Multiple sinusoid estimation

1 Package overview

This MATLAB package is designed to efficiently compute a sparse frequency domain explanation of measured responses using the Fast Fourier Transform and continuous optimization methods.

2 Contents

- `generateMeasMat.m` - Code for generating commonly used measurement matrices (regular and compressive)
- `extractSpectrum.m` - Greedy ℓ_0-regularized frequency estimation algorithm
- `example.m` - Example code for frequency estimation (with plots)
- `example_CS.m` - Example code for compressive frequency estimation (with plots)

3 Use case

We denote the unit norm sinusoid $[1 \; e^{j\omega} \; \cdots \; e^{j(N-1)\omega}]^T/\sqrt{N}$ of frequency ω by $\mathbf{x}(\omega)$. The signal is a mixture of K sinusoids: $\sum_{l=1}^{l=K} g_l \mathbf{x}(\omega_l)$ and the nature of measurements supported are:

$$\mathbf{y} = \boldsymbol{A} \times \sum_{l=1}^{l=K} g_l \mathbf{x}(\omega_l) + \mathbf{z}, \; \mathbf{z} \sim \mathcal{CN}\left(\mathbf{0}, \sigma^2 \mathbb{I}_M\right), \tag{1}$$

where the measurement matrix $\boldsymbol{A}$ can be any $M \times N$ matrix. The algorithm outputs estimates of $\{(g_l, \omega_l)\}$ and K, the number of sinusoids in the mixture.

Note on generality: This model supports regular frequency estimation ($\boldsymbol{A} = \mathbb{I}_N$) and compressive frequency estimation ($\boldsymbol{A}$ can be an $M \times N$ matrix with entries picked at random).

4 How to use this package?

Main modules:

- $\boldsymbol{A}$ = `generateMeasMat`(N, N, 'full') - regular measurements
  $\boldsymbol{A}$ = `generateMeasMat`(N, N, 'full', 'hamming') - regular with hamming weights
  $\boldsymbol{A}$ = `generateMeasMat`(N, M, 'cmplx_bernoulli') - M compressive measurements with i.i.d entries chosen from Uniform$\{\pm 1/\sqrt{N}, \pm j/\sqrt{N}\}$
  $\boldsymbol{A}$ = `generateMeasMat`(N, M, 'subsample') - M randomly picked samples (or rows of the identity matrix)

- [omegaList, gainList] = extractSpectrum($\mathbf{y}$, $\boldsymbol{A}$, τ), where $\{(\texttt{gainList}(l), \texttt{omegaList}(l))\}$ are our estimates of the gains and frequencies of the sinusoids in the mixture.

We go over how to use the package.

We start by defining the scenario of the simulations:

```
% Length of Sinusoid
N = 256;

% number of sinusoids in the mixture of sinusoids
K = 5;

% effective SNR for each sinusoid in dB scale
% (all N measurements put together)
SNR = 25*ones(5,1);

% noise level is 0 dB
sigma = 1;

% Choose gains and frequencies of the K sinusoids
omega_true = 2*pi*rand(K,1);
gain_true = sigma * (10 .^ (SNR/20)) .* exp(1j*2*pi*rand(K,1));
```

We then choose the measurement model:

Regular:

```
% normal measurements
M = N;
S = eye(N);
```

Compressive:

```
% Compressive measurements
% Number of compressive measurements
M = round(N/4);
% SNR penalty = N/M
% type of measurement matrix
measure_type = 'subsample';
S = generateMeasMat(N, M, measure_type);
```

We now make measurements using the measurement matrix S:

```
y = S * exp(1j*  (0:(N-1)).' * omega_true.')/sqrt(N) * gain_true + ...
                sigma*(randn(M,1) + 1j*randn(M,1))/sqrt(2);
```

In `example_CS.m` we show how to whiten noise and use `extractSpectrum` when measurements are of the form:

```
y = S * ( exp(1j*  (0:(N-1)).' * omega_true.')/sqrt(N) * gain_true + ...
                sigma*(randn(N,1) + 1j*randn(N,1))/sqrt(2) );
```

`extractSpectrum` expects the parameter τ, which it uses to decide how *sparse* an explanation to return.

```
p_0 = 1e-2;
tau = sigma^2*log(N) - sigma^2*log(log(1/(1-p_0))); % used in stopping criterion
                                % to decide how many sinusoids to use
                                % needs an estimate of the noise-level
```

We are ready to call `extractSpectrum`

```
% Estimation step
[omega_est, gain_est] = extractSpectrum(y, S, tau);
```

and compare estimated parameters with true values.

```
% Plot results
figure;
subplot(1,2,1); polar(omega_true, abs(gain_true), 'bo');
hold on; polar(omega_est, abs(gain_est),'rx');
title(sprintf('Magnitude and frequency of each sinusoid'));
legend({'Truth', 'Estimate'}, 'Location', 'SouthOutside');

subplot(1,2,2); polar(angle(gain_true), abs(gain_true), 'bo');
hold on; polar(angle(gain_est), abs(gain_est), 'rx');
title(sprintf('Magnitude and phase of each sinusoid'));
legend({'Truth', 'Estimate'}, 'Location', 'SouthOutside');
```

We also include files `example.m` and `example_CS.m` which illustrate how to use this package for regular and compressive measurements respectively.

5 Algorithm Details

Inputs to the algorithm are measurements $\mathbf{y}$, the measurement matrix $\boldsymbol{A}$ and an estimate of noise power given by τ. By choosing τ appropriately, it is possible to control sparsity of the resulting explanation. For e.g., large values of τ promote sparsity, whereas small values increase the size of the support set, given by K. When we know the noise-level, we recommend that $\tau = \sigma^2 \log N - \sigma^2 \log\log\left(1/(1-p_0)\right)$ for $p_0 = 10^{-2}$. The algorithm aims to minimize $\mathcal{E}$ given by:

$$\mathcal{E} = \left\| \mathbf{y} - \boldsymbol{A} \sum_{l=1}^{l=K} g_l \mathbf{x}(\omega_l) \right\|^2,$$

where $\{(g_l, \omega_l) \ : \ l = 1, \ldots, K\}$ are estimates of the parameters of the sinusoids in the mixture and the number of sinusoids respectively. Let $\mathcal{P} = \{(g_l, \omega_l)\,,\ l = 1, \ldots, K\}$ denote the set of estimates of the parameters of the sinusoids in the mixture. Let

$$\mathbf{y}_{\mathrm{r}}(\mathcal{P}) = \mathbf{y} - \boldsymbol{A} \sum_{l=1}^{l=K} g_l \mathbf{x}(\omega_l)$$

denote the residual measurement corresponding to this estimate. Note that

$$\mathcal{E}(\mathcal{P}) = \|\mathbf{y}_{\mathrm{r}}(\mathcal{P})\|^2 \tag{2}$$

and $K = |\mathcal{P}|$. We keep adding sinusoids as long as $\exists\ \omega = 2\pi k/N,\ k = 0, 1, \ldots, N-1$ s.t.

$$|\langle \boldsymbol{A}\mathbf{x}(\omega), \mathbf{y}_{\mathrm{r}}\left(\mathcal{P}\right)\rangle|^2 \Big/ \|\boldsymbol{A}\mathbf{x}(\omega)\|^2 \ > \tau.$$

We implement Algorithm 1 to minimize $\mathcal{E}(\mathcal{P})$ in a greedy manner in this package.

6 Results

A lower bound on frequency estimation error is given by the Cramér Rao Bound and we benchmark the algorithm by comparing the mean-squared-error (MSE) against the Cramér Rao Bound. The performance of the algorithm depends on the noise level σ^2 and the *minimum* separation $\min_{k \neq l} |\omega_k - \omega_l|$ between two sinusoids in the mixture which we denote by $\Delta\omega_{\min}$.

Simulation scenario: We consider the problem of estimating the frequencies in a mixture of 5 sinusoids of length $N = 256$. We perform simulations for both regular measurements ($\boldsymbol{A}$ given

Algorithm 1 ℓ_0-regularized frequency estimation: $\mathbf{y}$ are measurements (as given by (1)), $\boldsymbol{A}$ is an $M \times N$ measurement matrix and τ is a parameter which controls sparsity (larger values encourage sparser explanations; when we know the noise level σ^2 in (1), we set $\tau = \sigma^2 \log N - \sigma^2 \log\log\left(1/(1-p_0)\right)$ for p_0 having a small value. We observe that $p_0 \approx 10^{-2}$ works well for both low and high SNR regime). The algorithm returns estimates of all parameters in (1), given by $\{(g_l, \omega_l) \ : \ 1, \ldots, K\}$

Procedure EXTRACTSPECTRUM($\mathbf{y}$, $\boldsymbol{A}$, τ):
$\mathcal{P} = \{\}$

while $\exists\ \omega = 2\pi k/N,\ k = 0, 1, \ldots, N-1$ s.t. $\left|\langle \boldsymbol{A}\mathbf{x}(\omega), \mathbf{y}_{\mathrm{r}}\left(\mathcal{P}\right)\rangle\right|^2 \Big/ \left\|\boldsymbol{A}\mathbf{x}(\omega)\right\|^2 > \tau$ **do**

  IDENTIFY $\hat{\omega} \in [0, 2\pi)$ which maximizes $\left|\langle \boldsymbol{A}\mathbf{x}(\omega), \mathbf{y}_{\mathrm{r}}\left(\mathcal{P}\right)\rangle\right|^2 \Big/ \left\|\boldsymbol{A}\mathbf{x}(\omega)\right\|^2$

  Compute corresponding gain $\hat{g} \leftarrow \langle \boldsymbol{A}\mathbf{x}(\hat{\omega}), \mathbf{y}_{\mathrm{r}}\left(\mathcal{P}\right)\rangle \Big/ \left\|\boldsymbol{A}\mathbf{x}(\hat{\omega})\right\|^2$

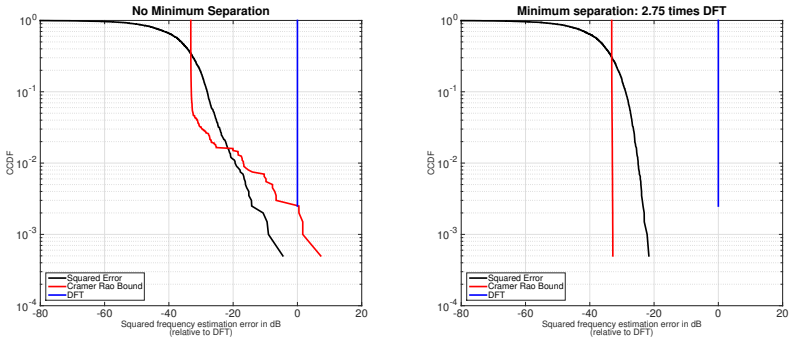
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\hat{g}, \hat{\omega})\}$

  SINGLE REFINEMENT: Refine $(\hat{g}, \hat{\omega})$ using single frequency Newton update algorithm (R_s Newton steps) to obtain improved estimates.

  CYCLIC REFINEMENT: Refine parameters in $\mathcal{P}$ one at a time: For each $(g, \omega) \in \mathcal{P}$ we treat $\mathbf{y}_{\mathrm{r}}(\mathcal{P} \setminus \{(g, \omega)\})$ as the measurement $\mathbf{y}$, and apply single frequency Newton update algorithm. We perform R_c rounds of cyclic refinements.

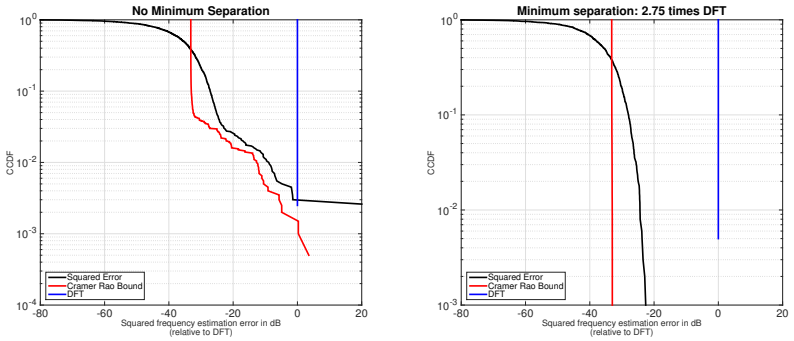
  UPDATE all gains in $\mathcal{P}$ by least squares:
  $X \triangleq [\boldsymbol{A}\mathbf{x}(\omega_1) \ldots \boldsymbol{A}\mathbf{x}(\omega_m)]$, $\{\omega_l\}$ are the frequencies in $\mathcal{P}$,
  $[g_1 \ldots g_m]^T = X^{\dagger}\mathbf{y}$ where $(\cdot)^{\dagger}$ denotes the pseudo-inverse operator.

return $\mathcal{P}$



(a) $\Delta\omega_{\min} = 0$ (b) $\Delta\omega_{\min} = 2.75 \times 2\pi/N$

Figure 1: Regular measurements ($\boldsymbol{A} = \mathbb{I}_N$): $N = 256$, $K = 5$ sinusoids and aggregate SNR of all the sinusoids are set to 25 dB. x-axis is squared error relative to DFT separation of $2\pi/N$ in dB scale and y-axis is CCDF $F(x)$ (how often do errors exceed x).



(a) $\Delta\omega_{\min} = 0$ (b) $\Delta\omega_{\min} = 2.75 \times 2\pi/N$

Figure 2: Compressive measurements (entries of $\boldsymbol{A}$ picked i.i.d. from Uniform$\{\pm 1/\sqrt{N}, \pm j/\sqrt{N}\}$): $N = 256$, $M = 64$, $K = 5$ sinusoids and aggregate SNR of all the sinusoids are set to 25 dB.. x-axis is squared error relative to DFT separation of $2\pi/N$ in dB scale and y-axis is CCDF $F(x)$ (how often do errors exceed x).

by the identity matrix) and $M = 64$ compressive measurements (entries of the $M \times N$ matrix $\boldsymbol{A}$ picked i.i.d. from Uniform$\{\pm 1/\sqrt{N}, \pm j/\sqrt{N}\}$). The noise levels are scaled so that the effective SNR remains same for both regular and compressive measurements. We perform 400 simulation runs for each scenario. In each simulation run, the magnitude of all the gains of the K sinusoids, denoted by g_l, is fixed at 1, while their phases are chosen uniformly at random from Uniform$[0, 2\pi)$. The frequencies of the sinusoids, denoted by ω_l, are chosen uniformly at random from Uniform$[0, 2\pi)$ while also respecting the minimum separation constraints as specified by $\Delta\omega_{\min}$.

We plot the Complementary Cumulative Distribution Function(CCDF) of mean-squared-error along with that of the Cramér Rao Bound for regular measurements in Figure 1 and for $M = 64$ compressive measurements in Figure 2. For both regular and compressive measurements, we perform simulations for two different values of $\Delta\omega_{\min}$, given by 0 and $2.75 \times (2\pi/N)$. From Figures 1 and 2, we see that the algorithm performance closely follows the Cramér Rao Bound in all four settings.