Note - 2017 Summer Research

July 2, 2017

Contents

1 Questions

1.1 MMF Related

1. Finding the optimal Givens rotation:
   (a) Proposition 4 proposes to find the roots of an optimality condition derived in the supplement. However, the pMMF library appears to avoid the A matrix altogether. We calculated that if we ignore A then the optimal alpha should be $\arctan(b/c)/2$.
   (b) If this is what is being done, how do we interpret the choice in terms of optimization?
   (c) Why is A in Proposition 4 supposed to be a diagonal matrix
2. In equation (10) on the multiresolution, should the second to last term be $dim(V_l), i$ instead of $dim(V_{l-1}), i$?
3. How is the clustering done?

   Function called compression map in MMFmatrix.hpp
4. Algorithm 2
   (a) If A is a tall, skinny matrix, how can it be reset by multiplying on left and right by a square matrix of same dimensions
   (b) Why is it the rows of A that are used to determine which one to eliminate? (or this is because A is always symmetric). In this case, A is referring to square matrix, not the tall skinny matrix defined in the in
5. Finding the optimal rotations when K>2

   In Section 2.2 of second paper, "actual rotation angle is determined by diagonalizing the Gram matrix." How? Maybe check the code.

1.2 other

1. How to quantify the relationship between reduced and original Laplacian eigenvectors?

2 Lit Review - May 22 - May 26

2.1 MMF, Parallel MMF

2.2 A Multiscale Pyramid Transform for Graph Signals

2.3 Signal Processing on Graphs

3 Givens Rotations, Sparse QR - May 26

3.1 A 2×2 Review

- A 2-by-2 orthogonal matrix Q is a **rotation** if it has the form

$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

  For $x \in \mathbb{R}^2$, $Q^T x$ rotates x counterclockwise through θ.

- A 2-by-2 orthogonal reflection matrix Q is a **reflection** if it has the form

$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$$

  For $x \in \mathbb{R}^2$, $Q^T x = Qx$ reflects x with respect to

$$\text{span}\left\{ \begin{bmatrix} \cos\theta/2 \\ \sin\theta/2 \end{bmatrix} \right\}$$

3.2 General Givens Rotations

- Givens rotations are used to selectively zero elements of a matrix
- Rank 2 corrections to the identity
- For $c = \cos\theta$ and $s = \sin\theta$,

$$G(i,k,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & s & & & \\ & & -s & c & & & \\ & & & & \ddots & & \\ & & & & & 1 & \end{bmatrix}$$

  G rotates $x \in \mathbb{R}^n$ through θ in (i,k) coordinate plane.

- The rotation $G(i,k,\theta)^T \cdot A$ only affects the i^{th} and k^{th} rows/columns of matrix A.

3.3 Sparse QR using Givens Rotation

Given a matrix A, the QR factorization is unique. In Q, some of the columns have to span a particular subspace and the remaining columns have to form an orthogonal basis for the complement to this subspace. Thus, the orthogonal matrix cannot be sparse. Sparse QR method represents Q as a product of Givens rotations, which are extremely sparse/structured matrices.(https://mathoverflow.net/questions/94198/sparsity-of-qr-decomposition)

3.4 Sparse QR Challenge

- Given a sparse matrix $A \in \mathbb{R}^{m \times n}$, determine a permutation p of $[n]$ such that, if $P = I_n(:,p)$, then the R-factor in the thin QR-factorization $AP^T = QR$ is close to being optimally sparse.
- Use orthogonal transformations to determine R from AP^T.
- Reduces the problem $Ax = b$ down to $R^T RPx = PA^T b$, which is easier computationally when R is optimally sparse. Additionally, we now no longer have to compute Q.

4 pMMF

4.1 Wavelet Transform - June 5-7

Given a symmetric matrix $A \in \mathbb{R}^{n \times n}$, a L level pMMF factorization will result in

$$H = P_{L+1}(\overline{Q}_L P_L) \cdots (\overline{Q}_2 P_2)(\overline{Q}_1 P_1) A (P_1^T \overline{Q}_1^T)(P_2^T \overline{Q}_2^T) \cdots (P_L \overline{Q}_L^T) P_{L+1}^T$$

Denote $Q_L = \overline{Q}_L P_L$ and we get

$$H = P_{L+1} Q_L \cdots Q_2 Q_1 A Q_1^T Q_2^T \cdots Q_L^T P_{L+1}^T$$

Notice that $(P_{L+1} Q_L \cdots Q_2 Q_1)^T$ forms the dictionary for level L. The first $\dim(V_L)$ columns are scaling functions and the rest of its columns are wavelets. The dictionary has the form of:

$$\left[\begin{array}{c|c|c|c|c} V_L & W_L & W_{L-1} & \cdots & W_1 \end{array}\right]$$

where V_L denotes the scaling functions at level L and each W_i denotes the wavelet functions at level i.

The scaling and wavelet functions have the following properties:

1. Scaling functions and wavelets are orthonormal. That is, the dictionary is an orthonormal matrix.
2. Scaling Wavelet functions are localized in both vertex and frequency domain.
3. Wavelets centered at the very first nodes eliminated have highest frequencies - these are the wavelet functions which make up W_1.
4. Last vertices remaining are the centers of the scaling functions in V_L.
5. $\text{span}(V_i) = \text{span}(V_{i+1} \oplus W_{i+1})$

4.1.1 Example 1: 16×16 Community Graph

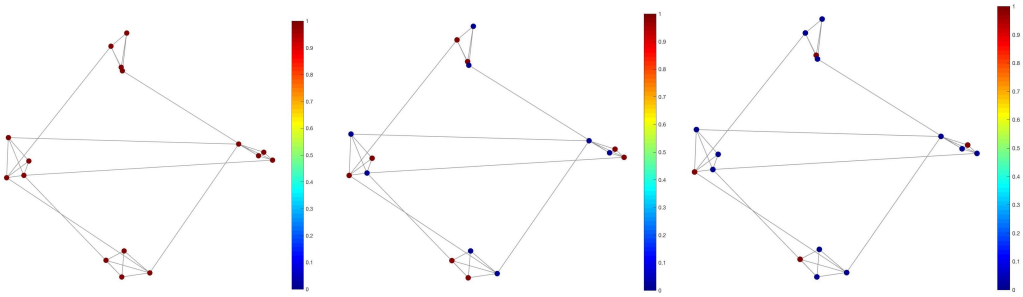

Figure 1: Downsampling 16-by-16 community graph. Red vertices are kept after each stage. Notice that after 2 stages, 4 kept vertices are from 4 different communities.

Dictionary Atoms:

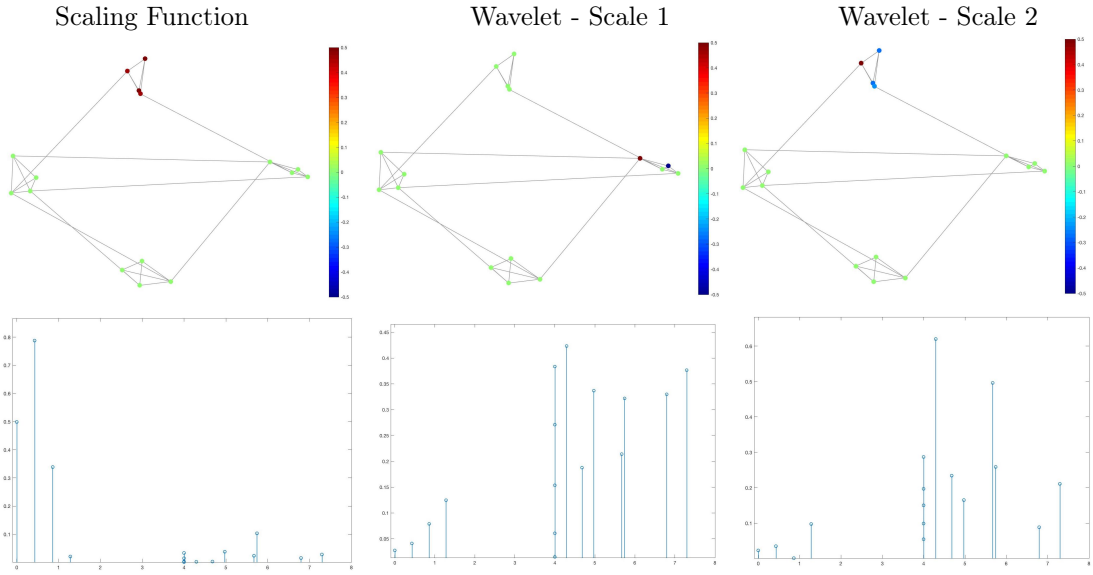

Figure 2: Dictionary atoms in both vertex and frequency domains.

4.1.2 Example 2: 500×500 Sensor Network

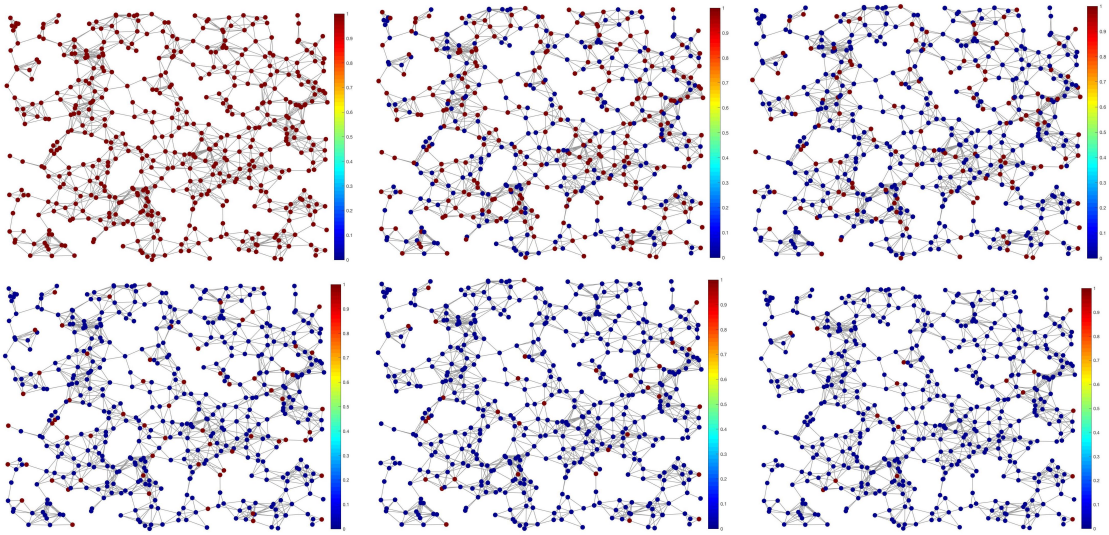

Figure 3: Downsampling 500-by-500 sensor network.
Notice that after each stage, kept vertices cover the whole graph instead of centering at a specific place.

Dictionary Atoms:

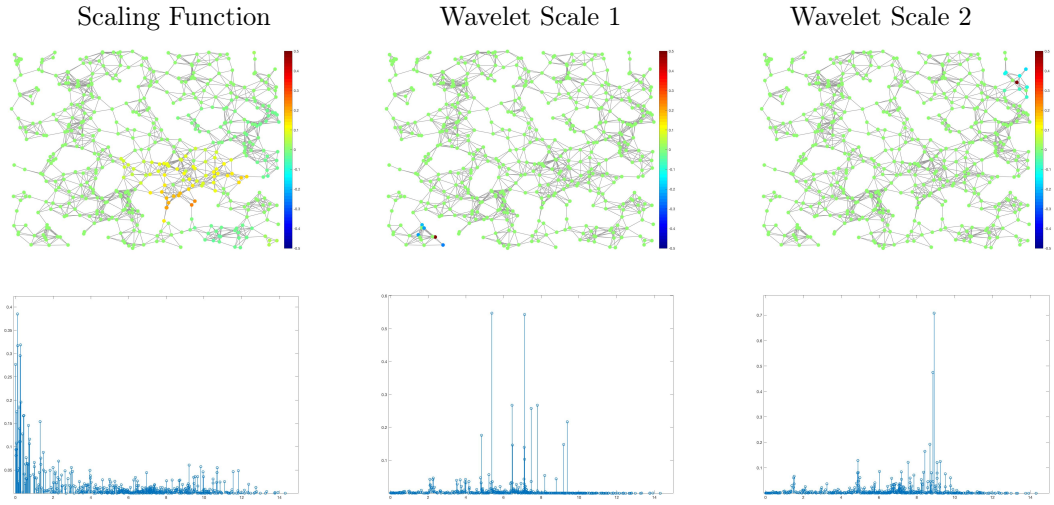

Figure 4: Dictionary Atoms in both vertex and frequency domain.

4.2 MMF on diffusion operator - June 8-12

Given graph G, let $G.U$ be eigenvectors of its graph Laplacian and $G.e$ be corresponding eigenvectors. Let $A = G.U * f(G.e) * G.u^T$, where $f(\lambda) = e^{-\tau\lambda}$. One level of MMF on matrix A results in

$$H = P_2 Q_1 A Q_1^T P_2^T$$

Assume H is a diffusion operator on some reduced graph.
Tried to (not sure if it's even a valid method):

1. Diagonalize $H = VDV^T$
2. Construct a graph Laplacian $L = Vf^{-1}(D)V^T$

Failed: L is not a Laplacian on downsampled vertices.

Random thoughts:

1. Are there any methods to construct a graph Laplacian from H?
2. Find L s.t. minimizes $|f(L) - H|$?

4.3 MMF on weighted adjacency matrix W - June 12

Some general observations:

1. The core H has non-zero diagonals.
2. The new graph is not necessarily connected.
3. The new graph is still pretty sparse.
4. The current clustering method doesn't yield great downsampling results for path and ring graphs.
5. The clusters have great influences on how downsampled graphs are rewired.

4.3.1 Experiments on path graphs

MMF works well for path graphs in general. The core has zero diagonals and the new graph is connected (if #clusters = 1). The lowest eigenvector of the reduced Laplacian and of the original Laplacian are both constants.

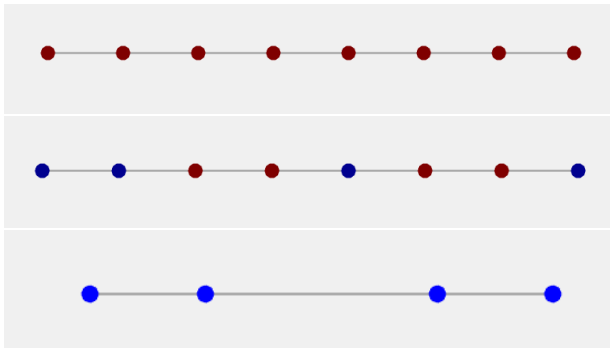

Figure 5: a.original graph b.downsampled graph(red vertices are kept) c. reduced graph

4.3.2 Experiments on ring graphs

For ring graphs, the core has zero diagonals, but the new graph is not necessarily connected. The new graph captures the ring structure. The lowest eigenvector of the reduced Laplacian and of the original Laplacian are both constants.

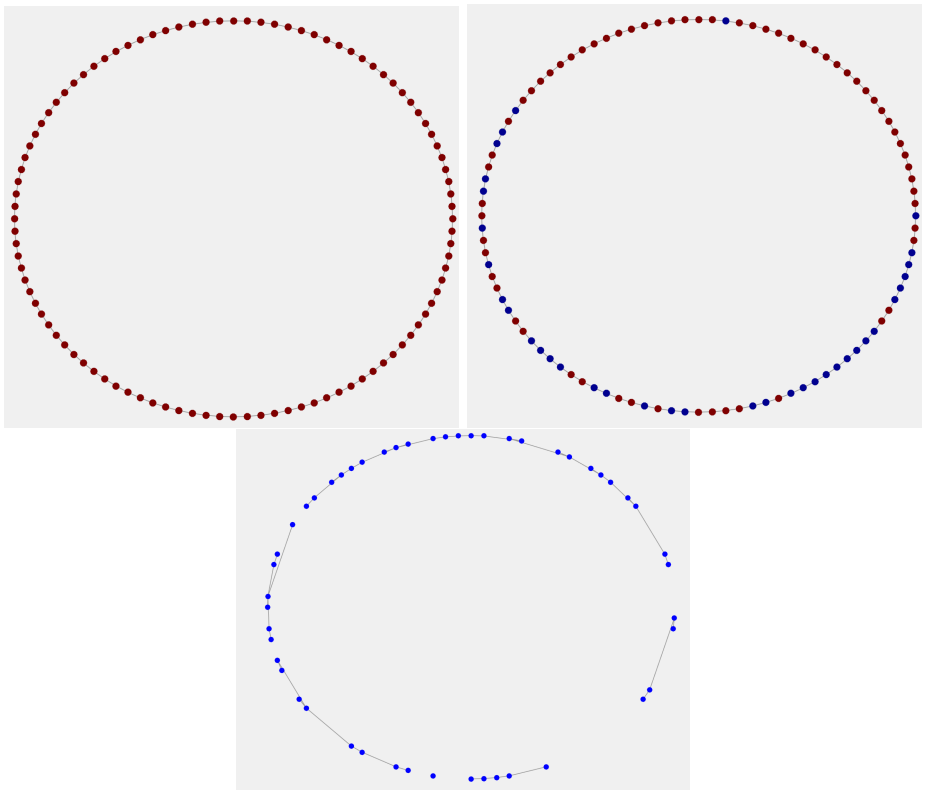

Figure 6: a.original graph b.downsampled graph(red vertices are kept) c. reduced graph

The downsampled vertices greatly influence how the reduced graph is rewired. It seems like a disconnection occurs if most of the vertices between the two are eliminated.

With a evenly downsampled graph (every other vertex is kept), can we get better results? Maybe experiment with different clustering method to get a evenly downsampled graph?

Note: If we perform MMF on the Laplacian matrix, the reduced matrix we get is not a valid graph Laplacian.

4.4 Clustering

- Although clustering was only implemented to speed up the algorithm, it seems like larger numbers of clusters lead to more disconnected graphs.

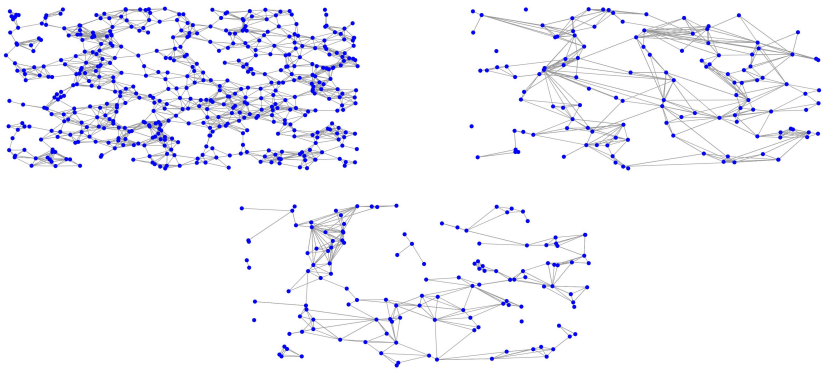

Figure 7: 1. Original graph (gsp_david _sensor_network) with 500 vertices. 2. Downsampled graph using 5 clusters. 3. Downsampled graph using 20 clusters.

- This seems to a relatively consistent pattern, but not a necessary one (i.e. sometimes a downsampled graph using 2 clusters has 3 components while a downsampled graph using 5 clusters will have 2).
- Also along these lines - the gsp_check_connectivity_undirected function doesn't seem to work correctly (at least as it's described in the documentation), it said that connected graphs were unconnected pretty frequently.

4.5 $A = L$ vs. $A = I - L$

- When $A = L$, our weighted adjacency matrix is $W = \text{diag}(\text{diag}(H))$ - H.
- When $A = I - L$, our Laplacian is $L = I - H$ and our weighted adjacency matrix is $W = \text{diag}(\text{diag}(L))$ - L.

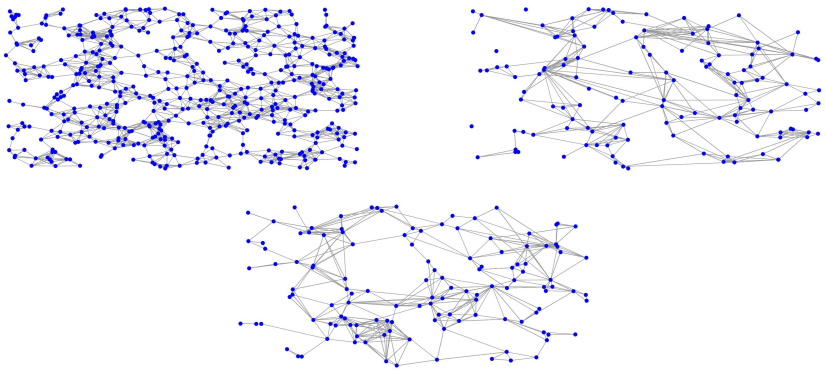

Figure 8: 1. Original graph (gsp_david _sensor_network) with 500 vertices. 2. Downsampled graph using $A = L$. 3. Downsampled graph using $A = I - L$.

- The two are very comparable, but it seems like the downsampling using $A = I - L$ captures the sparsity/density of the original graph a bit better.

4.6 Edge Weights

Compared the proportion of edges with high weight (> 0.9) in a gsp_david_sensor_network graph with 500 nodes and the downsampled graph using A = W, A = L, and A = I - L.

22.15% of the edges on the original graph had high weight, and

- using A = W, 29.28% of the edges had high weight
- using A = L, 21.82% of the edges had high weight
- using A = I - L, 22.00% of the edges had high weight

When we look at the n highest-weighted edges (however many have weight > 0.9) in the original graph G and the $n \cdot \frac{Gdownsampled.N}{G.N}$ highest-weighted edges in the downsampled graph, we see that the areas of the downsampled graph which contain high-weighted edges most closely mirror the analogous areas on the original graph when $A = I - L$. $A = W$ does okay, but when $A = L$, the downsampled graph seems to ignore some high-weight areas from the original graph.
These judgments are all just visual, so next steps would be to find a way to quantify high-weight edge retention numerically.

5 Plot Reduced Laplacian Eigenvectors

Parameters:

- G: gsp_david_sensor_network
- 2 stages
- 1 cluster

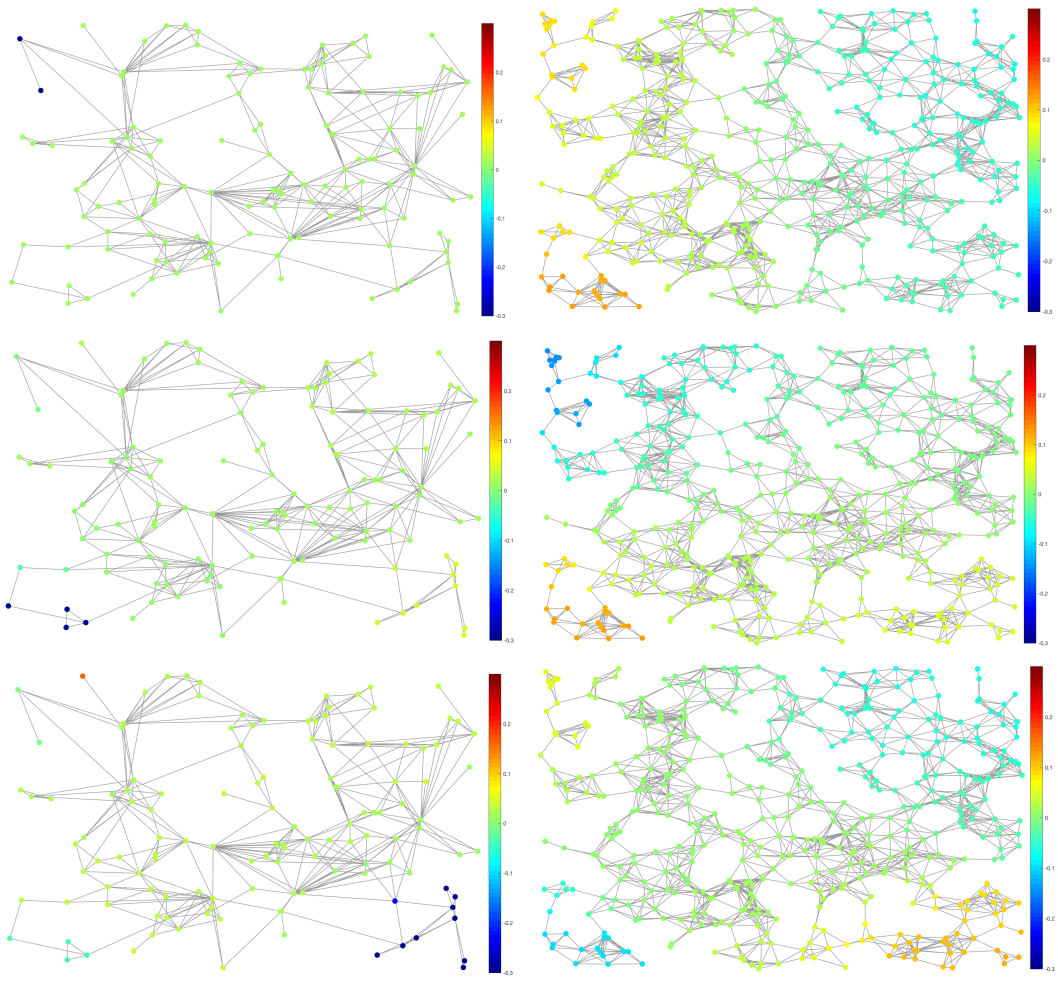

Figure 9: The left column shows the 2nd, 3rd and 4th eigenvectors of the reduced graph Laplacian. The right column shows the 2nd, 3rd and 4th smallest eigenvectors of the original graph Laplacian.

6 Interpolation

The adjacency matrix of the reduced graph is obtained from $H - Diag(H)$. Then we interpolate the eigenvectors of the reduced graph Laplacian and compare them to the eigenvectors of the original graph Laplacian.

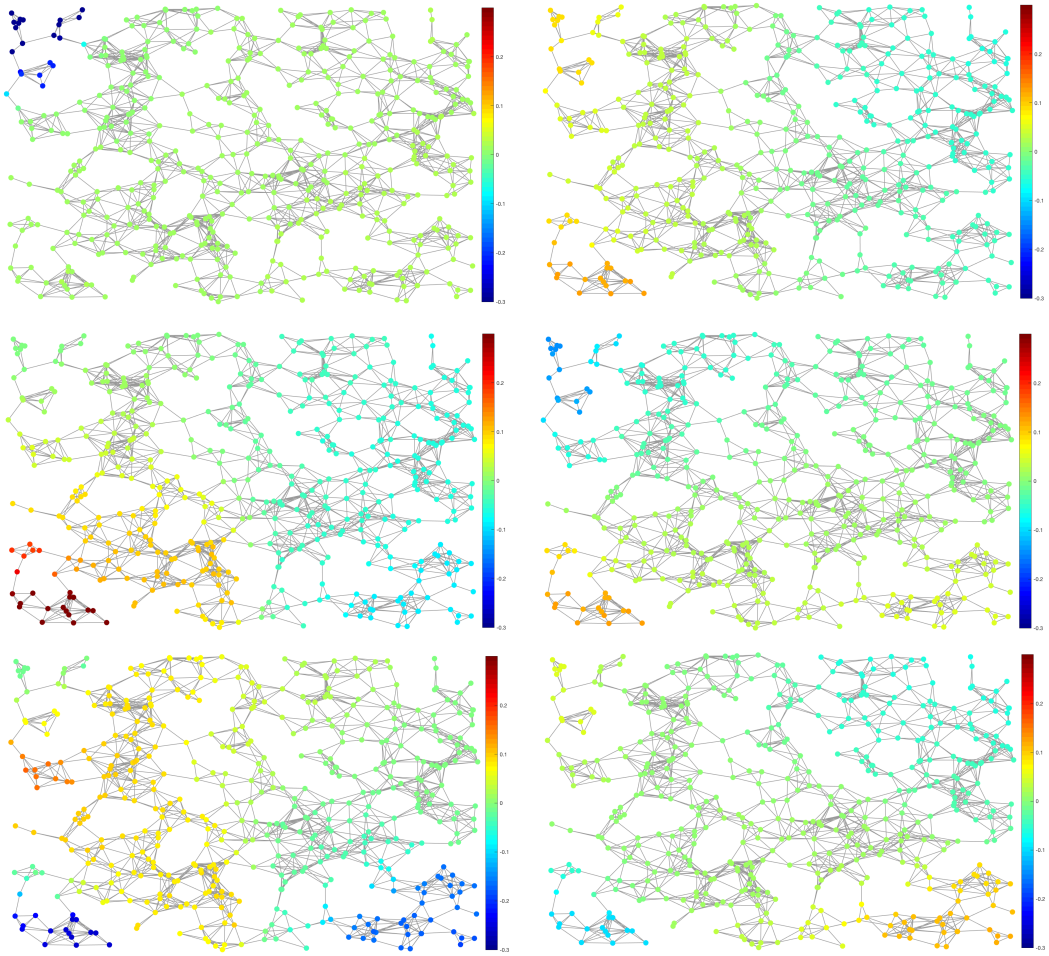

Figure 10: The left column shows interpolations of the 2nd, 3rd and 4th smallest eigenvectors. The right column shows the 2nd, 3rd and 4th smallest eigenvectors of the original graph

7 pMMF Multiresolution

7.1 pMMF Detailed Pseudocode

Algorithm 1 ComputeRotationMatrix

Input: current cluster A_1, row/column indices i and j, current Gram Matrix G
set $b = G(i, j)$
set $c = (G(j, j) - G(i, i))/2$;
if $g \neq 0$ **then**
    set $\beta = c/b$
    set $tan(\alpha) = \frac{1}{|\beta|+\sqrt{\beta*\beta+1}}$
    set $cos(\alpha) = \frac{1}{\sqrt{(t*t+1)}}$;
    set $Q = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$
else
    $Q = I$
Output: Q

Algorithm 1 finds an orthogonal rotation matrix $Q = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$ such that the second term of equation 20 is minimized when $k = 2$. That is to find an angle $\theta = 2\alpha$ such that

$$\sin(\theta + \omega + \pi/2) = 0, \text{, where } \omega = \arctan(\frac{G(i,i) - G(j,j)}{2G(i,j)})$$

Let $b = G(i, j)$ and $c = \frac{G(j,j)-G(i,i)}{2}$;
Equivalently,

$$\cos(\theta + \omega) = 0$$

$$\theta = \pi/2 - \omega = \pi/2 - \arctan(\frac{c}{b})$$

$$\tan(\alpha) = \tan(\theta/2) = \tan(\pi/4 - \frac{1}{2}\arctan(\frac{c}{b})) \text{ (if } \frac{c}{b} > 0)$$

$$\tan(\alpha) = \frac{1}{\frac{c}{b} + \sqrt{(\frac{c}{b})^2 + 1}}$$

$$\cos(\alpha) = \frac{1}{\sqrt{\tan(\alpha)^2 + 1}}$$

$$\sin(\alpha) = \tan(\alpha) * \sin(\alpha)$$

When $k > 2$, the rotation matrix Q is the eigenvectors of the $G_{I,I}$, where I contains k selected column indices. (why?)

Similarly, only the second term of equation 20 is minimized. That is,

$$\min\ [QG_{I,I}Q^T]_{k,k}$$

Equivalently,

$$\min\ x^T[G_{I,I}]x \text{ such that } ||x||_2 = 1.$$

By Courant-Fischer theorem,

$$\min\ x^T[G_{I,I}]x = \lambda_1$$

Let x be the smallest eigenvector v_1 of $G_{I,I}$. Then,

$$\min\ x^T[G_{I,I}]x = \min\ v_1^T[G_{I,I}]v_1 = v_1\lambda_1 v_1 = \lambda_1$$

To find k rows/clumns involved in each rotation, pMMF uses the randomized greedy method. First, randomly select a column i, and then compute the inner product of this column with all other columns. The selected column set will include $k - 1$ columns that have the largest inner

products with column i. Algorithm 2 in "Parallel MMF: a Multiresolution Approach to Matrix Computation" gives a good description of this process.

Note that after we get the rotation matrix Q, we have to update the original matrix $A = QAQ^T$ and the Gram matrix $G = QGQ^T$ before next channel.

Algorithm 2 pMMF

Input: weighted adjacency matrix A, number of stages P, number of clusters M, compression ratio μ, k-point rotation k

Set the current active columns $A_0 = A$
Set the current column indices $orig\_idx1 = 1 : G.N$
    // $orig\_idx\{p+1\}$ keeps track of the columns indices at stage p
Set the current permutation $idx1 = 1 : G.N$
    // $idx\{p+1\}$ records the permutation applied to active columns at stage p

for each stage $p = 1 : P$ **do**
    Cluster the active columns into M clusters;
    Set $perm$ = permutation that will permute A_{p-1} into correct cluster order
    Permute $A = A(perm, perm)$
    Update $orig\_idx\{p+1\} = orig\_idx\{p\}(perm)$
    Update $idx\{p+1\} = perm$

    for each cluster $m = 1 : M$ **do**
        Set $Qs\{m\}$ = FindRotsInCluster(cluster, μ, k, cInds)

    Merge Qs together to get the rotation $Q\{p\}$ for stage p
    Update $A = Q\{p\}AQ\{p\}^T$
    Set $perm$ = permutation that puts active columns together
    Permute $A = Q\{p\}AQ\{p\}^T$
    Update $orig\_idx\{p+1\} = orig\_idx\{p+1\}(perm)$
    Update $idx\{p+1\} = idx\{p+1\}(perm)$

Output: core A

7.2 Examples

Now we construct a downsampled graph at each stage, taking the core to be a weighted adjacency matrix. Note that the core has non-zero diagonals. we simply set the diagonals to zero.
Parameters:

- G = gsp_david_sensor_network
- 500 vertices
- $A = G.W$
- 4 stages
- 2 clusters
- 2-point rotations

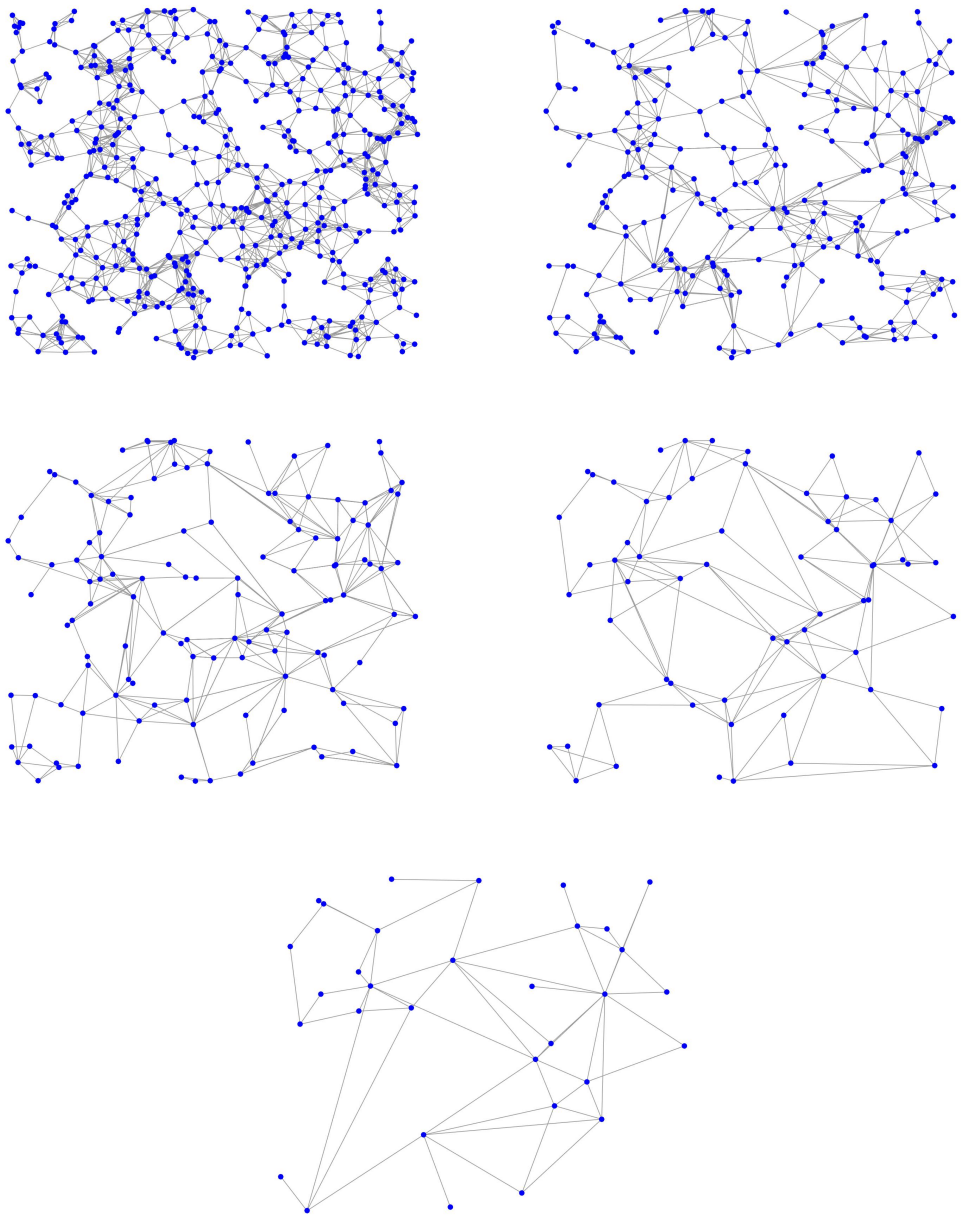

Figure 11: The top left figure is the original graph, gsp_david_sensor_network with 500 vertices, and each subsequent figure is the result of one stage of downsampling.

8 June 29

Take A as the the weighted adjacency matrix, we notice that the reduced matrix has non-zero diagonals. In this section, we zero out the diagonals at each stage. Also, we set the compression ratio $\mu = 0.2$. Smaller compression ratio seems to give better interpolation results.

First example is on the path graph. Parameters:

1. 128 vertices
2. compression ratio $\mu = 0.2$
3. 4 stages
4. 2 cluster

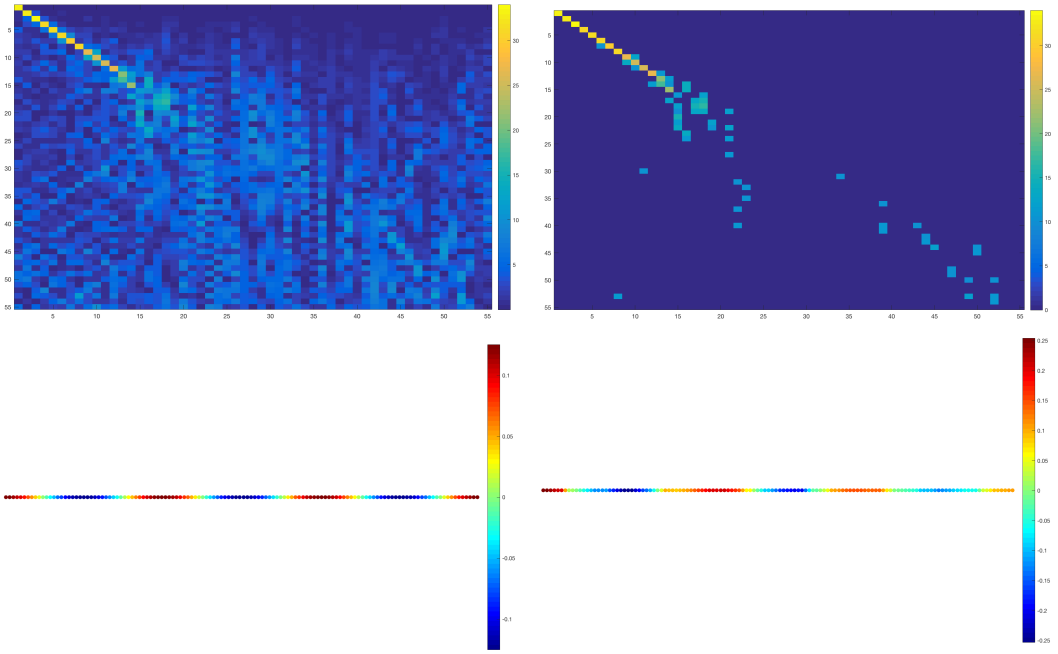

Figure 12: (a) interpolation coefficient matrix (b) threshold at $x > 10$
(c) original 7th eigenvector (d) interpolated 7th eigenvector

The following images are produced under same process except the diagonals of Abars are kept non-zero at each stage.

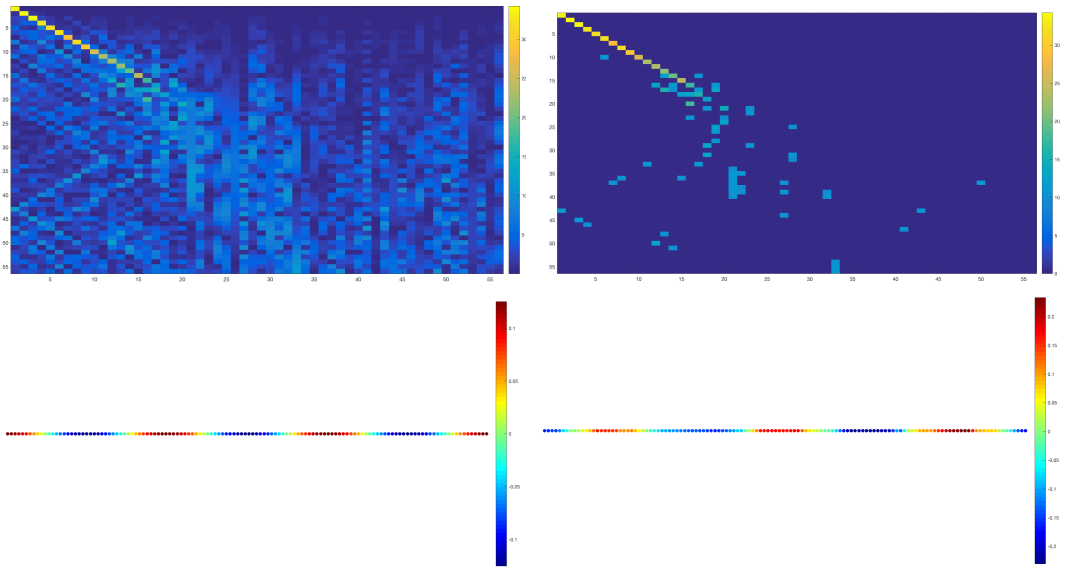

Figure 13: (a) interpolation coefficient matrix (b) threshold at $x > 10$
(c) original 7th eigenvector (d) interpolated 7th eigenvector

The second example is on david_sensor_network graph. Parameters:

1. 500 vertices
2. compression ratio $\mu = 0.2$
3. 4 stages
4. 2 cluster

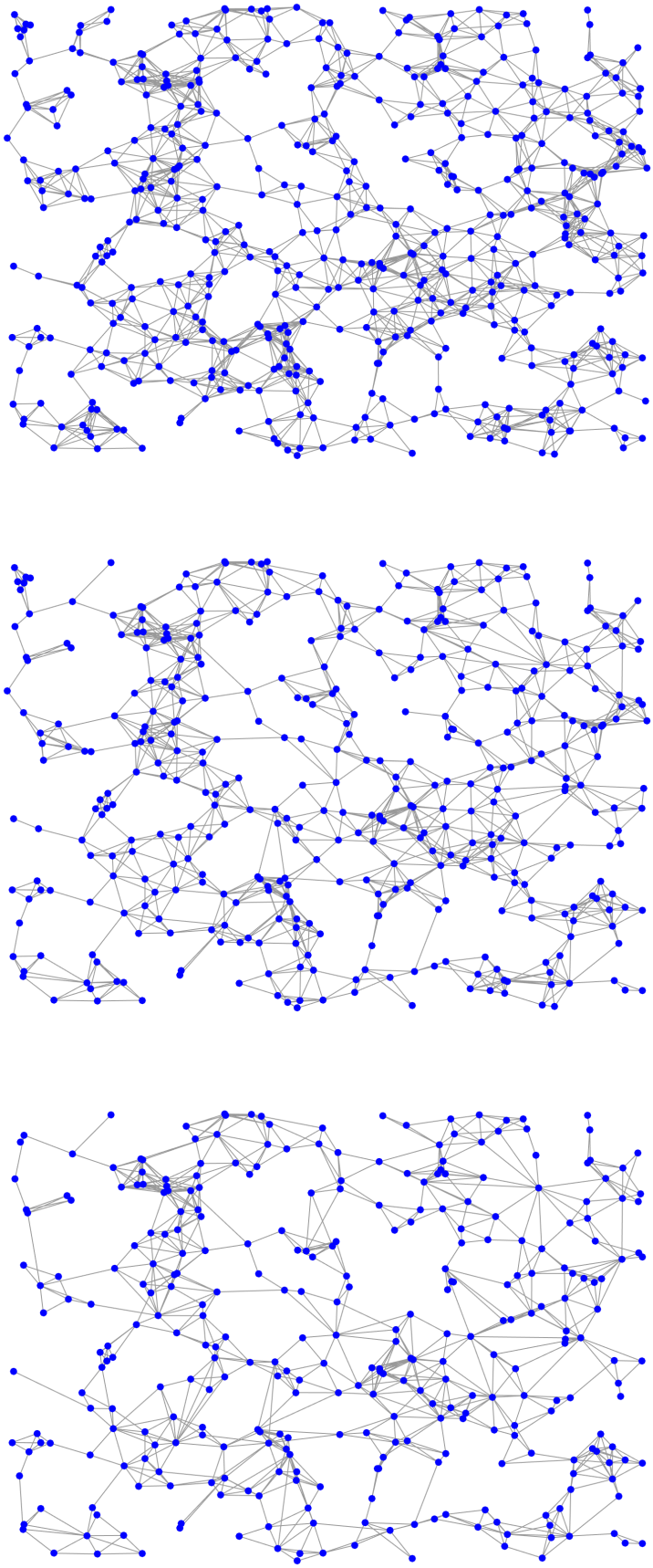

Figure 14: (a) original graph (b) 1st reduced grapg (c) 2nd reduced graph

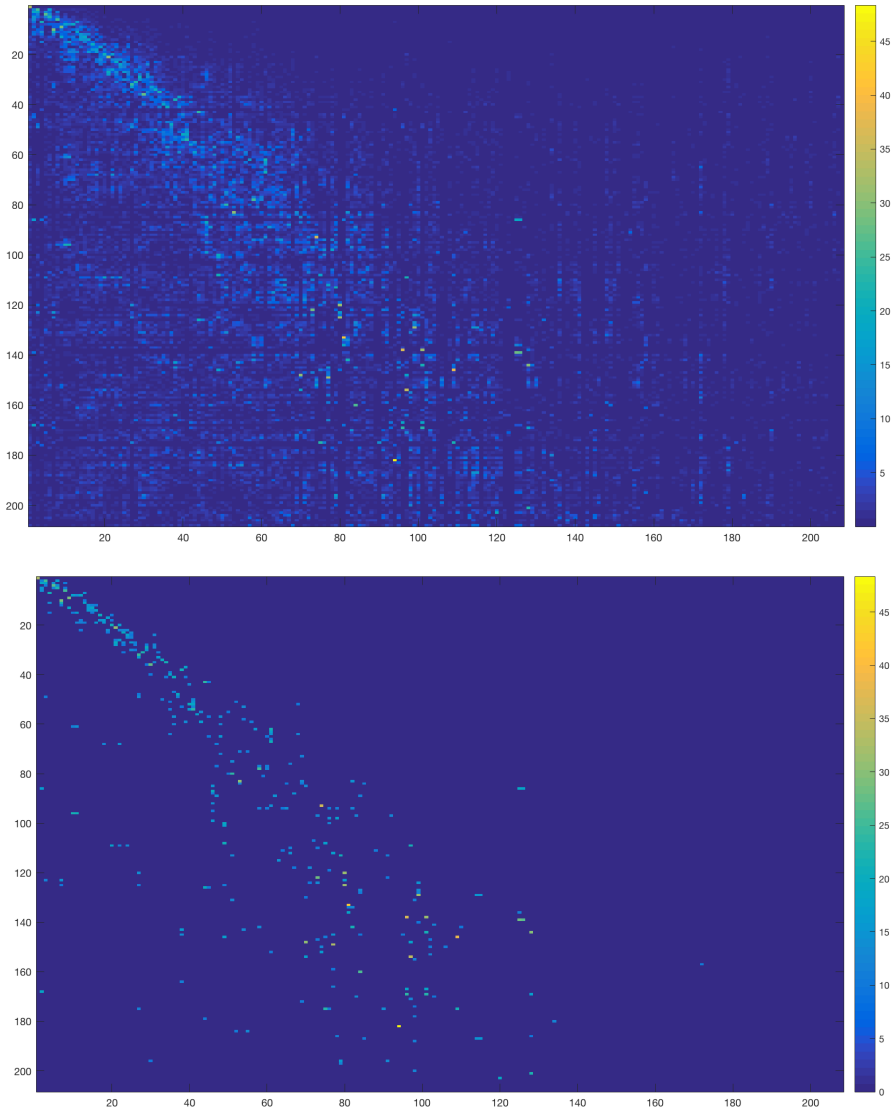

Figure 15: (a) interpolation coefficient matrix (b) threshold at $x > 10$

9 Double Procrustes Problem

Double Procrustes problem considers finding an orthogonal matrix Q, which minimizes

$$||Q^T AQ - B||$$

for symmetric A and B.

Since both A and B are symmetric, we can diagonalize these two matrices

$$A = V_1 D_1 V_2^T \text{ and } B = V_2 D_2 V_2^T.$$

Then in this case, $Q = V_1 V_2^T$. (Detailed solution can be found in "Procrustes Problems", Oxford University Press).

In the graph setting, we want to find an orthogonal Q, which minimizes $||Q^T LQ - L'||$, where L is the original graph Laplacian and L' is the reduced graph Laplacian. If we do multile levels, we could generate scaling and wavelet functions similar to pMMF. That is, the first $dim(V_L)$ rows of Q are scaling functions and the rest of its rows are wavelets.

Some general observations;

1. This method connects Kron reduction and pMMF.

2. Q is not sparse.

3. The energy of scaling functions and wavelets are not localized.

4. Kron reduction produced dense sub-graphs.

10 Downsampling G.W

The structure of the weighted adjacency matrix can be lost through downsampling – we are often left with a core that has non-zero diagonal entries. We can maintain a zero diagonal, however, by being selective when we rotate columns within a certain cluster. If columns i and j only rotate with one another, and if $[A_{n-1}]_{i,i} = 0$, then

$$[A_n]_{i,i} = [Q_n A_{n-1} Q_n^T]_{i,i} = 0$$

if vertices i and j are disconnected in the graph represented by the weighted adjacency matrix A_{n-1}:

$$\begin{aligned}
[A_n]_{i,i} &= [Q_n A_{n-1} Q_n^T]_{i,i} \\
&= [Q_n]_{i,:}[A_{n-1} Q_n^T]_{:,i} \\
&= [Q_n]_{i,:} < [A_{n-1}]_{1,:}[Q_n^T]_{:,i} + \cdots + [A_{n-1}]_{s,:}[Q_n^T]_{:,i} > \\
&= [Q_n]_{i,1}[A_{n-1}]_{1,:}[Q_n^T]_{:,i} + \cdots + [Q_n]_{i,s}[A_{n-1}]_{s,:}[Q_n^T]_{:,i}
\end{aligned}$$

Since column i only rotates with column j, the i^{th} row of Q_n will only have nonzero entries in the i^{th} and j^{th} columns. So, the above equation simplifies to

$$[A_n]_{i,i} = [Q_n]_{i,i}[A_{n-1}]_{i,:}[Q_n^T]_{:,i} + [Q_n]_{i,j}[A_{n-1}]_{j,:}[Q_n^T]_{:,i}$$

Likewise, the i^{th} column of Q_n^T will only have nonzero entries in the i^{th} and j^{th} rows. We now have

$$\begin{aligned}
[A_n]_{i,i} &= [Q_n]_{i,i}([A_{n-1}]_{i,i}[Q_n^T]_{i,i} + [A_{n-1}]_{i,j}[Q_n^T]_{j,i}) + [Q_n]_{i,j}([A_{n-1}]_{j,i}[Q_n^T]_{i,i} + [A_{n-1}]_{j,j}[Q_n^T]_{j,i}) \\
&= [Q_n]_{i,i}(0 + [A_{n-1}]_{i,j}[Q_n^T]_{j,i}) + [Q_n]_{i,j}([A_{n-1}]_{j,i}[Q_n^T]_{i,i} + 0)
\end{aligned}$$

We have $[A_{n-1}]_{i,j} = [A_{n-1}]_{j,i}$ since A is symmetric and $[Q_n]_{i,j} = [Q_n^T]_{j,i}$, so we are left with

$$[A_n]_{i,i} = 2[A_{n-1}]_{i,j}[Q_n]_{i,i}[Q_n]_{i,j}$$

Since both entries of Q_n are nonzero, $[A_n]_{i,i} =$ if and only if $[A_{n-1}]_{i,j} = 0$.
The case is complicated further when, say, column i rotates with both column j and column k. Then, Q_n has nonzero entries in the i^{th}, j^{th}, and k^{th} columns, so we would end up with

$$\begin{aligned}
[A_n]_{i,i} = [Q_n]_{i,i}([A_{n-1}]_{i,i}[Q_n^T]_{i,i} + [A_{n-1}]_{i,j}[Q_n^T]_{j,i} + [A_{n-1}]_{i,k}[Q_n^T]_{k,i}) + [Q_n]_{i,j}([A_{n-1}]_{j,i}[Q_n^T]_{i,i} + [A_{n-1}]_{j,j}[Q_n^T]_{j,i} \\
+ [A_{n-1}]_{j,k}[Q_n^T]_{k,i}) + [Q_n]_{i,k}([A_{n-1}]_{k,i}[Q_n^T]_{i,i} + [A_{n-1}]_{k,j}[Q_n^T]_{j,i} + [A_{n-1}]_{k,k}[Q_n^T]_{k,i}) \\
= 2[A_{n-1}]_{i,j}[Q_n]_{i,i}[Q_n]_{i,j} + 2[A_{n-1}]_{i,k}[Q_n]_{i,i}[Q_n]_{i,k} + 2[A_{n-1}]_{j,k}[Q_n]_{i,j}[Q_n]_{i,k}
\end{aligned}$$

We see that $[A_{n-1}]_{j,k}$ now contributes as well. So, not only must the pairs (i,j) and (i,k) be disconnected, vertices j and k must also be disconnected even if they are not one of the pairs of columns that is rotated.