Contents


```
clear; close all;
%
%
%%%%%%%%%%%%%%%%
%This is for JONES radar
%%%%%%%%%%%%%%%%
C = 299.792458;% Speed of light in Mega m/s
F0 = 31;% 47.0 MHz
lambda0=C/F0;% meter


xy = [...
    0, 2;...
    0, -2.5;...
    -2, 0;...
    2.5, 0;...
    0, 0];
xpos = zeros(size(xy,1),1)*lambda0;
ypos = zeros(size(xy,1),1)*lambda0;
zpos = zeros(size(xy,1),1)*lambda0;




c0 = 2.99792e8;

%%%%%%%%%%%%%%%%
%THIS IS FOR PANSY RADAR
%%%%%%%%%%%%%%%%%
%
% f0 = 46.5e6;
% % f0 = 50e6;
% lambda0 = c0/f0;
%

%
% load('subarrypos101.txt')
%
% core = find(sqrt(sum(subarrypos101.^2,2)) <= 50);
% outlier = find(sqrt(sum(subarrypos101.^2,2)) > 50);
%
% subarrypos101(outlier,:) = [];
%
% subarrypos101(6,:) = subarrypos101(end,:);
% subarrypos101(end,:) = [0,0];
%

%%%%%%%%%%%%%%%%
%THIS IS FOR EISCAT 3D
%%%%%%%%%%%%%%%%
%
% C = 299.792458;% Speed of light in Mega m/s
% F0 = 233;% 47.0 MHz
% lambda0=C/F0;% meter
%
% xy = subarrypos101./lambda0;
%
% xpos = zeros(size(xy,1),1)*lambda0;
% ypos = zeros(size(xy,1),1)*lambda0;
% zpos = zeros(size(xy,1),1)*lambda0;
%

%%%%%%%%%%%%%%%%
%THIS is for MU RADAR
%%%%%%%%%%%%%%%%
% C = 299.792458;% Speed of light in Mega m/s
% F0 = 46.5;% 46.5 MHz
% lambda0=C/F0;% meter
%
% load('/home/danielk/IRF/IRF_GITLAB/METEOR_MUSIC_MC_ORB_DISTRIBUTION/DEPLOY/xy_muant25.mat')
% [xpos,ypos,zpos] = muantennas();
%
% INDEX_OF_ASSYMETRIC = 1:4:21;
% INDEX_OF_SYMETRIC = 1:25;
% INDEX_OF_SYMETRIC(INDEX_OF_ASSYMETRIC) = [];
%
% xpos = xpos(INDEX_OF_SYMETRIC,:);
% ypos = ypos(INDEX_OF_SYMETRIC,:);
% zpos = zpos(INDEX_OF_SYMETRIC,:);
% xy = xy(INDEX_OF_SYMETRIC,:);

%%%%%%%%%%%%%%%%
% THIS IS FOR JICAMARCA RADAR
%%%%%%%%%%%%%%%%
% d = 144;
```

```
% xy = [-d/lambda0/2,-d/lambda0/2;...
%     d/lambda0/2,-d/lambda0/2;...
%     d/lambda0/2,d/lambda0/2;...
%     -(36+36/2)/lambda0,(36*3+36/2)/lambda0;...
%     -(36+36/2)/lambda0,(36*2+36/2)/lambda0;...
%     -(36*2+36/2)/lambda0,(36*2+36/2)/lambda0];
% xpos = zeros(6,1)*lambda0;
% ypos = zeros(6,1)*lambda0;
% zpos = zeros(6,1)*lambda0;

%%%%%%%%%%%%%%%%
```

TEST BY GENERATING A PERFECT SYMMETRIC ARRAY

```
%%%%%%%%%%%%%%%%
% d = 3;
% % xy = [d,0;...
% %     -d,0;...
% %     0,d;...
% %     0,-d;...
% %     d/sqrt(2),d/sqrt(2);...
% %     -d/sqrt(2),d/sqrt(2);...
% %     d/sqrt(2),-d/sqrt(2);...
% %     -d/sqrt(2),-d/sqrt(2);...
% %     0,0];
%
% xy = [d*cosd(67.5),d*sind(67.5);...
%     d*cosd(-67.5),d*sind(-67.5);...
%     d*cosd(112.5),d*sind(112.5);...
%     d*cosd(-112.5),d*sind(-112.5);...
%     0,0];
% xpos = zeros(5,1);
% ypos = zeros(5,1);
% zpos = zeros(5,1);
%
% xy = [d*cosd(67.5),d*sind(67.5);...
%     d*cosd(112.5),d*sind(112.5);...
%     0,-d;...
%     0,0];
% xpos = (zeros(4,4) + repmat([0,0,-d/6,d/6],4,1) + repmat(xy(:,1),1,4))*lambda0;
% ypos = (zeros(4,4) + repmat([-d/6,d/6,0,0],4,1) + repmat(xy(:,2),1,4))*lambda0;
% zpos = zeros(4,4);


% MP_tol = 1e-3
% MP_tol = 3.5e-2
MP_tol = 1e-1
% MP_tol = 1.13e-1



gain_yagi = 7.24;
gain_yagi_base = 10^(gain_yagi/10);
%figure size configurations
axis_font_size = 18;
title_font_size = 16;
legend_font_size = 14;

xant = xpos/lambda0;
yant = ypos/lambda0;

Zn = size(zpos,2); %subgroup size

% DO NOT COUNT THE LAST GROUP AS IT IS DEFINED AS THE ORIGIN
Sn = size(zpos,1); %sensorgroups
Sn = Sn - 1;

% Sn = 4;

r=zeros(3,Zn,Sn);
R=zeros(3,Sn);
R(:,:) = [xy(1:Sn,1).'; xy(1:Sn,2).'; (xy(1:Sn,1).')*0];

for i=1:Sn
    r(:,:,i) = [xpos(i,:); ypos(i,:); zpos(i,:)]./lambda0;
end


rho=zeros(3,Zn,Sn);
for i=1:Sn
    rho(:,:,i) = r(:,:,i) - repmat(R(:,i),1,Zn);
end





k=@(th,ph) [cos(th).*sin(ph); sin(th).*sin(ph); cos(ph)];

%calculate the linear coefficients
```

```
K = zeros(Sn,1);
for i=1:Sn
    Rni = norm(R(:,i));
    K(i) = R(1,i)^2/Rni + R(2,i)^2/Rni + R(3,i)^2/Rni;
end



% calculate the base numbers
n0 = floor(2*K);

% k's from 1 to 2n0+1
k_length = 2*n0+1;

%calculate all the integer planes
r_jk =@(j,k) (n0(j)+1-k)/K(j);
p0_jk =@(j,k) [r_jk(j,k)*R(1,j)/norm(R(:,j)); r_jk(j,k)*R(2,j)/norm(R(:,j)); r_jk(j,k)*R(3,j)/norm(R(:,j))];
nvec_j = @(j) R(:,j)./norm(R(:,j));
I_j =@(j,x,y,z) R(1,j)*x + R(2,j)*y + R(3,j)*z;

%create all possible permutations from 3 first sets of planes

PERMS_n = prod(k_length(1:3));
PERMS_J = zeros(PERMS_n,3);
cnt = 0;
for i1=1:k_length(1)
    for i2=1:k_length(2)
        for i3=1:k_length(3)
            cnt = cnt + 1;
            PERMS_J(cnt,:) = [i1,i2,i3];
        end
    end
end

fprintf('First intersection calculation: %i permutations of 3 planes\n',PERMS_n);

%3 first sets of planes
I = [1,2,3];

%start looping over all combinations

pinv_norm = zeros(PERMS_n,1);
intersection_line = zeros(3,PERMS_n);
parfor j=1:PERMS_n
    %intersection matrix for planes J from groups I

    J = PERMS_J(j,:);

    W_matrix = zeros(length(I),3);
    b_vector = zeros(length(I),1);
    for i=1:length(I)
        W_matrix(i,:) = nvec_j(I(i)).';
        b_vector(i) = -dot(nvec_j(I(i)),p0_jk(I(i),J(i)));
    end

    % solution_check
    Moore_Penrose_solution_check = W_matrix*pinv( W_matrix )*b_vector - b_vector;
    intersection_line(:,j) = pinv( W_matrix )*b_vector;
    pinv_norm(j) = norm(Moore_Penrose_solution_check);

end

intersection_line_norm = sqrt(sum(intersection_line.^2,1)).';

intersections3_inds = find(pinv_norm < MP_tol & intersection_line_norm <= 2 & intersection_line_norm ~= 0);
intersections3_n = length(intersections3_inds)
intersections3 =  PERMS_J(intersections3_inds,:);

SURVIVORS = zeros(Sn-2,1);
SURVIVORS(1) = intersections3_n;



intersections_n = intersections3_n;
intersections_inds = intersections3_inds;
intersections =  PERMS_J(intersections_inds,:);
intersections_integers = R.'*intersection_line(:,intersections_inds.');

%%%%% STEP 2

for ii=4:Sn



    PERMS_J_base = PERMS_J;

    %create all possible permutations from ii first sets of planes with only the
    %surviving set + all new
    %and recursibvly iterate

    PERMS_n = intersections_n*k_length(ii);
```

```
    fprintf('Starting plane intersections for new sensor %i of %i with %i permutations on %i remaining solutions\n',ii,Sn,PERMS_n, intersections_n);

    PERMS_J = zeros(PERMS_n,ii);
    cnt = 0;
    for iperm=1:intersections_n
        for i_add=1:k_length(ii)
            cnt = cnt + 1;
            PERMS_J(cnt,:) = [PERMS_J_base(intersections_inds(iperm),:), i_add];
        end
    end

    %4 first sets of planes
    I = [1:ii];

    %start looping over all combinations

    pinv_norm = zeros(PERMS_n,1);
    intersection_line = zeros(3,PERMS_n);
    parfor j=1:PERMS_n
        %intersection matrix for planes J from groups I

        J = PERMS_J(j,:);

        W_matrix = zeros(length(I),3);
        b_vector = zeros(length(I),1);
        for i=1:length(I)
            W_matrix(i,:) = nvec_j(I(i)).';
            b_vector(i) = -dot(nvec_j(I(i)),p0_jk(I(i),J(i)));
        end

        % solution_check
        Moore_Penrose_solution_check = W_matrix*pinv( W_matrix )*b_vector - b_vector;
        intersection_line(:,j) = pinv( W_matrix )*b_vector;
        pinv_norm(j) = norm(Moore_Penrose_solution_check);

    end

    intersections_inds = find(pinv_norm < MP_tol);
    intersections_n = length(intersections_inds);
    intersections =  PERMS_J(intersections_inds,:);
    intersections_integers = R.'*intersection_line(:,intersections_inds.');

    SURVIVORS(ii-2) = intersections_n;

end

intersections__last = intersections_n

intersections_integers_complete = [zeros(1,size(intersections_integers,2)); intersections_integers];

ambiguity_distances_INT_FORM_MAT = abs(intersections_integers_complete - round(intersections_integers_complete));
ambiguity_distances_INT_FORM_mean = mean(ambiguity_distances_INT_FORM_MAT,1)
ambiguity_distances_WAVE_FORM_MAT = exp(1i*2*pi*intersections_integers_complete) - exp(1i*2*pi*round(intersections_integers_complete));
ambiguity_distances_WAVE_FORM = sqrt(sum(ambiguity_distances_WAVE_FORM_MAT.*conj(ambiguity_distances_WAVE_FORM_MAT),1))

fign = 0;



el0 = 50;
az0 = 270;
k0x = sind(az0)*cosd(el0);
k0y = cosd(az0)*cosd(el0);
k0z = sind(el0);
k0 = [k0x; k0y; k0z];

cutoff_ph_ang = 90*pi/180;


%find all s-lines that intersect with the cap by range check
cap_intersections_of_slines = repmat([k0x; k0y],1,length(intersections_inds)) - intersection_line(1:2,intersections_inds);
cap_intersections_of_slines = sqrt(sum(cap_intersections_of_slines.^2,1));
cap_intersections_of_slines = cap_intersections_of_slines <= sin(cutoff_ph_ang);
cap_intersections_of_slines = find(cap_intersections_of_slines);

%from knowing what lines intersect with cap, find all possible DOA ambigs
%that are part of this

s_sel = intersection_line(1:3,intersections_inds(cap_intersections_of_slines));

k_finds = zeros(size(s_sel));
k_finds(1:2,:) = repmat([k0x; k0y],1,size(s_sel,2)) - s_sel(1:2,:);
k_finds(3,:) = sqrt(1- k_finds(1,:).^2 - k_finds(2,:).^2 );

SUBGROUP_signal=@(k) exp ( -1i*2*pi*( xy(:,1)*k(1) +...
                                      xy(:,2)*k(2) ) );



ambiguity_distances_EXPLICIT = zeros(1,size(k_finds,2));
```

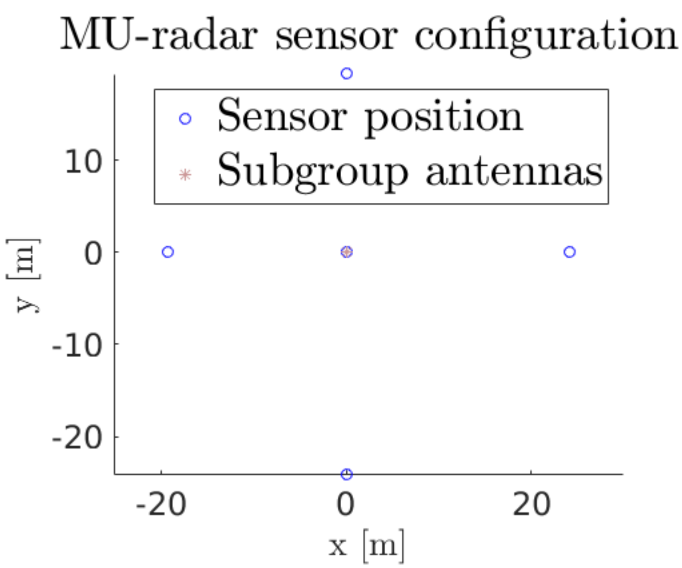
```
ambiguity_normal_EXPLICIT = zeros(size(xy,1),size(k_finds,2));
for i = 1:size(k_finds,2)
    ambiguity_distances_EXPLICIT(1,i) = norm(SUBGROUP_signal(k0) - SUBGROUP_signal(k_finds(:,i)));
    ambiguity_normal_EXPLICIT(:,i) = (SUBGROUP_signal(k0) - SUBGROUP_signal(k_finds(:,i)))/ambiguity_distances_EXPLICIT(1,i);
end
ambiguity_distances_EXPLICIT
ambiguity_normal_EXPLICIT


tick_font_size = 18;

fign = fign + 1; figure(fign); clf;
% set( gcf, 'Color', 'White', 'Unit', 'Normalized', ...
%     'Position', [0.1,0.1,0.8,0.6] ) ;
% - Build title axes and title.
% axes( 'Position', [0, 0.95, 1, 0.05] ) ;
% set( gca, 'Color', 'None', 'XColor', 'White', 'YColor', 'White' ) ;

subplot(1,1,1);
hold on
h = plot(xy(:,1)*lambda0,xy(:,2)*lambda0, 'ob');
for i=1:(size(xy,1))
plot(xant(i,:)*lambda0, yant(i,:)*lambda0, '*','color',[.8,.6,.6]);
if size(yant,2) > 1
K = convhull(xant(i,:)*lambda0, yant(i,:)*lambda0);
plot(xant(i,K)*lambda0, yant(i,K)*lambda0, '-k');
end
end
hold off
% axis('equal')
lh = legend('Sensor position','Subgroup antennas');
xh = xlabel('x [m]');
yh = ylabel('y [m]');
th = title('MU-radar sensor configuration');
set([xh,yh,th,lh],'Interpreter','latex','fontsize',axis_font_size+8)
axis equal

ax = ancestor(h, 'axes');
ax.XAxis.FontSize = tick_font_size;
ax.YAxis.FontSize = tick_font_size;


fign = fign + 1; figure(fign); clf;
plot(xant', yant', '*','color',[.8,.6,.6]);
hold on
plot(xy(:,1),xy(:,2), 'ob');
hold off
axis('equal')



fign = fign + 1; figure(fign); clf;
% plot(xant', yant', '*');
% hold on
plot(xy(:,1),xy(:,2), 'o');
% hold off
axis('equal')


fign = fign + 1; figure(fign); clf;
set( gcf, 'Color', 'White', 'Unit', 'Normalized', ...
    'Position', [0.1,0.1,0.8,0.6] ) ;
% - Build title axes and title.
axes( 'Position', [0, 0.95, 1, 0.05] ) ;
set( gca, 'Color', 'None', 'XColor', 'White', 'YColor', 'White' ) ;

subplot(1,1,1);
plot(3:Sn,SURVIVORS)
xh = xlabel('Number of sensors included');
yh = ylabel('Number of common plane intersections');
th = title('Intersection plane counts');
set([xh,yh,th],'Interpreter','latex','fontsize',axis_font_size)


fign = fign + 1; figure(fign); clf;
set( gcf, 'Color', 'White', 'Unit', 'Normalized', ...
    'Position', [0.1,0.1,0.8,0.6] ) ;
% - Build title axes and title.
axes( 'Position', [0, 0.95, 1, 0.05] ) ;
set( gca, 'Color', 'None', 'XColor', 'White', 'YColor', 'White' ) ;

subplot(1,1,1);
plot(intersection_line(1,intersections_inds),intersection_line(2,intersections_inds),'.b')
xh = xlabel('$s_{x}$ [1]');
yh = ylabel('$s_{y}$ [1]');
th = title('Intersection lines');
axis([-2,2,-2,2])
set([xh,yh,th],'Interpreter','latex','fontsize',axis_font_size)


% save('MUSIC_DOA_AMBIG_base.mat');
```

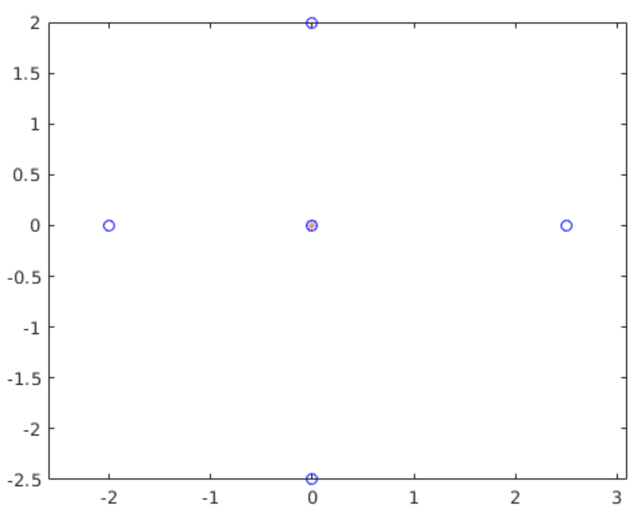
```
% inds_inside_circle = find(sqrt(sum(intersection_line(:,intersections_inds).^2,1)) < sind(18)*2.5)


fign = fign + 1; figure(fign); clf;
set( gcf, 'Color', 'White', 'Unit', 'Normalized', ...
    'Position', [0.1,0.1,0.8,0.6] ) ;
% - Build title axes and title.
axes( 'Position', [0, 0.95, 1, 0.05] ) ;
set( gca, 'Color', 'None', 'XColor', 'White', 'YColor', 'White' ) ;

subplot(1,1,1);
hold on
for S_ind = 1:length(intersections_inds)
    fprintf('Starting Sind %i of %i\n',S_ind,length(intersections_inds));
    % for S_ind = 3
    s_point = intersection_line(:,intersections_inds(S_ind));
    s_line = repmat(s_point,1,100);
    s_line(3,:) = linspace(-sqrt(4-dot(s_point,s_point)),sqrt(4-dot(s_point,s_point)),100);
    %     s_line(3,:) = linspace(0,sqrt(4-dot(s_point,s_point)),line_resolution);
    if S_ind == 25
        plot3(s_line(1,:),s_line(2,:),s_line(3,:),'.r')
    else
   h = plot3(s_line(1,:),s_line(2,:),s_line(3,:),'.b');
    end
end
hold off

xh = xlabel('$s_{x}$ [1]');
yh = ylabel('$s_{y}$ [1]');
zh = zlabel('$s_{z}$ [1]');
th = title('Solution set $\Omega$');
axis([-2,2,-2,2])
view([-143,53])
set([th],'Interpreter','latex','fontsize',axis_font_size+6)
set([xh,yh,zh],'Interpreter','latex','fontsize',axis_font_size+12)
ax = ancestor(h, 'axes');
ax.XAxis.FontSize = 19;
ax.YAxis.FontSize = 19;
ax.ZAxis.FontSize = 19;



fign = fign + 1; figure(fign); clf;
set( gcf, 'Color', 'White', 'Unit', 'Normalized', ...
    'Position', [0.1,0.1,0.8,0.6] ) ;
% - Build title axes and title.
axes( 'Position', [0, 0.95, 1, 0.05] ) ;
set( gca, 'Color', 'None', 'XColor', 'White', 'YColor', 'White' ) ;

subplot(1,1,1);
hold on
for S_ind = 1:length(intersections_inds)

    s_point = intersection_line(:,intersections_inds(S_ind));
   plot(s_point(1),s_point(2),'.b')
   text(s_point(1)+0.1,s_point(2),sprintf('%.2f',ambiguity_distances_WAVE_FORM(S_ind)));
end
hold off

xh = xlabel('$s_{x}$ [1]');
yh = ylabel('$s_{y}$ [1]');
th = title('Solution set $\Omega$');
axis([-2,2,-2,2])
set([xh,yh,th],'Interpreter','latex','fontsize',axis_font_size)



circ_cutoff_ph_ang_x = sin(cutoff_ph_ang)*cos(linspace(0,2*pi,100));
circ_cutoff_ph_ang_y = sin(cutoff_ph_ang)*sin(linspace(0,2*pi,100));

fign = fign + 1; figure(fign); clf;
set( gcf, 'Color', 'White', 'Unit', 'Normalized', ...
    'Position', [0.1,0.1,0.8,0.6] ) ;
% - Build title axes and title.
axes( 'Position', [0, 0.95, 1, 0.05] ) ;
set( gca, 'Color', 'None', 'XColor', 'White', 'YColor', 'White' ) ;

subplot(1,2,1);
hold on
for S_ind = 1:length(intersections_inds)

    s_point = intersection_line(:,intersections_inds(S_ind));
   plot(s_point(1),s_point(2),'.b')
   text(s_point(1)+0.1,s_point(2),sprintf('%.2f',ambiguity_distances_WAVE_FORM(S_ind)));
end
plot(k0x,k0y,'or')
plot(circ_cutoff_ph_ang_x ,circ_cutoff_ph_ang_y,'-r')
hold off

xh = xlabel('$s_{x}$ [1]');
```

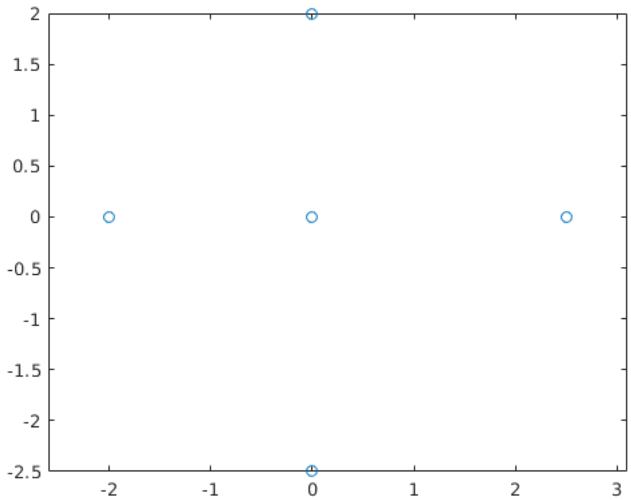
```
yh = ylabel('$s_{y}$ [1]');
th = title('Solution set $\Omega$');
axis([-2,2,-2,2])
set([xh,yh,th],'Interpreter','latex','fontsize',axis_font_size)

subplot(1,2,2);
hold on
plot(k0x,k0y,'or')
for i = 1:size(k_finds,2)
   plot(k_finds(1,i),k_finds(2,i),'.b')
   text(k_finds(1,i)+0.1,k_finds(2,i),sprintf('%.2f',ambiguity_distances_EXPLICIT(i)));
end
hold off

xh = xlabel('$k_{x}$ [1]');
yh = ylabel('$k_{y}$ [1]');
th = title('Explicit ambiguities $\Omega(\mathbf{k})$');
axis([-1,1,-1,1])
set([xh,yh,th],'Interpreter','latex','fontsize',axis_font_size)
```

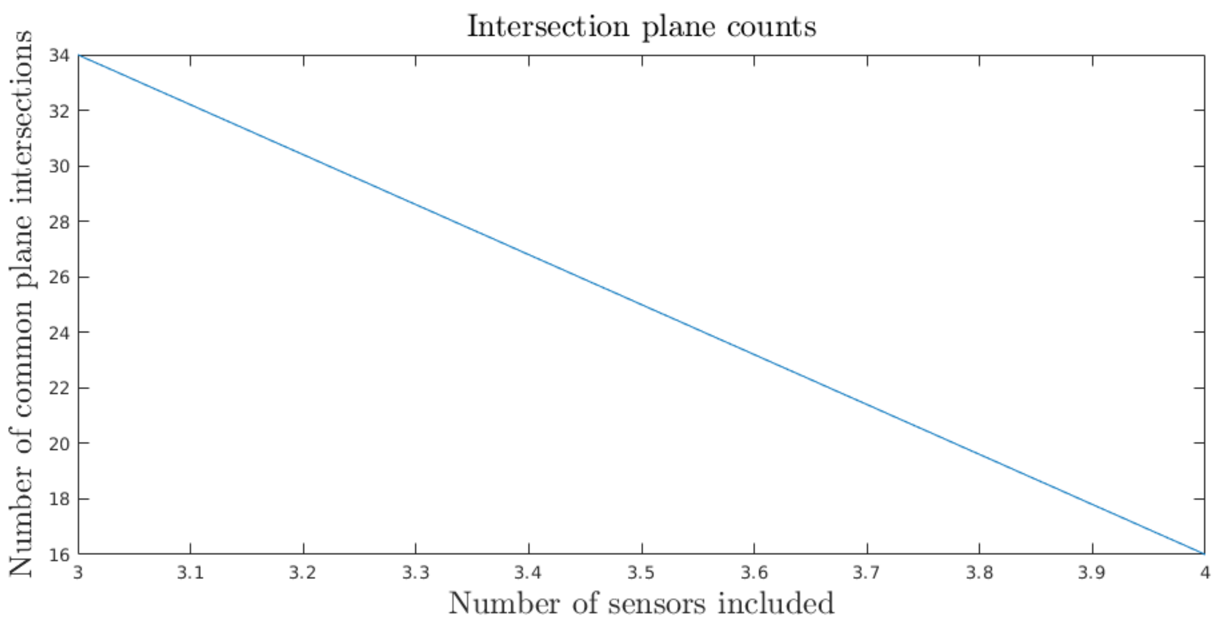
```
MP_tol =

    0.1000

First intersection calculation: 891 permutations of 3 planes
Starting parallel pool (parpool) using the 'local' profile ...
connected to 4 workers.

intersections3_n =

    34

Starting plane intersections for new sensor 4 of 4 with 374 permutations on 34 remaining solutions

intersections__last =

    16


ambiguity_distances_INT_FORM_mean =

  Columns 1 through 7

    0.0000    0.0450    0.0900    0.0450    0.0900    0.0000    0.0450

  Columns 8 through 14

    0.0450    0.0450    0.0450    0.0000    0.0900    0.0450    0.0900

  Columns 15 through 16

    0.0450    0.0000


ambiguity_distances_WAVE_FORM =

  Columns 1 through 7

    0.0000    0.9837    1.3912    0.9837    1.3912    0.0000    0.9837

  Columns 8 through 14

    0.9837    0.9837    0.9837    0.0000    1.3912    0.9837    1.3912

  Columns 15 through 16

    0.9837    0.0000


ambiguity_distances_EXPLICIT =

    0.9837    1.3912    0.9837    0.9837    0.9837    1.3912


ambiguity_normal_EXPLICIT =

  Columns 1 through 4

   0.1941 - 0.5975i   0.1373 - 0.4225i   0.0000 - 0.0000i   0.0000 - 0.0000i
   0.2977 - 0.7188i   0.2105 - 0.5083i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.3816 - 0.2275i   0.5396 - 0.3218i  -0.6257 - 0.0569i
   0.0000 + 0.0000i   0.1517 - 0.5288i   0.2146 - 0.7478i  -0.6805 + 0.3771i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 5 through 6

   0.1941 + 0.5975i   0.1373 + 0.4225i
   0.2977 + 0.7188i   0.2105 + 0.5083i
   0.0000 + 0.0000i   0.3816 - 0.2275i
   0.0000 + 0.0000i   0.1517 - 0.5288i
```

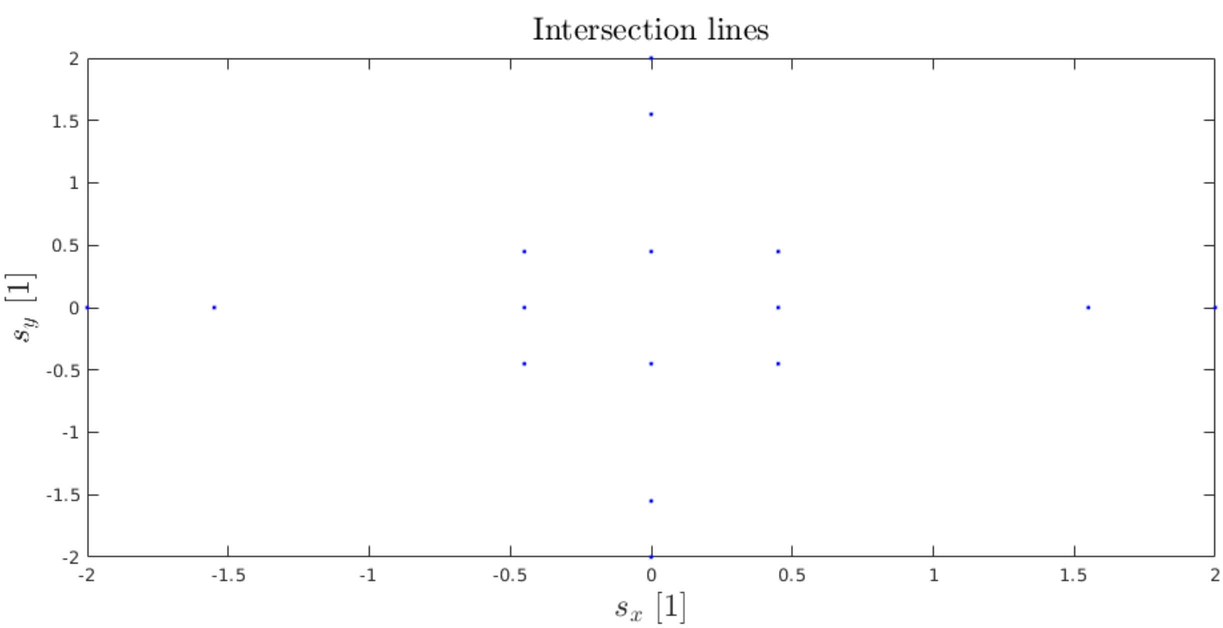
```
   0.0000 + 0.0000i   0.0000 + 0.0000i

Starting Sind 1 of 16
Warning: Imaginary parts of complex X, Y, and/or Z arguments ignored
Starting Sind 2 of 16
Starting Sind 3 of 16
Starting Sind 4 of 16
Starting Sind 5 of 16
Starting Sind 6 of 16
Starting Sind 7 of 16
Starting Sind 8 of 16
Starting Sind 9 of 16
Starting Sind 10 of 16
Starting Sind 11 of 16
Starting Sind 12 of 16
Starting Sind 13 of 16
Starting Sind 14 of 16
Starting Sind 15 of 16
Starting Sind 16 of 16
Warning: Imaginary parts of complex X, Y, and/or Z arguments ignored
```

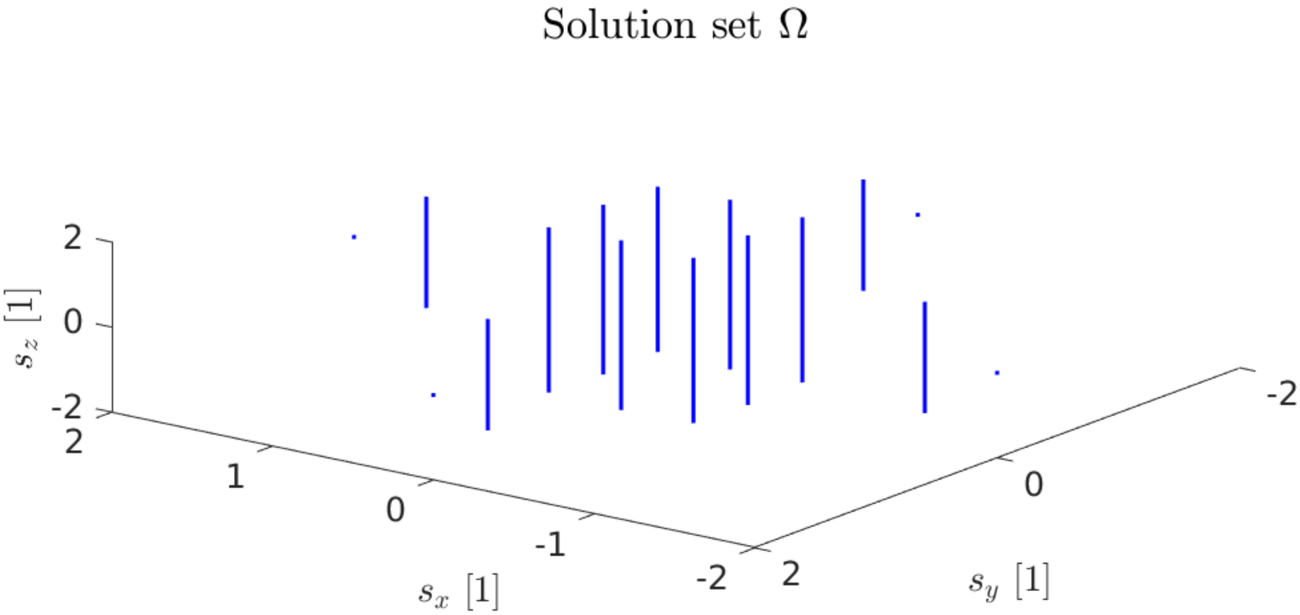


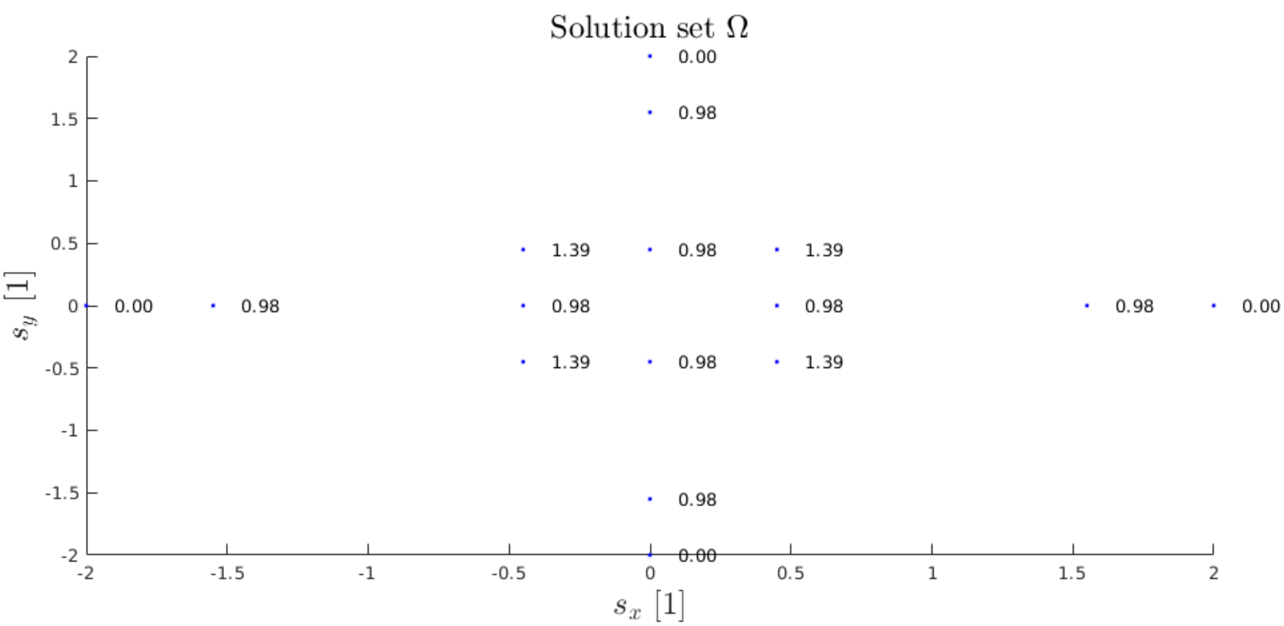


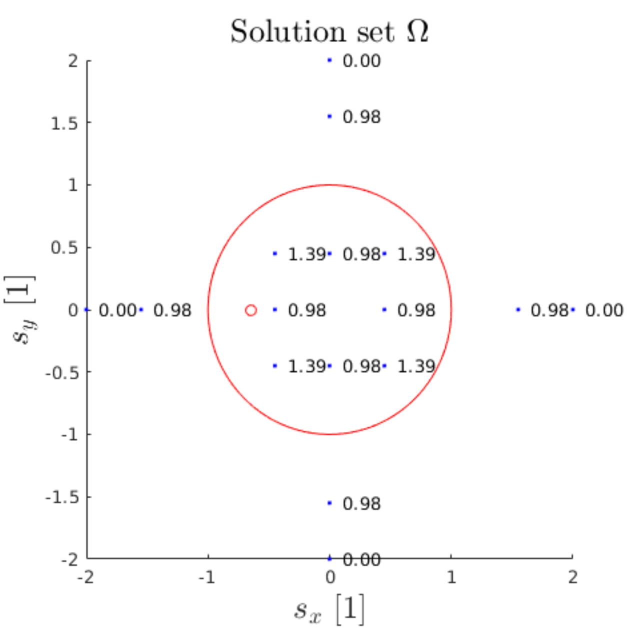

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2
3

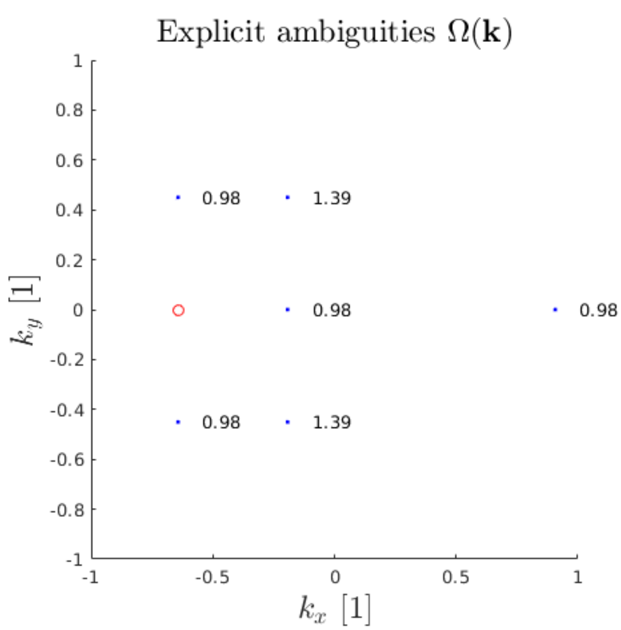

Intersection plane counts
Number of common plane intersections
Number of sensors included
34
32
30
28
26
24
22
20
18
16
3
3.1
3.2
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


Intersection lines
s_y [1]
s_x [1]
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Solution set Ω
s_z [1]
s_x [1]
s_y [1]
2
0
-2
2
1
0
-1
-2
2
0
-2


Solution set Ω
s_y [1]
s_x [1]
0.00
0.98
1.39
0.98
1.39
0.00
0.98
0.98
0.98
0.98
0.00
1.39
0.98
1.39
0.98
0.00


Solution set Ω
s_y [1]
s_x [1]
0.00
0.98
1.39
0.98
1.39
0.00
0.98
0.98
0.98
0.98
0.00
1.39
0.98
1.39
0.98
0.00


Explicit ambiguities Ω(**k**)
k_y [1]
k_x [1]
0.98
1.39
0.98
0.98
0.98
1.39

Published with MATLAB® R2018a