Wavelets and Filter Banks on Graphs

Pierre Vandergheynst

Signal Processing Lab, EPFL

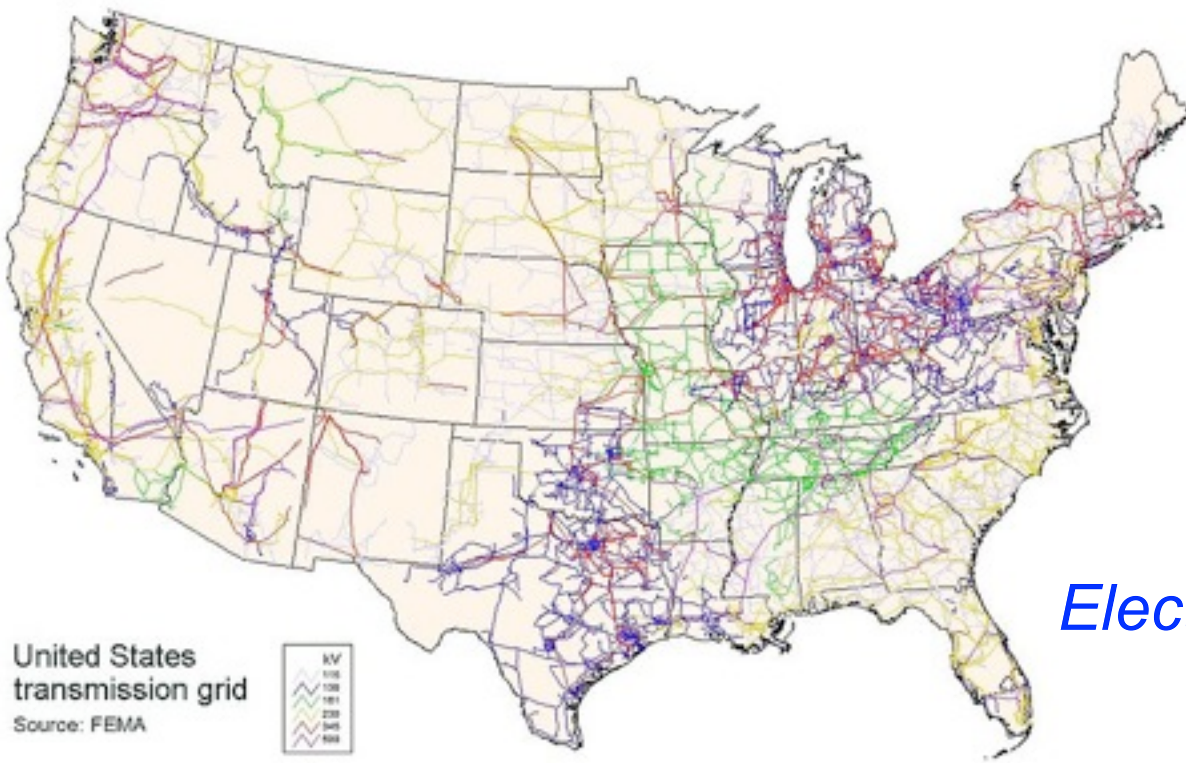
Joint work with David Shuman

Duke Workshop on Sensing and Analysis of High-Dimensional Data

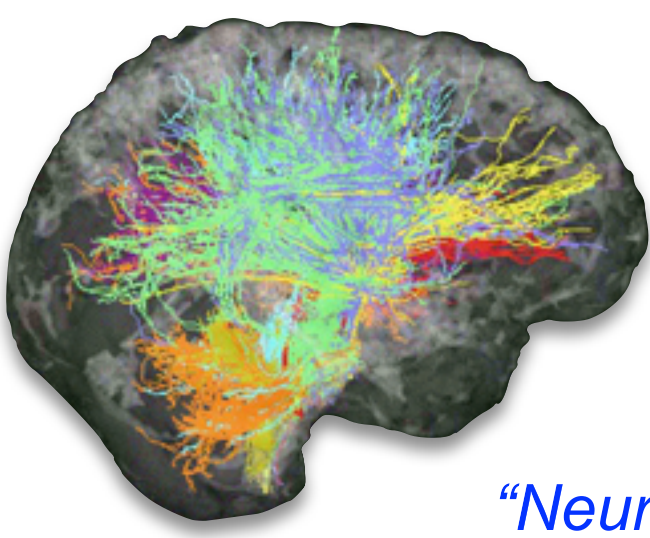
Duke University, July 2011

Processing Signals on Graphs

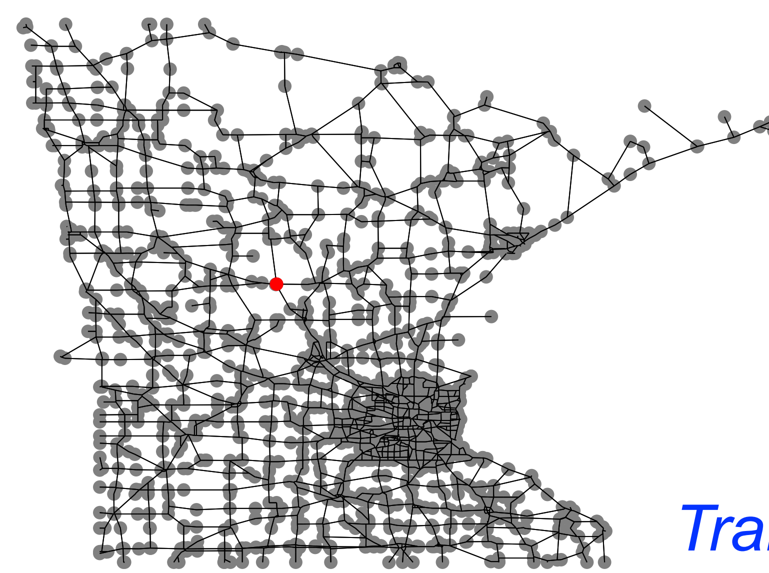
Social Network



Electrical Network

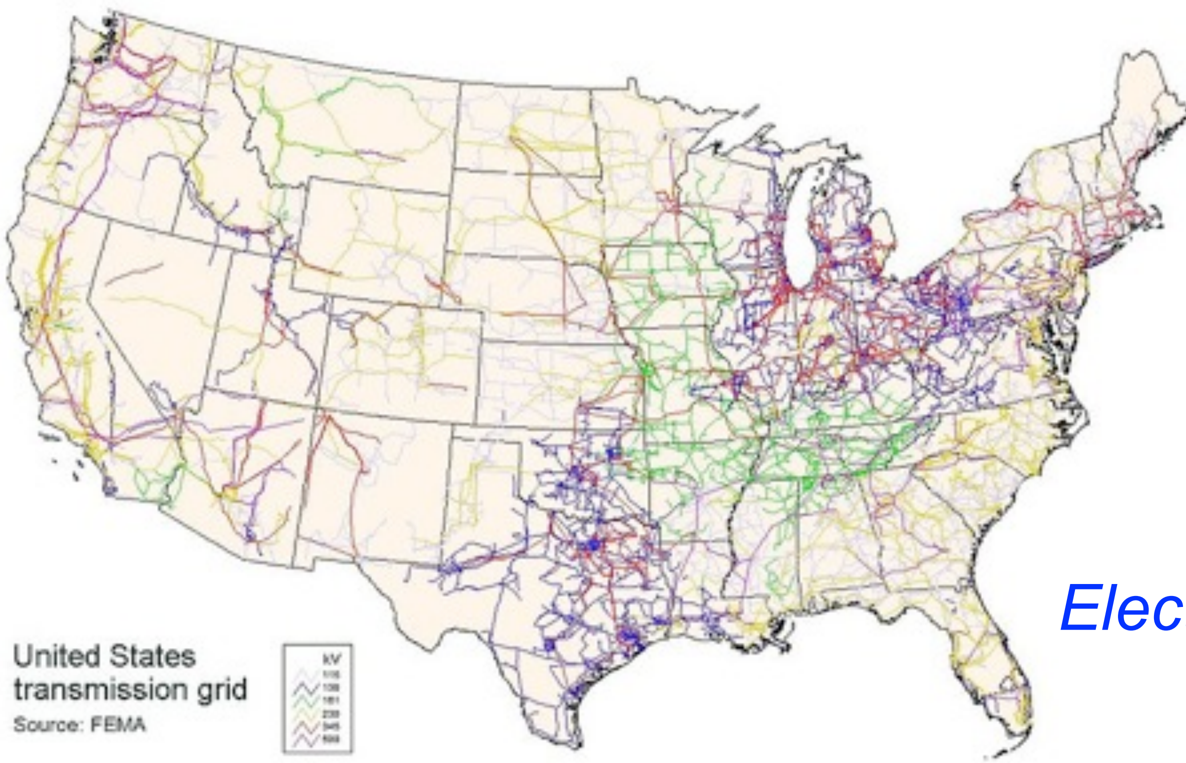


“Neuronal” Network



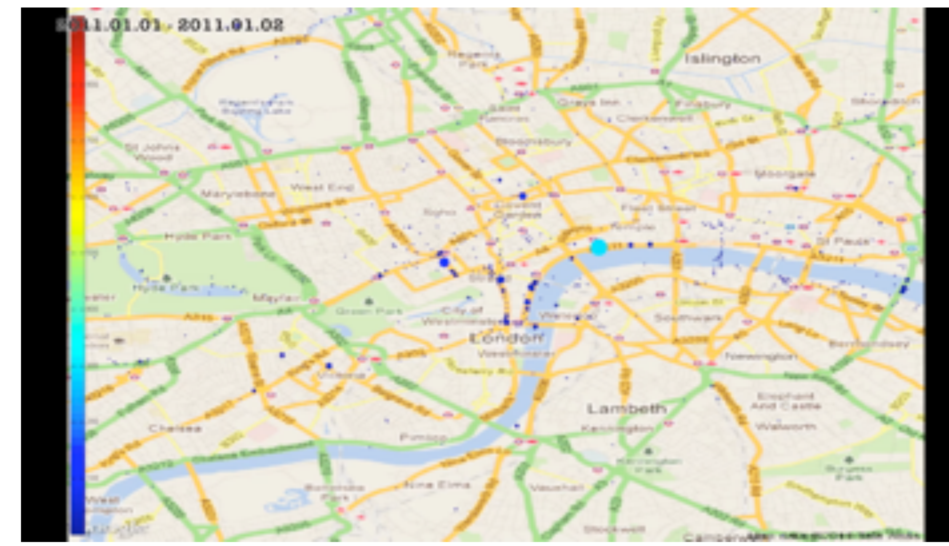
Transportation Network

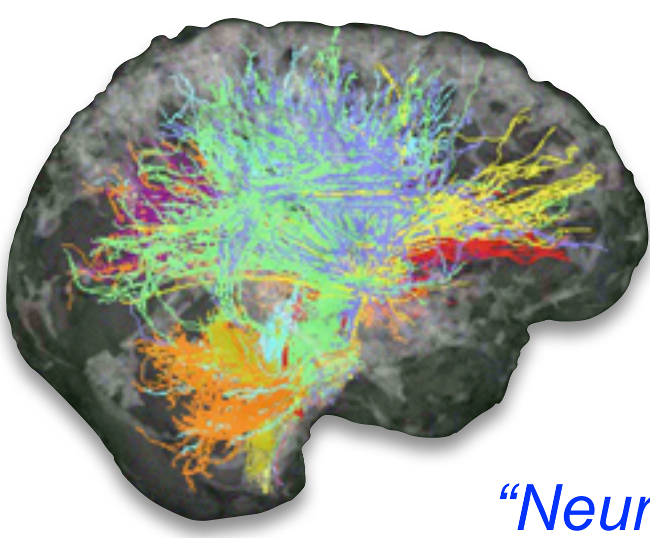
Processing Signals on Graphs



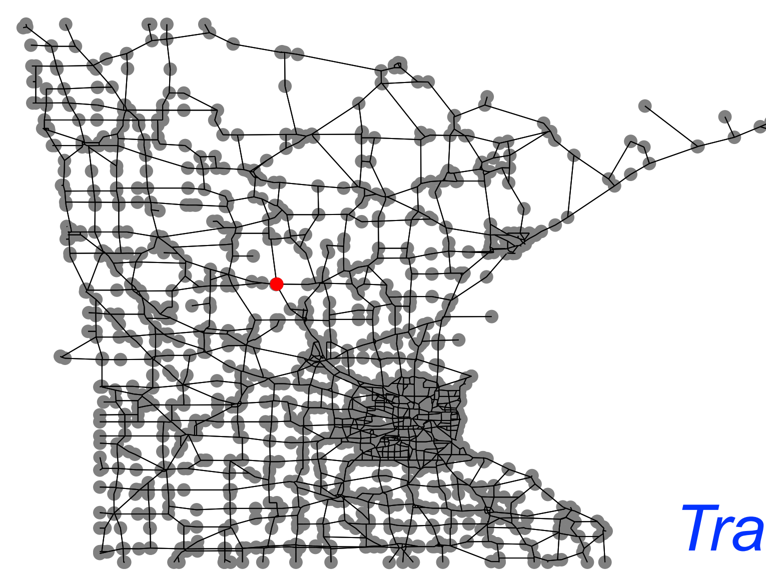
Electrical Network

Social Network





"Neuronal" Network



Transportation Network

Short outline

- Summary of one wavelet construction on graphs
  - multiscale, filtering
- Pyramidal algorithms
  - polyphase components and downsampling
  - the Laplacian Pyramid
  - 2-channels, critically sampled filter banks ?

Spectral Graph Wavelets

$G\!=\!(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Spectral Graph Wavelets

$G=(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Dilation operates through operator: $T_g^t = g(t\mathcal{L})$

Spectral Graph Wavelets

$G=(E,V)$ a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Dilation operates through operator: $T_g^t = g(t\mathcal{L})$

Translation (localization):

Define $\psi_{t,j} = T_g^t \delta_j$ response to a delta at vertex j

$$\psi_{t,j}(i) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\phi_\ell^*(j)\phi_\ell(i) \qquad \mathcal{L}\phi_\ell(j) = \lambda_\ell\phi_\ell(j)$$

$$\psi_{t,a}(u) = \int_{\mathbb{R}} d\omega\, \hat{\psi}(t\omega)e^{-j\omega a}e^{j\omega u}$$

Spectral Graph Wavelets

G=(E,V) a weighted undirected graph, with Laplacian $\mathcal{L} = D - A$

Dilation operates through operator: $T_g^t = g(t\mathcal{L})$

Translation (localization):

Define $\psi_{t,j} = T_g^t \delta_j$ response to a delta at vertex j

$$\psi_{t,j}(i) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\phi_\ell^*(j)\phi_\ell(i) \qquad \mathcal{L}\phi_\ell(j) = \lambda_\ell \phi_\ell(j)$$

$$\psi_{t,a}(u) = \int_{\mathbb{R}} d\omega\, \hat{\psi}(t\omega) e^{-j\omega a} e^{j\omega u}$$

And so formally define the graph wavelet coefficients of f:

$$W_f(t,j) = \langle \psi_{t,j}, f \rangle \qquad W_f(t,j) = T_g^t f(j) = \sum_{\ell=0}^{N-1} g(t\lambda_\ell)\hat{f}(\ell)\phi_\ell(j)$$

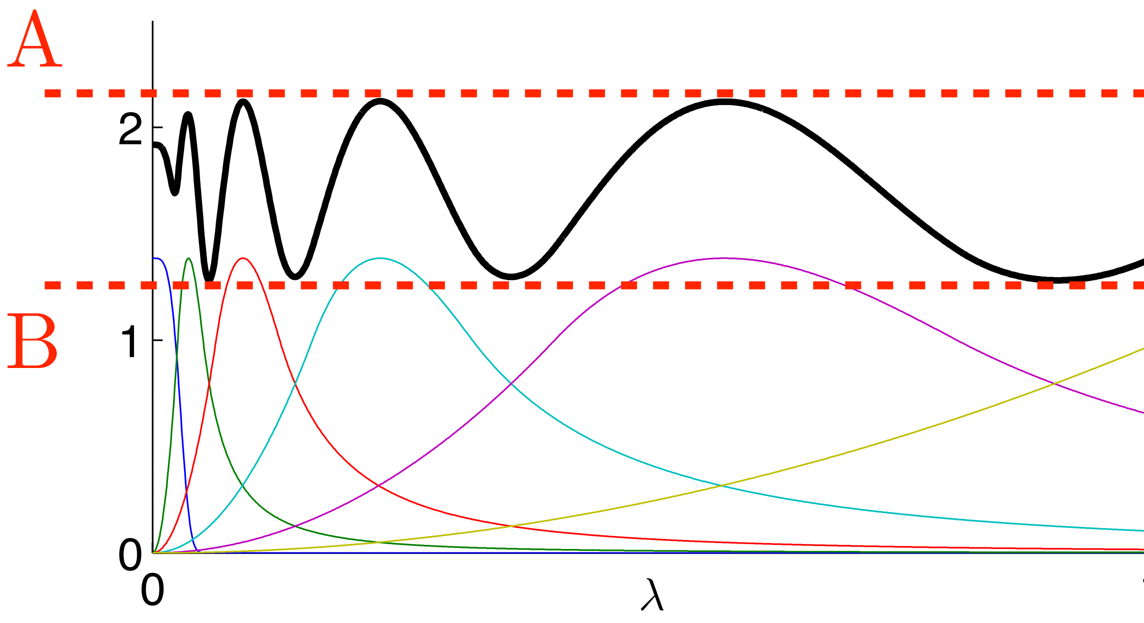
Frames

$\exists\, A, B > O,\ \exists h : \mathbb{R}_+ \to \mathbb{R}_+$ (i.e. scaling function)

$$0 < A \leqslant h^2(u) + \sum_s g(t_s u)^2 \leqslant B < \infty$$

scaling function   *wavelets*

$\phi_n = T_h \delta_n = h(\mathcal{L})\delta_n$

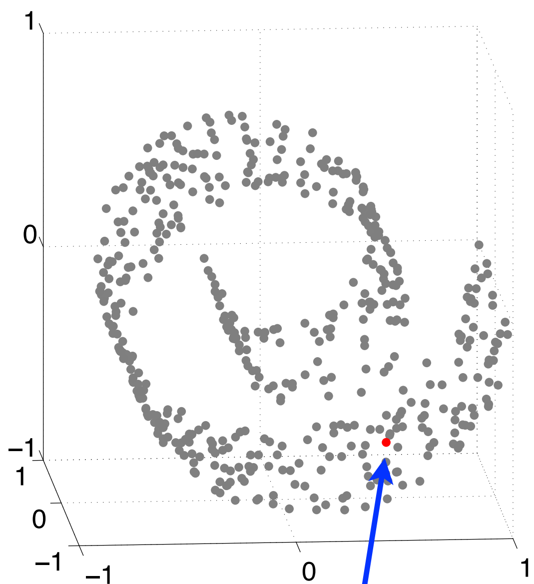


<u>A simple way to get a tight frame:</u>

$$\gamma(\lambda_\ell) = \int_{1/2}^{1} \frac{dt}{t}\, g^2(t\lambda_\ell) \quad \Longrightarrow \quad \tilde{g}(\lambda_\ell) = \sqrt{\gamma(\lambda_\ell) - \gamma(2\lambda_\ell)}$$

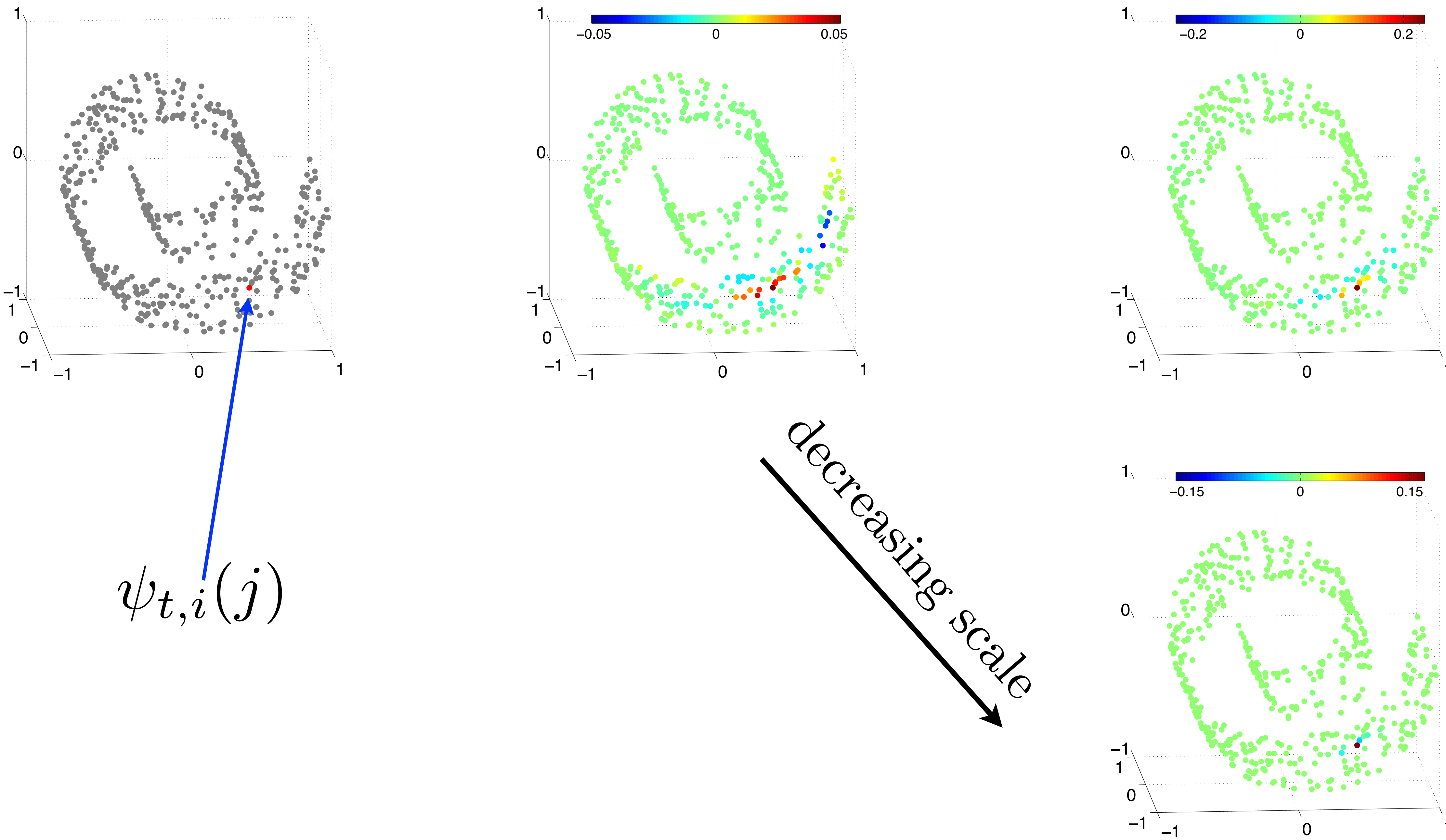
for any admissible kernel g

Scaling & Localization

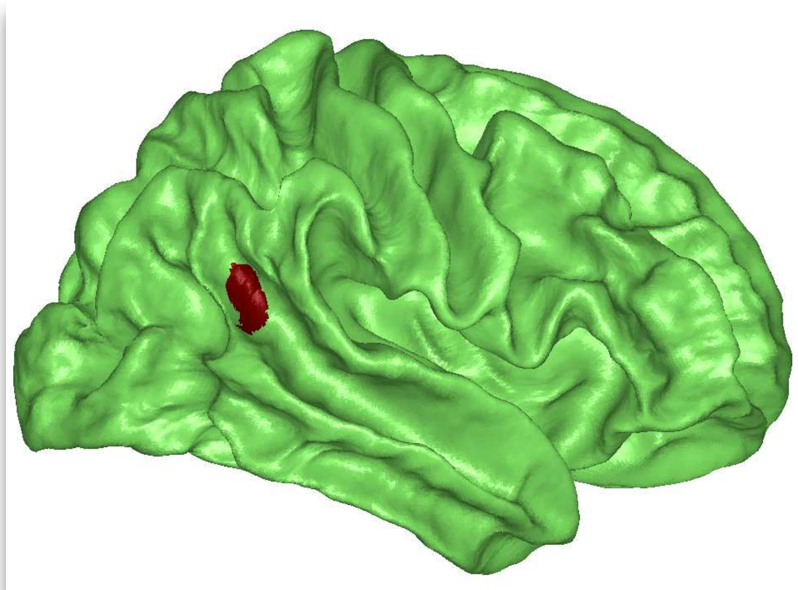


$\psi_{t,i}(j)$

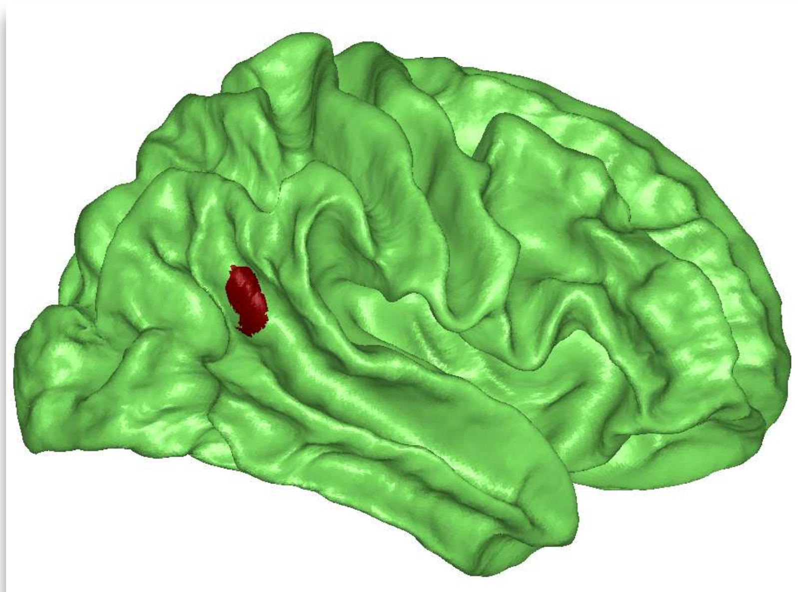
Scaling & Localization

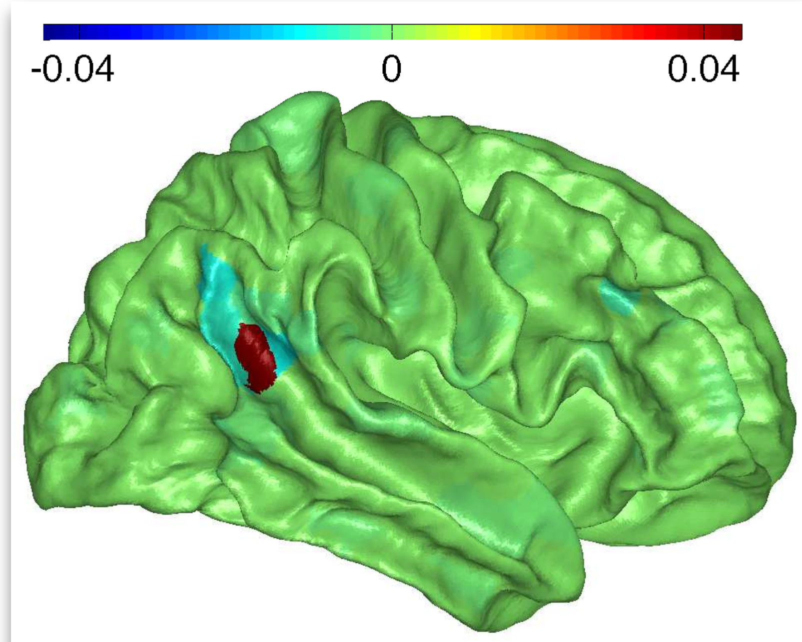


Example

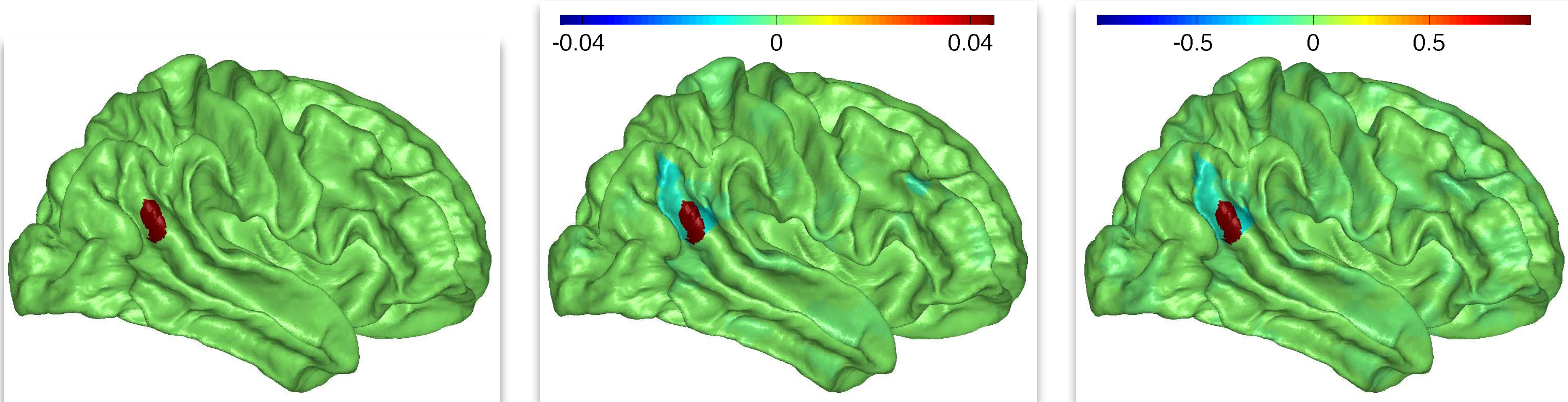


Example

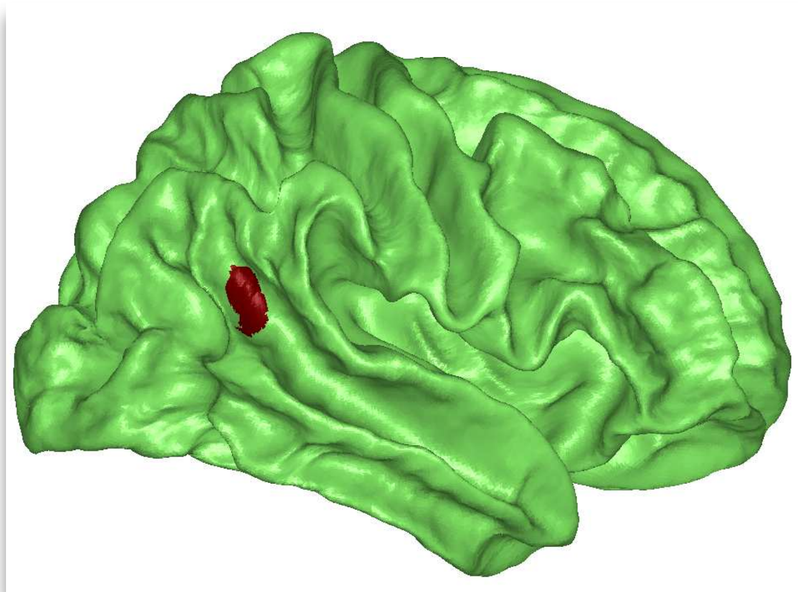


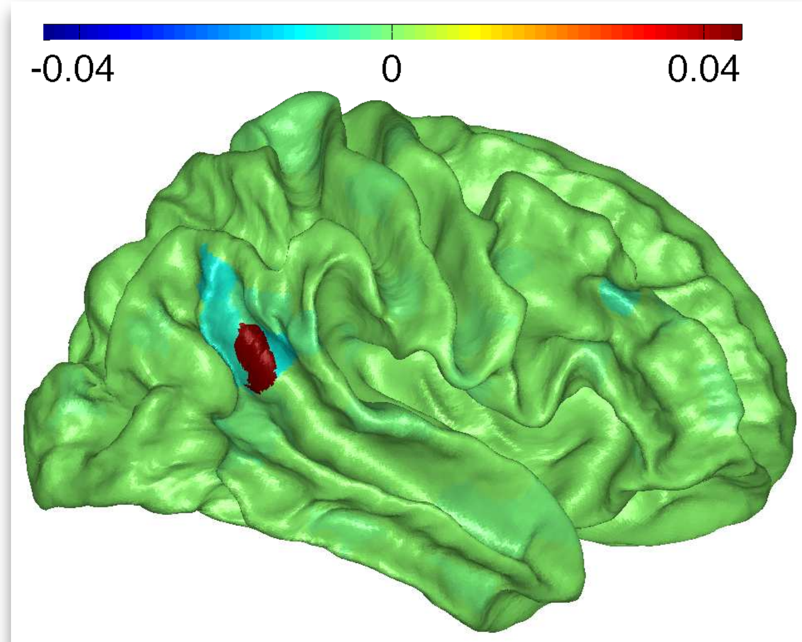


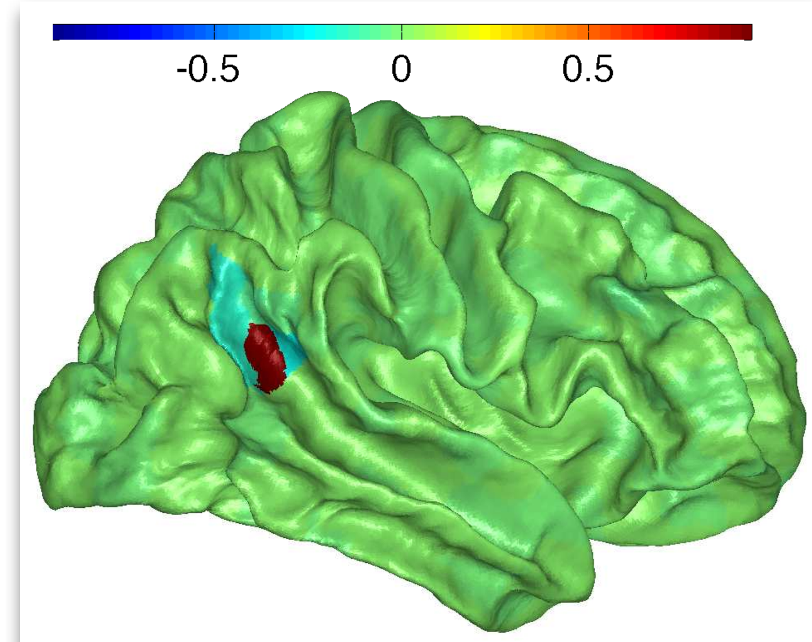
Example

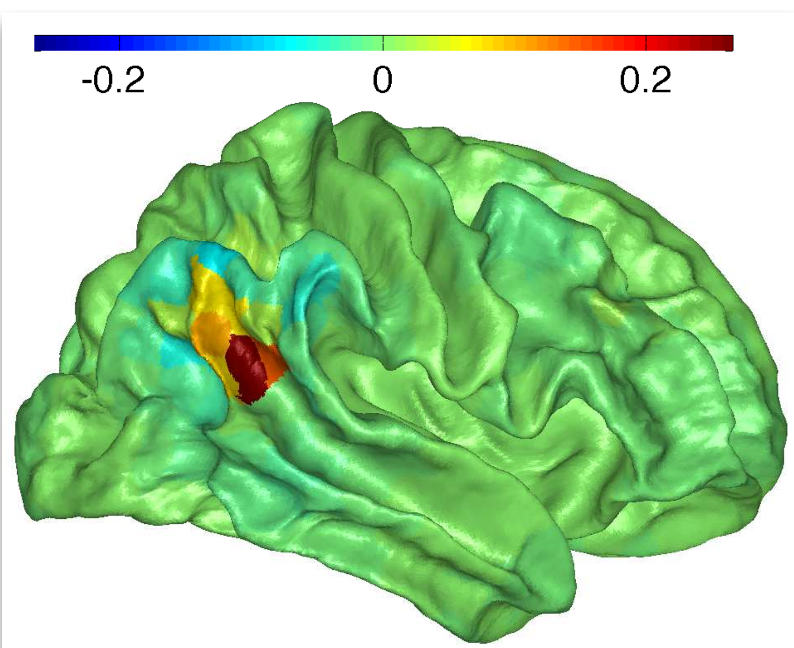


Example

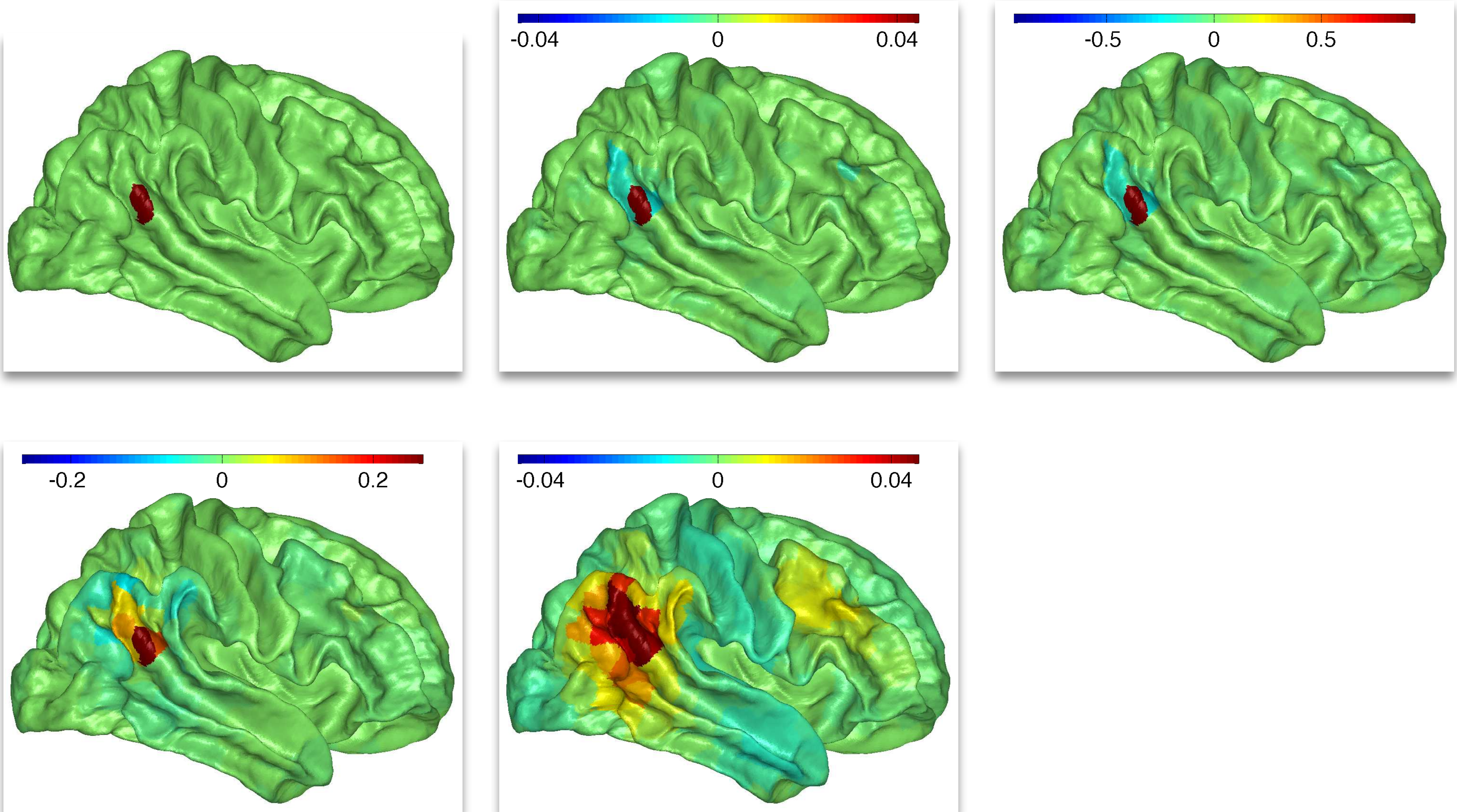




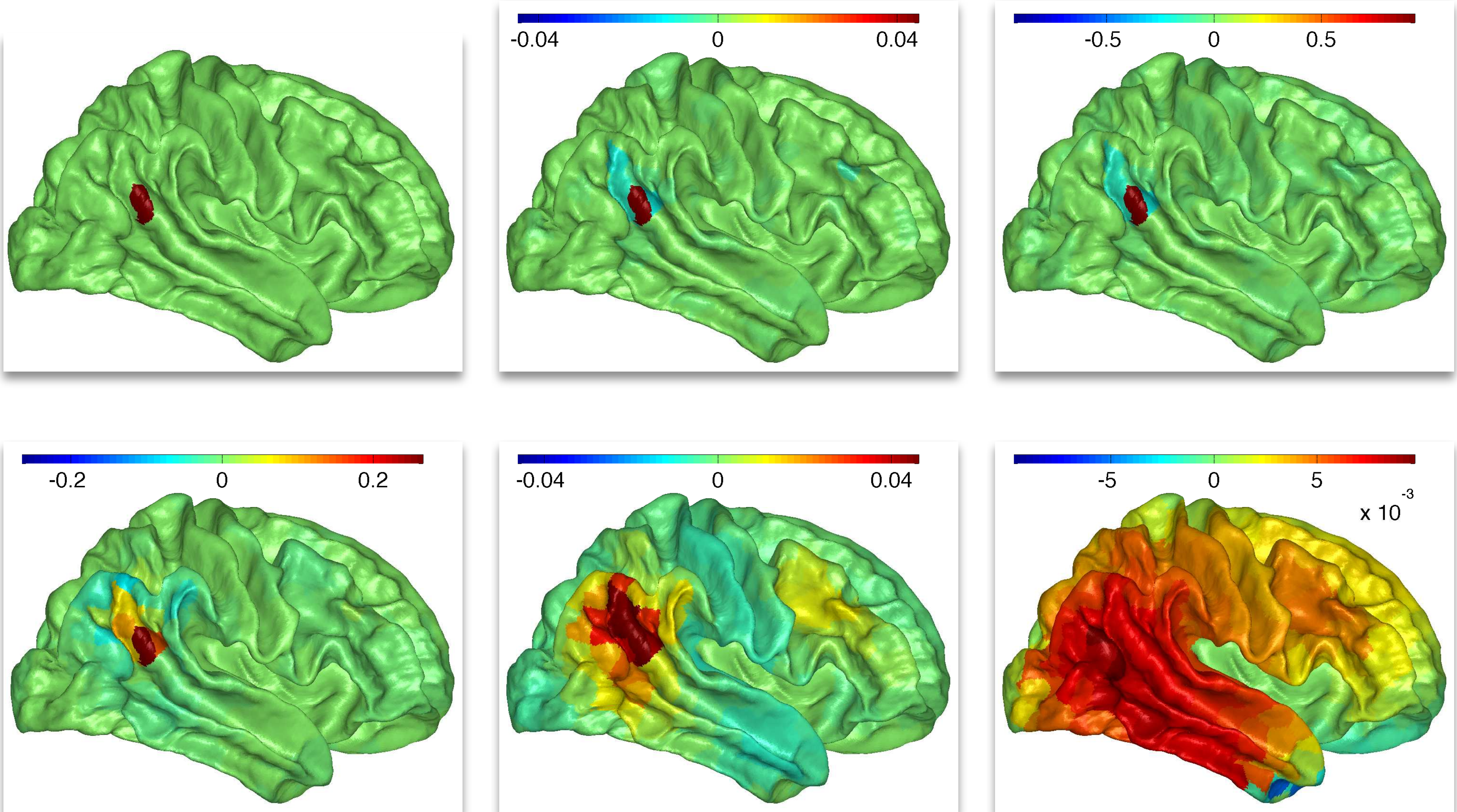




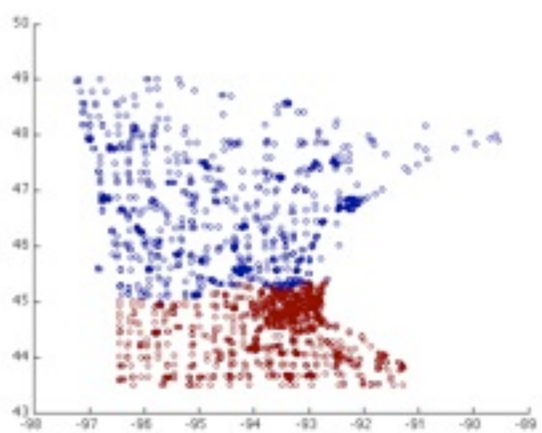
Example

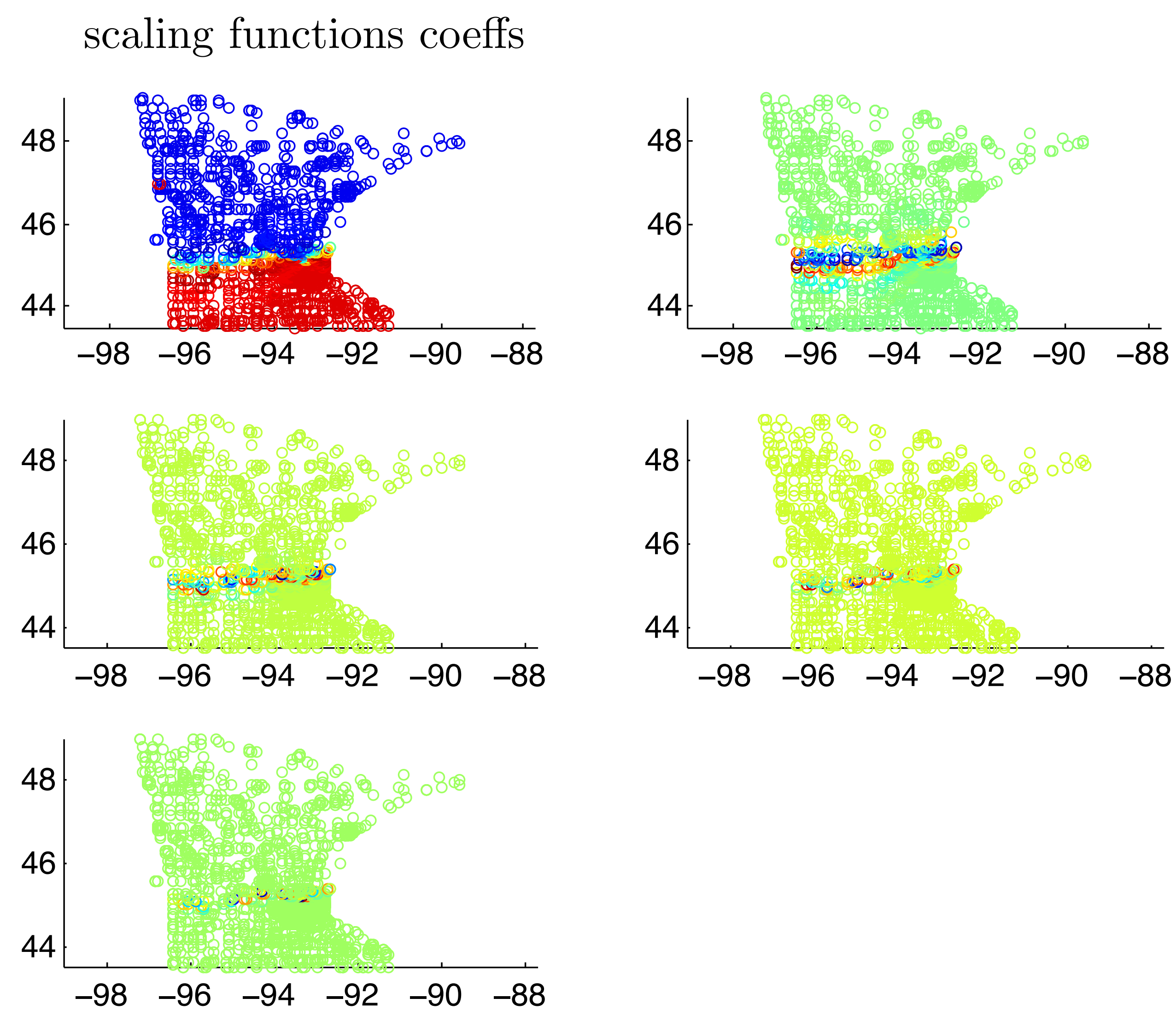


Example



Sparsity and Smoothness on Graphs



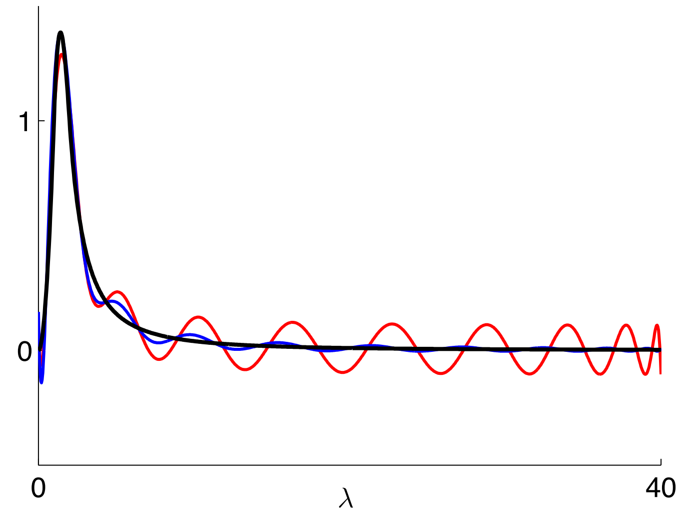


Remark on Implementation

Not necessary to compute spectral decomposition for filtering

Polynomial approximation : $$g(t\omega) \simeq \sum_{k=0}^{K-1} a_k(t) p_k(\omega)$$

ex: Chebyshev, minimax



It is to implement any Fourier multiplier

Then wavelet operator expressed with … acian:

$$T_g^t \simeq \sum_{k=0}^{K-1} a_k(t) \mathcal{L}^k$$

And use sparsity of Laplacian in an iterative way

Remark on Implementation

$$\tilde{W}_f(t,j) = \left(p(\mathcal{L})f^\#\right)_j \qquad |W_f(t,j) - \tilde{W}_f(t,j)| \leq B\|f\|$$

sup norm control (minimax or Chebyshef)

$$\tilde{W}_f(t_n,j) = \left(\frac{1}{2}c_{n,0}f^\# + \sum_{k=1}^{M_n} c_{n,k}\overline{T}_k(\mathcal{L})f^\#\right)_j$$

$$\overline{T}_k(\mathcal{L})f = \frac{2}{a_1}(\mathcal{L} - a_2 I)\left(\overline{T}_{k-1}(\mathcal{L})f\right) - \overline{T}_{k-2}(\mathcal{L})f$$

Computational cost dominated by matrix-vector multiply with (sparse) Laplacian matrix.

In particular $O(\sum_{n=1} M_n|E|)$

http://wiki.epfl.ch/sgwt

Note: "same" algorithm for adjoint !

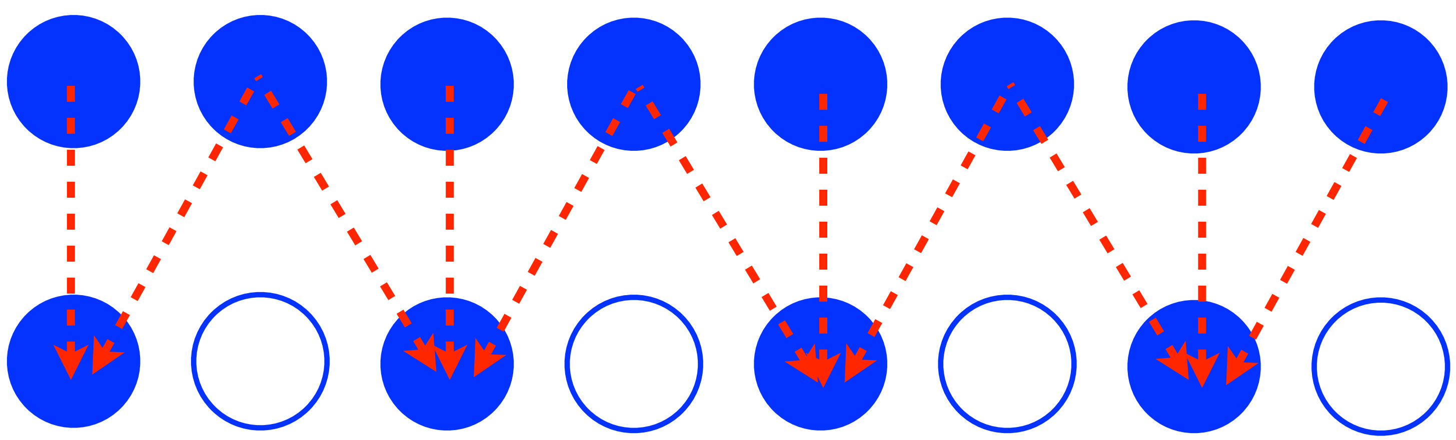
Graph wavelets

- Redundancy breaks sparsity
  - can we remove some or all of it ?
- Faster algorithms
  - traditional wavelets have fast filter banks implementation
  - whatever scale, you use the same filters
  - here: large scales -> more computations
- Goal: solve both problems at one

Basic Ingredients

Euclidean multiresolution is based on two main operations

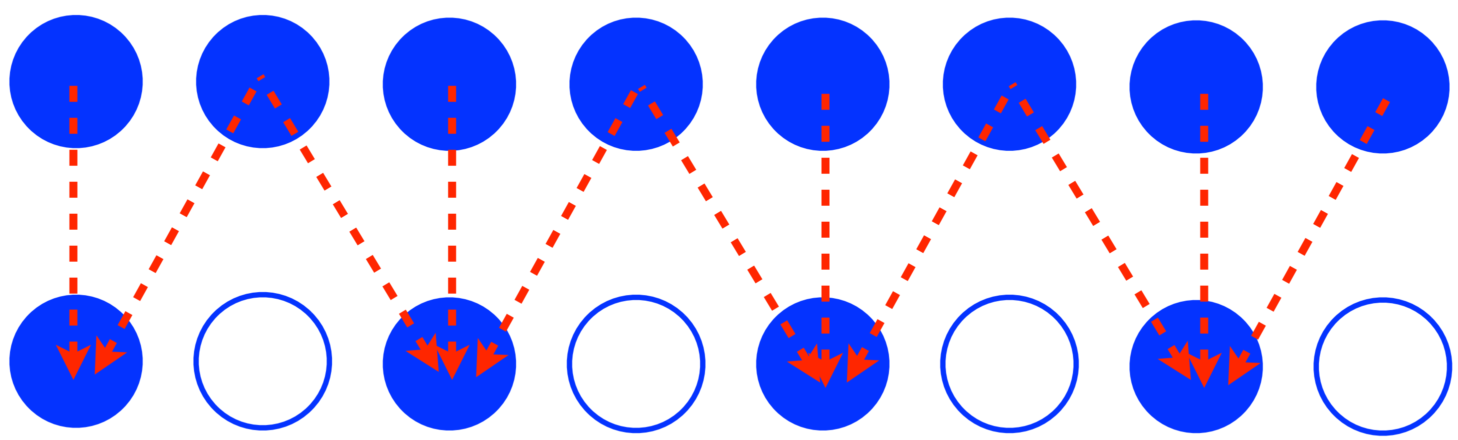
- Filtering (typically low-pass and high-pass)
- Down and Up sampling

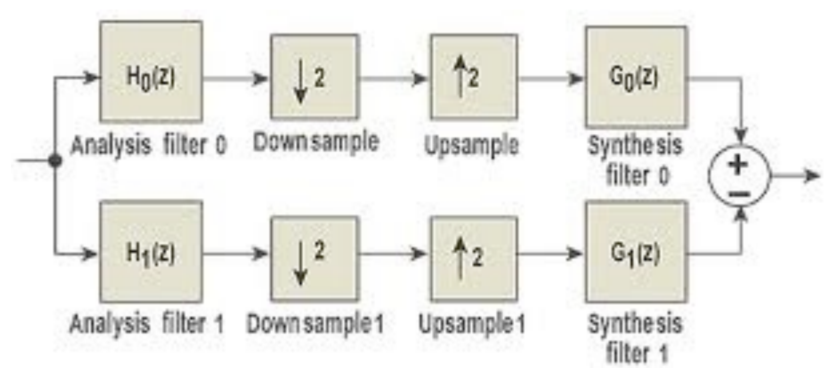


Basic Ingredients

Euclidean multiresolution is based on two main operations

- Filtering (typically low-pass and high-pass)
- Down and Up sampling



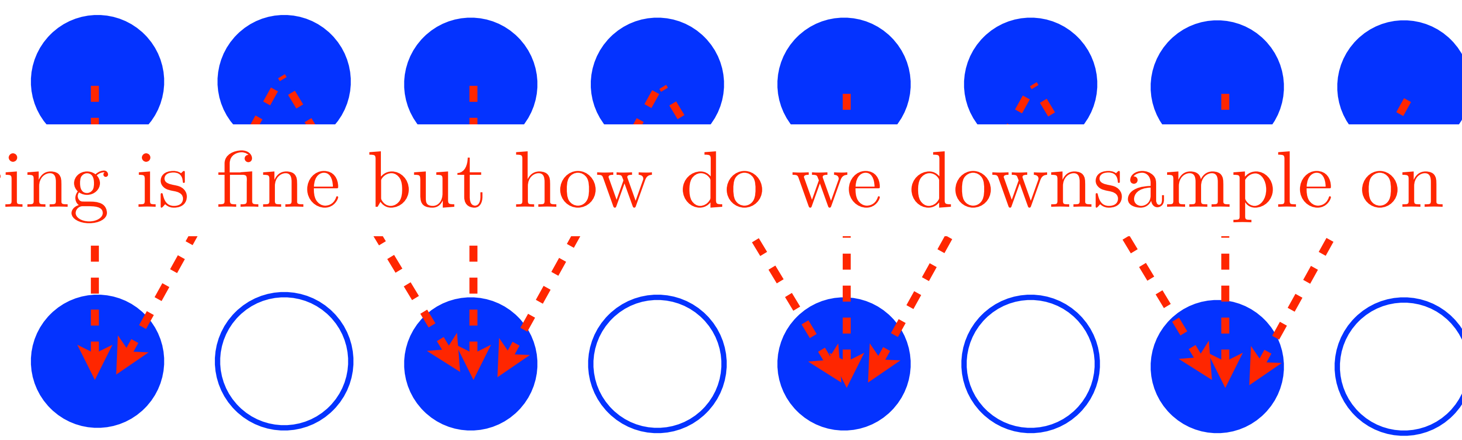


Basic Ingredients

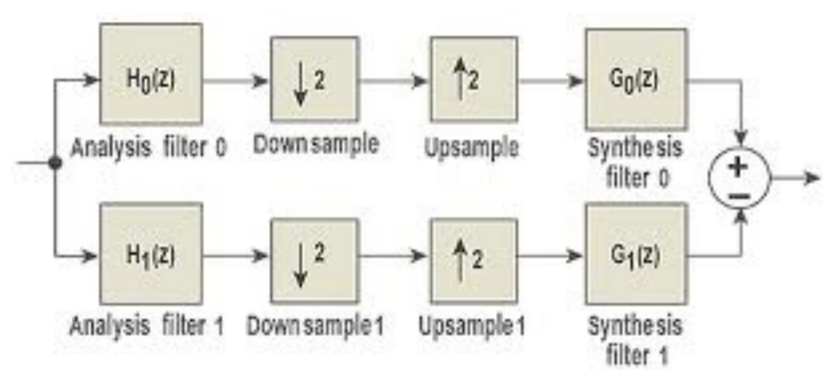
Euclidean multiresolution is based on two main operations

Filtering (typically low-pass and high-pass)

Down and Up sampling

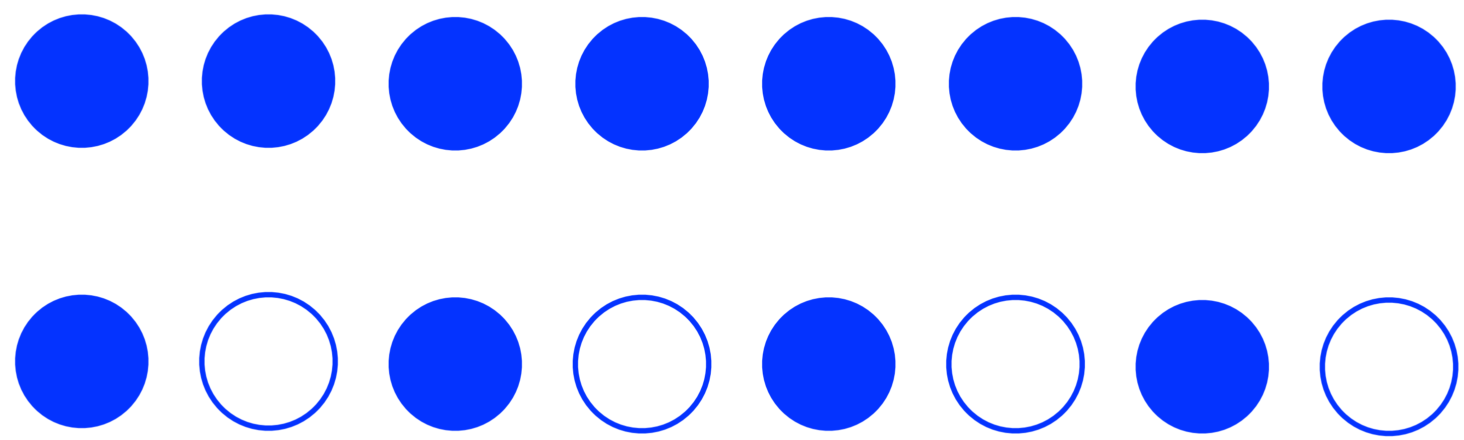


Filtering is fine but how do we downsample on graphs ???



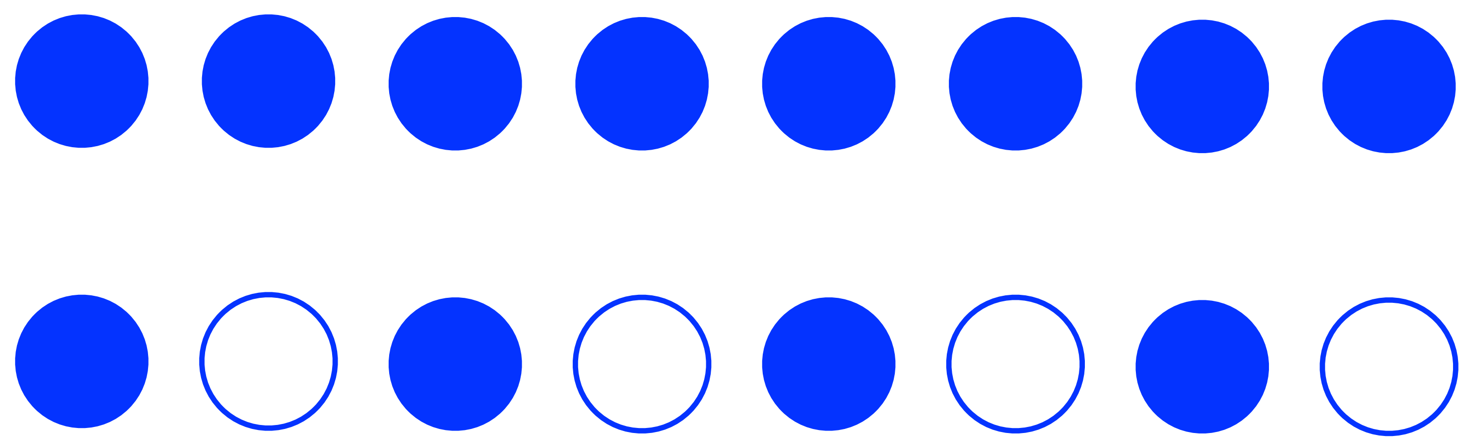
Basic Ingredients

Subsampling is equivallent to splitting in two cosets (even, odd)



Basic Ingredients

Subsampling is equivallent to splitting in two cosets (even, odd)



Questions: How do we partition a graph into meaningful cosets ?

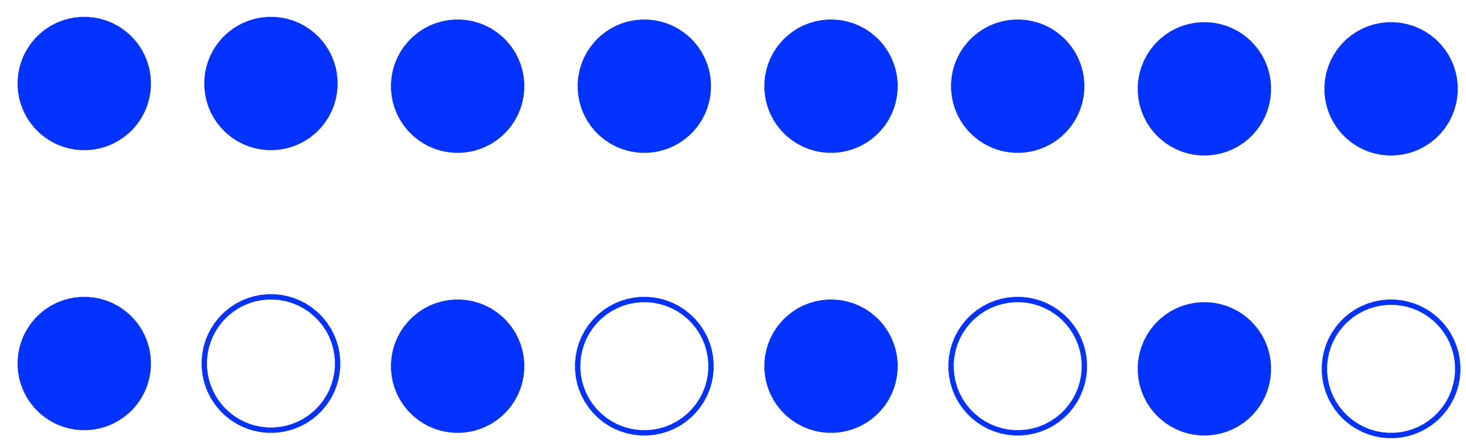
Are there efficient algorithms for these partitions ?

Are there theoretical guarantees ?

How do we define a new graph from the cosets ?

Cosets - A spectral view

Subsampling is equivallent to splitting in two cosets (even, odd)



Classically, selecting a coset can be interpreted easily in Fourier:

$$f_{\text{sub}}(i) = \frac{1}{2} f(i) \big(1 + \cos(\pi i)\big)$$

eigenvector of largest eigenvalue

Cosets and Nodal Domains

Nodal domain: maximally connected subgraph s.t. all vertices have same sign w.r.t a reference function

We would like to find a very large number of nodal domains, ideally $|V|$!

Nodal domains of Laplacian eigenvectors are special (and well studied)

Cosets and Nodal Domains

Nodal domain: maximally connected subgraph s.t. all vertices have same sign w.r.t a reference function

We would like to find a very large number of nodal domains, ideally $|V|$!

Nodal domains of Laplacian eigenvectors are special (and well studied)

Theorem: the number of nodal domains associated to the largest laplacian eigenvector of a connected graph is maximal,

$$\nu(\phi_{\max}) = \nu(G) = |V|$$

IFF G is bipartite

In general: $\nu(G) = |V| - \chi(G) + 2$ (extreme cases: bipartite and complete graphs)

Cosets and Nodal Domains

Nodal domain: maximally connected subgraph s.t. all vertices have same sign w.r.t a reference function

We would like to find a very large number of nodal domains, ideally $|V|$!

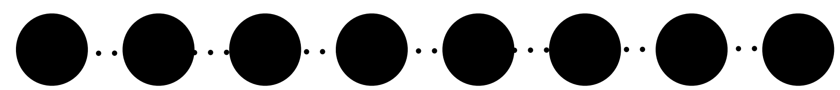
Nodal domains of Laplacian eigenvectors are special (and well studied)

For any connected graph we will thus naturally define cosets and their associated selection functions

$$V_+ = \{i \in V \text{ s.t. } \phi_{N-1}(i) \geq 0\} \qquad V_- = \{i \in V \text{ s.t. } \phi_{N-1}(i) < 0\}$$

$$M_+(i) = \frac{1}{2}\big(1 + \text{sgn}(\phi_{N-1}(i))\big) \qquad M_-(i) = \frac{1}{2}\big(1 - \text{sgn}(\phi_{N-1}(i))\big)$$

Examples of cosets

Simple line graph 

$\phi_k(u) = \sin(\pi k u/n + \pi/2n) \qquad \lambda_k = 2 - 2\cos(\pi k/n) \qquad 1 \leq k \leq n$

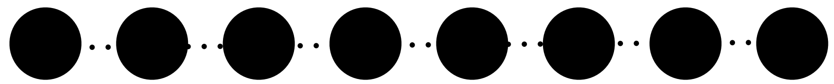
Examples of cosets

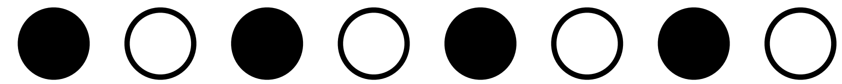
Simple line graph

$\phi_k(u) = \sin(\pi k u/n + \pi/2n)$ $\lambda_k = 2 - 2\cos(\pi k/n)$ $1 \leq k \leq n$

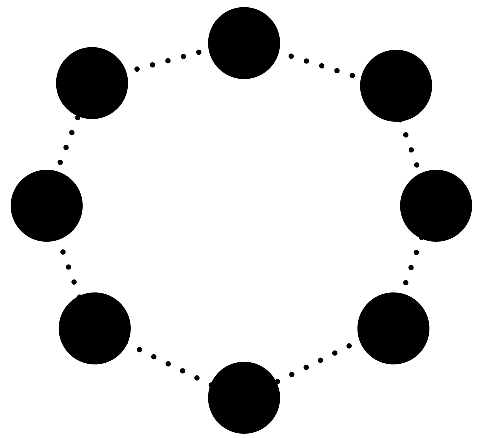
Examples of cosets

Simple line graph





Simple ring graph



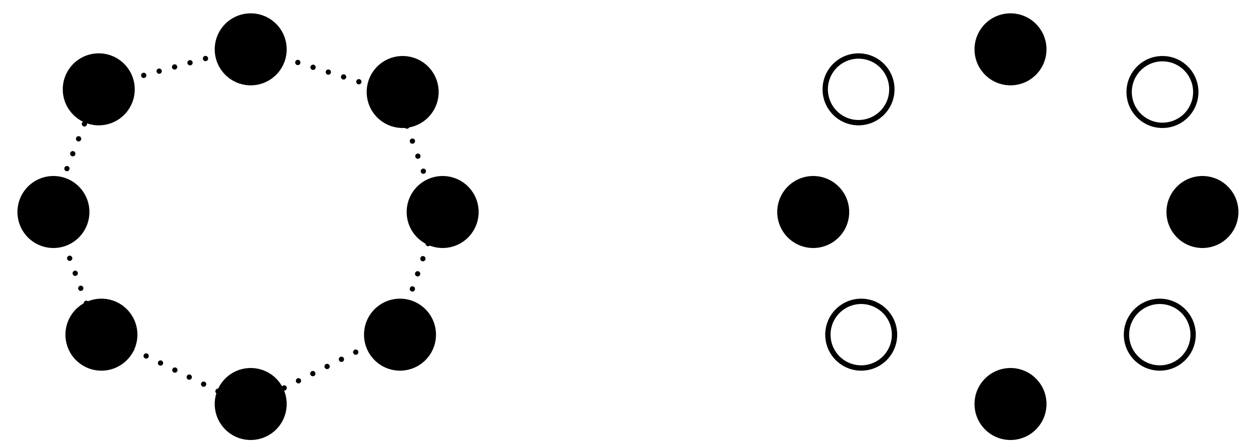
$$\phi_k^1(u) = \sin(2\pi k u / n) \qquad \phi_k^2(u) = \cos(2\pi k u / n) \quad 1 \leq k \leq n/2$$

$$\lambda_k = 2 - 2\cos(2\pi k / n)$$

Examples of cosets

Simple line graph

Simple ring graph



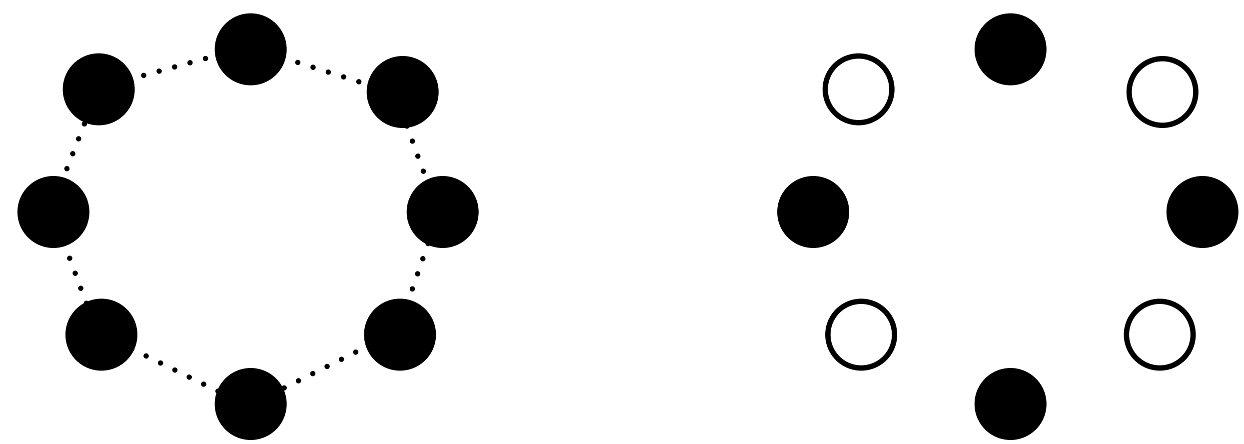
$$\phi_k^1(u) = \sin(2\pi k u/n) \qquad \phi_k^2(u) = \cos(2\pi k u/n) \quad 1 \leq k \leq n/2$$

$$\lambda_k = 2 - 2\cos(2\pi k/n)$$

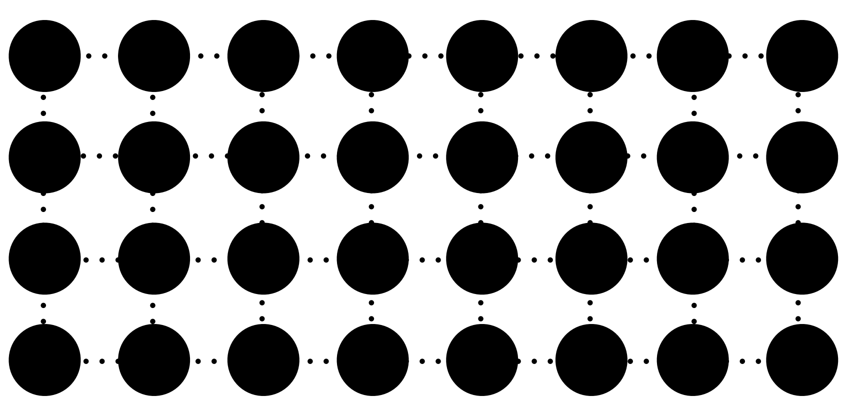
Examples of cosets

Simple line graph

Simple ring graph



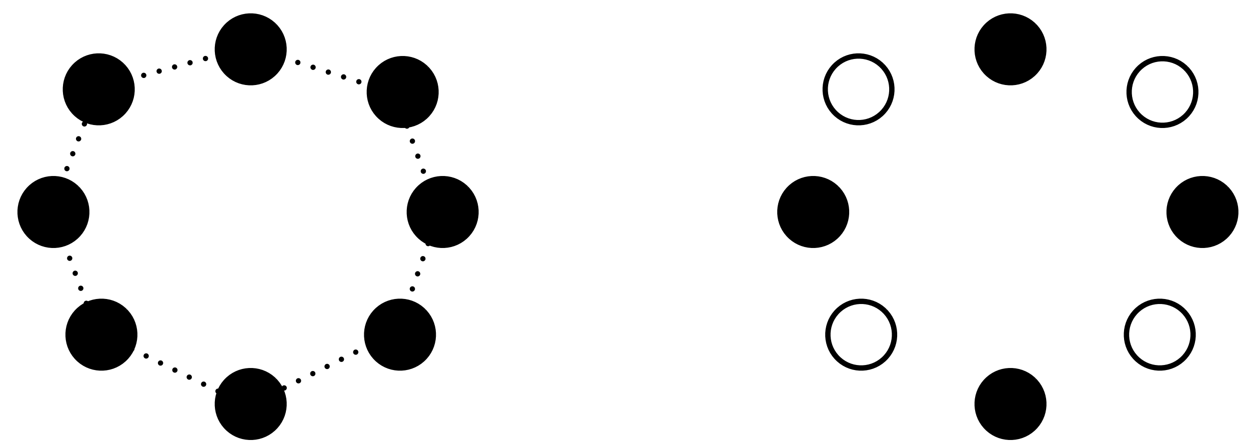
Lattice



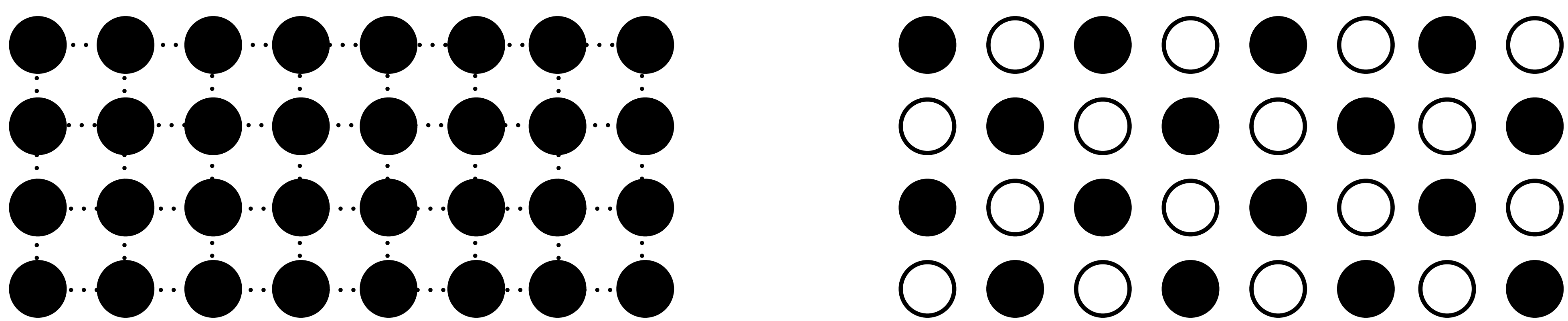
Examples of cosets

Simple line graph

Simple ring graph



Lattice



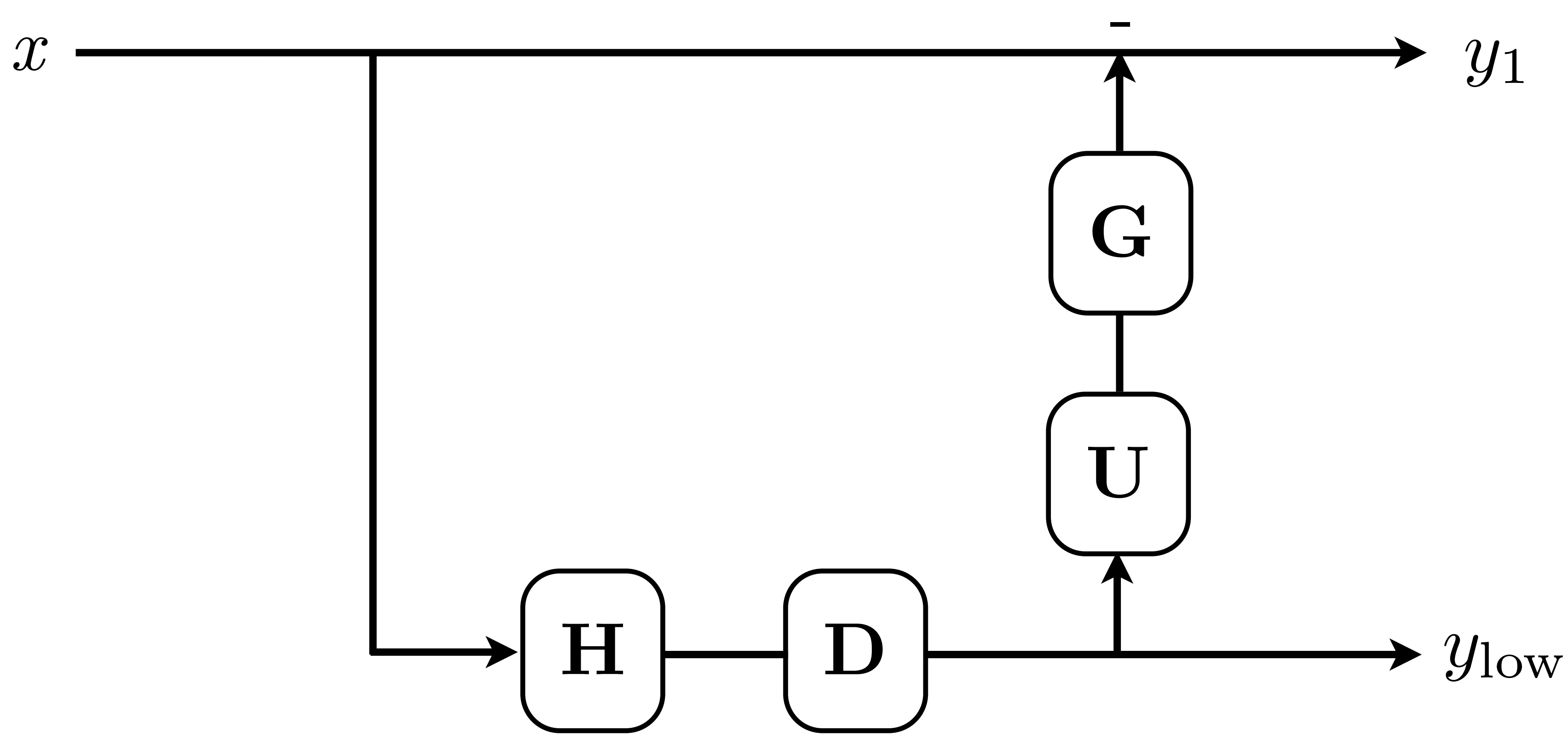
quincunx

The Agonizing Limits of Intuition

- Multiplicity of $\lambda_{\max}$
  - how do we choose the control vector in that subspace ?
  - even a prescription can be numerically ill-defined
  - graphs with "flat" spectrum in close to their spectral radius
- Laplacian eigenvectors do not always behave like global oscillations
  - seems to be true for random perturbations of simple graphs
  - true even for a class of trees [Saito2011]

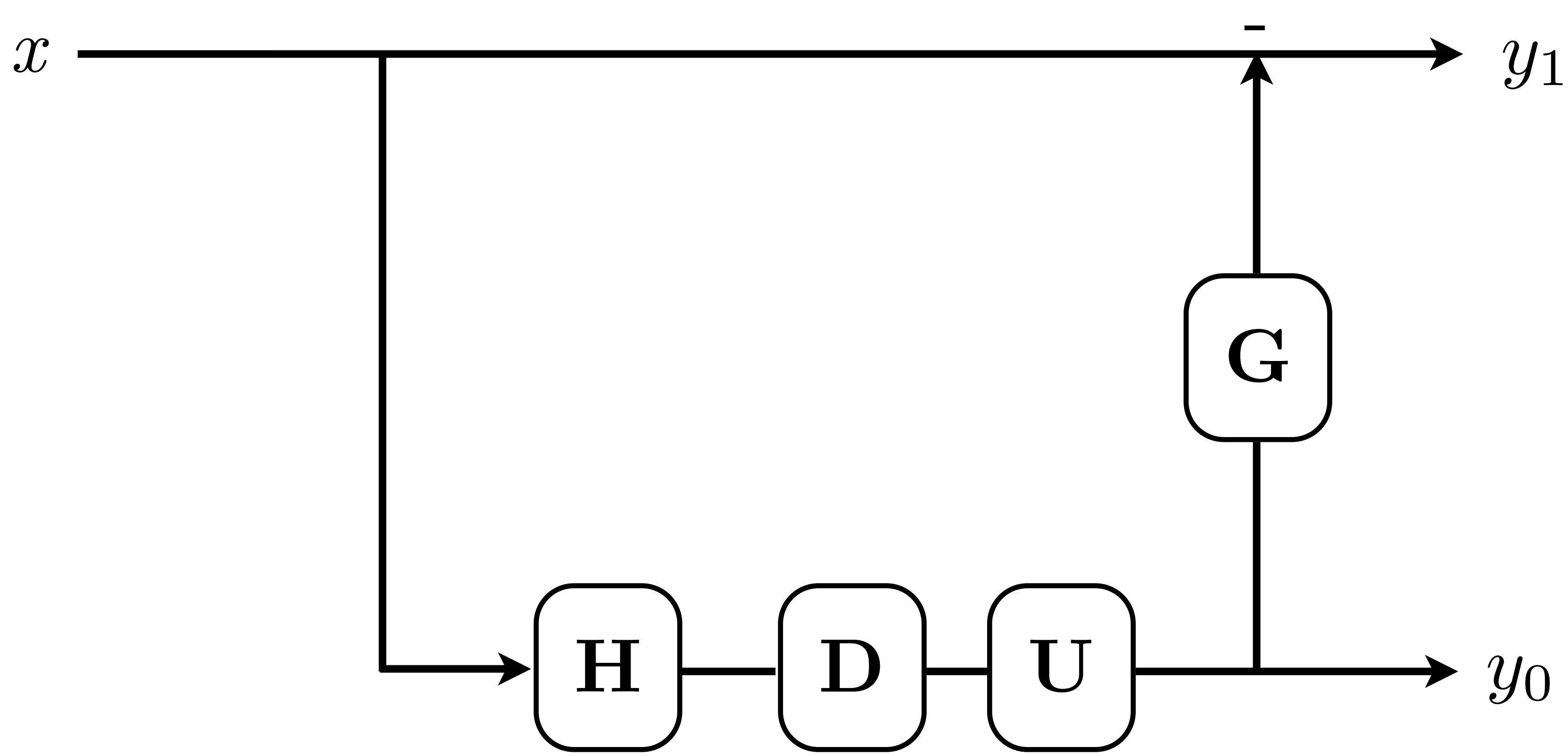
The Laplacian Pyramid

Analysis operator



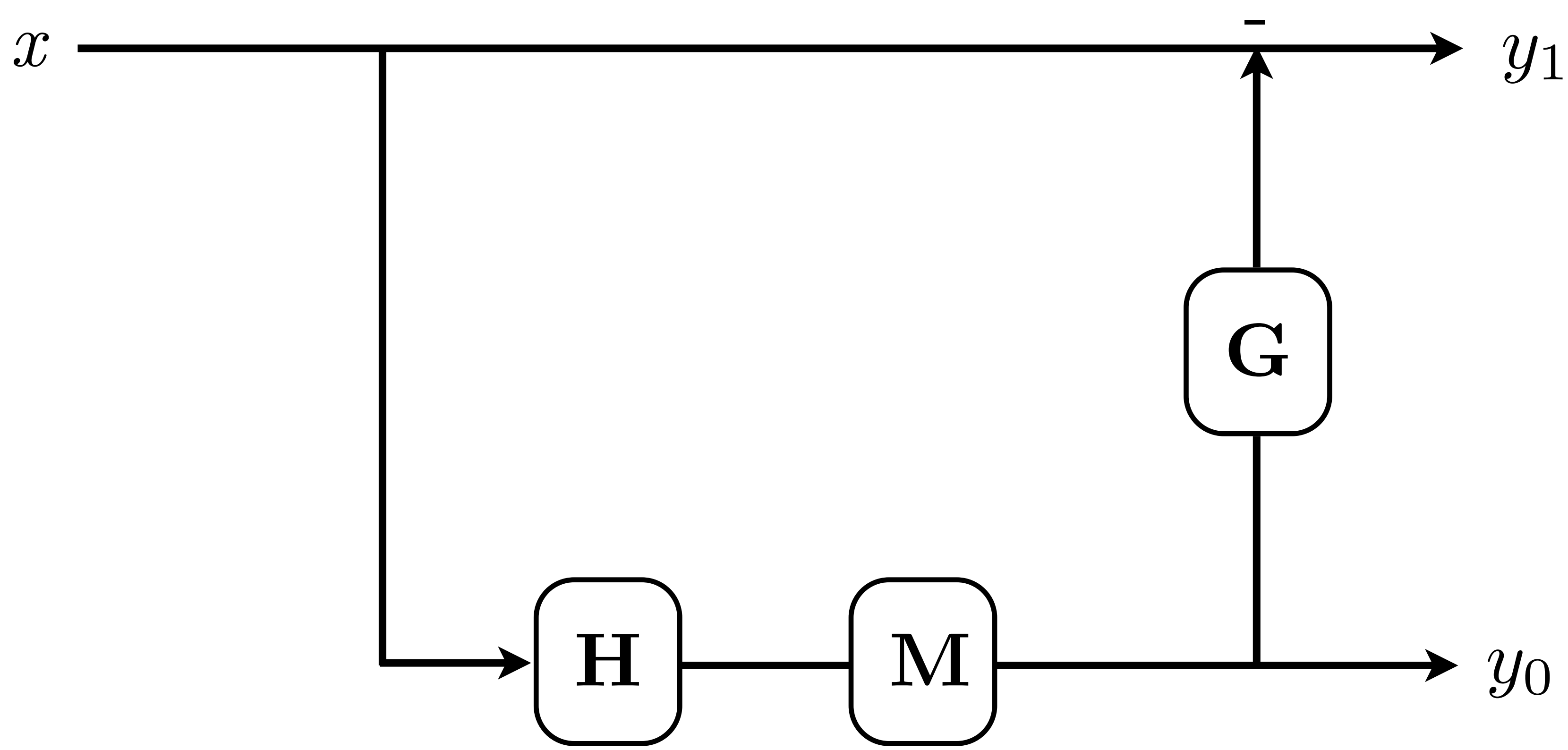
The Laplacian Pyramid

Analysis operator



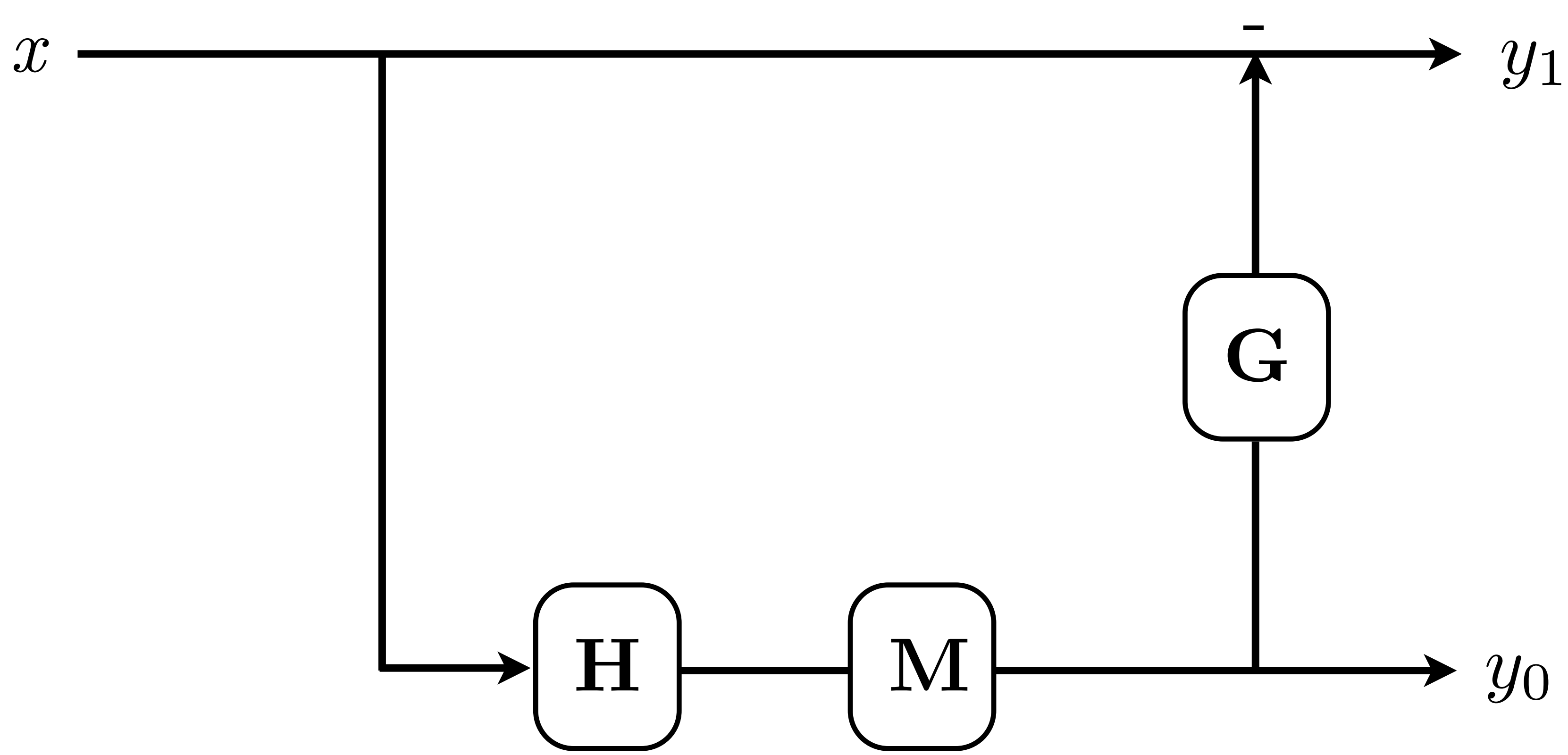
The Laplacian Pyramid

Analysis operator



The Laplacian Pyramid

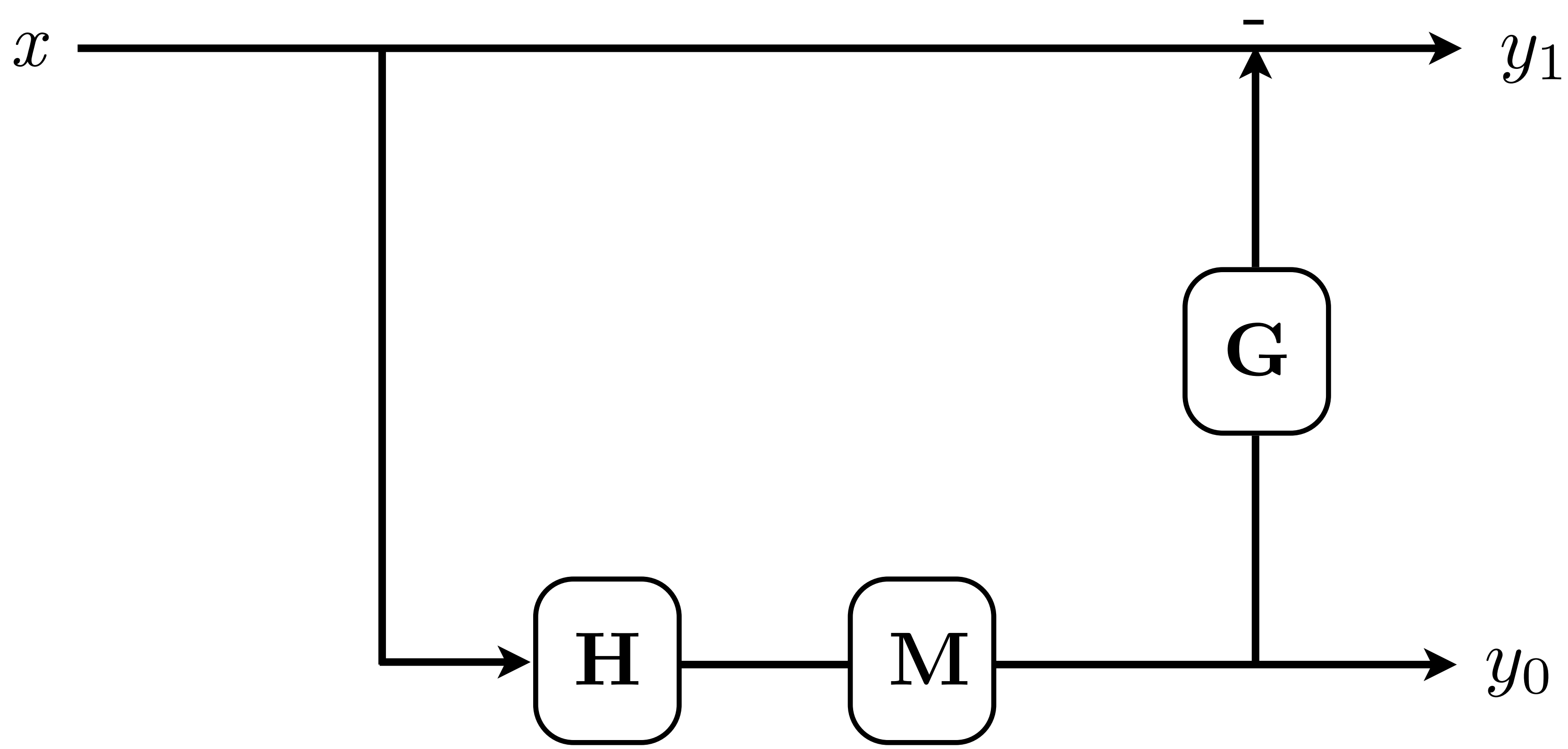
Analysis operator



$$
\begin{aligned}
y_0 &= \mathbf{H_m} x \\
&= \mathbf{MH} x
\end{aligned}
\qquad\qquad
\begin{aligned}
y_1 &= x - \mathbf{G} y_0 \\
&= x - \mathbf{G H_m} x
\end{aligned}
$$

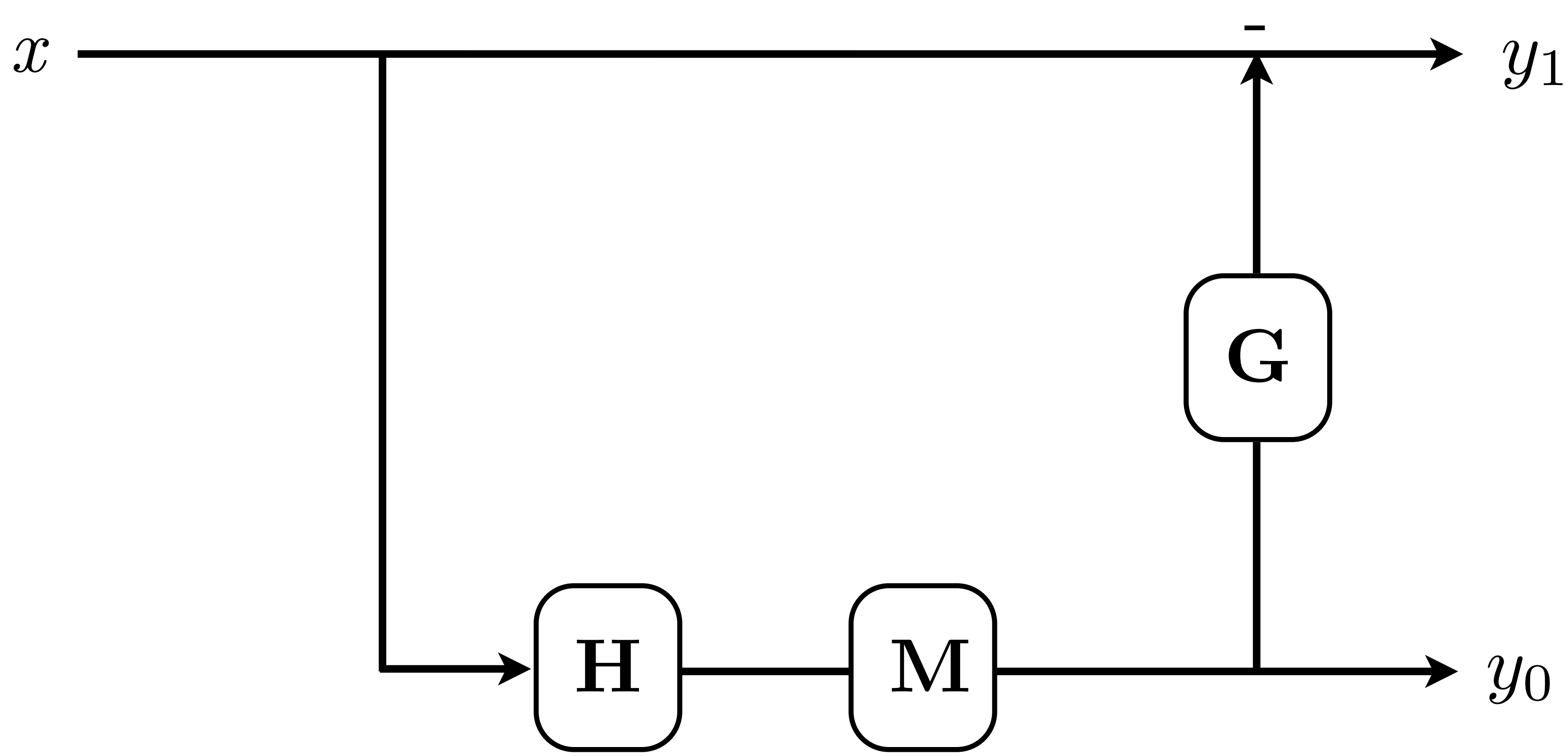
The Laplacian Pyramid

Analysis operator



The Laplacian Pyramid

Analysis operator



$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I} - \mathbf{G}\mathbf{H_m} \end{pmatrix}}_{\mathbf{T_a}} x,$$

The Laplacian Pyramid

Analysis operator

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I} - \mathbf{G}\mathbf{H_m} \end{pmatrix}}_{\mathbf{T_a}} x,$$

The Laplacian Pyramid

Analysis operator

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y} = \underbrace{\begin{pmatrix} \mathbf{H_m} \\ \mathbf{I} - \mathbf{GH_m} \end{pmatrix}}_{\mathbf{T_a}} x,$$

Simple (traditional) left inverse

$$\hat{x} = \underbrace{\begin{pmatrix} \mathbf{G} & \mathbf{I} \end{pmatrix}}_{\mathbf{T_s}} \underbrace{\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}}_{y}$$

$\mathbf{T_s}\mathbf{T_a} = \mathbf{I}$ with no conditions on $\mathbf{H}$ or $\mathbf{G}$

The Laplacian Pyramid

Pseudo Inverse ?

$$\mathbf{T_a}^{\dagger} = \left(\mathbf{T_a}^T \mathbf{T_a}\right)^{-1} \mathbf{T_a}^T$$

Let's try to use only filters

The Laplacian Pyramid

Pseudo Inverse ?

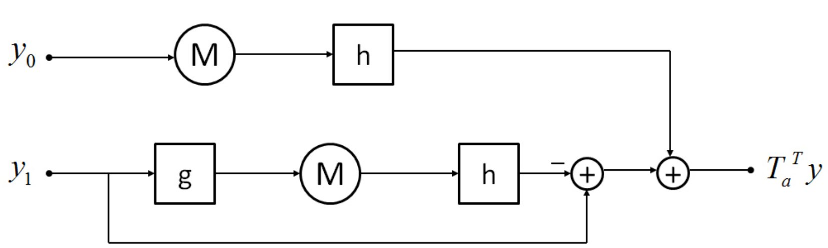
$$\mathbf{T_a}^{\dagger} = \left(\mathbf{T_a}^T \mathbf{T_a}\right)^{-1} \mathbf{T_a}^T$$

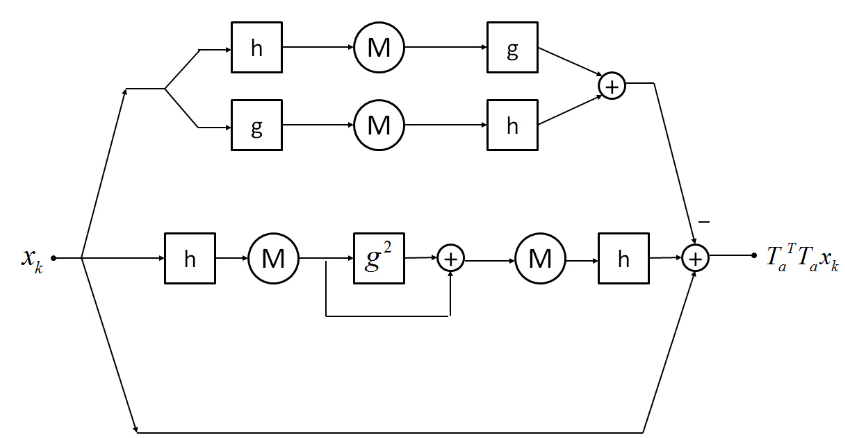
Let's try to use only filters

Define iteratively, through descent on LS:

$$\arg\min_x \|\mathbf{T_a} x - y\|_2^2 \longrightarrow \hat{x}_{k+1} = \hat{x}_k + \tau \mathbf{T_a}^T (y - \mathbf{T_a} \hat{x}_k)$$

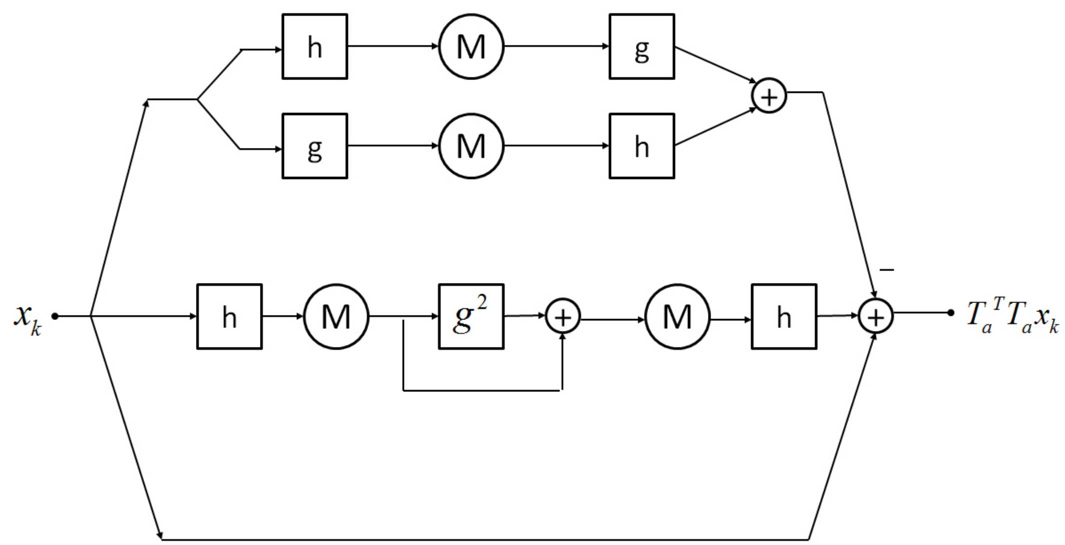
$$\mathbf{T_a}^T = (\mathbf{H_m}^T \quad \mathbf{I} - \mathbf{H_m}^T \mathbf{G}^T)$$





The Laplacian Pyramid

we can easily implement $\mathbf{T_a}^T\mathbf{T_a}$ with filters and masks:



With the real symmetric matrix $\mathbf{Q} = \mathbf{T_a}^T\mathbf{T_a}$ and $b = \mathbf{T_a}^T y$

$$x_N = \tau \sum_{j=0}^{N-1} (\mathbf{I} - \tau\mathbf{Q})^j b$$

Use Chebyshev approximation of: $L(\omega) = \tau \sum_{j=0}^{N-1} (1 - \tau\omega)^j$

Kron Reduction

In order to iterate the construction, we need to construct a graph on the reduced vertex set.

$$\mathbf{A}_{\mathrm{r}} = \mathbf{A}[\alpha, \alpha] - \mathbf{A}[\alpha, \alpha)\mathbf{A}(\alpha, \alpha)^{-1}\mathbf{A}(\alpha, \alpha]$$

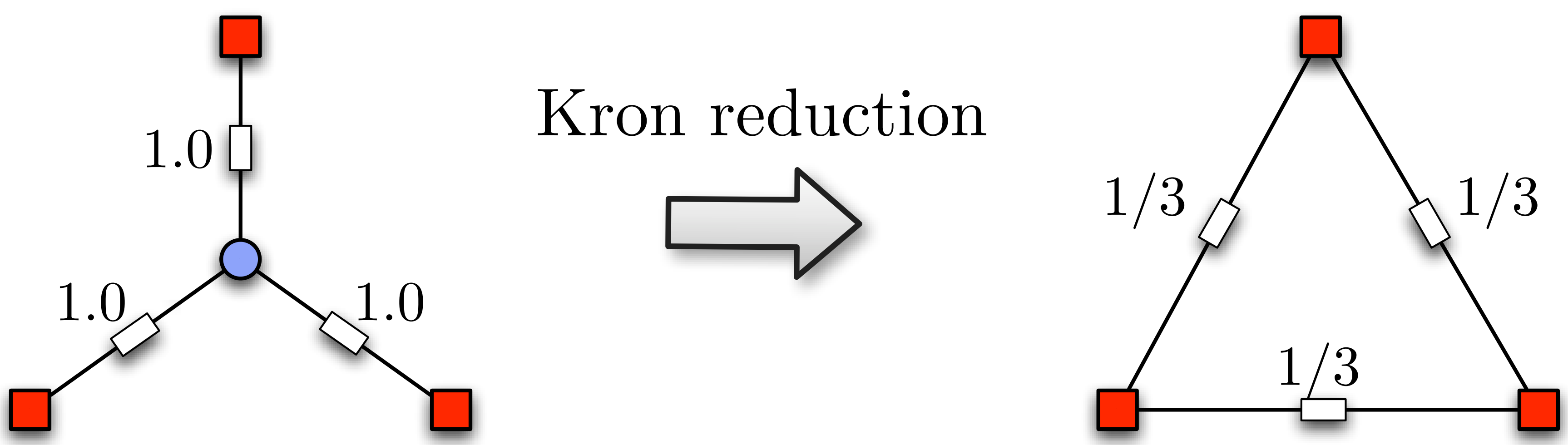
$$\mathbf{A} = \begin{bmatrix} \mathbf{A}[\alpha, \alpha] & \mathbf{A}[\alpha, \alpha) \\ \mathbf{A}(\alpha, \alpha] & \mathbf{A}(\alpha, \alpha) \end{bmatrix}$$

Kron Reduction

In order to iterate the construction, we need to construct a graph on the reduced vertex set.

$$\mathbf{A}_{\mathrm{r}} = \mathbf{A}[\alpha, \alpha] - \mathbf{A}[\alpha, \alpha)\mathbf{A}(\alpha, \alpha)^{-1}\mathbf{A}(\alpha, \alpha]$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}[\alpha, \alpha] & \mathbf{A}[\alpha, \alpha) \\ \mathbf{A}(\alpha, \alpha] & \mathbf{A}(\alpha, \alpha) \end{bmatrix}$$



[Dorfler et al, 2011]

Kron Reduction

In order to iterate the construction, we need to construct a graph on the reduced vertex set.

$$\mathbf{A}_{\mathrm{r}} = \mathbf{A}[\alpha,\alpha] - \mathbf{A}[\alpha,\alpha)\mathbf{A}(\alpha,\alpha)^{-1}\mathbf{A}(\alpha,\alpha]$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}[\alpha,\alpha] & \mathbf{A}[\alpha,\alpha) \\ \mathbf{A}(\alpha,\alpha] & \mathbf{A}(\alpha,\alpha) \end{bmatrix}$$

Properties:

maps a weighted undirected laplacian to a weighted undirected laplacian

spectral interlacing (spectrum does not degenerate)

$$\lambda_k(\mathbf{A}) \leq \lambda_k(\mathbf{A}_r) \leq \lambda_{k+n-|\alpha|}(\mathbf{A})$$

disconnected vertices linked in reduced graph IFF there is a path that runs only through eliminated nodes

Example

Note: For a k-regular bipartite graph

$$\mathbf{L} = \begin{bmatrix} k\mathbf{I}_n & -\mathbf{A} \\ -\mathbf{A}^T & k\mathbf{I}_n \end{bmatrix}$$

Kron-reduced Laplacian: $\mathbf{L}_r = k^2\mathbf{I}_n - \mathbf{A}\mathbf{A}^T$

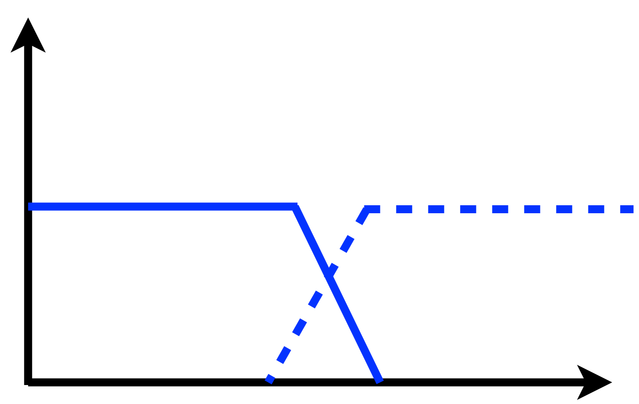
Example

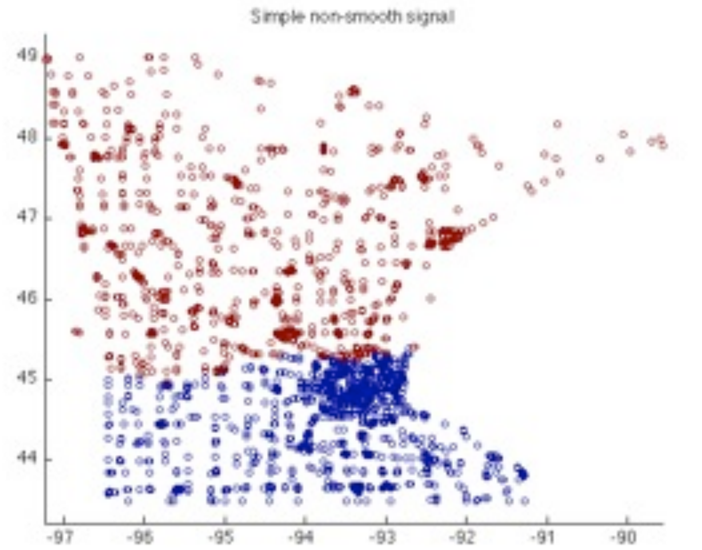
Note: For a k-regular bipartite graph

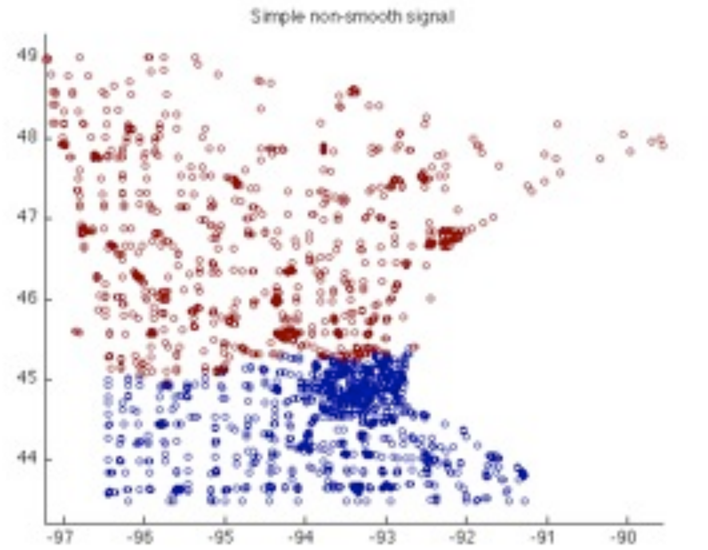
$$\mathbf{L} = \begin{bmatrix} k\mathbf{I}_n & -\mathbf{A} \\ -\mathbf{A}^T & k\mathbf{I}_n \end{bmatrix}$$

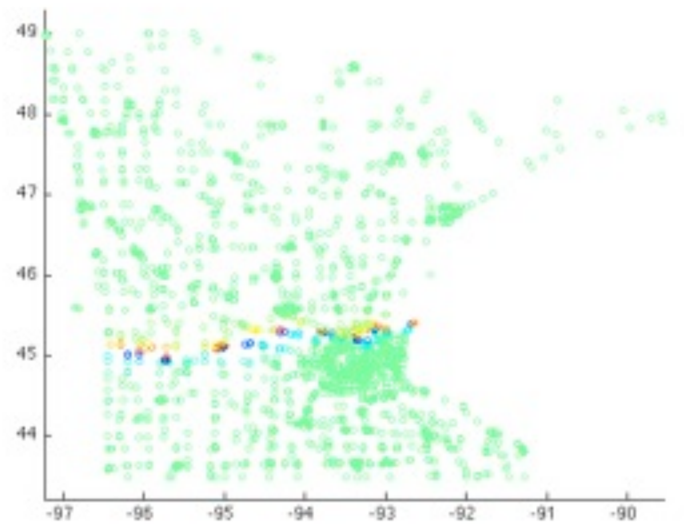
Kron-reduced Laplacian: $\mathbf{L}_r = k^2\mathbf{I}_n - \mathbf{A}\mathbf{A}^T$

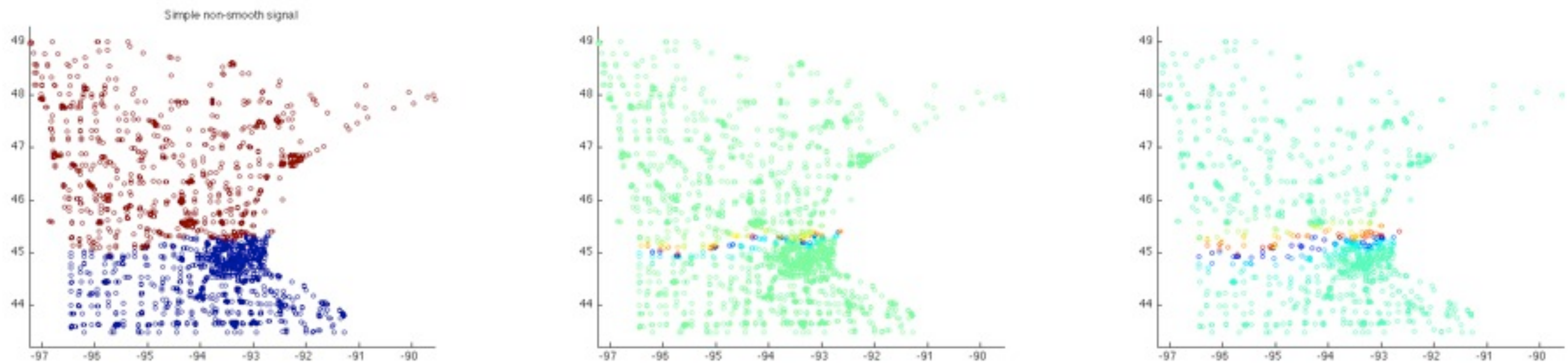
$$\hat{f}_r(i) = \hat{f}(i) + \hat{f}(N-i) \quad i = 1, ..., N/2$$

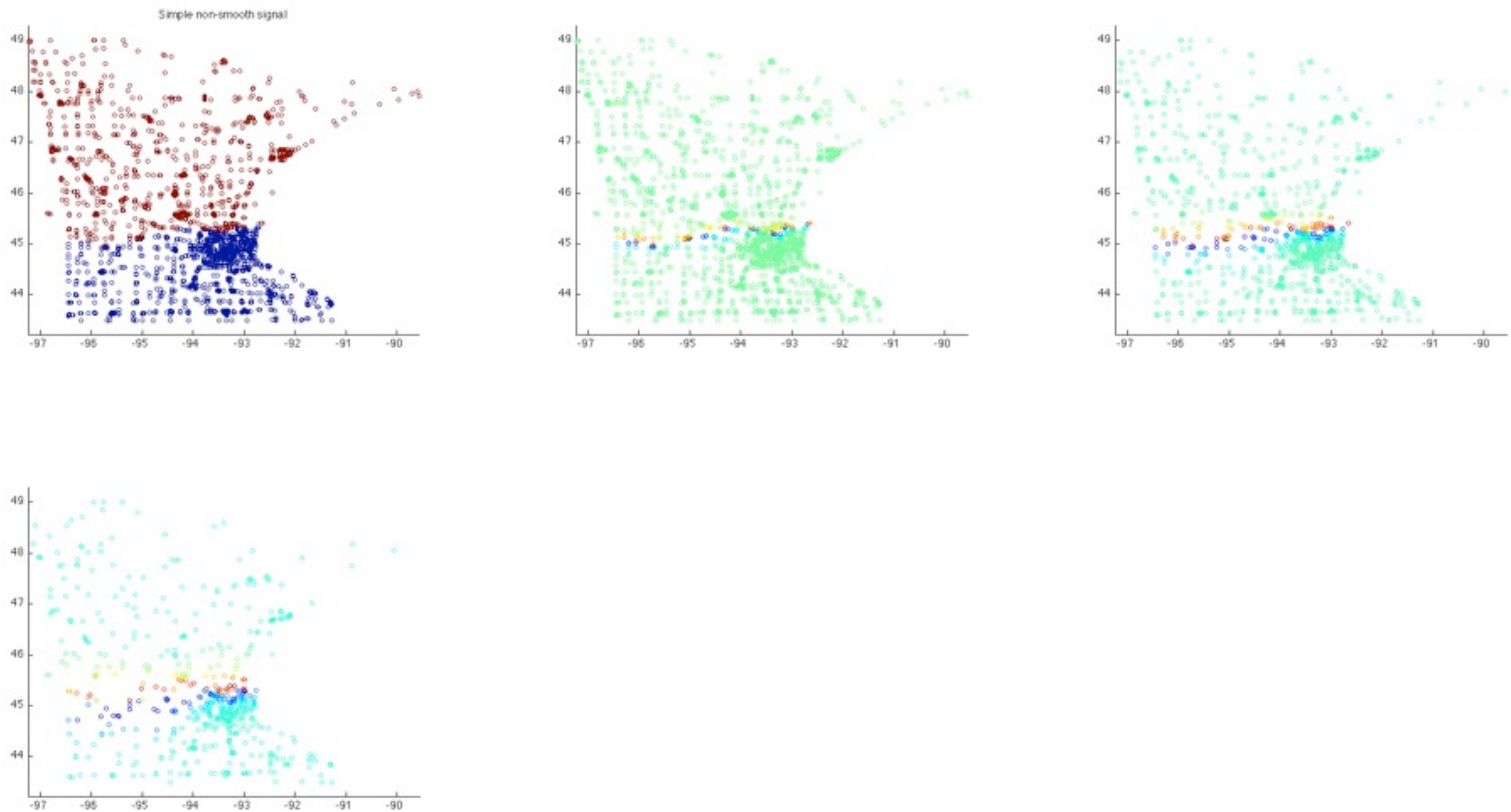


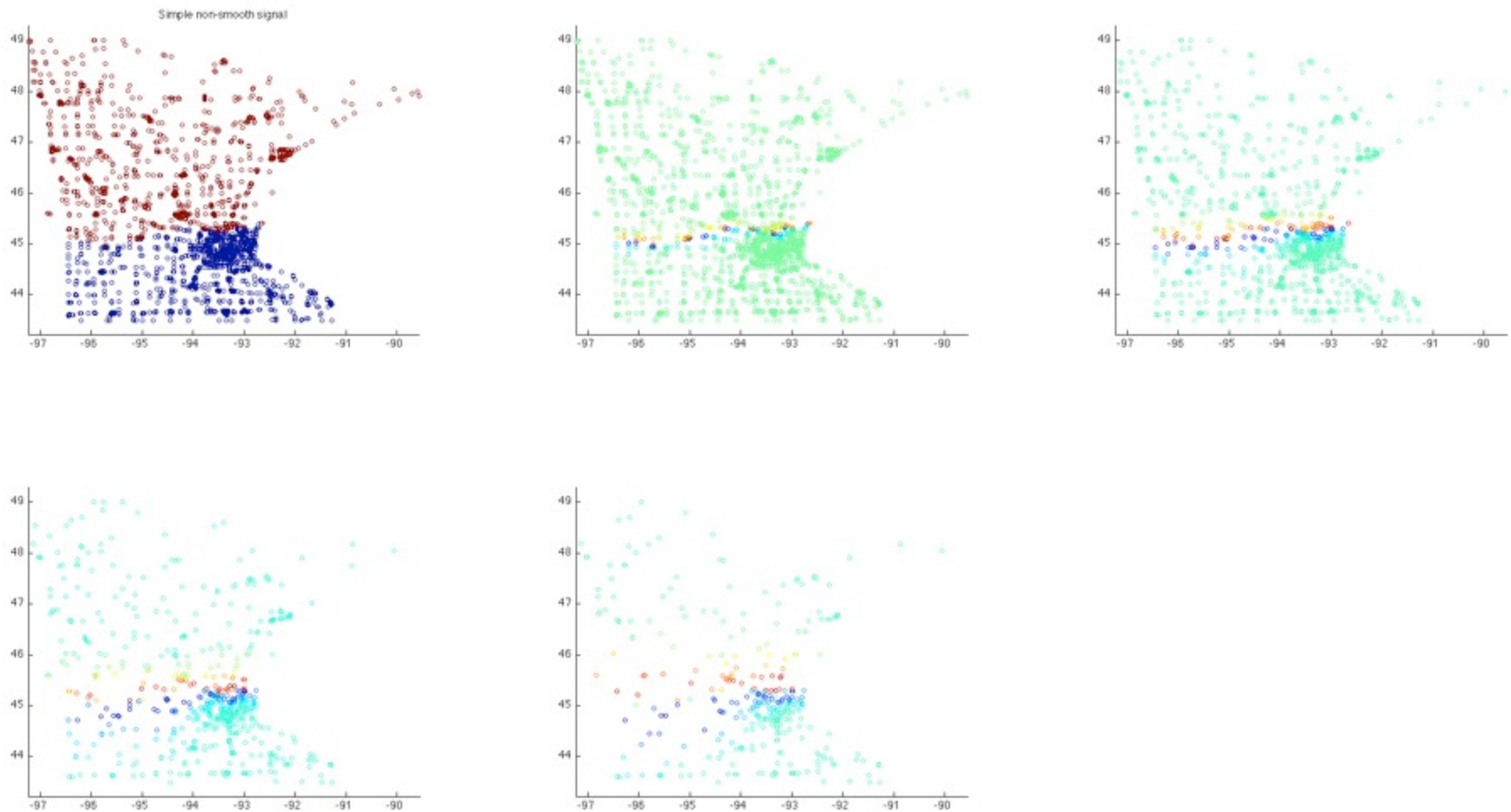

Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

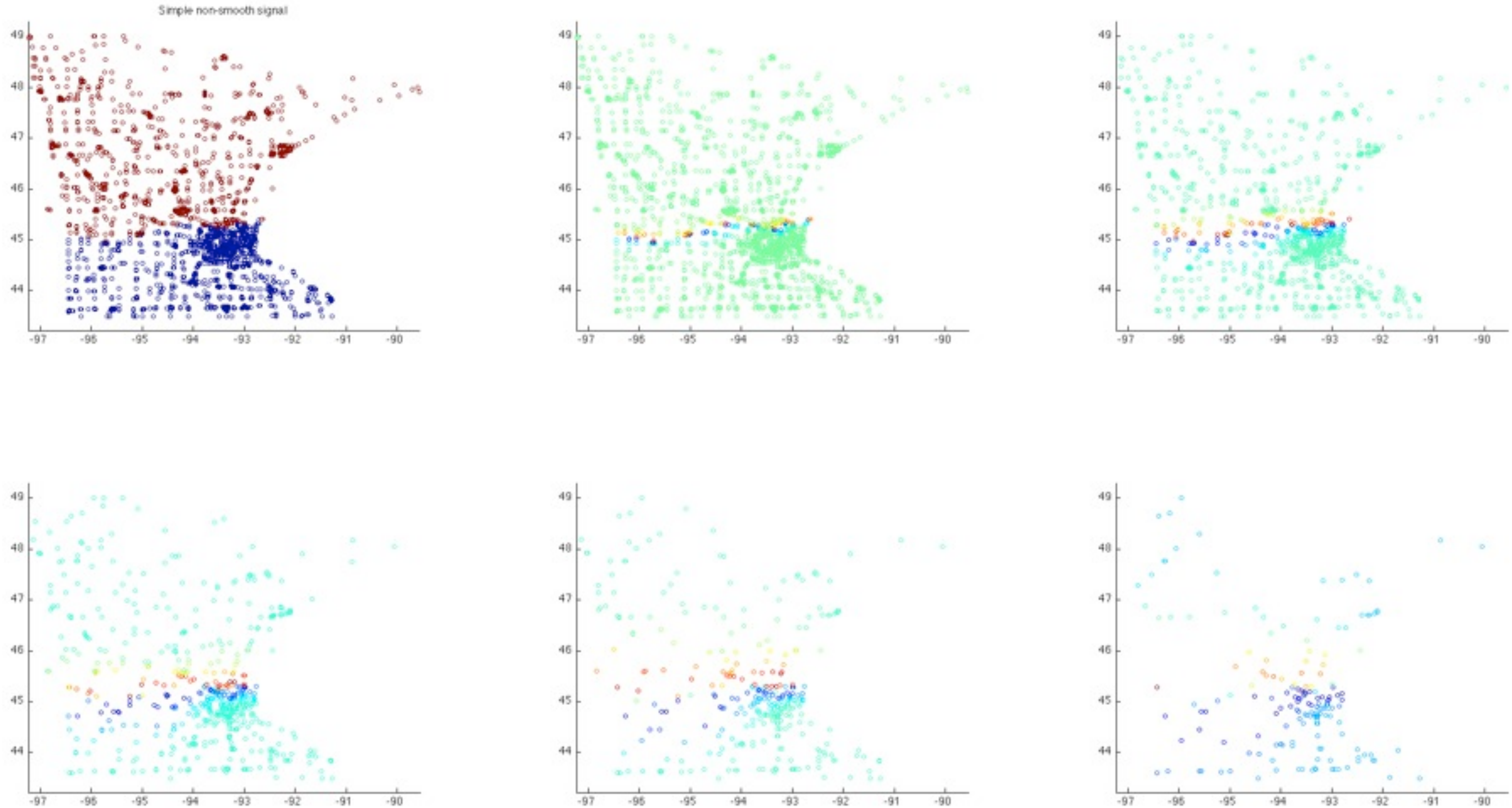

Simple non-smooth signal




Simple non-smooth signal
49
48
47
46
45
44
-97
-96
-95
-94
-93
-92
-91
-90

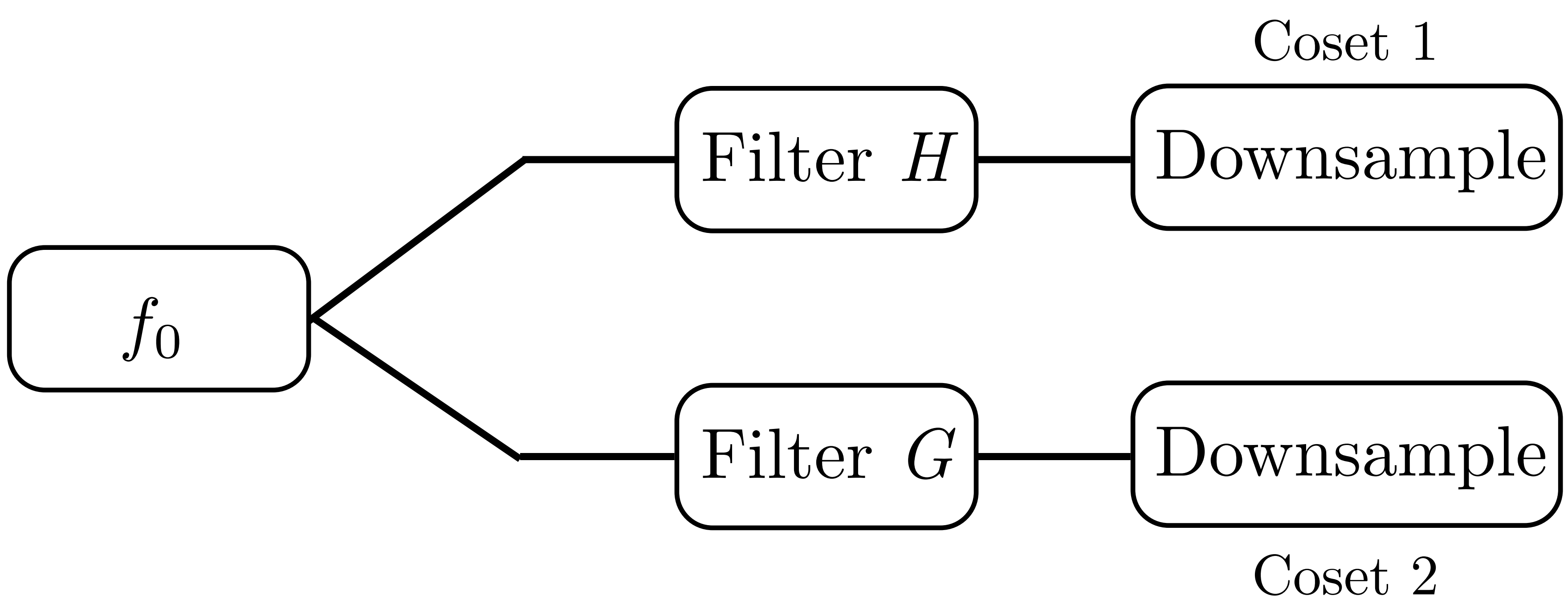

Simple non-smooth signal


Simple non-smooth signal


Simple non-smooth signal

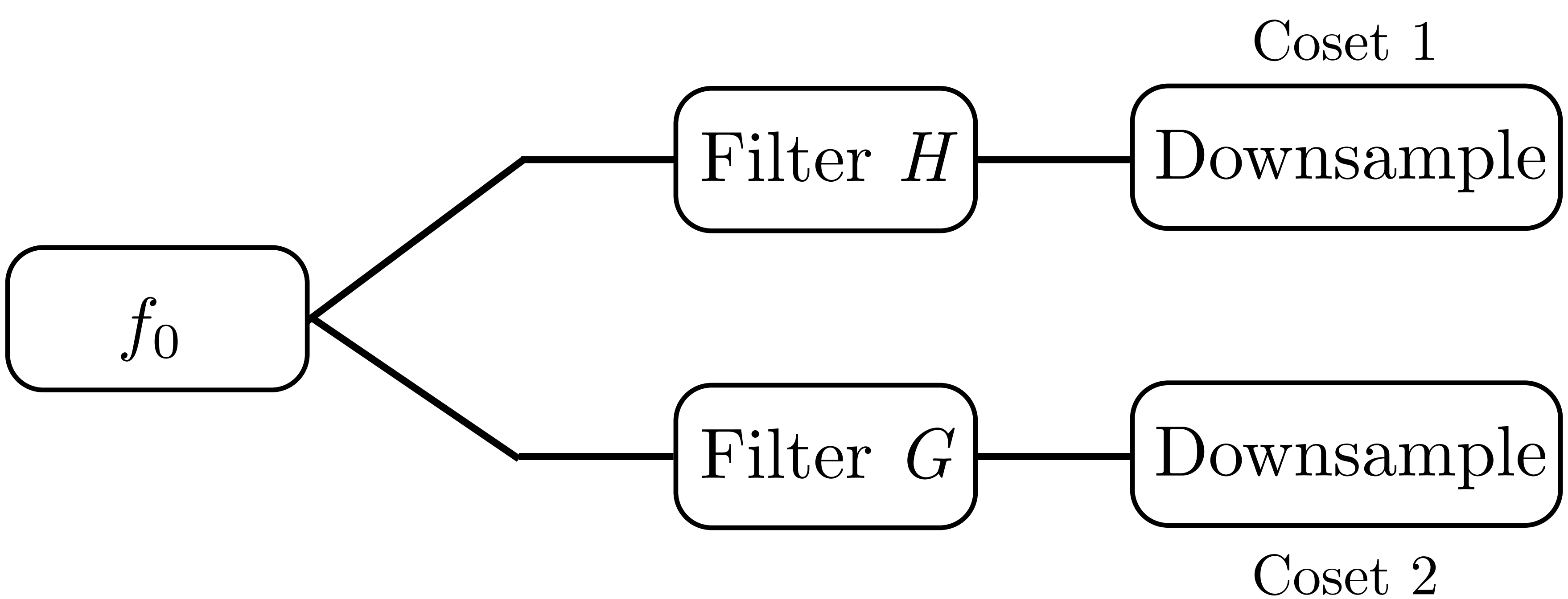
Filter Banks

2 critically sampled channels



Filter Banks

2 critically sampled channels



Theorem: For a k-RBG, the filter bank is perfect-reconstruction IFF

$$|H(i)|^2 + |G(i)|^2 = 2$$

$$H(i)G(N-i) + H(N-i)G(i) = 0$$

Conclusions

- Structured, data dependent dictionary of wavelets
  - sparsity and smoothness on graph are merged in simple and elegant fashion
  - fast algo, clean problem formulation
  - graph structure can be totally hidden in wavelets
- Filter banks based on nodal domains or coloring
  - Universal algo based on filtering and Kron reduction
  - Efficient IFF *some* structure in the graph
  - Unfortunately no closed form theory in general

Wavelet Ingredients

Wavelet transform based on two operations:

Dilation (or scaling) and Translation (or localization)

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

Wavelet Ingredients

Wavelet transform based on two operations:

Dilation (or scaling) and Translation (or localization)

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$(T^s f)(a) = \int \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)dx \qquad (T^s f)(a) = \langle \psi_{(s,a)}, f \rangle$$

Wavelet Ingredients

Wavelet transform based on two operations:

Dilation (or scaling) and Translation (or localization)

$$\psi_{s,a}(x) = \frac{1}{s}\psi\left(\frac{x-a}{s}\right)$$

$$(T^s f)(a) = \int \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right) f(x)dx \qquad (T^s f)(a) = \langle \psi_{(s,a)}, f\rangle$$

Equivalently: $$(T^s \delta_a)(x) = \frac{1}{s}\psi^*\left(\frac{x-a}{s}\right)$$

$$(T^s f)(x) = \frac{1}{2\pi}\int e^{i\omega x}\hat{\psi}^*(s\omega)\hat{f}(\omega)d\omega$$

Graph Laplacian and Spectral Theory

$G = (V, E, w)$ weighted, undirected graph

Non-normalized Laplacian: $\mathcal{L} = D - A$ Real, symmetric

$$(\mathcal{L}f)(i) = \sum_{i \sim j} w_{i,j}(f(i) - f(j))$$

Why Laplacian ?

Graph Laplacian and Spectral Theory

$G = (V, E, w)$ weighted, undirected graph

Non-normalized Laplacian: $\mathcal{L} = D - A$ Real, symmetric

$$(\mathcal{L}f)(i) = \sum_{i \sim j} w_{i,j}(f(i) - f(j))$$

Why Laplacian ? $\mathbb{Z}^2$ with usual stencil

$$(\mathcal{L}f)_{i,j} = 4f_{i,j} - f_{i+1,j} - f_{i-1,j} - f_{i,j+1} - f_{i,j-1}$$

In general, graph laplacian from nicely sampled manifold converges to Laplace-Beltrami operator

Graph Laplacian and Spectral Theory

$$\frac{d^2}{dx^2} \quad \Rightarrow \quad e^{i\omega x} \quad \Rightarrow \quad f(x) = \frac{1}{2\pi}\int \hat{f}(\omega) e^{i\omega x} d\omega$$

Graph Laplacian and Spectral Theory

$$\frac{d^2}{dx^2} \Rightarrow e^{i\omega x} \Rightarrow f(x) = \frac{1}{2\pi}\int \hat{f}(\omega)e^{i\omega x}d\omega$$

Eigen decomposition of Laplacian: $\mathcal{L}\phi_l = \lambda_l \phi_l$

Graph Laplacian and Spectral Theory

$$\frac{d^2}{dx^2} \Rightarrow e^{i\omega x} \Rightarrow f(x) = \frac{1}{2\pi}\int \hat{f}(\omega)e^{i\omega x}d\omega$$

Eigen decomposition of Laplacian: $\mathcal{L}\phi_l = \lambda_l \phi_l$

For simplicity assume connected graph and $0 = \lambda_0 < \lambda_1 \leq \lambda_2 ... \leq \lambda_{N-1}$

For any function on the vertex set (vector) we have:

$$\hat{f}(\ell) = \langle \phi_\ell, f \rangle = \sum_{i=1}^{N} \phi_\ell^*(i) f(i)$$ Graph Fourier Transform

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell)\phi_\ell(i)$$

Spectral Graph Wavelets

Remember good old Euclidean case:

$$(T^s f)(x) = \frac{1}{2\pi} \int e^{i\omega x} \hat{\psi}^*(s\omega) \hat{f}(\omega) d\omega$$

We will adopt this operator view

Spectral Graph Wavelets

Remember good old Euclidean case:

$$(T^s f)(x) = \frac{1}{2\pi} \int e^{i\omega x} \hat{\psi}^*(s\omega) \hat{f}(\omega) d\omega$$

We will adopt this operator view

Operator-valued function via continuous *Borel functional calculus*

$g : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ $\qquad$ $T_g = g(\mathcal{L})$ **Operator-valued function**

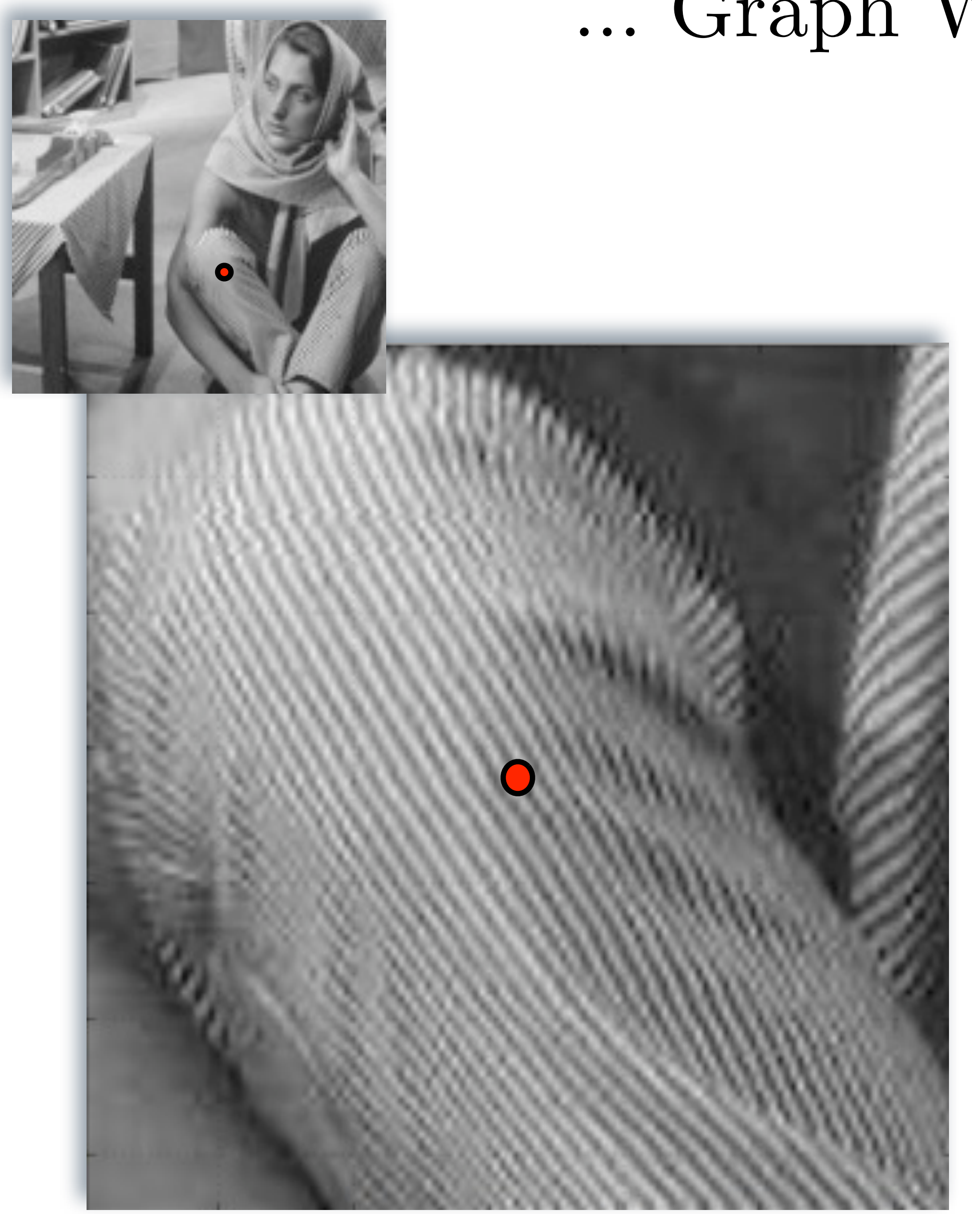
Action of operator is induced by its Fourier symbol

$$\widehat{T_g f}(\ell) = g(\lambda_\ell) \hat{f}(\ell) \qquad (T_g f)(i) = \sum_{\ell=0}^{N-1} g(\lambda_\ell) \hat{f}(\ell) \phi_\ell(i)$$

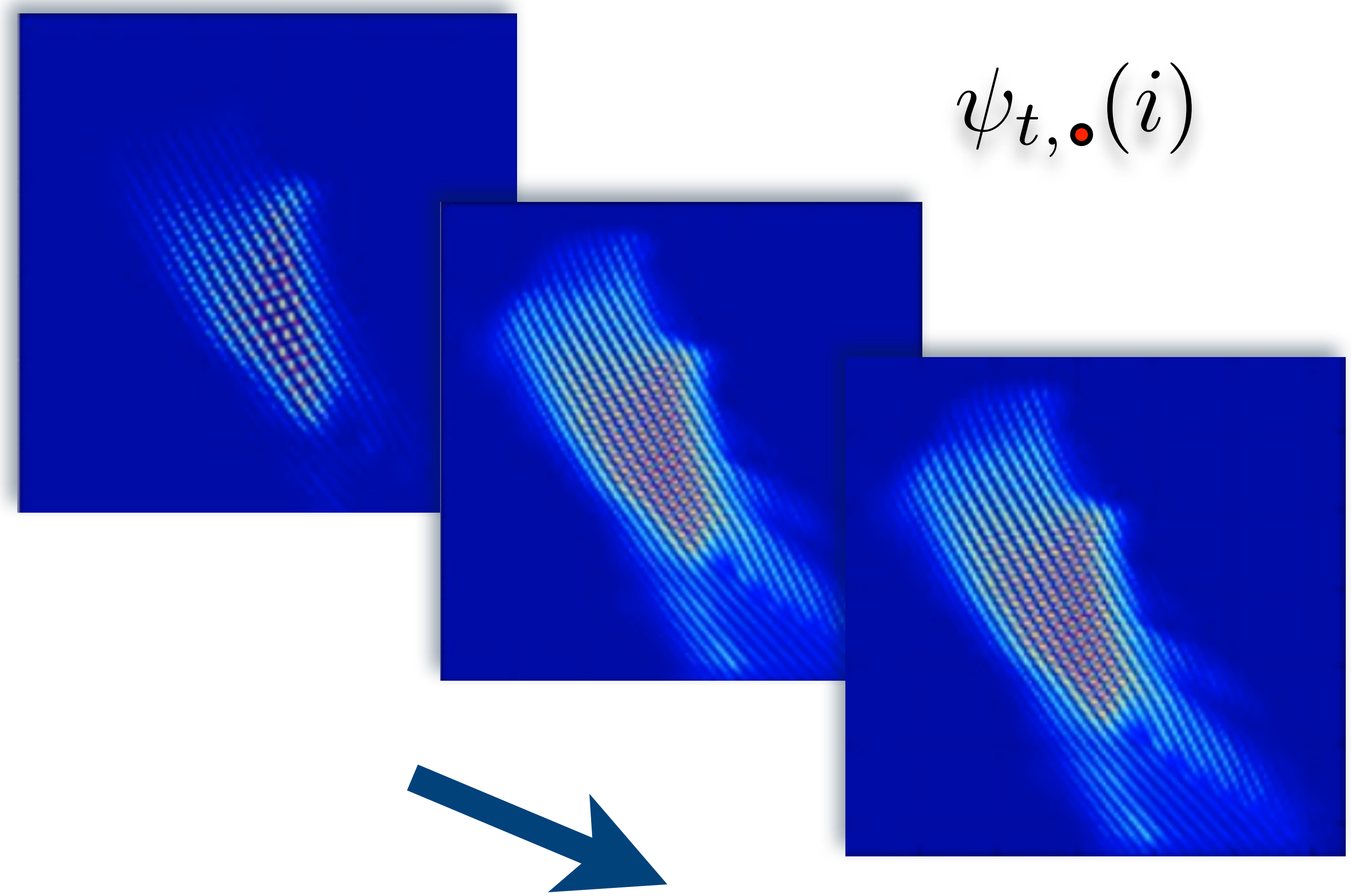
Non-local Wavelet Frame

- Non-local Wavelets are ...

... Graph Wavelets on Non-Local Graph



$\psi_{t,\bullet}(i)$



increasing scale

Interest: good *adaptive* sparsity basis

Distributed Computation

Scenario: Network of N nodes, each knows

- local data f(n)
- local neighbors
- M Chebyshev coefficients of wavelet kernel
- A global upper bound on largest eigenvalue of graph laplacian

Distributed Computation

Scenario: Network of N nodes, each knows

- local data f(n)
- local neighbors
- M Chebyshev coefficients of wavelet kernel
- A global upper bound on largest eigenvalue of graph laplacian

To compute: $$(\tilde{\mathbf{\Phi}} f)_{(j-1)N+n} = \Big(\frac{1}{2}c_{j,0}f + \sum_{k=1}^{M} c_{j,k}\overline{T}_k(\mathcal{L})f\Big)_n$$

Distributed Computation

Scenario: Network of N nodes, each knows

- local data f(n)
- local neighbors
- M Chebyshev coefficients of wavelet kernel
- A global upper bound on largest eigenvalue of graph laplacian

To compute: $$(\tilde{\mathbf{\Phi}} f)_{(j-1)N+n} = \left(\frac{1}{2} c_{j,0} f + \sum_{k=1}^{M} c_{j,k} \overline{T}_k(\mathcal{L}) f\right)_n$$

$$\left(\overline{T}_1(\mathcal{L}) f\right)_n = \left(\frac{2}{\alpha}(\mathcal{L} - \alpha I) f\right)_n$$

sensor only needs f(n) from its neighbors

Distributed Computation

Scenario: Network of N nodes, each knows

- local data f(n)
- local neighbors
- M Chebyshev coefficients of wavelet kernel
- A global upper bound on largest eigenvalue of graph laplacian

To compute: $$(\tilde{\mathbf{\Phi}} f)_{(j-1)N+n} = \left(\frac{1}{2} c_{j,0} f + \sum_{k=1}^{M} c_{j,k} \overline{T}_k(\mathcal{L}) f\right)_n$$

$$\left(\overline{T}_1(\mathcal{L}) f\right)_n = \left(\frac{2}{\alpha}(\mathcal{L} - \alpha I) f\right)_n$$

sensor only needs f(n) from its neighbors

$$\left(\overline{T}_k(\mathcal{L}) f\right) = \frac{2}{\alpha}(\mathcal{L} - \alpha I)\left(\overline{T}_{k-1}(\mathcal{L}) f\right) - \overline{T}_{k-2}(\mathcal{L}) f$$

Computed by exchanging last computed values

Distributed Computation

Communication cost: 2M|E| messages of length 1 per node

Example: distributed denoising, or distributed regression, with Lasso

$$\arg\min_a \frac{1}{2}\|y - \mathbf{\Phi}^* a\|_2^2 + \|a\|_{1,\mu}$$

$$a_i^{(k)} = \mathcal{S}_{\mu_i,\tau}\left(\left[a^{k-1} + \tau\mathbf{\Phi}(y - \mathbf{\Phi}^* a^{k-1})\right]_i\right)$$

$$\mathcal{S}_{\mu_i\tau}(z) := \begin{cases} 0 & , \text{ if } \mid z \mid\leq \mu_i\tau \\ z - \text{sgn}(z)\mu_i\tau & , \text{ o.w.} \end{cases}$$

Total communication cost:

Distributed Lasso [Mateos, Bazerque, Gianakis] Cost $\sim |E|N$

Chebyshev $\mathbf{\Phi} y$ 2M|E| messages of length 1

$\mathbf{\Phi\Phi}^* a$ 4M|E| messages of length J+1

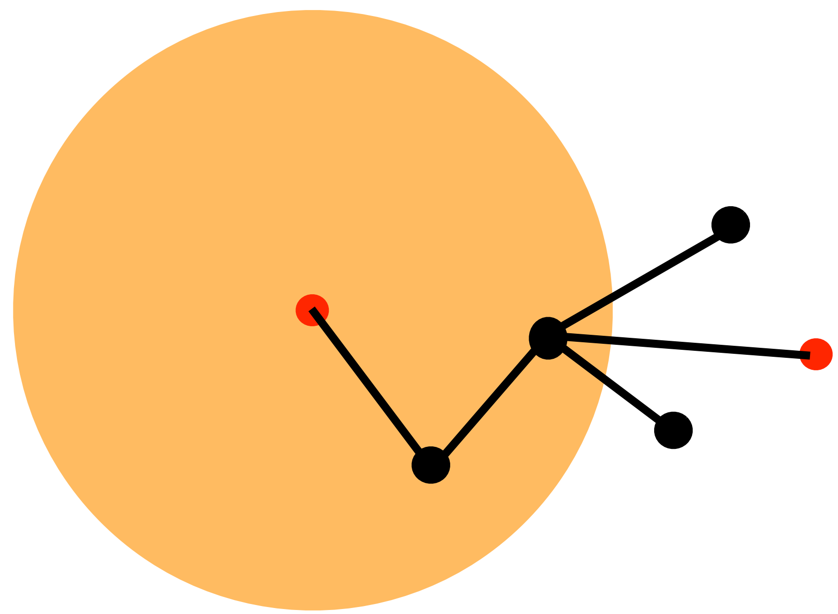
Cost $\sim |E|$

Wavelets on Graphs ?

- Existing constructions
  - wavelets on meshes (computer graphics, numerical analysis), often via lifting
  - diffusion wavelets [Maggioni, Coifman & others]
  - recently several other constructions based on "organizing" graph in a multiscale way [Gavish-Coifman]
- Goal
  - process signals on graphs
  - retain simplicity and signal processing flavor
  - algorithm to handle fairly large graphs

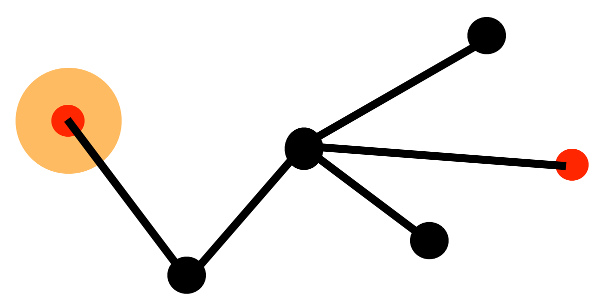
Scaling & Localization

Effect of operator dilation ?



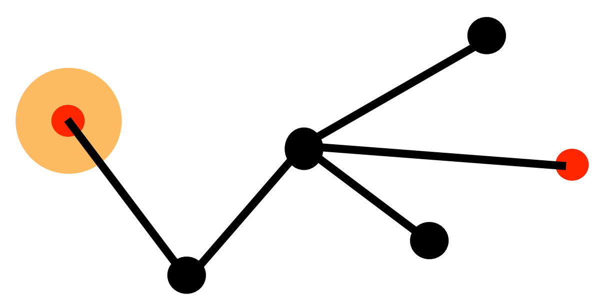
Scaling & Localization

Effect of operator dilation ?



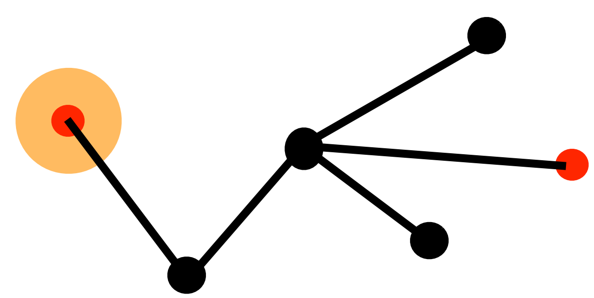
Scaling & Localization

Effect of operator dilation ?



Scaling & Localization

Effect of operator dilation ?



Theorem: $d_G(i,j) > K$ and g has K vanishing derivatives at *0*

$$\frac{\psi_{t,j}(i)}{\|\psi_{t,j}\|} \leq Dt$$ for any t smaller than a critical scale function of $d_G(i,j)$

Reason ? At small scale, wavelet operator behaves like power of Laplacian