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

Acknowledgment

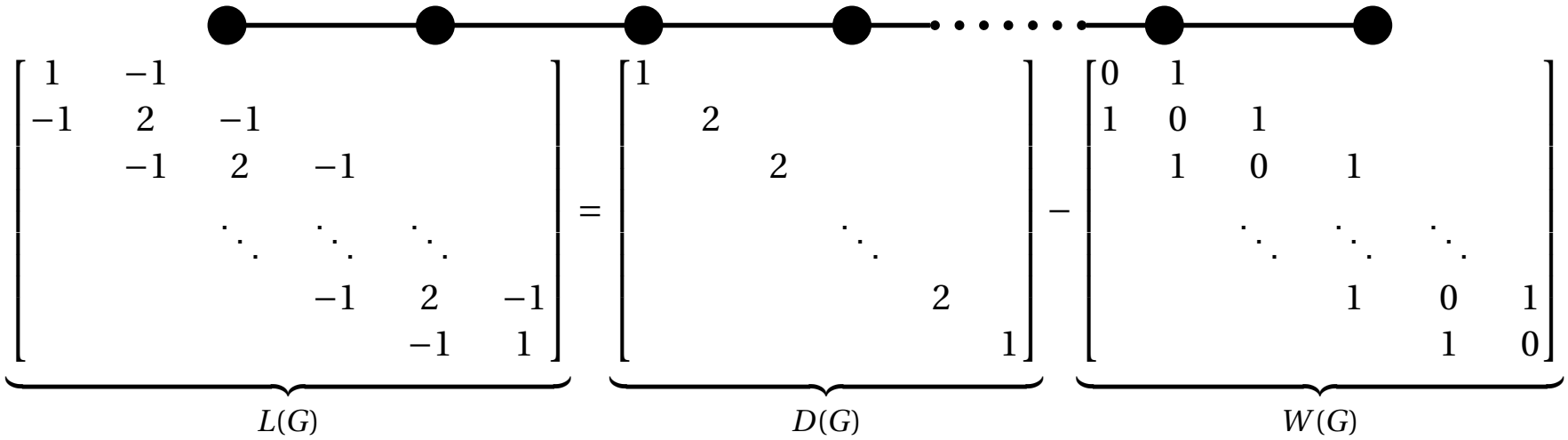
- NSF Grants: DMS-1418779, IIS-1631329
- ONR Grants: N00014-16-1-2255
- Qinglan Xia (UC Davis)

Outline

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).

- $\lambda_k = 2 - 2\cos(\pi k/n) = 4\sin^2(\pi k/2n)$, $k = 0 : n-1$.
- $\boldsymbol{\phi}_k(\ell) = a_{k;n} \cos\left(\pi k \left(\ell + \frac{1}{2}\right)/n\right)$, $k, \ell = 0 : n-1$; $a_{k;n}$ is a const. s.t. $\|\boldsymbol{\phi}_k\|_2 = 1$.
- In this simple case, λ (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.*

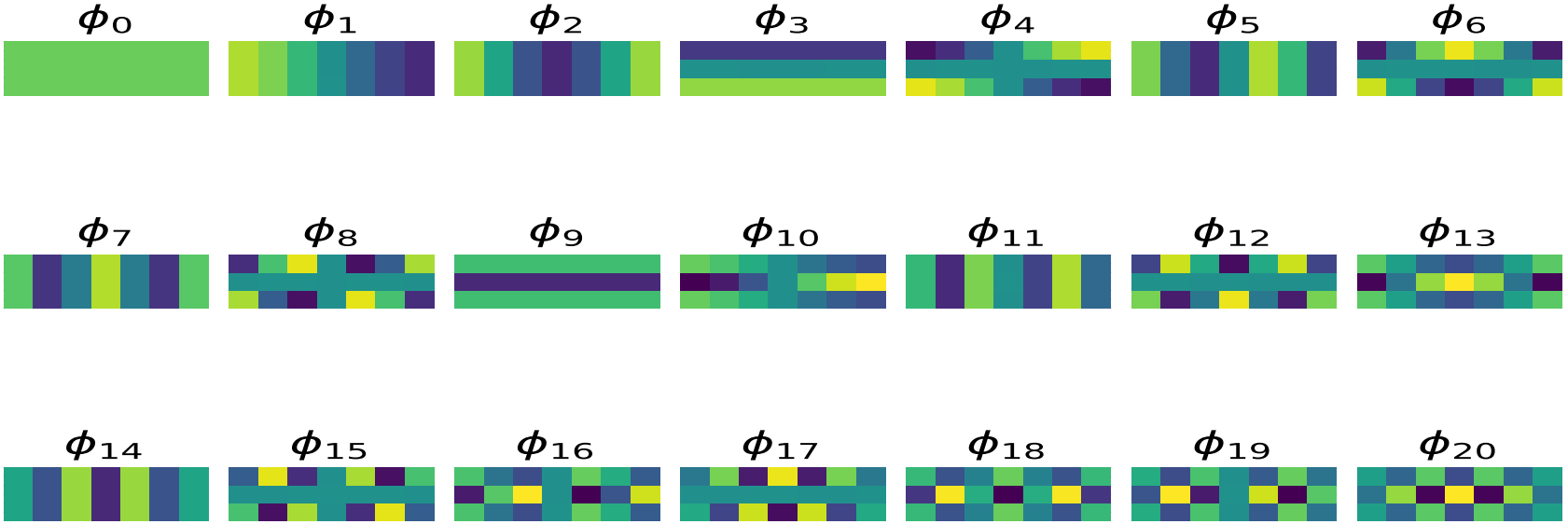
Problem with 2D Lattice Graph

- As soon as the domain becomes *even slightly more complicated than* unweighted and undirected paths/cylces, 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:

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

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

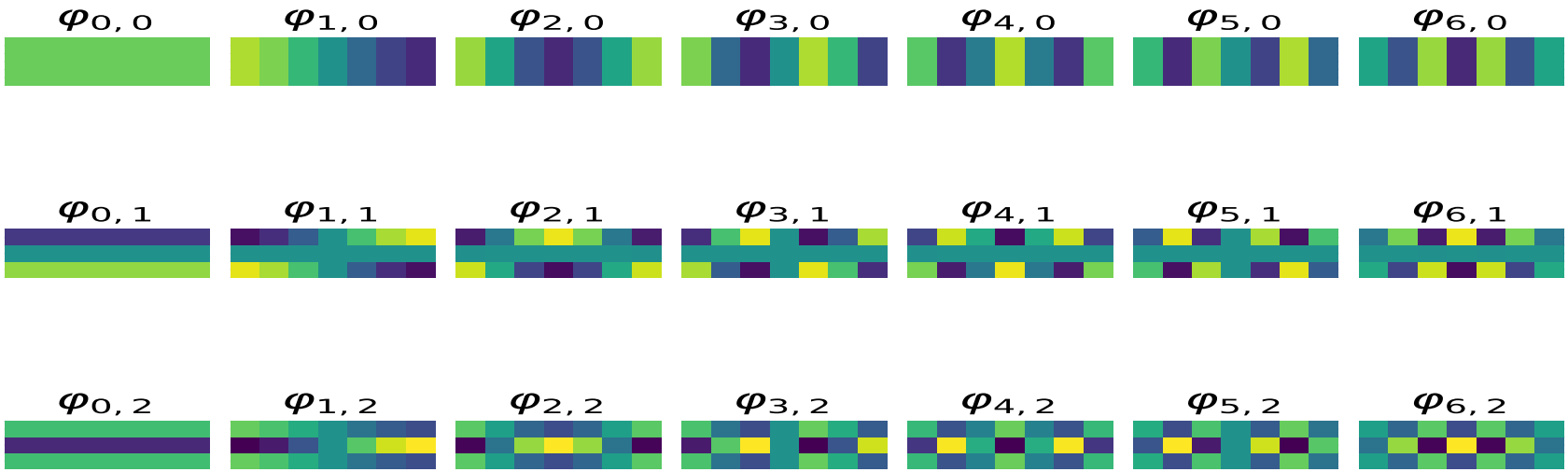
  where $k = 0:mn-1$; $k_x = 0:m-1$; $k_y = 0:n-1$; $x = 0:m-1$; and $y = 0:n-1$.
- As always, let $\{\lambda_k\}_{k=0:mn-1}$ be ordered in the nondecreasing manner. In this case, the smallest eigenvalue is still $\lambda_0 = \lambda_{(0,0)} = 0$, and the corresponding eigenvector is constant.



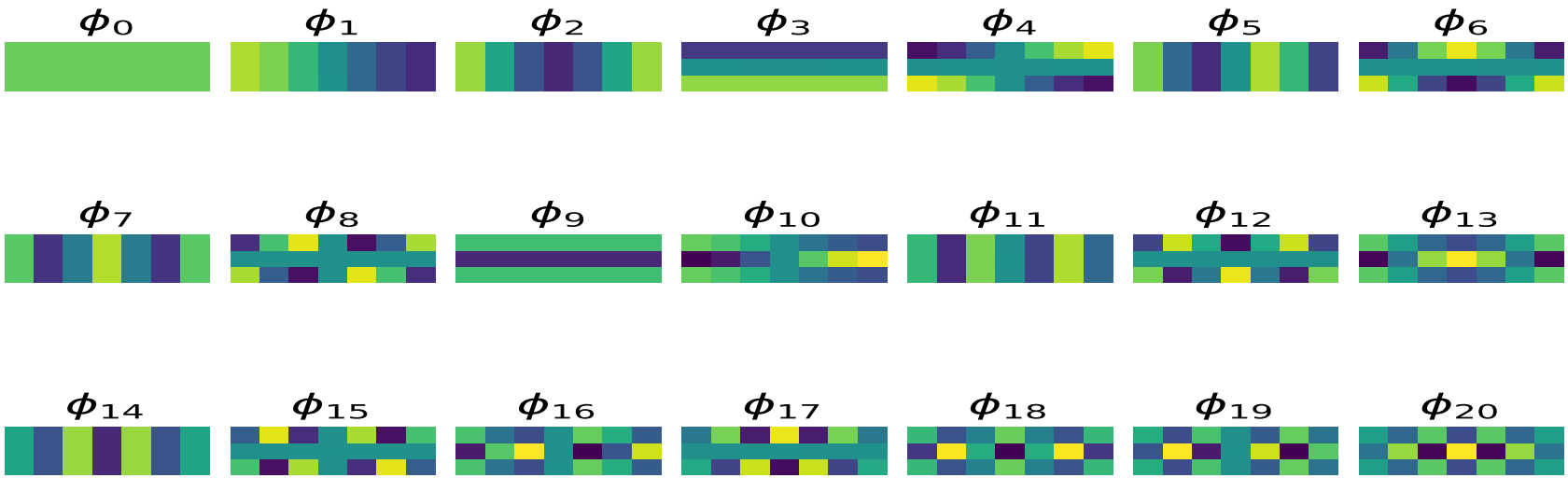
- The second smallest eigenvalue λ_1 is $\lambda_{(1,0)} = 4\sin^2(\pi/2m)$, since $\pi/2m < \pi/2n$, and its eigenvector has half oscillation in the x-direction.
- But, how about λ_2? Even for such a simple situation there are two possibilities: If $m > 2n$, then $\lambda_2 = \lambda_{(2,0)} < \lambda_{(0,1)}$. 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 λ_{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.

- 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 $\{\lambda_k\}_{k=0,1,\ldots}$, 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.

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



instead of



Outline

Plan

- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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 ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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 ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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 ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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 ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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 ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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

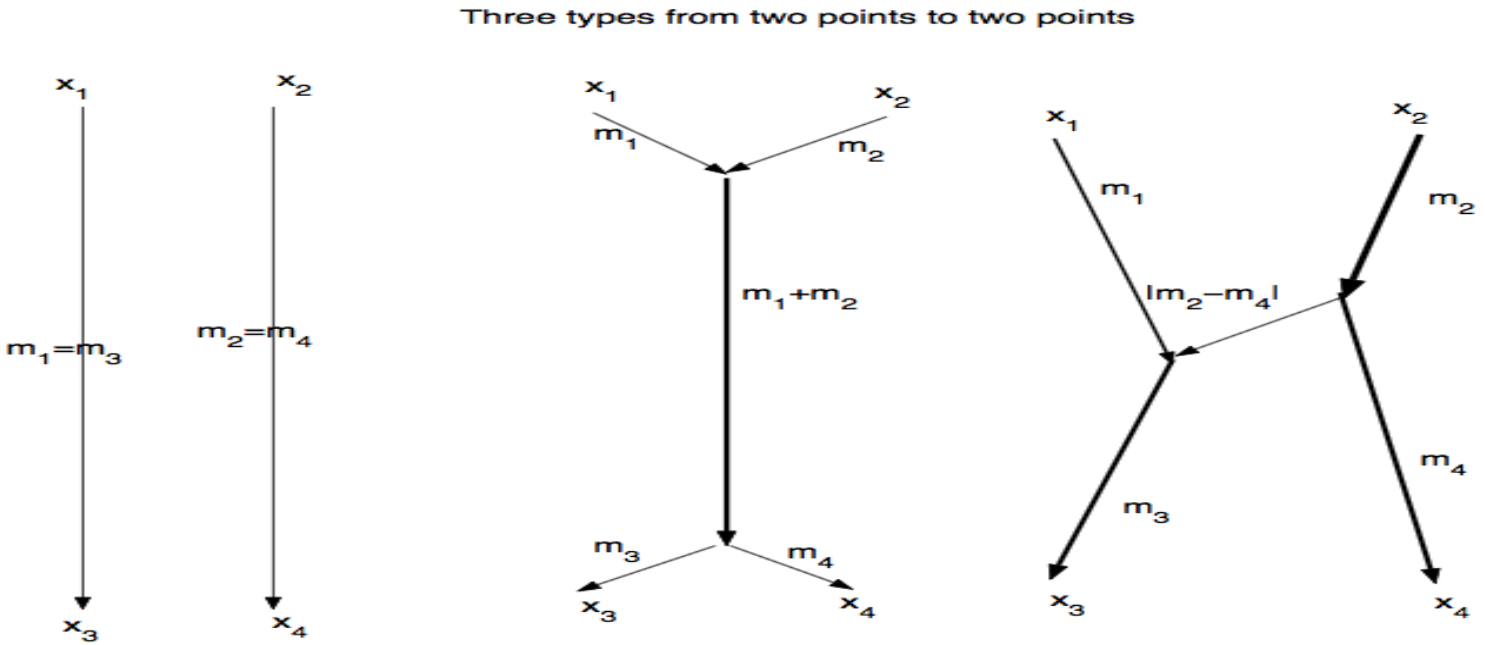
- How can we *quantify the difference between the eigenvectors*?
- The usual ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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 ℓ^2-distance doesn't work since $\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 = \sqrt{2}\delta_{i\neq j}$.
- Consider the *optimal transport theory*!
  - Convert each $\boldsymbol{\phi}_i$ to a probability mass function (pmf) $\boldsymbol{p}_i$ over a graph G (e.g., via squaring each component of $\boldsymbol{\phi}_i$)
  - Compute the cost to transport $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$ 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?

Ramified Optimal Transportation (ROT) by Q. Xia

- is the study of transporting "mass" from one Radon measure (or simply a probability measure) μ^+ to another μ^- 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.

ROT: Discrete Version

- Definitions: Two discrete mass distributions (aka atomic measures) in $\mathbb{R}^d$: $\boldsymbol{a} := \sum_{i=1}^{k} m_i \delta_{\boldsymbol{x}_i}$; $\boldsymbol{b} := \sum_{j=1}^{l} n_j \delta_{\boldsymbol{y}_j}$; $\{\boldsymbol{x}_i\}_i, \{\boldsymbol{y}_j\}_j \subset \mathbb{R}^d$; $\sum_{i=1}^{k} m_i = \sum_{j=1}^{l} n_j$.
- Let $\mathrm{Path}(\boldsymbol{a}, \boldsymbol{b})$ be all possible transport paths from $\boldsymbol{a}$ to $\boldsymbol{b}$ without cycles (Xia could manage to remove cycles), i.e., each $G \in \mathrm{Path}(\boldsymbol{a}, \boldsymbol{b})$ is a weighted acyclic directed graph with $\{\boldsymbol{x}_i\}_i \cup \{\boldsymbol{y}_j\}_j \subset V(G)$, whose edge weights (>0) satisfy *the Kirchhoff law* at each interior node $v \in V(G) \setminus \{\boldsymbol{x}_i, \boldsymbol{y}_j\}_{i,j}$:

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

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

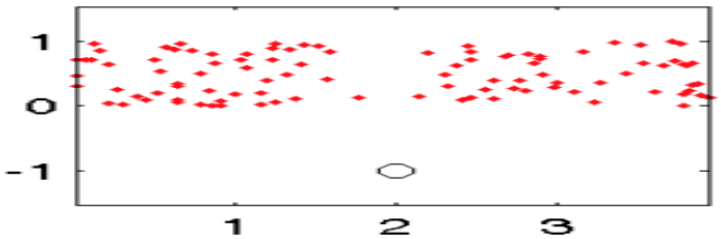
$$\boldsymbol{M}_\alpha(G) := \sum_{e \in E(G)} w(e)^\alpha \, \mathrm{length}(e), \quad \alpha \in [0, 1].$$

ROT: Discrete Version ...

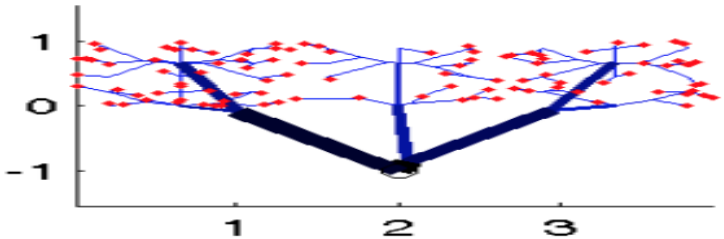
Xia further derived:

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

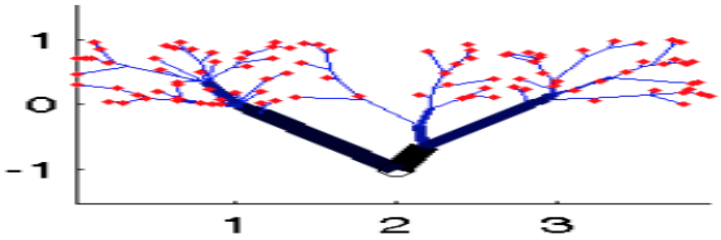
ROT: Numerical Examples



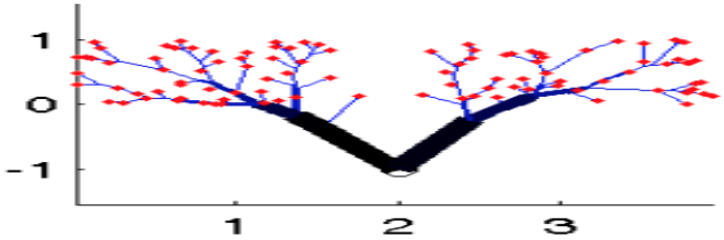
(a)



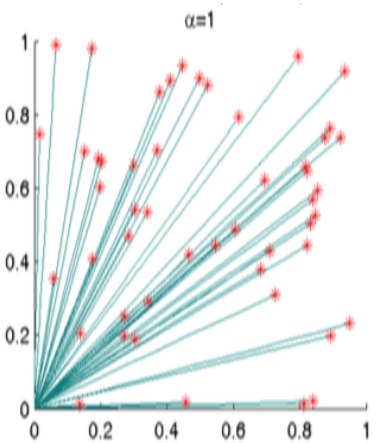
(b)

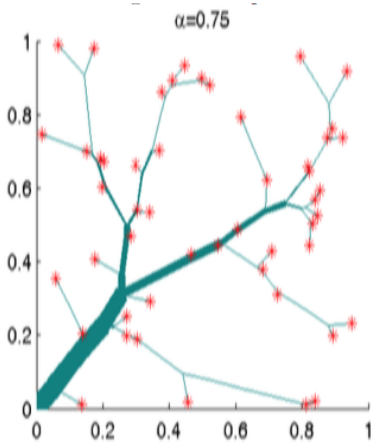


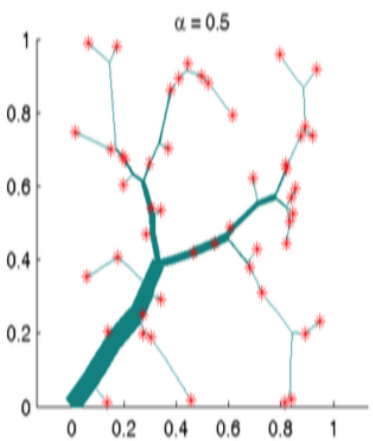
(c)

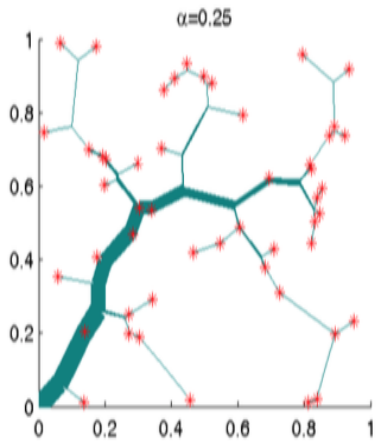


(d)









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 *undirected* graphs.
- The ROT only deals with *directed* graphs.
- Hence, we turn an undirected graph G into the *bidirected* graph $\tilde{\tilde{G}}$.
- To do so, we first compute the *incidence matrix* $Q = [\boldsymbol{q}_1 | \cdots | \boldsymbol{q}_m] \in \mathbb{R}^{n \times m}$ of the undirected graph $G = G(V, E)$ with $n = |V|$, $m = |E|$. Here, $\boldsymbol{q}_k$ represents the endpoints of e_k: if e_k joins nodes i and j, then $\boldsymbol{q}_k[l] = 1$ if $l = i$ or $l = j$; otherwise $\boldsymbol{q}_k[l] = 0$.
- 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.,

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

- Finally, form the bidirected graph $\tilde{\tilde{G}}$ with $\tilde{\tilde{Q}} := [\tilde{Q} \mid -\tilde{Q}] \in \mathbb{R}^{n \times 2m}$.

Our Method to Compute Transportation Costs ...

- Given $\tilde{\tilde{Q}}$, we solve the *balance equation* that forces the Kirchhoff law:

$$\tilde{\tilde{Q}}\boldsymbol{w}_{ij} = \boldsymbol{p}_j - \boldsymbol{p}_i, \quad \boldsymbol{w}_{ij} \in \mathbb{R}^{2m}_{\geq 0}. \qquad (*)$$

- The weight vector $\boldsymbol{w}_{ij}$ describes the transportation plan of mass from $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$, i.e., let $\tilde{\tilde{G}}_{ij}$ be the bidirected graph $\tilde{\tilde{G}}$ with these edge weights; then $\tilde{\tilde{G}}_{ij} \in \mathrm{Path}(\boldsymbol{p}_i, \boldsymbol{p}_j)$.
- Eqn. $(*)$ may have multiple solutions.

Our Method to Compute Transportation Costs ...

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

$$\min_{\boldsymbol{w}_{ij}\in\mathbb{R}^{2m}} \|\boldsymbol{w}_{ij}\|_1 \quad \text{subject to: } \tilde{\tilde{Q}}\boldsymbol{w}_{ij} = \boldsymbol{p}_j - \boldsymbol{p}_i; \boldsymbol{w}_{ij}[l] \geq 0, l = 0 : (2m-1)$$

  to obtain one of the *sparse* solutions of Eqn. $(*)$, which turned out to be better than using nonnegative least squares (NNLS) solver.

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

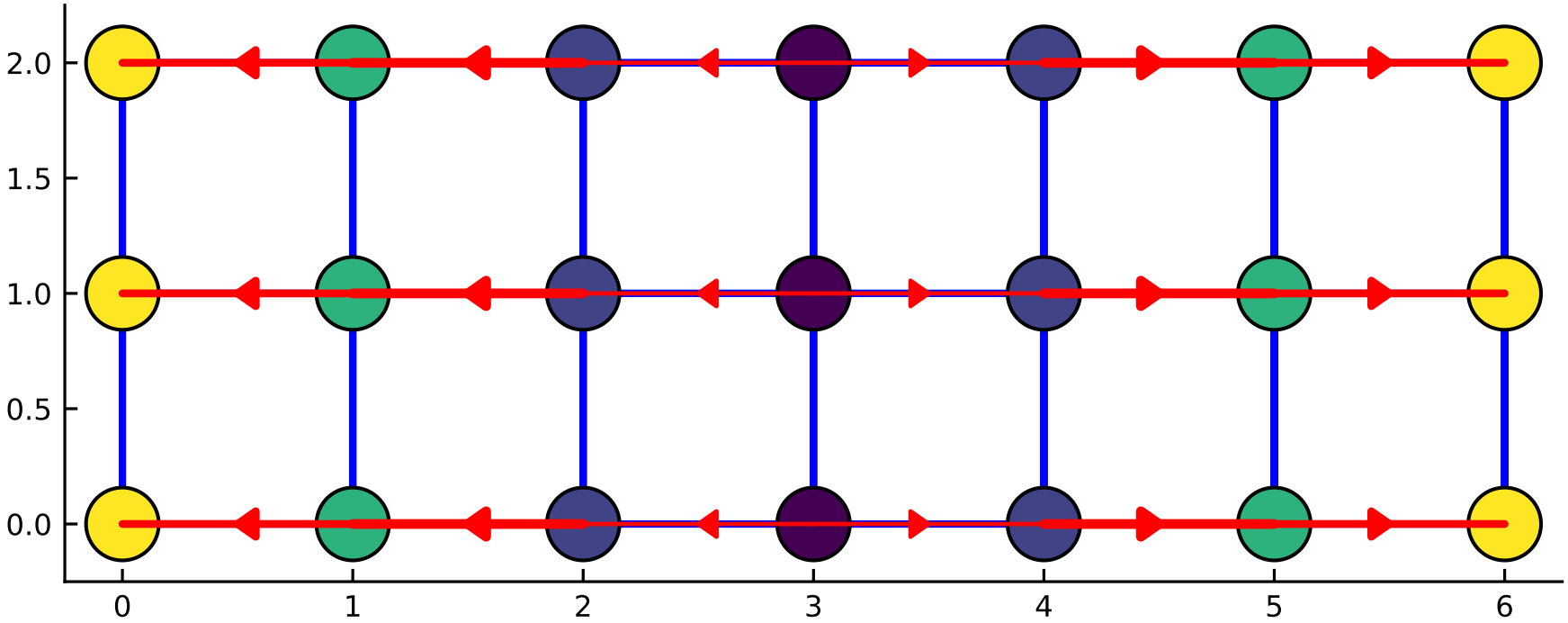
$$D_{ij} = \boldsymbol{M}_\alpha(\tilde{\tilde{G}}_{ij}) = \sum_{e\in E(\tilde{\tilde{G}}_{ij})} w_{ij}(e)^\alpha \,\text{length}(e), \quad \alpha \in [0,1].$$

- Note that currently we are *not* examining all possible solutions of Eqn. $(*)$ to search $\arg \min_{\tilde{\tilde{G}}_{ij}\in\text{Path}(\boldsymbol{p}_i,\boldsymbol{p}_j)} \boldsymbol{M}_\alpha(\tilde{\tilde{G}}_{ij})$.

Outline

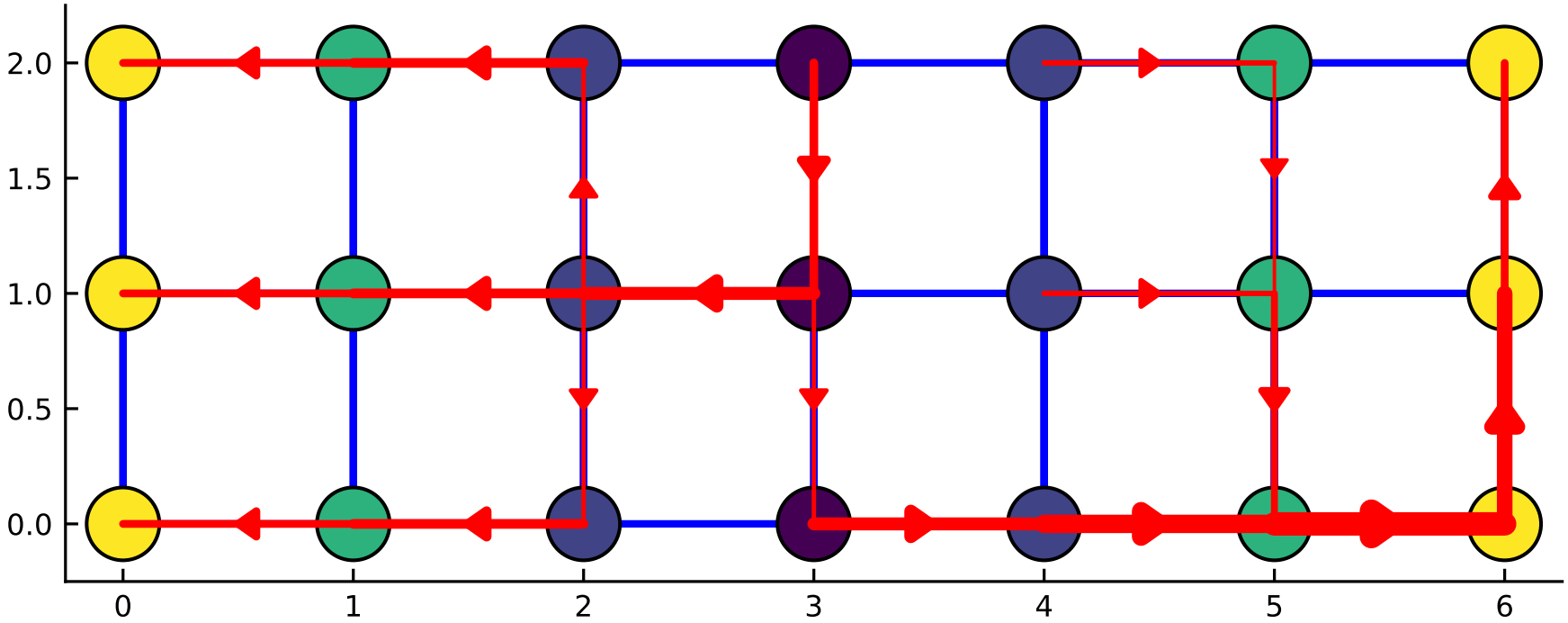
2D Regular Lattice: An LP Solution to (∗)

Consolidated $\boldsymbol{w}_{0,1}$: mass transport from $\boldsymbol{p}_0 = \boldsymbol{\phi}_0^2$ to $\boldsymbol{p}_1 = \boldsymbol{\phi}_1^2$

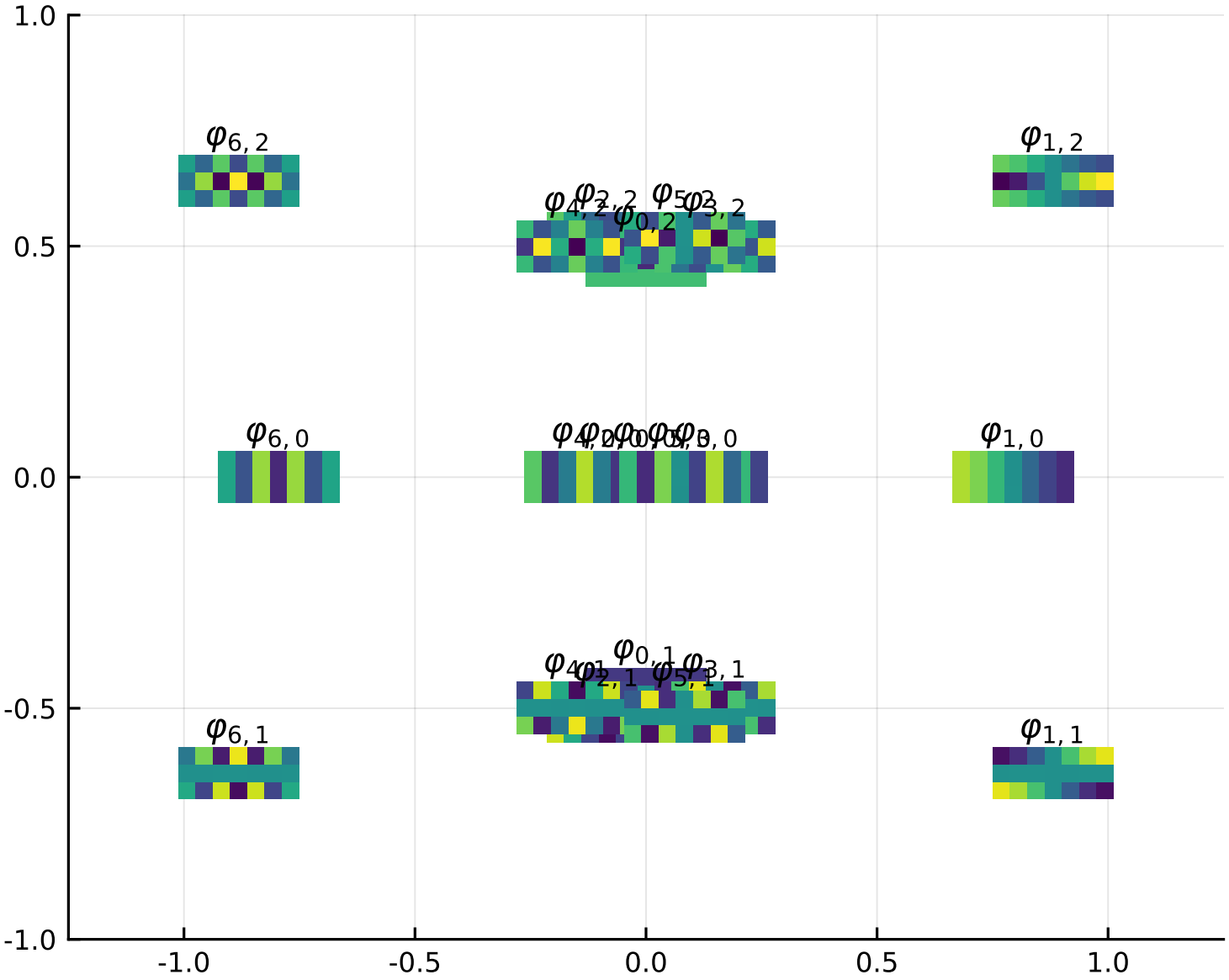


2D Regular Lattice: An NNLS Solution to (∗)

Consolidated $\boldsymbol{w}_{0,1}$: mass transport from $\boldsymbol{p}_0 = \boldsymbol{\phi}_0^2$ to $\boldsymbol{p}_1 = \boldsymbol{\phi}_1^2$



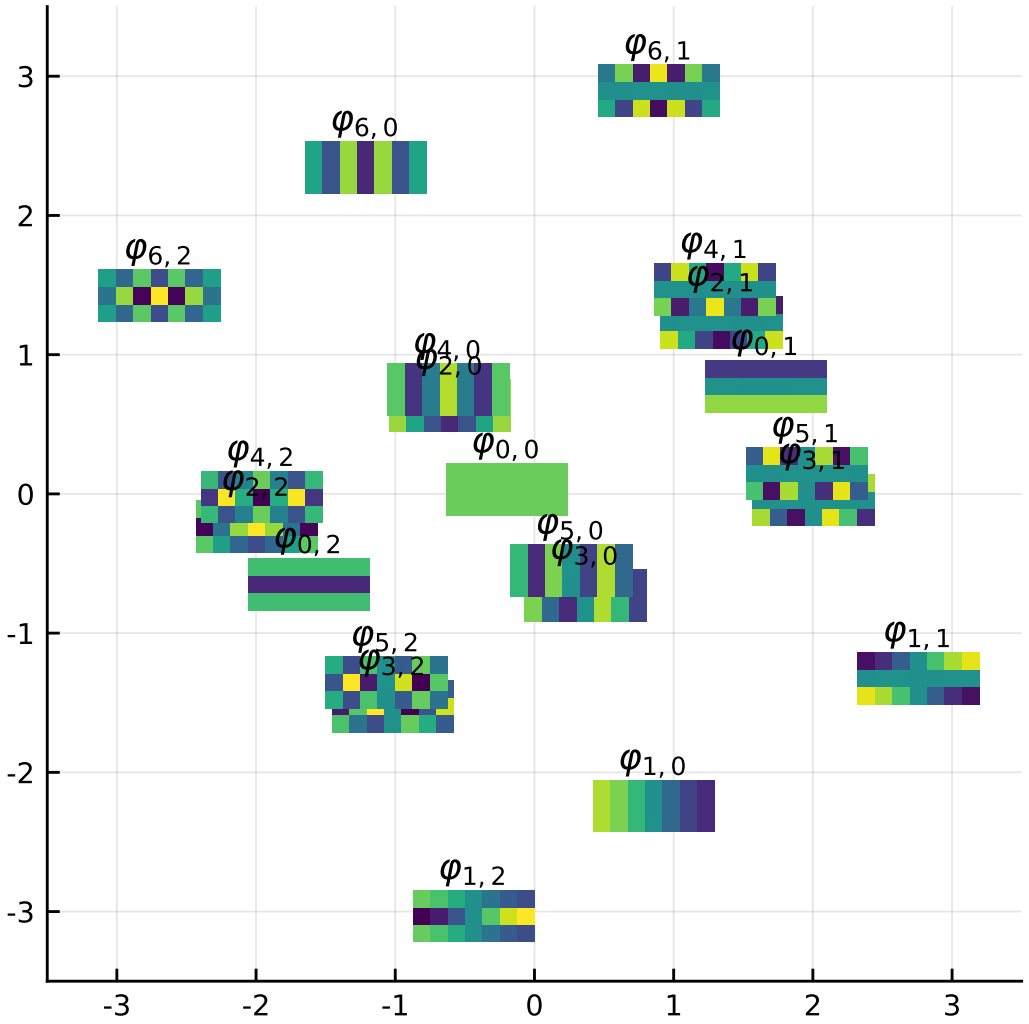
2D Regular Lattice: Embedding into $\mathbb{R}^2$; $\alpha = 1$



2D Regular Lattice: Embedding into $\mathbb{R}^2$; $\alpha = 0.5$

Some symmetry could be explained because of the symmetry of DCT vectors:

$$\phi^2_{k;n}[x] + \phi^2_{n-k;n}[x] \equiv a^2_{k;n} = 2/n,\ k = 1 : n-1,\ x = 0 : n-1.$$



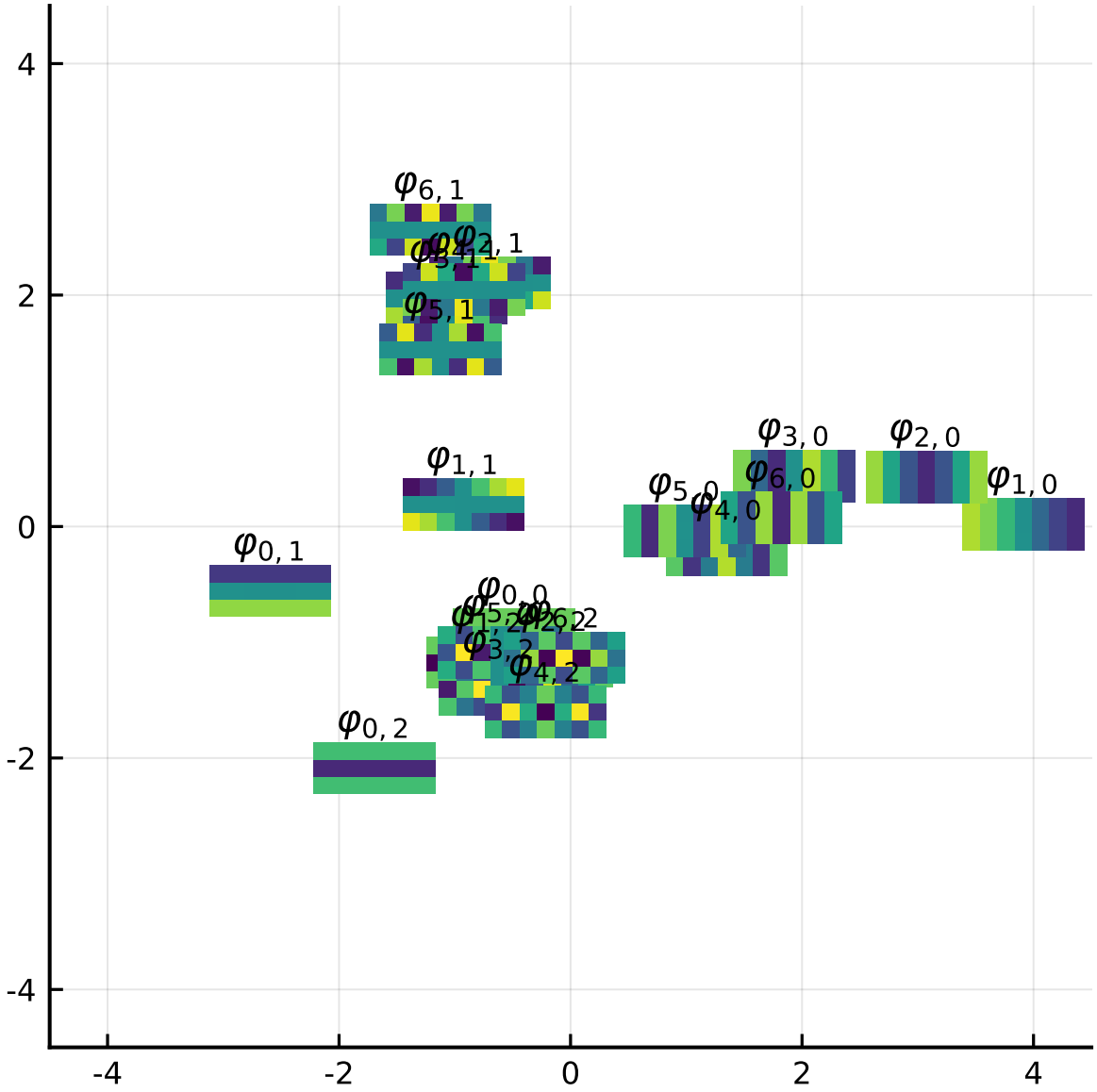
Other Ways to Turn $\boldsymbol{\phi}_i$ into $\boldsymbol{p}_i$

- Generating $\boldsymbol{\phi}_i^2$ is not the only way to turn $\boldsymbol{\phi}_i$ into a pmf $\boldsymbol{p}_i$.
- Other examples include:
  - Normalized ℓ^1: $\boldsymbol{\phi}_i^1 := \big(\big|\phi_i[0]\big|, \ldots, \big|\phi_i[n-1]\big|\big)^{\mathsf{T}} / \|\boldsymbol{\phi}_i\|_1$;
  - A constant addition followed by normalization:

$$\widetilde{\boldsymbol{\phi}}_i := \begin{cases} \boldsymbol{\phi}_0^1 & \text{if } i = 0; \\ \frac{\boldsymbol{\phi}_i - c_{\min} \cdot \mathbf{1}_n}{\|\boldsymbol{\phi}_i - c_{\min} \cdot \mathbf{1}_n\|_1} & \text{if } i \neq 0, \end{cases}$$

    where $c_{\min} := \min_{0<i<n; 0 \le l < n} \phi_i[l] < 0$;
  - Normalized exponentiation: $\boldsymbol{\phi}_i^{\mathrm{e}} := \exp(\boldsymbol{\phi}_i) / \|\exp(\boldsymbol{\phi}_i)\|_1$.

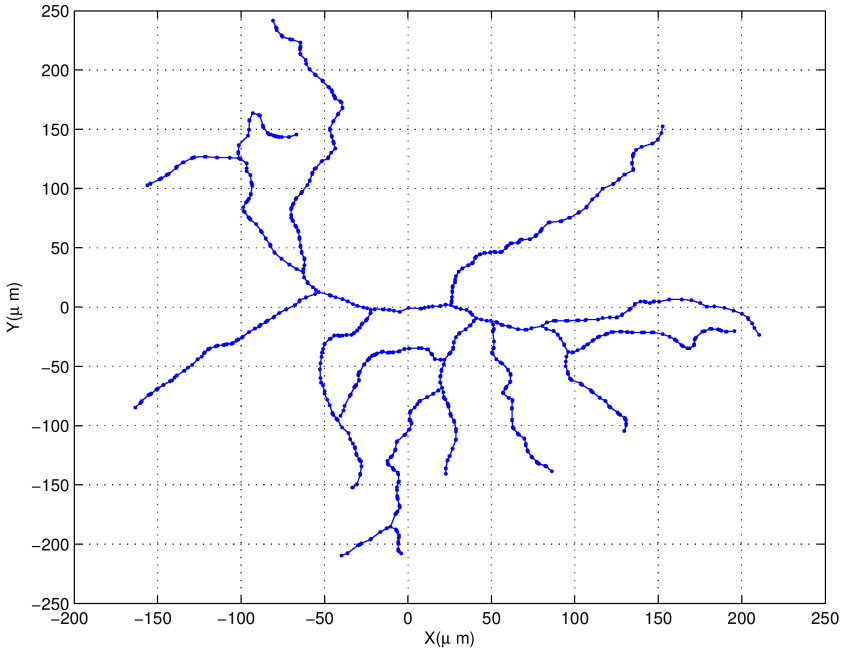
2D Regular Lattice; via $\{\boldsymbol{\phi}_i^{\mathrm{e}}\}_i$, $\alpha = 0.25$



Outline

A Peculiar Phase Transition Phenomenon

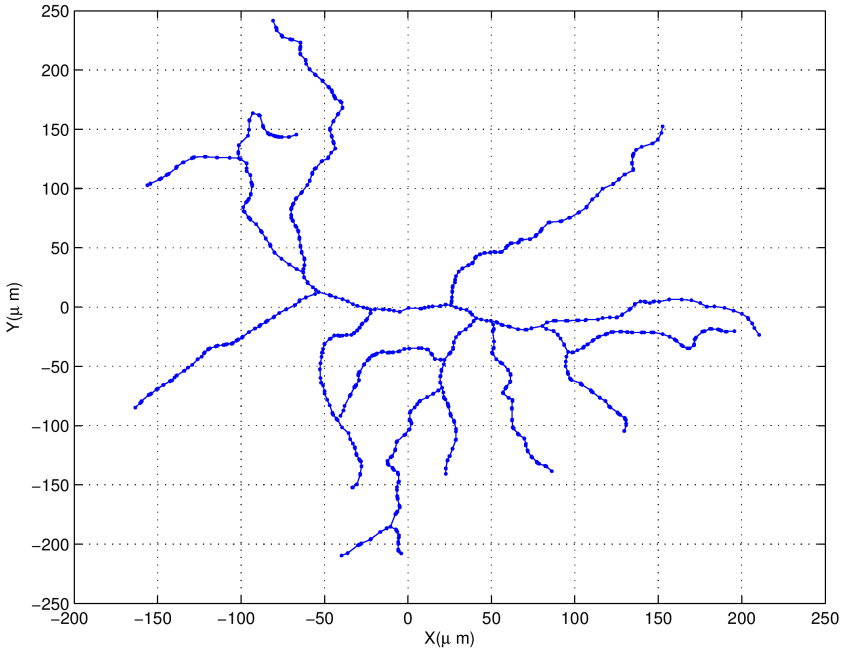
We observed an interesting phase transition phenomenon on the behavior of the eigenvalues of *graph Laplacians* defined on dendritic trees.



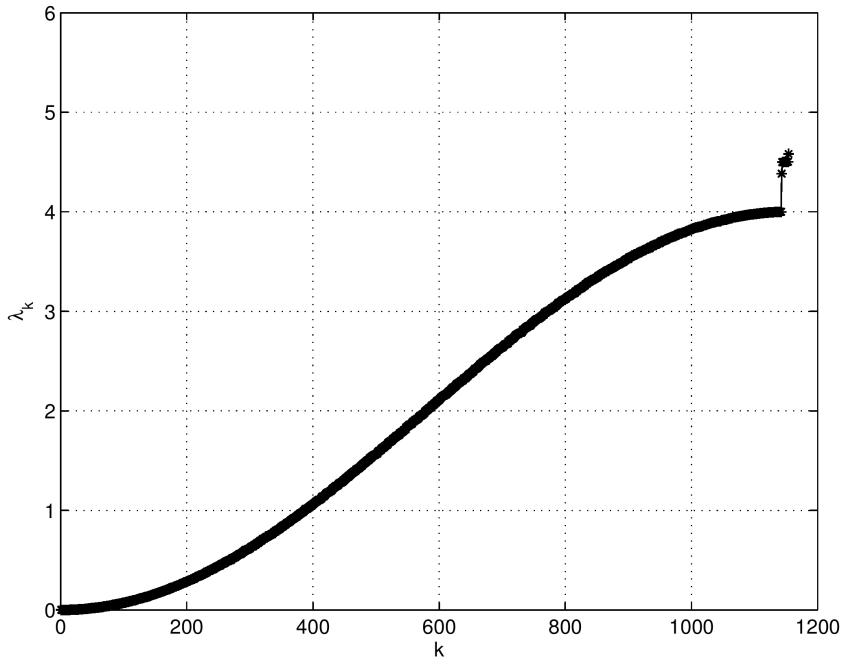
(a) RGC #100

A Peculiar Phase Transition Phenomenon

We observed an interesting phase transition phenomenon on the behavior of the eigenvalues of *graph Laplacians* defined on dendritic trees.



(a) RGC #100



(b) Eigenvalues of RGC #100

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 ...

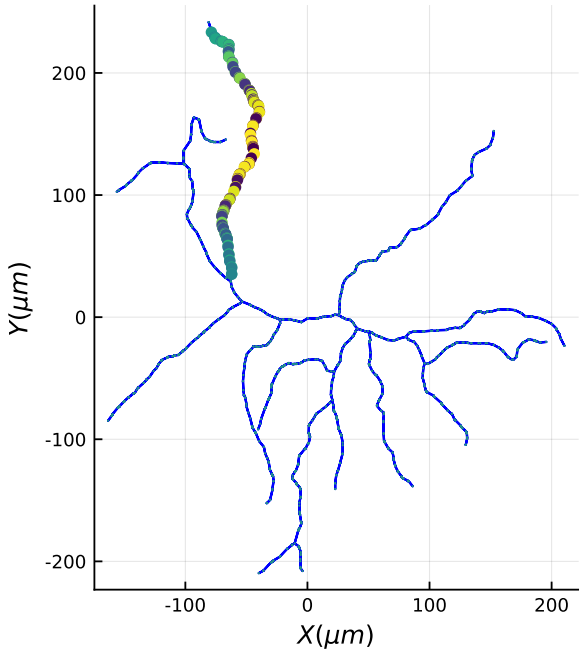
We have observed that this value 4 is critical since:

- 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 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*.

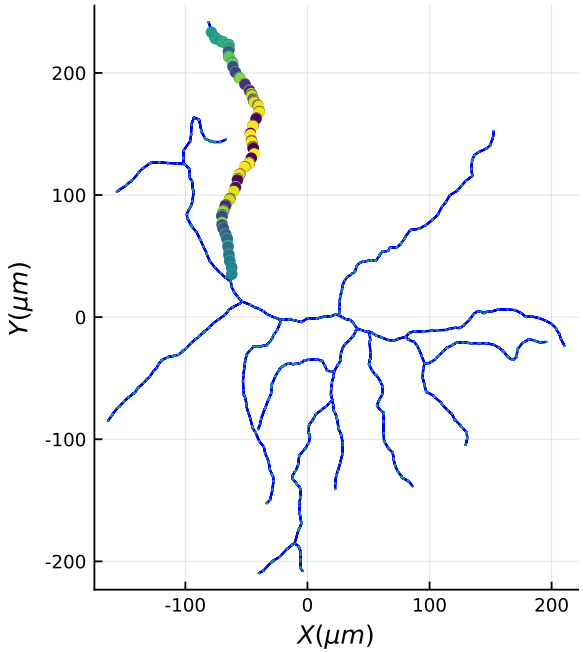


(a) RGC #100; $\lambda_{1141} = 3.9994$

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*.

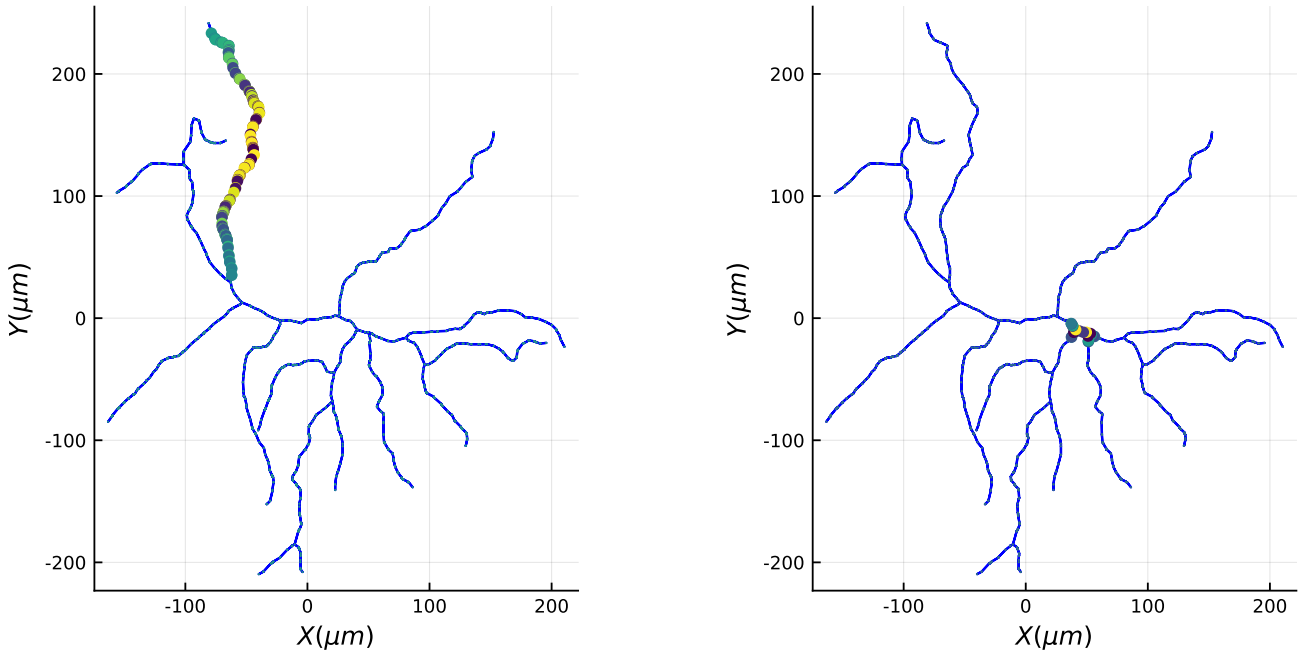


(a) RGC #100; $\lambda_{1141} = 3.9994$

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; $\lambda_{1141} = 3.9994$ (b) RGC #100; $\lambda_{1142} = 4.3829$

- We know why such localization/phase transition occurs $\Longrightarrow$ 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, ...
- See also an interesting related work for more general setting and for application in numerical linear algebra: I. Krishtal, T. Strohmer, & T. Wertz: "Localization of matrix factorizations," *Foundations of Comp. Math.*, vol. 15, no. 4, pp. 931–951, 2015.
- 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 $\mathbb{R}^3$

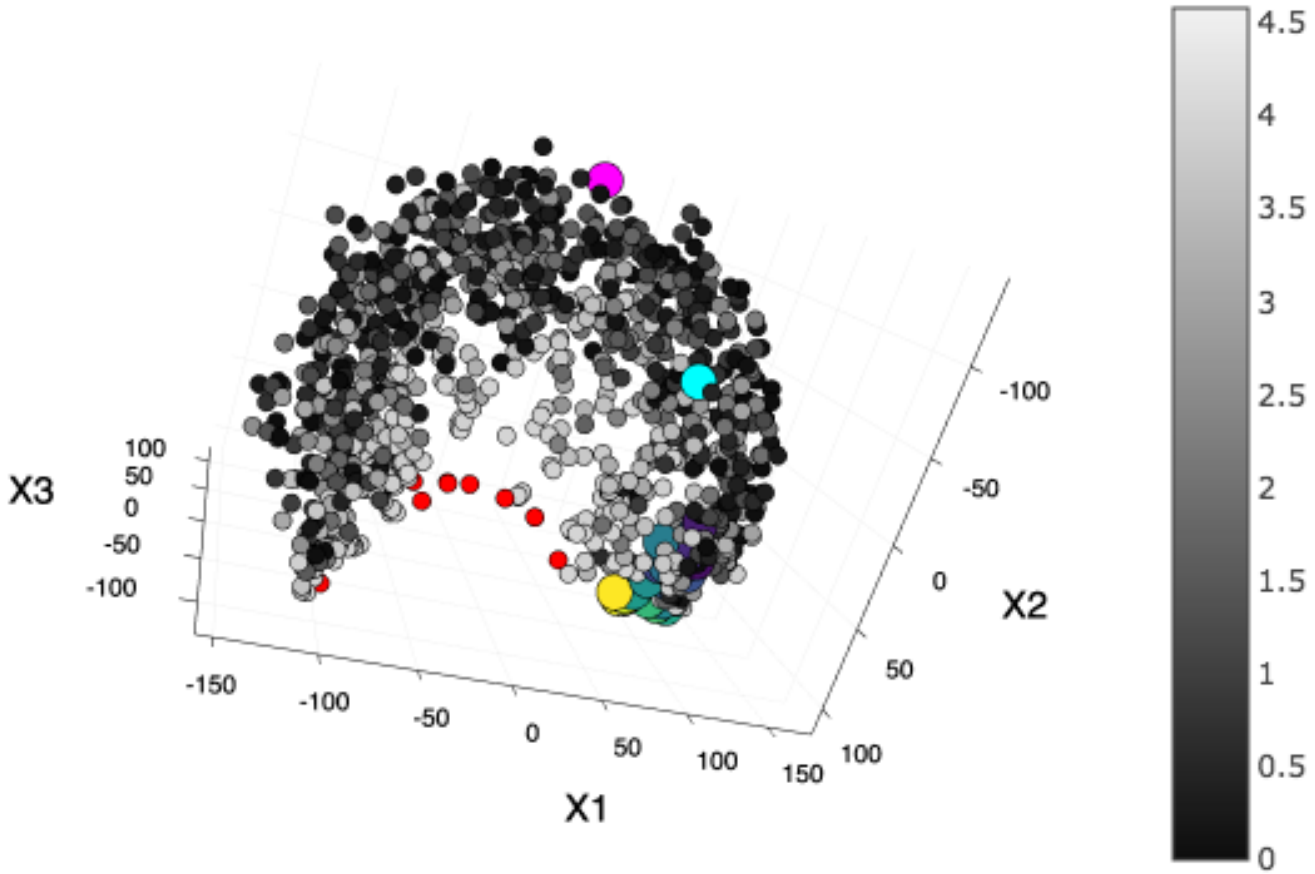


Figure: The magenta circle = the DC vector; the cyan circle = the Fiedler vector; the red circles = the localized eigenvectors; the larger colored circles = the eigenvectors supported on the upper-left branch

Outline

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

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, $\mathscr{P} = \{\mathscr{C}_1, \mathscr{C}_2, \cdots, \mathscr{C}_N\}$, $1 \le N \le n$, in which

$$\bigcup_{j=1}^{N} \mathscr{C}_j = \{1, 2, \cdots, n\} \quad \text{and} \quad \mathscr{C}_i \cap \mathscr{C}_j = \emptyset, i \neq j.$$

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

$$\boldsymbol{\psi}_{k,j} = \Phi \overbrace{F_j \Phi^{\mathsf{T}} \boldsymbol{e}_k}^{\text{Filtering}} \quad \text{for } j = 1, 2, \cdots, N \text{ and } k = 1, 2, \cdots, n$$

  in which, the diagonal matrix $F_j \in \mathbb{R}^{n \times n}$ satisfies $F_j(l, l) = \chi_{\mathscr{C}_j}(l)$ for $l = 1, 2, \cdots, n$, Φ stores all the graph Laplacian eigenvectors, and $\boldsymbol{e}_k$ is the canonical basis vector at vertex v_k.
- We can show that $\{\boldsymbol{\psi}_{k,j}\}_{k=1,\cdots,n;j=1,\cdots,N}$ *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 $\{\boldsymbol{\psi}_{k,j}\}$ by using $\{\boldsymbol{e}_k\}_{k\in T_j}$, in which $T_j \subset \{1,2,\cdots,n\}$, instead of all $\{\boldsymbol{e}_k\}_{k=1,\cdots,n}$ for each $\mathscr{C}_j$, 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 Φ into $\hat{\Phi}$ based on $\mathscr{P}$: $\hat{\Phi} = [\boldsymbol{\phi}_{\mathscr{C}_1}, \boldsymbol{\phi}_{\mathscr{C}_2}, \cdots, \boldsymbol{\phi}_{\mathscr{C}_N}]$ in which $\mathscr{C}_j = \{j_1, j_2, \cdots, j_l\}$ and $\boldsymbol{\phi}_{\mathscr{C}_j} = [\boldsymbol{\phi}_{j_1}, \boldsymbol{\phi}_{j_2}, \cdots, \boldsymbol{\phi}_{j_l}]$. Then, we rotate $\hat{\Phi}$ within each cluster $\mathscr{C}_j$ 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 $\{\boldsymbol{\psi}_{k,j}\}$ by using $\{\boldsymbol{e}_k\}_{k\in T_j}$, in which $T_j \subset \{1,2,\cdots,n\}$, instead of all $\{\boldsymbol{e}_k\}_{k=1,\cdots,n}$ for each $\mathscr{C}_j$, 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 Φ into $\hat{\Phi}$ based on $\mathscr{P}$: $\hat{\Phi} = [\boldsymbol{\phi}_{\mathscr{C}_1}, \boldsymbol{\phi}_{\mathscr{C}_2}, \cdots, \boldsymbol{\phi}_{\mathscr{C}_N}]$ in which $\mathscr{C}_j = \{j_1, j_2, \cdots, j_l\}$ and $\boldsymbol{\phi}_{\mathscr{C}_j} = [\boldsymbol{\phi}_{j_1}, \boldsymbol{\phi}_{j_2}, \cdots, \boldsymbol{\phi}_{j_l}]$. Then, we rotate $\hat{\Phi}$ within each cluster $\mathscr{C}_j$ 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 $\{\boldsymbol{\psi}_{k,j}\}$ by using $\{\boldsymbol{e}_k\}_{k\in T_j}$, in which $T_j \subset \{1,2,\cdots,n\}$, instead of all $\{\boldsymbol{e}_k\}_{k=1,\cdots,n}$ for each $\mathscr{C}_j$, 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 Φ into $\hat{\Phi}$ based on $\mathscr{P}$: $\hat{\Phi} = [\boldsymbol{\phi}_{\mathscr{C}_1}, \boldsymbol{\phi}_{\mathscr{C}_2}, \cdots, \boldsymbol{\phi}_{\mathscr{C}_N}]$ in which $\mathscr{C}_j = \{j_1, j_2, \cdots, j_l\}$ and $\boldsymbol{\phi}_{\mathscr{C}_j} = [\boldsymbol{\phi}_{j_1}, \boldsymbol{\phi}_{j_2}, \cdots, \boldsymbol{\phi}_{j_l}]$. Then, we rotate $\hat{\Phi}$ within each cluster $\mathscr{C}_j$ 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 $\{\boldsymbol{\psi}_{k,j}\}$ by using $\{\boldsymbol{e}_k\}_{k\in T_j}$, in which $T_j \subset \{1,2,\cdots,n\}$, instead of all $\{\boldsymbol{e}_k\}_{k=1,\cdots,n}$ for each $\mathscr{C}_j$, 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 Φ into $\hat{\Phi}$ based on $\mathscr{P}$: $\hat{\Phi} = [\boldsymbol{\phi}_{\mathscr{C}_1}, \boldsymbol{\phi}_{\mathscr{C}_2}, \cdots, \boldsymbol{\phi}_{\mathscr{C}_N}]$ in which $\mathscr{C}_j = \{j_1, j_2, \cdots, j_l\}$ and $\boldsymbol{\phi}_{\mathscr{C}_j} = [\boldsymbol{\phi}_{j_1}, \boldsymbol{\phi}_{j_2}, \cdots, \boldsymbol{\phi}_{j_l}]$. Then, we rotate $\hat{\Phi}$ within each cluster $\mathscr{C}_j$ 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 $\boldsymbol{\phi}_0$ 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 $\boldsymbol{\phi}_0$ 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

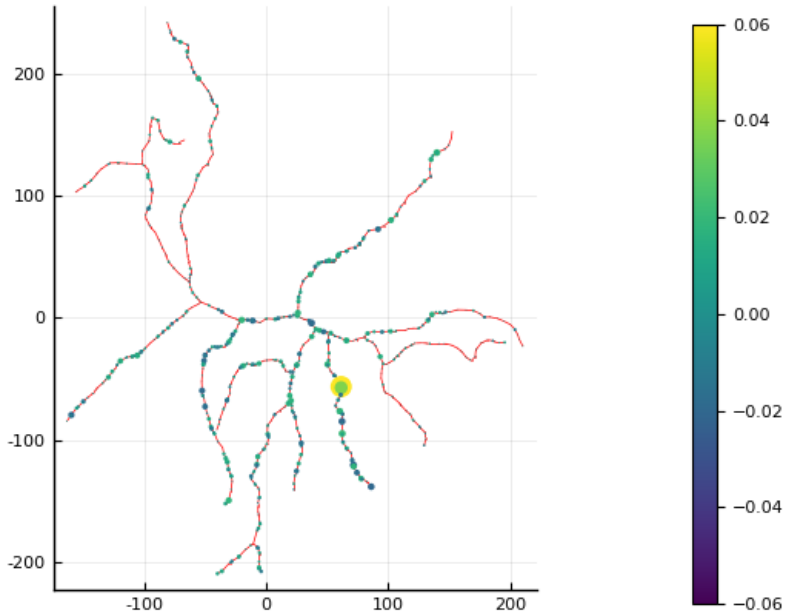
- Find k-nearest neighbor vectors of $\boldsymbol{\phi}_0$ w.r.t. the ROT distance, which form a space V_0 like in the *multiresolution analysis*
- For each eigenvector in V_0, find its k-NN vectors, collect them all to form $V_1 \supset V_0$, which is a space spanned by *father* wavelets
- Then set $W_0 := V_1 \setminus V_0$, 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

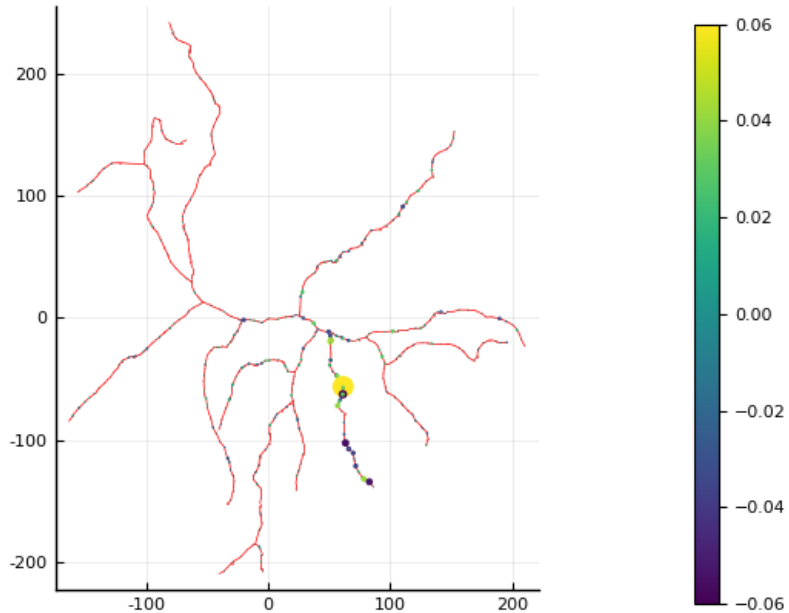
- Find k-nearest neighbor vectors of $\boldsymbol{\phi}_0$ w.r.t. the ROT distance, which form a space V_0 like in the *multiresolution analysis*
- For each eigenvector in V_0, find its k-NN vectors, collect them all to form $V_1 \supset V_0$, which is a space spanned by *father* wavelets
- Then set $W_0 := V_1 \setminus V_0$, 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 $\boldsymbol{\phi}_0$ 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 $F_0^j = \mathrm{diag}(\mathbf{1}_{n/2^j}, \mathbf{0}_{n-n/2^j})$; $F_1^j = \mathrm{diag}(\mathbf{0}_{n/2^j}, \mathbf{1}_{n/2^j}, \mathbf{0}_{n-n/2^{j-1}})$, and computing $\boldsymbol{\phi}_k^j = \Phi F_0^j \Phi^\top \boldsymbol{e}_k$ (father); $\boldsymbol{\psi}_k^j = \Phi F_1^j \Phi^\top \boldsymbol{e}_k$ (mother).

- 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 $F_0^j = \mathrm{diag}(\mathbf{1}_{n/2^j}, \mathbf{0}_{n-n/2^j})$; $F_1^j = \mathrm{diag}(\mathbf{0}_{n/2^j}, \mathbf{1}_{n/2^j}, \mathbf{0}_{n-n/2^{j-1}})$, and computing $\boldsymbol{\phi}_k^j = \Phi F_0^j \Phi^\top \boldsymbol{e}_k$ (father); $\boldsymbol{\psi}_k^j = \Phi F_1^j \Phi^\top \boldsymbol{e}_k$ (mother).

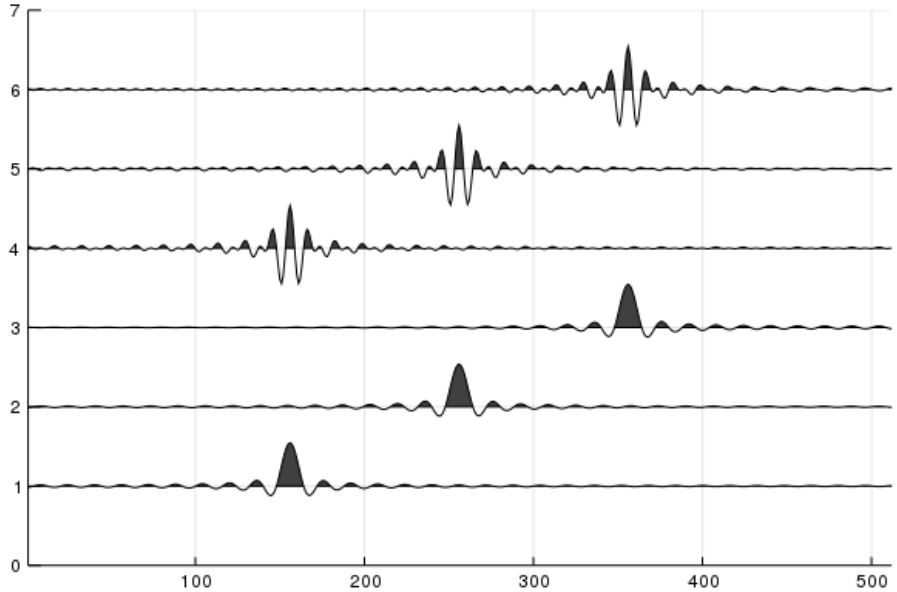


Figure: From DCT to Shannon wavelets ($j = 3$)

- 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 $F_0^j = \mathrm{diag}(\mathbf{1}_{n/2^j}, \mathbf{0}_{n-n/2^j})$; $F_1^j = \mathrm{diag}(\mathbf{0}_{n/2^j}, \mathbf{1}_{n/2^j}, \mathbf{0}_{n-n/2^{j-1}})$, and computing $\boldsymbol{\phi}_k^j = \Phi F_0^j \Phi^\top \boldsymbol{e}_k$ (father); $\boldsymbol{\psi}_k^j = \Phi F_1^j \Phi^\top \boldsymbol{e}_k$ (mother).

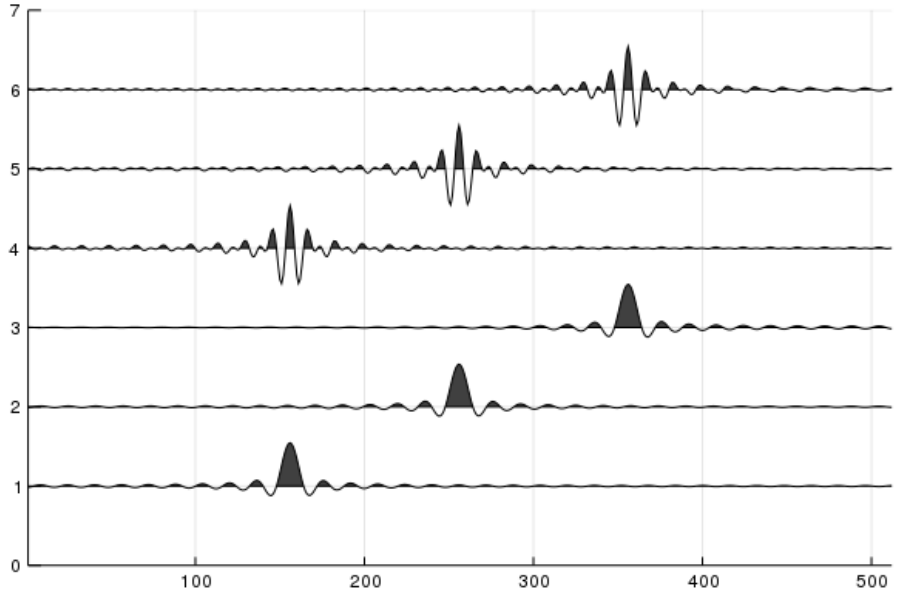


Figure: From DCT to Shannon wavelets ($j = 3$)

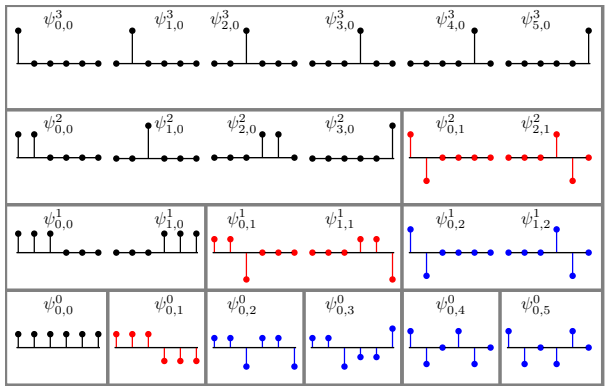
- Can generate smoother wavelets (e.g., Meyer wavelets) by using *smoother partition of unity* in the diagonals of F_*^j's

Outline

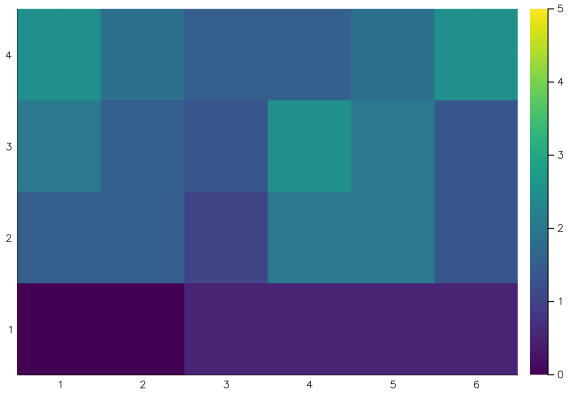
Summary and Future Projects

- Found a *natural* method to order graph Laplacian eigenvectors $\{\boldsymbol{\phi}_i\}_{i=0:n-1}$ 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 $\boldsymbol{\phi}_i$ into $\boldsymbol{p}_i$?
- 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 ...

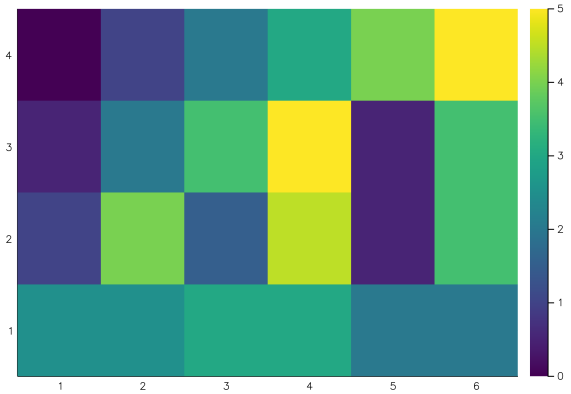
Transportation cost distances in the Haar-Walsh dictionary



(a) GHWT on P_6



(b) Dist. from DC



(c) Dist. from 1st Spike

References

- N. Saito & E. Woei: "Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians," *Japan SIAM Letters*, vol. 1, pp. 13–16, 2009.
- N. Saito & E. Woei: "On the phase transition phenomenon of graph Laplacian eigenfunctions on trees," *RIMS Kôkyûroku*, vol. 1743, pp. 77–90, 2011.
- Y. Nakatsukasa, N. Saito, & E. Woei: "Mysteries around graph Laplacian eigenvalue 4," *Linear Algebra Appl.*, vol. 438, no. 8, pp. 3231–3246, 2013.
- J. Irion & N. Saito: "The generalized Haar-Walsh transform," in *Proc. 2014 IEEE Workshop on Statistical Signal Processing*, pp. 472–475, 2014.
- J. Irion & N. Saito: "Applied and computational harmonic analysis on graphs and networks," in *Wavelets and Sparsity XVI, Proc. SPIE 9597*, Paper # 95971F, 2015.
- J. Irion & N. Saito: "Efficient approximation and denoising of graph signals using the multiscale basis dictionaries," *IEEE Trans. Signal and Inform. Process. Netw.*, vol. 3, no. 3, pp. 607–616, 2017.
- N. Saito: "How can we naturally order and organize graph Laplacian eigenvectors?" in *Proc. 2018 IEEE Workshop on Statistical Signal Processing*, pp. 483–487, 2018. Also ArXiv: 1801.06782 [math.SP].
- H. Li & N. Saito: "Metrics of graph Laplacian eigenvectors," in preparation, 2019.

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!