Autoregressive Moving Average Graph Filters

Geert Leus

Thanks:
Elvin Isufi, Andreas Loukas, Andrea Simonetto, Jiani Liu

Delft University of Technology
g.j.t.leus@tudelft.nl
Signal Processing over Graphs

Sensor networks (temperature, pollution)

Brain networks (fMRI time series)

Transport networks (number of vehicles crossing the junction)
Graph Filters

Denoising signals (e.g., Tikhonov, Wiener)

Interpolation (e.g., missing values, semi-supervised learning)

Analysing/performing diffusion
Roadmap

• Preliminaries
• Graph filters
• FIR and IIR/ARMA graph filters
  – Design and implementation
• Distributed ARMA implementation
  – Using heat kernels
  – Direct implementation
• 2-dimensional time-graph filters
• Stochastic analysis of graph filters
• Conclusions
Preliminaries

- Undirected graph $G = (\mathcal{V}, \mathcal{E})$
- $N$ nodes
- Graph signal $\mathbf{x}_t$
- Symmetric shift matrix $\mathbf{S}$
  - adjacency, Laplacian, translations, e.g., $\mathbf{S} = \mathbf{L}_n - \mathbf{I}$
- Graph Fourier transformation

  Eigenvalue decomposition

  $$\mathbf{S} = \mathbf{U} \Lambda \mathbf{U}^T \quad \Lambda = \text{diag}\{\lambda_n\}$$

  Transform and inverse transform

  $$\hat{\mathbf{x}}_t = \mathbf{U}^T \mathbf{x}_t \quad \mathbf{x}_t = \mathbf{U} \hat{\mathbf{x}}_t$$
Graph Filters

- Time-invariant:
  \[ x_t = x \]

- Graph filter:
  \[ \hat{y}_n = h(\lambda_n)\hat{x}_n \]
  \[ \hat{y} = h(\Lambda)\hat{x} \]
  \[ h(\Lambda) = \text{diag}\{h(\lambda_n)\} \]
  \[ y = U h(\Lambda) U^T x = Hx \]

- Universal graph filter (graph independent)
  \[ h(\lambda) \text{ designed for } [\lambda_{\text{min}}, \lambda_{\text{max}}] \]

[Shuman’11, DCOSS]
[Sandryhaila’13, TSP]
[Shuman’13, SPM]
FIR Graph Filters

\[ y = Hx \quad \text{for} \quad H = \sum_{k=0}^{K} \phi_k S^k \]

Frequency response
\[ h(\lambda_n) = \sum_{k=0}^{K} \phi_k \lambda_n^k \]

- There are efficient filter implementations ☺
- Distributed implementation \( S^k x = S (S^{k-1} x) \) ☺
- No stability issues ☺
- High approximation requires higher orders ☹

[Sandryhaila’13, TSP]
FIR Design

- Minimize the error
  \[ e_n = \hat{h}_n - \sum_{k=0}^{K} \phi_k \lambda_k^n \]

- Least squares: [Sandryhaila’13, TSP]
  \[ e = \hat{h} - \Psi_{K+1} \Phi \]
  \[ \min_{\Phi} \| \hat{h} - \Psi_{K+1} \Phi \|^2 \]
  \[ [\Psi_{K+1}]_{n,k} = \lambda_{n-1}^k \]
  \[ \Psi_{K+1} \in \mathbb{R}^{N \times (K+1)} \]

- Chebyshev: [Shuman’11, DCOSS]
  \[ \hat{h}(\lambda) = \sum_{k=0}^{\infty} c_k T_k(\lambda) \approx \sum_{k=0}^{K} c_k T_k(\lambda) \]
  - Closed form expressions for \( c_k \)
  - \( T_k(\lambda) \): modified Chebyshev polynomials; orthogonal over \([\lambda_{min}, \lambda_{max}]\)
**FIR Extensions**

- **FIR**: \( y = Hx \) for \( H = \sum_{k=0}^{K} \phi_k S^k \)

- **Node-varying graph filters** [Segarra’15, arXiv], [Segarra’17, TSP]
  \[ H = \sum_{k=0}^{K} \text{diag}\{\phi_k\} S^k \]

- **Nonlinear graph filters**
  - Weighted median graph filters [Segarra’16, GlobalSIP]
  - Activation functions (neural networks)
  - Volterra-like graph filters

- **IIR - ARMA** [Loukas’15, SPL], [Shi’15, SPL], [Isufi’17, TSP]
ARMA Graph Filters

\[ y = Hx \quad \text{for} \quad H = \left( I + \sum_{p=1}^{P} \psi_p S^p \right)^{-1} \left( \sum_{q=0}^{Q} \phi_q S^q \right) \]

- Frequency response

\[ h(\lambda_n) = \frac{\sum_{q=0}^{Q} \phi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p} \]

- Filter design is more involved than for FIR 😞
- Does not admit trivial distributed implementation 😞
- Stability is guaranteed by invertibility 😊
- Exact solution for denoising/interpolation/diffusion 😊
**ARMA Design**

- Minimize the error
  \[ e_n = \hat{h}_n - \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p} \]

- Prony’s method:
  \[ e'_n = \hat{h}_n \alpha_n - \beta_n \]

\[ e' = \hat{h} \circ \alpha - \beta \]

- Modified error is linear in \( \psi, \varphi \)

\[ \min_{\psi, \varphi} \| \hat{h} \circ (\Psi_{P+1} \psi) - \Psi_{Q+1} \varphi \|^2 \]

\[ \alpha = \Psi_{P+1} \psi, \quad \psi_0 = 1 \]

\[ \beta = \Psi_{Q+1} \varphi \]

\[ [\Psi_{P+1}]_{n,p} = \lambda_n^{p-1} \]

\[ [\Psi_{Q+1}]_{n,q} = \lambda_n^{q-1} \]
ARMA Design

- Minimize the error \( e_n = \hat{h}_n - \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p} \)

- Iterative method:

\[
e_n = (\hat{h}_n \alpha_n - \beta_n) \gamma_n, \quad \gamma_n = 1/\alpha_n
\]

\[e = (\hat{h} \circ \alpha - \beta) \circ \gamma\]

For known \( \gamma \), error is linear in \( \psi, \varphi \)

\[
\min_{\psi_i, \varphi_i} \| \gamma_{i-1} \circ [\hat{h} \circ (\Psi_{P+1} \psi_i) - \Psi_{Q+1} \varphi_i] \|^2
\]

\[
\alpha = \Psi_{P+1} \psi, \quad \psi_0 = 1
\]

\[
\beta = \Psi_{Q+1} \varphi
\]

\[
[\Psi_{P+1}]_{n,p} = \lambda_{n}^{p-1}
\]

\[
[\Psi_{Q+1}]_{n,q} = \lambda_{n}^{q-1}
\]
Numerical Results

Accuracy

- **Approximate universally** an ideal filter with $S = L_n$, $\lambda_c = 1$

![Graph showing NMSE vs Iteration index for ARMA(4,9) and ARMA(9,10) on a uniform grid.](image)

Few iterations are enough to reach the steady state.

$$P + Q \leq K$$

![Graph showing NMSE vs K for various methods like Prony's project, Prony's LS, FIR, and Iterative approach on a uniform grid with real frequencies.](image)
Numerical Results

Linear prediction

- Molene data set
- **Forward**: minimize residual energy \( r = x - h(S)x \)
- **Quantize** with \( B \) bits:
  \( \tilde{r} = Q(r) \)
- **Synthesis**: reconstruct from residual
  \( \tilde{x} = (I - h(S))^{-1}\tilde{r} \)

**FIR**

\[
h(S) = \sum_{k=1}^{K} \phi_k S^k
\]

**ARMA**

\[
h(S) = \left( I + \sum_{p=1}^{P} \psi_p S^p \right)^{-1} \left( \sum_{q=1}^{Q} \varphi_q S^p \right)
\]

- Similar ideas can be used to solve this, but data-driven:
  - **FIR**: least squares
  - **ARMA**: Prony’s method, iterative method

[Sandryhaila’13, TSP]
Numerical Results

Linear prediction

ARMA can improve linear prediction up to 1 order

Preferred filter orders are small for all B
ARMA Implementation

\[ y = \left( I + \sum_{p=1}^{P} \psi_p S^p \right)^{-1} \left( \sum_{q=0}^{Q} \varphi_q S^q \right) x \]

\[ y = P^{-1}(Qx) \]

Inversion of \( P \):

- Gradient descent  \([Shi'15, SPL]\)
- Conjugate gradient (CG)
  - For sparse graphs this can run fast
  - In practice CG is arrested after \( T \) iterations
- Any other Krylov-based inversion method can be used
Numerical Tests

- Implementation on Erdős-Rényi $N = 100, \ S = L_n, \ \lambda_c = 1$
- CG is arrested after $T$ such that $PT + Q \leq K$
- Heuristic search for $P, Q, T$

With similar complexity, ARMA does a bit better
With higher complexity it improves even more
Distributed ARMA Through Heat Kernel

- Distributed ARMA of order 1: ARMA₁ (cf. heat kernel)
  \[ y_{t+1} = \psi S y_t + \varphi x \]
  \[ y_t = (\psi S)^t y_0 + \varphi \sum_{\tau=0}^{t-1} (\psi S)^\tau x \]

- For \( \|S\|_2 < \rho \) and \( |\psi \rho| < 1 \) we obtain at steady state
  \[ y = \lim_{t \to \infty} y_t = \varphi \sum_{\tau=0}^{\infty} (\psi S)^\tau x = \varphi \left( I - \psi S \right)^{-1} x \]

- The graph frequency response is
  \[ h(\lambda_n) = \frac{r}{\lambda_n - p} \quad \text{s.t.} \quad |p| > \rho \]
  \[
  \begin{cases}
    r = -\varphi / \psi \\
    p = 1 / \psi
  \end{cases}
  \]
Distributed ARMA Through Heat Kernel

- Distributed ARMA of order $K$ : $\text{ARMA}_K$

$$y_t = \sum_{k=1}^{K} y^{(k)}_t$$

- The graph frequency response is

$$h(\lambda) = \sum_{k=0}^{K} \frac{r_k}{\lambda - p_k} \quad \text{s.t.} \quad |p_k| > \rho \quad \begin{cases} r_k = -\varphi^{(k)}/\psi^{(k)} \\ p_k = 1/\psi^{(k)} \end{cases}$$

constraints on poles
Application to Tikhonov Denoising

• Noisy smooth signal \( x = t + n \)

The Tikhonov denoised signal is

\[
t^* = \arg \min_x \| t - x \|^2_2 + wt^T S^K t
\]

with optimal solution

\[
x^* = (I + wS^K)^{-1} x
\]

• Exact solution is obtained with \( H = (I + wS^K)^{-1} \)

• Can be solved distributively via ARMA\(_K\)

• For \( K = 1, 2 \) and \( S = L_n - I \) the filter is always convergent
Application to Wiener Graph Filtering

• Noisy signal $x = t + n$, with $t \sim \mathcal{P}(0, \Sigma_t)$, $n \sim \mathcal{P}(0, \Sigma_n)$

The linear MMSE solves

$$H^* = \arg\min_H \mathbb{E} \left[ \| Hx - t \|_2^2 \right] \quad \text{and} \quad t^* = H^*x$$

leading to

$$t^* = \Sigma_t \left( \Sigma_t + \Sigma_n \right)^{-1} x$$

• For stationary graph signals

$$\hat{t}^*_n = \frac{\sigma_t^2(\lambda_n)}{\sigma_t^2(\lambda_n) + \sigma_n^2(\lambda_n)} \hat{x}_n$$

• For polynomial functions, can be solved distrib. via ARMA_K

[Girault’15, EUSIPCO] [Marques’16, arXiv] [Perraudin’16, TSP]
Tikhonov Results

- ARMA filters do not require eigendecomposition of $S$
- For $K = 1,2$ the recursion is always convergent
- For $K = 1$, a good NMSE is reached in 10 iterations
Tikhonov Results

\[ p = 0.05 \]
\[ w = 0.5 \]
\[ K = 1 \]
\[ \hat{t}_n = e^{-5\lambda n} \]
\[ n \sim \mathcal{N}(0, I) \]
Issues in the Parallel Implementation

- Design with guaranteed convergence is challenging
- Problem more severe for high $\|S\|_2$, i.e., regular Laplacian
- Convergence time hard to control

Proposed Solution, ARMA\textsubscript{P,Q}

\[
y_t = - \sum_{p=1}^{P} \psi_p S^p y_{t-1} + \sum_{q=0}^{Q} \varphi_q S^q x
\]

- Under convergence, we obtain the steady-state response

\[
h(\lambda_n) = \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p}
\]

subject to \( \|P\|_2 = \max_n \left\{ \left| \sum_{p=1}^{P} \psi_p \lambda_n^p \right| \right\} < 1 \)

- The smaller \( \|P\|_2 \), the faster the convergence
- \( \|Q\|_2 \) has only a small effect on convergence time

convex constraint on coefficients
ARMA\(_{P,Q}\) Design

- Minimize the error \(e_n = \hat{h}_n - \frac{\sum_{q=0}^{Q} \varphi_q \lambda_n^q}{1 + \sum_{p=1}^{P} \psi_p \lambda_n^p}\)

- Prony’s method with constraint:

\[
\begin{align*}
\hat{e}'_n &= \hat{h}_n + \hat{h}_n (\sum_{p=1}^{P} \psi_p \lambda_n^p) - (\sum_{q=0}^{Q} \varphi_q \lambda_n^q) \\
\end{align*}
\]

\[
\Rightarrow \quad \hat{e}' = \hat{h} + \hat{h} \circ (\Psi_P \psi) - \Psi_{Q+1} \varphi
\]

\[
\begin{align*}
\min_{\psi, \varphi} & \|\hat{h} + \hat{h} \circ (\Psi_P \psi) - \Psi_{Q+1} \varphi\|^2 \\
\text{subject to} & \quad \|P\|_2 = \|\Psi_P \psi\|_\infty \leq \delta_P < 1
\end{align*}
\]

\[
[\Psi_P]_{n,p} = \lambda_n^p \\
[\Psi_{Q+1}]_{n,q} = \lambda_n^{q-1}
\]
ARMA_ {P,Q} Results

- Random graph, \( N = 100 \), \( S = L_n - \frac{\lambda_{\max}(L_n)}{2} I \), \( \lambda_c = \lambda_{N/2}(S) \)
- \( x \) is white, \( y_0 = x \), \( \delta_P = 0.65 \)

\[ P + Q \leq K \]
2-Dimensional Time-Graph Filters
Filtering time-varying graph signals

- FIR filters have difficulties handling time-varying signals
- ARMA filters are more appropriate in this context
- The ARMA$_K$ recursion for time-varying signals becomes

$$y_{t+1}^{(k)} = \psi^{(k)} S y_t^{(k)} + \varphi^{(k)} x_t$$

with 2D transfer function

$$h(z, \lambda) = \sum_{k=0}^{K} \frac{\psi^{(k)} z^{-1}}{1 - \psi^{(k)} \lambda z^{-1}}$$

- We can capture graph and time variations 😊
- The class of filters we can design is limited 😞
ARMA for Time-Varying Signals: Example

- Approximate an ideal frequency response with an ARMA$_3$ and analyze the behavior for different temporal frequencies.

- For small frequencies the response is good 😊
- For higher frequencies we observe a larger error 😞
2D – ARMA General Form

- Possible by considering more shifts in \textit{time} and \textit{graph}

\[
\sum_{l=0}^{L} \sum_{p=0}^{P} \tilde{\psi}_{l,p} S^l y_{t-p} = \sum_{k=0}^{K} \sum_{q=0}^{Q} \tilde{\varphi}_{k,q} S^k x_{t-q}
\]

- We keep track of the \textit{history} of the \textit{output} and \textit{input}
2D – ARMA General Form

\[
\sum_{l=0}^{L} \sum_{p=0}^{P} \tilde{\psi}_{l,p} S^l y_{t-p} = \sum_{k=0}^{K} \sum_{q=0}^{Q} \tilde{\varphi}_{k,q} S^k x_{t-q}
\]

- under stability conditions

\[
h(z, \lambda) = \frac{\sum_{k=0}^{K} \sum_{q=0}^{Q} \tilde{\varphi}_{k,q} \lambda^k z^{-q}}{\sum_{l=0}^{L} \sum_{p=0}^{P} \tilde{\psi}_{l,p} \lambda^l z^{-p}}
\]

- accurate 2D design is challenging
- more appropriate data driven design
- joint stability complicates things
- different orders in time and graph
- larger filter class

[Mei17,TSP ]
[Loucas’16, arXiv]
2D – ARMA Separable

\[
\sum_{l=0}^{L} \sum_{p=0}^{P} \tilde{\psi}_{l,p} S^l y_{t-p} = \sum_{k=0}^{K} \sum_{q=0}^{Q} \tilde{\varphi}_{k,q} S^k x_{t-q}
\]

when \( \tilde{\psi}_{l,p} = \psi_{l}a_{p} \) and \( \tilde{\varphi}_{k,q} = \varphi_{k}b_{q} \)

\[
\sum_{l=0}^{L} \sum_{p=0}^{P} \psi_{l} a_{p} S^l y_{t-p} = \sum_{k=0}^{K} \sum_{q=0}^{Q} \varphi_{k} b_{q} S^k x_{t-q}
\]

\[
H(z, \lambda) = \frac{\sum_{k=0}^{K} \sum_{q=0}^{Q} \varphi_{k} b_{q} \lambda^k z^{-q}}{\sum_{l=0}^{L} \sum_{p=0}^{P} \psi_{l} a_{p} \lambda^l z^{-p}} = \frac{\sum_{k=0}^{K} \varphi_{k} \lambda^k}{\sum_{l=0}^{L} \psi_{l} \lambda^l} \frac{\sum_{q=0}^{Q} b_{q} z^{-q}}{\sum_{p=0}^{P} a_{p} z^{-p}}
\]

\(H_g(\lambda) = H_t(z)\)

ARMA(K,L) in graph

ARMA(P,Q) in time
2D – ARMA Separable

\[ H(z, \lambda) = \frac{\sum_{k=0}^{K} \sum_{q=0}^{Q} \varphi_k b_q z^{-q}}{\sum_{l=0}^{L} \sum_{p=0}^{P} \psi_l a_p z^{-p}} = \begin{bmatrix} \frac{\sum_{k=0}^{K} \varphi_k \lambda^k}{\sum_{l=0}^{L} \psi_l \lambda^l} \\ \frac{\sum_{q=0}^{Q} b_q z^{-q}}{\sum_{p=0}^{P} a_p z^{-p}} \end{bmatrix} \]

- stability issue is alleviated
- accurate design in each domain
- freedom of choosing the design technique independently
- Limited to separable designs

ARMA(K,L) in graph \( H_g(\lambda) \)

ARMA(P,Q) in time \( H_t(z) \)
Separable ARMA: Results

• 100 nodes randomly placed in a square area
• Neighbors if nodes are closer than 15% of the max distance

• Graph signal of the form \( x_t = s_t + i_t + n_t \)

• with signal of interest \( \hat{s}_t = e^{j \pi t/4} \) if \( \lambda_n < 0.5 \)

• Interferer \( \hat{i}_t = e^{j 3\pi t/4} \) \( \forall \lambda_n \)

• noise \( n_t \), zero mean Gaussian with \( \Sigma_x = \sigma^2 I \), \( \sigma^2 = 0.1 \)

• Goal: Recover through a 2D graph filter the signal of interest
Separable ARMA: Results

- Graph signal of the form \( x_t = s_t + i_t + n_t \)

- Define the errors
  \[
  e_t^{(\text{interf})} = \frac{\| \hat{y}_t - \hat{y}_t^* \|}{\| \hat{y}_t^* \|} \quad e_t^{(\text{total})} = \frac{\| \hat{y}_t - \hat{s}_t \|}{\| \hat{s}_t \|}
  \]

- \( \hat{y}_t^* \) is the spectrum of the filter output without the interferer

- \( \hat{s}_t \) is the spectrum of the signal of interest
Stochastic analysis of graph filters

- **Graph** may be random in time
  - traffic diffusion in a city network (street closures)
  - link loss in sensor networks
  - personal relationships in social networks

- **Signal** may be random in time
  - noise corrupted
  - belong to a certain distribution (accident probability)

In these cases the graph filters are run over a *stochastic* graph with *stochastic* graph signals
Stochastic analysis of graph filters

- **Graph model** (i.i.d. in time)
  - link \((i, j)\) is active with probability \(0 < p_{i,j} \leq 1\)

  \[
  \mathcal{G} \rightarrow S
  \]

  \[
  \mathcal{G}_{\tau_1} \rightarrow S_{\tau_1}
  \]

  \[
  \mathcal{G}_{\tau_2} \rightarrow S_{\tau_2}
  \]

- **Signal model** (i.i.d. in time)
  - includes stationary graph signals and GRMF signals

- **Signal realizations are independent from graph**
  - signal and graph fluctuations not related, e.g., additive noise
  - underlying signal can still be related to the graph
Stochastic ARMA$_K$

ARMA$_K$ stochastic output on $\mathcal{G}_t$

$$y^{(K)}_{t+1} = \psi^{(K)} S_t y^{(K)}_t + \varphi^{(K)} x_t$$

expected ARMA$_K$ output

$$\bar{y}^{(K)}_{t+1} = \psi^{(K)} \bar{S} \bar{y}^{(K)}_t + \varphi^{(K)} \bar{x}$$

output steady state variance

$$\lim_{t \to \infty} \text{var}[y_{t+1}] \leq \frac{K \|\varphi\|^2}{(1 - \rho |\psi_{\text{max}}|)^2} \left(\text{var}[x_t] + \frac{\|\bar{x}\|^2}{N}\right)$$

$$\varphi = [\varphi_1, \ldots, \varphi_K]^T$$

$$\psi_{\text{max}} = \max\{\psi_1, \ldots, \psi_K\}$$

Deterministic ARMA in the mean

Results extend to time-varying mean
Stochastic output

- $x_t = t + n_t$ with $\hat{t}_n$ equal to 1 for $\lambda_n < 1$ and 0 for $1 \leq \lambda_n < 2$
- FIR and ARMA, designed to meet $t$ for $\Sigma_n = \sigma_n^2 I$
- Results compared with the deterministic output

- Random geometric graph $N = 100$
- ARMA performs better up to $K = 8$
- FIR performs better for $K > 8$
- $\text{ARMA}_1$ performs well
**Stochastic Sparsification**

*Reduce the cost:* Each node, for each \( t \), transmits data to its neighbors with probability \( p \)

\[
S \rightarrow \mathcal{G} \quad S_t \rightarrow \mathcal{G}_t \quad \bar{S} = \mathbb{E}\{S_t\} = pS
\]

Instead of performing the filtering on \( \mathcal{G} \), it can be done on its stochastically time-varying realizations \( \mathcal{G}_t \)

The filter output is stochastic: characterize **mean** and **variance**
Stochastic Sparsification

ARMA₁, det. output on $G$

\[ y_t = \psi S y_{t-1} + \varphi x \]

stoch. output on $G_t$

\[ y^{(s)}_t = \psi^{(s)} S_t y^{(s)}_{t-1} + \varphi^{(s)} x \]

expected output

\[ \bar{y}^{(s)}_t = \psi^{(s)} (pS) \bar{y}^{(s)}_{t-1} + \varphi^{(s)} x \]

where \( \psi^{(s)} = \psi / p \) and \( \varphi^{(s)} = \varphi \)

\[ \bar{y}^{(s)}_t = y_t \]

where \( \psi^{(s)} = \psi / p \) and \( \varphi^{(s)} = \varphi \)

\[ \bar{y}^{(s)}_t = y_t \]

variance of the steady state \( t \to \infty \)

\[ \lim_{t \to \infty} \text{var}[y^{(s)}_t] \leq \frac{1}{N} \left( \frac{\varphi^2 \|x\|^2}{(1 - \|S\| \psi p^{-1})^2} \right) \]
Numerical Results

$\mathcal{G}$ is geometric graph with $N=1000$ nodes in a squared area

Nodes connected if they are closer than 15% of the diagonal

Noisy graph signal $x = u + n$ with $\hat{u}_n = e^{-\lambda_n}$

Error between sparsified and regular output $e = y^{(s)} - y$

Shift operator $S = \frac{1}{\lambda_{\text{max}}}D - 0.5I$
Conclusions

• We introduced ARMA graph filters
• Studied centralized and distributed implementations
• Applications: denoising, interpolation, Wiener filtering
• These filters have the potential to improve FIRs
  – They naturally extend to the temporal dimension to perform joint graph-time processing
• Stochastic analysis of graph filters
  – Output is characterized statistically
  – Can be used to perform adaptive denoising/filtering
  – Can be used to perform sparsified filtering
Thank You!

Questions?