EE 150 - Applications of Convex Optimization in Signal Processing and Communications
Dr. Andre Tkacenko, JPL
Third Term 2011-2012

Homework Set #7 - Solutions

1. First note that we can express $f(t)$ as

$$f(t) = \mathbf{c}(t)^T \mathbf{x}\,,$$

where we have

$$\begin{aligned}
\mathbf{c}(t) &\triangleq \left[\begin{array}{ccccccc} \frac{1}{2} & \cos\left(\frac{2\pi\,(1)\,t}{T}\right) & \cdots & \cos\left(\frac{2\pi K t}{T}\right) & \sin\left(\frac{2\pi\,(1)\,t}{T}\right) & \cdots & \sin\left(\frac{2\pi K t}{T}\right) \end{array}\right]^T, \\
\mathbf{x} &\triangleq \left[\begin{array}{ccccccc} a_0 & a_1 & \cdots & b_1 & \cdots & b_K \end{array}\right]^T.
\end{aligned}$$

To compute the L_2 approximation, we can equivalently minimize the L_2-norm of the error. This leads to the following.

$$\begin{aligned}
||f-y||_2^2 &= \int_{-T/2}^{T/2} \Big(\mathbf{c}(t)^T\mathbf{x} - y(t)\Big)^2\,dt \\
&= \mathbf{x}^T \underbrace{\left[\int_{-T/2}^{T/2} \mathbf{c}(t)\,\mathbf{c}(t)^T\,dt\right]}_{\mathbf{P}} \mathbf{x} - 2\underbrace{\left[\int_{-T/2}^{T/2} \mathbf{c}(t)\,y(t)\right]^T}_{\mathbf{q}^T} + \underbrace{\int_{-T/2}^{T/2} y^2(t)\,dt}_{r} \qquad (1)
\end{aligned}$$

Note that by the orthogonality of the Fourier series basis functions, we have that

$$\mathbf{P} = \frac{T}{2}\left[\begin{array}{cc} 1/2 & \mathbf{0}_{1\times 2K} \\ \mathbf{0}_{2K\times 1} & \mathbf{I}_{2K} \end{array}\right], \qquad (2)$$

and hence $\mathbf{P} \succ \mathbf{0}$. Thus, the optimal choice of $\mathbf{x}$ for the L_2 approximation problem is given by

$$\mathbf{x}^\star = \mathbf{P}^{-1}\mathbf{q}\,. \qquad (3)$$

This can be simplified further as follows. From our expression for $\mathbf{P}$ given in (2), it is clear that we have

$$\mathbf{P}^{-1} = \frac{2}{T}\left[\begin{array}{cc} 2 & \mathbf{0}_{1\times 2K} \\ \mathbf{0}_{2K\times 1} & \mathbf{I}_{2K} \end{array}\right]. \qquad (4)$$

Also, note that from (1), we have

$$[\mathbf{q}]_k = \begin{cases} \frac{1}{2}\int_{-T/2}^{T/2} y(t)\,dt\,, & k = 1 \\ \int_{-T/2}^{T/2} y(t)\cos\left(\frac{2\pi\,(k-1)\,t}{T}\right)dt\,, & k = 2,\ldots,K+1 \\ \int_{-T/2}^{T/2} y(t)\sin\left(\frac{2\pi\,(k-K-1)\,t}{T}\right)dt\,, & k = K+2,\ldots,2K+1 \end{cases}.$$

From this, it can be shown that we have the following for our choice of $y(t)$.

$$[\mathbf{q}]_k = \begin{cases} \frac{1}{4}\,, & k = 1 \\ \frac{1}{(k-1)\,\pi} \sin\left(\frac{\pi\,(k-1)}{2}\right)\,, & k = 2, \ldots, K+1 \\ 0\,, & k = K+2, \ldots, 2K+1 \end{cases} .$$

This can be further simplified to

$$[\mathbf{q}]_k = \begin{cases} \frac{1}{4}\,, & k = 1 \\ \frac{1}{(k-1)\,\pi}\,(-1)^{\frac{k}{2}} \left[\frac{(-1)^{k-1}-1}{2}\right]\,, & k = 2, \ldots, K+1 \\ 0\,, & k = K+2, \ldots, 2K+1 \end{cases} .$$

Combining this result with (3) and (4), we have that

$$[\mathbf{x}^\star]_k = \begin{cases} 1\,, & k = 1 \\ \frac{(-1)^{\frac{k}{2}}\left[(-1)^{k-1}-1\right]}{(k-1)\,\pi}\,, & k = 2, \ldots, K+1 \\ 0\,, & k = K+2, \ldots, 2K+1 \end{cases} .$$

To compute the L_1 approximation, using the summation approximation to the L_1-norm advocated in the problem, we get

$$||f-y||_1 = \frac{T}{2N} \sum_{m=0}^{2N-1} \left|\mathbf{c}(t_m)^T\,\mathbf{x} - y(t_m)\right| .$$

Now, the problem of minimizing the L_1-norm of the error can be posed as the following LP.

$$\begin{array}{ll} \text{minimize} & \frac{T}{2N}\mathbf{1}^T\mathbf{s} \\ \text{subject to} & -s_i \le \mathbf{c}(t_{i-1})^T\,\mathbf{x} - y(t_{i-1}) \le s_i\,,\ i = 1, \ldots, 2N \end{array} .$$

A plot of the periodic function $y(t)$ along with its optimal L_2 and L_1 Fourier series approximations are shown in Figure 1. From the plot, we can see that the L_2 optimal approximation has the familiar Gibbs phenomenon near the discontinuities in y, but the L_1 optimal approximation has much less pronounced oscillation in this neighborhood. Furthermore, the L_1 approximation has much smaller error than the L_2 approximation, except near the discontinuities.

A plot of the histogram of the residuals for both the L_2 and L_1 optimal solutions is shown in Figure 2. As can be seen, the optimal L_1 approximation exhibits a much larger number of residuals near zero than the L_2 approximation. In addition, the L_1 approximation also has a larger number of outlier residuals than the L_2 approximation, as expected.

Sample MATLAB code using `cvx` that can be used to generate the figures created here is shown below.

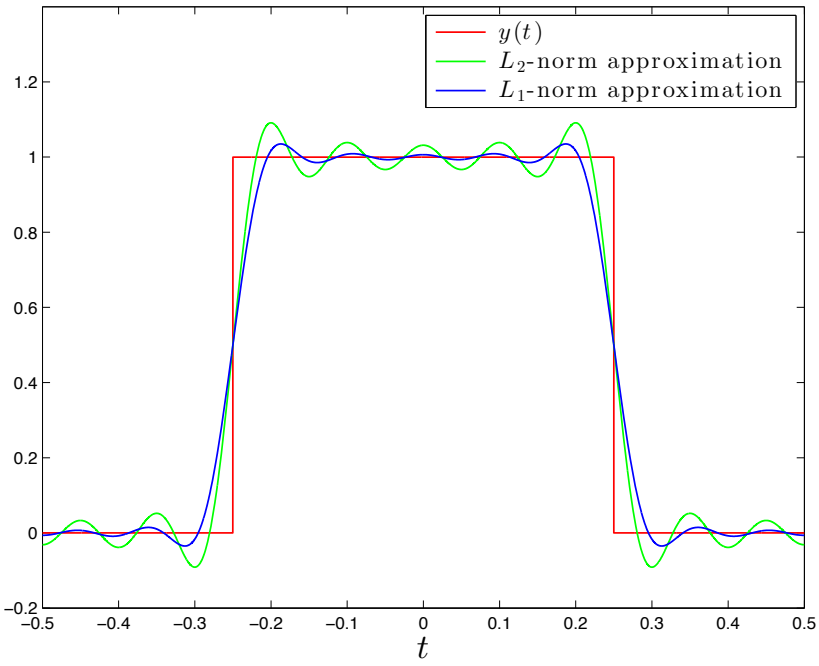


Figure 1: Plot of the 1-periodic function $y(t)$ along with its optimal L_2 and L_1 Fourier series approximations.

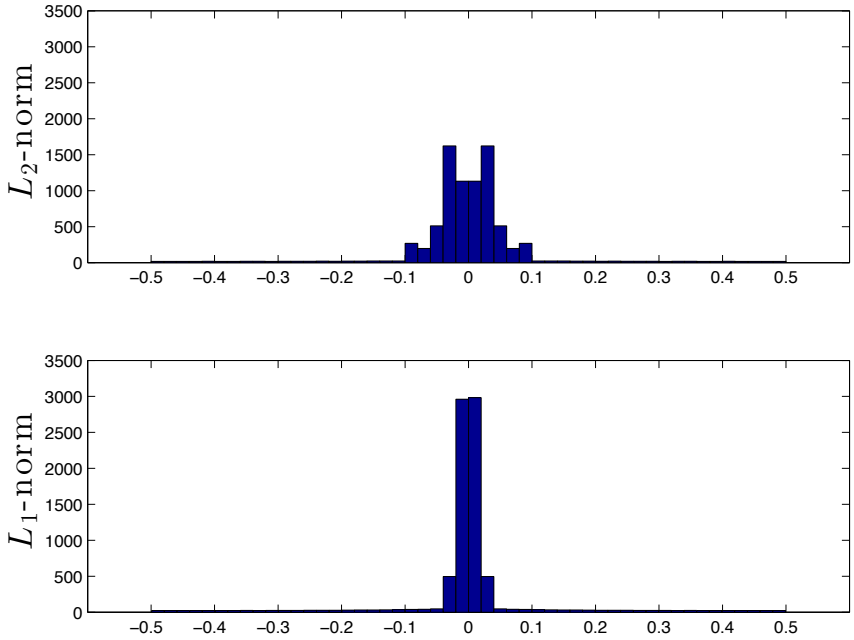


Figure 2: Plot of the histogram of the residuals for both the L_2 and L_1 optimal approximations to y.

```
% Initialize problem parameters
N = 4096;
n = 0:(2.*N-1);
T = 1;
t = (-T/2) + (n.*T./(2.*N));
K = 10;

% Create matrix of Fourier series coefficients
for k = 1:K
cos_mat(k,:) = cos(2.*pi.*k.*t./T);
sin_mat(k,:) = sin(2.*pi.*k.*t./T);
end
coef_mat = [ repmat(0.5,1,2.*N) ; cos_mat ; sin_mat];

% Generate samples of the periodic signal
y = zeros(2.*N,1);
i_live = find(t >= -0.25 & t < 0.25);
y(i_live) = 1;

% Calculate the optimal L_2 approximation
x_L_2 = zeros(2.*K+1,1);
x_L_2(1) = 1;
for k = 2:K+1
x_L_2(k) = (((-1).^(k./2)).*(((-1).^(k-1)) - 1))./((k-1).*pi);
end

% Calculate the optimal L_1 approximation
cvx_begin
    variable x_L_1(2.*K+1)
    minimize(norm(coef_mat'*x_L_1 - y,1))
cvx_end

% Alternate method for calculating the optimal L_1 approximation
cvx_begin
    variable x_L_1(2.*K+1)
    variable s(2.*N)
    minimize((T./(2.*N)).*ones(1,2.*N)*s)
    subject to
        - coef_mat'*x_L_1 + y - s <= 0
        coef_mat'*x_L_1 - y - s <= 0
cvx_end

% Compute the optimal function approximation from the Fourier series
% coefficients
f_L_2 = coef_mat'*x_L_2;
f_L_1 = coef_mat'*x_L_1;

% Plot the periodic function and its L_2 and L_1 Fourier series
```

```
% approximations
figure
plot(t,y,'r')
hold on
plot(t,f_L_2,'g')
plot(t,f_L_1,'b')
xlabel('$t$','Interpreter','LaTeX','FontSize',20)
%print -depsc2 Fourier_series_approx_plot

% Plot histograms or distributions of the residual magnitudes
figure
subplot(2,1,1)
hist(f_L_2 - y,50)
ylabel('$L_{2}$-norm','Interpreter','LaTeX','FontSize',20)
axis([-0.6 0.6 0 3500])
subplot(2,1,2)
hist(f_L_1 - y,50)
ylabel('$L_{1}$-norm','Interpreter','LaTeX','FontSize',20)
axis([-0.6 0.6 0 3500])
%print -depsc2 Fourier_series_hist_plot
```

2. This problem can be simplified through vectorization of the matrix quantities that appear throughout the problem. For example, by defining the horizontal and vertical difference vectors as

$$\mathbf{u}_x \triangleq \mathrm{vec}(U_{k,\ell} - U_{k-1,\ell}) \ , \ \mathbf{u}_y \triangleq \mathrm{vec}(U_{k,\ell} - U_{k,\ell-1}) \ ,$$

with some abuse of notation, then the ℓ_2 variation problem can be equivalently expressed as

$$\begin{array}{ll} \text{minimize} & \left\| \begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \end{bmatrix} \right\|_2 \\ \text{subject to} & U_{k,\ell} = U_{k,\ell}^{\text{orig}} \end{array} ,$$

while the total variation problem can be expressed as

$$\begin{array}{ll} \text{minimize} & \left\| \begin{bmatrix} \mathbf{u}_x \\ \mathbf{u}_y \end{bmatrix} \right\|_1 \\ \text{subject to} & U_{k,\ell} = U_{k,\ell}^{\text{orig}} \end{array} ,$$

In MATLAB, with the help of `cvx`, this can be implemented as follows.

```
% Load total variation image interpolation data
tv_img_interp

% Calculate the optimal l_2 variation interpolant
cvx_begin
    variable Ul2(m,n);
    Ux = Ul2(2:end,2:end) - Ul2(2:end,1:end-1); % Compute the x
                                                % (horizontal) variations
```

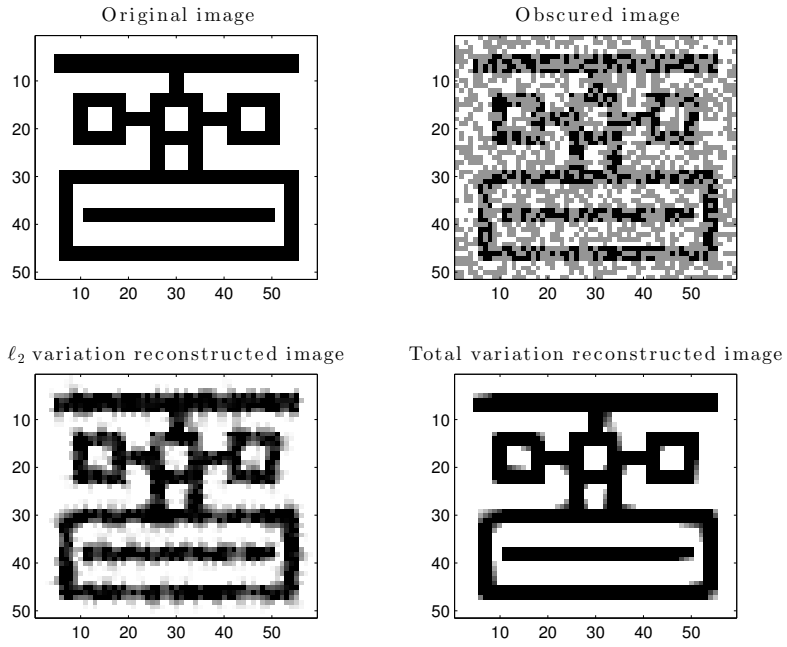


Figure 3: Plot of the original, obscured, ℓ_2 variation interpolated, and total variation interpolated images.

```
    Uy = Ul2(2:end,2:end) - Ul2(1:end-1,2:end); % Compute the y (vertical)
                                                % variations
    minimize(norm([Ux(:);Uy(:)],2));            % minimize the l_2
                                                % roughness measure
    subject to
        Ul2(Known) == Uorig(Known);             % Fix known pixel values
                                                % values
cvx_end

% Calculate the optimal total variation interpolant
cvx_begin
    variable Utv(m,n);
    Ux = Utv(2:end,2:end) - Utv(2:end,1:end-1); % Compute the x
                                                % (horizontal) variations
    Uy = Utv(2:end,2:end) - Utv(1:end-1,2:end); % Compute the y (vertical)
                                                % variations
    minimize(norm([Ux(:);Uy(:)],1));            % minimize the total
                                                % variation measure
    subject to
        Utv(Known) == Uorig(Known);             % Fix known pixel values
cvx_end
```

A plot of the original and obscured image, along with the ℓ_2 variation and total variation interpolants are shown in Figure 3. As can be seen, for this particular type of sparse original image, the ℓ_1 based total variation reconstruction interpolates the missing data much better than the ℓ_2 variation based approach.

3. (a) Ideally, the problem we would like to solve is the following one.

$$\begin{array}{ll} \text{minimize} & \text{card}\left(\widehat{\mathbf{A}}\right) + \text{card}\left(\widehat{\mathbf{B}}\right) \\ \text{subject to} & \displaystyle\sum_{t=1}^{T-1} \left|\left|\mathbf{W}^{-1/2}\left(\mathbf{x}(t+1) - \widehat{\mathbf{A}}\mathbf{x}(t) - \widehat{\mathbf{B}}\mathbf{u}(t)\right)\right|\right|_2^2 \leq n\,(T-1) + 2\sqrt{2n\,(T-1)} \end{array}.$$

Here, $\text{card}(\mathbf{X})$ is the cardinality of matrix $\mathbf{X}$, i.e., the number of nonzero entries. The issue with trying to solve this problem is that the objective is nonconvex. To address this issue, we will use the common heuristic of minimizing the ℓ_1 norm of the entries of $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$. Using the standard vectorization operator, this leads to the following convex optimization problem.

$$\begin{array}{ll} \text{minimize} & \left|\left|\text{vec}\left(\widehat{\mathbf{A}}\right)\right|\right|_1 + \left|\left|\text{vec}\left(\widehat{\mathbf{B}}\right)\right|\right|_1 \\ \text{subject to} & \displaystyle\sum_{t=1}^{T-1} \left|\left|\mathbf{W}^{-1/2}\left(\mathbf{x}(t+1) - \widehat{\mathbf{A}}\mathbf{x}(t) - \widehat{\mathbf{B}}\mathbf{u}(t)\right)\right|\right|_2^2 \leq n\,(T-1) + 2\sqrt{2n\,(T-1)} \end{array}.$$

Intuitively, it can be said that the constraint will always be tight as relaxing the requirement on the implied errors allows for more freedom to reduce the sum of the ℓ_1-norms of $\text{vec}\left(\widehat{\mathbf{A}}\right)$ and $\text{vec}\left(\widehat{\mathbf{B}}\right)$.

(b) This problem can be easily solved in MATLAB with the help of `cvx` using the following code.

```
% Load problem data
sparse_lds_data

% Set the fit tolerance
fit_tol = sqrt(n*(T-1) + 2*sqrt(2*n*(T-1)));

% Compute the l_1 approximation to the sparse dynamical system fit problem
cvx_begin
    variables Ahat(n,n) Bhat(n,m);
    minimize(sum(norms(Ahat,1)) + sum(norms(Bhat,1)));
    subject to
    norm(inv(Whalf)*(xs(:,2:T) - Ahat*xs(:,1:T-1) - Bhat*us),'fro') ...
        <= fit_tol;
cvx_end
disp(cvx_status)

% Round near-zero elements to zero
Ahat = Ahat .* (abs(Ahat) >= 0.01);
Bhat = Bhat .* (abs(Bhat) >= 0.01);

% Display the number of false positives and negatives
disp(['false positives, Ahat: ' num2str(nnz((Ahat ~= 0) & (A == 0)))])
disp(['false negatives, Ahat: ' num2str(nnz((Ahat == 0) & (A ~= 0)))])
disp(['false positives, Bhat: ' num2str(nnz((Bhat ~= 0) & (B == 0)))])
disp(['false negatives, Bhat: ' num2str(nnz((Bhat == 0) & (B ~= 0)))])
```

With the given problem data, we get 1 false positive (at the $(6,5)$-th entry) and 2 false negatives (at the $(4,1)$-th and $(6,1)$-th entries) for $\widehat{\mathbf{A}}$, and no false positives and 1 false negative (at the $(7,2)$-th entry) for $\widehat{\mathbf{B}}$. The matrix estimates are

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.1483 & -0.0899 & 0.1375 & -0.0108 & 0 & 0 \\ 0 & 0 & 0 & 0.9329 & 0 & 0 & 0.2868 & 0 \\ 0 & 0.2055 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.0190 & 0.9461 & 0 & 0.8697 \\ 0 & 0 & 0 & 0 & 0.2065 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and

$$\widehat{\mathbf{B}} = \begin{bmatrix} -1.4717 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.2832 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.6363 & -0.0456 & 0 & 0 \\ 0 & 1.4117 & 0 & 0 \\ -0.0936 & 0 & 0 & -0.7755 \\ 0 & -0.5705 & 0 & 0 \end{bmatrix}$$

Finally, there are a lot of methods that will do better than this, usually be taking the above solutions as a starting point and then 'polishing' the result after that. Several of these have been shown to give fairly reliable, if modest, improvements.

4. (a) To find $q_i^{\min}$, we solve the convex optimization problem

$$\begin{array}{ll} \text{minimize} & q_i \\ \text{subject to} & \mathbf{l} \preceq \mathbf{S}\mathbf{q} \preceq \mathbf{u}\,,\ \mathbf{q} \succeq \mathbf{0} \end{array},$$

with variable $\mathbf{q} \in \mathbb{R}^n$. Then, we set $q_i^{\min} = q_i^\star$. Similarly, to find $q_i^{\max}$, we solve the convex optimization problem

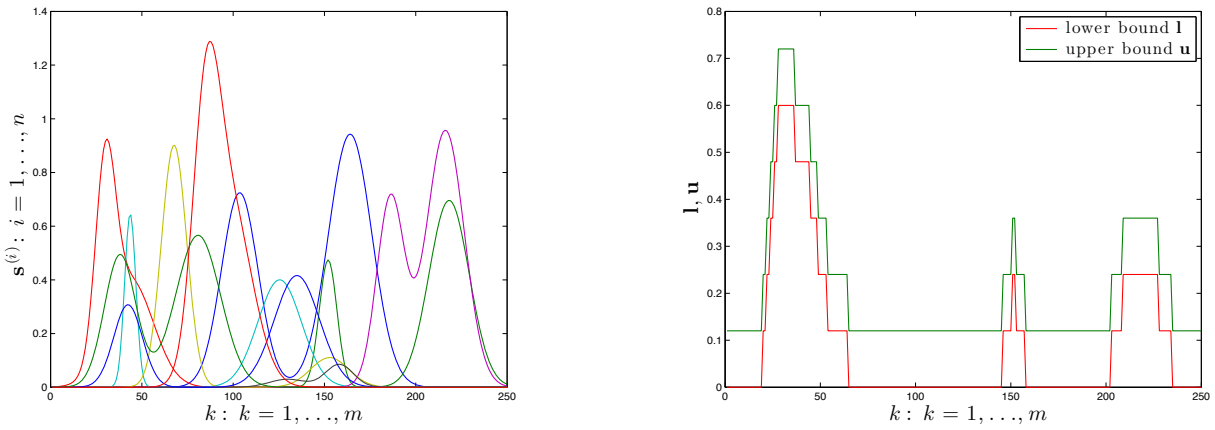
$$\begin{array}{ll} \text{maximize} & q_i \\ \text{subject to} & \mathbf{l} \preceq \mathbf{S}\mathbf{q} \preceq \mathbf{u}\,,\ \mathbf{q} \succeq \mathbf{0} \end{array},$$

and then we set $q_i^{\max} = q_i^\star$.

(b) Running `spectrum_data.m` yields plots of the spectra of the compounds $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(n)}$, alongside the lower and upper bounds $\mathbf{l}$ and $\mathbf{u}$ shown in Figure 4(a) and (b), respectively.

The following MATLAB code using `cvx` can be used to find the range of possible values for each compound quantity $q_i \in \left[q_i^{\min}, q_i^{\max}\right]$.

```
% Load the spectrum compound data
spectrum_data
```



(a) (b)

Figure 4: Plots of compound data: (a) spectra of the compounds and (b) lower and upper bounds for the sample.

```
% Calculate the quantity bounds for each compound using a for-loop
qmin = zeros(n,1); qmax = zeros(n,1);
for i = 1:n
    % Compute the minimum bound for the i-th quantity
    cvx_begin
        variable q(n)
        l <= S*q; u >= S*q;
        q >= 0;
        minimize(q(i))
    cvx_end
    qmin(i) = q(i);
    % Compute the maximum bound for the i-th quantity
    cvx_begin
        variable q(n)
        l <= S*q; u >= S*q;
        q >= 0;
        maximize(q(i))
    cvx_end
    qmax(i) = q(i);
end

% Display the quantity bounds
[qmin qmax]

% Plot the quantity bounds
figure; hold on;
for i = 1:n
    plot([i,i],[qmax(i),qmin(i)],'o-');
end
```

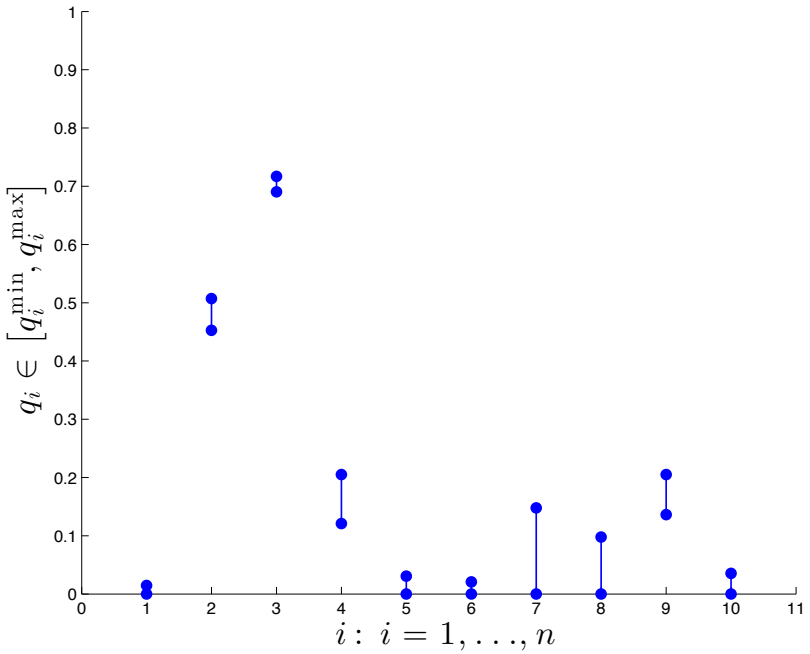


Figure 5: Plot of the range of possible values for each quantity $q_i \in \left[q_i^{\min}, q_i^{\max}\right]$ for $i = 1, \ldots, n$.

```
axis([0,11,0,1]);
```

A plot of the range of possible values for each quantity q_i is shown in Figure 5. From the output of `cvx`, we have $q_4^{\min} = 0.1211$ and $q_4^{\max} = 0.2052$.

***5.** (a) First note that we have

$$|G(\theta)| = \left|\left|\begin{bmatrix} \mathrm{Re}[G(\theta)] \\ \mathrm{Im}[G(\theta)] \end{bmatrix}\right|\right|_2 .$$

Then, note that if we define the following quantities

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{w}_{\mathrm{re}} \\ \mathbf{w}_{\mathrm{im}} \end{bmatrix} \in \mathbb{R}^{2n}, \ \text{where} \ [\mathbf{w}_{\mathrm{re}}]_k = w_{\mathrm{re},k}, [\mathbf{w}_{\mathrm{im}}]_k = w_{\mathrm{im},k} ,$$

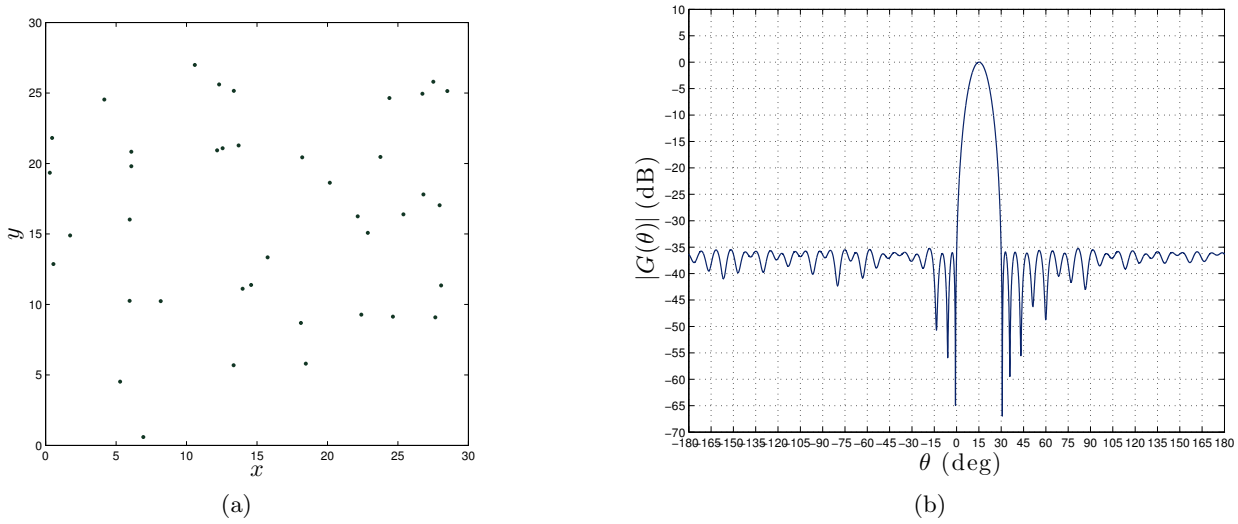
$$\mathbf{C}(\theta) \triangleq \begin{bmatrix} \cos\phi_1(\theta) & \cdots & \cos\phi_n(\theta) & -\sin\phi_1(\theta) & \cdots & -\sin\phi_n(\theta) \\ \sin\phi_1(\theta) & \cdots & \sin\phi_n(\theta) & \cos\phi_1(\theta) & \cdots & \cos\phi_n(\theta) \end{bmatrix} \in \mathbb{R}^{2\times 2n} ,$$

then we have

$$\begin{bmatrix} \mathrm{Re}[G(\theta)] \\ \mathrm{Im}[G(\theta)] \end{bmatrix} = \mathbf{C}(\theta)\,\mathbf{x} .$$

To express the design problem as an SOCP, define the following quantities.

$$\begin{aligned} \mathbf{A}_\ell &\triangleq \mathbf{C}(\theta_\ell) , \\ \mathbf{B} &\triangleq \mathbf{C}\left(\theta^{\mathrm{tar}}\right) , \\ \mathbf{d} &\triangleq \begin{bmatrix} 1 \\ 0 \end{bmatrix} . \end{aligned}$$



(a) (b)

Figure 6: Plots of (a) the antenna array geometry and (b) antenna array magnitude response $|G(\theta)|$ in dB.

Note that the constraint $G\big(\theta^{\rm tar}\big) = 1$ is equivalent to the following.

$$\left[\begin{array}{c} \mathrm{Re}\big[G\big(\theta^{\rm tar}\big)\big] \\ \mathrm{Im}\big[G\big(\theta^{\rm tar}\big)\big] \end{array} \right] = \left[\begin{array}{c} 1 \\ 0 \end{array} \right] \Longleftrightarrow \mathbf{B}\mathbf{x} = \mathbf{d}\,,$$

which is an affine equality constraint on $\mathbf{x}$. Thus, the antenna array design problem becomes the following SOCP.

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & ||\mathbf{A}_\ell \mathbf{x}||_2 \leq t\,,\ \ell = 1, \ldots, N \\ & \mathbf{B}\mathbf{x} = \mathbf{d} \end{array} .$$

(b) This part of the problem can be simplified by noting that we can alternatively express $G(\theta)$ as

$$G(\theta) = \mathbf{e}^\dagger(\theta)\,\mathbf{w}\,,$$

where $\mathbf{e}(\theta) \in \mathbb{C}^n$ and $\mathbf{w} \in \mathbb{C}^n$ are given by

$$\begin{array}{rcl} [\mathbf{e}(\theta)]_k & = & e^{-j\left(\frac{2\pi x_k}{\lambda}\cos\theta + \frac{2\pi y_k}{\lambda}\sin\theta\right)}\,,\ k = 1, \ldots, n \\ [\mathbf{w}]_k & = & w_k\,,\ k = 1, \ldots, n \end{array}$$

Using this simplification, for the antenna array geometry shown in Figure 6(a), we obtain the antenna array magnitude response $|G(\theta)|$ in dB as shown in Figure 6(b). From Figure 6(b), it can be seen that the array response yields unity gain at the target angle of arrival $\theta^{\rm tar}$ and over 35 dB of attenuation outside of the beamwidth 2Δ.

Sample MATLAB code using `cvx` which was used to obtain the results here is shown below.

```
% Define design problem parameters
n = 40;
```

```
lambda = 2*pi;
theta_tar_deg = 15; theta_tar = deg2rad(theta_tar_deg);
Delta_deg = 15;     Delta = deg2rad(Delta_deg);
N = 1000;

% Generate antenna positions
rand('state',0);
x = 30*rand(n,1);
y = 30*rand(n,1);

% Construct the discretized angle of arrival values
theta_des = linspace(theta_tar+Delta,theta_tar+(2*pi)-Delta);
wn = 2*pi/lambda;
e = exp(-j.*(((wn.*x)*cos(theta_des)) + ((wn.*y)*sin(theta_des))));
e_tar = exp(-j.*(((wn.*x)*cos(theta_tar)) + ((wn.*y)*sin(theta_tar))));

% Compute the optimal antenna array using cvx
cvx_begin
    variable w(n) complex
    minimize(max(abs(e'*w)))
    subject to
        e_tar'*w == 1;
cvx_end

% Plot the antenna array response
theta_plot_deg = linspace(-180,180,8192);
theta_plot = deg2rad(theta_plot_deg);
e = exp(-j.*(((wn.*x)*cos(theta_plot)) + ((wn.*y)*sin(theta_plot))));
G = e'*w;
plot(theta_plot_deg,20*log10(abs(G)))
xlim([-180 180])
grid
xlabel('$\theta$ (deg)','Interpreter','LaTeX','FontSize',20)
ylabel('$\left| G\!\left( \theta \right) \right|$ (dB)','Interpreter',...
    'LaTeX','FontSize',20)
```