Dsc somp

function [indices,h]=DCSSOMP(Y,A,L)

% Y = input (one column vector for each channel / subcarrier)

% A = sensing matrix for each channel

% L = the sparsity level

%

% output:

% indices = index of L chosen columns of A

% h = the recovered coefficients

K=size(Y,2); % number of channels

N=size(Y,1); % observations per channel

M=size(A,2); % size of sparse vector (M>>N)

if (L<=0)

L=N;

end

% 1. initialization

R=Y; % residual

psi=zeros(N,L,K);

indices=zeros(1,L);

columns=zeros(N,L,K);

betamatrix=zeros(L,K);

for counter=1:L

% 2. find maximum correlation between residual and columns of A

cost=zeros(1,M);

for m=1:M

for k=1:K

cost(m)=cost(m)+abs(A(:,m,k)'\*R(:,k))/norm(A(:,m,k));

end

end

[maxval,maxi]=max(cost);

indices(counter)=maxi;

for k=1:K

% 3. orthogonalize

columns(:,counter,k)=A(:,maxi,k);

omega=A(:,maxi,k);

psi(:,counter,k)=omega;

for counter2=1:counter-1

psi(:,counter,k)=psi(:,counter,k)-(psi(:,counter2,k)'\*omega)\*psi(:,counter2,k)/(norm(psi(:,counter2,k)))^2;

end

% 4. update coefficients and residual

beta=psi(:,counter,k)'\*R(:,k)/(norm(psi(:,counter,k)))^2;

betamatrix(counter,k)=beta;

R(:,k)=R(:,k)-beta\*psi(:,counter,k);

end

end

% 6. Deorthogonalize

h=zeros(L,K);

for k=1:K

[Q,Rqr]=qr(columns(:,:,k),0);

h(:,k)=inv(Rqr)\*Q'\*psi(:,:,k)\*betamatrix(:,k);

end

function a = getResponse(N, angle)

for i=1:N

a(i) = exp(-j\*(i-1)\*pi\*angle/2)/sqrt(N);

end

a = a';

clear;

close all;

clc;

%count = zeros(1,9);

NMSE = zeros(1,9);

for w = 1:9

for iter = 1:1000

%% System Params

L = 2; % number of paths (including LOS)

Rs = 100e6; % total BW in Hz

N = 1; % number of subcarriers

Nt = 16; % number of TX antennas

Nr = Nt; % number of RX antennas

Nb = Nt\*2; % number of beams in dictionary

Ns = 5; % number of beams sent (randomly beams)

c = 3e8; % speed of light m/s

Ts = 1/Rs; % sampling period in us

posTx = [0 0]'; % TX is assumed to be in [0, 0]

posRx = [4 1]'; % RX (user) position

SP = [2, 2]; % scatter point position

alpha = 0.2; % user orientation

h = 10\*ones(1,L); % channel gain

GR = (sqrt(5) - 1)/2; % Golden number

% sigma = 0.1; % noise std

SNR\_db = -10:5:30;

W = length(SNR\_db);

%% Compute Channel Parameters for L paths

TOA = zeros(1, L); AOD = zeros(1, L); AOA = zeros(1, L);

TOA(1) = norm(posRx)/c; % LOS TOA

AOD(1) = atan2(posRx(2) - posTx(2), posRx(1) - posTx(1)); % LOS AOD

AOA(1) = atan2(posTx(2) - posRx(2), posTx(1) - posRx(1)) - alpha; % LOS AOA

for p = 1:L-1

TOA(p+1) = (norm(SP(p,:)) + norm(posRx - SP(p,:)'))/c; % NLOS TOA

AOD(p+1) = atan2(SP(p,2), SP(p,1)); % NLOS AOD

AOA(p+1) = atan2(SP(p,2) - posRx(2), SP(p,1) - posRx(1)) - alpha; % NLOS AOD

end

%% Create dictionary

Ut = zeros(Nt,Nb);

Ur = zeros(Nr,Nb);

aa = -Nb/2:Nb/2-1;

aa = 2\*aa/Nb;

for m = 1:Nb

Ut(:,m) = getResponse(Nt,aa(m))\*sqrt(Nt);

Ur(:,m) = getResponse(Nr,aa(m))\*sqrt(Nr);

end

%% Generate channel

H = zeros(Nr,Nt,N); A\_rx = zeros(Nr,L); A\_tx = zeros(Nt,L); Gamma = zeros(L, L, N);

Hb = zeros(Nb, Nb, N);

for n = 1:N

for p = 1:L

A\_rx(:,p) = getResponse(Nr,sin(AOA(p)))\*sqrt(Nr);

A\_tx(:,p) = getResponse(Nt,sin(AOD(p)))\*sqrt(Nt);

Gamma(p,p,n) = h(p)\*exp(-1j\*2\*pi\*TOA(p)\*(n-1)/(N\*Ts));

H(:,:,n) = H(:,:,n) + A\_rx(:,p)\*Gamma(p,p,n)\*A\_tx(:,p)';

end

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

Hb = zeros(Nb, Nb, N);

for n = 1:N

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

Hb\_mat = reshape(Hb, [Nb\*Nb N]);

Hb\_vec = reshape(Hb, [Nb\*Nb\*N 1]);

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

% figure;

% subplot(1, 2, 1);

Hb=Ur'\*H(:,:,1)\*Ut;

% mesh(abs(Hb));

% xlabel('AOD'); ylabel('AOA');

%% Generate the observation and beamformers

y=zeros(Nr,Ns,N);

F=zeros(Nt,Ns,N);

for g = 1:Ns

F(:,g) = exp(1j\*rand(Nt,1)\*2\*pi); % random beamformers (note: we don't add data symbols, they are part of F)

% F = F/ norm(F, 'fro'); % normalize power of F to 1 (fro norm)

end

for g = 1:Ns

signal(:,g) = H(:,:,1)\*F(:,g);

end

signal\_vec = signal(:);

for n = 1:W

PoS(n) = 1/(Nr\*Ns) \* sum(abs(signal\_vec).^2);

PoN(n) = PoS(n)/(10^(SNR\_db(n)/10));

sigma(n) = sqrt(PoN(n));

end

%

for g = 1:Ns

for n = 1:W

noise(:,g,n) = sigma(n)/sqrt(2)\*(randn(Nr,1) + 1i\*randn(Nr,1)); % noise

y(:,g,n) = signal(:,g) + noise(:,g,n);

end

end

%% Vectorize and generation of the basis

Omega=zeros(Nr\*Ns,Nb\*Nb);

yb=zeros(Nr\*Ns,W);

h = Hb(:); % Vecterize

for n=1:W

Omega = kron((Ut'\*F).', Ur);

% yb(:,n) = Omega(:,:,n)\*h; % \* (randn(Nr\*Ns, 1)+1j\*randn(Nr\*Ns, 1)); % y=Omega\*h

yb(:,n)= reshape(y(:,:,n),Nr\*Ns,1);

end

b = yb(:,1);

A = Omega(:,:,1);

%% DCS-SOMP

[indices,h\_hat]=DCSSOMP(yb(:,w),Omega,L); % the last input is the number of paths it recovers

support\_set = sort(abs(indices));

%for i=1:L

% support\_set(i) = find(abs(indices) == h(Nb\*Nb - L + i));

%end

%support\_set = sort(support\_set)

%if (support\_set(1) == 656 && support\_set(2) == 891)

%count(w) = count(w) + 1

%end

end

hb\_est = zeros(Nb\*Nb, 1);

%hb\_est(Omega\_k) = x\_k;

for i = 1:length(indices)

hb\_est(indices(i)) = Hb(indices(i))

end

Hb\_est = reshape(hb\_est, Nb, Nb);

%H\_b = (Hb./(Nt\*Nr));

NMSE(w) = norm(Hb\_est - Hb , 'fro')/norm(H\_b, 'fro')

end

%% Plot

% %figure(1)

% %subplot(121)

% mesh(asin(aa),asin(aa),abs(Hb(:,:,1)));

% title('Channel in Angular Domain');

% xlabel('AOD (rad)'); ylabel('AOA (rad)');

% subplot(122)

% mesh(asin(aa),asin(aa), abs(h\_hat));

% title('Estimated channel in Angular Domain by SOMP')

% xlabel('AOD (rad)'); ylabel('AOA (rad)'); zlabel('|Gain|');

% toc

Omp

clear; close all; clc;

count = 0;

for iter = 1:1000

%% System Params

L = 2; % number of paths (including LOS)

Rs = 100e6; % total BW in Hz

N = 1; % number of subcarriers

Nt = 16; % number of TX antennas

Nr = Nt; % number of RX antennas

Nb = Nt\*2; % number of beams in dictionary

Ns = 5; % number of beams sent (randomly beams)

c = 3e8; % speed of light m/s

Ts = 1/Rs; % sampling period in us

posTx = [0 0]'; % TX is assumed to be in [0, 0]

posRx = [4 1]'; % RX (user) position

SP = [2, 2]; % scatter point position

alpha = 0.2; % user orientation

h = 10\*ones(1,L); % channel gain

GR = (sqrt(5) - 1)/2; % Golden number

% sigma = 0.1; % noise std

SNR\_db = -10:5:20;

W = length(SNR\_db);

%% Compute Channel Parameters for L paths

TOA = zeros(1, L); AOD = zeros(1, L); AOA = zeros(1, L);

TOA(1) = norm(posRx)/c; % LOS TOA

AOD(1) = atan2(posRx(2) - posTx(2), posRx(1) - posTx(1)); % LOS AOD

AOA(1) = atan2(posTx(2) - posRx(2), posTx(1) - posRx(1)) - alpha; % LOS AOA

for p = 1:L-1

TOA(p+1) = (norm(SP(p,:)) + norm(posRx - SP(p,:)'))/c; % NLOS TOA

AOD(p+1) = atan2(SP(p,2), SP(p,1)); % NLOS AOD

AOA(p+1) = atan2(SP(p,2) - posRx(2), SP(p,1) - posRx(1)) - alpha; % NLOS AOD

end

%% Create dictionary

Ut = zeros(Nt,Nb);

Ur = zeros(Nr,Nb);

aa = -Nb/2:Nb/2-1;

aa = 2\*aa/Nb;

for m = 1:Nb

Ut(:,m) = getResponse(Nt,aa(m))\*sqrt(Nt);

Ur(:,m) = getResponse(Nr,aa(m))\*sqrt(Nr);

end

%% Generate channel

H = zeros(Nr,Nt,N); A\_rx = zeros(Nr,L); A\_tx = zeros(Nt,L); Gamma = zeros(L, L, N);

Hb = zeros(Nb, Nb, N);

for n = 1:N

for p = 1:L

A\_rx(:,p) = getResponse(Nr,sin(AOA(p)))\*sqrt(Nr);

A\_tx(:,p) = getResponse(Nt,sin(AOD(p)))\*sqrt(Nt);

Gamma(p,p,n) = h(p)\*exp(-1j\*2\*pi\*TOA(p)\*(n-1)/(N\*Ts));

H(:,:,n) = H(:,:,n) + A\_rx(:,p)\*Gamma(p,p,n)\*A\_tx(:,p)';

end

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

Hb = zeros(Nb, Nb, N);

for n = 1:N

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

Hb\_mat = reshape(Hb, [Nb\*Nb N]);

Hb\_vec = reshape(Hb, [Nb\*Nb\*N 1]);

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

% figure;

% subplot(1, 2, 1);

Hb=Ur'\*H(:,:,1)\*Ut;

% mesh(abs(Hb));

% xlabel('AOD'); ylabel('AOA');

%% Generate the observation and beamformers

y=zeros(Nr,Ns,N);

F=zeros(Nt,Ns,N);

for g = 1:Ns

F(:,g) = exp(1j\*rand(Nt,1)\*2\*pi); % random beamformers (note: we don't add data symbols, they are part of F)

% F = F/ norm(F, 'fro'); % normalize power of F to 1 (fro norm)

end

for g = 1:Ns

signal(:,g) = H(:,:,1)\*F(:,g);

end

signal\_vec = signal(:);

for n = 1:W

PoS(n) = 1/(Nr\*Ns) \* sum(abs(signal\_vec).^2);

PoN(n) = PoS(n)/(10^(SNR\_db(n)/10));

sigma(n) = sqrt(PoN(n));

end

%

for g = 1:Ns

for n = 1:W

noise(:,g,n) = sigma(n)/sqrt(2)\*(randn(Nr,1) + 1i\*randn(Nr,1)); % noise

y(:,g,n) = signal(:,g) + noise(:,g,n);

end

end

%% Vectorize and generation of the basis

Omega=zeros(Nr\*Ns,Nb\*Nb);

yb=zeros(Nr\*Ns,W);

h = Hb(:); % Vecterize

for n=1:W

Omega = kron((Ut'\*F).', Ur);

% yb(:,n) = Omega(:,:,n)\*h; % \* (randn(Nr\*Ns, 1)+1j\*randn(Nr\*Ns, 1)); % y=Omega\*h

yb(:,n)= reshape(y(:,:,n),Nr\*Ns,1);

end

b = yb(:,1);

A = Omega(:,:,1);

%% OMP

[Omega\_k, x\_k] = OMP(yb(:,n), A, L);

Omega\_k = sort(abs(Omega\_k))

if (Omega\_k(1) == 656 && Omega\_k(2) == 891)

count = count + 1;

end

end

hb\_est = zeros(Nb\*Nb, 1);

hb\_est(Omega\_k) = x\_k;

Hb\_est = reshape(hb\_est, Nb, Nb);

% subplot(1, 2, 2);

% mesh(abs(Hb\_est));

% xlabel('AOD'); ylabel('AOA');

StOMP

clear; close all; clc;

count = 0;

for iter = 1:1000

%% System Params

L = 2; % number of paths (including LOS)

Rs = 100e6; % total BW in Hz

N = 1; % number of subcarriers

Nt = 16; % number of TX antennas

Nr = Nt; % number of RX antennas

Nb = Nt\*2; % number of beams in dictionary

Ns = 5; % number of beams sent (randomly beams)

c = 3e8; % speed of light m/s

Ts = 1/Rs; % sampling period in us

posTx = [0 0]'; % TX is assumed to be in [0, 0]

posRx = [4 1]'; % RX (user) position

SP = [2, 2]; % scatter point position

alpha = 0.2; % user orientation

h = 10\*ones(1,L); % channel gain

GR = (sqrt(5) - 1)/2; % Golden number

% sigma = 0.1; % noise std

SNR\_db = -10:5:20;

W = length(SNR\_db);

%% Compute Channel Parameters for L paths

TOA = zeros(1, L); AOD = zeros(1, L); AOA = zeros(1, L);

TOA(1) = norm(posRx)/c; % LOS TOA

AOD(1) = atan2(posRx(2) - posTx(2), posRx(1) - posTx(1)); % LOS AOD

AOA(1) = atan2(posTx(2) - posRx(2), posTx(1) - posRx(1)) - alpha; % LOS AOA

for p = 1:L-1

TOA(p+1) = (norm(SP(p,:)) + norm(posRx - SP(p,:)'))/c; % NLOS TOA

AOD(p+1) = atan2(SP(p,2), SP(p,1)); % NLOS AOD

AOA(p+1) = atan2(SP(p,2) - posRx(2), SP(p,1) - posRx(1)) - alpha; % NLOS AOD

end

%% Create dictionary

Ut = zeros(Nt,Nb);

Ur = zeros(Nr,Nb);

aa = -Nb/2:Nb/2-1;

aa = 2\*aa/Nb;

for m = 1:Nb

Ut(:,m) = getResponse(Nt,aa(m))\*sqrt(Nt);

Ur(:,m) = getResponse(Nr,aa(m))\*sqrt(Nr);

end

%% Generate channel

H = zeros(Nr,Nt,N); A\_rx = zeros(Nr,L); A\_tx = zeros(Nt,L); Gamma = zeros(L, L, N);

Hb = zeros(Nb, Nb, N);

for n = 1:N

for p = 1:L

A\_rx(:,p) = getResponse(Nr,sin(AOA(p)))\*sqrt(Nr);

A\_tx(:,p) = getResponse(Nt,sin(AOD(p)))\*sqrt(Nt);

Gamma(p,p,n) = h(p)\*exp(-1j\*2\*pi\*TOA(p)\*(n-1)/(N\*Ts));

H(:,:,n) = H(:,:,n) + A\_rx(:,p)\*Gamma(p,p,n)\*A\_tx(:,p)';

end

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

Hb = zeros(Nb, Nb, N);

for n = 1:N

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

Hb\_mat = reshape(Hb, [Nb\*Nb N]);

Hb\_vec = reshape(Hb, [Nb\*Nb\*N 1]);

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

% figure;

% subplot(1, 2, 1);

Hb=Ur'\*H(:,:,1)\*Ut;

% mesh(abs(Hb));

% xlabel('AOD'); ylabel('AOA');

%% Generate the observation and beamformers

y=zeros(Nr,Ns,N);

F=zeros(Nt,Ns,N);

for g = 1:Ns

F(:,g) = exp(1j\*rand(Nt,1)\*2\*pi); % random beamformers (note: we don't add data symbols, they are part of F)

% F = F/ norm(F, 'fro'); % normalize power of F to 1 (fro norm)

end

for g = 1:Ns

signal(:,g) = H(:,:,1)\*F(:,g);

end

signal\_vec = signal(:);

for n = 1:W

PoS(n) = 1/(Nr\*Ns) \* sum(abs(signal\_vec).^2);

PoN(n) = PoS(n)/(10^(SNR\_db(n)/10));

sigma(n) = sqrt(PoN(n));

end

%

for g = 1:Ns

for n = 1:W

noise(:,g,n) = sigma(n)/sqrt(2)\*(randn(Nr,1) + 1i\*randn(Nr,1)); % noise

y(:,g,n) = signal(:,g) + noise(:,g,n);

end

end

%% Vectorize and generation of the basis

Omega=zeros(Nr\*Ns,Nb\*Nb);

yb=zeros(Nr\*Ns,W);

h = Hb(:); % Vecterize

for n=1:W

Omega = kron((Ut'\*F).', Ur);

% yb(:,n) = Omega(:,:,n)\*h; % \* (randn(Nr\*Ns, 1)+1j\*randn(Nr\*Ns, 1)); % y=Omega\*h

yb(:,n)= reshape(y(:,:,n),Nr\*Ns,1);

end

b = yb(:,1);

A = Omega(:,:,1);

% % StOMP

[sol, ~] = StOMP(Omega, yb(:,7), Nb\*Nb);

k = sort(abs(sol));

for i=1:L

support\_set(i) = find(abs(sol) == k(Nb\*Nb - L + i));

end

support\_set = sort(support\_set);

if (support\_set(1) == 656 && support\_set(2) == 891)

count = count + 1;

end

end

%hb\_est = zeros(Nb\*Nb, 1);

%hb\_est(Omega\_k) = x\_k;

%Hb\_est = reshape(hb\_est, Nb, Nb);

% subplot(1, 2, 2);

% mesh(abs(Hb\_est));

% xlabel('AOD'); ylabel('AOA');

SBL

clear; close all; clc;

%% System Params

L = 2; % number of paths (including LOS)

Rs = 100e6; % total BW in Hz

N = 10; % number of subcarriers

Nt = 16; % number of TX antennas

Nr = Nt; % number of RX antennas

Nb = Nt\*2; % number of beams in dictionary

Ns = 5; % number of beams sent (randomly beams)

c = 3e8; % speed of light m/s

Ts = 1/Rs; % sampling period in us

posTx = [0 0]'; % TX is assumed to be in [0, 0]

posRx = [4 1]'; % RX (user) position

SP = [2, 2]; % scatter point position

alpha = 0.2; % user orientation

h = 10\*ones(1,L); % channel gain

GR = (sqrt(5) - 1)/2; % Golden number

sigma = 0.1; % noise std

%% Compute Channel Parameters for L paths

TOA = zeros(1, L); AOD = zeros(1, L); AOA = zeros(1, L);

TOA(1) = norm(posRx)/c; % LOS TOA

AOD(1) = atan2(posRx(2) - posTx(2), posRx(1) - posTx(1)); % LOS AOD

AOA(1) = atan2(posTx(2) - posRx(2), posTx(1) - posRx(1)) - alpha; % LOS AOA

for p = 1:L-1

TOA(p+1) = (norm(SP(p,:)) + norm(posRx - SP(p,:)'))/c; % NLOS TOA

AOD(p+1) = atan2(SP(p,2), SP(p,1)); % NLOS AOD

AOA(p+1) = atan2(SP(p,2) - posRx(2), SP(p,1) - posRx(1)) - alpha; % NLOS AOD

end

%% Create dictionary

Ut = zeros(Nt,Nb);

Ur = zeros(Nr,Nb);

aa = -Nb/2:Nb/2-1;

aa = 2\*aa/Nb;

for m = 1:Nb

Ut(:,m) = getResponse(Nt,aa(m))\*sqrt(Nt);

Ur(:,m) = getResponse(Nr,aa(m))\*sqrt(Nr);

end

%% Generate channel

H = zeros(Nr,Nt,N); A\_rx = zeros(Nr,L); A\_tx = zeros(Nt,L); Gamma = zeros(L, L, N);

Hb = zeros(Nb, Nb, N);

for n = 1:N

for p = 1:L

A\_rx(:,p) = getResponse(Nr,sin(AOA(p)))\*sqrt(Nr);

A\_tx(:,p) = getResponse(Nt,sin(AOD(p)))\*sqrt(Nt);

Gamma(p,p,n) = h(p)\*exp(-1j\*2\*pi\*TOA(p)\*(n-1)/(N\*Ts));

H(:,:,n) = H(:,:,n) + A\_rx(:,p)\*Gamma(p,p,n)\*A\_tx(:,p)';

end

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

%% Visualize the beamspace channel for 1 subcarrier in AOA/AOD space

Hb = zeros(Nb, Nb, N);

for n = 1:N

Hb(:,:,n) = Ur'\*H(:,:,n)\*Ut;

end

Hb\_mat = reshape(Hb, [Nb\*Nb N]);

Hb\_vec = reshape(Hb, [Nb\*Nb\*N 1]);

%% Generate the observation and beamformers

y = zeros(Nr,Ns,N); signal = zeros(Nr,Ns,N); noise = zeros(Nr,Ns,N); F = zeros(Nt,Ns,N);

for g = 1:Ns

for n = 1:N

F(:,g,n) = exp(1j\*rand(Nt,1)\*2\*pi); % random beamformers (note: we don't add data symbols, they are part of F)

F(:,:,n) = F(:,:,n)/norm(F(:,:,n), 'fro'); % normalize power of F to 1 (fro norm)

end

end

for g = 1:Ns

for n = 1:N

signal(:,g,n) = H(:,:,n)\*F(:,g,n);

noise(:,g,n) = sigma/sqrt(2)\*(randn(Nr,1) + 1i\*randn(Nr,1)); % noise

y(:,g,n) = signal(:,g,n) + noise(:,g,n);

end

end

%% Vectorize and generation of the basis

yb = zeros(Nr\*Ns,N);

Omega = zeros(Nr\*Ns,Nb\*Nb,N);

for n = 1:N

yb(:,n) = reshape(y(:,:,n), Nr\*Ns,1);

nb(:,n) = reshape(noise(:,:,n), Nr\*Ns,1);

Omega(:,:,n) = kron((Ut'\*F(:,:,n)).',Ur);

end

y\_vec = reshape(yb, [Nr\*Ns\*N 1]);

noise\_vec = reshape(nb, [Nr\*Ns\*N 1]);

%test = norm(yb(:,3) - (1/Nb)^2\*Omega(:,:,3)\*Hb\_mat(:,3))

%% SBL

gm = ones(Nb\*Nb, 1);

GM = diag(gm);

Q = Omega(:,:,1);

v = nb(:,1)\*nb(:,1)';

gm\_trc = zeros(Nb\*Nb, 1);

p = 0;

p\_max = 100;

tic

while (p < p\_max)

p = p + 1;

GM = diag(gm);

Sigma = inv(Q'\*Q + inv(GM));

Mu = Sigma\* Q' \*yb(:,1);

for i = 1:Nb\*Nb

gm(i) = Sigma(i,i) + abs(Mu(i))^2;

end

end

h\_hat = reshape(Mu, Nb, Nb);

figure(2)

subplot(121)

mesh(asin(aa),asin(aa),abs(Hb(:,:,1)));

title('Noise-free Channel in Angular Domain');

xlabel('AOD (rad)'); ylabel('AOA (rad)');

subplot(122)

mesh(asin(aa),asin(aa), abs(h\_hat));

title('Estimated channel in Angular Domain by Pre-Defined Discrete Basis')

xlabel('AOD (rad)'); ylabel('AOA (rad)'); zlabel('|Gain|');

toc