Hierarchical Graph Laplacian Eigen Transforms

Jeff Irion & Naoki Saito

Department of Mathematics
University of California, Davis

Bay Area Scientific Computing Day 2013
Lawrence Berkeley National Laboratory
December 11, 2013

1. Aims & Objectives
2. Basics of Graph Laplacians
3. Hierarchical Graph Laplacian Eigen Transform (HGLET)
   - HGLET Variation: Generalized Haar Transform (GHT)
4. Best-Basis Algorithm for HGLET
5. Approximation Experiments
6. Summary and Future Work

Aims & Objectives

Wavelets

- Successful on regular domains
- Extend to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: graphs lack the notion of *frequency*!

Common strategy: develop wavelet-*like* multiscale transforms

Aims & Objectives

Wavelets

- Successful on regular domains
- Extend to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: graphs lack the notion of *frequency*!

Common strategy: develop wavelet-*like* multiscale transforms

Aims & Objectives

Wavelets

- Successful on regular domains
- Extend to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: graphs lack the notion of *frequency*!

Common strategy: develop wavelet-*like* multiscale transforms

Aims & Objectives

Wavelets

- Successful on regular domains
- Extend to irregular domains ⇒ "2nd Generation Wavelets"

For example:

- Hammond, Vandergheynst, and Gribonval (2011): wavelets via spectral graph theory
- Coifman and Maggioni (2006): diffusion wavelets
  - Bremer *et al.* (2006): diffusion wavelet packets

Key difficulty: graphs lack the notion of *frequency*!

Common strategy: develop wavelet-*like* multiscale transforms

Aims & Objectives

✓ Step 1. Develop and implement multiscale transforms for data on graphs and networks.

Step 2. Investigate usefulness for:

1. Approximation/Denoising.
   - Smoothing crime rate data
2. Classification.

Aims & Objectives

✓ Step 1. Develop and implement multiscale transforms for data on graphs and networks.

Step 2. Investigate usefulness for:

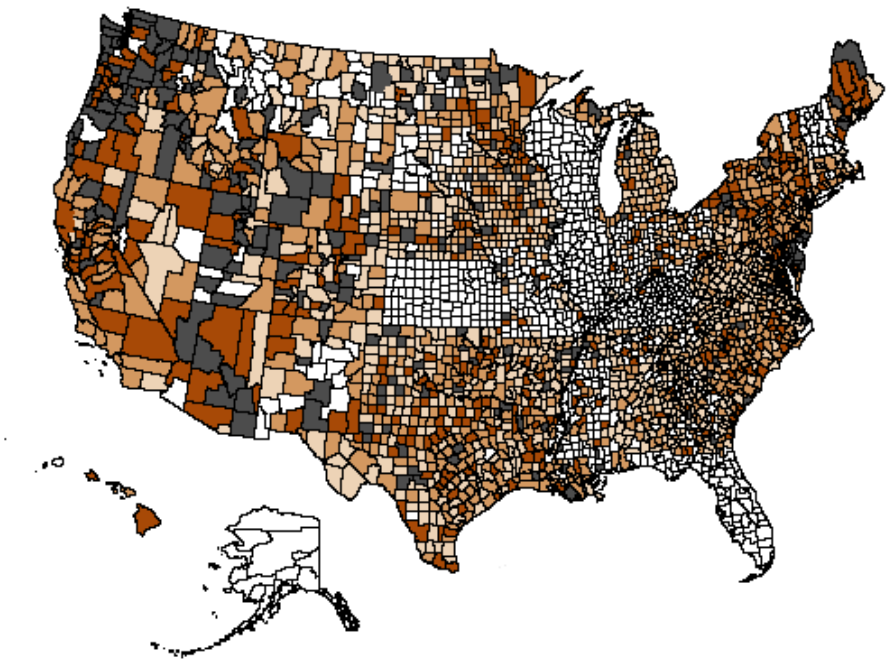
1. Approximation/Denoising.
   - Smoothing crime rate data
2. Classification.

Aims & Objectives

✓ Step 1. Develop and implement multiscale transforms for data on graphs and networks.

Step 2. Investigate usefulness for:

1. **Approximation/Denoising.**
   - Smoothing crime rate data
2. Classification.



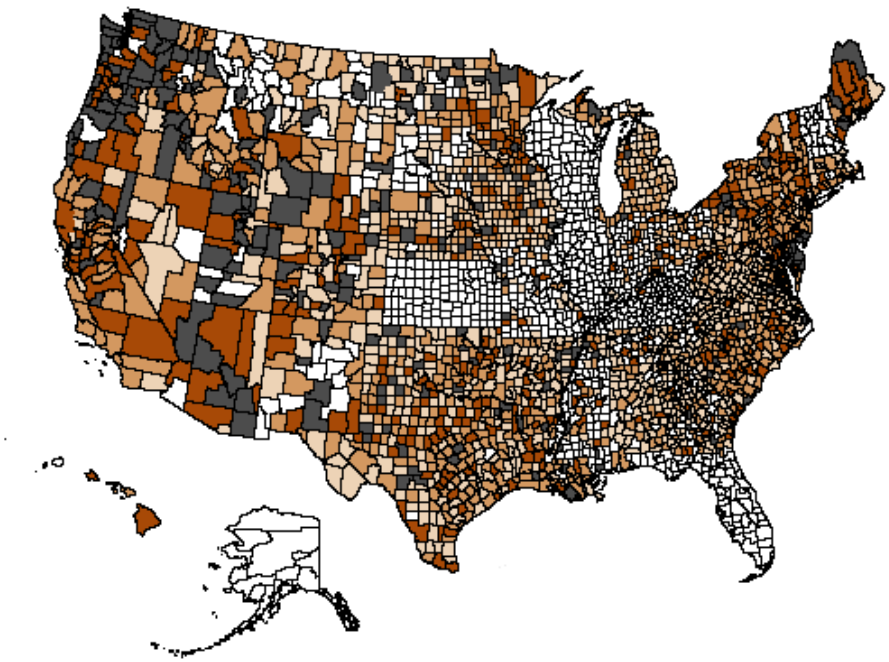
https://www.ncjrs.gov

Aims & Objectives

✓ Step 1. Develop and implement multiscale transforms for data on graphs and networks.

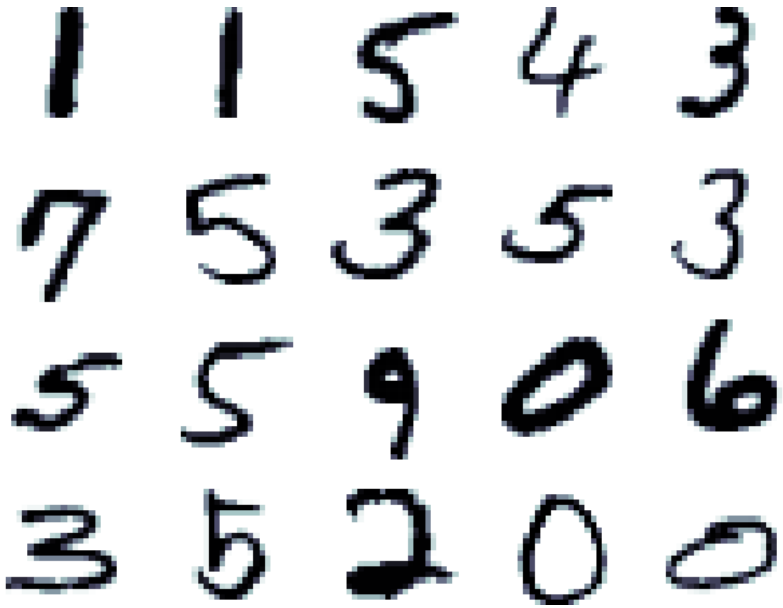
Step 2. Investigate usefulness for:

1. **Approximation/Denoising.**
   - Smoothing crime rate data



https://www.ncjrs.gov

2. **Classification.**
   - Handwritten digit (MNIST) classification

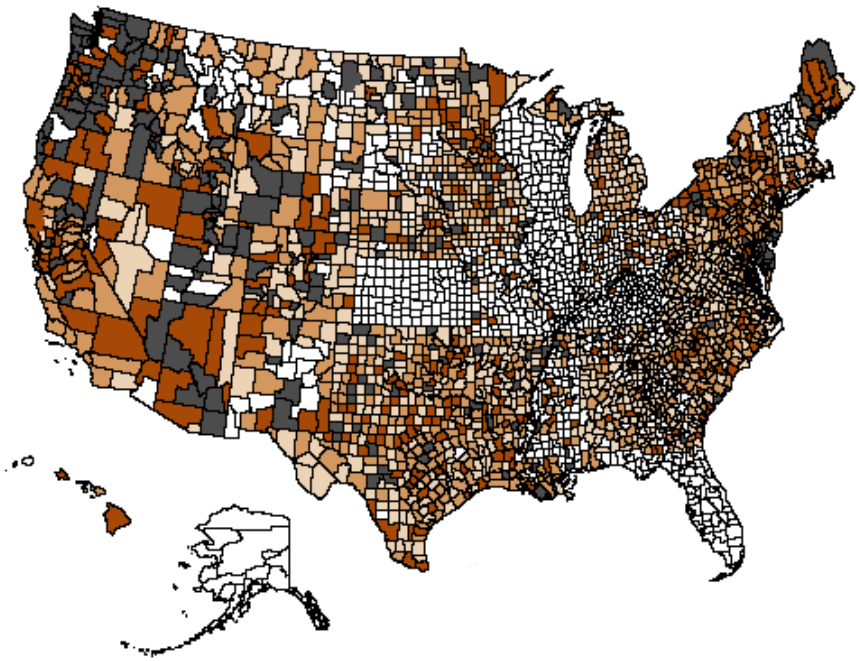


Aims & Objectives

✓ Step 1. Develop and implement multiscale transforms for data on graphs and networks.

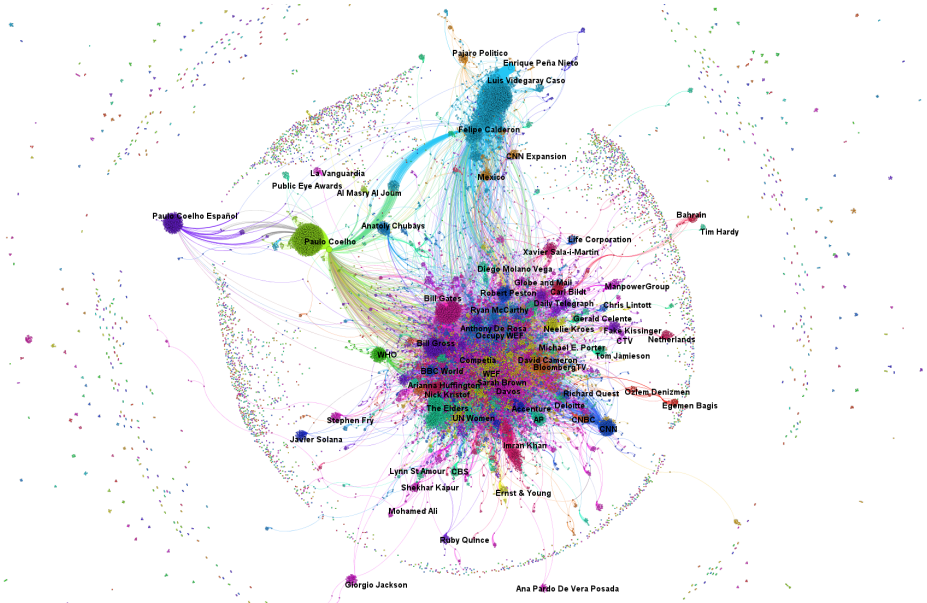
Step 2. Investigate usefulness for:

1. **Approximation/Denoising.**
   - Smoothing crime rate data



https://www.ncjrs.gov

2. **Classification.**
   - Twitter spam classification/detection



http://beautifuldata.net

Basic Definitions and Notation

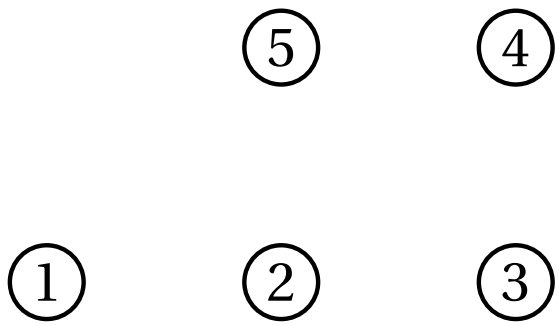
Consider a graph G.

- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the symmetric weight matrix, where w_{ij} denotes the edge weight between vertices i and j

Basic Definitions and Notation

Consider a graph G.

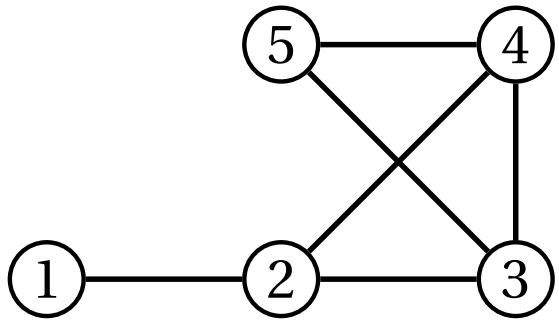
- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the symmetric weight matrix, where w_{ij} denotes the edge weight between vertices i and j



Basic Definitions and Notation

Consider a graph G.

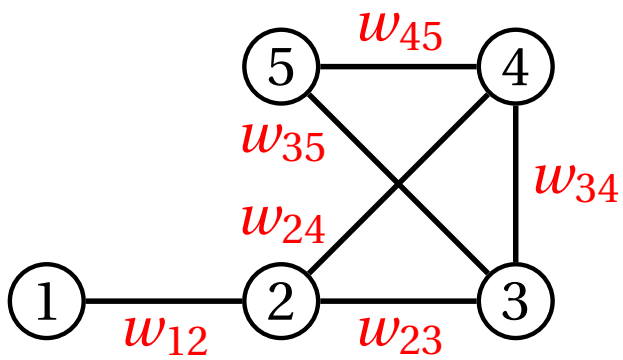
- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the symmetric weight matrix, where w_{ij} denotes the edge weight between vertices i and j



Basic Definitions and Notation

Consider a graph G.

- $V = V(G) = \{v_1, \ldots, v_N\}$ is the set of vertices
- $E = E(G) = \{e_1, \ldots, e_{N'}\}$ is the set of edges, where $e_k = (v_i, v_j)$ represents an edge (or line segment) connecting between adjacent vertices v_i, v_j for some $1 \leq i, j \leq N$
- $W = W(G) \in \mathbb{R}^{N \times N}$ is the symmetric weight matrix, where w_{ij} denotes the edge weight between vertices i and j



Basic Definitions and Notation

Note that there are many ways to define w_{ij}.

For example, for *unweighted* graphs, we typically use

$$w_{ij} := \begin{cases} 1 & \text{if } v_i \sim v_j \text{ (i.e., } v_i \text{ and } v_j \text{ are adjacent);} \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as the adjacency matrix and denoted by $A(G)$.

For *weighted* graphs, w_{ij} should reflect the similarity (or affinity) of information at v_i and v_j, e.g., if $v_i \sim v_j$, then

$$w_{ij} := 1/\mathrm{dist}(v_i, v_j) \quad \text{or} \quad \exp(-\mathrm{dist}(v_i, v_j)^2/\epsilon^2),$$

where $\mathrm{dist}(\cdot,\cdot)$ is a certain measure of dissimilarity and $\epsilon > 0$ is an appropriate scale parameter.

Basic Definitions and Notation

Note that there are many ways to define w_{ij}.

For example, for *unweighted* graphs, we typically use

$$w_{ij} := \begin{cases} 1 & \text{if } v_i \sim v_j \text{ (i.e., } v_i \text{ and } v_j \text{ are adjacent);} \\ 0 & \text{otherwise.} \end{cases}$$

This is often referred to as the adjacency matrix and denoted by $A(G)$.

For *weighted* graphs, w_{ij} should reflect the similarity (or affinity) of information at v_i and v_j, e.g., if $v_i \sim v_j$, then

$$w_{ij} := 1/\mathrm{dist}(v_i, v_j) \quad \text{or} \quad \exp(-\mathrm{dist}(v_i, v_j)^2/\epsilon^2),$$

where $\mathrm{dist}(\cdot,\cdot)$ is a certain measure of dissimilarity and $\epsilon > 0$ is an appropriate scale parameter.

Our Assumptions

We assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is symmetric.

The graph may be weighted or unweighted.

Our Assumptions

We assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is symmetric.

The graph may be weighted or unweighted.

Our Assumptions

We assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is symmetric.

The graph may be weighted or unweighted.

Our Assumptions

We assume that the graph is

- **connected.** Otherwise, we would simply consider the components separately.
- **undirected.** Edges do not have direction, which means that $w_{ij} = w_{ji}$ and thus W is symmetric.

The graph may be weighted or unweighted.

Graph Laplacians

$$\begin{cases} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \operatorname{diag}(d_{v_1}, \ldots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^{N} w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{cases}$$

We also define the random walk Laplacian and symmetric normalized Laplacian matrices:

$$L_{\text{rw}}(G) := D^{-1}L = I - D^{-1}W$$
$$L_{\text{sym}}(G) := D^{-1/2}LD^{-1/2}$$

Graph Laplacians

$$\begin{cases} W(G) = (w_{ij}) & \text{the weight matrix} \\ D(G) := \operatorname{diag}(d_{v_1}, \ldots, d_{v_n}) & \text{the degree matrix, where } d_{v_i} := \sum_{j=1}^{N} w_{ij}. \\ L(G) := D(G) - W(G) & \text{the Laplacian matrix} \end{cases}$$

We also define the random walk Laplacian and symmetric normalized Laplacian matrices:

$$L_{\text{rw}}(G) := D^{-1}L = I - D^{-1}W$$
$$L_{\text{sym}}(G) := D^{-1/2}LD^{-1/2}$$

Graph Laplacian Eigenvalues & Eigenvectors

$$L := D - W$$
$$L_{\text{rw}} := D^{-1}L = I - D^{-1}W$$
$$L_{\text{sym}} := D^{-1/2}LD^{-1/2}$$

We have ($\diamondsuit$ denotes that it could refer to L, L_{rw}, or L_{sym}):

- sorted eigenvalues $0 = \lambda_0^{\diamondsuit} < \lambda_1^{\diamondsuit} \leq \cdots \leq \lambda_{N-1}^{\diamondsuit}$
- associated eigenvectors $\boldsymbol{\phi}_0^{\diamondsuit}$, $\boldsymbol{\phi}_1^{\diamondsuit}$, $\boldsymbol{\phi}_{N-1}^{\diamondsuit}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L and L_{sym} are symmetric, their eigenvectors form orthonormal bases
- $\boldsymbol{\phi}_0 = \boldsymbol{\phi}_0^{\text{rw}} = \mathbf{1}/\sqrt{N}$
- $\lambda_k^{\text{rw}} = \lambda_k^{\text{sym}}$ and $D^{1/2}\boldsymbol{\phi}_k^{\text{rw}} = \boldsymbol{\phi}_k^{\text{sym}}$

Graph Laplacian Eigenvalues & Eigenvectors

$$L := D - W$$
$$L_{\text{rw}} := D^{-1}L = I - D^{-1}W$$
$$L_{\text{sym}} := D^{-1/2}LD^{-1/2}$$

We have ($\lozenge$ denotes that it could refer to L, L_{rw}, or L_{sym}):

- sorted eigenvalues $0 = \lambda_0^{\lozenge} < \lambda_1^{\lozenge} \leq \cdots \leq \lambda_{N-1}^{\lozenge}$
- associated eigenvectors $\boldsymbol{\phi}_0^{\lozenge}$, $\boldsymbol{\phi}_1^{\lozenge}$, $\boldsymbol{\phi}_{N-1}^{\lozenge}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L and L_{sym} are symmetric, their eigenvectors form orthonormal bases
- $\boldsymbol{\phi}_0 = \boldsymbol{\phi}_0^{\text{rw}} = \mathbf{1}/\sqrt{N}$
- $\lambda_k^{\text{rw}} = \lambda_k^{\text{sym}}$ and $D^{1/2}\boldsymbol{\phi}_k^{\text{rw}} = \boldsymbol{\phi}_k^{\text{sym}}$

Graph Laplacian Eigenvalues & Eigenvectors

$$L := D - W$$
$$L_{\text{rw}} := D^{-1}L = I - D^{-1}W$$
$$L_{\text{sym}} := D^{-1/2}LD^{-1/2}$$

We have ($\lozenge$ denotes that it could refer to L, L_{rw}, or L_{sym}):

- sorted eigenvalues $0 = \lambda_0^{\lozenge} < \lambda_1^{\lozenge} \leq \cdots \leq \lambda_{N-1}^{\lozenge}$
- associated eigenvectors $\boldsymbol{\phi}_0^{\lozenge}$, $\boldsymbol{\phi}_1^{\lozenge}$, $\boldsymbol{\phi}_{N-1}^{\lozenge}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L and L_{sym} are symmetric, their eigenvectors form orthonormal bases
- $\boldsymbol{\phi}_0 = \boldsymbol{\phi}_0^{\text{rw}} = \mathbf{1}/\sqrt{N}$
- $\lambda_k^{\text{rw}} = \lambda_k^{\text{sym}}$ and $D^{1/2}\boldsymbol{\phi}_k^{\text{rw}} = \boldsymbol{\phi}_k^{\text{sym}}$

Graph Laplacian Eigenvalues & Eigenvectors

$$L := D - W$$
$$L_{\text{rw}} := D^{-1}L = I - D^{-1}W$$
$$L_{\text{sym}} := D^{-1/2}LD^{-1/2}$$

We have ($\diamondsuit$ denotes that it could refer to L, L_{rw}, or L_{sym}):

- sorted eigenvalues $0 = \lambda_0^{\diamondsuit} < \lambda_1^{\diamondsuit} \leq \cdots \leq \lambda_{N-1}^{\diamondsuit}$
- associated eigenvectors $\boldsymbol{\phi}_0^{\diamondsuit}$, $\boldsymbol{\phi}_1^{\diamondsuit}$, $\boldsymbol{\phi}_{N-1}^{\diamondsuit}$

The eigenvectors form a basis for $\mathbb{R}^N$. In particular:

- since L and L_{sym} are symmetric, their eigenvectors form orthonormal bases
- $\boldsymbol{\phi}_0 = \boldsymbol{\phi}_0^{\text{rw}} = \mathbf{1}/\sqrt{N}$
- $\lambda_k^{\text{rw}} = \lambda_k^{\text{sym}}$ and $D^{1/2}\boldsymbol{\phi}_k^{\text{rw}} = \boldsymbol{\phi}_k^{\text{sym}}$

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i} f(v_i) - \sum_{j \neq i} w_{ij} f(v_j).$$

This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines* (*cosines*) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.
  - Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian corresponds to *Fourier (or spectral) analysis of the data on that graph.*

Why Graph Laplacians?

- Let $f \in \mathbb{R}^N$. Then

$$Lf(v_i) = d_{v_i} f(v_i) - \sum_{j \neq i} w_{ij} f(v_j).$$

  This is a generalization of *the finite difference approximation to the Laplace operator*.

- After all, *sines* (*cosines*) are the eigenfunctions of the Laplacian on the rectangular domain with Dirichlet (Neumann) boundary conditions.
  - Hence, the expansion of data measured at the vertices using the eigenvectors of a graph Laplacian corresponds to *Fourier (or spectral) analysis of the data on that graph*.

A Simple Yet Important Example: A Path Graph

$$\underbrace{\begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 1 \end{bmatrix}}_{L(G)} = \underbrace{\begin{bmatrix} 1 & & & & & & \\ & 2 & & & & & \\ & & 2 & & & & \\ & & & \ddots & & & \\ & & & & 2 & & \\ & & & & & 1 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & & & & & \\ 1 & 0 & 1 & & & & \\ & 1 & 0 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & 1 & \\ & & & & 1 & 0 \end{bmatrix}}_{A(G)}$$

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors used for the JPEG image compression standard! (See e.g., Strang, SIAM Review, 1999).

- $\lambda_k = 2 - 2\cos(\pi k/N) = 4\sin^2(\pi k/2N),\ k = 0, 1, \ldots, N-1.$
- $\boldsymbol{\phi}_k(\ell) = \sqrt{2/N}\cos\left(\pi k(\ell + \frac{1}{2})/N\right),\ k, \ell = 0, 1, \ldots, N-1.$
  - λ (eigenvalue) is a monotonic function w.r.t. k (frequency). However, for general graphs, λ does not have a simple relationship with k.

A Simple Yet Important Example: A Path Graph

$$\underbrace{\begin{bmatrix} 1 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 1 \end{bmatrix}}_{L(G)} = \underbrace{\begin{bmatrix} 1 & & & & & & \\ & 2 & & & & & \\ & & 2 & & & & \\ & & & \ddots & & & \\ & & & & 2 & & \\ & & & & & 1 \end{bmatrix}}_{D(G)} - \underbrace{\begin{bmatrix} 0 & 1 & & & & & \\ 1 & 0 & 1 & & & & \\ & 1 & 0 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & 1 & \\ & & & & 1 & 0 \end{bmatrix}}_{A(G)}$$

The eigenvectors of this matrix are exactly the *DCT Type II* basis vectors used for the JPEG image compression standard! (See e.g., Strang, SIAM Review, 1999).

- $\lambda_k = 2 - 2\cos(\pi k/N) = 4\sin^2(\pi k/2N),\ k = 0, 1, \ldots, N-1.$
- $\boldsymbol{\phi}_k(\ell) = \sqrt{2/N}\cos\left(\pi k(\ell + \frac{1}{2})/N\right),\ k, \ell = 0, 1, \ldots, N-1.$
  - λ (eigenvalue) is a monotonic function w.r.t. k (frequency). However, for general graphs, λ does not have a simple relationship with k.

Now we turn our focus to a novel transform that can be viewed as a generalization of the block Discrete Cosine Transform. We refer to this transform as the Hierarchical Graph Laplacian Eigen Transform (HGLET).

In order to utilize a hierarchical scheme, we will need to partition the graph. Therefore, we will now review some information about graph partitioning.

Now we turn our focus to a novel transform that can be viewed as a generalization of the block Discrete Cosine Transform. We refer to this transform as the Hierarchical Graph Laplacian Eigen Transform (HGLET).

In order to utilize a hierarchical scheme, we will need to partition the graph. Therefore, we will now review some information about graph partitioning.

Graph Partitioning via Spectral Clustering

Goal: split the vertices V into two "good" subsets, X and X^c

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1]:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

Graph Partitioning via Spectral Clustering

Goal: split the vertices V into two "good" subsets, X and X^c

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1]:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

Graph Partitioning via Spectral Clustering

Goal: split the vertices V into two "good" subsets, X and X^c

Plan: use the signs of the entries in $\boldsymbol{\phi}_1$, which is known as the Fiedler vector

Why? Using $\boldsymbol{\phi}_1$ to generate X and X^c yields an approximate minimizer of the RatioCut function[1]:

$$\text{RatioCut}(X, X^c) := \frac{\text{cut}(X, X^c)}{|X|} + \frac{\text{cut}(X, X^c)}{|X^c|},$$

where

$$\text{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

[1] L. Hagen and A. B. Kahng: "New spectral methods for ratio cut partitioning and clustering," *IEEE Trans. Comput.-Aided Des.*, vol. 11, no. 9, pp. 1074-1085, 1992.

Graph Partitioning via Spectral Clustering

Instead of RatioCut, we could consider the Normalized Cut function[1]:

$$\mathrm{NCut}(X, X^c) := \frac{\mathrm{cut}(X, X^c)}{\mathrm{vol}(X)} + \frac{\mathrm{cut}(X, X^c)}{\mathrm{vol}(X^c)},$$

where

$$\mathrm{cut}(X, X^c) := \sum_{\substack{v_i \in X \\ v_j \in X^c}} W_{ij}$$

$$\mathrm{vol}(X) := \sum_{v_i \in X} d_{v_i}$$

Like before, an approximate solution is given by using the sign of $\boldsymbol{\phi}_1^{\mathrm{rw}}$ (or equivalently, $\boldsymbol{\phi}_1^{\mathrm{sym}} = D^{1/2}\boldsymbol{\phi}_1^{\mathrm{rw}}$).

[1] J. Shi & J. Malik: "Normalized cuts and image segmentation", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

Example of Graph Partitioning

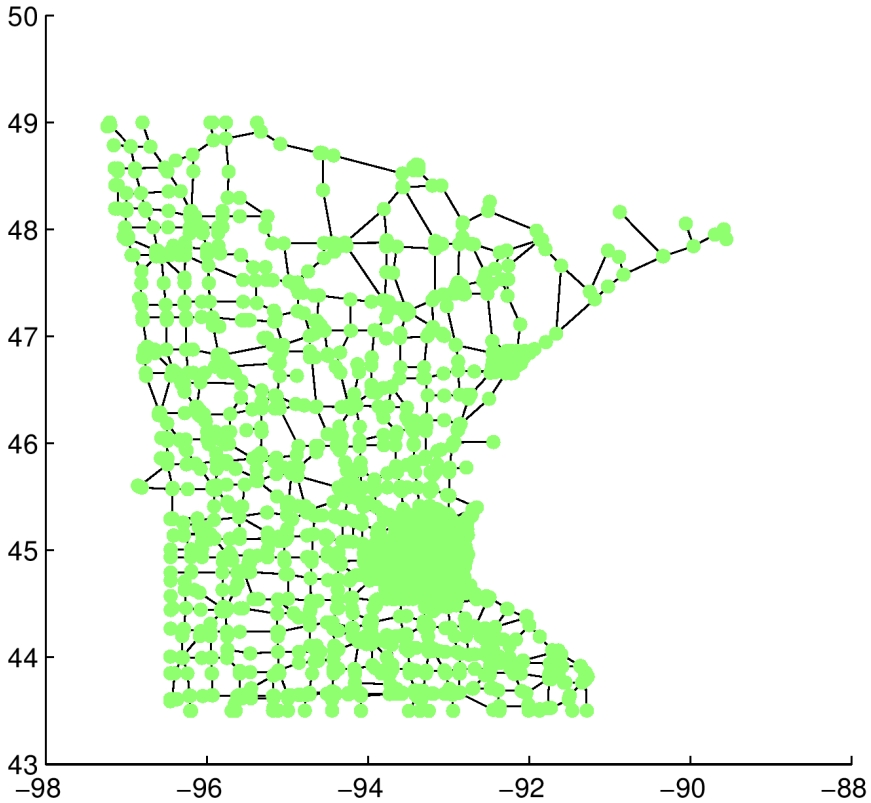


Figure: The MN road network

Example of Graph Partitioning

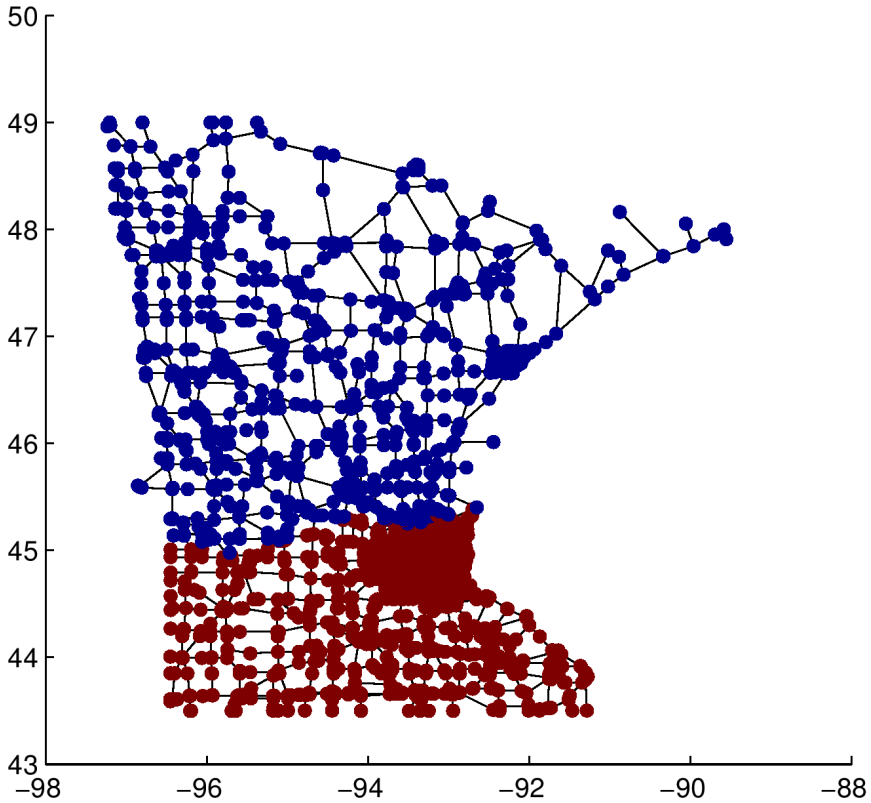


Figure: The MN road network partitioned via the Fiedler vector of L_{rw}

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph $\Rightarrow$ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions $\Rightarrow$ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array} \right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array} \right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array} \right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array} \right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array} \right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array} \right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array} \right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array} \right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array} \right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array} \right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array} \right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\,\boldsymbol{\phi}_{0,1}^{2}\cdots\boldsymbol{\phi}_{0,N_0-1}^{2}\right]\left[\boldsymbol{\phi}_{1,0}^{2}\,\boldsymbol{\phi}_{1,1}^{2}\cdots\boldsymbol{\phi}_{1,N_1-1}^{2}\right]\left[\boldsymbol{\phi}_{2,0}^{2}\,\boldsymbol{\phi}_{2,1}^{2}\cdots\boldsymbol{\phi}_{2,N_2-1}^{2}\right]\left[\boldsymbol{\phi}_{3,0}^{2}\,\boldsymbol{\phi}_{3,1}^{2}\cdots\boldsymbol{\phi}_{3,N_3-1}^{2}\right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\, \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,N_0-1}^{2}\right] \left[\boldsymbol{\phi}_{1,0}^{2}\, \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,N_1-1}^{2}\right] \left[\boldsymbol{\phi}_{2,0}^{2}\, \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,N_2-1}^{2}\right] \left[\boldsymbol{\phi}_{3,0}^{2}\, \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,N_3-1}^{2}\right]$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\, \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,N_0-1}^{2}\right] \left[\boldsymbol{\phi}_{1,0}^{2}\, \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,N_1-1}^{2}\right] \left[\boldsymbol{\phi}_{2,0}^{2}\, \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,N_2-1}^{2}\right] \left[\boldsymbol{\phi}_{3,0}^{2}\, \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,N_3-1}^{2}\right]$$

$$\vdots$$

And now, we present our Hierarchical Graph Laplacian Eigen Transform:

1. Generate an orthonormal basis for the entire graph ⇒ Laplacian eigenvectors (Notation is $\boldsymbol{\phi}_{k,l}^{j}$ with $j=0$)
2. Partition the graph using the Fiedler vector $\boldsymbol{\phi}_{k,1}^{j}$
3. Generate an orthonormal basis for each of the partitions ⇒ Laplacian eigenvectors
4. Repeat...
5. Select an orthonormal basis from this collection of orthonormal bases

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{0} & \boldsymbol{\phi}_{0,1}^{0} & \boldsymbol{\phi}_{0,2}^{0} & \cdots & \boldsymbol{\phi}_{0,N-1}^{0} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \boldsymbol{\phi}_{0,0}^{1} & \boldsymbol{\phi}_{0,1}^{1} & \boldsymbol{\phi}_{0,2}^{1} & \cdots & \boldsymbol{\phi}_{0,N_0-1}^{1} \end{array}\right] \quad \left[\begin{array}{ccccc} \boldsymbol{\phi}_{1,0}^{1} & \boldsymbol{\phi}_{1,1}^{1} & \boldsymbol{\phi}_{1,2}^{1} & \cdots & \boldsymbol{\phi}_{1,N_1-1}^{1} \end{array}\right]$$

$$\left[\boldsymbol{\phi}_{0,0}^{2}\, \boldsymbol{\phi}_{0,1}^{2} \cdots \boldsymbol{\phi}_{0,N_0-1}^{2}\right] \left[\boldsymbol{\phi}_{1,0}^{2}\, \boldsymbol{\phi}_{1,1}^{2} \cdots \boldsymbol{\phi}_{1,N_1-1}^{2}\right] \left[\boldsymbol{\phi}_{2,0}^{2}\, \boldsymbol{\phi}_{2,1}^{2} \cdots \boldsymbol{\phi}_{2,N_2-1}^{2}\right] \left[\boldsymbol{\phi}_{3,0}^{2}\, \boldsymbol{\phi}_{3,1}^{2} \cdots \boldsymbol{\phi}_{3,N_3-1}^{2}\right]$$

$$\vdots$$

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{array} \right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{array} \right]$$

$$\left[\begin{array}{ccc} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{array} \right]$$

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{array} \right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{array} \right]$$

$$\left[\begin{array}{ccc} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{array} \right]$$

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{array} \right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{array} \right]$$

$$\left[\begin{array}{ccc} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{array} \right]$$

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{array} \right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{array} \right]$$

$$\left[\begin{array}{ccc} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{array} \right]$$

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{array} \right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{array} \right]$$

$$\left[\begin{array}{ccc} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{array} \right]$$

Observations

- For an unweighted path graph, this yields a dictionary of the block DCT-II
- Similar to wavelet packet or local cosine dictionaries in that it generates an *overcomplete basis* from which we can select a basis useful for the task at hand ⇒ best-basis algorithm, local discriminant basis algorithm, ...
  - A union of bases on disjoint subsets is obviously orthonormal

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{array} \right]$$

$$\left[\begin{array}{cccc} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{array} \right] \left[\begin{array}{cccc} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{array} \right]$$

$$\left[\begin{array}{ccc} \boldsymbol{\phi}^2_{0,0} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{1,0} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{2,0} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{array} \right] \left[\begin{array}{ccc} \boldsymbol{\phi}^2_{3,0} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{array} \right]$$

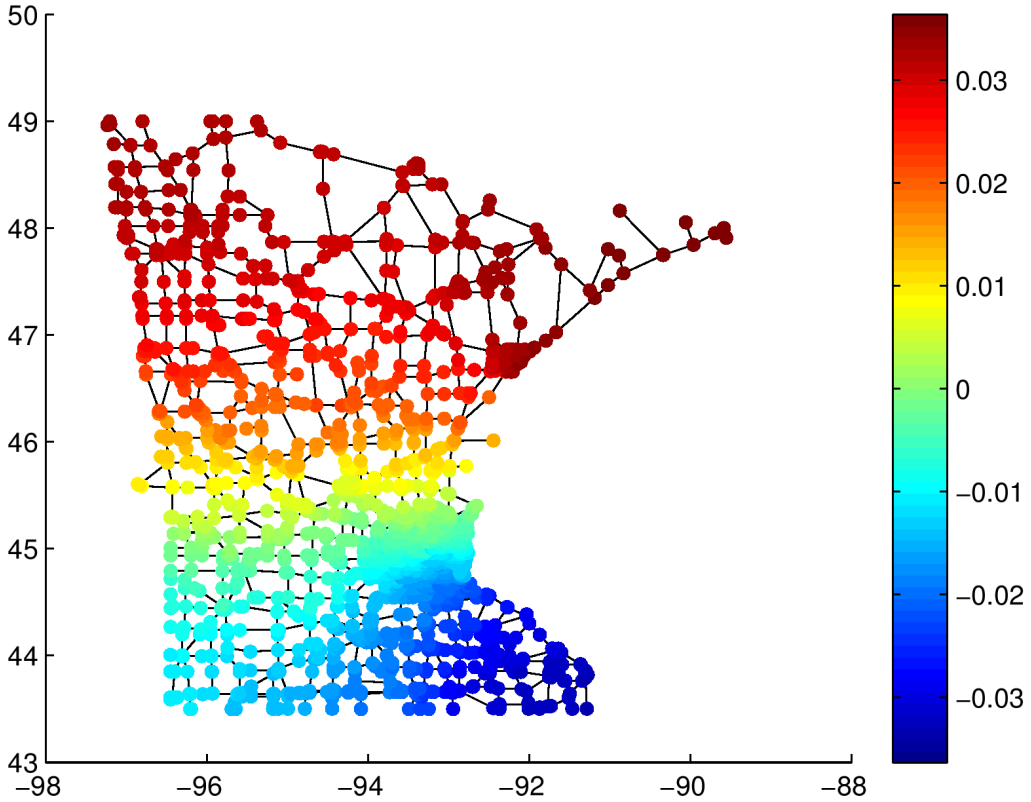
HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

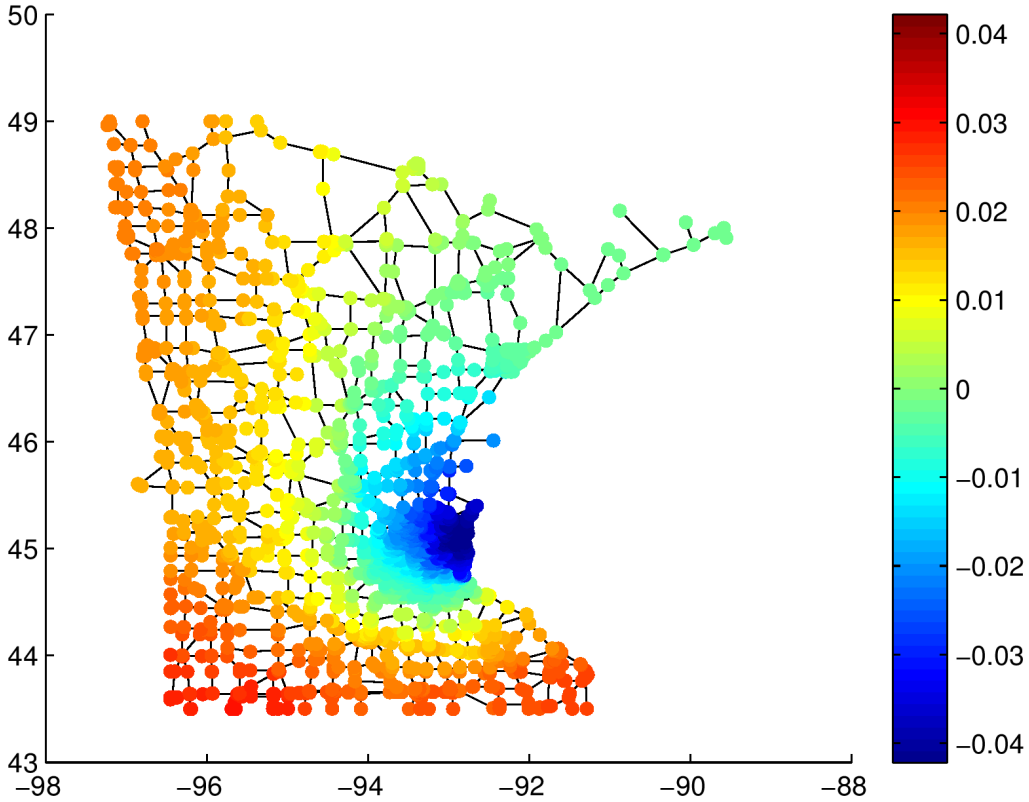
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{0}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

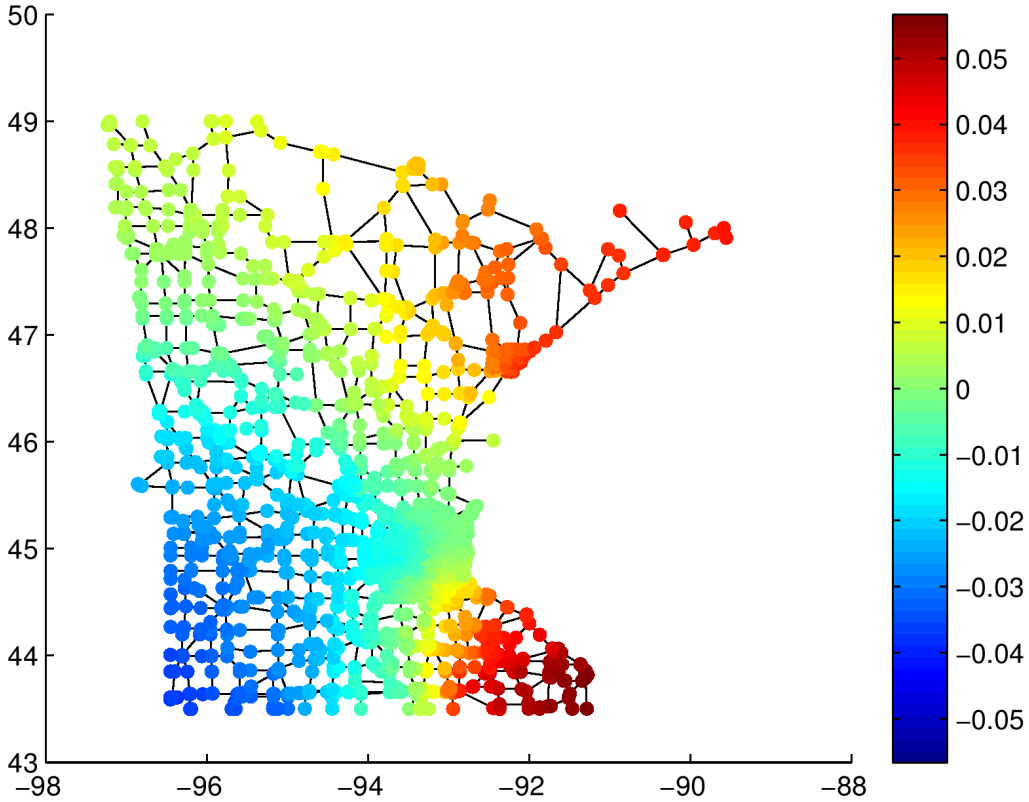
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^{0}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

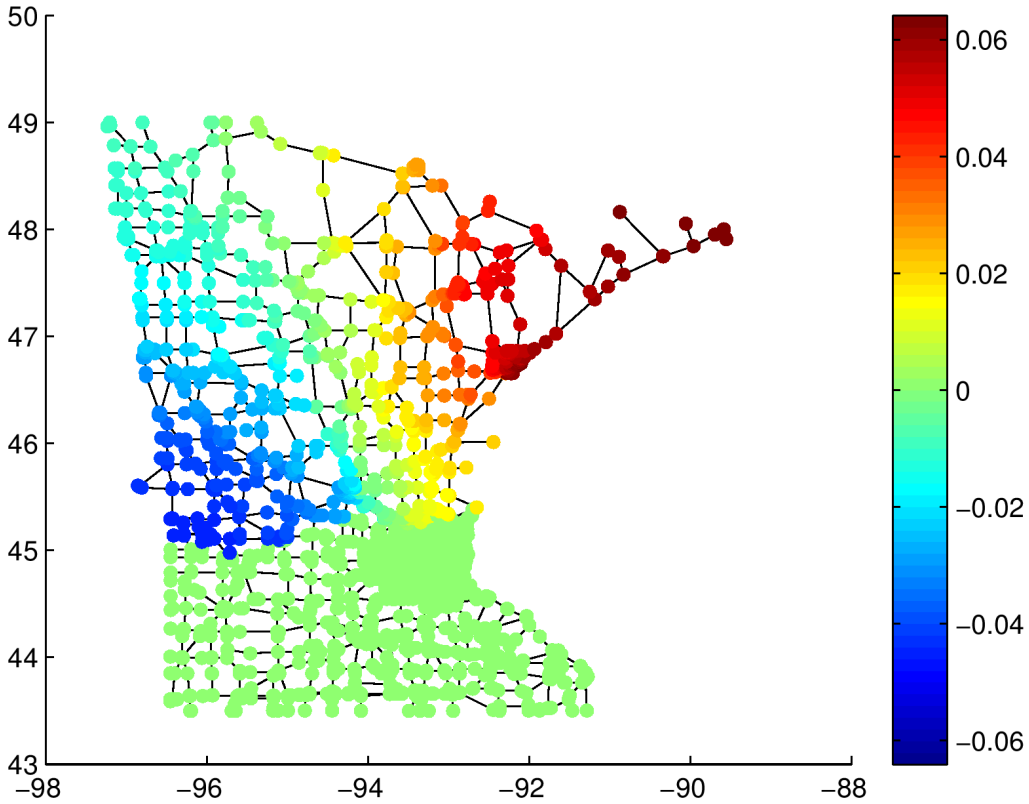
Level $j = 0$, Region $k = 0$, $\boldsymbol{\phi}_{0,3}^{0}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

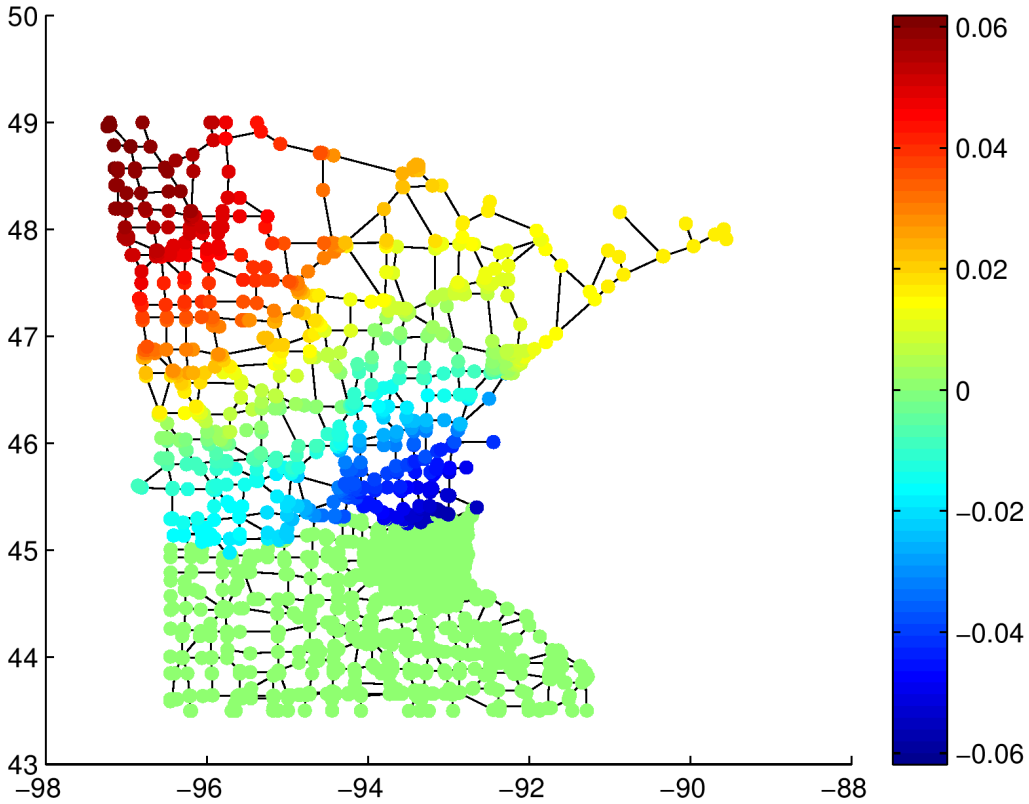
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{1}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

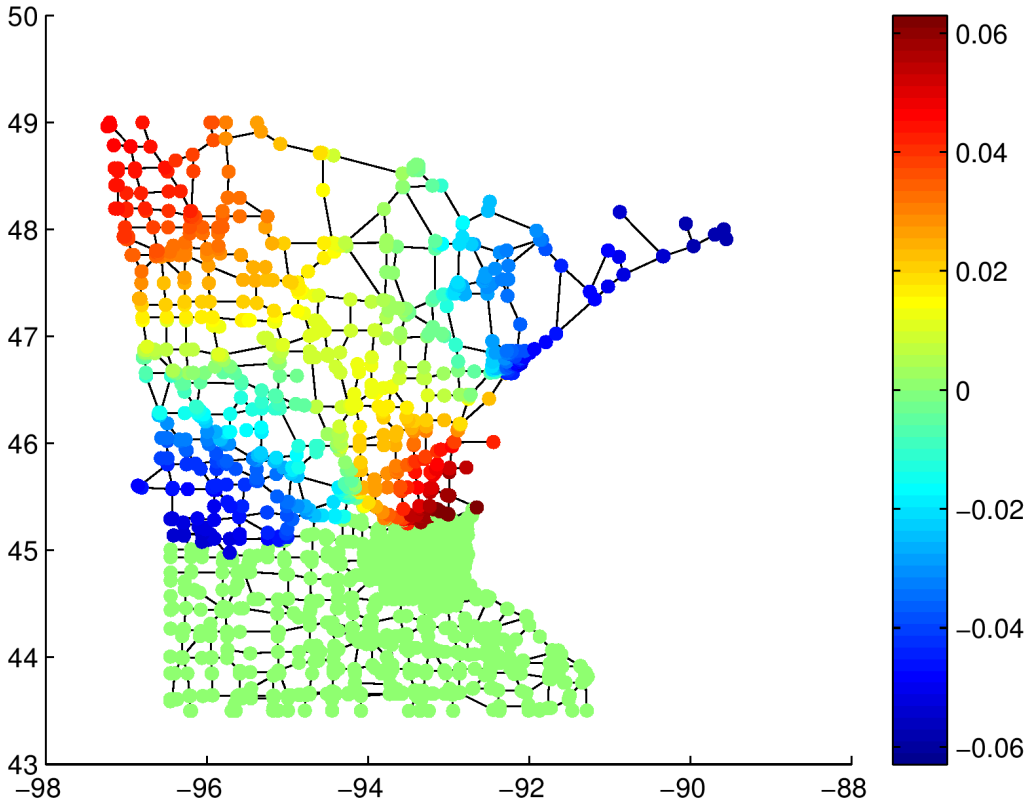
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^{1}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

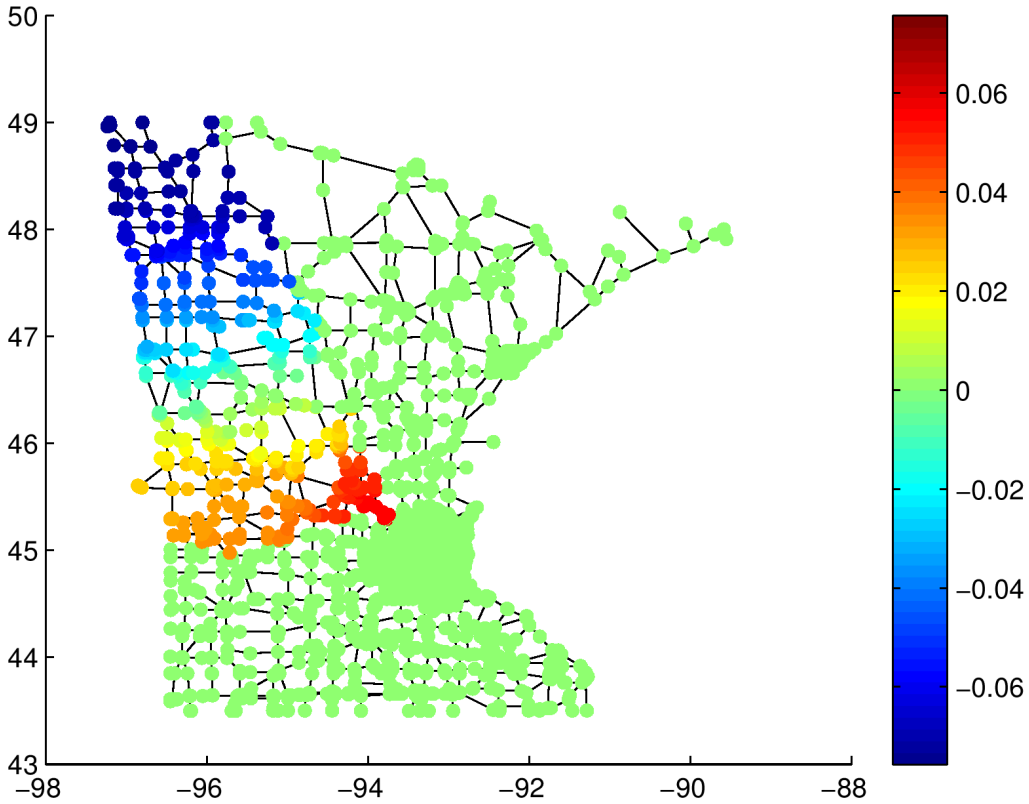
Level $j = 1$, Region $k = 0$, $\boldsymbol{\phi}_{0,3}^{1}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

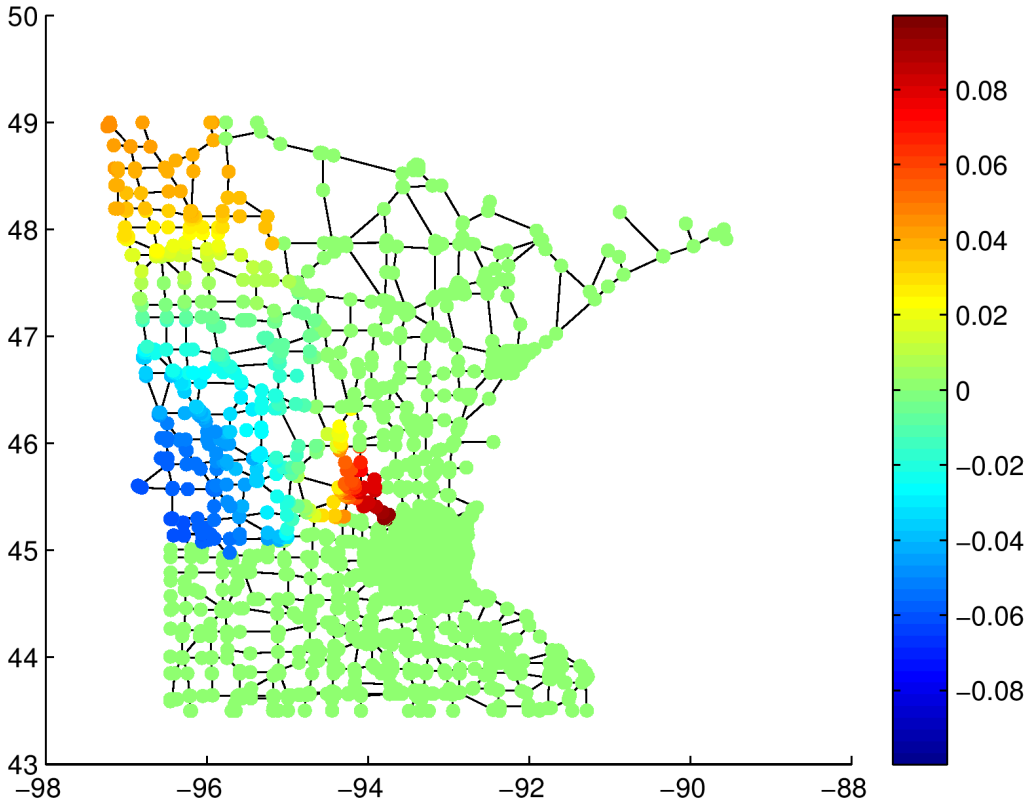
Level $j = 2$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{2}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

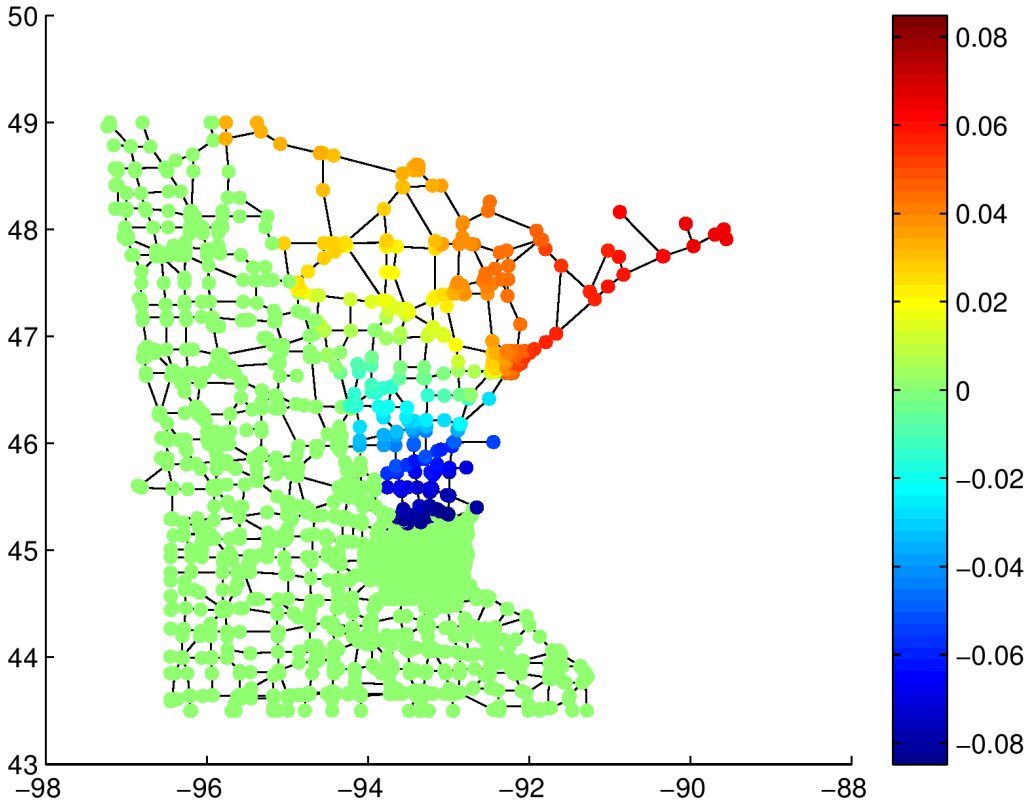
Level $j = 2$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^2$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

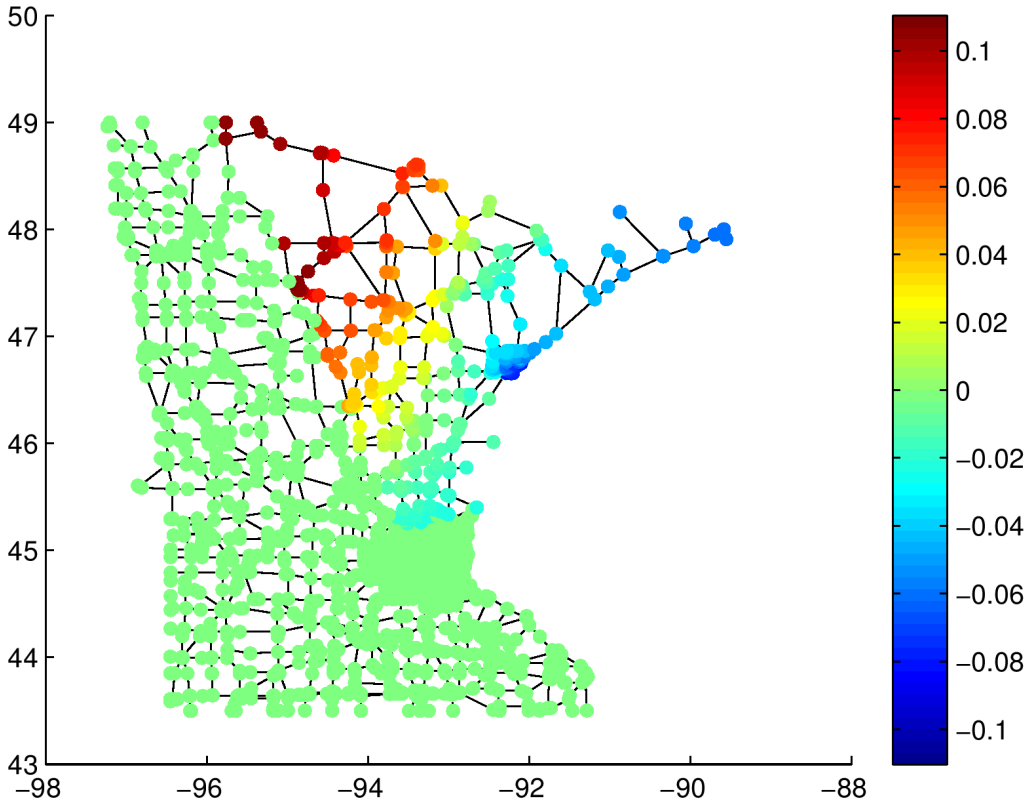
Level $j = 2$, Region $k = 1$, $\boldsymbol{\phi}_{1,1}^{2}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

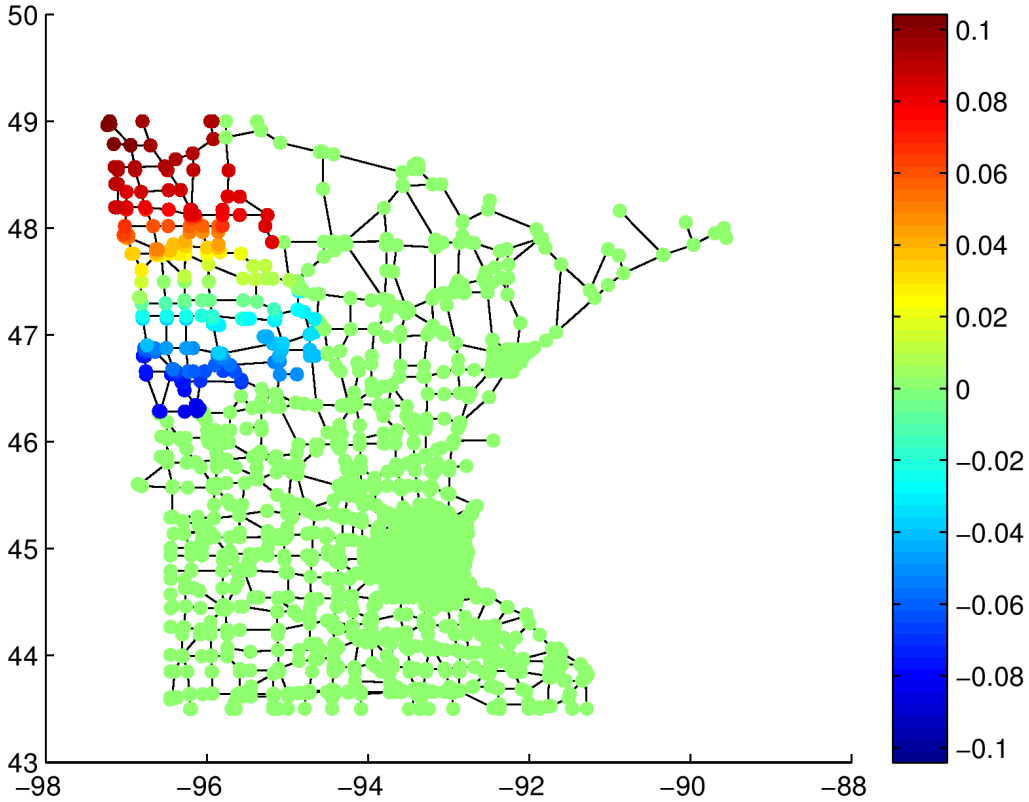
Level $j = 2$, Region $k = 1$, $\boldsymbol{\phi}_{1,2}^2$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

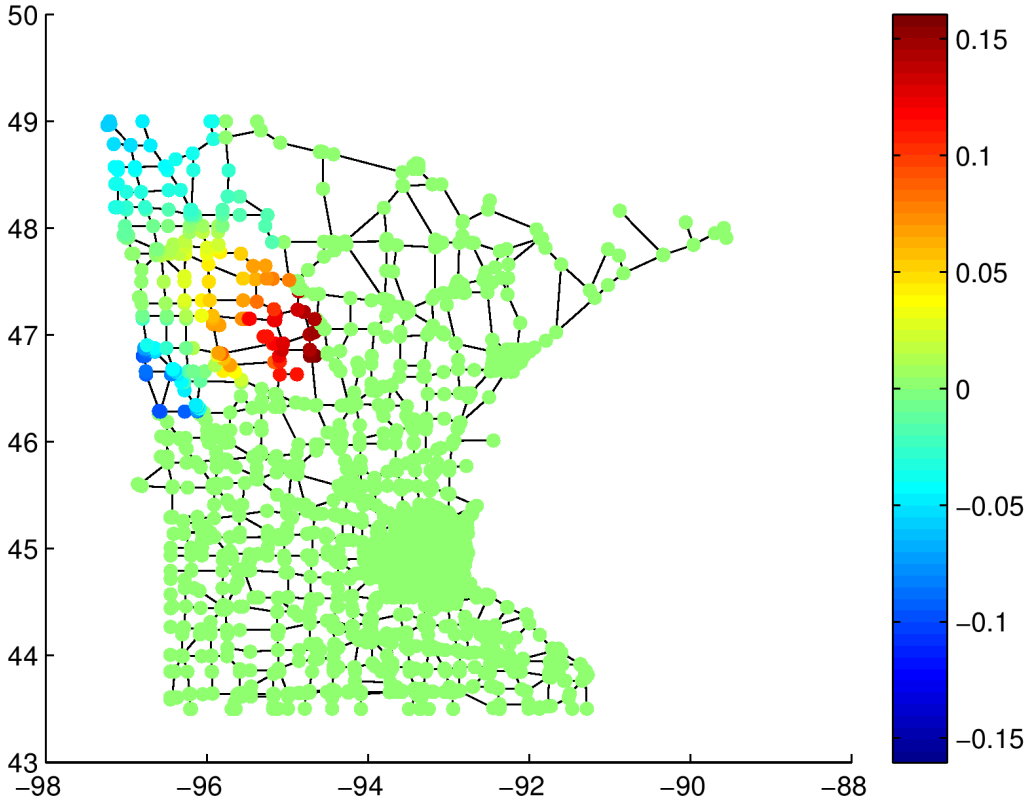
Level $j = 3$, Region $k = 0$, $\boldsymbol{\phi}_{0,1}^{3}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

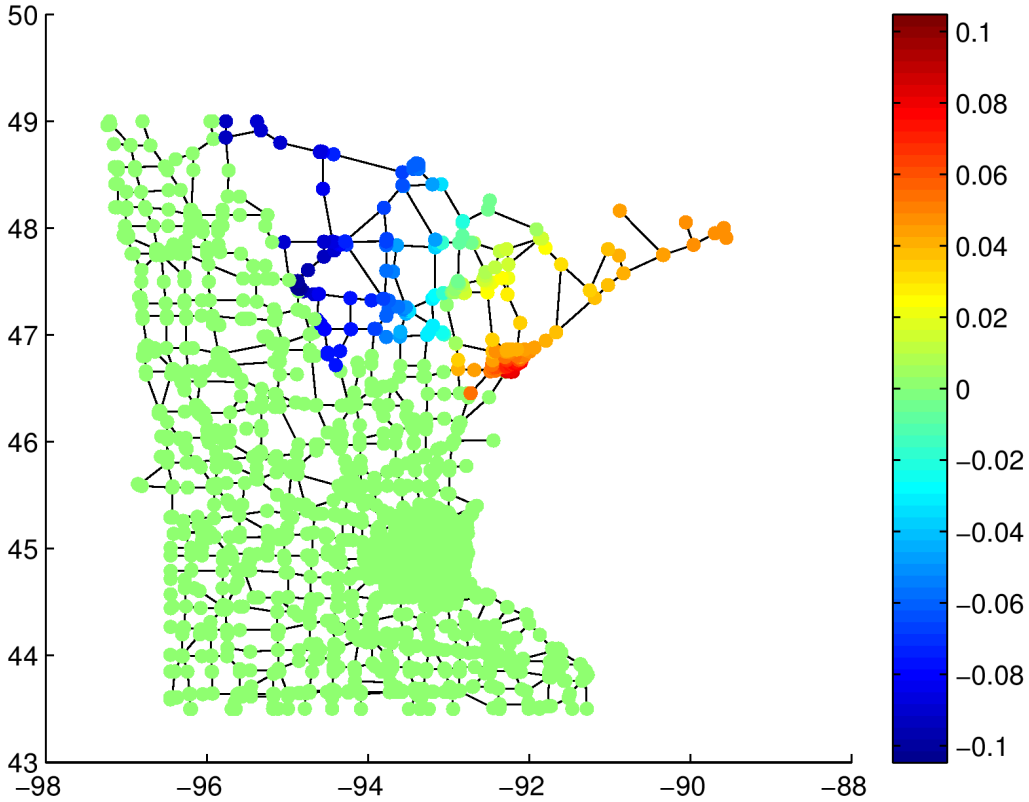
Level $j = 3$, Region $k = 0$, $\boldsymbol{\phi}_{0,2}^{3}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

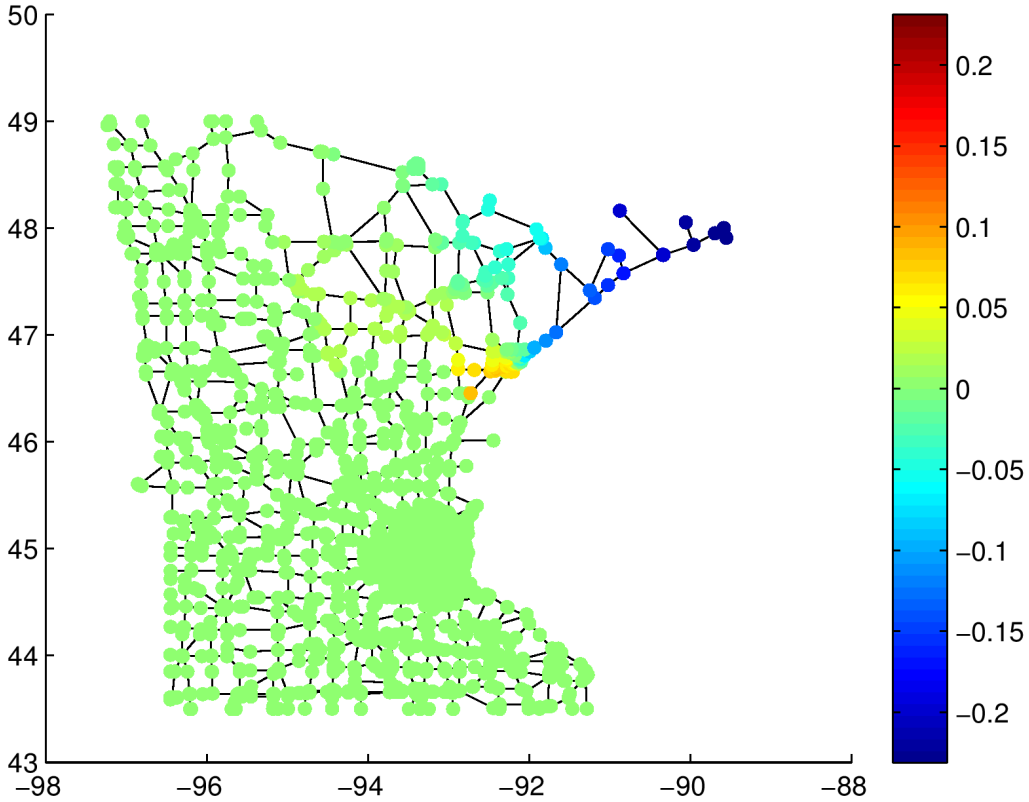
Level $j = 3$, Region $k = 1$, $\boldsymbol{\phi}_{1,1}^{3}$



HGLET Basis Vectors on MN

Here we display some of the basis vectors generated by our HGLET scheme on the MN road network. (Note: $j = 0$ is the coarsest scale, $j = 14$ is the finest.)

Level $j = 3$, Region $k = 1$, $\boldsymbol{\phi}_{1,2}^{3}$



Computational Complexity: HGLET

	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	67 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and $\texttt{nnz}(W) = 6604$.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] We have also developed a method that, like the HGLET, generates a full orthonormal basis on each each region at each level. The basis consists of piecewise-constant orthonormal vectors, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] We have also developed a method that, like the HGLET, generates a full orthonormal basis on each each region at each level. The basis consists of piecewise-constant orthonormal vectors, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow \psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] We have also developed a method that, like the HGLET, generates a full orthonormal basis on each each region at each level. The basis consists of piecewise-constant orthonormal vectors, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow$ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] We have also developed a method that, like the HGLET, generates a full orthonormal basis on each each region at each level. The basis consists of piecewise-constant orthonormal vectors, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow$ $\psi_{j,k}$
4. Repeat...

This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] We have also developed a method that, like the HGLET, generates a full orthonormal basis on each each region at each level. The basis consists of piecewise-constant orthonormal vectors, similar to the *Walsh-Hadamard transform*.

Now we present a Haar-like modification of our scheme:

1. Starting with the entire graph (i.e., level $j = 0$), compute the Fiedler vector $\boldsymbol{\phi}_1$ ($\boldsymbol{\phi}_0$ is trivially known, and we denote it by $\varphi_{0,0}$). Convert $\boldsymbol{\phi}_1$ to a Haar-like vector:[1]

$$\psi_{0,0}(i) := \begin{cases} 1 & \text{if } \boldsymbol{\phi}_1(i) \geq 0 \\ -\frac{\#\text{ nonnegative}}{\#\text{ negative}} & \text{if } \boldsymbol{\phi}_1(i) < 0 \end{cases}$$

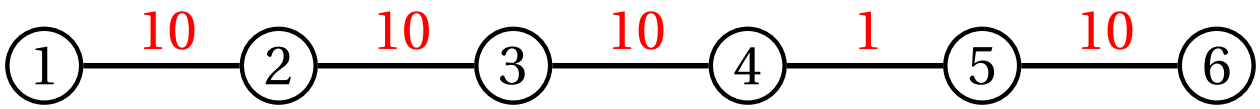
   and then ℓ^2-normalize it

2. Partition the graph $\Rightarrow$ Fiedler vector
3. Compute the Fiedler vector for each partition and convert it to a Haar-like vector on its respective partition[1] $\Rightarrow$ $\psi_{j,k}$
4. Repeat...

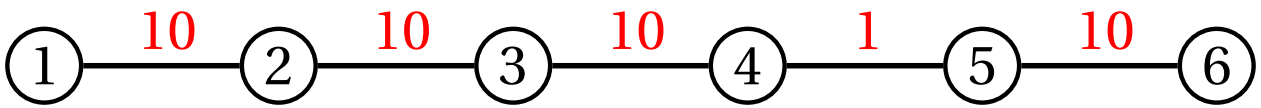
This yields an orthonormal basis: $\varphi_{0,0} \cup \{\psi_{j,k}\}_{0 \leq j < J,\ k}$

[1] We have also developed a method that, like the HGLET, generates a full orthonormal basis on each each region at each level. The basis consists of piecewise-constant orthonormal vectors, similar to the *Walsh-Hadamard transform*.

GHT Basis Example

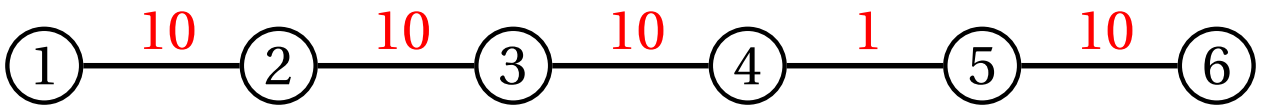


GHT Basis Example



$$\begin{bmatrix} \varphi_{0,0} & \psi_{0,0} & & & & \end{bmatrix}$$

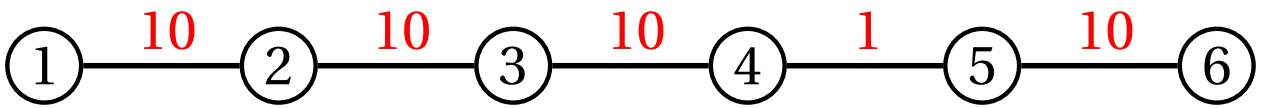
GHT Basis Example



$$\begin{bmatrix} \varphi_{0,0} & \psi_{0,0} & & & & \end{bmatrix}$$

$$\begin{bmatrix} & \psi_{1,0} & & & \end{bmatrix} \quad \begin{bmatrix} & & & \psi_{1,1} & \end{bmatrix}$$

GHT Basis Example

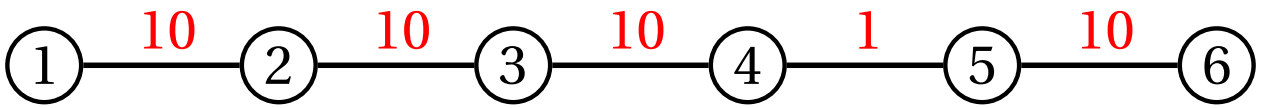


$$\begin{bmatrix} \varphi_{0,0} & \psi_{0,0} & & & & \end{bmatrix}$$

$$\begin{bmatrix} & \psi_{1,0} & & & \end{bmatrix} \quad \begin{bmatrix} & \psi_{1,1} & \end{bmatrix}$$

$$\begin{bmatrix} & \psi_{2,0} & \end{bmatrix} \quad \begin{bmatrix} & \psi_{2,1} & \end{bmatrix} \quad \begin{bmatrix} & \psi_{1,1} & \end{bmatrix}$$

GHT Basis Example



$$\begin{bmatrix} \varphi_{0,0} & \psi_{0,0} & & & & \end{bmatrix}$$
$$\begin{bmatrix} & \psi_{1,0} & & & \end{bmatrix} \quad \begin{bmatrix} & \psi_{1,1} & \end{bmatrix}$$
$$\begin{bmatrix} & \psi_{2,0} & \end{bmatrix} \quad \begin{bmatrix} & \psi_{2,1} & \end{bmatrix} \quad \begin{bmatrix} & \psi_{1,1} & \end{bmatrix}$$

Thus, we generate a matrix whose columns (after ℓ^2-normalization) form an orthonormal basis:

$\varphi_{0,0}$	$\psi_{0,0}$	$\psi_{1,0}$	$\psi_{1,1}$	$\psi_{2,0}$	$\psi_{2,1}$
1	1	1	0	1	0
1	1	1	0	−1	0
1	1	−1	0	0	1
1	1	−1	0	0	−1
1	−2	0	1	0	0
1	−2	0	−1	0	0

Computational Complexity: GHT

	Computational Complexity	Run Time for MN[1]
HGLET (redundant)	$O(N^3)$	67 sec
GHT	$O(N \log N)$	4 sec

[1] Computations performed on a personal laptop (4.00 GB RAM, 2.26 GHz), $N = 2640$ and $\mathtt{nnz}(W) = 6604$.

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is best for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET bases that is best for approximation.

As before, we require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \texttt{norm(x,p)}$$

- For my approximation experiments, I use $p = 0.1$.

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is best for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET bases that is best for approximation.

As before, we require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \texttt{norm(x,p)}$$

- For my approximation experiments, I use $p = 0.1$.

Coifman and Wickerhauser (1992) developed the best-basis algorithm as a means of selecting the basis from a dictionary of wavelet packets that is best for approximation/compression.

We generalize this approach, developing and implementing an algorithm for selecting the basis from the dictionary of HGLET bases that is best for approximation.

As before, we require a cost functional $\mathscr{J}$. For example:

$$\mathscr{J}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \texttt{norm(x,p)}$$

- For my approximation experiments, I use $p = 0.1$.

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{bmatrix}$$
$$\begin{matrix} d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{matrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{bmatrix}$$
$$\begin{matrix} d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{matrix} \qquad \begin{matrix} d^1_{1,0} & d^1_{1,1} & d^1_{1,2} & \cdots & d^1_{1,N_1-1} \end{matrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N_1-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \end{bmatrix}$$
$$\begin{matrix} d^2_{0,0} & d^2_{0,1} \cdots & d^2_{0,N_0-1} \end{matrix} \quad \begin{matrix} d^2_{1,0} & d^2_{1,1} \cdots & d^2_{1,N_1-1} \end{matrix} \quad \begin{matrix} d^2_{2,0} & d^2_{2,1} \cdots & d^2_{2,N_2-1} \end{matrix} \quad \begin{matrix} d^2_{3,0} & d^2_{3,1} \cdots & d^2_{3,N_3-1} \end{matrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \\ d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \\ d^1_{1,0} & d^1_{1,1} & d^1_{1,2} & \cdots & d^1_{1,N_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} \boldsymbol{\phi}^2_{0,1} \cdots \boldsymbol{\phi}^2_{0,N_0-1} \\ d^2_{0,0}\ d^2_{0,1} \cdots d^2_{0,N_0-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} \boldsymbol{\phi}^2_{1,1} \cdots \boldsymbol{\phi}^2_{1,N_1-1} \\ d^2_{1,0}\ d^2_{1,1} \cdots d^2_{1,N_1-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0}\ d^2_{2,1} \cdots d^2_{2,N_2-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0}\ d^2_{3,1} \cdots d^2_{3,N_3-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \end{bmatrix}$$
$$\begin{matrix} d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{matrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \end{bmatrix}$$
$$\begin{matrix} d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{matrix} \qquad \begin{matrix} d^1_{1,0} & d^1_{1,1} & d^1_{1,2} & \cdots & d^1_{1,N_1-1} \end{matrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{0,0} & \boldsymbol{\phi}^2_{0,1} & \cdots & \boldsymbol{\phi}^2_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{1,0} & \boldsymbol{\phi}^2_{1,1} & \cdots & \boldsymbol{\phi}^2_{1,N_1-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \boldsymbol{\phi}^2_{2,1} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \boldsymbol{\phi}^2_{3,1} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \end{bmatrix}$$
$$\begin{matrix} d^2_{0,0} & d^2_{0,1} & \cdots & d^2_{0,N_0-1} \end{matrix} \quad \begin{matrix} d^2_{1,0} & d^2_{1,1} & \cdots & d^2_{1,N_1-1} \end{matrix} \quad \begin{matrix} d^2_{2,0} & d^2_{2,1} & \cdots & d^2_{2,N_2-1} \end{matrix} \quad \begin{matrix} d^2_{3,0} & d^2_{3,1} & \cdots & d^2_{3,N_3-1} \end{matrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \\ d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \\ d^1_{1,0} & d^1_{1,1} & d^1_{1,2} & \cdots & d^1_{1,N_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0}\ d^2_{2,1} \cdots d^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0}\ d^2_{3,1} \cdots d^2_{3,N_3-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \\ d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^1_{1,0} & \boldsymbol{\phi}^1_{1,1} & \boldsymbol{\phi}^1_{1,2} & \cdots & \boldsymbol{\phi}^1_{1,N_1-1} \\ d^1_{1,0} & d^1_{1,1} & d^1_{1,2} & \cdots & d^1_{1,N_1-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0}\ d^2_{2,1} \cdots\ d^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0}\ d^2_{3,1} \cdots\ d^2_{3,N_3-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \\ d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0}\ d^2_{2,1} \cdots\ d^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0}\ d^2_{3,1} \cdots\ d^2_{3,N_3-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^0_{0,0} & \boldsymbol{\phi}^0_{0,1} & \boldsymbol{\phi}^0_{0,2} & \cdots & \boldsymbol{\phi}^0_{0,N-1} \\ d^0_{0,0} & d^0_{0,1} & d^0_{0,2} & \cdots & d^0_{0,N-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} & \boldsymbol{\phi}^2_{2,1} & \cdots & \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0} & d^2_{2,1} & \cdots & d^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} & \boldsymbol{\phi}^2_{3,1} & \cdots & \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0} & d^2_{3,1} & \cdots & d^2_{3,N_3-1} \end{bmatrix}$$

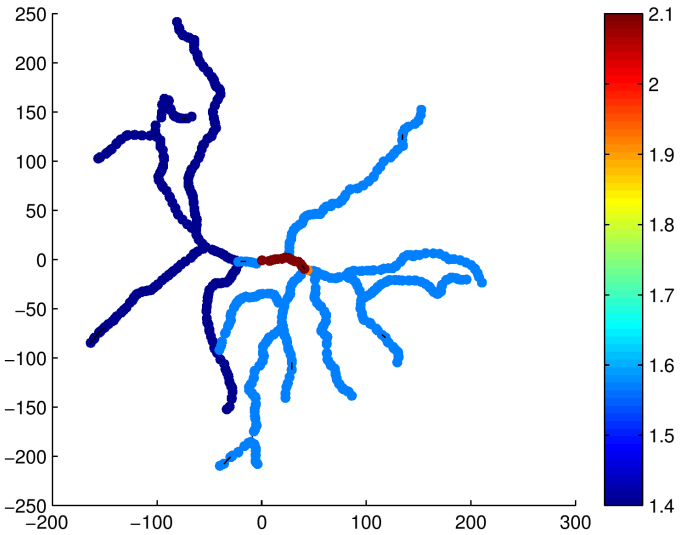
$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0}\ d^2_{2,1} \cdots\ d^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0}\ d^2_{3,1} \cdots\ d^2_{3,N_3-1} \end{bmatrix}$$

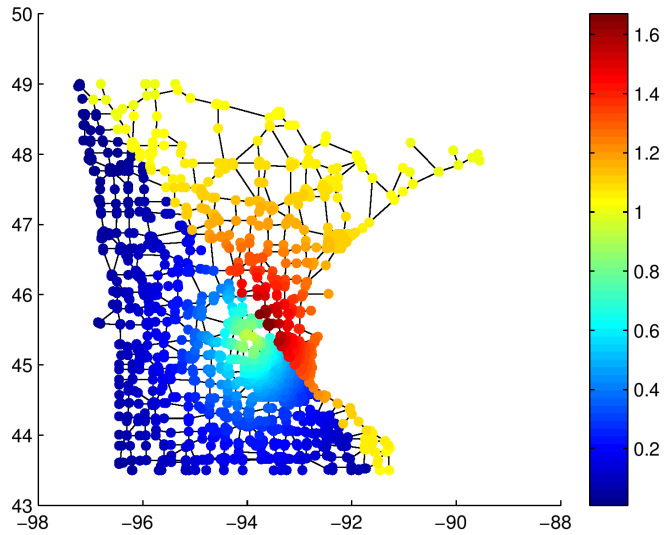
$$\begin{bmatrix} \boldsymbol{\phi}^1_{0,0} & \boldsymbol{\phi}^1_{0,1} & \boldsymbol{\phi}^1_{0,2} & \cdots & \boldsymbol{\phi}^1_{0,N_0-1} \\ d^1_{0,0} & d^1_{0,1} & d^1_{0,2} & \cdots & d^1_{0,N_0-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{\phi}^2_{2,0} \boldsymbol{\phi}^2_{2,1} \cdots \boldsymbol{\phi}^2_{2,N_2-1} \\ d^2_{2,0}\ d^2_{2,1} \cdots\ d^2_{2,N_2-1} \end{bmatrix} \quad \begin{bmatrix} \boldsymbol{\phi}^2_{3,0} \boldsymbol{\phi}^2_{3,1} \cdots \boldsymbol{\phi}^2_{3,N_3-1} \\ d^2_{3,0}\ d^2_{3,1} \cdots\ d^2_{3,N_3-1} \end{bmatrix}$$

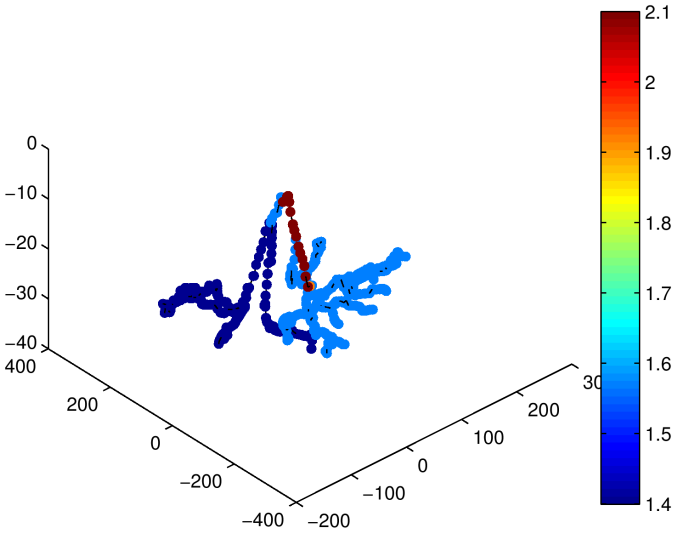
According to cost functional $\mathscr{J}$, this is the best basis for approximation.



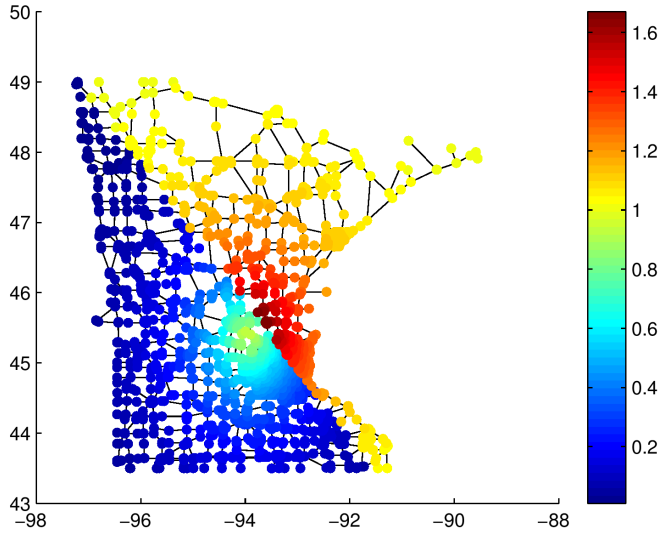
(a) Thickness data on a dendritic tree



(b) A mutilated Gaussian on the MN road network

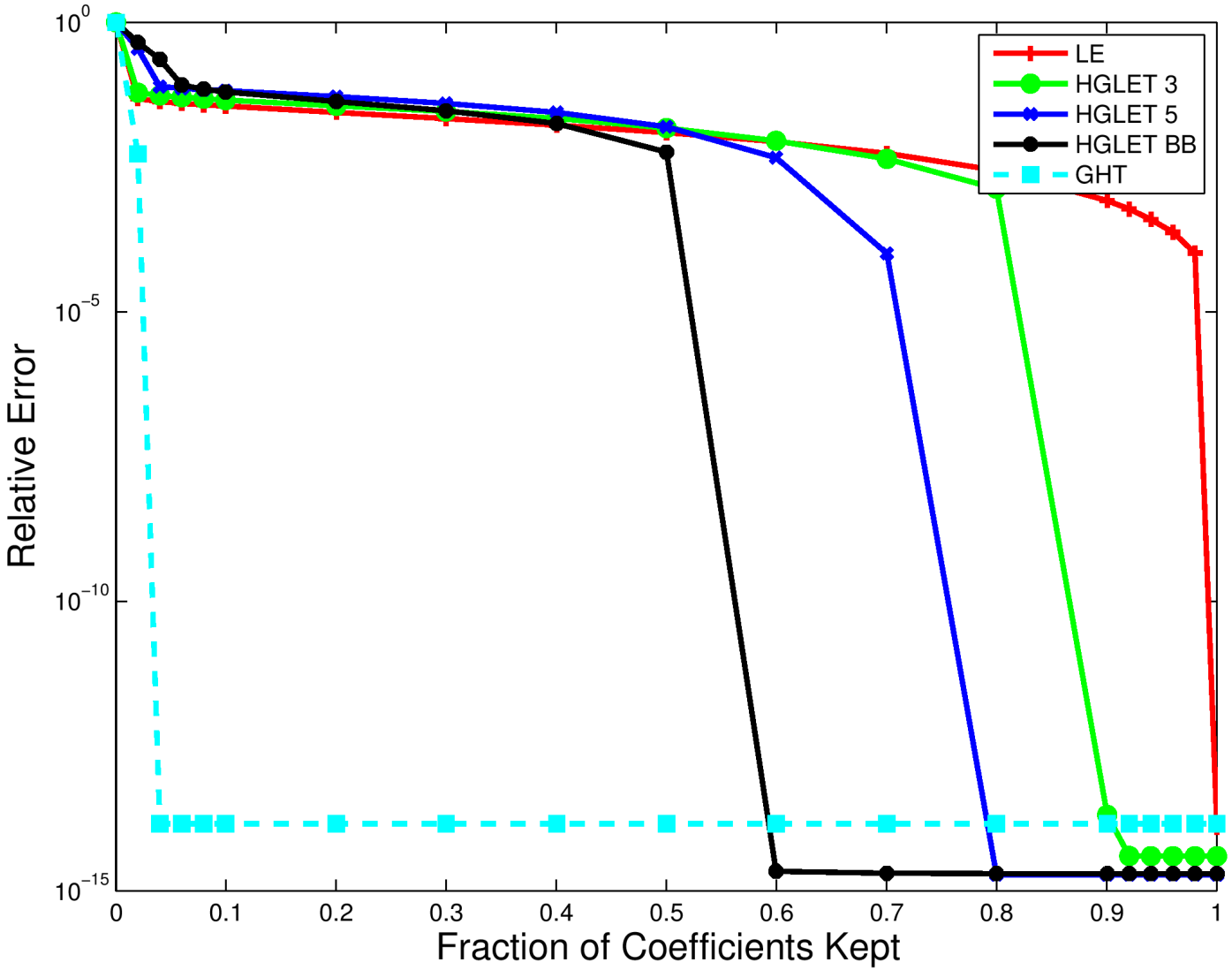


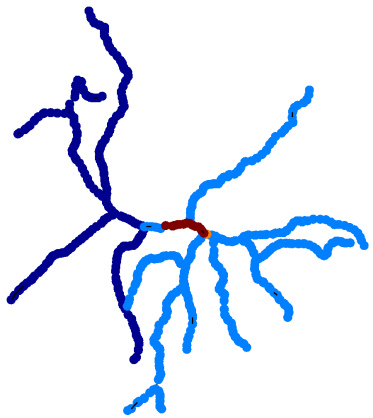
(a) Thickness data on a dendritic tree



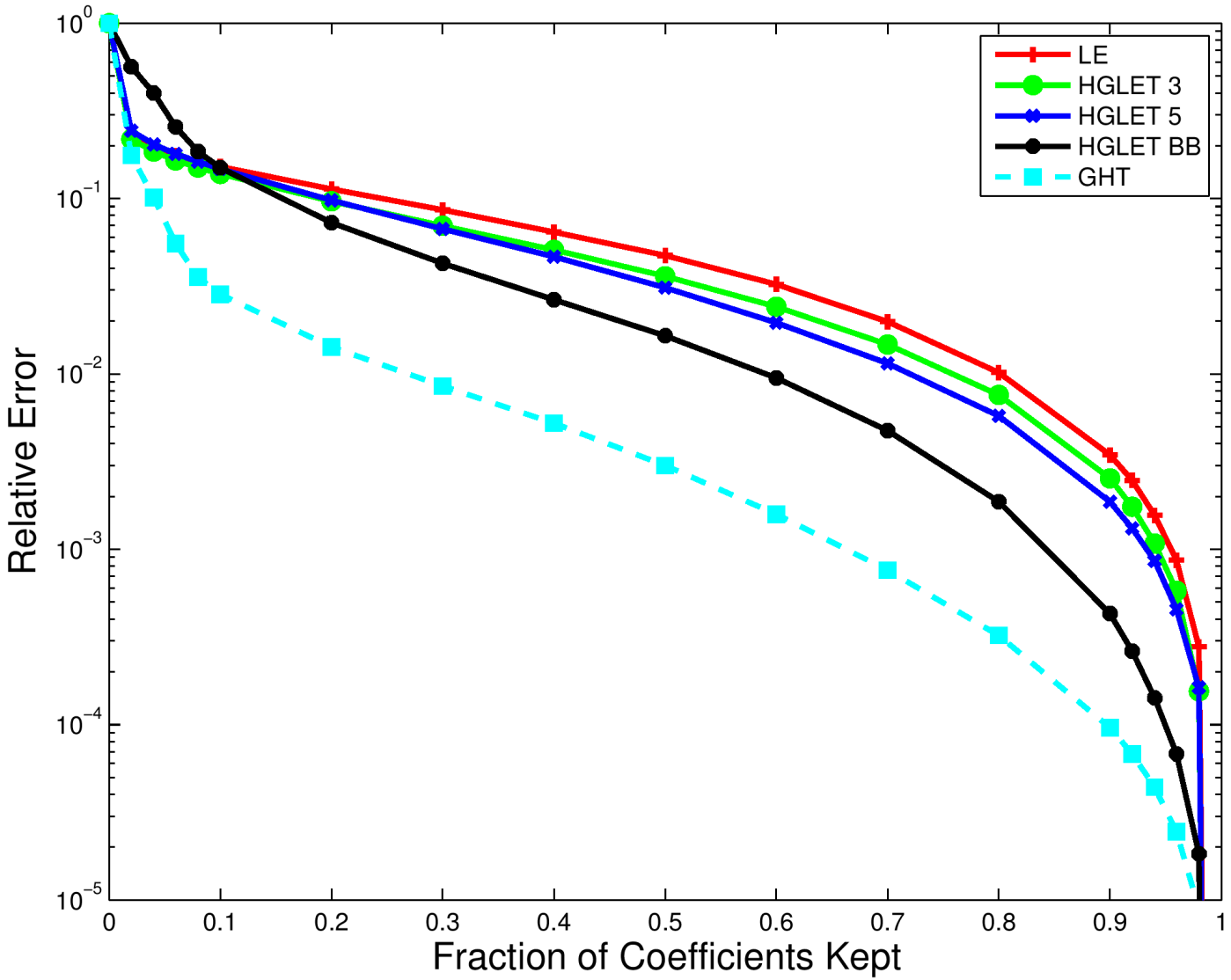
(b) A mutilated Gaussian on the MN road network

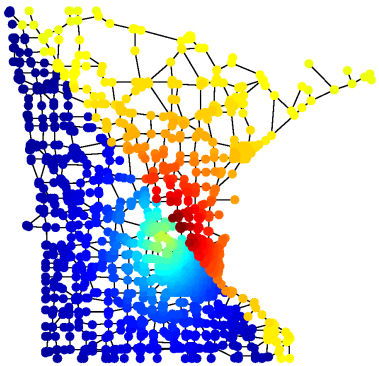
Dendrite (L_{rw}, adjacency matrix)



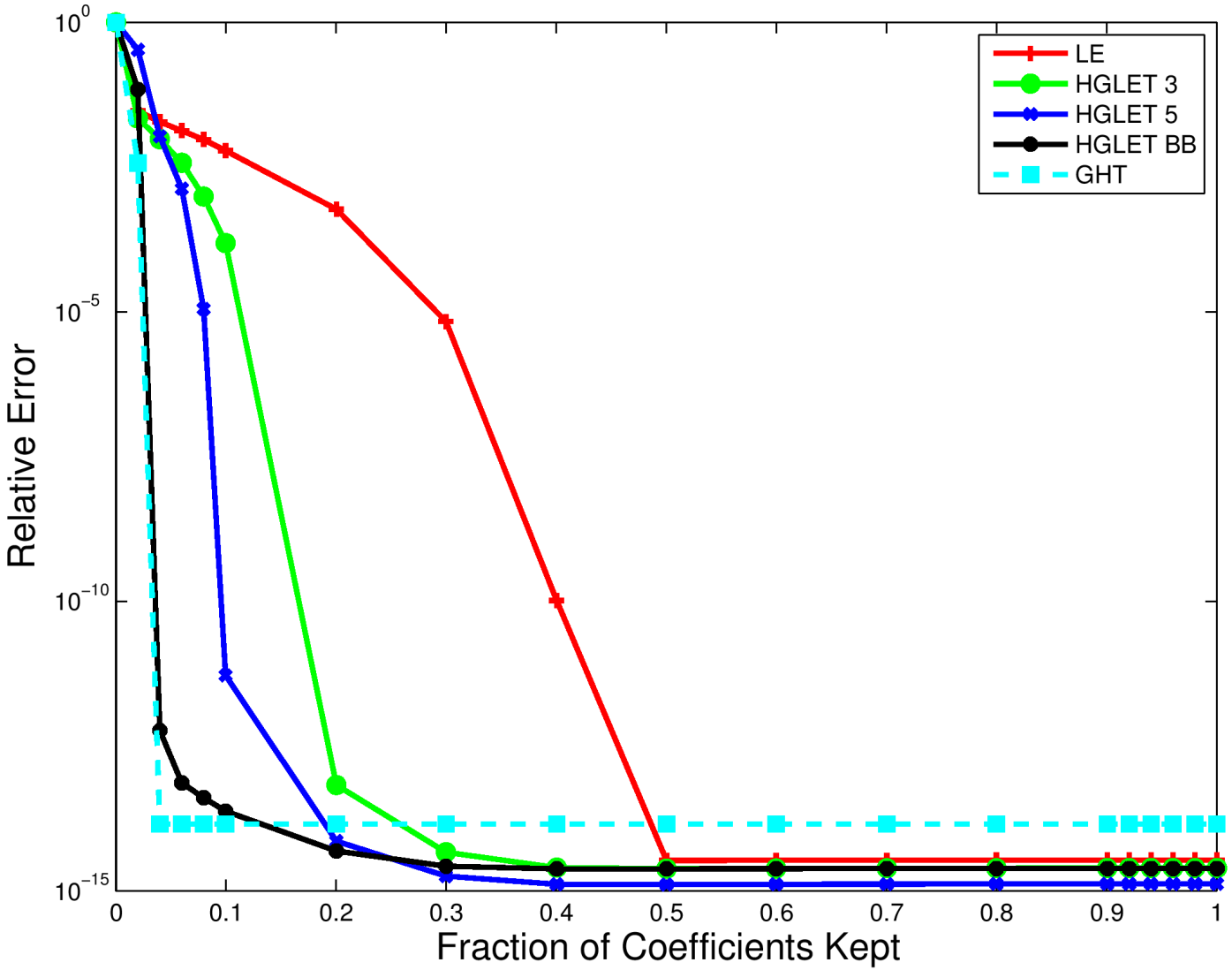


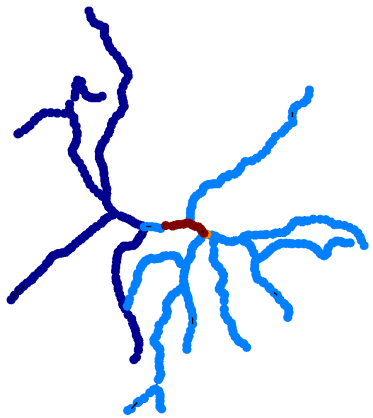
MN Mutilated Gaussian (L_{rw}, adjacency matrix)



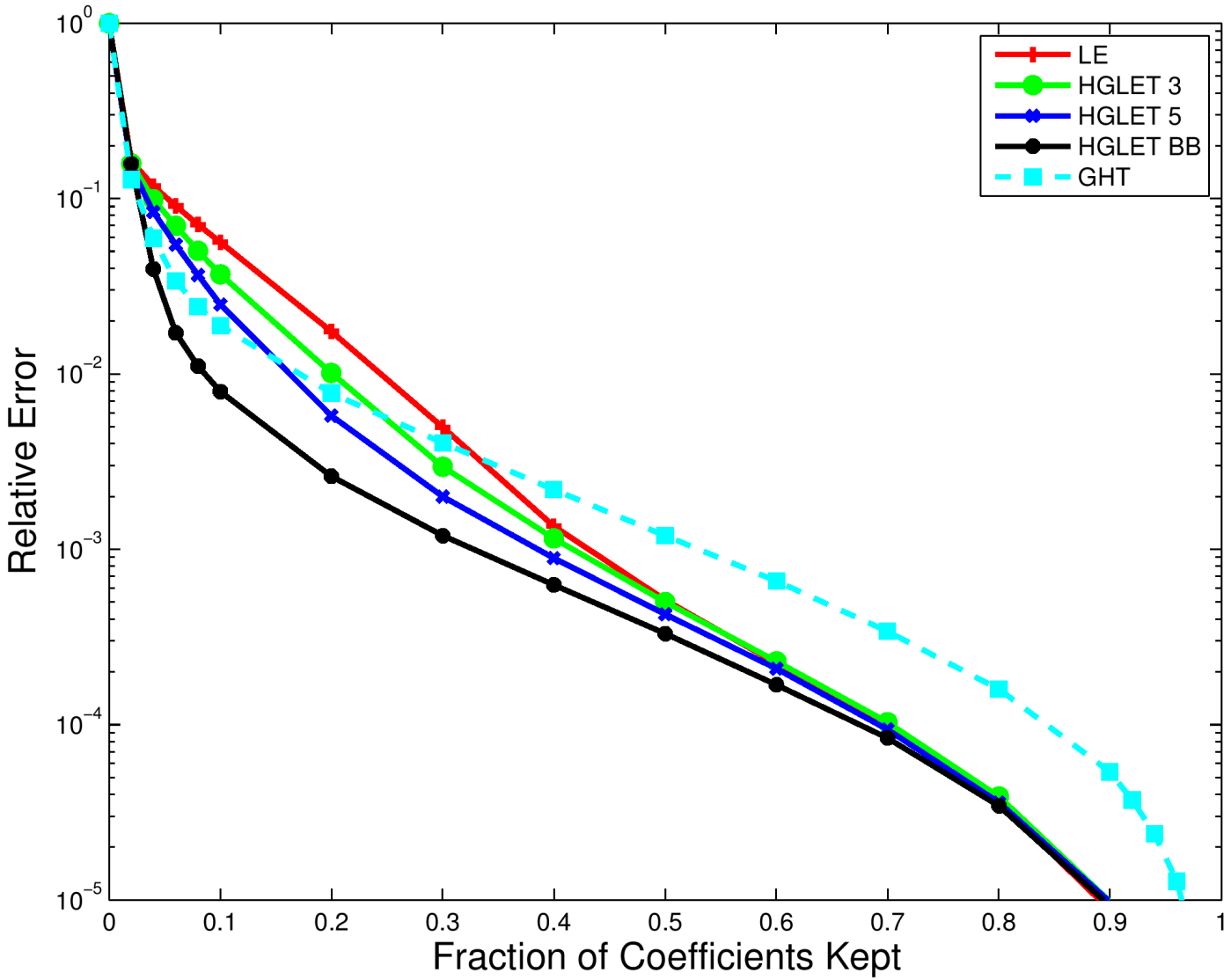


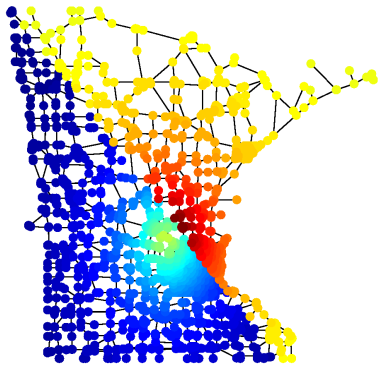
Dendrite (L, inverse Euclidean weights)





MN Mutilated Gaussian (L, inverse Euclidean weights)





Discussion of Approximation Results

- Overall, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
  - This demonstrates the merit of using *localized basis vectors*.
- For all data sets, the best performer is either the GHT or the best-basis HGLET.
  - This makes a strong case for using localized basis vectors on *multiple scales*.
  - It's no surprise that the GHT is the best performer on the Dendrite data set.
  - How does the GHT compete on the smoother MN data set?
    - $\Rightarrow$ Its basis contains one Haar-like vector for *each and every region* in the hierarchical partitioning.

Discussion of Approximation Results

- Overall, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
  - This demonstrates the merit of using *localized basis vectors*.
- For all data sets, the best performer is either the GHT or the best-basis HGLET.
  - This makes a strong case for using localized basis vectors on *multiple scales*.
  - It's no surprise that the GHT is the best performer on the Dendrite data set.
  - How does the GHT compete on the smoother MN data set?
    - $\Rightarrow$ Its basis contains one Haar-like vector for *each and every region* in the hierarchical partitioning.

Discussion of Approximation Results

- Overall, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
  - This demonstrates the merit of using *localized basis vectors*.
- For all data sets, the best performer is either the GHT or the best-basis HGLET.
  - This makes a strong case for using localized basis vectors on *multiple scales*.
  - It's no surprise that the GHT is the best performer on the Dendrite data set.
  - How does the GHT compete on the smoother MN data set?
    - ⇒ Its basis contains one Haar-like vector for *each and every region* in the hierarchical partitioning.

Discussion of Approximation Results

- Overall, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
  - This demonstrates the merit of using *localized basis vectors*.
- For all data sets, the best performer is either the GHT or the best-basis HGLET.
  - This makes a strong case for using localized basis vectors on *multiple scales*.
  - It's no surprise that the GHT is the best performer on the Dendrite data set.
  - How does the GHT compete on the smoother MN data set?
    - ⇒ Its basis contains one Haar-like vector for *each and every region* in the hierarchical partitioning.

Discussion of Approximation Results

- Overall, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
  - This demonstrates the merit of using *localized basis vectors*.
- For all data sets, the best performer is either the GHT or the best-basis HGLET.
  - This makes a strong case for using localized basis vectors on *multiple scales*.
  - It's no surprise that the GHT is the best performer on the Dendrite data set.
  - How does the GHT compete on the smoother MN data set?

    $\Rightarrow$ Its basis contains one Haar-like vector for *each and every region* in the hierarchical partitioning.

Discussion of Approximation Results

- Overall, we see that HGLET best-basis > HGLET Level 5 > HGLET Level 3 > Laplacian eigenvectors (HGLET Level 0)
  - This demonstrates the merit of using *localized basis vectors*.
- For all data sets, the best performer is either the GHT or the best-basis HGLET.
  - This makes a strong case for using localized basis vectors on *multiple scales*.
  - It's no surprise that the GHT is the best performer on the Dendrite data set.
  - How does the GHT compete on the smoother MN data set?
    - ⇒ Its basis contains one Haar-like vector for *each and every region* in the hierarchical partitioning.

Summary

- We developed multiscale transforms on graphs and networks: HGLET and GHT.
  - Developing a *true* generalization of wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
  - It allows us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, ...
- They may also be useful for regularly-sampled signals.

Summary

- We developed multiscale transforms on graphs and networks: HGLET and GHT.
  - Developing a *true* generalization of wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
  - It allows us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, ...
- They may also be useful for regularly-sampled signals.

Summary

- We developed multiscale transforms on graphs and networks: HGLET and GHT.
  - Developing a *true* generalization of wavelet and wavelet packet transforms is more challenging due to the difficulty of the notion of the *frequency domain* of a given graph.
- The HGLET is a direct generalization of *Hierarchical Block Discrete Cosine Transforms* originally developed for regularly-sampled signals and images.
  - It allows us to choose an orthonormal basis most suitable for one's task at hand, e.g., approximation, classification, regression, ...
- They may also be useful for regularly-sampled signals.

Future Work

- ✓ Develop and implement a Generalized Haar-Walsh Transform (GHWT) for signals on graphs and networks.
- Perform classification experiments and compare the results using HGLET, GHT, and GHWT.
- Explore other methods for graph partitioning
  - Allow for splitting of a region into an arbitrary number of subregions
  - Consider a bottom-up clustering method, rather than a top-down partitioning method

Future Work

- ✓ Develop and implement a Generalized Haar-Walsh Transform (GHWT) for signals on graphs and networks.
- Perform classification experiments and compare the results using HGLET, GHT, and GHWT.
- Explore other methods for graph partitioning
  - Allow for splitting of a region into an arbitrary number of subregions
  - Consider a bottom-up clustering method, rather than a top-down partitioning method

Future Work

- ✓ Develop and implement a Generalized Haar-Walsh Transform (GHWT) for signals on graphs and networks.
- Perform classification experiments and compare the results using HGLET, GHT, and GHWT.
- Explore other methods for graph partitioning
  - Allow for splitting of a region into an arbitrary number of subregions
  - Consider a bottom-up clustering method, rather than a top-down partitioning method

References & Acknowledgments

- http://www.math.ucdavis.edu/~saito/courses/HarmGraph/ contains Naoki Saito's course slides and useful information on "Harmonic Analysis on Graphs and Networks"
- Also visit http://www.math.ucdavis.edu/~saito/publications/ for various related publications including:
  - N. Saito: "Data analysis and representation using eigenfunctions of Laplacian on a general domain," *Applied & Computational Harmonic Analysis*, vol. 25, no. 1, pp. 68–97, 2008.
  - N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Letters*, vol. 1, pp. 13–16, 2009.
  - J. Irion & N. Saito: "Hierarchical graph Laplacian eigen transforms," submitted for publication, 2013.

This research was partially supported by the grants received from the *Office of Naval Research* and the *National Defense Science and Engineering Graduate Fellowship*.