Midterm Presentation

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 Overview

Original Signal

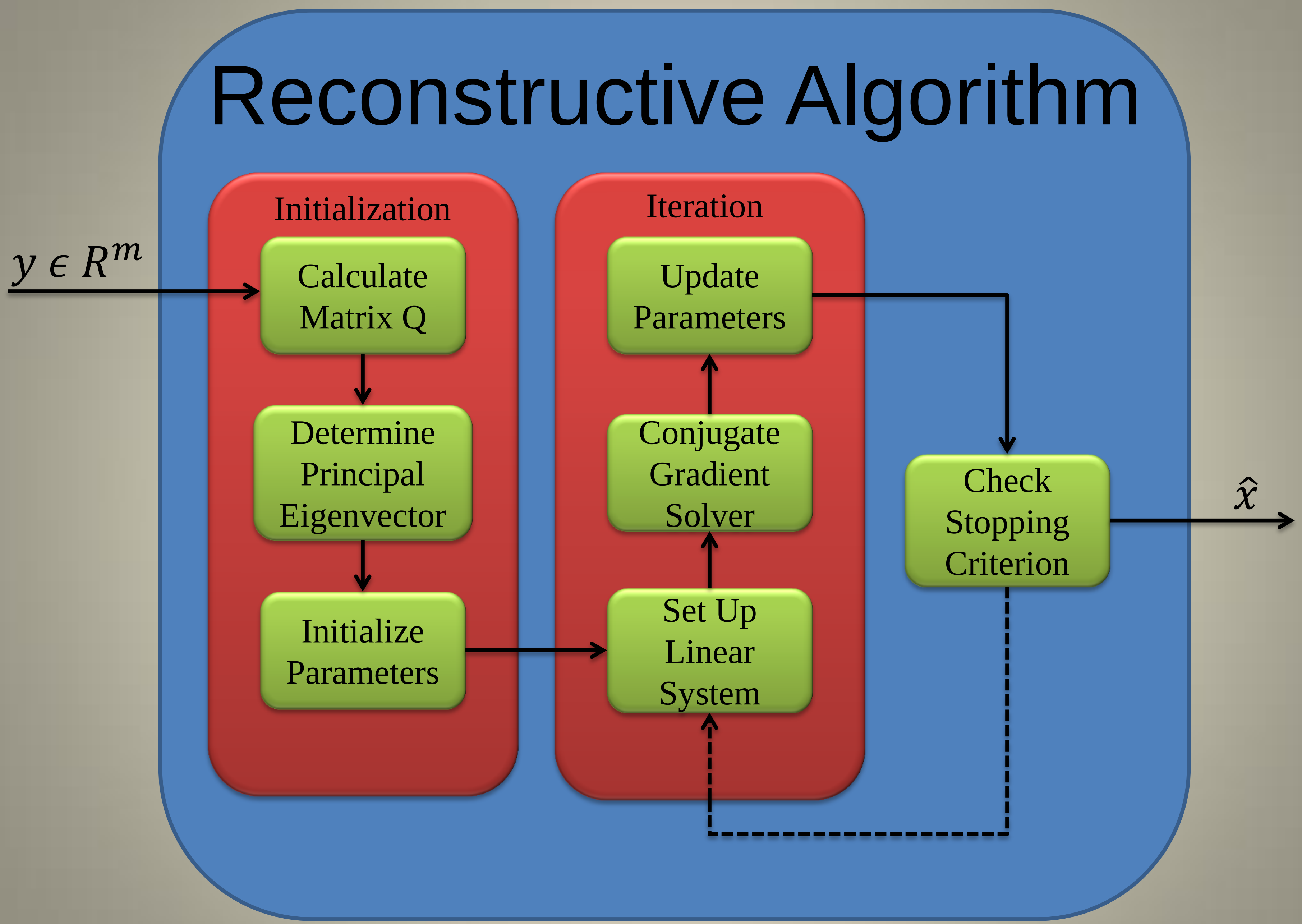
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \epsilon\ \mathbb{C}^n \longrightarrow \boxed{\text{Transformation}} \longrightarrow c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_m \end{bmatrix} \epsilon\ \mathbb{C}^m$$

Transformation

Transformation Magnitudes

$$\alpha = \begin{bmatrix} |c_1|^2 \\ |c_2|^2 \\ \vdots \\ \vdots \\ |c_m|^2 \end{bmatrix} \epsilon\ \mathbb{R}^m \longrightarrow \boxed{\begin{array}{c}\text{Reconstruct} \\ \text{Original Signal}\end{array}} \longrightarrow \hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \vdots \\ \hat{x}_n \end{bmatrix} \epsilon\ \mathbb{C}^n$$

Original Signal Approximation


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

Transformation $T(x) = c$

- Redundant linear transformation
- Maps vector in $\mathbb{C}^n$ to $\mathbb{C}^m$
  - $m = R \cdot n$
  - R is redundancy of the Transformation $T(x)$
- Defined by m vectors in $\mathbb{C}^n$ labeled $f_{1:m}$ such that:

$$T(x) = c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_m \end{bmatrix} \text{ and } c_i = \langle x, f_i \rangle$$

Transformation $T(x) = c$

- Weighted Discrete Fourier Transform

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

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

$$T(x) = c = \frac{1}{\sqrt{R \cdot n}} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ \vdots \\ B_R \end{bmatrix}$$

Transformation $T(x) = c$

$$T(x) = \begin{bmatrix} \langle x, f_1 \rangle \\ \langle x, f_2 \rangle \\ \vdots \\ \langle x, f_m \rangle \end{bmatrix}$$

For the weighted DFT f_k is defined as:

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

Algorithm Initialization

$$Q = \sum_{k=1}^{m} y_k f_k f_k^*$$ [4]

e: *principal eigenvector of* Q^+

a: *associated eigenvalue of* Q

ρ: *constant between* $(0, 1)$

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

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

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

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

$\Phi_k = \phi_k \phi_k^T + J\phi_k \phi_k^T \mathrm{J}^{\mathrm{T}}$, where $\phi_k = \begin{bmatrix} real(f_k) \\ imag(f_k) \end{bmatrix}$ 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]

Program Goals

- Approximate signal x for $n \sim 10{,}000$
- Cannot store $f_{1:m}$ for large n
- Write a Sample implementation for small n ($\sim$100) that stores Transformation vectors $f_{1:m}$
- Write Efficient implementation that avoids this storage
  - Uses the transformation and its adjoint to compute $Q \cdot u$, $A \cdot u$, and b when needed.

Efficient Implementation

$$Q \cdot u = T^*\big(y \mathbin{.*} T(u)\big) + \|y\|_\infty$$

$$A \cdot u =$$

$$\begin{bmatrix} Re\left\{T^*\left(real\left\{T(u) \mathbin{.*} conj\left\{T\left(x^{(t)}\right)\right\}\right\} \mathbin{.*} T\left(x^{(t)}\right)\right) + (\lambda + \mu) \cdot u\right\} \\ Im\left\{T^*\left(real\left\{T(u) \mathbin{.*} conj\left\{T\left(x^{(t)}\right)\right\}\right\} \mathbin{.*} T\left(x^{(t)}\right)\right) + (\lambda + \mu) \cdot u\right\} \end{bmatrix}$$

$$b = \begin{bmatrix} Re\left\{T^*\left(y \mathbin{.*} T\left(x^{(t)}\right)\right) + \mu \cdot x^{(t)}\right\} \\ Im\left\{T^*\left(y \mathbin{.*} T\left(x^{(t)}\right)\right) + \mu \cdot x^{(t)}\right\} \end{bmatrix}$$

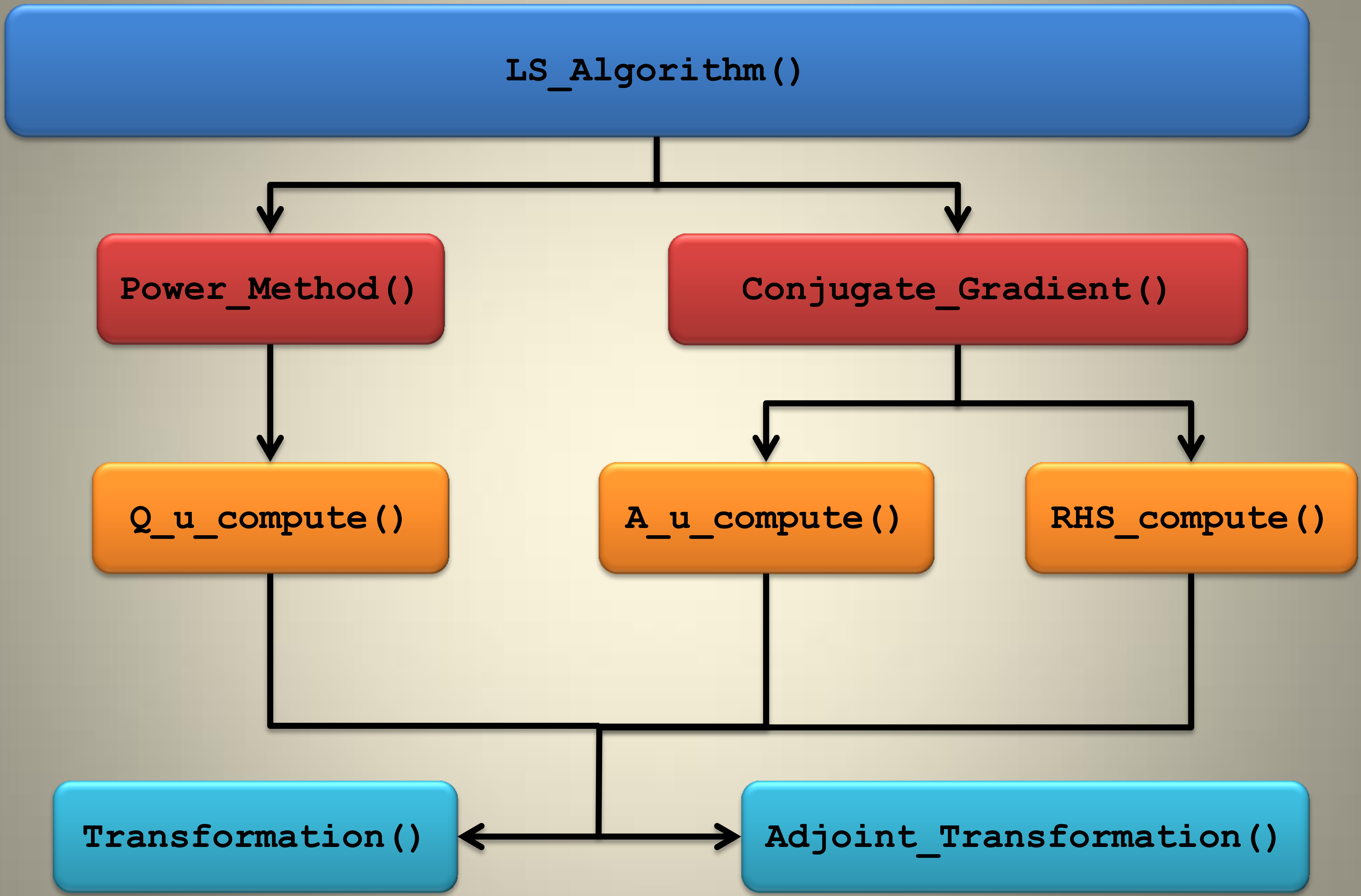
Adjoint Transformation $T^*(c)$

$$\sum_{k=1}^{R} \frac{1}{\sqrt{R \cdot n}} \cdot n \cdot \overline{w^k} \cdot ifft\left(c_{(k-1)\cdot n+1:k\cdot n}\right)$$

$$where\ ifft\ is\ the\ inverse\ fft$$

Implementation thus far

- ✓ Write sample implementation for small data sets
- ✓ Write memory efficient implementation avoiding the storage of the transformation matrix.
  - ✓ Write Power Method for finding the principal eigenpair
  - ✓ Write Conjugate Gradient Method for solving the linear system.
- ✓ Cross-Validate Programs


LS_Algorithm()
Power_Method()
Conjugate_Gradient()
Q_u_compute()
A_u_compute()
RHS_compute()
Transformation()
Adjoint_Transformation()

Preliminary Tests

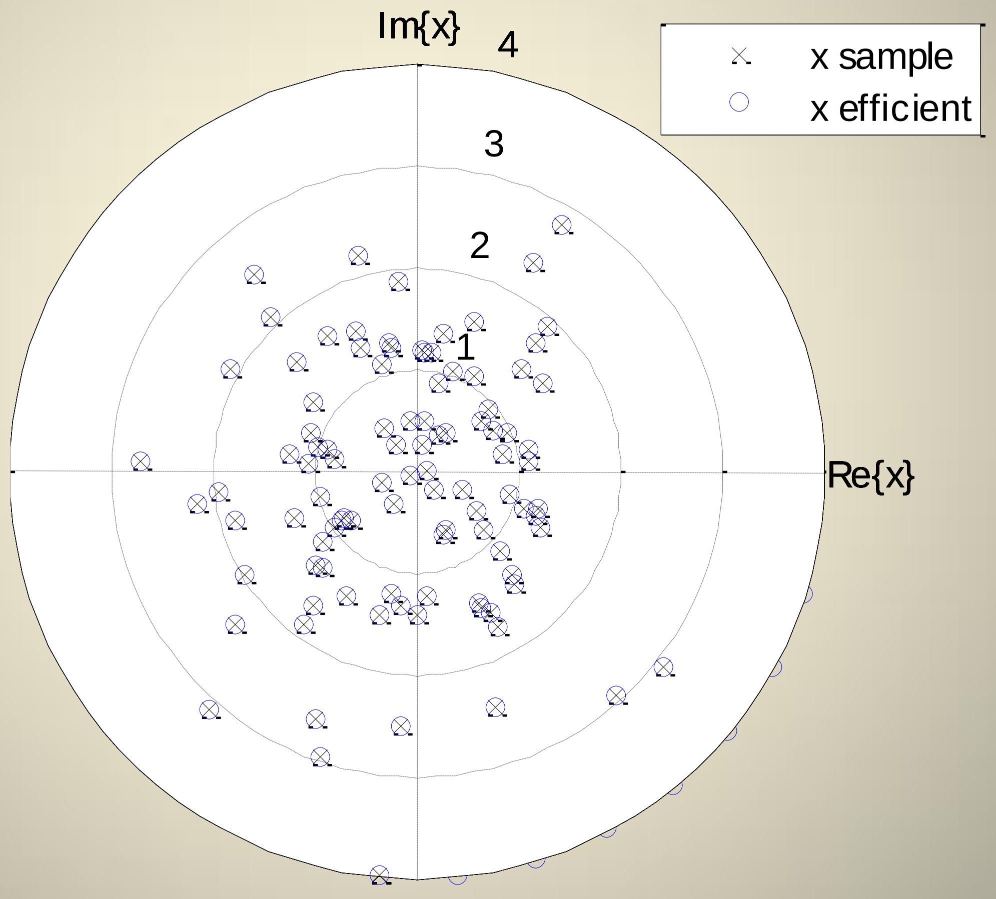
- Validation
- Study of iterative solvers
  - Power Method
  - Conjugate Gradient
- Time efficiency and memory efficiency
- Preliminary Testing Parameters
- Study of iterative solvers
  - $R = 8, SNR_{dB} = 10\ dB$
  - Program comparisons: $n = 100$
  - Other tests: $n = 10{,}000$

Validation

- Sample implementation and Efficient implementation can be compared for small problem sizes

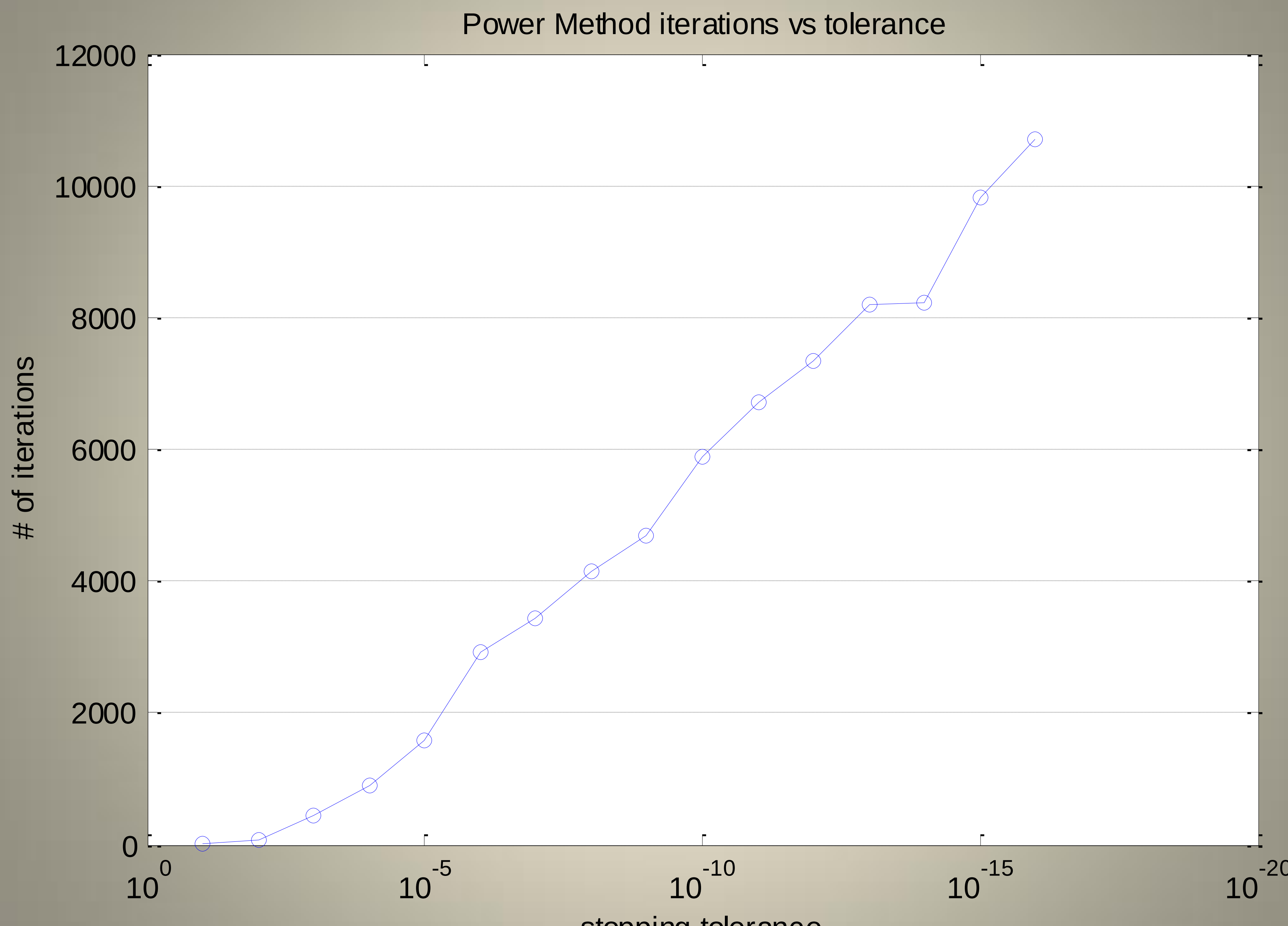
- Algorithms produce identical results off by a phase factor
  - Principal eigenvector used in initialization are off by a constant

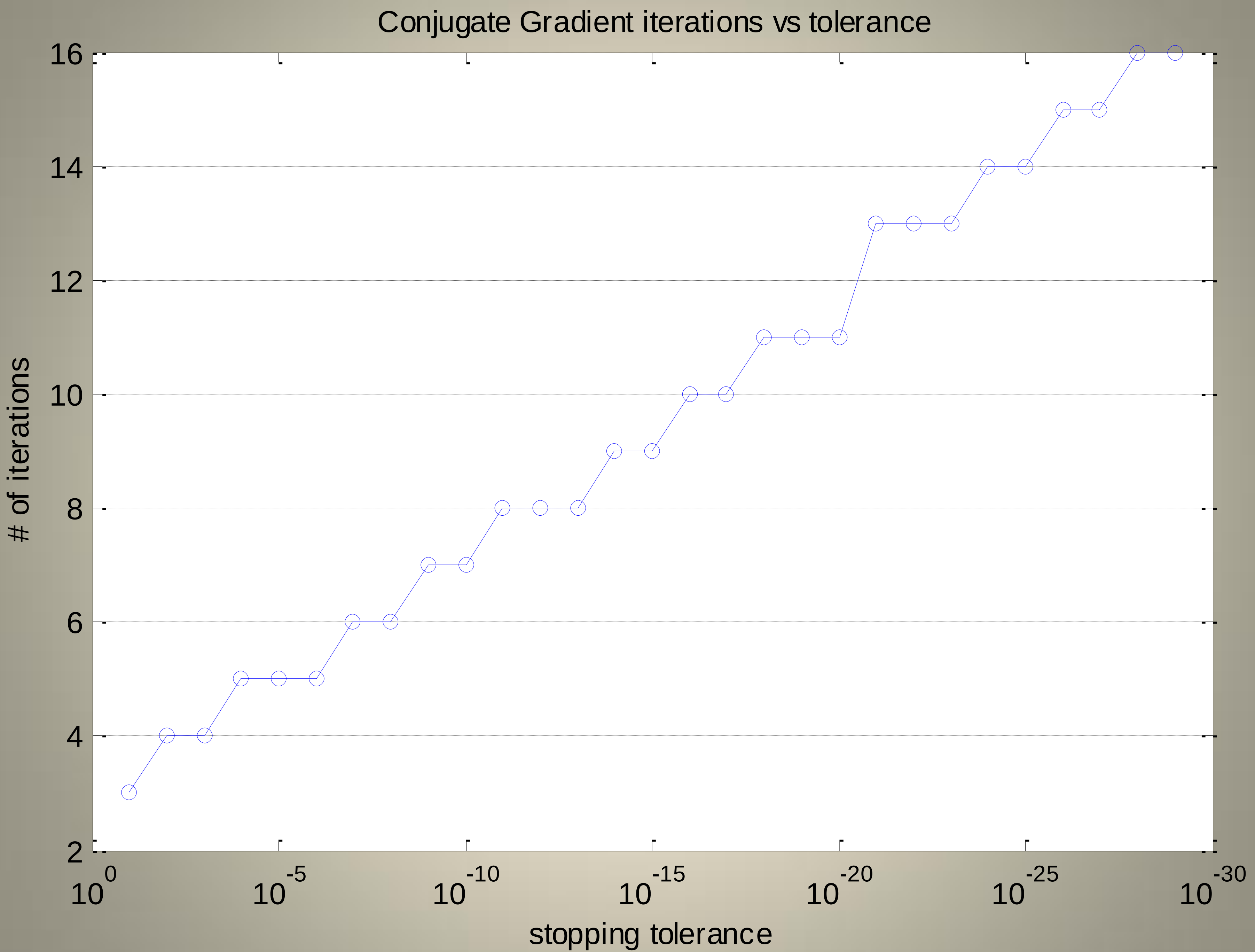
Output Results $\hat{x}$



Iterative Solvers

- Both the Power Method and the Conjugate Gradient Method require a stopping tolerance
- Required # of iterations vs stopping tolerance was investigated
- $n = 10{,}000$


Power Method iterations vs tolerance
12000
10000
8000
6000
4000
2000
0
# of iterations
10^{0}
10^{-5}
10^{-10}
10^{-15}
10^{-20}
stopping tolerance

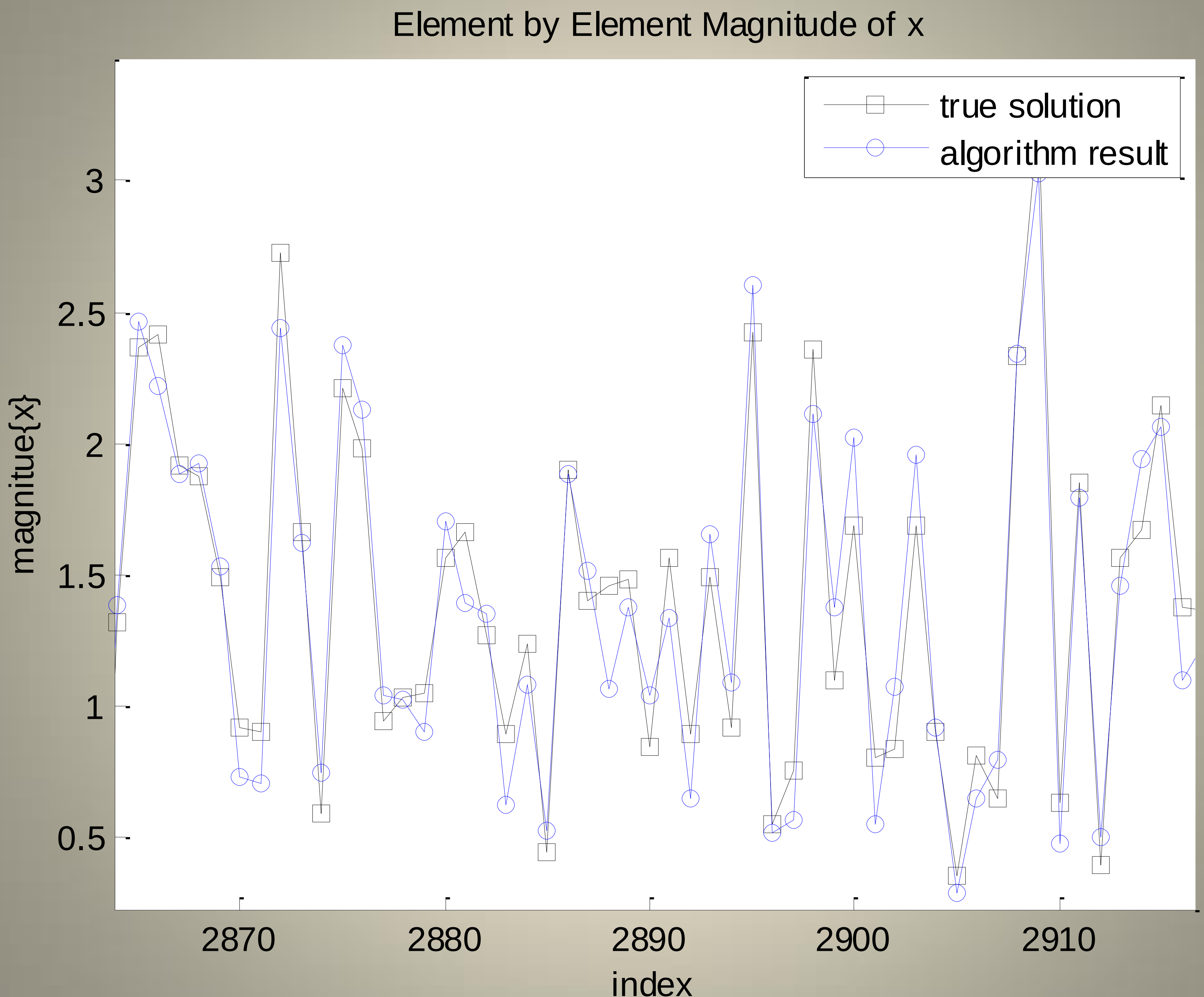

Conjugate Gradient iterations vs tolerance
# of iterations
stopping tolerance
16
14
12
10
8
6
4
2
10^0
10^-5
10^-10
10^-15
10^-20
10^-25
10^-30

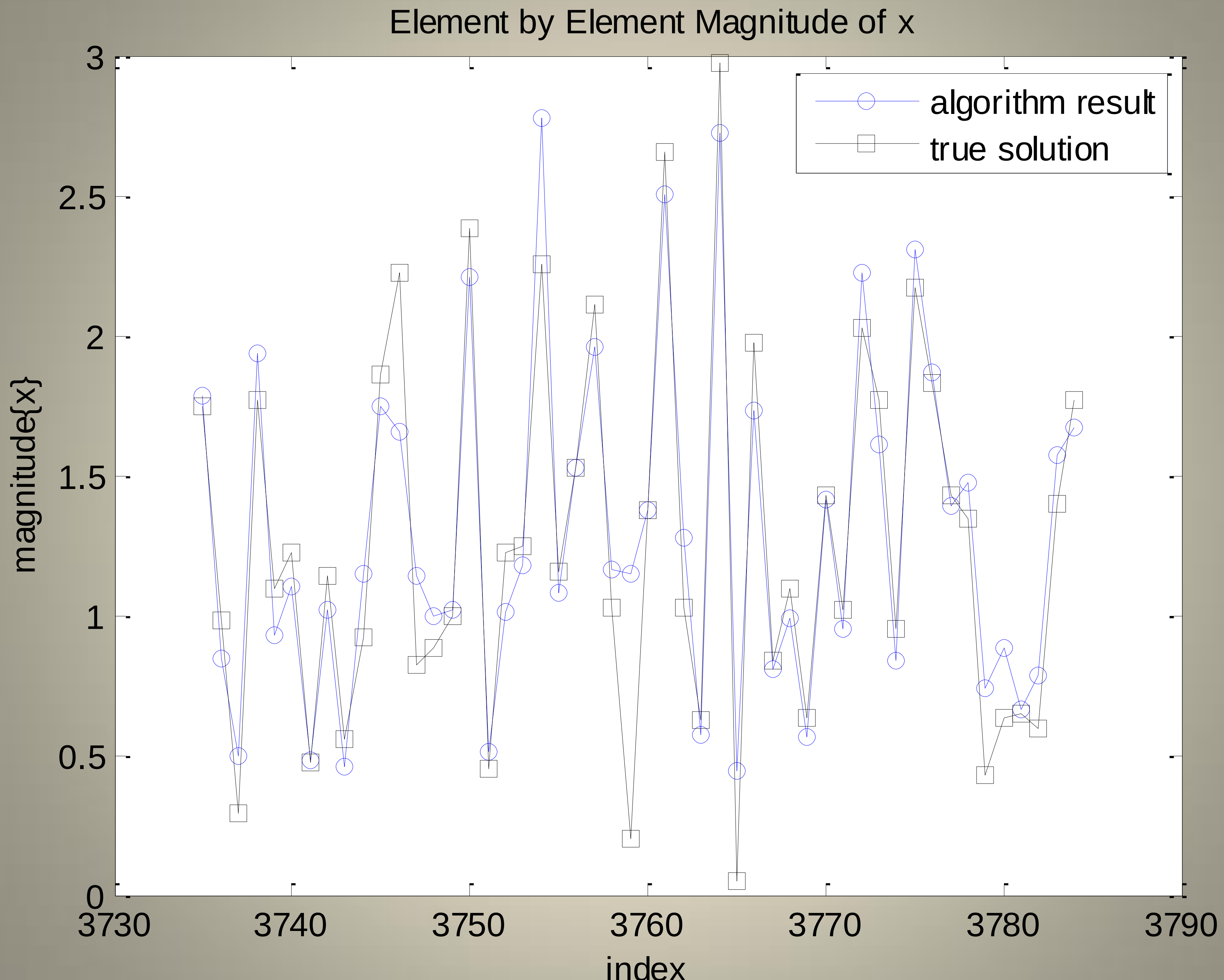
Output vs Original Signal

- Output, $\hat{x}$, of the efficient implementation is compared to the original signal

- $n = 10{,}000, SNR_{dB} = 10\ dB$

- Magnitude of each element is plotted.

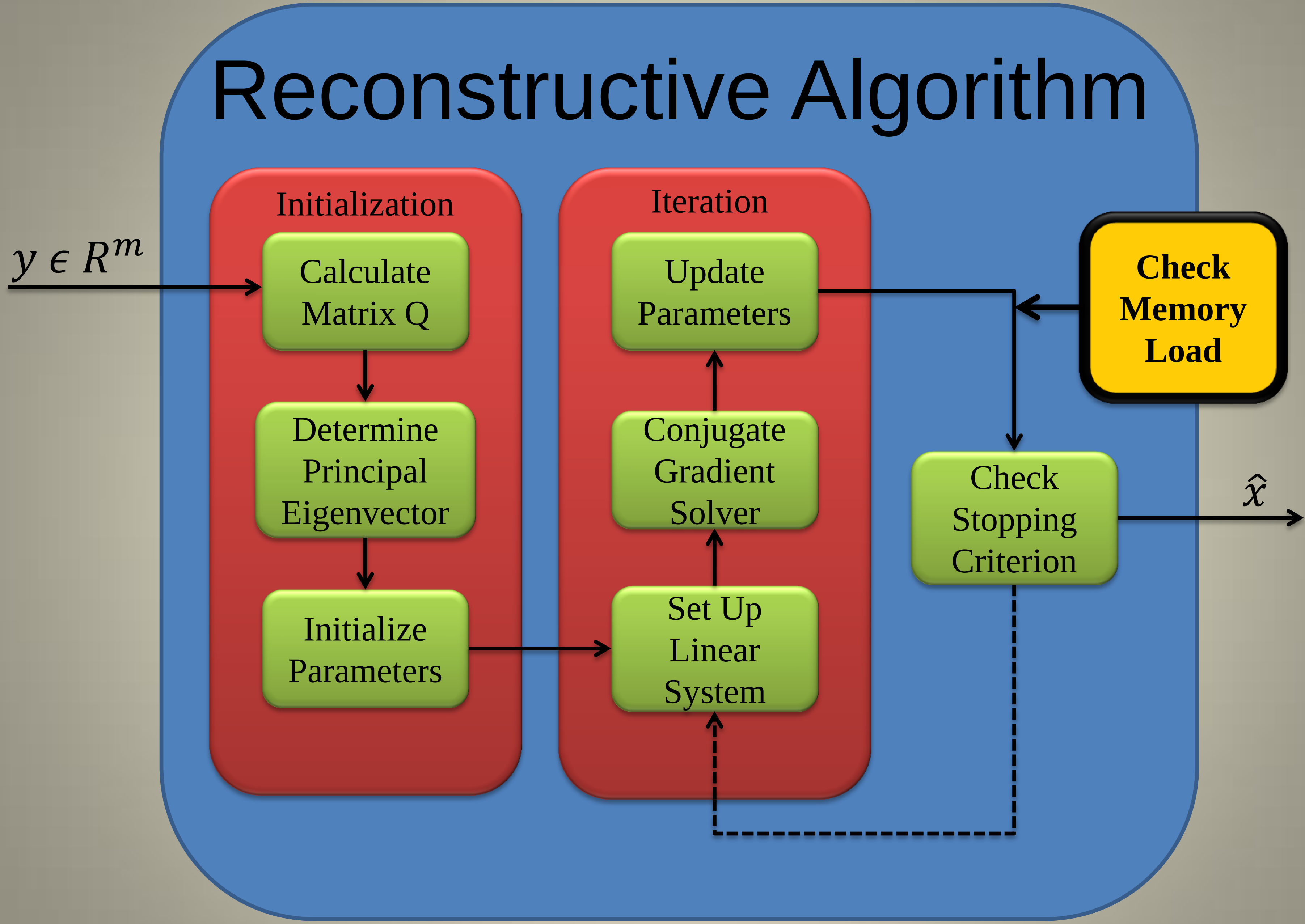

Element by Element Magnitude of x
true solution
algorithm result
magnitue{x}
index
3
2.5
2
1.5
1
0.5
2870
2880
2890
2900
2910

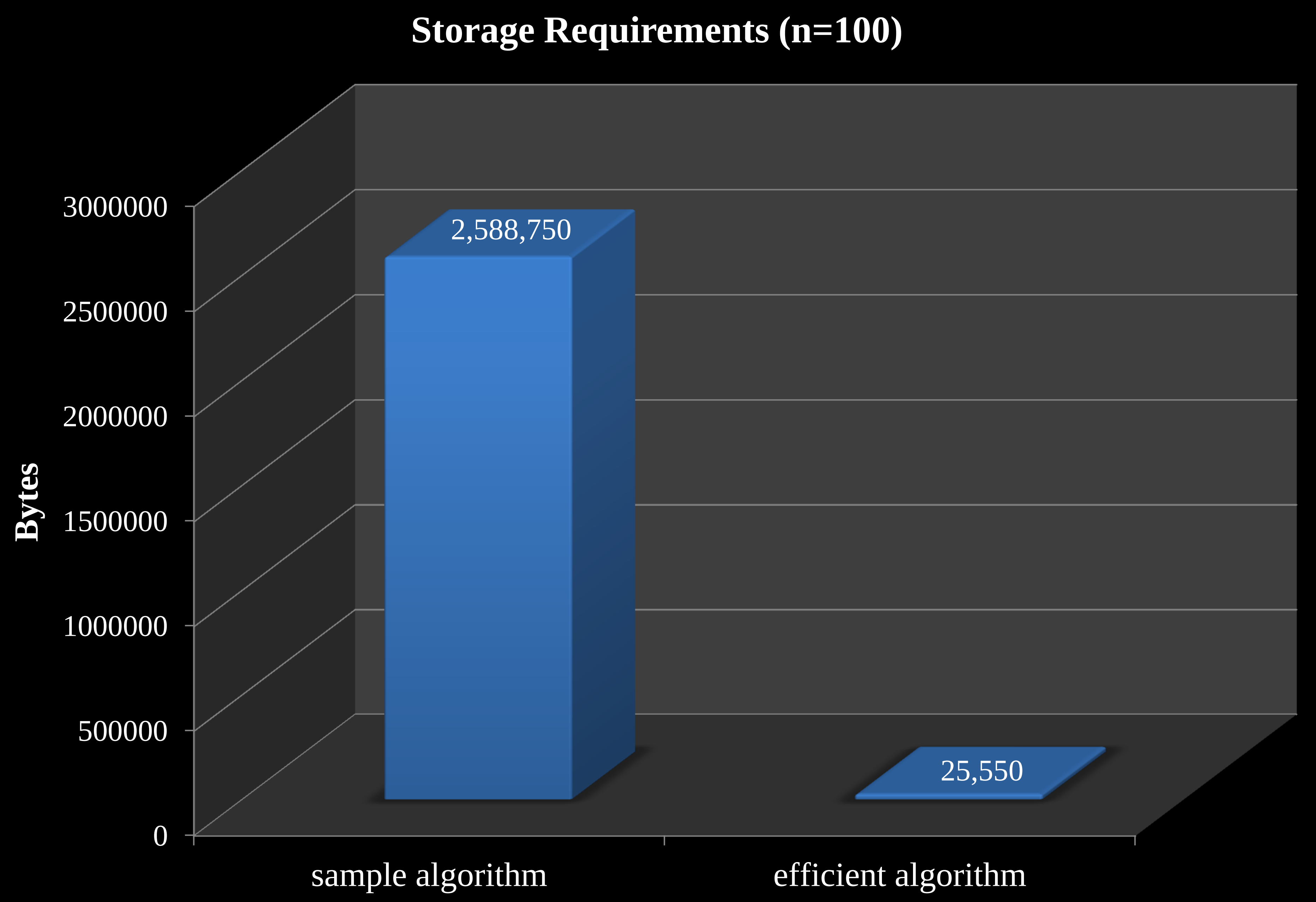


Element by Element Magnitude of x
algorithm result
true solution
magnitude{x}
index
0
0.5
1
1.5
2
2.5
3
3730
3740
3750
3760
3770
3780
3790

Storage Requirements

- Memory use is important for usability on large data sets
- Track memory load of stored variables using $MATLAB$'s $whos()$ at the end of Recursive LS Algorithm iteration ($n = 100$)
- Efficient implementation avoids the storage of Transformation vectors
  - Significantly more memory efficient


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


Storage Requirements (n=100)
Bytes
3000000
2500000
2000000
1500000
1000000
500000
0
2,588,750
25,550
sample algorithm
efficient algorithm

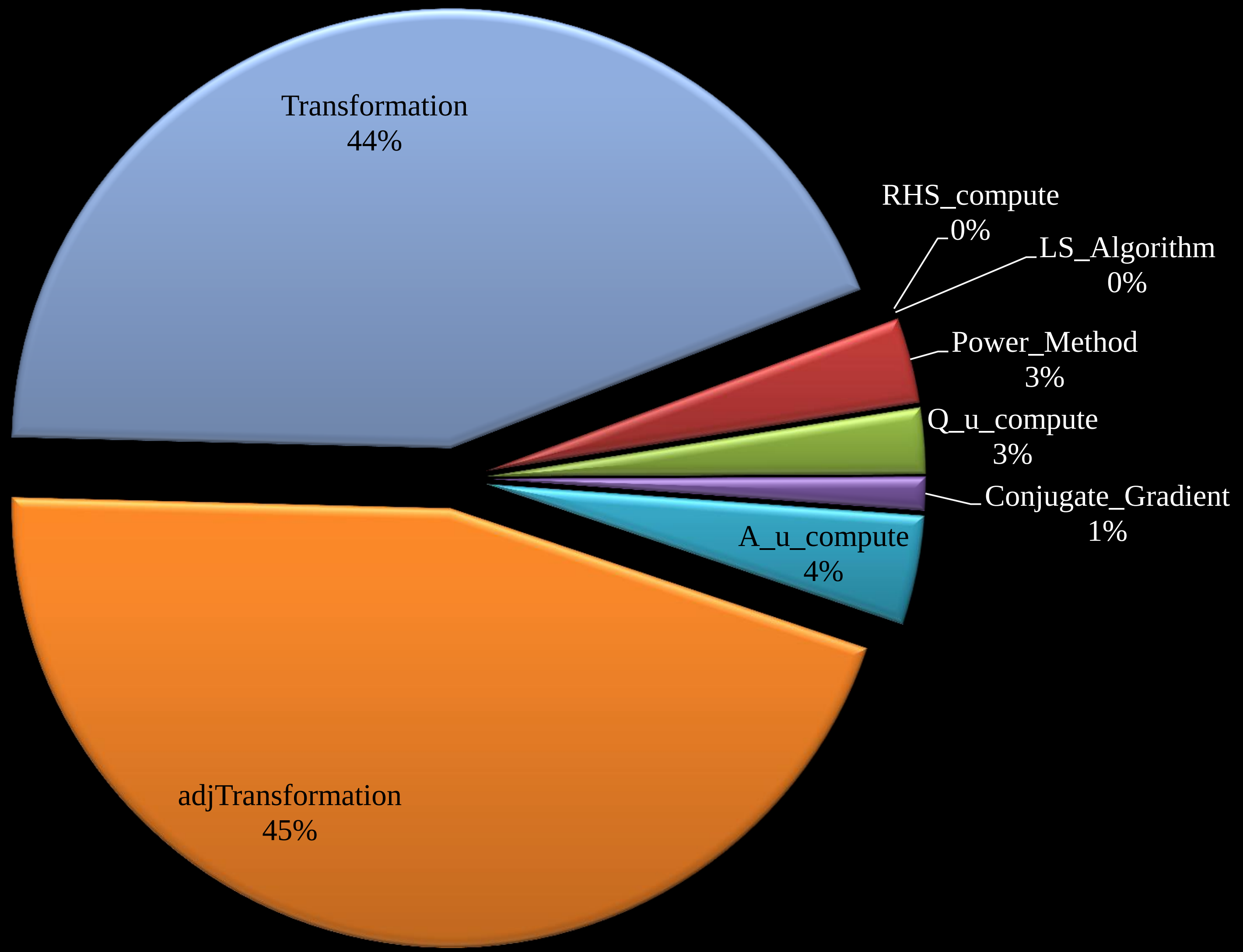
Time Efficiency

- Studying the time consumption of the efficient implementation of $LS\_Algorithm()$
- $n = 10{,}000$
- Stopping tol for P.M. and C.G. $= 10^{-14}$
- Power_Method takes very long to complete
- Most time is spent within $Transformation()$ and $adjTransformation()$

Time Consumption by Function

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
LS_Algorithm	1	181.935 s	0.377 s	
Power_Method	1	94.735 s	5.717 s	
Q_u_compute	12717	89.018 s	4.378 s	
Conjugate_Gradient	219	86.267 s	2.215 s	
A_u_compute	7957	82.393 s	7.321 s	
adjTransformation	20893	82.302 s	82.302 s	
Transformation	29071	79.532 s	79.532 s	
RHS_compute	219	1.659 s	0.093 s	

Time Consumption by Function



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

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).