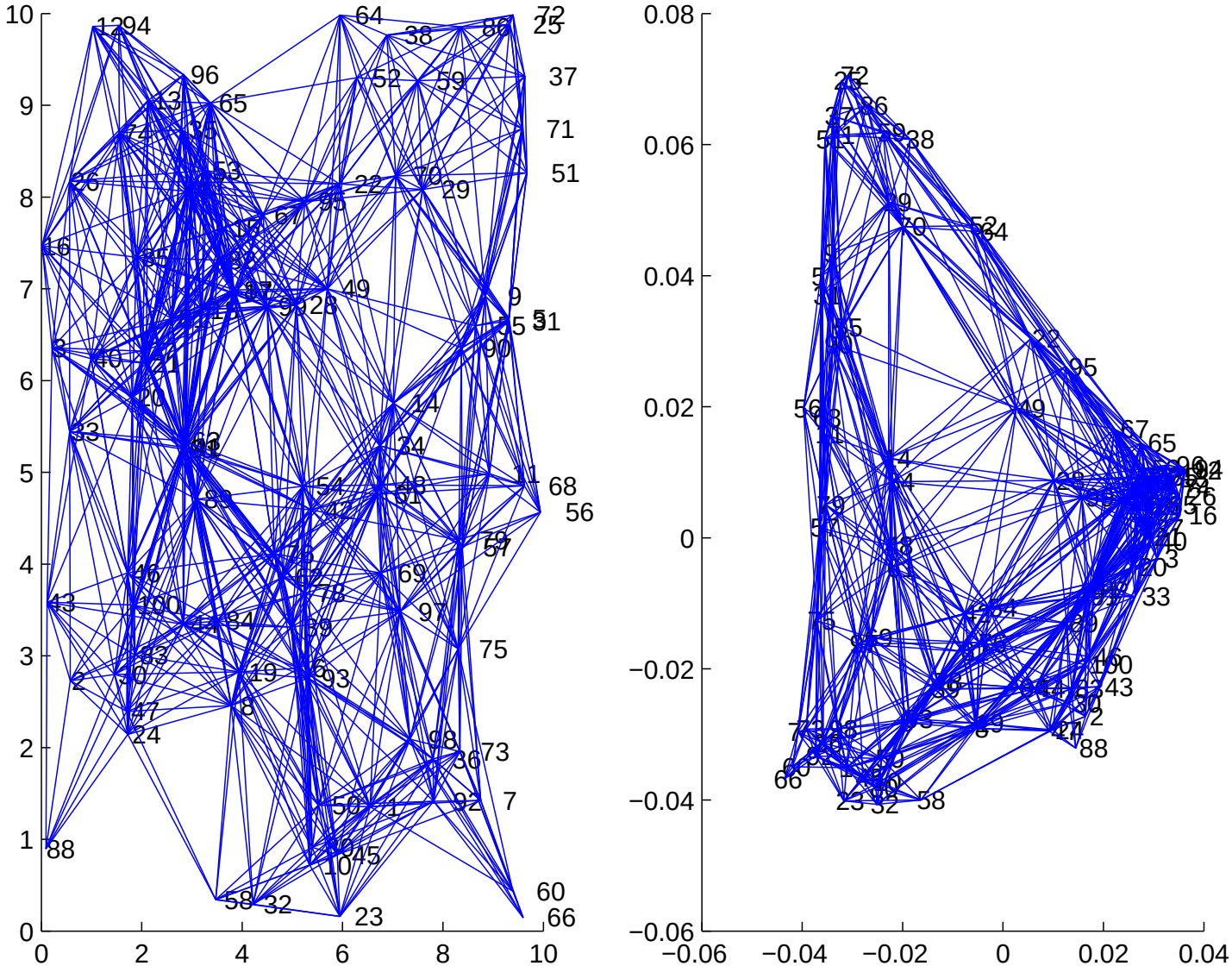
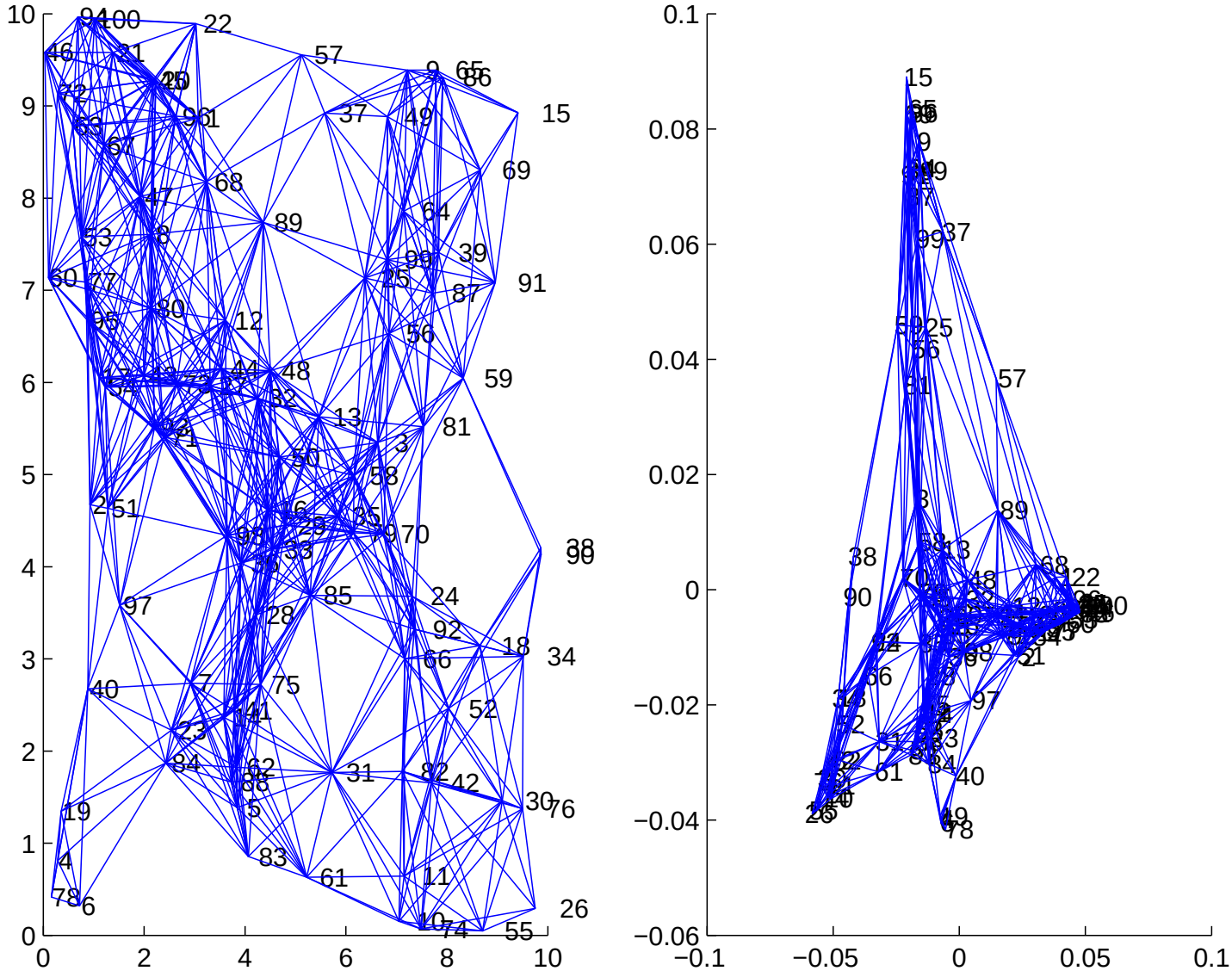


Figure 2 : cutoff=3, degree: [20,10,21.2], noise = 20%

a random example



another random example



Remarks about Laplacian eigenmap

- Accurate locations can not be recovered directly.
- Useful relative relation can be inferred.
- Balanced graph are preferred.

Outline

Laplacian eigenmap-based localization algorithm

Laplacian Eigenmap-based Localization Algorithm for DGP

Step 1 Construct the Laplacian matrix L and calculate its eigenvalue decomposition $L = VDV^T$ such that the eigenvalues in D are in ascending order.

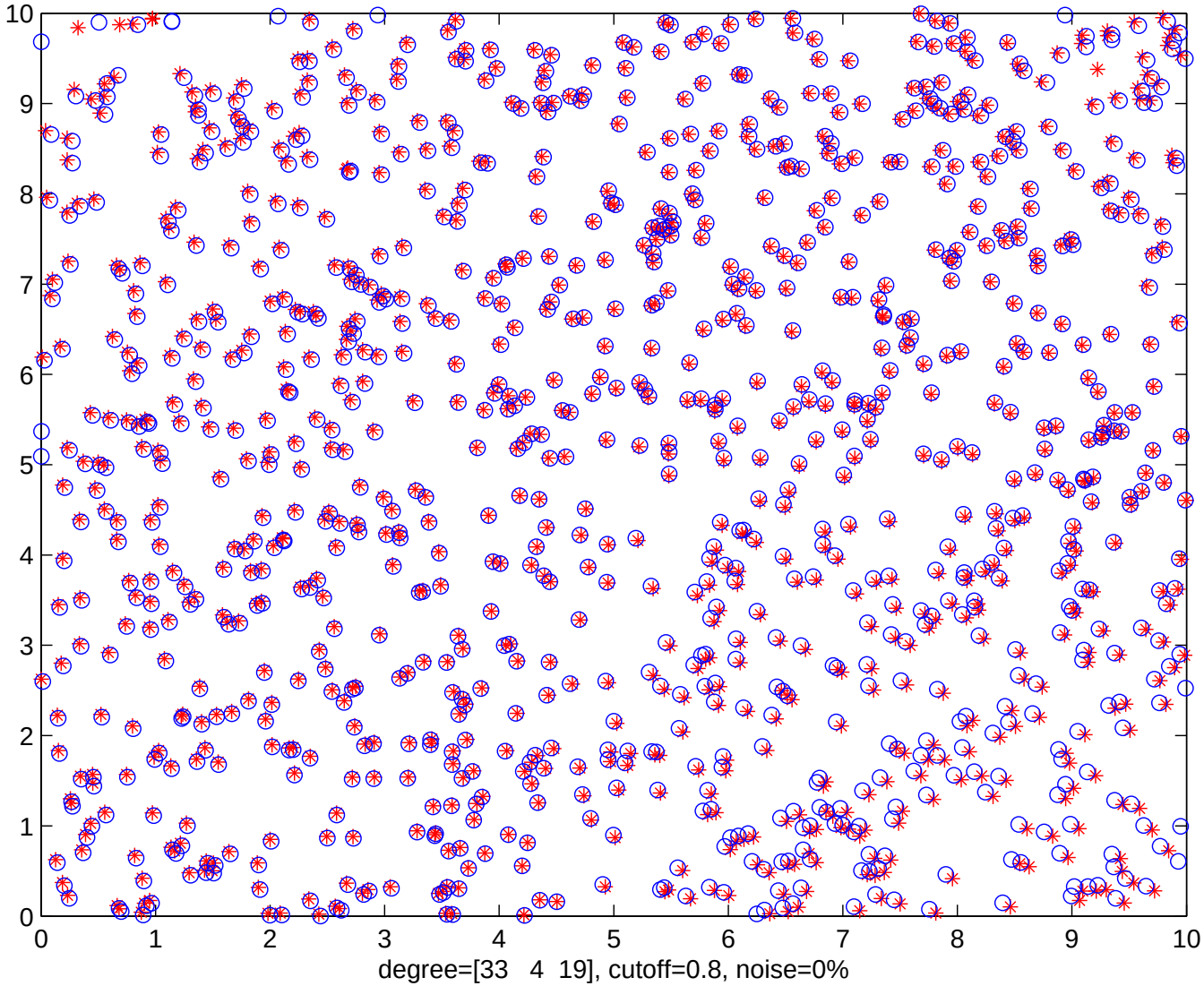
Step 2 Set $X0 = V(:, 2 : m + 1)$ and scale it to the proper magnitude.

Step 3 Using $X0$ as the initial point, apply error function minimization method to further improve the result.

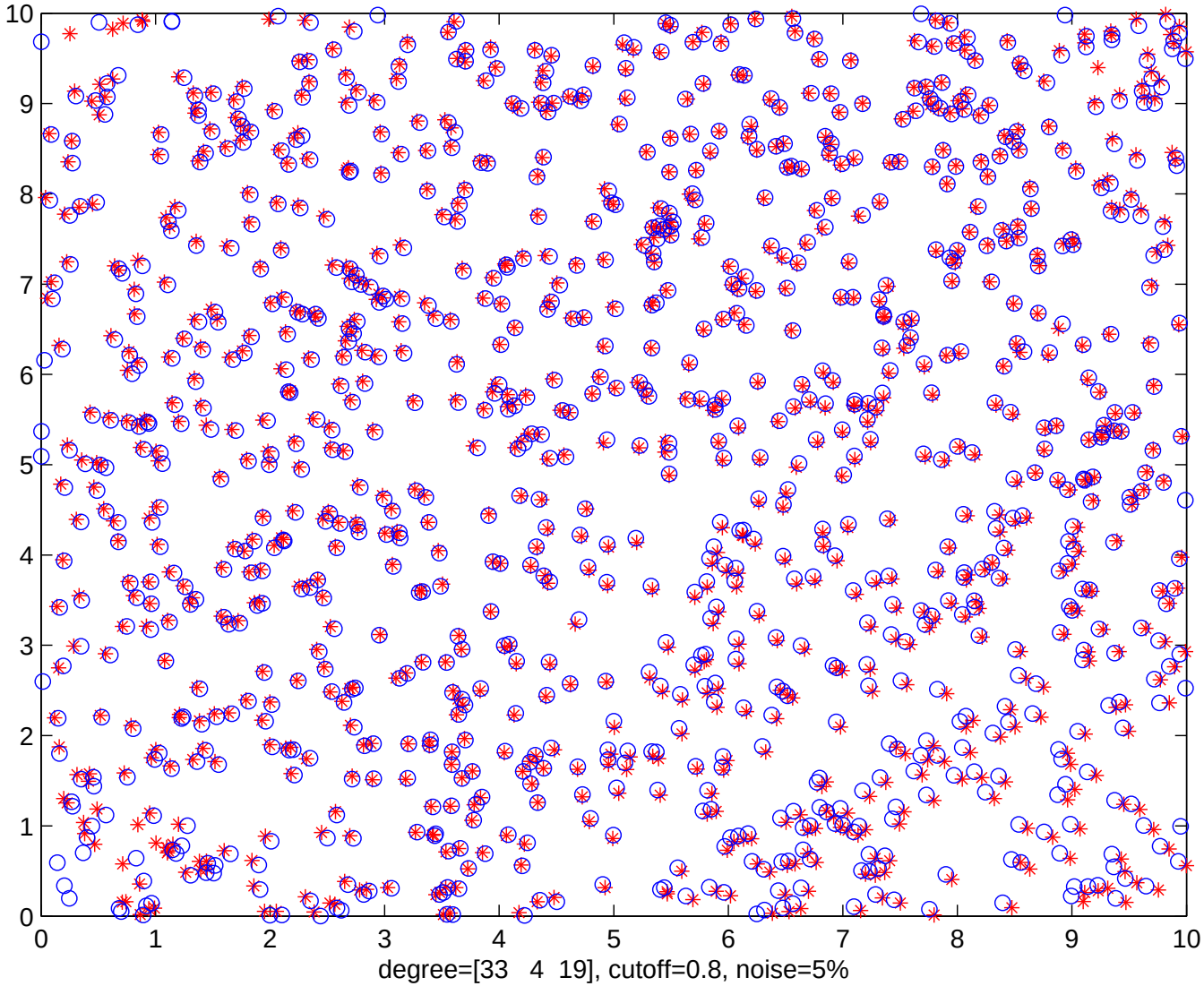
Scaling method

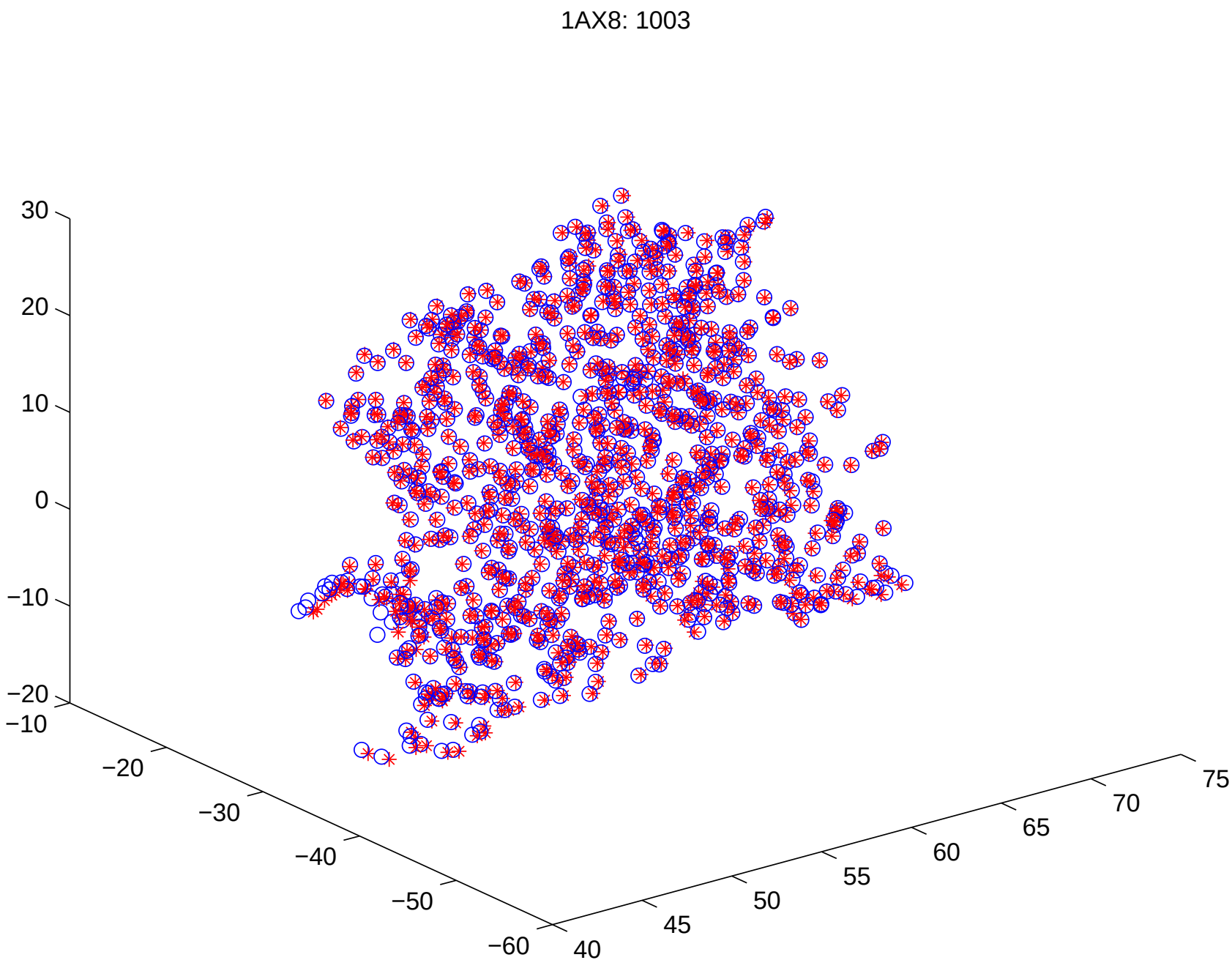
1. Construct a new distance matrix $\tilde{D}$ according to $X0$ and E.
2. Calculate
$$ratio = \frac{sum(D)}{sum(\tilde{D})}.$$
3. Set $X0 = X0 * ratio$.

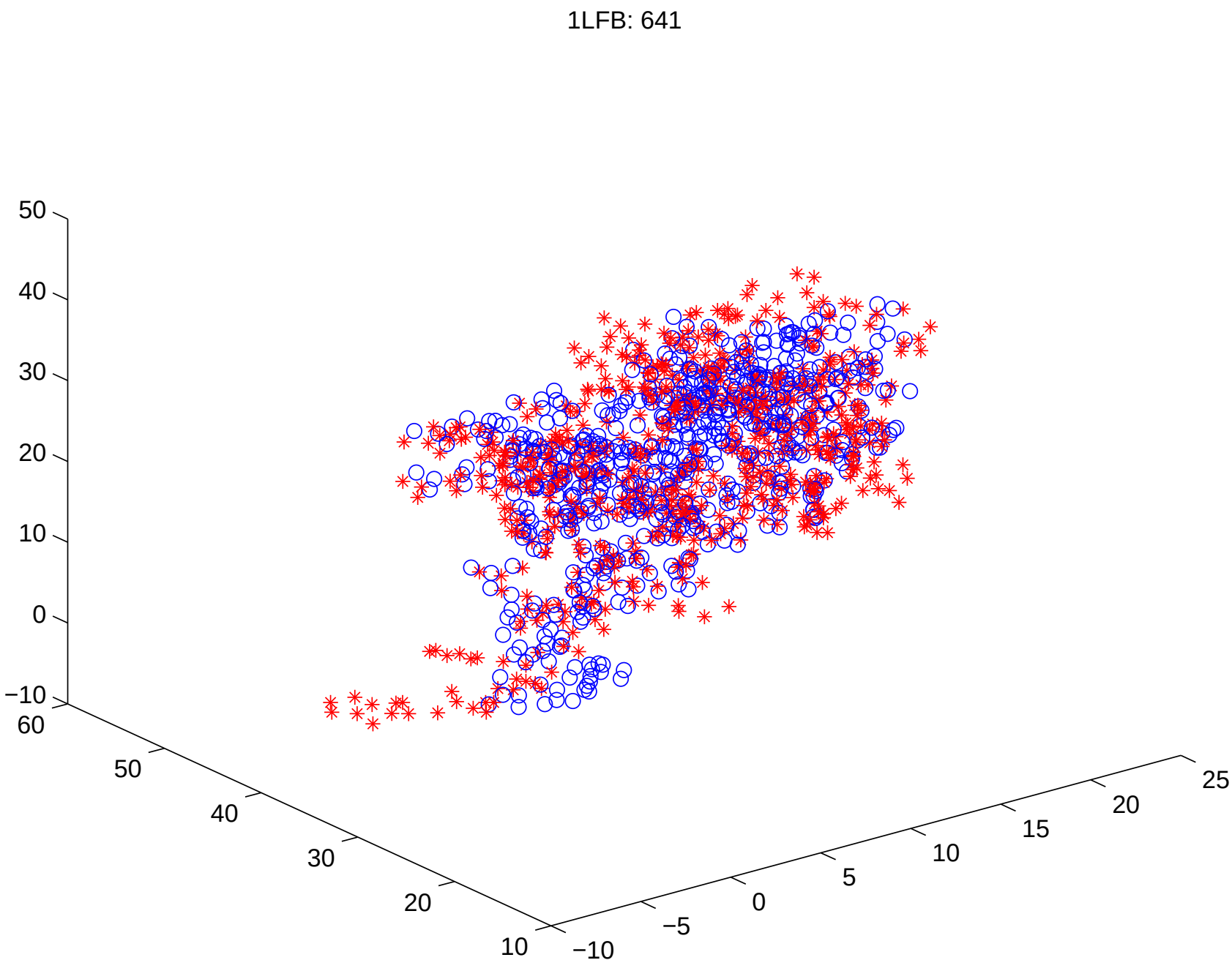
Random example



Random example



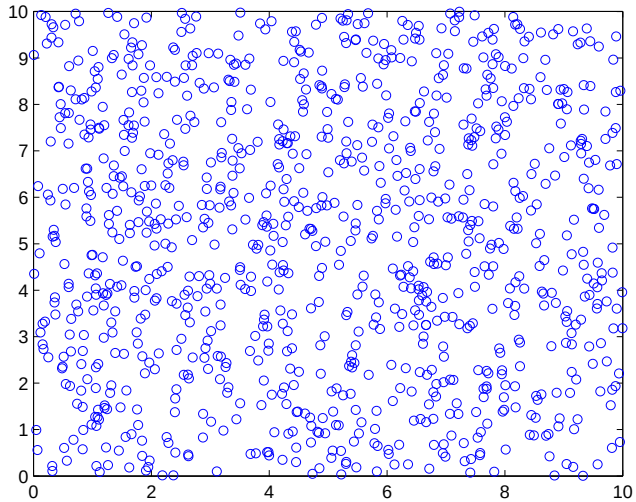

1AX8: 1003
30
20
10
0
−10
−20
−10
−20
−30
−40
−50
−60
40
45
50
55
60
65
70
75

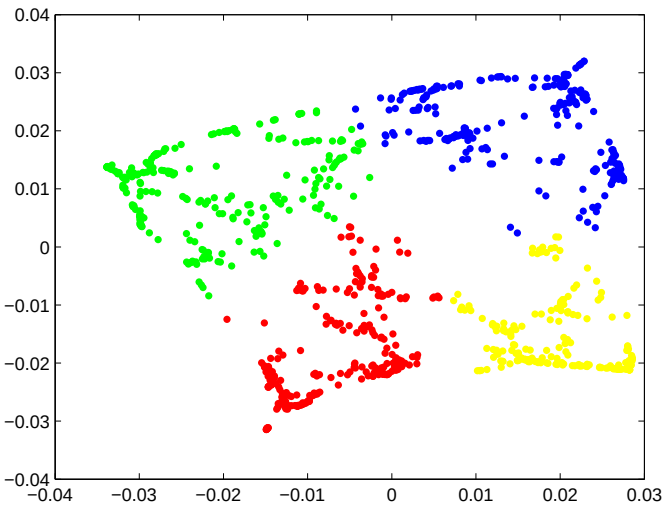

1LFB: 641
50
40
30
20
10
0
−10
60
50
40
30
20
10
−10
−5
0
5
10
15
20
25

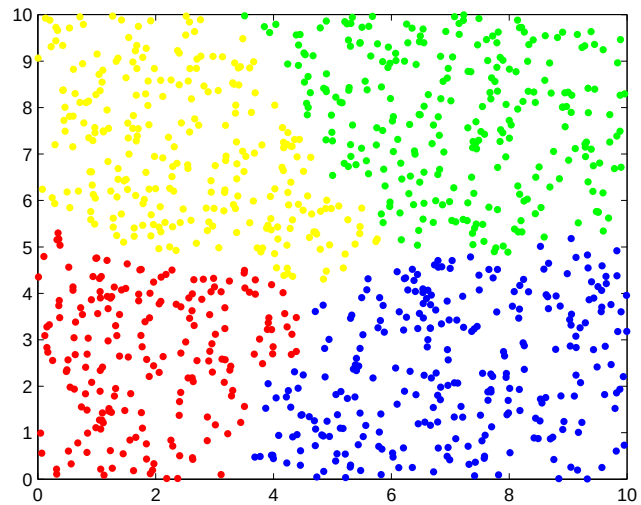
Discussions

- Pros
  - very simple and fast
  - relatively robust to the noise
  - good in uniformly generated random instances
- Cons
  - sensitive to the parameter t and no theoretically-guaranteed good way to choose it
  - terrible in most of real protein instances, except for 1PTQ, 1HOE, 1AX8

possible distributed algorithm







Outline

Conclusion and Future Work

- Conclusion
  - Proposed a Laplacian Eigenmap-based algorithm for DGP, which is a totally new idea for solving this NP-hard problem.
  - Finished some preliminary numerical experiments to show its efficiency.
- Future Work
  - Figure out the problem with protein computation.
  - Carefully study the parameters in Laplacian matrix.
  - Develop distributed algorithm based Laplacian Eigenmap.

Thank you for your attention!

szl@lsec.cc.ac.cn