The First Steps toward Building Natural Graph Wavelets

Naoki Saito
partially in collaboration with
Haotian Li (UCD) and Alex Cloninger (UCSD)

Department of Mathematics
University of California, Davis

Graph Signal Processing Workshop
University of Minnesota, Minneapolis, MN
June 6, 2019
Outline

1 Motivations

2 Measuring Differences between Eigenvectors

3 Numerical Experiments

4 Organizing Laplacian Eigenvectors of Dendritic Trees

5 Other Methods for Eigenvector Organization

6 Further Plan for Building Natural Graph Wavelets

7 Summary
Acknowledgment

- NSF Grants: DMS-1418779, IIS-1631329
- ONR Grants: N00014-16-1-2255
- Qinglan Xia (UC Davis)
Outline

1. Motivations
2. Measuring Differences between Eigenvectors
3. Numerical Experiments
4. Organizing Laplacian Eigenvectors of Dendritic Trees
5. Other Methods for Eigenvector Organization
6. Further Plan for Building Natural Graph Wavelets
7. Summary
Motivations

- Using graph Laplacian eigenvectors as “cosines” or Fourier modes on graphs with eigenvalues as (the square of) their “frequencies” has been quite popular.
- However, the notion of *frequency* is ill-defined on general graphs and the Fourier transform is not properly defined on graphs.
- Graph Laplacian eigenvectors may also exhibit peculiar behaviors depending on *topology* and *structure* of given graphs!
- Spectral Graph Wavelet Transform (SGWT) of Hammond et al. derived wavelets on a graph based on *the Littlewood-Paley theory* that organized the graph Laplacian eigenvectors corresponding to dyadic partitions of eigenvalues by viewing the eigenvalues as “frequencies”.
- Unfortunately, this view is wrong other than very simple graphs, e.g., undirected unweighted paths and cycles.
A Simple Yet Important Example: A Path Graph

The eigenvectors of this matrix are exactly the **DCT Type II** basis vectors (used for the JPEG standard) while those of the **symmetrically-normalized Graph Laplacian matrix** \( L_{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \) are the **DCT Type I** basis! (See G. Strang, “The discrete cosine transform,” *SIAM Review*, vol. 41, pp. 135–147, 1999).

- \( \lambda_k = 2 - 2 \cos(\pi k/n) = 4 \sin^2(\pi k/2n), \ k = 0 : n - 1. \)
- \( \phi_k(\ell) = a_{k;n} \cos\left(\pi k \left(\ell + \frac{1}{2}\right) / n\right), \ k, \ell = 0 : n - 1; \ a_{k;n} \text{ is a const. s.t. } \|\phi_k\|_2 = 1. \)
- In this simple case, \( \lambda \) (eigenvalue) is a monotonic function w.r.t. the frequency, which is the eigenvalue index \( k. \) **For a general graph, however, the notion of frequency is not well defined.**
Problem with 2D Lattice Graph

- As soon as the domain becomes *even slightly more complicated than* unweighted and undirected paths/cycles, the situation completely changes: we cannot view the eigenvalues as a simple monotonic function of frequency anymore.

- For example, consider a thin strip in $\mathbb{R}^2$, and suppose that the domain is discretized as $P_m \times P_n$ ($m > n$), whose Laplacian eigenpairs are:

  \[
  \lambda_k = 4 \left[ \sin^2 \left( \frac{\pi k_x}{2m} \right) + \sin^2 \left( \frac{\pi k_y}{2n} \right) \right],
  \]

  \[
  \phi_k(x, y) = a_{k_x,m} a_{k_y,n} \cos \left( \frac{\pi k_x}{m} \left( x + \frac{1}{2} \right) \right) \cos \left( \frac{\pi k_y}{n} \left( y + \frac{1}{2} \right) \right),
  \]

  where $k = 0 : mn - 1$; $k_x = 0 : m - 1$; $k_y = 0 : n - 1$; $x = 0 : m - 1$; and $y = 0 : n - 1$.

- As always, let $\{\lambda_k\}_{k=0:mn-1}$ be ordered in the nondecreasing manner. In this case, the smallest eigenvalue is still $\lambda_0 = \lambda_{(0,0)} = 0$, and the corresponding eigenvector is constant.
The second smallest eigenvalue $\lambda_1$ is $\lambda_{(1,0)} = 4\sin^2(\pi/2m)$, since $\pi/2m < \pi/2n$, and its eigenvector has half oscillation in the $x$-direction.

But, how about $\lambda_2$? Even for such a simple situation there are two possibilities: If $m > 2n$, then $\lambda_2 = \lambda_{(2,0)} < \lambda_{(0,1)}$. On the other hand, if $n < m < 2n$, then $\lambda_2 = \lambda_{(0,1)} < \lambda_{(2,0)}$.

More generally, if $Kn < m < (K + 1)n$ for some $K \in \mathbb{N}$, then $\lambda_k = \lambda_{(k,0)} = 4\sin^2(k\pi/2m)$ for $k = 0, \ldots, K$. Yet we have $\lambda_{K+1} = \lambda_{(0,1)} = 4\sin^2(\pi/2n)$ and $\lambda_{K+2}$ is equal to either $\lambda_{(K+1,0)} = 4\sin^2((K+1)\pi/2m)$ or $\lambda_{(1,1)} = 4[\sin^2(\pi/2m) + \sin^2(\pi/2n)]$ depending on $m$ and $n$. 

saito@math.ucdavis.edu (UC Davis)
As one can see from this, the mapping between $k$ and $(k_x, k_y)$ is quite nontrivial. Notice that $\phi_{(k,0)}$ has $k/2$ oscillations in the $x$-direction whereas $\phi_{(0,1)}$ has only half oscillation in the $y$-direction.

In other words, all of a sudden the eigenvalue of a completely different type of oscillation sneaks into the eigenvalue sequence.

Hence, on a general domain or a general graph, by simply looking at the Laplacian eigenvalue sequence $\{\lambda_k\}_{k=0,1,...}$, it is almost impossible to organize the eigenpairs into physically meaningful dyadic blocks and apply the Littlewood-Paley approach unless the underlying domain is of very simple nature, e.g., $P_n$ or $C_n$.

For complicated domains, the notion of frequency is not well-defined anymore, and thus wavelet construction methods that rely on the Littlewood-Paley theory by viewing eigenvalues as the square of frequencies, such as the spectral graph wavelet transform (SGWT) of Hammond et al. may lead to unexpected problems on general graphs.
Motivations

What we want to do is to *organize* those eigenvectors as

\[
\begin{array}{cccccccc}
\Phi_{0, 0} & \Phi_{1, 0} & \Phi_{2, 0} & \Phi_{3, 0} & \Phi_{4, 0} & \Phi_{5, 0} & \Phi_{6, 0} \\
\Phi_{0, 1} & \Phi_{1, 1} & \Phi_{2, 1} & \Phi_{3, 1} & \Phi_{4, 1} & \Phi_{5, 1} & \Phi_{6, 1} \\
\Phi_{0, 2} & \Phi_{1, 2} & \Phi_{2, 2} & \Phi_{3, 2} & \Phi_{4, 2} & \Phi_{5, 2} & \Phi_{6, 2}
\end{array}
\]

instead of

\[
\begin{array}{cccccccc}
\Phi_{0} & \Phi_{1} & \Phi_{2} & \Phi_{3} & \Phi_{4} & \Phi_{5} & \Phi_{6} \\
\Phi_{7} & \Phi_{8} & \Phi_{9} & \Phi_{10} & \Phi_{11} & \Phi_{12} & \Phi_{13} \\
\Phi_{14} & \Phi_{15} & \Phi_{16} & \Phi_{17} & \Phi_{18} & \Phi_{19} & \Phi_{20}
\end{array}
\]
Outline

1. Motivations
2. Measuring Differences between Eigenvectors
3. Numerical Experiments
4. Organizing Laplacian Eigenvectors of Dendritic Trees
5. Other Methods for Eigenvector Organization
6. Further Plan for Building Natural Graph Wavelets
7. Summary
Measuring Differences between Eigenvectors

Plan

- **How can we** *quantify the difference between the eigenvectors?*
  - The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.
  - Consider the *optimal transport theory*!
    - Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$).
    - Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. Earth Mover’s Distance or 1st Wasserstein Distance), for all $i, j = 0:i-1$, which results in a “distance” matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$.
    - *Embed* the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., Multidimensional Scaling (MDS)).
    - Organize and group those points to generate wavelet-like vectors on $G$.
  - Can we get the “*dual geometry*” of $G$ in that embedded space?
Measuring Differences between Eigenvectors

Plan

- How can we quantify the difference between the eigenvectors?
- The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the optimal transport theory!
  - Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$).
  - Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. Earth Mover’s Distance or 1st Wasserstein Distance), for all $i, j = 0 : n-1$, which results in a “distance” matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$.
  - Embed the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., Multidimensional Scaling (MDS)).
  - Organize and group those points to generate wavelet-like vectors on $G$.
- Can we get the “dual geometry” of $G$ in that embedded space?
Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$)
  - Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. *Earth Mover’s Distance* or *1st Wasserstein Distance*), for all $i, j = 0 : n - 1$, which results in a “distance” matrix $D \in \mathbb{R}_{\geq 0}^{n \times n}$
  - *Embed* the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., *Multidimensional Scaling (MDS)*)
  - *Organize* and *group* those points to generate wavelet-like vectors on $G$
- Can we get the “*dual geometry*” of $G$ in that embedded space?
Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$)
  - Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. *Earth Mover’s Distance* or *1st Wasserstein Distance*), for all $i, j = 0 : n-1$, which results in a “distance” matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$
  - *Embed* the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., *Multidimensional Scaling (MDS)*)
  - Organize and group those points to generate wavelet-like vectors on $G$
- Can we get the “*dual geometry*” of $G$ in that embedded space?
How can we **quantify the difference between the eigenvectors**?

The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.

Consider the **optimal transport theory**!

- Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$)
- Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. *Earth Mover’s Distance* or *1st Wasserstein Distance*), for all $i, j = 0 : n - 1$, which results in a “distance” matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$
- **Embed** the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., *Multidimensional Scaling (MDS)*)
- **Organize** and **group** those points to generate wavelet-like vectors on $G$
- Can we get the “**dual geometry**” of $G$ in that embedded space?
Plan

- How can we \textit{quantify the difference between the eigenvectors}? 
- The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the \textit{optimal transport theory}!
  - Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$)
  - Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. \textit{Earth Mover’s Distance} or \textit{1st Wasserstein Distance}), for all $i, j = 0 : n - 1$, which results in a “distance” matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$
  - \textit{Embed} the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., \textit{Multidimensional Scaling (MDS)})
  - Organize and group those points to generate wavelet-like vectors on $G$
- Can we get the “\textit{dual geometry}” of $G$ in that embedded space?
Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$)
  - Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. *Earth Mover’s Distance or 1st Wasserstein Distance*), for all $i, j = 0 : n - 1$, which results in a “distance” matrix $D \in \mathbb{R}^{n \times n}_{\geq 0}$
  - *Embed* the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., *Multidimensional Scaling (MDS))*
  - *Organize* and *group* those points to *generate wavelet-like vectors on $G$*
  - Can we get the “*dual geometry*” of $G$ in that embedded space?
How can we quantify the difference between the eigenvectors?

The usual $\ell^2$-distance doesn’t work since $\|\phi_i - \phi_j\|_2 = \sqrt{2}\delta_{i\neq j}$.

Consider the optimal transport theory!

- Convert each $\phi_i$ to a probability mass function (pmf) $p_i$ over a graph $G$ (e.g., via squaring each component of $\phi_i$).
- Compute the cost to transport $p_i$ to $p_j$ optimally (a.k.a. Earth Mover’s Distance or 1st Wasserstein Distance), for all $i, j = 0 : n - 1$, which results in a “distance” matrix $D \in \mathbb{R}_{\geq 0}^{n \times n}$.
- Embed the eigenvectors into a lower dimensional Euclidean space, say, $\mathbb{R}^m$, $m \ll n$ (typically $m = 2$ or $m = 3$) so that the distances among those embedded points match with those given in $D$ (can use, e.g., Multidimensional Scaling (MDS)).
- Organize and group those points to generate wavelet-like vectors on $G$.

Can we get the “dual geometry” of $G$ in that embedded space?
Ramified Optimal Transportation (ROT) by Q. Xia

- is the study of transporting “mass” from one Radon measure (or simply a probability measure) $\mu^+$ to another $\mu^-$ along ramified transport paths with some specific transport cost functional.

- is the study of **branching** structures, e.g., trees; veins on a leaf; cardiovascular systems; river channel networks; electrical grids; communication networks, etc.
Definitions: Two discrete mass distributions (aka atomic measures) in $\mathbb{R}^d$: 

$$ a := \sum_{i=1}^{k} m_i \delta_{x_i}; \quad b := \sum_{j=1}^{l} n_j \delta_{y_j}; \quad \{x_i\}_i, \{y_j\}_j \subset \mathbb{R}^d; \quad \sum_{i=1}^{k} m_i = \sum_{j=1}^{l} n_j. $$

Let $\text{Path}(a, b)$ be all possible transport paths from $a$ to $b$ without cycles (Xia could manage to remove cycles), i.e., each $G \in \text{Path}(a, b)$ is a weighted acyclic directed graph with $\{x_i\}_i \cup \{y_j\}_j \subset V(G)$, whose edge weights ($>0$) satisfy the Kirchhoff law at each interior node $v \in V(G) \setminus \{x_i, y_j\}_i, j$:

$$ \sum_{e \in E(G); e^-=v} w(e) = \sum_{e \in E(G); e^+=v} w(e) + \begin{cases} 
    m_i & \text{if } v = x_i \text{ for some } i \in 1:k \\
    -n_j & \text{if } v = y_j \text{ for some } j \in 1:l \\
    0 & \text{otherwise.}
  \end{cases} $$

Define the cost of a transport path $G \in \text{Path}(a, b)$:

$$ M_\alpha(G) := \sum_{e \in E(G)} w(e)^\alpha \text{length}(e), \quad \alpha \in [0, 1]. $$
Xia further derived:

- Number of branching nodes in $\text{Path}(a, b)$ can be bounded from above by $k + l - 2$.
- The uniform lower bounds of minimum angle between any two edges in any $\alpha$-optimal path in $\text{Path}(a, b)$.
- The minimum transportation cost $d_\alpha(a, b) := \min_{G \in \text{Path}(a, b)} M_\alpha(G)$ is a metric on the space of atomic measures of equal mass and is of homogeneous of degree $\alpha$, i.e., $d_\alpha(\lambda a, \lambda b) = \lambda^\alpha d_\alpha(a, b), \forall \lambda > 0$.
- Numerical algorithms to compute the $\alpha$-optimal path for a given pair $(a, b)$. 

saito@math.ucdavis.edu (UC Davis)
Measuring Differences between Eigenvectors

ROT: Numerical Examples

(a)  
(b)  
(c)  
(d)  

\( \alpha = 1 \)  
\( \alpha = 0.75 \)  
\( \alpha = 0.5 \)  
\( \alpha = 0.25 \)
Our Method to Compute Transportation Costs

- Unlike the general ROT setting, a graph $G$ is fixed and given.
- In general, we want to deal with \textit{undirected} graphs.
- The ROT only deals with \textit{directed} graphs.
- Hence, we turn an undirected graph $G$ into the \textit{bidirected} graph $\tilde{G}$.
- To do so, we first compute the \textit{incidence matrix} $Q = [q_1|\cdots|q_m] \in \mathbb{R}^{n \times m}$ of the undirected graph $G = G(V,E)$ with $n = |V|$, $m = |E|$. Here, $q_k$ represents the endpoints of $e_k$: if $e_k$ joins nodes $i$ and $j$, then $q_k[l] = 1$ if $l = i$ or $l = j$; otherwise $q_k[l] = 0$.
- Then orient the edges in $E(G)$ in an arbitrary manner to form a directed graph $\tilde{G}$ whose incidence matrix $\tilde{Q}$ is, e.g.,

$$\tilde{q}_k[l] = \begin{cases} 
-1 & \text{if } l = i; \\
1 & \text{if } l = j; \\
0 & \text{otherwise.}
\end{cases}$$

- Finally, form the bidirected graph $\tilde{G}$ with $\tilde{Q} := [\tilde{Q} | -\tilde{Q}] \in \mathbb{R}^{n \times 2m}$. 
Given $\tilde{Q}$, we solve the *balance equation* that forces the Kirchhoff law:

$$\tilde{Q}w_{ij} = p_j - p_i, \quad w_{ij} \in \mathbb{R}^{2m}_{\geq 0}. \quad (\ast)$$

- The weight vector $w_{ij}$ describes the transportation plan of mass from $p_i$ to $p_j$, i.e., let $\tilde{G}_{ij}$ be the bidirected graph $\tilde{G}$ with these edge weights; then $\tilde{G}_{ij} \in \text{Path}(p_i, p_j)$.
- Eqn. (\ast) may have multiple solutions.
Our Method to Compute Transportation Costs . . .

- Currently, we use the following *Linear Programming* (LP):

\[
\min_{w_{ij} \in \mathbb{R}^{2m}} \|w_{ij}\|_1 \quad \text{subject to: } \tilde{Q}w_{ij} = p_j - p_i; w_{ij}[l] \geq 0, l = 0 : (2m - 1)
\]

which turned out to be better than using nonnegative least squares (NNLS) solver.

- Finally fill the distance matrix entries \( D = (D_{ij}) \):

\[
D_{ij} = M_\alpha(\tilde{G}_{ij}) = \sum_{e \in E(\tilde{G}_{ij})} w_{ij}(e)^\alpha \text{length}(e), \quad \alpha \in [0, 1].
\]

- Note that currently we are *not* examining all possible solutions of Eqn. (*) to search \( \min_{\tilde{G}_{ij} \in \text{Path}(p_i, p_j)} M_\alpha(\tilde{G}_{ij}) \).
Outline

1. Motivations
2. Measuring Differences between Eigenvectors
3. Numerical Experiments
4. Organizing Laplacian Eigenvectors of Dendritic Trees
5. Other Methods for Eigenvector Organization
6. Further Plan for Building Natural Graph Wavelets
7. Summary
Consolidated $w_{0,1}$: mass transport from $p_0 = \phi_0^2$ to $p_1 = \phi_1^2$
Consolidated $w_{0,1}$: mass transport from $p_0 = \phi_0^2$ to $p_1 = \phi_1^2$
2D Regular Lattice: Embedding into $\mathbb{R}^2$; $\alpha = 1$
Some symmetry could be explained because of the symmetry of DCT vectors:
\[
\phi_{k;n}^2[x] + \phi_{n-k;n}^2[x] = a_{k;n}^2 = 2/n, \ k = 1 : n - 1, \ x = 0 : n - 1.
\]
Other Ways to Turn $\phi_i$ into $p_i$

- Generating $\phi_i^2$ is not the only way to turn $\phi_i$ into a pmf $p_i$.
- Other examples include:
  - Normalized $\ell^1$: $\phi_i^1 := (|\phi_i[0]|, \ldots, |\phi_i[n-1]|)^T / \|\phi_i\|_1$;
  - A constant addition followed by normalization:
    $$\tilde{\phi}_i := \begin{cases} \phi_i^1 & \text{if } i = 0; \\ \frac{\phi_i - c_{\min} \cdot 1_n}{\|\phi_i - c_{\min} \cdot 1_n\|_1} & \text{if } i \neq 0, \end{cases}$$
    where $c_{\min} := \min_{0<i<n;0\leq l<n} \phi_i[l] < 0$;
  - Normalized exponentiation: $\phi_i^e := \exp(\phi_i) / \|\exp(\phi_i)\|_1$. 

saito@math.ucdavis.edu (UC Davis)
2D Regular Lattice; via $\{ \phi_i^e \}_i$, $\alpha = 0.25$
Outline

1. Motivations
2. Measuring Differences between Eigenvectors
3. Numerical Experiments
4. Organizing Laplacian Eigenvectors of Dendritic Trees
5. Other Methods for Eigenvector Organization
6. Further Plan for Building Natural Graph Wavelets
7. Summary
We observed an interesting phase transition phenomenon on the behavior of the eigenvalues of graph Laplacians defined on dendritic trees.

(a) RGC #100
We observed an interesting phase transition phenomenon on the behavior of the eigenvalues of graph Laplacians defined on dendritic trees.
We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*. 
A Peculiar Phase Transition Phenomenon . . .

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around junctions/bifurcation vertices.
A Peculiar Phase Transition Phenomenon ...

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like Fourier cosines/sines) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like wavelets) around junctions/bifurcation vertices.

(a) RGC #100; $\lambda_{1141} = 3.9994$
A Peculiar Phase Transition Phenomenon . . .

We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.

(a) RGC #100; \( \lambda_{1141} = 3.9994 \)
We have observed that this value 4 is critical since:

- the eigenfunctions corresponding to the eigenvalues below 4 are *semi-global oscillations* (like *Fourier cosines/sines*) over the entire dendrites or one of the dendrite arbors;
- those corresponding to the eigenvalues above 4 are much more *localized* (like *wavelets*) around *junctions/bifurcation vertices*.

(a) RGC #100; $\lambda_{1141} = 3.9994$

(b) RGC #100; $\lambda_{1142} = 4.3829$
We know why such localization/phase transition occurs \( \Rightarrow \) See our article for the detail: Y. Nakatsukasa, N. Saito, & E. Woei: “Mysteries around graph Laplacian eigenvalue 4,” *Linear Algebra & Its Applications*, vol. 438, no. 8, pp. 3231–3246, 2013. The key was the *discriminant* of a quadratic equation.

Any physiological consequence? Importance of branching vertices?

Many such eigenvector localization phenomena have been reported: Anderson localization, scars in quantum chaos, …


Our point is that eigenvectors, especially those corresponding to high eigenvalues, are quite sensitive to *topology and geometry of the underlying domain* and cannot really be viewed as high frequency oscillations unless the underlying graph is a simple unweighted path or cycle.

Hence, one must be very careful to develop an analog of *the Littlewood-Paley theory* for general graphs!
Embedding of Eigenvectors on the Dendritic Tree into $\mathbb{R}^3$

**Figure:** The magenta circle = the DC vector; the cyan circle = the Fiedler vector; the red circles = the localized eigenvectors; the larger colored circles = the eigenvectors supported on the upper-left branch.
Outline

1 Motivations
2 Measuring Differences between Eigenvectors
3 Numerical Experiments
4 Organizing Laplacian Eigenvectors of Dendritic Trees
5 Other Methods for Eigenvector Organization
6 Further Plan for Building Natural Graph Wavelets
7 Summary
Various Methods for Eigenvector Organization

- A similarity measure based on the average of local correlations of eigenvectors (A. Cloninger & S. Steinerberger, 2018)
- The difference of absolute gradient (DAG) method (H. Li & N. Saito, 2019)
- The time-stepping diffusion (TSD) method (H. Li & N. Saito, 2019)
- For the details of the latter two, see our forthcoming paper (to be presented at the SPIE Conference on Wavelets & Sparsity XVIII in San Diego, CA, this August!)
Outline

1 Motivations
2 Measuring Differences between Eigenvectors
3 Numerical Experiments
4 Organizing Laplacian Eigenvectors of Dendritic Trees
5 Other Methods for Eigenvector Organization
6 Further Plan for Building Natural Graph Wavelets
7 Summary
Natural Graph Wavelet Frame

- Given a graph $G = \{V, E, W\}$ with $|V| = n$ and the distance matrix $D$ of its eigenvectors, we can get a partition of all the eigenvectors based on some clustering method, $\mathcal{P} = \{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_N\}$, $1 \leq N \leq n$, in which
  \[
  \bigcup_{j=1}^{N} \mathcal{C}_j = \{1, 2, \cdots, n\} \quad \text{and} \quad \mathcal{C}_i \cap \mathcal{C}_j = \emptyset, \ i \neq j.
  \]

- In the following notation, the subindex $j$ stands for the cluster and the subindex $k$ represents the localization.

  \[
  \psi_{k,j} = \Phi F_j \Phi^T e_k \quad \text{for} \ j = 1, 2, \cdots, N \ \text{and} \ k = 1, 2, \cdots, n
  \]

  in which, the diagonal matrix $F_j \in \mathbb{R}^{n \times n}$ satisfies $F_j(l, l) = \chi_{\mathcal{C}_j}(l)$ for $l = 1, 2, \cdots, n$, $\Phi$ stores all the graph Laplacian eigenvectors, and $e_k$ is the canonical basis vector at vertex $v_k$.

- We can show that $\{\psi_{k,j}\}_{k=1,\cdots,n; j=1,\cdots,N}$ is a $N$ times redundant wavelet frame.
Natural Graph Wavelet Basis

- One way is to pick \( n \) vectors out of \( nN \) vectors in the frame \( \{ \psi_{k,j} \} \) by using \( \{ e_k \}_{k \in T_j} \), in which \( T_j \subset \{1,2,\ldots,n\} \), instead of all \( \{ e_k \}_{k=1,\ldots,n} \) for each \( C_j \), so that \( \sum_{j=1}^{N} |T_j| = n \). These \( n \) vectors may not be mutually orthogonal, so we may need some orthogonalization procedure.

- Another way is by sparsifying rotation. First, we permute \( \Phi \) into \( \hat{\Phi} \) based on \( P \): \( \hat{\Phi} = [\phi_{C_1}, \phi_{C_2}, \ldots, \phi_{C_N}] \) in which \( C_j = \{ j_1, j_2, \ldots, j_l \} \) and \( \phi_{C_j} = [\phi_{j_1}, \phi_{j_2}, \ldots, \phi_{j_l}] \). Then, we rotate \( \hat{\Phi} \) within each cluster \( C_j \) for \( j = 1,2,\ldots,N \) to get an sparse orthonormal wavelet basis \( \Psi \in \mathbb{R}^{n \times n} \).

- Yet another way is not to use the MDS embedding and clustering, but use weighted linear combinations of the eigenvectors where the weights reflect the similarity between them, which can be derived from the distance matrix already computed.

  Or ...
Natural Graph Wavelet Basis

- One way is to pick \( n \) vectors out of \( nN \) vectors in the frame \( \{ \psi_{k,j} \} \) by using \( \{ e_k \}_{k \in T_j} \), in which \( T_j \subset \{1,2,\cdots,n\} \), instead of all \( \{ e_k \}_{k=1,\cdots,n} \) for each \( C_j \), so that \( \sum_{j=1}^{N} |T_j| = n \). These \( n \) vectors may not be mutually orthogonal, so we may need some orthogonalization procedure.

- Another way is by sparsifying rotation. First, we permute \( \Phi \) into \( \hat{\Phi} \) based on \( \mathcal{P} \): \( \hat{\Phi} = [ \phi_{C_1}, \phi_{C_2}, \cdots, \phi_{C_N} ] \) in which \( C_j = \{ j_1, j_2, \cdots, j_l \} \) and \( \phi_{C_j} = [ \phi_{j_1}, \phi_{j_2}, \cdots, \phi_{j_l} ] \). Then, we rotate \( \hat{\Phi} \) within each cluster \( C_j \) for \( j = 1,2,\cdots,N \) to get an sparse orthonormal wavelet basis \( \Psi \in \mathbb{R}^{n \times n} \).

- Yet another way is not to use the MDS embedding and clustering, but use weighted linear combinations of the eigenvectors where the weights reflect the similarity between them, which can be derived from the distance matrix already computed.

- Or ...
Natural Graph Wavelet Basis

- One way is to **pick n vectors out of nN vectors** in the frame \( \{\psi_{k,j}\} \) by using \( \{e_k\}_{k \in T_j} \), in which \( T_j \subset \{1, 2, \cdots, n\} \), instead of all \( \{e_k\}_{k=1,\cdots,n} \) for each \( C_j \), so that \( \sum_{j=1}^{N} |T_j| = n \). These \( n \) vectors may not be mutually orthogonal, so we may need some **orthogonalization procedure**.

- Another way is by **sparsifying rotation**. First, we permute \( \Phi \) into \( \hat{\Phi} \) based on \( \mathcal{P} \): \( \hat{\Phi} = [\phi_{C_1}, \phi_{C_2}, \cdots, \phi_{C_N}] \) in which \( C_j = \{j_1, j_2, \cdots, j_l\} \) and \( \phi_{C_j} = [\phi_{j_1}, \phi_{j_2}, \cdots, \phi_{j_l}] \). Then, we rotate \( \hat{\Phi} \) within each cluster \( C_j \) for \( j = 1, 2, \cdots, N \) to get an sparse orthonormal wavelet basis \( \Psi \in \mathbb{R}^{n \times n} \).

- Yet another way is **not** to use the MDS embedding and clustering, but use **weighted linear combinations of the eigenvectors** where the weights reflect the **similarity** between them, which can be derived from the **distance matrix** already computed.

- Or ...
Natural Graph Wavelet Basis

- One way is to *pick n vectors out of nN vectors* in the frame \( \{ \psi_k, j \} \) by using \( \{ e_k \}_{k \in T_j} \), in which \( T_j \subset \{1, 2, \cdots, n\} \), instead of all \( \{ e_k \}_{k=1, \cdots, n} \) for each \( C_j \), so that \( \sum_{j=1}^{N} |T_j| = n \). These \( n \) vectors may not be mutually orthogonal, so we may need some *orthogonalization procedure*.

- Another way is by *sparsifying rotation*. First, we permute \( \Phi \) into \( \hat{\Phi} \) based on \( \mathcal{P} \): \( \hat{\Phi} = [ \phi_{C_1}, \phi_{C_2}, \cdots, \phi_{C_N} ] \) in which \( C_j = \{ j_1, j_2, \cdots, j_l \} \) and \( \phi_{C_j} = [ \phi_{j_1}, \phi_{j_2}, \cdots, \phi_{j_l} ] \). Then, we rotate \( \hat{\Phi} \) within each cluster \( C_j \) for \( j = 1, 2, \cdots, N \) to get an sparse orthonormal wavelet basis \( \Psi \in \mathbb{R}^{n \times n} \).

- Yet another way is *not* to use the MDS embedding and clustering, but use *weighted linear combinations of the eigenvectors* where the weights reflect the *similarity* between them, which can be derived from the *distance matrix* already computed.

- Or . . .
Using $k$-NN connection

- Find $k$-nearest neighbor vectors of $\phi_0$ w.r.t. the ROT distance, which form a space $V_0$ like in the multiresolution analysis
- For each eigenvector in $V_0$, find its $k$-NN vectors, collect them all to form $V_1 \supset V_0$, which is a space spanned by father wavelets
- Then set $W_0 := V_1 \setminus V_0$, which is a space spanned by mother wavelets
- Iterate this process to have $V_{j+1} = V_j \oplus W_j$
Using $k$-NN connection

- Find $k$-nearest neighbor vectors of $\phi_0$ w.r.t. the ROT distance, which form a space $V_0$ like in the *multiresolution analysis*
- For each eigenvector in $V_0$, find its $k$-NN vectors, collect them all to form $V_1 \supset V_0$, which is a space spanned by *father* wavelets
- Then set $W_0 := V_1 \setminus V_0$, which is a space spanned by *mother* wavelets
- Iterate this process to have $V_{j+1} = V_j \oplus W_j$
Using $k$-NN connection

- Find $k$-nearest neighbor vectors of $\phi_0$ w.r.t. the ROT distance, which form a space $V_0$ like in the *multiresolution analysis*.
- For each eigenvector in $V_0$, find its $k$-NN vectors, collect them all to form $V_1 \supseteq V_0$, which is a space spanned by *father* wavelets.
- Then set $W_0 := V_1 \setminus V_0$, which is a space spanned by *mother* wavelets.
- Iterate this process to have $V_{j+1} = V_j \oplus W_j$. 

saito@math.ucdavis.edu  (UC Davis)
Further Plan for Building Natural Graph Wavelets

**Using \( k \)-NN connection**

- Find \( k \)-nearest neighbor vectors of \( \phi_0 \) w.r.t. the ROT distance, which form a space \( V_0 \) like in the *multiresolution analysis*
- For each eigenvector in \( V_0 \), find its \( k \)-NN vectors, collect them all to form \( V_1 \supset V_0 \), which is a space spanned by *father* wavelets
- Then set \( W_0 := V_1 \setminus V_0 \), which is a space spanned by *mother* wavelets
- Iterate this process to have \( V_{j+1} = V_j \oplus W_j \)
Further Plan for Building Natural Graph Wavelets

Using $k$-NN connection

- Find $k$-nearest neighbor vectors of $\phi_0$ w.r.t. the ROT distance, which form a space $V_0$ like in the multiresolution analysis.
- For each eigenvector in $V_0$, find its $k$-NN vectors, collect them all to form $V_1 \supset V_0$, which is a space spanned by father wavelets.
- Then set $W_0 := V_1 \setminus V_0$, which is a space spanned by mother wavelets.
- Iterate this process to have $V_{j+1} = V_j \oplus W_j$.

(a) A father wavelet ($j = 2$)

(b) A mother wavelet ($j = 2$)
An Ideal Case: Shannon Wavelets from DCT

- We can generate *Shannon wavelets* from the graph Laplacian eigenvectors of a 1D path (i.e., the DCT-II basis vectors) by simply setting $F_j^0 = \text{diag}(1_{n/2j}, 0_{n-n/2j})$; $F_j^1 = \text{diag}(0_{n/2j}, 1_{n/2j}, 0_{n-n/2j-1})$, and computing $\phi_k^j = \Phi F_j^0 \Phi^T e_k$ (father); $\psi_k^j = \Phi F_j^1 \Phi^T e_k$ (mother).

- Can generate smoother wavelets (e.g., Meyer wavelets) by using a *smoother partition of unity* in the diagonals of $F_j^*$'s.
An Ideal Case: Shannon Wavelets from DCT

- We can generate *Shannon wavelets* from the graph Laplacian eigenvectors of a 1D path (i.e., the DCT-II basis vectors) by simply setting $F^j_0 = \text{diag}(1_{n/2^j}, 0_{n-n/2^j})$; $F^j_1 = \text{diag}(0_{n/2^j}, 1_{n/2^j}, 0_{n-n/2^j-1})$, and computing $\phi^j_k = \Phi F^j_0 \Phi^\top e_k$ (father); $\psi^j_k = \Phi F^j_1 \Phi^\top e_k$ (mother).

**Figure**: From DCT to Shannon wavelets ($j = 3$)

- Can generate smoother wavelets (e.g., Meyer wavelets) by using *smoother partition of unity* in the diagonals of $F^j_*$'s.
An Ideal Case: Shannon Wavelets from DCT

- We can generate *Shannon wavelets* from the graph Laplacian eigenvectors of a 1D path (i.e., the DCT-II basis vectors) by simply setting $F^j_0 = \text{diag}(1_{n/2^j}, 0_{n-n/2^j})$; $F^j_1 = \text{diag}(0_{n/2^j}, 1_{n/2^j}, 0_{n-n/2^{j-1}})$, and computing $\phi^j_k = \Phi F^j_0 \Phi^T e_k$ (father); $\psi^j_k = \Phi F^j_1 \Phi^T e_k$ (mother).

![Figure: From DCT to Shannon wavelets (j = 3)](image)

- Can generate smoother wavelets (e.g., Meyer wavelets) by using *smoother partition of unity* in the diagonals of $F^j_*$'s.
Outline

1 Motivations
2 Measuring Differences between Eigenvectors
3 Numerical Experiments
4 Organizing Laplacian Eigenvectors of Dendritic Trees
5 Other Methods for Eigenvector Organization
6 Further Plan for Building Natural Graph Wavelets
7 Summary
Summary and Future Projects

- Found a *natural* method to order graph Laplacian eigenvectors \( \{\phi_i\}_{i=0:n-1} \) using the transportation cost as their mutual distances based on the ROT theory on a fixed graph.

- How to examine all possible solutions of Eqn. (*) and find the true cost minimizing transportation plan?

- How to find the *sparsest* nonnegative solution of Eqn. (*)?

- How to select the best \( \alpha \in [0,1] \)?

- Which way should we turn \( \phi_i \) into \( p_i \)?

- How to improve computational complexity for solving Eqn. (*)?

- How to choose a good metric among several possibilities (ROT; TSD; DAG; ...)?

- How to proceed to *the Littlewood-Paley theory truly adapted to the graph setting* more precisely like the 1D path case?

- Can apply for comparing more general basis vectors (*not* from eigenvectors) such as those in a wavelet packet dictionary ...
Transportation cost distances in the Haar-Walsh dictionary

(a) GHWT on $P_6$
(b) Dist. from DC
(c) Dist. from 1st Spike
References

Thank you very much for your attention!

Alex Cloninger and I are organizing 3 part minisymposia (12 speakers) on “Distance Metrics and Mass Transfer Between High Dimensional Point Clouds” at ICIAM 2019, Valencia, Spain on July 17, 2019. So, if you plan to participate in ICIAM 2019, please come to our sessions!