REFERENCES

[1] Carl B. Dietrich, Jr., "Adaptive Array and Diversity Antenna Configuration for Handheld Wireless Communication Terminals", Electrical Engineering, Virginia Tech, Blackburg, VA, pp. 1-2, May 2000.

[2] Raj Pandya, "Mobile and Personal Communication Services and Systems", Prentice-Hall of India, New Delhi, pp 7-8, 2000

[3] Raj Pandya, "Mobile and Personal Communication Services and Systems", Prentice-Hall of India, New Delhi, pp 16-17, 2000

[4] Fred Halsall, "Data Communications, Computer Networks and Open Systems", Addison-Weslet Publishing Company, pp 336-340, 1996.

[5] A.Bruce Carlson, "Communication Systems", McGraw-Hill Book Company, Singapore, pp 4-5, 1986.

[6] Carl B. Dietrich, Jr., "Adaptive Array and Diversity Antenna Configuration for Handheld Wireless Communication Terminals", Electrical Engineering, Virginia Tech, Blackburg, VA, pp. 19-20, May 2000.

[7] J. C. Haartsen, "Mobile Radio Communications", Telecommunication Engineering Session 6, University of Twente, pp. 30-32, 2002

[8] Raj Pandya, "Mobile and Personal Communication Services and Systems", Prentice-Hall of India, New Delhi, pp 16-17, 2000

[9] Simon C. Swales, Mark A. Beach, David J. Edwards and Joseph P. McGeehan, "The Performance Enhancement of Multibeam Adaptive Base-Station Antennas for Cellular Land Mobile Radio Systems", IEEE Transactions on Vehicular Technology, pp 4-5,Vol.39, No.1, February 1990.

[10] Rajeswari Chatterjee, "Antenna Theory and Practice", Second Edition, New Age International Publishers Limited, New Delhi.pp 86-87, 1996

[11] John D. Kraus, "Antennas", Tata McGraw-Hill Publishing Company Limited, New Delhi. pp 128, Second Edition, 2000.

[12] Tapan K. Sarkar and Jinwan Koh, "A Pragmatic Approach to Adaptive Antennas", IEEE Antenna and Propagation Magazine, Vol.42, No.2, April 2000.

[13] Seungwon Choi, Donghee Shim and Tapan K. Sarkar, "A Comparison of Tracking-Beam Arraya and Switching-Beam Arrays Operating in a CDMA Mobile Communication Channel", IEEE Antenna and Propagation Magazine, Vol.42, No.2, April 2000.

[14] MATLAB version 5.2 Neural Network Tool box guide.

APPENDIX A

MATLAB CODE TO OBTAIN THE TRAINING DATA SET

```
%Finding Maximums when delta changes from -pi~pi in steps of 10 degrees
%=====================================================================
dr=pi;

%Defining 10 degrees steps
phi = -(150/180)*pi:(10/180)*pi:(150/180)*pi;

%Maximums in an arbitrary direction is given by
% 0 = dr*sin(theta) + phi
Max_theta = asin(-1*(phi/dr));

%To find Maximums in the -60 -> +60 sector
MaxPoints = (Max_theta/pi)*180 ;
%N = Number of elements in the array
N=10;

%Loop to find Nulls corresponding to maximums
for i=1:1:37
   for k=1:1:20
         phi_ = (i-1)*(10/180)*pi;
         K= (k-10);
         Null(i,k)=((asin((2*K*pi/N - phi_)/dr))/pi )*180;
   end
end

%Loop to find Nulls fall in the -60 -> +60 sector
for i=1:31
   index =1;
   for k=1:20

         if real(Null(i,k))<60 & real(Null(i,k))>-60 & real(Null(i,k))~=
MaxPoints(i)
            Null_120(i,index) = Null(i,k);
            index = index+1;
         end

   end

end

disp('Training Data Set');
disp('=================');
disp('Maximums in the third quadrant =');disp(MaxPoints);
disp('Nulls Corresponding to above maximums(9 nulls per maximum)
=');disp(Null_120);
%End
```

-33.7490 -37.6699 -41.8103 -46.2383 -51.0576 -56.4427 56.4427
-36.0782 -40.1240 -30.0000 -33.7490 -37.6699 -27.8181 -23.5782

Columns 64 through 70

51.0576 46.2383 41.8103 37.6699 33.7490 30.0000 26.3878
-27.1007 -18.1262 -21.5102 -24.9750 -16.1276 -19.4712 -22.8854

Columns 71 through 77

22.8854 19.4712 16.1276 12.8396 9.5941 6.3794 3.1847
-26.3878 -17.4576 -20.8275 -24.2747 -15.4660 -18.7974 -22.1961

Columns 78 through 84

0 -3.1847 -6.3794 -9.5941 -12.8396 -16.1276 -19.4712
-25.6793 -16.7914 -20.1479 -23.5782 -27.1007 -18.1262 -21.5102

Columns 85 through 91

-22.8854 -26.3878 -30.0000 -33.7490 -37.6699 -41.8103 -46.2383
-24.9750 -16.1276 -19.4712 -22.8854 -26.3878 -17.4576 -20.8275

Columns 92 through 98

-51.0576 -56.4427 56.4427 51.0576 46.2383 41.8103 37.6699
-24.2747 -15.4660 -11.5370 -14.8065 -6.3794 -9.5941 -12.8396

Columns 99 through 105

33.7490 30.0000 26.3878 22.8854 19.4712 16.1276 12.8396
-4.4608 -7.6623 -10.8879 -14.1490 -5.7392 -8.9490 -12.1875

Columns 106 through 112

9.5941 6.3794 3.1847 0 -3.1847 -6.3794 -9.5941
-3.8226 -7.0204 -10.2403 -13.4934 -5.0997 -8.3051 -11.5370

Columns 113 through 119

-12.8396 -16.1276 -19.4712 -22.8854 -26.3878 -30.0000 -33.7490
-14.8065 -6.3794 -9.5941 -12.8396 -4.4608 -7.6623 -10.8879

Columns 120 through 126

-37.6699 -41.8103 -46.2383 -51.0576 -56.4427 56.4427 51.0576
-14.1490 -5.7392 -8.9490 -12.1875 -3.8226 0 -3.1847

Columns 127 through 133

16.1276 12.8396 9.5941 6.3794 3.1847 0 -3.1847
26.3878 22.8854 32.2310 28.5404 24.9750 21.5102 30.7379

Columns 204 through 210

-6.3794 -9.5941 -12.8396 -16.1276 -19.4712 -22.8854 -26.3878
27.1007 23.5782 20.1479 29.2676 25.6793 22.1961 31.4814

Columns 211 through 217

-30.0000 -33.7490 -37.6699 -41.8103 -46.2383 -51.0576 -56.4427
27.8181 24.2747 20.8275 30.0000 26.3878 22.8854 0

Columns 218 through 224

56.4427 51.0576 46.2383 41.8103 37.6699 33.7490 30.0000
36.8699 32.9867 43.5422 39.2965 35.2944 46.2383 41.8103

Columns 225 through 231

26.3878 22.8854 19.4712 16.1276 12.8396 9.5941 6.3794
37.6699 33.7490 44.4270 40.1240 36.0782 47.1666 42.6702

Columns 232 through 238

3.1847 0 -3.1847 -6.3794 -9.5941 -12.8396 -16.1276
38.4786 34.5181 45.3254 40.9618 36.8699 32.9867 43.5422

Columns 239 through 245

-19.4712 -22.8854 -26.3878 -30.0000 -33.7490 -37.6699 -41.8103
39.2965 35.2944 46.2383 41.8103 37.6699 33.7490 0

Columns 246 through 252

-46.2383 -51.0576 -56.4427 56.4427 51.0576 46.2383 41.8103
0 0 0 53.1301 48.1114 0 56.4427

Columns 253 through 259

37.6699 33.7490 30.0000 26.3878 22.8854 19.4712 16.1276
51.0576 0 0 54.2046 49.0739 0 57.6125

Columns 260 through 266

12.8396 9.5941 6.3794 3.1847 0 -3.1847 -6.3794
52.0818 0 0 55.3079 50.0555 0 58.8212

Columns 267 through 273

APPENDIX C

MATLAB CODES FOR BEAMSTEER(DIPOLE) V.1 AND BEAMSTEER(MICROSTRIP) V.1 PROGRAMS

Code for BeamSteer(Dipole) V.1

```
%Program to Predict the Phase Angle of the Feed Current
%=======================================================
%(-60~+60 sector was considered)
%Neural Network Used = GRNN
clear;
clc
disp('    ');
disp('    ');
disp('BeamStear ver 1.0');
disp('==================');
disp('Designed for the -60~+60 sector');


Finding_Training_Data_Set_120_m;
Neural_120_m;

t=-pi/2:0.01:pi/2;


count = 1;
while count == 1
   disp('    ');
   disp('    ');

   x=input('Enter the direction of the signal source in degrees
=>');disp('     ');
   X = pi*(x/180);
   z=input('Enter the direction of the interference signal in degrees
=>');
   Z = pi*(z/180);

   in_signal = [x;z];
   out_signal=sim(net,in_signal);

   disp('    ');
   if ((-160<out_signal) & (out_signal<160))
      disp('=======================================================');
      disp('Predicted phase angle of the feed current in degrees =')
      disp(out_signal)

      %Radiation pattern corresponding to the predicted phase angle
      ph1 = pi*(out_signal/180);


      E_array=(1/10)*(1+ cos(ph1+pi*sin(t))+imag(sin(ph1+pi*sin(t))) +
cos(2*(ph1+pi*sin(t)))+ imag(sin(2*(ph1+pi*sin(t)))) +
cos(3*(ph1+pi*sin(t)))+ imag(sin(3*(ph1+pi*sin(t)))) +
cos(4*(ph1+pi*sin(t)))+ imag(sin(4*(ph1+pi*sin(t)))) +
cos(5*(ph1+pi*sin(t)))+ imag(sin(5*(ph1+pi*sin(t)))) +
cos(6*(ph1+pi*sin(t)))+ imag(sin(6*(ph1+pi*sin(t)))) +
```

```
imag(sin(3*(ph1+pi*sin(X)))) + cos(4*(ph1+pi*sin(X)))+
imag(sin(4*(ph1+pi*sin(X)))) + cos(5*(ph1+pi*sin(X)))+
imag(sin(5*(ph1+pi*sin(X)))) + cos(6*(ph1+pi*sin(X)))+
imag(sin(6*(ph1+pi*sin(X)))) + cos(7*(ph1+pi*sin(X)))+
imag(sin(7*(ph1+pi*sin(X)))) + cos(8*(ph1+pi*sin(X)))+
imag(sin(8*(ph1+pi*sin(X)))) + cos(9*(ph1+pi*sin(X)))+
imag(sin(9*(ph1+pi*sin(X))))));
      MagNul =
abs((((sin(0.0166*cos(Z)))/(0.0166*cos(Z)))*cos((pi/2)*sin(Z)))*(1/10)*(
1+ cos(ph1+pi*sin(Z))+imag(sin(ph1+pi*sin(Z))) + cos(2*(ph1+pi*sin(Z)))+
imag(sin(2*(ph1+pi*sin(Z)))) + cos(3*(ph1+pi*sin(Z)))+
imag(sin(3*(ph1+pi*sin(Z)))) + cos(4*(ph1+pi*sin(Z)))+
imag(sin(4*(ph1+pi*sin(Z)))) + cos(5*(ph1+pi*sin(Z)))+
imag(sin(5*(ph1+pi*sin(Z)))) + cos(6*(ph1+pi*sin(Z)))+
imag(sin(6*(ph1+pi*sin(Z)))) + cos(7*(ph1+pi*sin(Z)))+
imag(sin(7*(ph1+pi*sin(Z)))) + cos(8*(ph1+pi*sin(Z)))+
imag(sin(8*(ph1+pi*sin(Z)))) + cos(9*(ph1+pi*sin(Z)))+
imag(sin(9*(ph1+pi*sin(Z))))));
      disp('    ');
      disp('=============================================');
      disp('Normalised magnitude of the field strength');
      disp('in the direction of the signal source =>');disp(MagReq);
      disp('Normalised magnitude of the field strength');
      disp('in the direction of the interferer    =>');disp(MagNul);
      disp('    ');
      disp('=============================================');

      %Calculating the isolation between two signals
      iso = 10*log10(MagReq/MagNul);
      disp('Isolation between two signals in dB=>');disp(iso);
```



```
   else
      disp('Sorry! The interference signal cannot be successfully')
      disp('isolated by this antenna array');
   end

   %More calculations
   disp('    ');
   loop_end = input('Do you want to do another calculation[1/0]? = ');
   if loop_end == 1
      count = 1;
      clc;
   else
      count = 0;
   end
end
clc
disp('This is the end of the program. Thank You!!!');
```

Code for BeamSteer(Microstrip) V.1

```
%Program to Predict the Phase Angle of the Feed Current
%=======================================================
%(-60~+60 sector was considered)
%Neural Network Used = GRNN
clear;
clc
disp('    ');
disp('    ');
```

```
      [pl_1,ql_1] = pol2cart(t1,((1/max_E1)*(abs(E1_1))));

      %-pi/3 to pi/3
      t2 = [-pi/3:0.01:pi/3];
      size(t2)
      for i=53:1:262
      E1_2(i-52) = E1(i);
      end
      [pl_2,ql_2] = pol2cart(t2,((1/max_E1)*(abs(E1_2))));

      %pi/3 to pi/2 (Suppressed)
      t3 = [pi/3:0.01:pi/2];
      size(t3)
      for i=263:1:315
         E1_3(i-262) = E1(i);
      end
      [pl_3,ql_3] = pol2cart(t3,((1/max_E1)*(abs(E1_3))));



      % Points to plot the sector boudaries
      Boundary = [0:0.1:1];
      [Boundary1x, Boundary1y] = pol2cart(-pi/3, Boundary);
      [Boundary2x, Boundary2y] = pol2cart(pi/3, Boundary);

      %Points to draw the cell

      Cell = [0:pi/100:2*pi];
      [Cellx,Celly] = pol2cart(Cell, 1);

   %Plotting the predicted phase angle
   plot(pl_1,ql_1,'r:',pl_2,ql_2,'r-
',pl_3,ql_3,'r:',Boundary1x,Boundary1y,'k-',Boundary2x,Boundary2y,'k-
',Cellx,Celly,'k-');


   %Magnitude of the field in the required direction

      MagReq =
abs((((sin(0.0166*cos(X)))/(0.0166*cos(X)))*cos((pi/2)*sin(X)))*(1/10)*(
1+ cos(ph1+pi*sin(X))+imag(sin(ph1+pi*sin(X))) + cos(2*(ph1+pi*sin(X)))+
imag(sin(2*(ph1+pi*sin(X)))) + cos(3*(ph1+pi*sin(X)))+
imag(sin(3*(ph1+pi*sin(X)))) + cos(4*(ph1+pi*sin(X)))+
imag(sin(4*(ph1+pi*sin(X)))) + cos(5*(ph1+pi*sin(X)))+
imag(sin(5*(ph1+pi*sin(X)))) + cos(6*(ph1+pi*sin(X)))+
imag(sin(6*(ph1+pi*sin(X)))) + cos(7*(ph1+pi*sin(X)))+
imag(sin(7*(ph1+pi*sin(X)))) + cos(8*(ph1+pi*sin(X)))+
imag(sin(8*(ph1+pi*sin(X)))) + cos(9*(ph1+pi*sin(X)))+
imag(sin(9*(ph1+pi*sin(X))))));
      MagNul =
abs((((sin(0.0166*cos(Z)))/(0.0166*cos(Z)))*cos((pi/2)*sin(Z)))*(1/10)*(
1+ cos(ph1+pi*sin(Z))+imag(sin(ph1+pi*sin(Z))) + cos(2*(ph1+pi*sin(Z)))+
imag(sin(2*(ph1+pi*sin(Z)))) + cos(3*(ph1+pi*sin(Z)))+
imag(sin(3*(ph1+pi*sin(Z)))) + cos(4*(ph1+pi*sin(Z)))+
imag(sin(4*(ph1+pi*sin(Z)))) + cos(5*(ph1+pi*sin(Z)))+
imag(sin(5*(ph1+pi*sin(Z)))) + cos(6*(ph1+pi*sin(Z)))+
imag(sin(6*(ph1+pi*sin(Z)))) + cos(7*(ph1+pi*sin(Z)))+
imag(sin(7*(ph1+pi*sin(Z)))) + cos(8*(ph1+pi*sin(Z)))+
imag(sin(8*(ph1+pi*sin(Z)))) + cos(9*(ph1+pi*sin(Z)))+
imag(sin(9*(ph1+pi*sin(Z))))));
      disp('   ');
```

APPENDIX D

MATLAB CODES FOR BEAMSTEER(DIPOLE) V.2 AND BEAMSTEER(MICROSTRIP) V.2 PROGRAMS

Code for BeamSteer(Dipole) V.2

```
%Program to Simulate Multiple Beamforming within 120 sector
%=======================================
%(-60~+60 sector was considered)
%Neural Network Used = GRNN
clc
disp('    ');
disp('    ');
disp('BeamStear ver 2.0');
disp('==================');
disp('Designed for the 180~270 sector');


Finding_Training_Data_Set_120_m;
Neural_120_m;

%phase1=[0:10:180];
t=-pi/2:0.01:pi/2;


count = 1;
while count == 1
   disp('    ');
   disp('    ');

   %Entering Data

   in_check = 1;
   while in_check == 1

      x1=input('Enter the direction of the first signal source in
degrees      =>');disp('      ');
      if (x1<-60)
         disp('Please enter a value between -60 and +60!')
      else
         if (x1>60)
            disp('Please enter a value between -60 and +60!');
         else
            in_check = 0;
         end
      end
   end


   in_check = 1;
   while in_check == 1
       x2=input('Enter the direction of the second signal source in
degrees     =>');disp('      ');
       if (x2<-60)
          disp('Please enter a value between -60 and +60!');
```

```
      [max_E1,I_E1] = max(abs(E1));
      clear E;
      clear E_array;


      E_array=(1/10)*(1+
cos(ph2+pi*sin(t+k3))+imag(sin(ph2+pi*sin(t+k3))) +
cos(2*(ph2+pi*sin(t+k3)))+ imag(sin(2*(ph2+pi*sin(t+k3)))) +
cos(3*(ph2+pi*sin(t+k3)))+ imag(sin(3*(ph2+pi*sin(t+k3)))) +
cos(4*(ph2+pi*sin(t+k3)))+ imag(sin(4*(ph2+pi*sin(t+k3)))) +
cos(5*(ph2+pi*sin(t+k3)))+ imag(sin(5*(ph2+pi*sin(t+k3)))) +
cos(6*(ph2+pi*sin(t+k3)))+ imag(sin(6*(ph2+pi*sin(t+k3)))) +
cos(7*(ph2+pi*sin(t+k3)))+ imag(sin(7*(ph2+pi*sin(t+k3)))) +
cos(8*(ph2+pi*sin(t+k3)))+ imag(sin(8*(ph2+pi*sin(t+k3)))) +
cos(9*(ph2+pi*sin(t+k3)))+ imag(sin(9*(ph2+pi*sin(t+k3)))));
      microstrip_120;
      E2=E;
      [max_E2,I_E2] = max(abs(E2));
      clear E;
      clear E_array;

      %Beam 1
      %=======
      %-pi/2 to -pi/3 (Suppressed Beam)


      t1 = [-pi/2:0.01:-pi/3];
      size(t1)
      for i=1:1:53
      E1_1(i) = E1(i);
      end
      [p1_1,q1_1] = pol2cart(t1,((1/max_E1)*(abs(E1_1))));

%-pi/3 to pi/3
t2 = [-pi/3:0.01:pi/3];
size(t2)
for i=53:1:262
   E1_2(i-52) = E1(i);
end
[p1_2,q1_2] = pol2cart(t2,((1/max_E1)*(abs(E1_2))));

%pi/3 to pi/2 (Suppressed)
t3 = [pi/3:0.01:pi/2];
size(t3)
for i=263:1:315
   E1_3(i-262) = E1(i);
end
[p1_3,q1_3] = pol2cart(t3,((1/max_E1)*(abs(E1_3))));

%Beam 2
%======
%-pi to -pi/3 (Suppressed Beam)
t1 = [-pi/2:0.01:-pi/3];
for i=1:1:53
   E2_1(i) = E2(i);
end
[p2_1,q2_1] = pol2cart(t1,((1/max_E2)*(abs(E2_1))));

%-pi/3 to pi/3
t2 = [-pi/3:0.01:pi/3];
for i=53:1:262
```

```
   else
      disp('Sorry! The interference signal cannot be successfully')
      disp('isolated by this antenna array');
   end

   %More calculations
   disp('    ');
   loop_end = input('Do you want to do another calculation[1/0]? = ');
   if loop_end == 1
      count = 1;
      clc;
   else
      count = 0;
   end
end
clc
disp('This is the end of the program. Thank You!!!');
```

Code for BeamSteer(Microstrip) V.2

```
%Program to Simulate Multiple Beamforming within 120 sector
%=========================================
%(-60~+60 sector was considered)
%Neural Network Used = GRNN
clc
clear
disp('    ');
disp('    ');
disp('BeamStear ver 2.0');
disp('==================');
disp('Designed for the 180~270 sector');


Finding_Training_Data_Set_120_m;
Neural_120_m;

%phase1=[0:10:180];
t=-pi/2:0.01:pi/2;


count = 1;
while count == 1
   disp('    ');
   disp('    ');

   %Entering Data

   in_check = 1;
   while in_check == 1

      x1=input('Enter the direction of the first signal source in
degrees      =>');disp('      ');
      if (x1<-60)
         disp('Please enter a value between -60 and +60!')
      else
         if (x1>60)
            disp('Please enter a value between -60 and +60!');
         else
            in_check = 0;
         end
```

```
cos(2*(ph1+pi*sin(t+k3)))+ imag(sin(2*(ph1+pi*sin(t+k3)))) +
cos(3*(ph1+pi*sin(t+k3)))+ imag(sin(3*(ph1+pi*sin(t+k3)))) +
cos(4*(ph1+pi*sin(t+k3)))+ imag(sin(4*(ph1+pi*sin(t+k3)))) +
cos(5*(ph1+pi*sin(t+k3)))+ imag(sin(5*(ph1+pi*sin(t+k3)))) +
cos(6*(ph1+pi*sin(t+k3)))+ imag(sin(6*(ph1+pi*sin(t+k3)))) +
cos(7*(ph1+pi*sin(t+k3)))+ imag(sin(7*(ph1+pi*sin(t+k3)))) +
cos(8*(ph1+pi*sin(t+k3)))+ imag(sin(8*(ph1+pi*sin(t+k3)))) +
cos(9*(ph1+pi*sin(t+k3)))+ imag(sin(9*(ph1+pi*sin(t+k3)))));
      microstrip_120;
      E1=E;
      [max_E1,I_E1] = max(abs(E1));
      clear E;
      clear E_array;


      E_array=(1/10)*(1+
cos(ph2+pi*sin(t+k3))+imag(sin(ph2+pi*sin(t+k3))) +
cos(2*(ph2+pi*sin(t+k3)))+ imag(sin(2*(ph2+pi*sin(t+k3)))) +
cos(3*(ph2+pi*sin(t+k3)))+ imag(sin(3*(ph2+pi*sin(t+k3)))) +
cos(4*(ph2+pi*sin(t+k3)))+ imag(sin(4*(ph2+pi*sin(t+k3)))) +
cos(5*(ph2+pi*sin(t+k3)))+ imag(sin(5*(ph2+pi*sin(t+k3)))) +
cos(6*(ph2+pi*sin(t+k3)))+ imag(sin(6*(ph2+pi*sin(t+k3)))) +
cos(7*(ph2+pi*sin(t+k3)))+ imag(sin(7*(ph2+pi*sin(t+k3)))) +
cos(8*(ph2+pi*sin(t+k3)))+ imag(sin(8*(ph2+pi*sin(t+k3)))) +
cos(9*(ph2+pi*sin(t+k3)))+ imag(sin(9*(ph2+pi*sin(t+k3)))));
      microstrip_120;
      E2=E;
      [max_E2,I_E2] = max(abs(E2));
      clear E;
      clear E_array;

      %Beam 1
%=======
%-pi/2 to -pi/3 (Suppressed Beam)


t1 = [-pi/2:0.01:-pi/3];
size(t1)
clear E1_1;
for i=1:1:53
   E1_1(i) = E1(i);
end
[p1_1,q1_1] = pol2cart(t1,((1/max_E1)*(abs(E1_1))));

%-pi/3 to pi/3
t2 = [-pi/3:0.01:pi/3];
size(t2)
clear E1_2;
for i=53:1:262
   E1_2(i-52) = E1(i);
end
[p1_2,q1_2] = pol2cart(t2,((1/max_E1)*(abs(E1_2))));

%pi/3 to pi/2 (Suppressed)
t3 = [pi/3:0.01:pi/2];
size(t3)
clear E1_3;
for i=263:1:315
   E1_3(i-262) = E1(i);
end
[p1_3,q1_3] = pol2cart(t3,((1/max_E1)*(abs(E1_3))));
```

```
cos(6*(ph+pi*sin(Z)))+ imag(sin(6*(ph+pi*sin(Z)))) +
cos(7*(ph+pi*sin(Z)))+ imag(sin(7*(ph+pi*sin(Z)))) +
cos(8*(ph+pi*sin(Z)))+ imag(sin(8*(ph+pi*sin(Z)))) +
cos(9*(ph+pi*sin(Z)))+ imag(sin(9*(ph+pi*sin(Z)))));
      %disp('  ');
      %disp('=============================================');
      %disp('Normalised magnitude of the field strength');
      %disp('in the direction of the signal source =>');disp(MagReq);
      %disp('Normalised magnitude of the field strength');
      %disp('in the direction of the interferer    =>');disp(MagNul);
      %disp('  ');
      %disp('=============================================');

      %Calculating the isolation between two signals
      %iso = 10*log10(MagReq/MagNul);
      %disp('Isolation between two signals in dB=>');disp(iso);
      %
   else
      disp('Sorry! The interference signal cannot be successfully')
      disp('isolated by this antenna array');
   end

   %More calculations
   disp('    ');
   loop_end = input('Do you want to do another calculation[1/0]? = ');
   if loop_end == 1
      count = 1;
      clc;
   else
      count = 0;
   end
end
clc
disp('This is the end of the program. Thank You!!!');
```

```
        disp ('Two beams will serve three clusters');
        disp ('Users served from Beam 1 =');disp(b(2)+d(2));
        disp ('Users served from Beam 2 =');disp(a(2));
        beam_ = 3
    elseif ((abs(x2-x1)>5)& (abs(x3-x1)<5)& (abs(x2-x3)>5))
        disp ('Two beams will serve three clusters');
        disp ('Users served from Beam 1 =');disp(a(2)+d(2));
        disp ('Users served from Beam 2 =');disp(b(2));
        beam_ = 4
    elseif ((abs(x2-x1)>5)& (abs(x3-x1)>5)& (abs(x2-x3)>5))
        disp ('Three beams will serve three clusters');
        disp ('Users served from Beam 1 =');disp(a(2));
        disp ('Users served from Beam 2 =');disp(b(2));
        disp ('Users served from Beam 3 =');disp(d(2));
        beam_ = 5
    end

    x11=x1;
    x22=x2;
    x33=x3;


    %Converting entered values from degrees to radians
    X1 = pi*(x11/180);
    X2 = pi*(x22/180);
    X3 = pi*(x33/180);


    %Both signal sources become interferring sources to each other
    z1 = x22;
    z2 = x11;

    % To find the Interferring Direction for x3
    a = abs(x3-x1);
    b = abs(x3-x2);
    if a>b
        z3 = x22;
    else
        z3 = x11;
    end


    Z1 = pi*(z1/180);
    Z2 = pi*(z2/180);
    Z3 = pi*(z3/180);

    in_signal = [x11;z1];
    out_signal=sim(net,in_signal);

    in_signal2 = [x22;z2];
    out_signal2 = sim(net,in_signal2);

    in_signal3 = [x33;z3];
    out_signal3 = sim(net,in_signal3);

    disp('    ');
    if (((-160<out_signal) & (out_signal<160)) & ((-160<out_signal2) &
(out_signal2<160))&((-160<out_signal3) & (out_signal3<160)))

disp('=================================================================
=======================================');
```

```
E3=E;
[max_E3,I_E3] = max(abs(E3));
clear E;
clear E_array;



R1_ = abs(E1);
R2_ = abs(E2);
R3_ = abs(E3);


DOA_points = 1:0.05:1;
DOA_points1 = 1:0.05:1;
DOA_points2 = 1:0.05:1;
DOA_points3 = 1:0.05:1;

%Defining the beams to be displayed depending on the seperability
if beam_==1
   R1 = (1/3)*(R1_+R2_+R3_);
   R2 = 0*R2_;
   R3 = 0*R3_;

elseif beam_==2
   R1 = 0.5*(R1_+R2_);
   R2 = 0*R2_;
   R3 = R3_;
   DOA_points2= 0;
elseif beam_==3
   R1 = R1_;
   R2 = 0.5*(R2_+R3_);
   R3=0*R3_;
   DOA_points3 = 0;

elseif beam_==4
   R1 = 0.5*(R1_+R3_);
   R2 = R2_;
   R3 =0*R3_;
   DOA_points3 = 0;

elseif beam_==5
   R1 = R1_;
   R2 = R2_;
   R3 = R3_;
end

%Finding the maximum of lobes

[R1__ , I1] = min(-1*R1 + 1);
[R2__ , I2] = min(-1*R2 + 1);
[R3__ , I3] = min(-1*R3 + 1);

Max_1 = 0.01*I1 + (-pi);
Max_2 = 0.01*I2 + (-pi);
Max_3 = 0.01*I3 + (-pi);
clear I1;
clear I2;
clear I3;
```

```
size(t3)
for i=263:1:315
   E1_3(i-262) = E1(i);
end
[p1_3,q1_3] = pol2cart(t3,((1/max_E1)*(abs(E1_3))));

%Beam 2
%======
%-pi to -pi/3 (Suppressed Beam)
t1 = [-pi/2:0.01:-pi/3];
for i=1:1:53
   E2_1(i) = E2(i);
end
[p2_1,q2_1] = pol2cart(t1,((1/max_E2)*(abs(E2_1))));

%-pi/3 to pi/3
t2 = [-pi/3:0.01:pi/3];
for i=53:1:262
   E2_2(i-52) = E2(i);
end
[p2_2,q2_2] = pol2cart(t2,((1/max_E2)*(abs(E2_2))));

%pi/3 to pi/2 (Suppressed)
t3 = [pi/3:0.01:pi/2];
for i=263:1:315
   E2_3(i-262) = E2(i);
end
[p2_3,q2_3] = pol2cart(t3,((1/max_E2)*(abs(E2_3))));

%Beam 3
%======
%-pi to -pi/3 (Suppressed Beam)
t1 = [-pi/2:0.01:-pi/3];
for i=1:1:53
   E3_1(i) = E3(i);
end
[p3_1,q3_1] = pol2cart(t1,((1/max_E3)*(abs(E3_1))));

%-pi/3 to pi/3
t2 = [-pi/3:0.01:pi/3];
for i=53:1:262
   E3_2(i-52) = E3(i);
end
[p3_2,q3_2] = pol2cart(t2,((1/max_E3)*(abs(E3_2))));

%pi/3 to pi/2 (Suppressed)
t3 = [pi/3:0.01:pi/2];
for i=263:1:315
   E3_3(i-262) = E3(i);
end
[p3_3,q3_3] = pol2cart(t3,((1/max_E3)*(abs(E3_3))));



% Points to plot the sector boudaries
Boundary = [0:0.1:1];
[Boundary1x, Boundary1y] = pol2cart(-pi/3, Boundary);
[Boundary2x, Boundary2y] = pol2cart(pi/3, Boundary);

%Points to draw the cell
```

APPENDIX F

MATLAB CODE FOR BEAMSTEER(MICROSTRIP) V.10 PROGRAM

```
clear;
UserPositionGeneration_120_m;
%Output from UserPositionGeneration -> Inputs

%To find the training data set
Finding_Training_Data_Set_120_m;
Neural_120_m;
%Output from Neural -> net
%HP Beamwidth calculation
t=-pi/2:0.01:pi/2;
variable_beam_120_m;

count =1;

while count < 30
   count
   %Input to the data_sorting -> Data
   time_step = count;
   Inputs = rot90(UTMatrix(:,time_step));
   Data = Inputs;
   data_sorting_120_m;
   %Output from data_sorting -> BeamWidth_1 , BeamDirection_1 ,
BeamDirection_2, BeamDirection_3
```



```
   %Converting beam directions from degrees to radians
   X1 = pi*(BeamDirection_1/180);
   X2 = pi*(BeamDirection_2/180);
   X3 = pi*(BeamDirection_3/180);

   BeamDirection_1
   BeamDirection_2
   BeamDirection_3

   %%Determines arrays for Beams 2 and 3

   x11=BeamDirection_1;
   x22=BeamDirection_2;
   x33=BeamDirection_3;

%if (BeamDirection_2<225)
 %     disp('Array 0-180(2) is suitable for Beam 2');
  %    k2=0;
   %    x22=BeamDirection_2;
   %else
    %  disp('Array 90-270(2) is suitable for Beam 2');
     % k2=pi/2;
      %x22=(270-BeamDirection_2)+180;
   %end

   %if (BeamDirection_3<225)
    %  disp('Array 0-180(3) is suitable for Beam 3');
     % k3=0;
```

```
      E_array=(1/10)*(1+ cos(ph1+pi*sin(t))+imag(sin(ph1+pi*sin(t))) +
cos(2*(ph1+pi*sin(t)))+ imag(sin(2*(ph1+pi*sin(t)))) +
cos(3*(ph1+pi*sin(t)))+ imag(sin(3*(ph1+pi*sin(t)))) +
cos(4*(ph1+pi*sin(t)))+ imag(sin(4*(ph1+pi*sin(t)))) +
cos(5*(ph1+pi*sin(t)))+ imag(sin(5*(ph1+pi*sin(t)))) +
cos(6*(ph1+pi*sin(t)))+ imag(sin(6*(ph1+pi*sin(t)))) +
cos(7*(ph1+pi*sin(t)))+ imag(sin(7*(ph1+pi*sin(t)))) +
cos(8*(ph1+pi*sin(t)))+ imag(sin(8*(ph1+pi*sin(t)))) +
cos(9*(ph1+pi*sin(t)))+ imag(sin(9*(ph1+pi*sin(t)))));
      microstrip;
      E1=E;
      clear E;
      clear E_array;
   elseif NumberOfElements == 9
      E_array=(1/9)*(1+ cos(ph1+pi*sin(t))+imag(sin(ph1+pi*sin(t))) +
cos(2*(ph1+pi*sin(t)))+ imag(sin(2*(ph1+pi*sin(t)))) +
cos(3*(ph1+pi*sin(t)))+ imag(sin(3*(ph1+pi*sin(t)))) +
cos(4*(ph1+pi*sin(t)))+ imag(sin(4*(ph1+pi*sin(t)))) +
cos(5*(ph1+pi*sin(t)))+ imag(sin(5*(ph1+pi*sin(t)))) +
cos(6*(ph1+pi*sin(t)))+ imag(sin(6*(ph1+pi*sin(t)))) +
cos(7*(ph1+pi*sin(t)))+ imag(sin(7*(ph1+pi*sin(t)))) +
cos(8*(ph1+pi*sin(t)))+ imag(sin(8*(ph1+pi*sin(t)))));
      microstrip;
      E1=E;
      clear E;
      clear E_array;
   elseif NumberOfElements == 8
      E_array=(1/8)*(1+ cos(ph1+pi*sin(t))+imag(sin(ph1+pi*sin(t))) +
cos(2*(ph1+pi*sin(t)))+ imag(sin(2*(ph1+pi*sin(t)))) +
cos(3*(ph1+pi*sin(t)))+ imag(sin(3*(ph1+pi*sin(t)))) +
cos(4*(ph1+pi*sin(t)))+ imag(sin(4*(ph1+pi*sin(t)))) +
cos(5*(ph1+pi*sin(t)))+ imag(sin(5*(ph1+pi*sin(t)))) +
cos(6*(ph1+pi*sin(t)))+ imag(sin(6*(ph1+pi*sin(t)))) +
cos(7*(ph1+pi*sin(t)))+ imag(sin(7*(ph1+pi*sin(t)))));
      microstrip;
      E1=E;
      clear E;
      clear E_array;
   elseif NumberOfElements == 7
      E_array=(1/7)*(1+ cos(ph1+pi*sin(t))+imag(sin(ph1+pi*sin(t))) +
cos(2*(ph1+pi*sin(t)))+ imag(sin(2*(ph1+pi*sin(t)))) +
cos(3*(ph1+pi*sin(t)))+ imag(sin(3*(ph1+pi*sin(t)))) +
cos(4*(ph1+pi*sin(t)))+ imag(sin(4*(ph1+pi*sin(t)))) +
cos(5*(ph1+pi*sin(t)))+ imag(sin(5*(ph1+pi*sin(t)))) +
cos(6*(ph1+pi*sin(t)))+ imag(sin(6*(ph1+pi*sin(t)))));
      microstrip;
      E1=E;
      clear E;
      clear E_array;
   elseif NumberOfElements == 6
      E_array=(1/6)*(1+ cos(ph1+pi*sin(t))+imag(sin(ph1+pi*sin(t))) +
cos(2*(ph1+pi*sin(t)))+ imag(sin(2*(ph1+pi*sin(t)))) +
cos(3*(ph1+pi*sin(t)))+ imag(sin(3*(ph1+pi*sin(t)))) +
cos(4*(ph1+pi*sin(t)))+ imag(sin(4*(ph1+pi*sin(t)))) +
cos(5*(ph1+pi*sin(t)))+ imag(sin(5*(ph1+pi*sin(t)))));
      microstrip;
      E1=E;
      clear E;
      clear E_array;
    elseif NumberOfElements == 5
```

```
cos(8*(ph3+pi*sin(t+k3)))+ imag(sin(8*(ph3+pi*sin(t+k3)))) +
cos(9*(ph3+pi*sin(t+k3)))+ imag(sin(9*(ph3+pi*sin(t+k3))));
  microstrip;
  E3=E;
  [max_E3,I_E3] = max(abs(E3));
  clear E;
  clear E_array;

%%Points to draw sources

%Defining points for SortedData
SortedDataAmount = length(SortedData);
SortedDataPosition = zeros(1,SortedDataAmount);
for SortedDataIndex=1:1:SortedDataAmount
   SortedDataPosition(SortedDataIndex) = (1-0.25*rand);
end

SortedDataSecondAmount = length(SortedDataSecond);
SortedDataSecondPosition = zeros(1,SortedDataSecondAmount);
for SortedDataSecondIndex=1:1:SortedDataSecondAmount
   SortedDataSecondPosition(SortedDataSecondIndex) = (1-0.25*rand);
end

SortedDataThirdAmount = length(SortedDataThird);
SortedDataThirdPosition = zeros(1,SortedDataThirdAmount);
for SortedDataThirdIndex=1:1:SortedDataThirdAmount
   SortedDataThirdPosition(SortedDataThirdIndex) = (1-0.25*rand);
end

RemainingDataAmount = length(RemainingData);
RemainingDataPosition = zeros(1,RemainingDataAmount);
for RemainingDataIndex=1:1:RemainingDataAmount
   RemainingDataPosition(RemainingDataIndex) = (1-0.25*rand);
end

[S1x,S1y] = pol2cart((SortedData/180)*pi,SortedDataPosition); %Inputs ->
UserPostitionGeneration
[S2x,S2y] =
pol2cart((SortedDataSecond/180)*pi,SortedDataSecondPosition);
[S3x,S3y] = pol2cart((SortedDataThird/180)*pi,SortedDataThirdPosition);
[SRx,SRy] = pol2cart((RemainingData/180)*pi,RemainingDataPosition);


%%Points to plot beams

%Beam 1
%=======
%-pi/2 to -pi/3 (Suppressed Beam)


t1 = [-pi/2:0.01:-pi/3];
size(t1)
for i=1:1:53
   E1_1(i) = E1(i);
end
[p1_1,q1_1] = pol2cart(t1,((1/max_E1)*(abs(E1_1))));

%-pi/3 to pi/3
t2 = [-pi/3:0.01:pi/3];
size(t2)
for i=53:1:262
```

```
%[p1,q1] = pol2cart(t,abs(E1));
%[p2,q2] = pol2cart(t,abs(E2));
%[p3,q3] = pol2cart(t,abs(E3));

% Points to plot the sector boudaries
Boundary = [0:0.1:1];
[Boundary1x, Boundary1y] = pol2cart(-pi/3, Boundary);
[Boundary2x, Boundary2y] = pol2cart(pi/3, Boundary);

%Points to draw the cell

Cell = [0:pi/100:2*pi];
[Cellx,Celly] = pol2cart(Cell, 1);

%%Plotting the graph

plot(S1x,S1y,'r*',S2x,S2y,'b+',S3x,S3y,'g^',SRx,SRy,'ks',p1_1,q1_1,'r:',
p1_2,q1_2,'r-',p1_3,q1_3,'r:',p2_1,q2_1,'b:',p2_2,q2_2,'b-
',p2_3,q2_3,'b:',p3_1,q3_1,'g:',p3_2,q3_2,'g-
',p3_3,q3_3,'g:',Boundary1x,Boundary1y,'k-',Boundary2x,Boundary2y,'k-
',Cellx,Celly,'k-');
pause(0.001);

count = count+1;
if count == 30
   continue = input('Do you want to continue (1/0) ?');
   if continue == 1
      count = 1    end
end
end
```


UserPositionGeneration

```
%Defining points in time - In minutes
clc;
PointsInTime = [0:1:30];
%Angular Positions of mobile stations at 'Defined Points in Time'

%User1
%=====
%Start = -60 , Step = 1
u1 = [-60:1:-31];

%User2
%=====
%Start = -60 , Step = 2
u2 = [-60:2:-1];

%User3
%=====
%Start = 181 , Step = 3 ;
u3 = [-60:3:29];

%User4
%=====
%Start = -55 , Step = 1
u4 = [-55:1:-26];

%User5
```

```
%Start = 54 , Step = -2
u17 = [54:-2:-5];

%User18
%=====
%Start = 54 , Step = -3
u18 = [54:-3:-35];

%User19
%=====
%Start = 49 , Step = -1
u19 = [49:-1:20];

%User20
%=====
%Start = 49 , Step = -2;
u20 = [49:-2:-10];

%User21
%=====
%Start = 49 , Step = -3
u21 = [49:-3:-40];

%User22
%=====
%Start = 44 , Step = -1
u22 = [44:-1:15];

%User23
%=====
%Start = 44 , Step = -2
u23 = [44:-2:-15];

%User24
%=====
%Start = 44 , Step = -3
u24 = [44:-3:-45];

%User25
%======
%Start = -40, Step = 1
u25 = [-40:1:-11];

%User26
%=======
%Start = -38, Step = 1
u26 = [-38:1:-9];

%User27
%=======
%Start = -36, Step = 1
u27 = [-36:1:-7];

%User28
%=======
%Start = -34, Step = 1
u28 = [-34:1:-5];

%User29
%=======
%Start = -32, Step = 1
```

```
%User42
%=======
%Start = 28, Step = -1
u42 = [28:-1:-1];

%User43
%=======
%Start = 26, Step = -1
u43 = [26:-1:-3];

%User44
%=======
%Start = 24, Step = -1
u44 = [24:-1:-5];

%User45
%=======
%Start = 22, Step = -1
u45 = [22:-1:-7];

%User46
%=======
%Start = 20, Step = -1
u46 = [20:-1:-9];

%User47
%=======
%Start = 18, Step = -1
u47 = [18:-1:-11];

%User48
%=======
%Start = 16, Step = -1
u48 = [16:-1:-13];

%User49
%=======
%Start = 14, Step = -1
u49 = [14:-1:-15];

%User50
%=======
%Start = 12, Step = -1
u50 = [12:-1:-17];

%User-Time Metrix
UTMatrix =
[u1;u2;u3;u4;u5;u6;u7;u8;u9;u10;u11;u12;u13;u14;u15;u16;u17;u18;u19;u20;
u21;u22;u23;u24;u25;u26;u27;u28;u29;u30;u31;u32;u33;u34;u35;u36;u37;u38;
u39;u40;u41;u42;u43;u44;u45;u46;u47;u48;u49];
```


Neural

```
%%Neural Network Training
%Designed for 180-270 sector

phase1=[-150:10:150];
t = (-pi/3):0.01:pi/3;
```

```
cos(7*(ph+pi*sin(T)))+ imag(sin(7*(ph+pi*sin(T)))) +
cos(8*(ph+pi*sin(T)))+ imag(sin(8*(ph+pi*sin(T)))));
   NumElements = 9;
   microstrip;
   HPBW_120_m;

   E_array=(1/8)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T))) +
cos(2*(ph+pi*sin(T)))+ imag(sin(2*(ph+pi*sin(T)))) +
cos(3*(ph+pi*sin(T)))+ imag(sin(3*(ph+pi*sin(T)))) +
cos(4*(ph+pi*sin(T)))+ imag(sin(4*(ph+pi*sin(T)))) +
cos(5*(ph+pi*sin(T)))+ imag(sin(5*(ph+pi*sin(T)))) +
cos(6*(ph+pi*sin(T)))+ imag(sin(6*(ph+pi*sin(T)))) +
cos(7*(ph+pi*sin(T)))+ imag(sin(7*(ph+pi*sin(T)))));
   NumElements = 8;
   microstrip;
   HPBW_120_m;

   E_array=(1/7)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T))) +
cos(2*(ph+pi*sin(T)))+ imag(sin(2*(ph+pi*sin(T)))) +
cos(3*(ph+pi*sin(T)))+ imag(sin(3*(ph+pi*sin(T)))) +
cos(4*(ph+pi*sin(T)))+ imag(sin(4*(ph+pi*sin(T)))) +
cos(5*(ph+pi*sin(T)))+ imag(sin(5*(ph+pi*sin(T)))) +
cos(6*(ph+pi*sin(T)))+ imag(sin(6*(ph+pi*sin(T)))));
   NumElements = 7;
   microstrip;
   HPBW_120_m;

   E_array=(1/6)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T))) +
cos(2*(ph+pi*sin(T)))+ imag(sin(2*(ph+pi*sin(T)))) +
cos(3*(ph+pi*sin(T)))+ imag(sin(3*(ph+pi*sin(T)))) +
cos(4*(ph+pi*sin(T)))+ imag(sin(4*(ph+pi*sin(T)))) +
cos(5*(ph+pi*sin(T)))+ imag(sin(5*(ph+pi*sin(T)))));
   NumElements = 6;
   microstrip;
   HPBW_120_m;

   E_array=(1/5)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T))) +
cos(2*(ph+pi*sin(T)))+ imag(sin(2*(ph+pi*sin(T)))) +
cos(3*(ph+pi*sin(T)))+ imag(sin(3*(ph+pi*sin(T)))) +
cos(4*(ph+pi*sin(T)))+ imag(sin(4*(ph+pi*sin(T)))));
   NumElements = 5;
   microstrip;
   HPBW_120_m;

   E_array=(1/4)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T))) +
cos(2*(ph+pi*sin(T)))+ imag(sin(2*(ph+pi*sin(T)))) +
cos(3*(ph+pi*sin(T)))+ imag(sin(3*(ph+pi*sin(T)))) );
   NumElements = 4;
   microstrip;
   HPBW_120_m;

   E_array=(1/3)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T))) +
cos(2*(ph+pi*sin(T)))+ imag(sin(2*(ph+pi*sin(T)))));
   NumElements = 3;
   microstrip;
   HPBW_120_m;

   E_array=(1/2)*(1+ cos(ph+pi*sin(T))+imag(sin(ph+pi*sin(T)))) ;
   NumElements = 2;
   microstrip;
   HPBW_120_m;
```

```
end
SortedData2 = SortedData22;

%%SortedData2 includes data that do not fall under beam 1

%%Finding the Second Beam
SortedDataSecond = sort(SortedData2)
SortedData2Bk = SortedData2;
No_of_Data2 = length(SortedData2)
Deviation2 = std(SortedDataSecond)


while Deviation2 > 5
   Difference2 = zeros(1,No_of_Data2)
   Mean2 = mean(SortedDataSecond)
   Deviation2 = std(SortedDataSecond)
   for i=1:1:No_of_Data2
      Difference2(i) = abs(Mean2 - SortedDataSecond(i))
   end

   [x,I] = max(Difference2)
   SortedDataSecond(I) = 0
   NonZeroElements2 = find(SortedDataSecond)
   SortedData_ = zeros(1,(No_of_Data2 - 1))
   No_of_Data2 = No_of_Data2 - 1
   for i=1:1:length(NonZeroElements2)
      SortedData_(i)=SortedDataSecond(NonZeroElements2(i))
   end
   SortedDataSecond = SortedData_
   No_of_Data2 = length(SortedDataSecond);
   clear SortedData_;
end
SortedDataSecond
mean(SortedDataSecond)
std(SortedDataSecond)

%Calculating the Beam width required to feed cluster 2

BeamWitdh_2 = max(SortedDataSecond)- min(SortedDataSecond)
BeamDirection_2 = mean(SortedDataSecond)

%%Isolating data relevent to Beam 3
SortedData3 = SortedData2Bk
for i=1:1:length(SortedDataSecond)
   for j=1:1:length(SortedData3)
      if SortedDataSecond(1,i)==SortedData3(1,j)
         SortedData3(1,j) = 0;
      end
   end
end

NonZeroElementsInSortedData3 = find(SortedData3);
SortedData33 = length(NonZeroElementsInSortedData3);
for i=1:1:length(NonZeroElementsInSortedData3)
   SortedData33(i)=SortedData3(NonZeroElementsInSortedData3(i))
end
SortedData3 = SortedData33;

%%SortedData3 includes data that do not fall under beam 1 and 2

%%Finding the third Beam
```