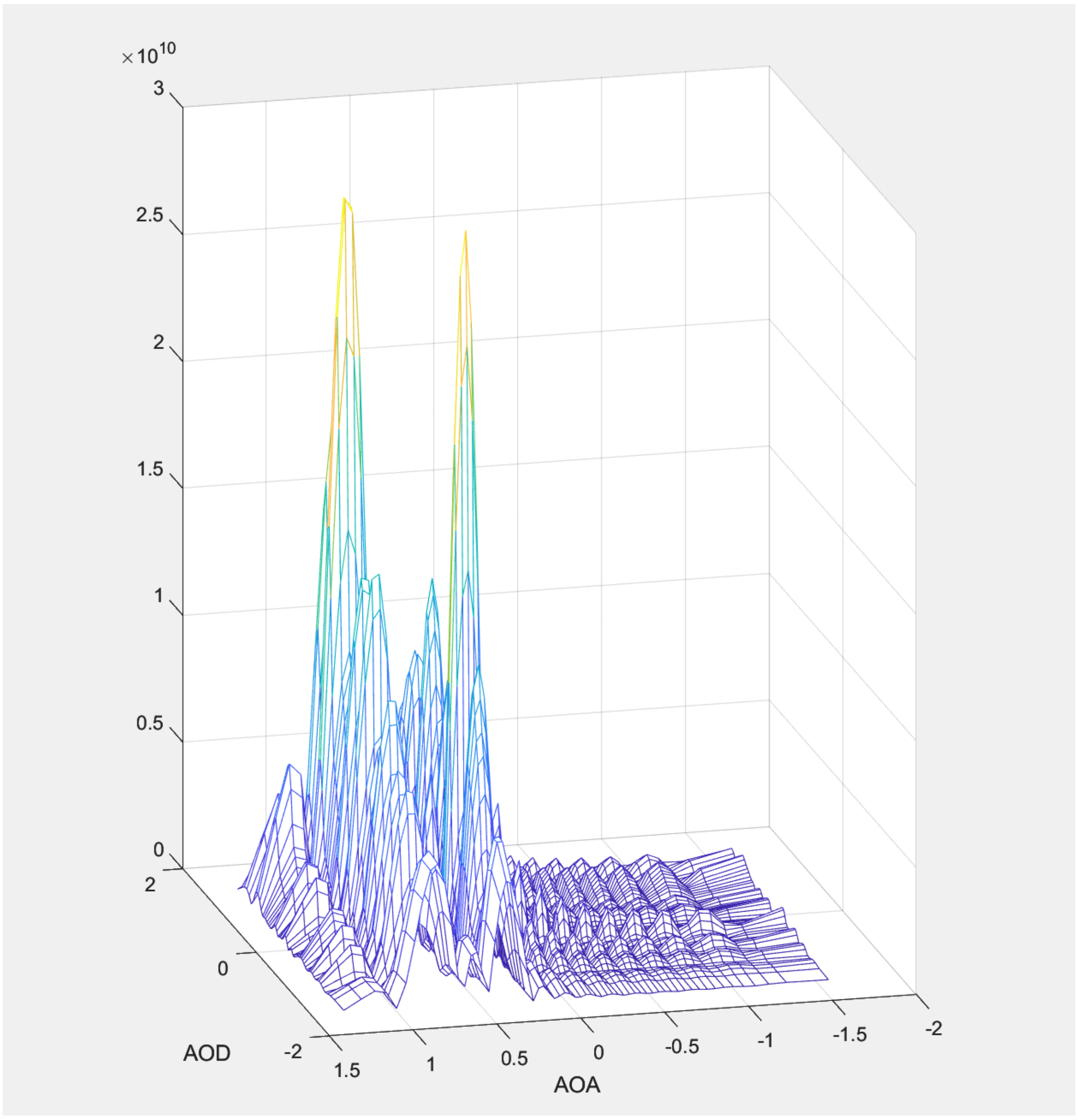
```matlab
close all; clear all; clc;
%% System parameter
L = 2; % Number of path L0 is LOS
BW = 100; % Bandwidth
N = 1; % Number of subcarrier
Ts = 1/BW; % Sampling Period
Nt = 32; % Number of Tx
Nr = Nt; % Number of Rx
Nb=Nt*2; % Number of beams in dictionary;
Ns = 20; % Number of beams sent
posTx = [0 0]'; % position of Tx
posRx = [3 1]'; % position of Rx
c = 300; % Speed of light meter/us
sigma=0.1; % Noise standard deviation
%% Scatter point
SP = [2 2]; % random points uniformly placed in a 20 m x 20 m area
%% Channel parameter
TOA(1) = norm(abs(posRx))/c; % t = s/v, LOS Time of arrival
AOD(1) = atan2(posRx(2), posRx(1)); % LOS Angle of departure
AOA(1) = atan2(posRx(2), posRx(1)) - pi; % LOS Angle of arrival
for i=1:L-1
   AOD(i+1) = atan2(SP(i, 2), SP(i, 1));
   AOA(i+1) = atan2(abs(SP(i, 2)-posRx(2)), abs(SP(i, 1)-posRx(1)));
   TOA(i+1) = (norm(SP(i,:)-posTx)+norm(posRx-SP(i,:)))/c;
end
h=10*ones(1,L); % Channel coefficient
%% Channel
H=zeros(Nr,Nt,N);
for i=1:N % i-th subcarrier
   g = zeros(L, L); % gamma
   At = []; % angular domain correlation matries of tramsmitter
   Ar = []; % angular domain correlation matries of arriver
   for k=1:L % k-th path
       g(k, k) = sqrt(Nt*Nr) * h(k) * exp(-j*2*pi*(i-1)*TOA(k)/(N*Ts));
       At = [At, getResponse(Nt, sin(AOA(k)))];
       Ar = [Ar, getResponse(Nr, sin(AOD(k)))];
   end
   At = At';
   H(:,:,i) = Ar * g * At;
end
%% Create Dictionary
Ut=zeros(Nt,Nb);
Ur=zeros(Nr,Nb);
aa=-Nb/2:Nb/2-1;
aa=2*aa/Nb; % dictionary of spatial frequencies
for m=1:Nb
   Ut(:,m)=getResponse(Nt,aa(m))*sqrt(Nt);
   Ur(:,m)=getResponse(Nr,aa(m))*sqrt(Nr);
end
%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space
figure;
subplot(1, 2, 1);
Hb=Ur'*H(:,:,1)*Ut;
```

```matlab
mesh(asin(aa),asin(aa),abs(Hb));
xlabel('AOD'); ylabel('AOA');
%% Generate the observation and beamformers
y=zeros(Nr,Ns,N);
F=zeros(Nt,Ns,N);
for k=1:Ns
   for n=1:N
       F(:,k,n)=exp(1j*rand(Nt,1)*2*pi); % beamformers
       y(:,k,n)=H(:,:,n)*F(:,k,n)+sigma/sqrt(2)*(randn(Nr,1)+1j*randn(Nr,1));
% y = HFx + N
   end
end
%% Vectorize and generation of the basis
Omega=zeros(Nr*Ns,Nb*Nb,N);
yb=zeros(Nr*Ns,N);
h = Hb(:); % Vecterize
for n=1:N
   Omega(:,:,n) = kron((Ut'*F(:,:,n)).', Ur);
   yb(:,n) = Omega(:,:,n)*h + sigma/sqrt(2); % * (randn(Nr*Ns,
1)+1j*randn(Nr*Ns, 1)); % y=Omega*h
end
%% OMP
% [Omega_k, x_k] = OMP(yb, Omega, L);
% hb = zeros(size(Hb));
% for i=1:length(Omega_k)
%     hb(Omega_k(i)) = x_k(i);
% end
% subplot(1, 2, 2);
% mesh(asin(aa),asin(aa),abs(hb));
% xlabel('AOD'); ylabel('AOA');
%% Transform
yb_tr = zeros(size(yb, 1)*2, 1);
Omega_tr = zeros(size(Omega, 1)*2, size(Omega, 2)*2);
for i=1:length(yb)
   yb_tr(2*i-1) = real(yb(i));
   yb_tr(2*i) = imag(yb(i));
end
for i=1:size(Omega, 1)
   for j=1:size(Omega, 2)
       Omega_tr(2*i-1, 2*j-1) = real(Omega(i, j));
       Omega_tr(2*i-1, 2*j) = -imag(Omega(i, j));
       Omega_tr(2*i, 2*j-1) = imag(Omega(i, j));
       Omega_tr(2*i, 2*j) = real(Omega(i, j));
   end
end
%% Norm 1
beta = 10;
lamda = 0.05;
nIter = 100;
h_k = zeros(size(Omega_tr, 2), nIter+1);
r_k = zeros(size(Omega_tr, 2), nIter+1);
h_k(:, 1) = 50;
for i=1:nIter
```

```
   r_k(:,i) = h_k(:,i) - beta*Omega_tr' * (yb_tr - Omega_tr*h_k(:,i));
   for j=1:length(h_k)
       h_k(j, i+1) = sign(r_k(j, i)) * (abs(r_k(j, i) - beta*lamda/2));
   end
   if (i~=nIter)
       h_k(:, i+1) = h_k(:, i+1)/max(h_k(:, i+1));
   end
end
h_est = zeros(size(Omega, 2), 1);
for i=1:length(h_est)
   h_est(i) = h_k(2*i-1, nIter+1) + 1j*h_k(2*i, nIter+1);
end
hh = zeros(Nb, Nb);
for i=1:length(hh)
   for j=1:length(hh)
       hh(i, j) = h_est(i+Nb*(j-1));
   end
end
subplot(1, 2, 2);
mesh(asin(aa),asin(aa),abs(hh));
xlabel('AOD'); ylabel('AOA');
```


×10^10
3
2.5
2
1.5
1
0.5
0
2
0
-2
AOD
1.5
1
0.5
0
-0.5
-1
-1.5
-2
AOA

```matlab
close all; clear all; clc;
%% System parameter
L = 2; % Number of path L0 is LOS
BW = 100; % Bandwidth
N = 1; % Number of subcarrier
Ts = 1/BW; % Sampling Period
Nt = 32; % Number of Tx
Nr = Nt; % Number of Rx
Nb=Nt*2; % Number of beams in dictionary;
Ns = 20; % Number of beams sent
posTx = [0 0]'; % position of Tx
posRx = [3 1]'; % position of Rx
c = 300; % Speed of light meter/us
sigma=0.1; % Noise standard deviation
%% Scatter point
SP = [2 2]; % random points uniformly placed in a 20 m x 20 m area
%% Channel parameter
TOA(1) = norm(abs(posRx))/c; % t = s/v, LOS Time of arrival
AOD(1) = atan2(posRx(2), posRx(1)); % LOS Angle of departure
AOA(1) = atan2(posRx(2), posRx(1)) - pi; % LOS Angle of arrival
for i=1:L-1
   AOD(i+1) = atan2(SP(i, 2), SP(i, 1));
   AOA(i+1) = atan2(abs(SP(i, 2)-posRx(2)), abs(SP(i, 1)-posRx(1)));
   TOA(i+1) = (norm(SP(i,:)-posTx)+norm(posRx-SP(i,:)))/c;
end
h=10*ones(1,L); % Channel coefficient
%% Channel
H=zeros(Nr,Nt,N);
for i=1:N % i-th subcarrier
   g = zeros(L, L); % gamma
   At = []; % angular domain correlation matries of tramsmitter
   Ar = []; % angular domain correlation matries of arriver
   for k=1:L % k-th path
       g(k, k) = sqrt(Nt*Nr) * h(k) * exp(-j*2*pi*(i-1)*TOA(k)/(N*Ts));
       At = [At, getResponse(Nt, sin(AOA(k)))];
       Ar = [Ar, getResponse(Nr, sin(AOD(k)))];
   end
   At = At';
   H(:,:,i) = Ar * g * At;
end
%% Create Dictionary
Ut=zeros(Nt,Nb);
Ur=zeros(Nr,Nb);
aa=-Nb/2:Nb/2-1;
aa=2*aa/Nb; % dictionary of spatial frequencies
for m=1:Nb
   Ut(:,m)=getResponse(Nt,aa(m))*sqrt(Nt);
   Ur(:,m)=getResponse(Nr,aa(m))*sqrt(Nr);
end
%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space
figure;
subplot(1, 2, 1);
Hb=Ur'*H(:,:,1)*Ut;
```

```matlab
mesh(asin(aa),asin(aa),abs(Hb));
xlabel('AOD'); ylabel('AOA');
%% Generate the observation and beamformers
y=zeros(Nr,Ns,N);
F=zeros(Nt,Ns,N);
for k=1:Ns
   for n=1:N
       F(:,k,n)=exp(1j*rand(Nt,1)*2*pi); % beamformers
       y(:,k,n)=H(:,:,n)*F(:,k,n)+sigma/sqrt(2)*(randn(Nr,1)+1j*randn(Nr,1));
% y = HFx + N
   end
end
%% Vectorize and generation of the basis
Omega=zeros(Nr*Ns,Nb*Nb,N);
yb=zeros(Nr*Ns,N);
h = Hb(:); % Vecterize
for n=1:N
   Omega(:,:,n) = kron((Ut'*F(:,:,n)).', Ur);
   yb(:,n) = Omega(:,:,n)*h + sigma/sqrt(2); % * (randn(Nr*Ns,
1)+1j*randn(Nr*Ns, 1)); % y=Omega*h
end
%% OMP
% [Omega_k, x_k] = OMP(yb, Omega, L);
% hb = zeros(size(Hb));
% for i=1:length(Omega_k)
%     hb(Omega_k(i)) = x_k(i);
% end
% subplot(1, 2, 2);
% mesh(asin(aa),asin(aa),abs(hb));
% xlabel('AOD'); ylabel('AOA');
[xk, ~, ~] = solveLASSOProblem(Omega,yb,1000);

%% Transform
yb_tr = zeros(size(yb, 1)*2, 1);
Omega_tr = zeros(size(Omega, 1)*2, size(Omega, 2)*2);
for i=1:length(yb)
   yb_tr(2*i-1) = real(yb(i));
   yb_tr(2*i) = imag(yb(i));
end
for i=1:size(Omega, 1)
   for j=1:size(Omega, 2)
       Omega_tr(2*i-1, 2*j-1) = real(Omega(i, j));
       Omega_tr(2*i-1, 2*j) = -imag(Omega(i, j));
       Omega_tr(2*i, 2*j-1) = imag(Omega(i, j));
       Omega_tr(2*i, 2*j) = real(Omega(i, j));
   end
end

%% Norm 1
beta = 10;
lamda = 0.05;
nIter = 100;
h_k = zeros(size(Omega_tr, 2), nIter+1);
```

```
r_k = zeros(size(Omega_tr, 2), nIter+1);
y_k = zeros(size(Omega_tr, 2), nIter+1);
tk = 1;
tp = 1;

h_k(:, 1) = sparse(size(h_k, 1), 1);

for i = 1:nInter
   y_k(:, i+1) = h_k(:,i) + ((tp-1)/tk)*(h_k
   for
end


h_est = zeros(size(Omega, 2), 1);
for i=1:length(h_est)
   h_est(i) = h_k(2*i-1, nIter+1) + 1j*h_k(2*i, nIter+1);
end
hh = zeros(Nb, Nb);
for i=1:length(hh)
   for j=1:length(hh)
       hh(i, j) = h_est(i+Nb*(j-1));
   end
end
subplot(1, 2, 2);
mesh(asin(aa),asin(aa),abs(hh));
xlabel('AOD'); ylabel('AOA');
```

ISTA

```
clear all;
close all;
clc;
m = 50;
n = 90;
sigma = 0.1;
A = randn(m,n);

s0 = 9; % No of non zero elements in x

% Range of x is from [-10 -1] U [1 10] and it is a uniform distribution
% So we will construct a variable int_ind (interval indicator) which is a
% uniform variable which can take values <0.5 for the negative interval and
% >=0.5 for the positive interval.

% Based on the int_ind, we will now pick up a uniform sample from the
% selected interval

x = zeros(n,1); % Initializing x

% The intervals from which we pick up x
int_1 = [-10 -1];
int_2 = [10 1];

%Deciding the random positions of x, which contain the non zero values
x_pos = [];
while length(x_pos)<s0
    x_pos = unique(randi(n, [1 s0]));
end

% Filling up the non zero values from a uniform distribution
for j=1:s0

    % Choosing the interval
    int_ind = rand(1);

    if int_ind >=0.5
        int_curr = int_1;
    else
        int_curr = int_2;
    end

    % Drawing the uniform random value from the chosen interval
    curr_val = int_curr(1)*rand(1) + int_curr(2);

    % Filling up the non zero positions
    x(x_pos(j)) = curr_val;

end
figure(1);
subplot(2,1,1);
```

```
stem(x);
title('The sparse vector x');
b = A*x + sigma/sqrt(2) * randn(m,1);

beta = 0.01;
lamda = 1000;
nIter = 1000;
h_k = zeros(size(A, 2), nIter+1);
r_k = zeros(size(A, 2), nIter+1);
h_k(:, 1) = sparse(n, 1);

for i=1:nIter
   r_k(:,i) = h_k(:,i) - beta*A' * (b - A*h_k(:,i));
   for j=1:size(h_k, 1)
       h_k(j, i+1) = sign(r_k(j, i)) * max((abs(r_k(j, i) -
beta*lamda/2)),0);
   end
   if (i~=nIter)
       h_k(:, i+1) = h_k(:, i+1)/max(h_k(:, i+1));
   end
end

subplot(2,1,2);
stem(h_k(:, nIter+1));
title('x estimated');
```

FISTA

```
clear all;
close all;
clc;
m = 50;
n = 90;
sigma = 0.1;
A = randn(m,n);

s0 = 9; % No of non zero elements in x

% Range of x is from [-10 -1] U [1 10] and it is a uniform distribution
% So we will construct a variable int_ind (interval indicator) which is a
% uniform variable which can take values <0.5 for the negative interval and
% >=0.5 for the positive interval.

% Based on the int_ind, we will now pick up a uniform sample from the
% selected interval

x = zeros(n,1); % Initializing x

% The intervals from which we pick up x
int_1 = [-10 -1];
int_2 = [10 1];

%Deciding the random positions of x, which contain the non zero values
x_pos = [];
while length(x_pos)<s0
    x_pos = unique(randi(n, [1 s0]));
end

% Filling up the non zero values from a uniform distribution
for j=1:s0

    % Choosing the interval
    int_ind = rand(1);

    if int_ind >=0.5
        int_curr = int_1;
    else
        int_curr = int_2;
    end

    % Drawing the uniform random value from the chosen interval
    curr_val = int_curr(1)*rand(1) + int_curr(2);

    % Filling up the non zero positions
    x(x_pos(j)) = curr_val;

end
figure(1);
subplot(2,1,1);
```

```
stem(x);
title('The sparse vector x');
b = A*x + sigma/sqrt(2) * randn(m,1);

beta = 0.01;
lamda = 1000;
nIter = 1000;
h_k = zeros(size(A, 2), nIter+1);
r_k = zeros(size(A, 2), nIter+1);
h_k(:, 1) = sparse(n, 1);
tk = 1;
tp = 1;

for i=1:nIter
   r_k(:,i) = h_k(:,i) - beta*A' * (b - A*h_k(:,i));
   for j=1:size(h_k, 1)
       h_k(j, i+1) = sign(r_k(j, i)) * max((abs(r_k(j, i) -
beta*lamda/2)),0);
   end
   if (i~=nIter)
       h_k(:, i+1) = h_k(:, i+1)/max(h_k(:, i+1));
   end
end

subplot(2,1,2);
stem(h_k(:, nIter+1));
title('x estimated');
```