A Random Matrix Theory Model for the Dominant Mode Rejection Beamformer Notch depth (Prof Kathleen Wage and John R. Buck)

- Sai Kasyap Kamaraju
G00845717

Introduction :

Passive sonar algorithms should detect low power signal in the presence of high power interferer. Thus, Adaptive Beam formers like DMR ABF are used. Adaptive Beam formers need to covariance matrices to compute array weights by maintaining unity gain in the look direction.

In reality, the true Ensemble covariance matrix is not available, so Sample Covariance Matrix is used to compute in place of ECM. The SCM is computed from number of snapshots, extending across an interval in the environment which is assumed to be statistically stationary.

For infinite set of snapshots SCM converges to ECM. The higher the snapshots to average , implies that SCM converges closer to ECM.

Random Matrix Theory provides new set of results on the convergence of SCM to the ECM. recent work on RMT contains valuable insights about the behavior of the eigen values andEigen vectors of large random matrices.RMT differs fundamentally from the classic methods used to analyze the asymptotic performance of ABFs. Classic asymptotic performance of ABF results assume that the array size N remains fixed, but the number of snapshots L→ infinite. RMT results instead assume that both N,L→ infinite but the ratio N/L = c (constant aspect ratio) is fixed. This paper incorporates RMT results on the fidelity of the SCM eigenvectors into the Dominant Mode Rejection ABF to obtain a model that predicts the mean notch depth in the direction of a single loud interferer.

Summary :

The models exploits RMT results on the fidelity of the SCM eigenvectors to characterize the notch depth as a function of the array sizeN, the number of snapshots L, the INRσ^2 and the interferer location relative to the look direction as represented by cos^2(vm,v1).

The first model characterizes ND as a function of the number of snapshots, treating the INR as parameter.The equations 8 – 10 are the derivation preliminaries where vm and v1 are projected onto unit vectors in the directions of the ensemble interferer eigenvector and an orthogonal residual.
The equations 11-12 are the RMT results on Eigen vectors above to get ND results.

The equations 13-17 result can interpreted in a manner similar to a Bode plot which can be approximated to piecewise linear approximation of Notch depth vs snapshots on a log-log plot.
The equations 18 -20 are breakpoints for figure 1. Below L1 snapshots, the DMR ABF is no better than the CBF at attenuating a loud interferer. The model also provides a clear prediction of the number of snapshots (L3) required to achieve the ensemble notch depth (NDens) .

The second model treats INR as the independent variable and the ratio of sensors/snapshots as the parameter. The equation 21,22 &7 determine the linear piecewise model for ND vs log INR.he figures 2 and 3 are cases for c1 <= and c1>>1.

Figure 4 are the simulation results where compares the RMT model's prediction for ND as a function of snapshots L with the average notch depth observed in simulations.
For all five INRs, the model accurately predicts the mean notch depth seen in the simulations, and also predicts the number of snapshots required to obtain the ensemble performance.
For INR = 0 interferer signal also, the DMR ABF requires at least 1000 snapshots to approach the notch depth predicted from ensemble statistics, and for the louder interferers even a million snapshots is not sufficient to achieve the ensemble notch depth. For most of the observed region, the notch grows 10 dB deeper for each decade of snapshots. All of the scenarios have L2 <3snapshots, so are in the -10 dB/decade region by the first data point at L=3 snapshots.

Figure 5 are the simulation results where the RMT predictions are compared with monte carlo simulations for ND as function of INR for different values of c(L). An important result from the ND vs INR model is that the ABF does not benefit from INR increases above the breakpointINR2. Above this breakpoint, the notch grows deeper at exactly the same rate as the interferer grows louder, so there is no net change in the interferer energy appearing in the ABF output. The critical INR2 determining the limit of DMR's ability to attenuate very loud interferers in the ABF output is set by the snapshot to sensor ratio c and interferer location via tan2(vm,v1).

Explanation how algorithm works:

The algorithm implemented is Dominant mode Rejection ABF which was introduced by [Abraham & Owsley, Proc. Oceans, 1990]. In this beamformer, large Eigen values are assumed as interferers and small Eigen values are averaged.

For the Ensemble Case :
1. First the form the covariance matrix

$$\mathbf{\Sigma} = \sigma_1^2 \mathbf{v}_1 \mathbf{v}_1^{\mathrm{H}} + \sigma_w^2 \mathbf{I}.$$

2. Then form the Ensemble Covariance Matrix ECM

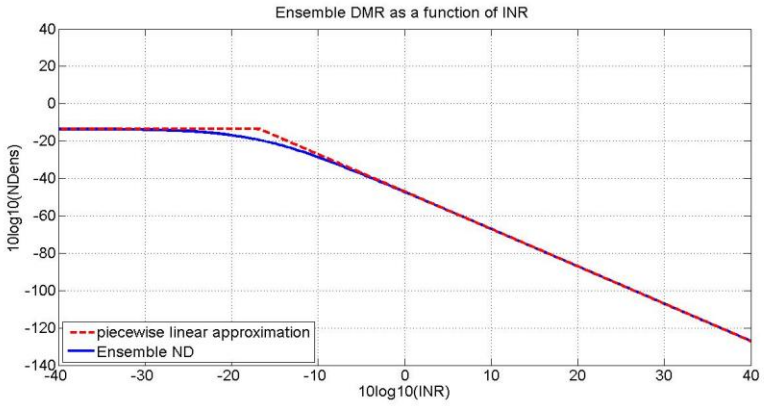
$$\mathbf{\Sigma}_{\mathrm{DMR}} = \sum_{n=1}^{D} \gamma_n \boldsymbol{\xi}_n \boldsymbol{\xi}_n^{\mathrm{H}} + \sum_{n=D+1}^{N} \sigma_w^2 \boldsymbol{\xi}_n \boldsymbol{\xi}_n^{\mathrm{H}}.$$

3. The weight vector is computed in the following way :

$$\mathbf{w}_{\mathrm{DMR}} = \left(\mathbf{v}_m^{\mathrm{H}} \mathbf{\Sigma}_{\mathrm{DMR}}^{-1} \mathbf{v}_m \right)^{-1} \mathbf{\Sigma}_{\mathrm{DMR}}^{-1} \mathbf{v}_m$$

4. The Notch depth is defined as ND = |wHvi|2, vi replica vector of the interferer.

$$\mathrm{ND}_{\mathrm{ens}} = \left| \mathbf{w}_{\mathrm{DMR}}^{\mathrm{H}} \mathbf{v}_1 \right|^2 = \frac{\cos^2(\boldsymbol{\xi}_1, \mathbf{v}_m)}{\left(1 + N \frac{\sigma_1^2}{\sigma_w^2} \sin^2(\boldsymbol{\xi}_1, \mathbf{v}_m) \right)^2} \quad (19)$$



But in practice, we won't have the knowledge of ensemble,hence we need to consider the DMR for snapshots case .
For the snapshot case, The Notch depth is computed using Sample covariance matrix in a similar way above.

$$\mathbf{S} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{p}_l\mathbf{p}_l^{\mathrm{H}} = \sum_{n=1}^{N} g_n\mathbf{e}_n\mathbf{e}_n^{\mathrm{H}} = \mathbf{E}\mathbf{G}\mathbf{E}^{\mathrm{H}} \qquad \mathbf{S}_{\mathrm{DMR}} = \sum_{n=1}^{D} g_n\mathbf{e}_n\mathbf{e}_n^{\mathrm{H}} + \sum_{n=D+1}^{N} s_w^2\mathbf{e}_n\mathbf{e}_n^{\mathrm{H}}$$

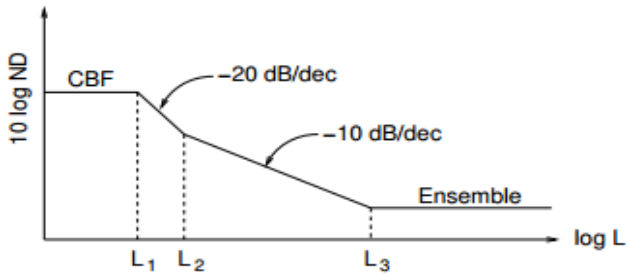
The weight vector in case of snapshot case is computed as follows:

$$\mathbf{w}_{\mathrm{DMR}} = \frac{\mathbf{v}_m - \sum_{i=1}^{D}\left(\frac{g_i - s_w^2}{g_i}\right)\mathbf{e}_i\mathbf{e}_i^{\mathrm{H}}\mathbf{v}_m}{\mathbf{v}_m^{\mathrm{H}}\mathbf{v}_m\left(1 - \sum_{i=1}^{D}\left(\frac{g_i - s_w^2}{g_i}\right)\cos^2(\mathbf{e}_i, \mathbf{v}_m)\right)}.$$

Random Matrix Theory Model for mean Notch depth for DMR:

3.1 Notch Depth vs. Snapshots

The equations 13-17 are used for creating a linear piecewise model for ND vs Snapshots.



The Break points are defined as follows:

$$L_1 = 1/\left(\sigma_1^2\sin^2(\mathbf{v}_m, \mathbf{v}_1)\right)$$
$$L_2 = N\cot^2(\mathbf{v}_m, \mathbf{v}_1)/\sigma_1^2,$$
$$L_3 = N\sigma_1^2\tan^2(\mathbf{v}_m, \mathbf{v}_1).$$

The values of break points can also be defined in c (constant aspect ratio) c = N/L

c1 = (cotq/INR);
c2 = (INR*tanq);
c3 = (1+N*INR*sinq); (all trigonometric functions are squared functions).

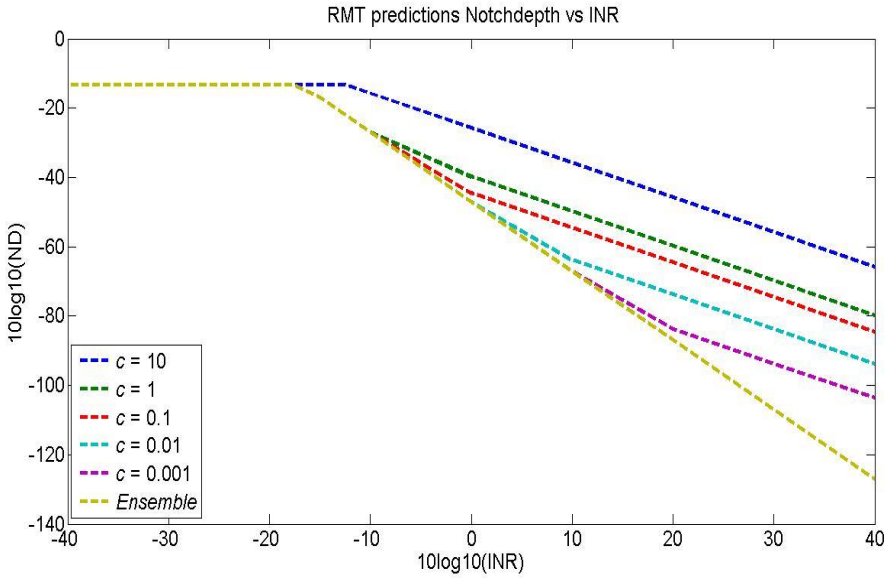
Now we need construct linear piecewise model in Matlab.

1. If the value of c > c1 ,Notchdepth = NDens(m);
2. For asymptote value, c = c1 ,Notchdepth = NDens(m,q) +10*log10(sqrt(2));
3. If the value of c lies in between c1and c2 (-10 dB region)
   ND = NDens(m)+10*log10(c(q))-10*log10(c1(m));
4. if (c(q)>c2(m)&&c(q)<c3(m)) (-20 dB region)
   ND = NDens(m)+20*log10(c(q))-20*log10(c2(m))-10*log(c1(m));
5. if (c(q) > c3(m)) (before poleP
   ND= NDcbf;

NDens(m) → Ensemble ND

The value of c at breakpoints give a + or -3 dB gain depending on whether it is zero or pole.

RMT predictions for ND vs Snapshots



3.2 Notchdepth vs INR

The equation 21,22 and 7 determine the linear piecewise model

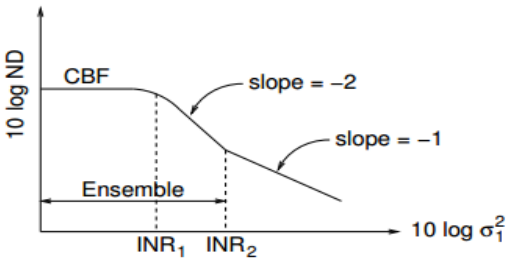


Fig. 2. Model for beampattern notch depth in dB vs. INR in dB for a given array size N and number of snapshots L for the snapshot sufficient and rich cases, $c \leq 1$.

For c <=1 ,
The breakpoints are :
inr1 = 1/(N*sinq);
inr2(q) = ((1+c(q))).^2/(c(q)*tanq);

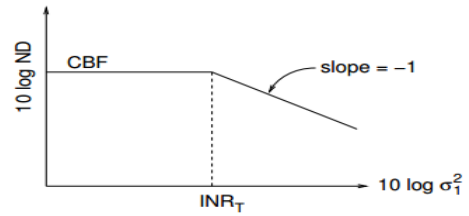


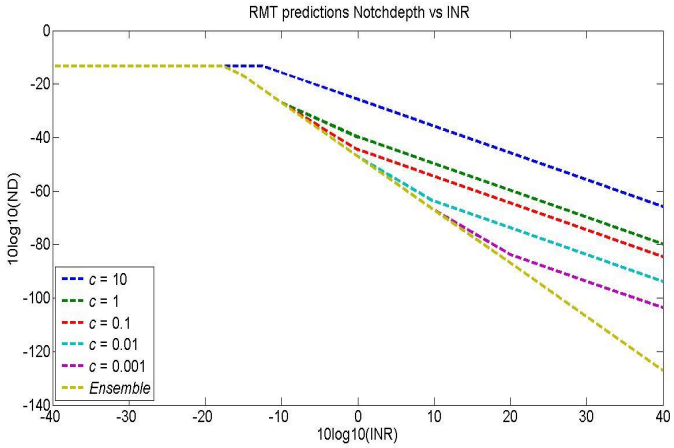
Fig. 3. Model for beampattern notch depth in dB vs. INR in dB for a given array size N and number of snapshots L for the severely snapshot deficient case $c \gg 1$.

For c>1

INRt = sqrt(c(q))/N; (Phase TRansistion)

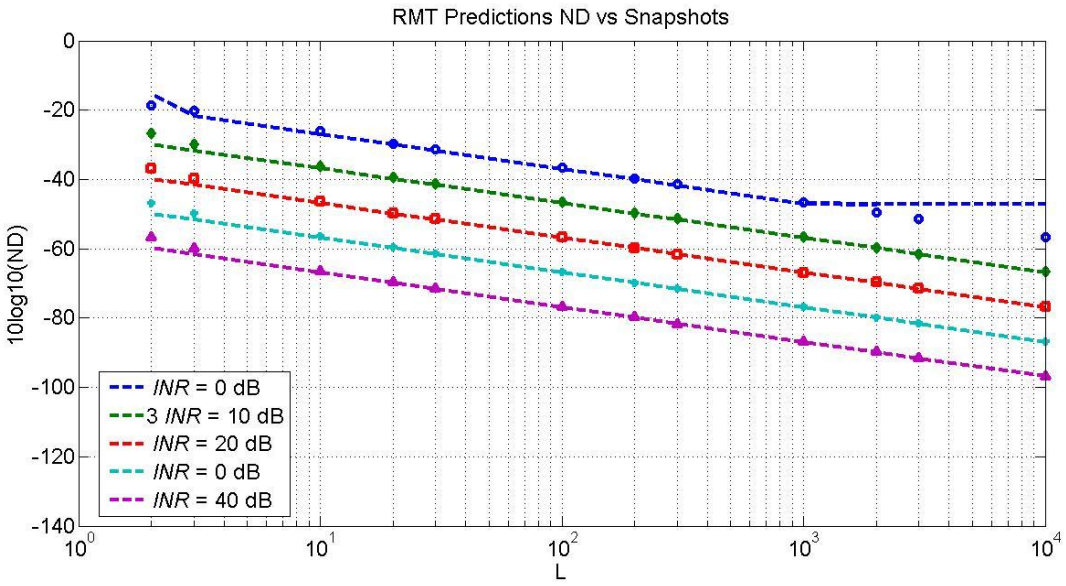
For c = 10 , this is model applies.

RMT Predictions :



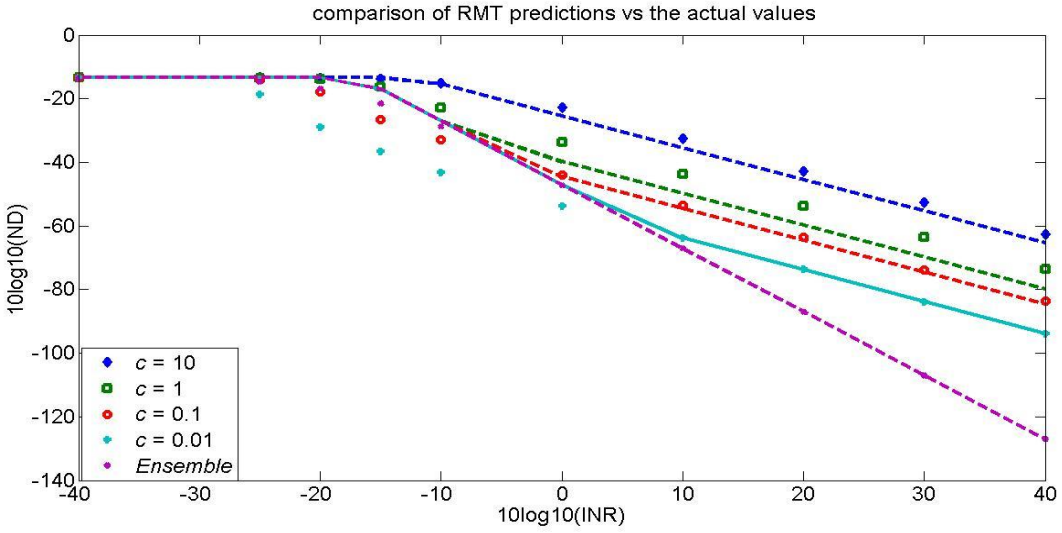
Comparison of Monte carlo simulations and RMT predictions

For ND vs Snapshots



The snapshots are considered till 10^4 only. (Due to time taking simulation for 1e6)

For ND vs INR



Observation :

The ND vs INR for L = 5000 snapshots seems to be wrong for lower INRs ranging from -20 dB to -10dB.

Critique of the paper:

How do the RMT predictions perform?

The RMT predictions are accurate and correct approximations for the simulations of DMR ABF.

Does it do what the authors claim? Why or why not?

The RMT predictions work perfectly. The RMT model predicts correctly the mean ND for DMR ABF.

So it does whatever the author claims.

Could the approach be improved? If so, how?

The approach can be improved by extending it to multiple interferer case and also it can be applied to various other beamformers like MVDR etc.

Lessons Learnt from the paper:

1. Random Matrix Theory and its applications for predicting the mean notch of DMR ABF.
2. Creating Linear piecewise linear interpolation models in MATLAB
3. Better understanding of Dominant Mode Rejection ABF, Notch depth and RMT.
4. There is slightly wrong in going in my MATLAB code while plotting comparisons of RMT and Monte carlo simulations of ND vs INR plot.

APPENDIX

MATLAB CODE

```
% Comparision of RMT predictions and Monte Carlo Simulations
% Figure 4 IN RMT Model paper
clc;
clear all;
d = 0.5;
N = 50 ; % Number of sensors
D = [0:1:N-1].';
vm = exp(j*2*pi*d*D*0); % Broad Side
% replica vector
% Interferer location
ui = 0.06;
% intialising SCM as zero intially
vi = exp(-j*2*pi*d*D*ui);
nt = 3000;
% Number of snapshots
INRrange = [-40 -25 -20  -15  -10 0 10 20 30 40];
INR = 10.^(INRrange/10);
L = [5 50 500 5000];
ui = 0.06;
Vi = exp(-j*2*pi*d*D*ui);
cosq =gencos(vm,vi);
sinq = (1-cosq);

tanq = sinq/cosq;
cotq = cosq/sinq;


for q = 1: length(L)

c(q) =N/L(q);
end

%CBF
NDcbf = 10*log10(cosq);
%Ensmeble Notchdepth

for t = 1:length(INR)
if INRrange(t) < 10*log10(1/(N*sinq))

    Ndl(t) = 10*log10(abs(cosq));
  %constant offset
end
% At Break point
if INRrange(t) == 10*log10(1/(N*sinq))

Ndl(t) = 10*log10(cosq)+10*log10(sqrt(2));
end

    if INRrange(t) >= 10*log10(1/(N*sinq))

    Ndl(t) = 10*log10(cosq) + -20*log10((sinq*N*10.^(INRrange(t)/10)));
```

```
    end
end

NDens = Ndl;


for m=1:length(INR)
    disp(['loop ' int2str(m) ' of 5 ...'])
    for q=1:length(L)

         if (c(q)>=1/sinq^2)
           inrphase = sqrt(c(q))/N;

    ND1(INR<=inrphase) = NDcbf;
    ND1(INR>inrphase) = NDcbf-10*log10(INR(INR>inrphase))+10*log10(inrphase);

     else
           inr1 = 1/(N*sinq);
            inr2(q) = ((1+c(q))).^2/(c(q)*tanq);


       if (INR(m)<inr1)
           ND(m,q) = NDcbf;
       end
       if (INR(m)==inr1)
           ND(m,q) = NDcbf-10*log(sqrt(2));
       end
       if(INR(m)>inr1&&INR(m)<inr2(q))
           r20(m,q)= NDcbf-20*log10(INR(m))+20*log10(inr1);
           ND(m,q) = NDcbf-20*log10(INR(m))+20*log10(inr1);
       end
       if(INR(m)==inr2(q))
           ND(m,q) =r20(m,q)+10*log(sqrt(2));
       end
       if(INR(m)>inr2(q))
           ND(m,q) = NDcbf-10*log10(INR(m))-10*log10(inr2(q))+20*log10(inr1);
       end
       end
    end

end
Sc = zeros(N,N);
S2= zeros(50,50);
%nt = 3000;
% Number of snapshots


% number of monte carlo trails
for m=1:length(INR)
    disp(['loop ' int2str(m) ' of 5 ...'])
    for q=1:length(L)
        for k1 = 1:nt
```

```
b = sqrt(INR(m)/2)*(randn(1,L(q))+j*randn(1,L(q))); % complex circular
gaussian RV
n = sqrt(1/2)*(randn(N,L(q))+j*randn(N,L(q)));
p = vi*b+n;
S  = p*p';
SCM = S/L(q); % Structured Covariance matrix

% Finding eigen values and eigen vectors

[Sevec1,Seval]=eig(SCM);

% Sorting them in descending order
[Seval,ind] = sort(diag((Seval)),'descend');   % sort eigenvalues in
descending order
  Sevec = Sevec1(:,ind);          % arrange eigenvectors in same order

% finding the estimated noise power..
% Dl - Number of planewaves assumed to be 1.
dl =1;
sn =  (L(q)/L(q)-1)*(1/(N-dl))*(sum(Seval(2:N)));
% Finding s- dmr
S1 = Seval(1,:)*(Sevec(:,1)*Sevec(:,1)');

for i = dl+1:N
S2 = S2+(sn*(Sevec(:,i)*Sevec(:,i)'));
end
Sdmr = S1+S2;

mcosq = gencos(Sevec(:,1),vm);
% % Need to find cosq b/w SCM of ei,vm

gw = (Seval(1,:)-sn)/Seval(1,:);

Wdnum = vm-(gw*Sevec(:,1)*Sevec(:,1)'*vm);
Wdden = vm'*vm*(1-gw*mcosq);
Wdmr = Wdnum/Wdden;
 Ne(k1) = (abs(Wdmr'*vi).^2);


end
 Nd(m,q) = mean(Ne);
end
end

Notchdepth = 10*log10(Nd);
%ENsemble case
k1 =1;
for INR = 10.^(INRrange/10)


    S5 = INR*(vi*vi') + eye(N);
dl =1 ; % Number of Strong planewavesignals (interferers);
% Finding Eigen vectors and eigen values
[evec1,egval] = eig(S5);
```

```
% Sorting them in descending order
[egval,ind] = sort(diag((egval)),'descend'); % sort eigenvalues in descending
order
evec = evec1(:,ind); % arrange eigenvectors in same order
% Finding Sdmr
S1 = egval(1,:)*(evec(:,1)*evec(:,1)');
sn = (1/(N-dl))*(sum(egval(2:N)));
S2= zeros(50,50);
for i = dl+1:N
S2 = S2+(sn*(evec(:,i)*evec(:,i)'));
end
Sdmr = S1+S2;
% Weight vector of dmr
Wd1 = inv(vm'*(inv(Sdmr))*vm)*((inv(Sdmr))*vm);
ND(k1) =  abs(Wd1'*vi).^2;
NdENS(k1) = 10*log10(ND(k1));
k1 = k1+1;
end
figure
plot(INRrange,Notchdepth(:,1),'d',INRrange,Notchdepth(:,2),'s',INRrange,Notch
depth(:,3),'o',INRrange,Notchdepth(:,4),'*',INRrange,NdENS,'x');
hold on
plot(INRrange,ND1,'--',INRrange,ND(:,2),'--',INRrange,ND(:,3),'--
',INRrange,ND(:,4),INRrange,NDens,'--');
ylim([-140 0]);
xlim([-40 40]);
 legend('{\it c} = 10','{\it c} = 1','{\it c} = 0.1','{\it c} = 0.01','{\it
Ensemble}')
 title(' comparison of RMT predictions vs the actual values')
xlabel('10log10(INR)')
  ylabel('10log10(ND)')
```

```
% Henr y cox paper
function cossq=gencos(a,b,C)
if nargin<3
 cossq=abs(a'*b).^2/((a'*a)*(b'*b));
else
 cossq=abs(a'*C*b).^2/((a'*C*a)*(b'*C*b));
end
%--------------------------------------------------------------------
return
```

```
% Random Matrix Theory Model for DMR Notchdepth
%%%By Sai Kasyap%%%%%
% p--> Nx1 column vector
% D --> Number of narrowband planewave Signals(Interferers)
%N ---> Number of Elements in ULA
%Vm--> Replica Vector in the look direction
% Cosine is found as mentioned in the cox paper
% FIgure 5 in RMT model
clc;
clear all;
%   Detailed explanation goes here
d = 0.5;
N = 50 ; % Number of sensors
D = [0:1:N-1].';
vm = exp(j*2*pi*d*D*0); % Broad Side
% replica vector
u=[-1:0.001:1];
V = exp(j*2*pi*d*D*u);
% Interferer location
ui = 0.06;
% intialising SCM as zero intially
Sc = zeros(N,N);
S2= zeros(50,50);
%nt = 3000;
% Number of snapshots
L=[2 3 10 20 30 100 200 300 1000 2000 3000 10000];

INRrange = [0 10 20 30 40];
Nd = zeros(length(INRrange),length(L));
WNG = zeros(length(INRrange),length(L));
INR = 10.^(INRrange/10);

ui = 0.06;
vi = exp(-j*2*pi*d*D*ui);

% number of monte carlo trails
nt = 3000;

% number of monte carlo trails
for m=1:length(INR)
    disp(['loop ' int2str(m) ' of 5 ...'])
    for q=1:length(L)
        for k1 = 1:nt

b = sqrt(INR(m)/2)*(randn(1,L(q))+j*randn(1,L(q))); % complex circular
gaussian RV
n = sqrt(1/2)*(randn(N,L(q))+j*randn(N,L(q)));
p = vi*b+n;
S  = p*p';
SCM = S/L(q); % Structured Covariance matrix

% Finding eigen values and eigen vectors

[Sevec1,Seval]=eig(SCM);
```

```
% Sorting them in descending order
[Seval,ind] = sort(diag((Seval)),'descend');   % sort eigenvalues in
descending order
  Sevec = Sevec1(:,ind);          % arrange eigenvectors in same order

% finding the estimated noise power..
% Dl - Number of planewaves assumed to be 1.
dl =1;
sn =  (L(q)/L(q)-1)*(1/(N-dl))*(sum(Seval(2:N)));
% Finding s- dmr
S1 = Seval(1,:)*(Sevec(:,1)*Sevec(:,1)');

for i = dl+1:N
S2 = S2+(sn*(Sevec(:,i)*Sevec(:,i)'));
end
Sdmr = S1+S2;

mcosq = gencos(Sevec(:,1),vm);
% % Need to find cosq b/w SCM of ei,vm

gw = (Seval(1,:)-sn)/Seval(1,:);

Wdnum = vm-(gw*Sevec(:,1)*Sevec(:,1)'*vm);
Wdden = vm'*vm*(1-gw*mcosq);
Wdmr = Wdnum/Wdden;
 Ne(k1) = (abs(Wdmr'*vi).^2);


end
 Nd(m,q) = mean(Ne);
end
end


Notchdepth = 10*log10(Nd);


cosq =gencos(vm,vi);
sinq = (1-cosq);

tanq = sinq/cosq;
cotq = cosq/sinq;


for q = 1: length(L)

c(q) =N/L(q);
end

%CBF
NDcbf = 10*log10(cosq);
%Ensmeble Notchdepth
NDens = NDcbf-20*log10(N*INR*sinq+1);
for m=1:length(INR)
```

```
    disp(['loop ' int2str(m) ' of 5 ...'])
    for q=1:length(L)

%Defining Trignometric functions
% Defining Break points
     c1(m) =(cotq/INR(m));
     c2(m) = (INR(m)*tanq);
    c3(m) =(1+N*INR(m)*sinq);

    if ((N*INR)<0.1)
    warning('for weak interferer ');
    ND = NDcbf*ones(size(c));
end
    if (INR<1)
    warning('RMT model requires INR>>1');
end
if (N*INR*sinq<1)
    warning('RMT model requires N*INR*sin2>>1');
end
if ((N*INR)^2< c(q))
    warning('RMT model requires (N*INR)^2>c');
end
if (c(q) < c1(m))
    ND(m,q) = NDens(m);
end
   if (c(q) == c1(m))
     ND(m,q) = NDens(m,q) +10*log10(sqrt(2));
   end
 if ((c1(m))<c(q))&&(c(q)< c2(m))
     O(q) = 10*log10(c(q));
   r2(m,q) =  NDens(m)+10*log10(c(q))-10*log10(c1(m));
   ND(m,q) = r2(m,q);
 end
 if (c(q) ==c2(m))
     ND(m,q) = r2(m,q)+10*log10(sqrt(2));
 end
 if (c(q)>c2(m)&&c(q)<c3(m))
   r3(m,q) = NDens(m)+20*log10(c(q))-20*log10(c2(m))-10*log(c1(m));
 ND(m,q) = r3(m,q);
%r3(m,q) = r2(m,q)+20*log10(c(q))-20*log10(c2(m));
%ND(m,q) = r2(m,q)+20*log10(c(q))-20*log10(c2(m));
 end

   if (c(q)==c3(m))
      ND(m,q) = r3(m,q)-10*log10(sqrt(2));
   end
  if (c(q) > c3(m))
    ND(m,q)= NDcbf;
  end
    end
end

figure
semilogx(L,ND(1,:),'--',L,ND(2,:),'--',L,ND(3,:),'--',L,ND(4,:),'--
',L,ND(5,:),'--');
hold on
```

```
semilogx(L,Notchdepth(1,:),'o',L,Notchdepth(2,:),'d',L,Notchdepth(3,:),'s',L,
Notchdepth(4,:),'*',L,Notchdepth(5,:),'^');
title('RMT Predictions ND vs Snapshots')
grid
xlabel('L')
ylabel('10log10(ND)')
ylim([-140 0])
legend('{\it INR} = 0 dB','3{\it INR} = 10 dB','{\it INR} = 20 dB','{\it INR}
= 0 dB','{\it INR} = 40 dB')
```