Memory Efficient Signal Reconstruction from Phaseless Coefficients of a Linear Mapping

Naveed Haghani
nhaghan1@math.umd.edu

Project Advisor:
Dr. Radu Balan
rvbalan@cscamm.umd.edu
Professor of Applied Mathematics, University of Maryland
Department of Mathematics
Center for Scientific Computation and Mathematical Modeling
Norbert Weiner Center

Problem

Original Signal

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \epsilon\ \mathbb{C}^n$$

Transformation

Transformation

$$c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_m \end{bmatrix} \epsilon\ \mathbb{C}^m$$

Transformation Magnitudes

$$\alpha = \begin{bmatrix} |c_1|^2 \\ |c_2|^2 \\ \vdots \\ \vdots \\ |c_m|^2 \end{bmatrix} \epsilon\ \mathbb{R}^m$$

Reconstruct Original Signal

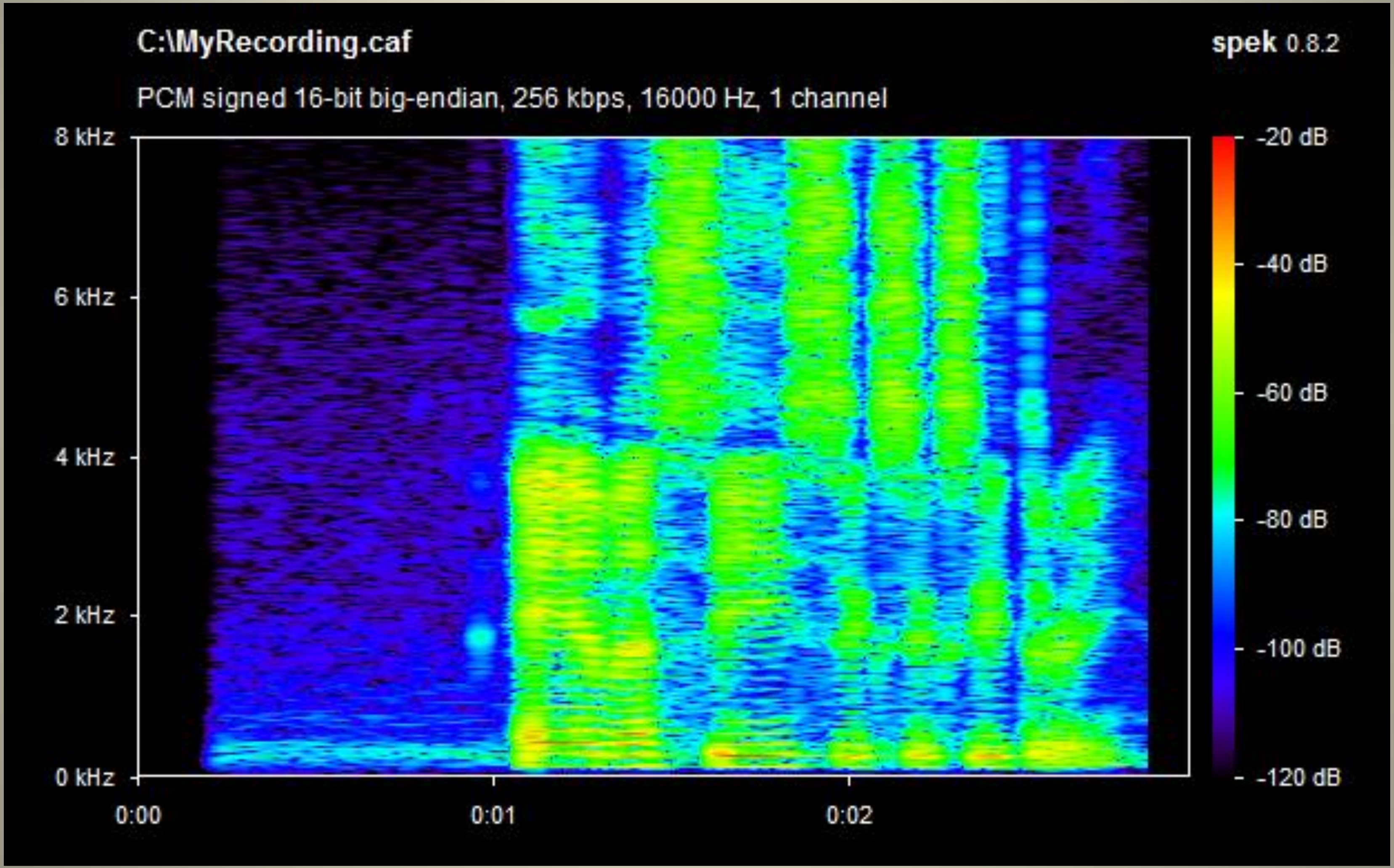
Original Signal Approximation

$$\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \vdots \\ \hat{x}_n \end{bmatrix} \epsilon\ \mathbb{C}^n$$

Application: Audio Processing

- Employs the use of an audio signals spectrogram
- Spectrogram- time-frequency representation of an audio signal
  - Useful in the processing and manipulation of audio signals
  - Does not carry phase information
- Would like to recover an audio signal after processing its spectrogram

Example Spectrogram



created using spek 0.8.2

Transformation $c = T(x)$

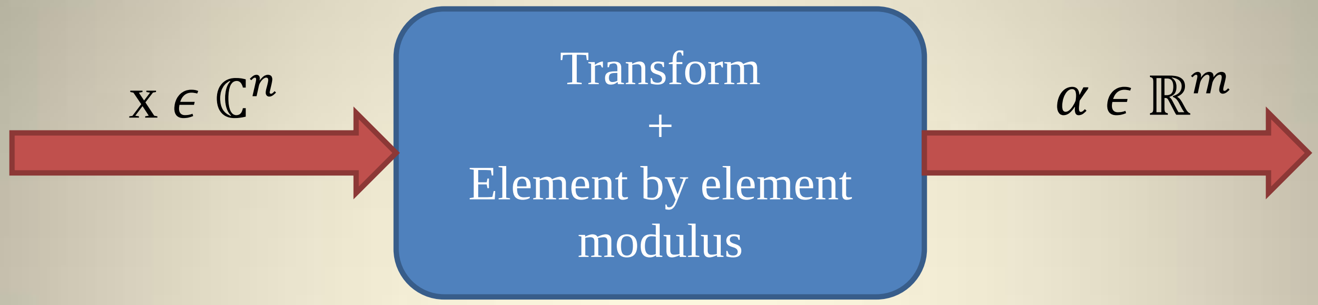
- Weighted Discrete Fourier Transform

$$B_j = Discrete\ Fourier\ Transform\left\{\begin{bmatrix} w_1^{(j)} & & 0 \\ & \ddots & \\ 0 & & w_n^{(j)} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}\right\}$$

$$for\ 1 \leq j \leq R\ randomly\ generated\ arrays\ of\ complex\ weights \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ w_n \end{bmatrix}$$

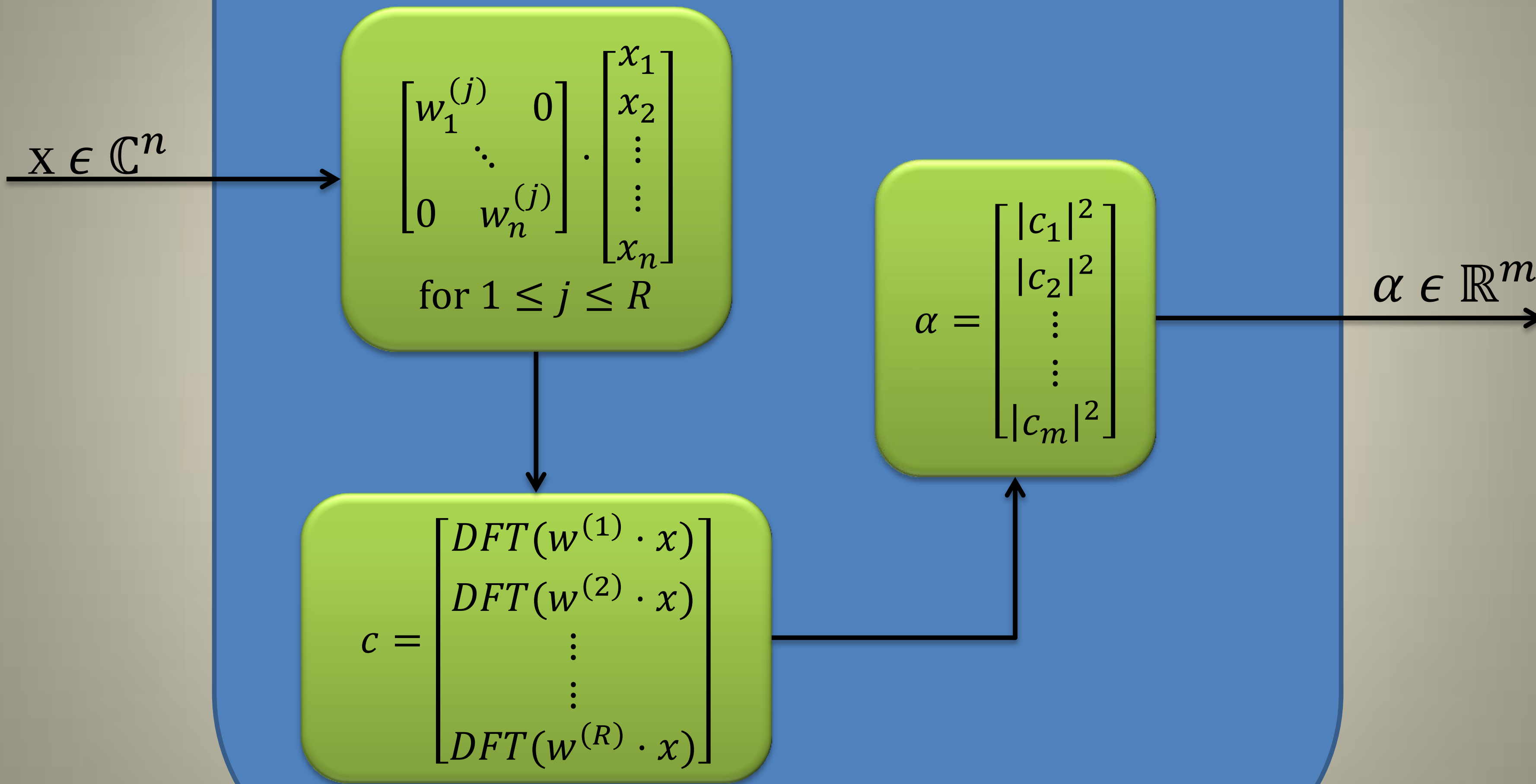
$$c = T(x) = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ \vdots \\ B_R \end{bmatrix}$$

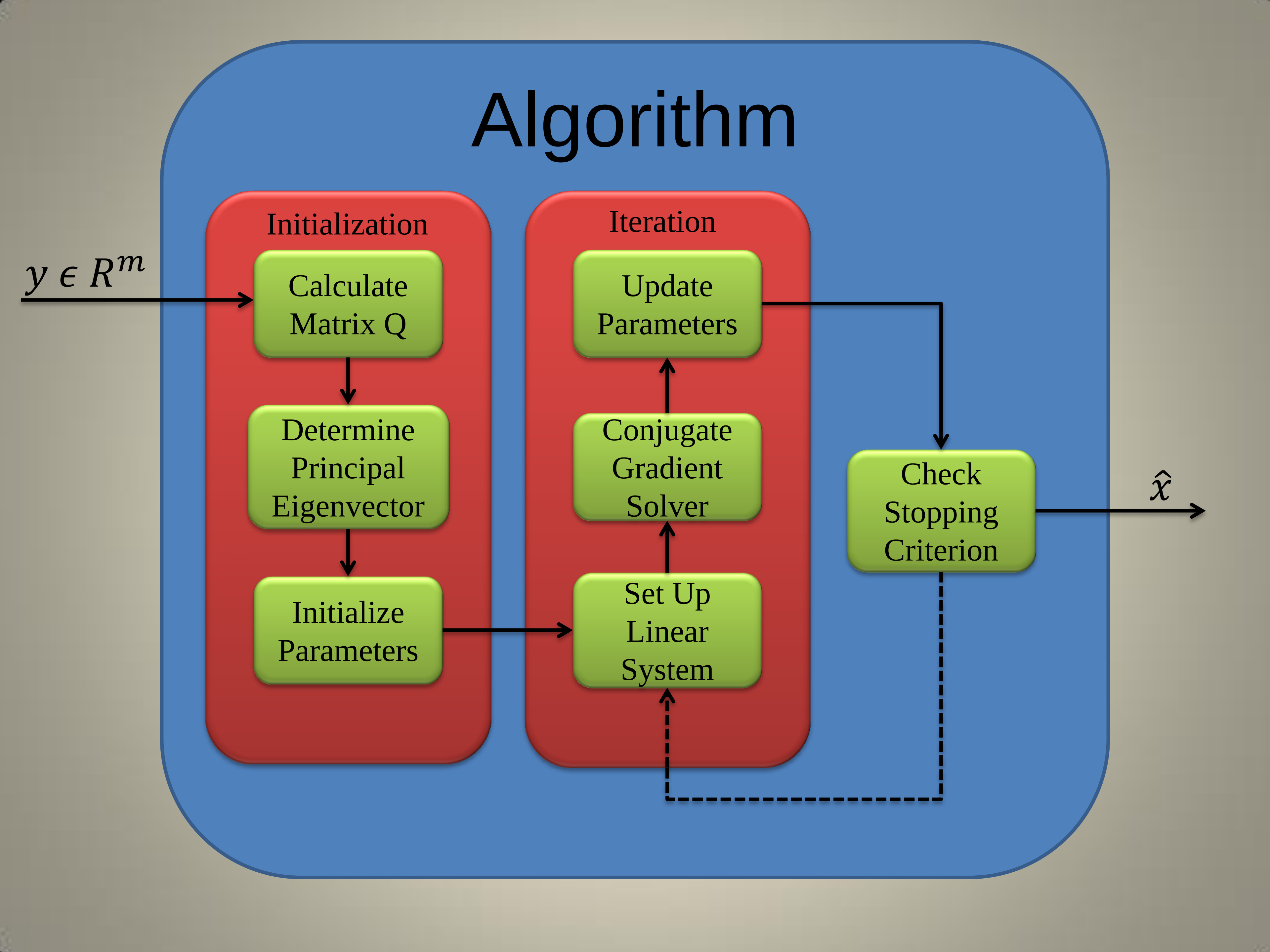
Approach



$$y = \alpha + \sigma \cdot \nu, \qquad \sigma \cdot \nu: noise$$

Transform + Modulus




Algorithm
Initialization
Iteration
$y \in R^m$
Calculate Matrix Q
Determine Principal Eigenvector
Initialize Parameters
Update Parameters
Conjugate Gradient Solver
Set Up Linear System
Check Stopping Criterion
$\hat{x}$

Algorithm Initialization

$$Q = \sum_{k=1}^{m} y_k f_k f_k^*$$

[4]

f_k is kth frame vector from transformation T(x)

For the Discrete Fourier Transform ⇨

$$f_k = \frac{1}{\sqrt{R \cdot n}} \begin{bmatrix} w_1^{(j)} \cdot 1 \\ w_2^{(j)} e^{-i2\pi j \cdot 1/n} \\ \vdots \\ \vdots \\ w_n^{(j)} e^{-i2\pi j \cdot n/n} \end{bmatrix} \quad where \;\; j = ceiling\left(\frac{k}{R}\right)$$

Algorithm Initialization (cont.)

- Find Principal Eigenvector of $Q^+ = Q + q \cdot I$
  - I is identity matrix
  - $q = \|y\|_\infty$ a positive constant to ensure positive-definiteness of Q
- Use Power Iteration Method
  - Initialize $e_k^{(0)} \sim N(0,1), for\ k = [1, n]$
  - Repeat:
    - $e^{(t+1)} = \frac{Q^+ \cdot e^{(t)}}{\left\|Q^+ \cdot e^{(t)}\right\|}$
    - If $\left\|e^{(t+1)} - e^{(t)}\right\| < tolerance$, end repeat

Algorithm Initialization (cont.)

$$\hat{x}^{(0)} = e\sqrt{\frac{(1-\rho)\cdot a}{\sum_{k=1}^{m}|\langle e, f_k\rangle|^4}}$$ [4]

e: *principal eigenvector of* Q^+

a: *associated eigenvalue*

ρ: *constant between* $(0, 1)$

$$\mu_0 = \lambda_0 = \rho \cdot a$$ [4]

Algorithm Iteration

- Work in Real space
  - $\xi = \begin{bmatrix} real(\hat{x}) \\ imag(\hat{x}) \end{bmatrix}$
- Solve linear system $A\xi^{(t+1)} = b$, where

$$A = \sum_{k=1}^{m} (\Phi_k \xi^{(t)}) \cdot (\Phi_k \xi^{(t)})^* + (\lambda_t + \mu_t) \cdot 1$$ [4]

$$b = \left(\sum_{k=1}^{m} y_k \Phi_k + \mu_t 1 \right) \cdot \xi^t$$ [4]

$$\Phi_k = \phi_k \phi_k^T + J \phi_k \phi_k^T \mathrm{J}^\mathrm{T}, \text{where } \phi_k = \begin{bmatrix} real(f_k) \\ imag(f_k) \end{bmatrix} \text{ and } \mathrm{J} = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix}$$

$\xi^{(t+1)}$ = next approximation

Algorithm Iteration (cont.)

- Update λ, μ

$$\lambda_{t+1} = \gamma\lambda_{t+1}, \qquad \mu_{t+1} = \max\left(\gamma\mu_t, \mu^{min}\right), \qquad \text{where } 0 < \gamma < 1$$ [4]

- Stopping criterion

$$\sum_{k=1}^{m} \left| y_k - \left|\left\langle x^{(t)}, f_k \right\rangle\right|^2 \right|^2 \leq \kappa m \sigma^2, where\ \kappa\ is\ a\ constant > 1$$ [4]

Conjugate Gradient

- Iterative solver for linear systems that are symmetric and positive definite
- Travel towards solution along mutually conjugate directions
  - Vectors $p^1 and\ p^2$ are conjugate if ${p^1}^T A p^2 = 0$
- For a matrix in $\mathbb{R}^n$ there are n different conjugate directions, forming a complete basis
- Traveling along each of the n directions should converge to the true solution

Conjugate Gradient

$r^{(k)} = b - A\hat{x}^{(k)}$

$r^{(k)}$: residual at kth iteration

$\hat{x}^{(k)}$: approximate solution at k^{th} iteration

$p^{(0)} = r^{(0)}$

Repeat until $\left\|r^{(k)}\right\|^2 < tolerance$

$$\alpha = \frac{\langle r^{(k)}, r^{(k)} \rangle}{{p^{(k)}}^T A p^{(k)}}$$

$$\hat{x}^{(k+1)} = \hat{x}^{(k)} + \alpha p^{(k)}$$

$$r^{(k+1)} = r^{(k)} - \alpha A p^{(k)}$$

$$p^{(k+1)} = r^{(k+1)} + p^{(k)} \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle}$$

Implementation

- MATLAB
  - Will use *fft()* function for Fourier Transform

- As memory efficient as possible
  - Avoid allocating memory for entire linear system
    - Linear system is 2n x 2n
    - n is large
    - Compute each vector component as it is needed

Data Construction

- Input data synthetically generated
- $n \sim 10{,}000, R = 8, m = R \cdot n$
- 10 realizations of signal sample x
  - $x_k \sim N(0,1) + iN(0,1), for\ k\ [1, n]$
- 10 realizations of $w^{(1:R)}$
  - $w_k^{(j)} \sim N(0,1) + iN(0,1), for\ k\ [1, n], j\ [1\ R]$
- 10,000 realizations of noise ν
  - $\nu_k \sim N(0,1), for\ k\ [1, m]$

Testing

- $y = \alpha + \sigma \cdot \nu$
- For each ν, vary σ to achieve desired SNR
- SNR [-30 dB, 30 dB] increments of 5
  - $SNR_{dB} = 10 \cdot \log_{10}\left[\frac{\sum_{k=1}^{m}|c_k|^2}{\sigma^2 \sum_{k=1}^{m}|\nu_k|^2}\right]$
- Obtain 10,000 output values for each input y at each SNR

Post Processing

- For each SNR level of each input, calculate
  - mean($\hat{x}$)
  - Variance($\hat{x}$)
  - MSE($\hat{x}$)
- Study trend of each output vs SNR level

Metrics

- Memory usage
- Scaling of numerical complexity with problem size
- Time efficiency of algorithm
- Accuracy vs SNR

Validation

- Power Iteration
  - Validate using Matlab’s eigenvalue solver, *eig()*
- Conjugate Gradient
  - Validate on small sample input using exact solution from decomposition (Matlab’s *mldivide()*)
  - Compare output with conjugate gradient’s complete convergence
- Validate complete system by proximity to true solution

Schedule

October	▪ Post processing framework ▪ Database generation
November	▪ MATLAB implementation of iterative recursive least squares algorithm
December	▪ Validate modules written so far
February	▪ Implement power iteration method ▪ Implement conjugate gradient
By March 15	▪ Validate power iteration and conjugate gradient
March 15 – April 15	▪ Test on synthetic databases ▪ Extract metrics
April 15 – end of semester	▪ Write final report

Deliverables

- Presentations
- Proposal
- Final Report
- Program
- Input data
- Output data
- Output charts and graphs

References

[1] R. Balan, On Signal Reconstruction from Its Spectrogram, Proceedings of the CISS Conference, Princeton, NJ, May 2010.

[2] R. Balan, P. Casazza, D. Edidin, On signal reconstruction without phase, Appl.Comput.Harmon.Anal. 20 (2006), 345-356.

[3] R. Balan, Reconstruction of signals from magnitudes of redundant representations. 2012.

[4] R. Balan, Reconstruction of signals from magnitudes of redundant representations: the complex case. 2013.

[5] Christensen, Ole. "Frames in Finite-dimensional Inner Product Spaces."*Frames and Bases*. Birkhäuser Boston, 2008. 1-32.

[6] Allaire, Grâegoire, and Sidi Mahmoud Kaber. *Numerical linear algebra*. Springer, 2008.

[7] Shewchuk, Jonathan Richard. "An introduction to the conjugate gradient method without the agonizing pain." (1994).