AEROSP- 584

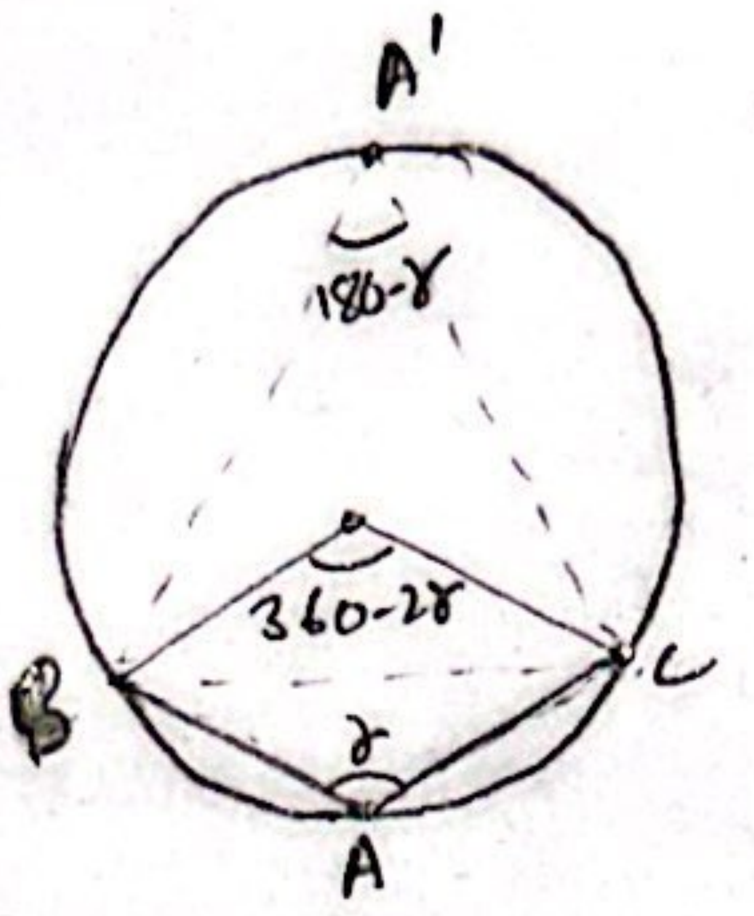
[illegible] Mole

Homework - 2

Que-1 We know that,

A unique circle can pass through any 3 given points.

Let A, B, C are points on the circle and 'r' is radius of circle



# Given:-

$\overline{BC} = a$

$\angle BAC = \gamma$ ($\gamma > 90^\circ$ but $\gamma < 180^\circ$)

Let us consider any point, A' on the circle on the upper portion of the chord

→ The quadrilateral formed by BA'CA is cyclic quadrilatera

$$\Rightarrow \quad \angle BAC + \angle BA'C = 180^\circ$$

$$\angle BA'C = 180 - \gamma$$

→ ∵ from the "double angle lemma taught in class

we get
$$\angle BOC = 2\angle BA'C$$
$$\Rightarrow \quad \angle BOC = 2(180-\gamma)$$
$$\boxed{\angle BOC = 360 - 2\gamma}$$

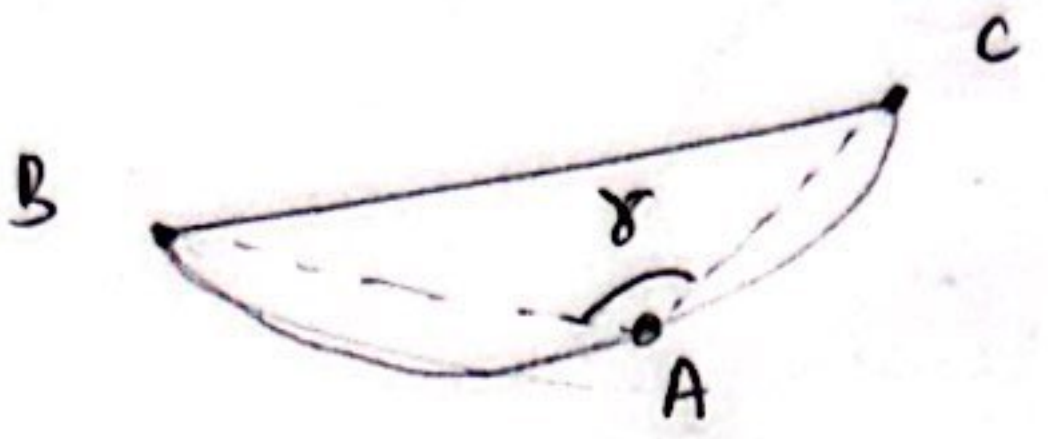
→ From using cosine rule on ΔBOC, we get,

$$BC^2 = OB^2 + OC^2 - 2(OB)(OC)\cos(\angle BOC)$$
$$a^2 = p^2 + p^2 - 2p^2\cos(360-2\gamma)$$
$$a^2 = 2p^2 - 2p^2\cos(2\gamma)$$
$$(\because \cos(360-2\gamma) = \cos(2\gamma))$$

$$p^2 = \frac{a^2}{2(1-\cos 2\gamma)} \quad , \quad \cos(2r) = 1 - 2\sin^2\gamma$$

$$p^2 = \frac{a^2}{4\sin^2\gamma}$$

$$\boxed{p = \left|\frac{a}{2\sin\gamma}\right|}$$



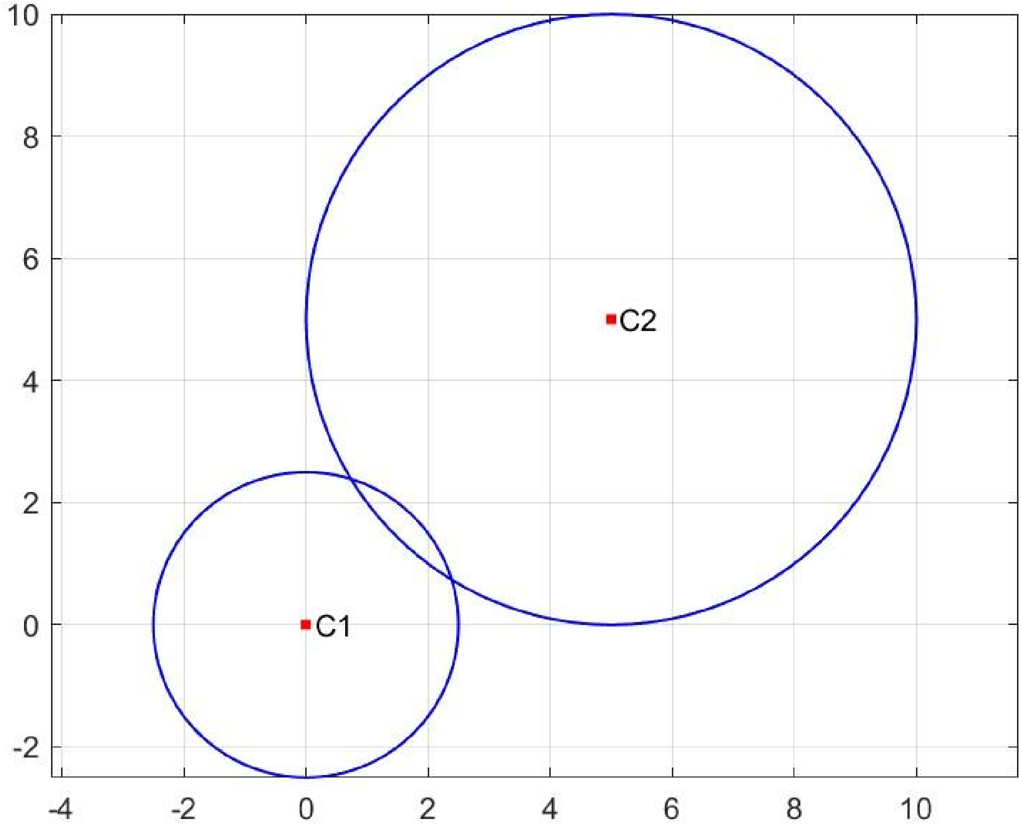
Furthermore, since we have not assumed anything other than $90 < \gamma < 180$, the point A can be anywhere in the short circular arc $\overset{\frown}{BAC}$.

Homework 2

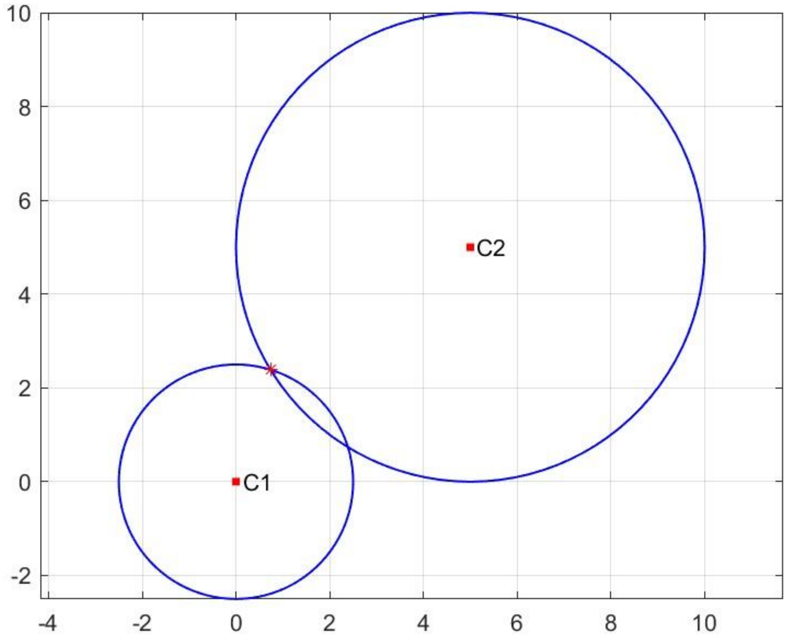
Problem 2

Part a:

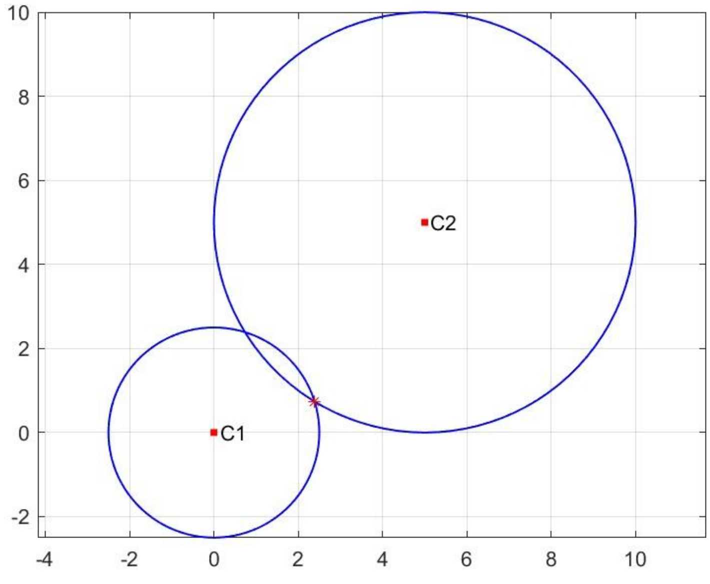
Visualization



Solution 1: Marked in red star, Initial guess = {10,0}



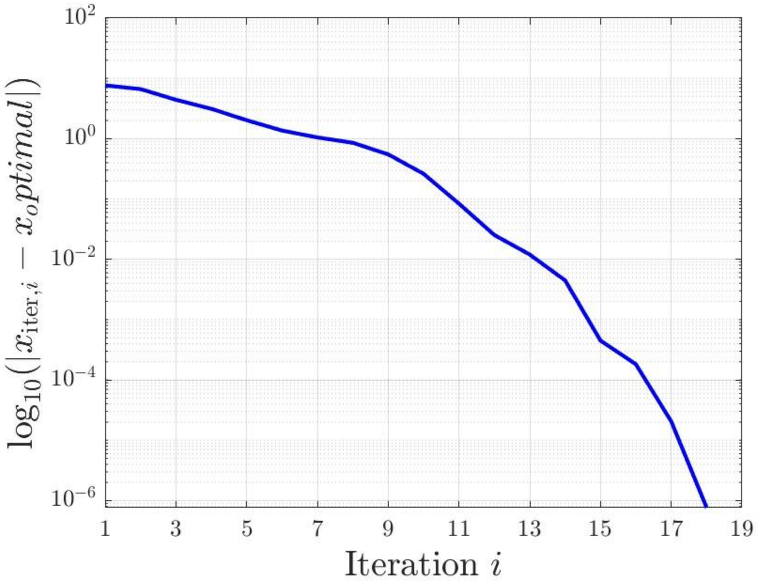
Solution 2: Marked in red star, Initial guess = {0,10}



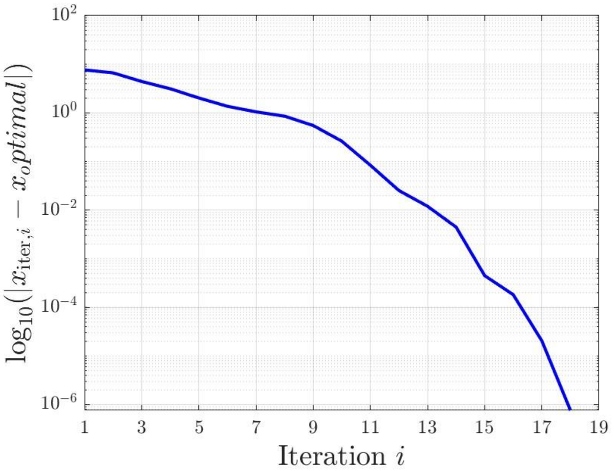
Part b:

Plot of position at each iteration:

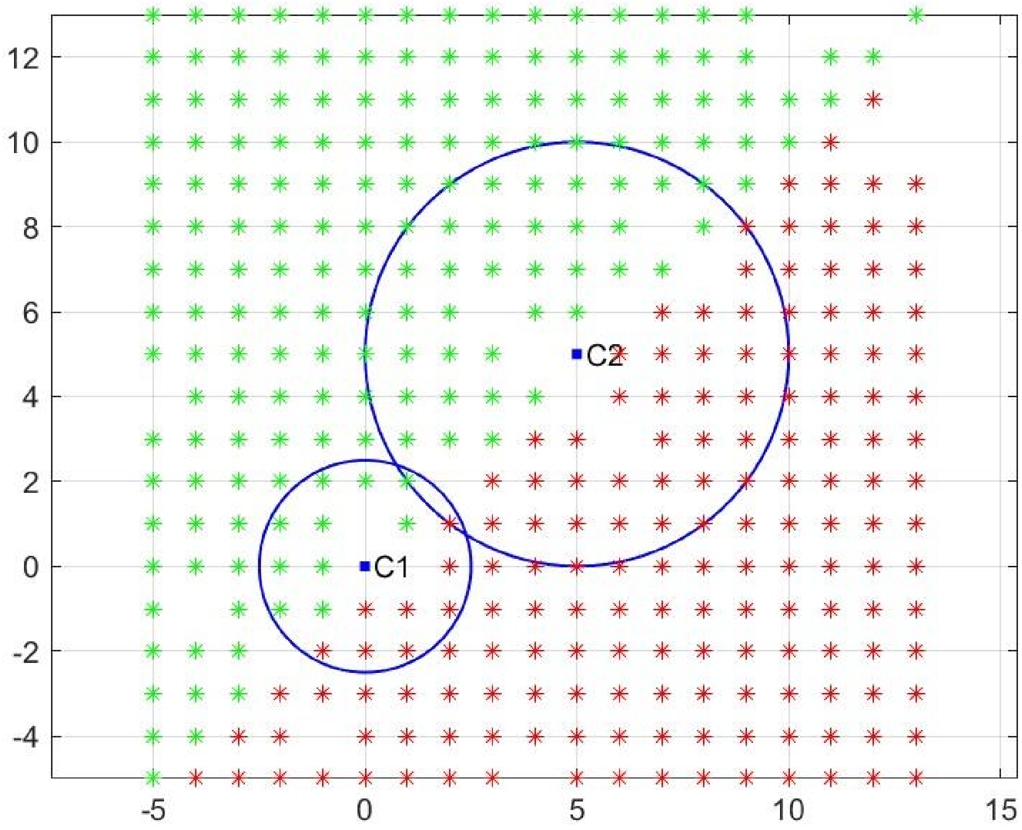
Initial guess: {0,10}



Initial guess: {10,0}



Part C: Green star is solution1, Red star is solution2



Code:

Prob_2:

```
%%% Problem 2
close all
clear all
clc
%% Initialise Paremeters
r1 = 2.5;
r2 = 5;
x1=0;
y1=0;
x2=5;
y2=5;
global h;
h.x = [];
h.fval = [];
%% Part (a)
x0 = [0,10];
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12, OutputFcn=@outfun);
[sol1,fval1] = fminunc(@find_norm,x0,options);
plot_2_a('../results/prob_2a_sol1',x1,y1,x2,y2,r1,r2,sol1(1),sol1(2));
plot_2_b('../results/plot_2b_sol1',sol1,h);
h.x = [];
h.fval = [];
x0 = [10,0];
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12, OutputFcn=@outfun);
[sol2,fval2] = fminunc(@find_norm,x0,options);
plot_2_a('../results/prob_2a_sol2',x1,y1,x2,y2,r1,r2,sol2(1),sol2(2))
plot_2_b('../results/plot_2b_sol2',sol2,h);
h.x = [];
h.fval = [];
%
x0 = [0,0];
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12);
% sol3 = fminunc(@find_norm,x0, options);
fig_plot = figure('visible','off');
th = 0:pi/50:2*pi;
xunit = r1 * cos(th) + x1;
yunit = r1 * sin(th) + y1;
plot(xunit, yunit, 'b', LineWidth=1);
hold on
th = 0:pi/50:2*pi;
xunit = r2 * cos(th) + x2;
yunit = r2 * sin(th) + y2;
plot(x1,y1,'b.', MarkerSize=10);
text(x1,y1,' C1');
plot(x2,y2,'b.', MarkerSize=10);
text(x2,y2,' C2')
plot(xunit, yunit, 'b', LineWidth=1);
name = '../results/plot_2_c';
for ii = -5:13
   for jj = -5:13
       jk = zeros(2,1);
       jk(1) = ii;
       jk(2) = jj;
       try
           [solmin,fval] = fminunc(@find_norm,jk,options);
           rr = [(solmin(1)-sol1(1)),(solmin(2)-sol1(2))];
           rr1 = sqrt(rr*rr');
           rr = [(solmin(1)-sol2(1)),(solmin(2)-sol2(2))];
           rr2 = sqrt(rr*rr');
           if(rr1<=rr2)
               plot(ii,jj,'g*')
           else
               plot(ii,jj,'r*')
           end
       catch
           fprintf('Inconsistent data in iteration %s, skipped.\n', ii,jj);
       end
   end
end
grid on
axis equal
hold off
saveas(fig_plot,name,'jpg');
```

Prob_2_a.m

```
function plot_2_a(name,x1,y1,x2,y2,r1,r2,x,y)
f = figure('visible','off');
% f = figure;
th = 0:pi/50:2*pi;
xunit = r1 * cos(th) + x1;
yunit = r1 * sin(th) + y1;
plot(xunit, yunit, 'b', LineWidth=1);
hold on
th = 0:pi/50:2*pi;
xunit = r2 * cos(th) + x2;
yunit = r2 * sin(th) + y2;
plot(x1,y1,'r.', MarkerSize=10);
text(x1,y1,' C1');
plot(x2,y2,'r.', MarkerSize=10);
text(x2,y2,' C2')
plot(xunit, yunit, 'b', LineWidth=1);
plot(x,y,'r*');
grid on
axis equal
hold off
saveas(f,name,'jpg');
end
```

Prob_2_b.m

```
function plot_2_b(file_name,sol_needed,h)
required_plot = ones(length(h.x),1);
iter = ones(length(h.x),1);
for i = 1:length(h.x)
   iter(i) = i;
   r = ones(2);
   r(1) = h.x(i,1) - sol_needed(1);
   r(2) = h.x(i,2) - sol_needed(2);
   required_plot(i) = sqrt((r(1)*r(1)) + (r(2)*r(2)));
end
plot_2bx = figure(2);
xx = required_plot;
semilogy(iter, xx,'b','linewidth',2);
xlim([min(iter) max(iter)])
xticks(1:2:max(iter))
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('Iteration $i$','fontsize',18,'interpreter','latex')
ylabel('$\log_{10} (|x_{{\rm iter},i} -
x_optimal|)$','fontsize',18,'interpreter','latex')
grid on
saveas(plot_2bx,file_name,"jpg")
end
```

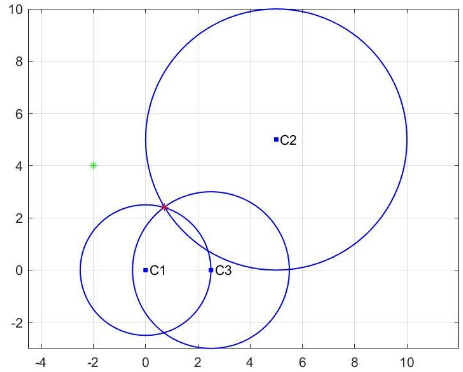
```
function stop = outfun(x,optimValues,state)
    global h
    stop = false;
    switch state
        case 'iter'
            h.fval = [h.fval; optimValues.fval];
            h.x = [h.x; x];
        otherwise
    end
end

function norm = find_norm(x0)
x = x0(1);
y = x0(2);
r1 = 2.5;
r2 = 5;
x1=0;
y1=0;
x2=5;
```

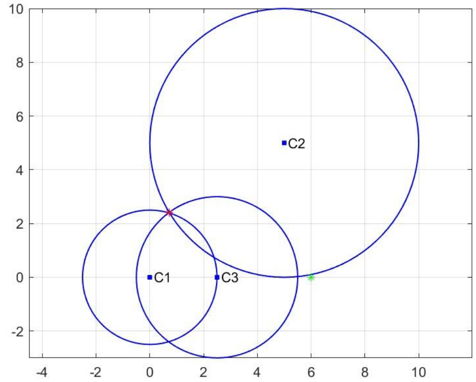
```
y2=5;
r = [((x-x1)^2)+((y-y1)^2)-(r1^2),((x-x2)^2)+((y-y2)^2)-(r2^2)]';
% storer.x = [storer.x,x0];
% storer.fx = [storer.fx,sqrt(r'*r)];
norm = r(1)^2 + r(2)^2;
```

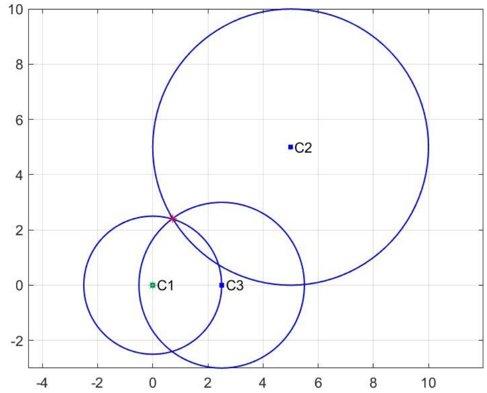
Problem 3:

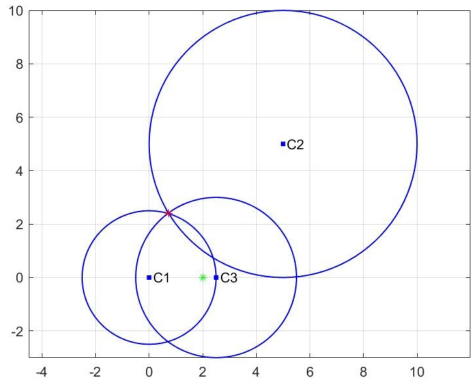
Part a

Green star is Initial guess, Red star is converged solution



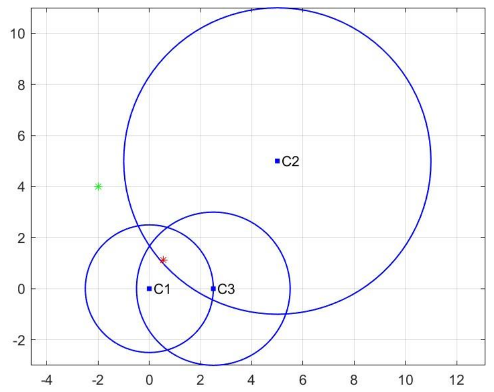


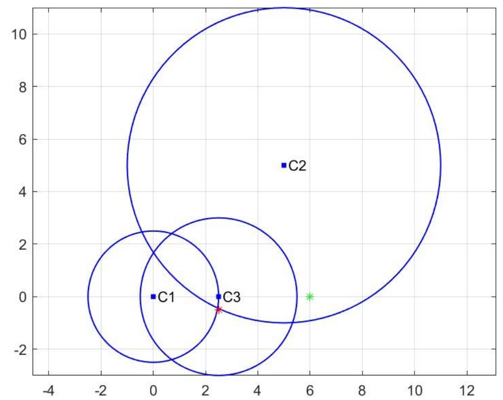


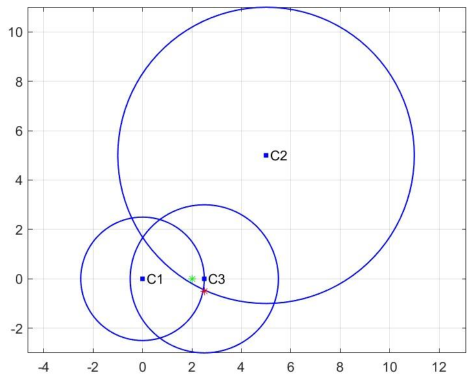


Part b:

Green star is Initial guess, Red star is converged solution

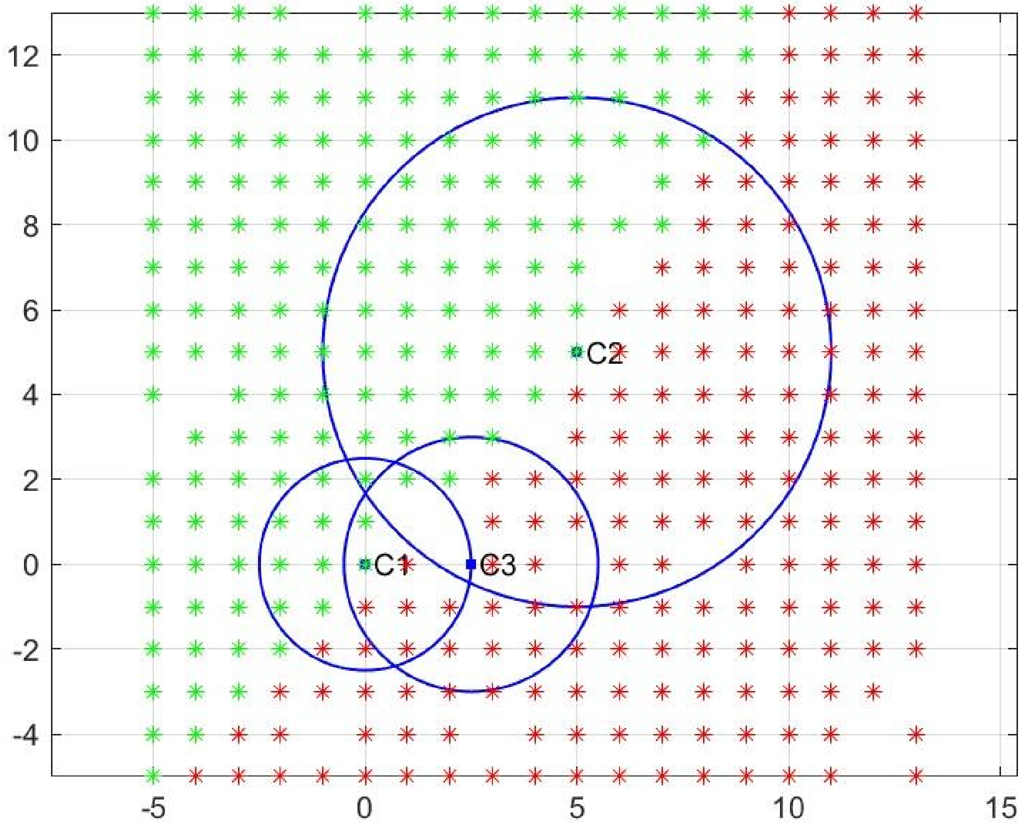






Part C:

Green is sol1 in the above case and red is soln2.



Prob_3_a.m

```
%%% Problem 3
close all
clear all
clc
%% Initialise Paremeters
r1 = 2.5;
r2 = 5;
r3 = 3;
x1=0;
y1=0;
x2=5;
y2=5;
x3=2.5;
y3=0;
xx=0.7212;
yy=2.4080;
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12);
for i=0:5
   fig_plot = figure('visible','off');
   th = 0:pi/50:2*pi;
   xunit = r1 * cos(th) + x1;
   yunit = r1 * sin(th) + y1;
   plot(xunit, yunit, 'b', LineWidth=1);
   hold on

   xunit = r2 * cos(th) + x2;
   yunit = r2 * sin(th) + y2;
   plot(xunit, yunit, 'b', LineWidth=1);
   xunit = r3 * cos(th) + x3;
   yunit = r3 * sin(th) + y3;
   plot(xunit, yunit, 'b', LineWidth=1);

   plot(x1,y1,'b.', MarkerSize=10);
   text(x1,y1,' C1');
   plot(x2,y2,'b.', MarkerSize=10);
   text(x2,y2,' C2')
   plot(x3,y3,'b.', MarkerSize=10);
   text(x3,y3,' C3')
   if(i==6)
       name = '../../results/plot_3_c_1';

       for ii = -5:13
           for jj = -5:13
               try
                   [solmin,fval] = fminunc(@find_norm_3a,[ii,jj]',options);
                   rr = [(solmin(1)-sol1(1)),(solmin(2)-sol1(2))];
                   rr1 = sqrt(rr*rr');
                   rr = [(solmin(1)-sol2(1)),(solmin(2)-sol2(2))];
                   rr2 = sqrt(rr*rr');
                   if(rr1<=rr2)
                       plot(ii,jj,'g*')
                   else
                       plot(ii,jj,'r*')
                   end
               catch
                   fprintf('Inconsistent data in iteration %s, skipped.\n',
ii,jj);
               end
           end
       end
   elseif(i==0)
       name = '../../results/plot_3_a_1';
       [sol1,fval] = fminunc(@find_norm_3a,[-2,4],options);
       plot(-2, 4,'g*')
       plot(sol1(1), sol1(2),'r*')
   elseif(i==1)
       name = '../../results/plot_3_a_2';
       jk = zeros(2,1);
       jk(1) = 6;
       jk(2) = 0;
       [sol2,fval] = fminunc(@find_norm_3a,jk,options);
       plot(6,0,'g*')
       plot(sol2(1), sol2(2),'r*')
   else
       name = '../../results/plot_3_a_4';

       [sol4,fval] = fminunc(@find_norm_3a,[2,0],options);
       plot(2,0,'g*')
       plot(sol4(1), sol4(2),'r*')
   end

   grid on
   axis equal
   hold off
   saveas(fig_plot,name,'jpg');
end
```

Find_norm_3a.m

```
function norm = find_norm_3a(x0)
x = x0(1);
y = x0(2);
r1 = 2.5;
r2 = 5;
r3 = 3;
x1=0;
y1=0;
```

```
x2=5;
y2=5;
x3=2.5;
y3=0;
r = [(x-x1)^2+(y-y1)^2-r1^2,(x-x2)^2+(y-y2)^2-r2^2, (x-x3)^2+(y-y3)^2-r3^2]';
norm = r(1)^2 + r(2)^2 + r(3)^2;
```

Prob_3b.m

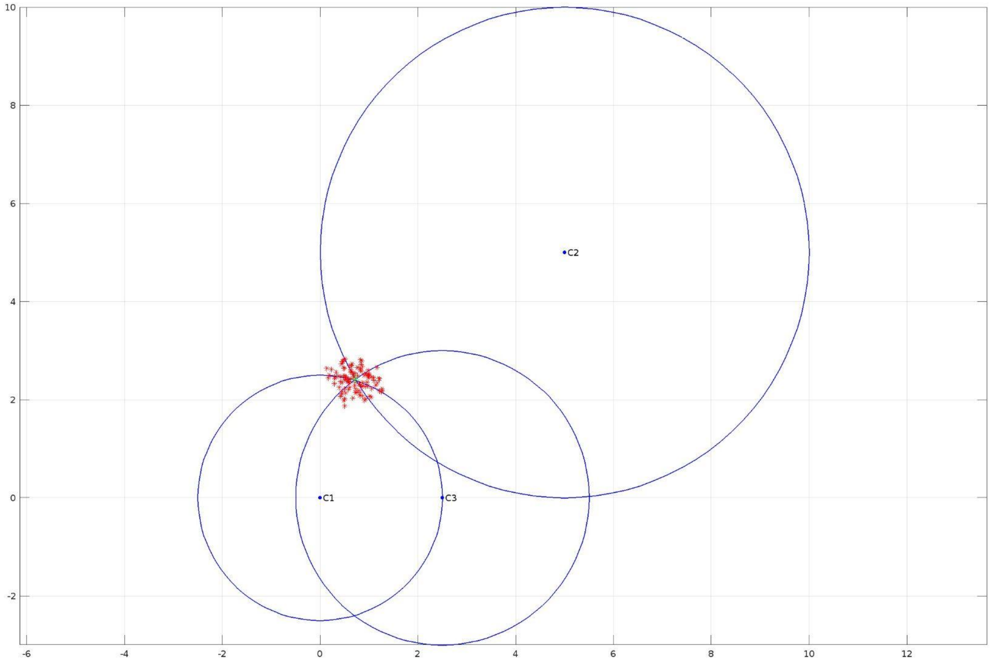
```
%%% Problem 3
close all
clear all
clc
%% Initialise Paremeters
r1 = 2.5;
r2 = 6;
r3 = 3;
x1=0;
y1=0;
x2=5;
y2=5;
x3=2.5;
y3=0;
xx=-2;
yy=4;
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12);
for i=0:5
   fig_plot = figure('visible','on');
   th = 0:pi/50:2*pi;
   xunit = r1 * cos(th) + x1;
   yunit = r1 * sin(th) + y1;
   plot(xunit, yunit, 'b', LineWidth=1);
   hold on

   xunit = r2 * cos(th) + x2;
   yunit = r2 * sin(th) + y2;
   plot(xunit, yunit, 'b', LineWidth=1);
   xunit = r3 * cos(th) + x3;
   yunit = r3 * sin(th) + y3;
   plot(xunit, yunit, 'b', LineWidth=1);

   plot(x1,y1,'b.', MarkerSize=10);
   text(x1,y1,' C1');
   plot(x2,y2,'b.', MarkerSize=10);
   text(x2,y2,' C2')
   plot(x3,y3,'b.', MarkerSize=10);
   text(x3,y3,' C3')
   if(i==5)
       name = '../../results/plot_3_c_1';
       for ii = -5:13
           for jj = -5:13
               try
                   [solmin,fval] = fminunc(@find_norm_3b,[ii,jj]',options);
                   rr = [(solmin(1)-sol1(1)),(solmin(2)-sol1(2))];
                   rr1 = sqrt(rr*rr');
                   rr = [(solmin(1)-sol2(1)),(solmin(2)-sol2(2))];
                   rr2 = sqrt(rr*rr');
                   if(rr1<rr2)
                       plot(ii,jj,'g*')
                   else
                       plot(ii,jj,'r*')
                   end
               catch
                   fprintf('Inconsistent data in iteration %s, skipped.\n',
ii,jj);
               end
           end
       end
   elseif(i==0)
       name = '../../results/plot_3_b_1';
       [sol1,fval] = fminunc(@find_norm_3b,[xx,yy],options);
       plot(xx,yy,'g*')
       plot(sol1(1), sol1(2),'r*')
   elseif(i==1)
       name = '../../results/plot_3_b_2';
       jk = zeros(2,1);
       jk(1) = 6;
       jk(2) = 0;
       [sol2,fval] = fminunc(@find_norm_3b,jk,options);
       plot(6,0,'g*')
       plot(sol2(1), sol2(2),'r*')
   elseif(i==2)
       name = '../../results/plot_3_b_3';
       [sol3,fval] = fminunc(@find_norm_3b,[0,0],options);
       plot(0,0,'g*')
       plot(sol3(1), sol3(2),'r*')
   end

   grid on
   axis equal

   saveas(fig_plot,name,'jpg');
   hold off
end
```

Find_norm_3b.m

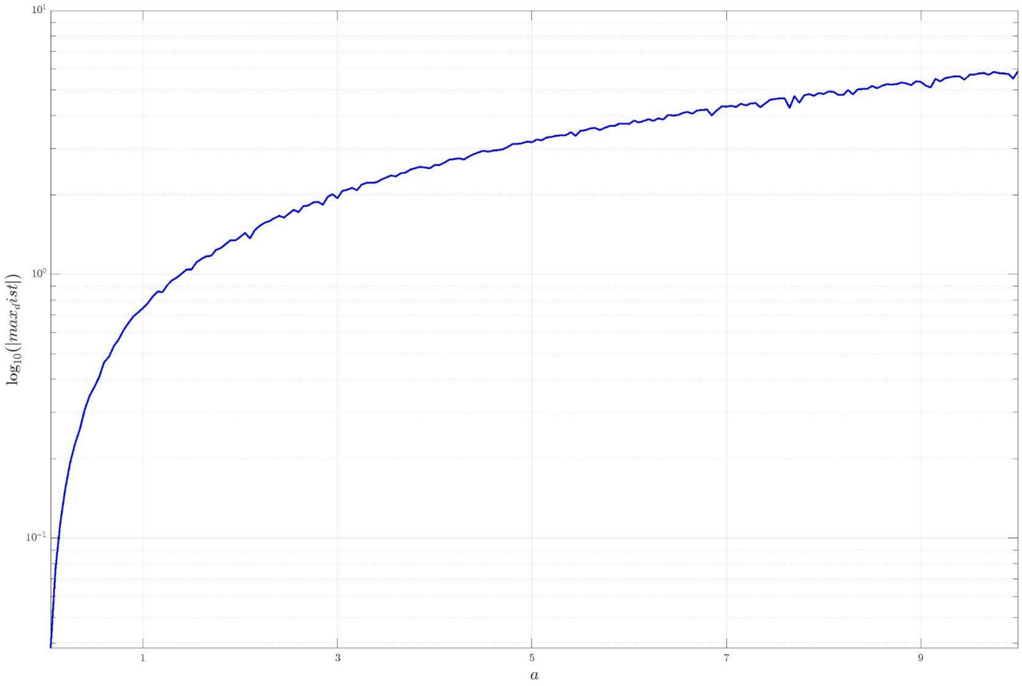
```
function norm = find_norm_3b(x0)
```

```
x = x0(1);
y = x0(2);
r1 = 2.5;
r2 = 6;
r3 = 2;
x1=0;
y1=0;
x2=5;
y2=5;
x3=2.5;
y3=0;
r = [(x-x1)^2+(y-y1)^2-r1^2,(x-x2)^2+(y-y2)^2-r2^2, (x-x3)^2+(y-y3)^2-r3^2]';
norm = r(1)^2 + r(2)^2 + r(3)^2;
```

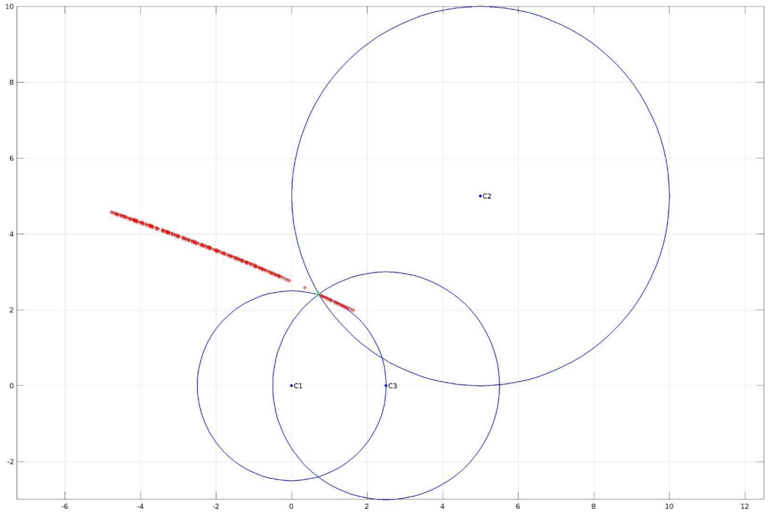
Problem 4:
Part a



Part b:



10^{1}
10^{0}
10^{-1}
$\log_{10}(|max_dist|)$
1
3
5
7
9
a



10
8
6
4
2
0
-2
-6
-4
-2
0
2
4
6
8
10
12
C1
C2
C3

Code:

```
%%% Problem 3
close all
clear all
clc
%% Initialise Paremeters
global r1 r2 r3;
r1 = 2.5;
r2 = 5;
r3 = 3;
x1=0;
y1=0;
x2=5;
y2=5;
x3=2.5;
y3=0;
fig_plot = figure('visible','off', 'Position', get(0, 'Screensize'));
th = 0:pi/50:2*pi;
xunit = r1 * cos(th) + x1;
yunit = r1 * sin(th) + y1;
plot(xunit, yunit, 'b', LineWidth=1);
hold on
xunit = r2 * cos(th) + x2;
yunit = r2 * sin(th) + y2;
plot(xunit, yunit, 'b', LineWidth=1);
xunit = r3 * cos(th) + x3;
yunit = r3 * sin(th) + y3;
plot(xunit, yunit, 'b', LineWidth=1);
plot(x1,y1,'b.', MarkerSize=10);
text(x1,y1,' C1');
plot(x2,y2,'b.', MarkerSize=10);
text(x2,y2,' C2')
plot(x3,y3,'b.', MarkerSize=10);
text(x3,y3,' C3')
jk = ones(2,1);
jk(1) = -4.0;
jk(2) = -4.0;
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12);
[solorg,fval] = fminunc(@find_norm,jk, options);
name = "./results/prob_4_a";
for i=0:100
  try
       r1 = 2.5 + (rand(1)-0.5);
       r2 = 5 + (rand(1)-0.5);
       r3 = 3 + (rand(1)-0.5);
       [solmin,fval] = fminunc(@find_norm,[-4.0,-4.0],options);
       plot(solmin(1), solmin(2),'r*')
   catch
```

```
       fprintf('Inconsistent data in iteration skipped.\n');
  end
end
plot(solorg(1),solorg(2),'g*', MarkerSize=10);

grid on
axis equal
hold off
saveas(fig_plot,name,'jpg');
```

prob_4b.m

```
%%% Problem 3
close all
clear all
clc
%% Initialise Paremeters
global r1 r2 r3;
r1 = 2.5;
r2 = 5;
r3 = 3;
x1=0;
y1=0;
x2=5;
y2=5;
x3=2.5;
y3=0;
fig_plot = figure('visible','off', 'Position', get(0, 'Screensize'));
th = 0:pi/50:2*pi;
xunit = r1 * cos(th) + x1;
yunit = r1 * sin(th) + y1;
plot(xunit, yunit, 'b', LineWidth=1);
hold on
xunit = r2 * cos(th) + x2;
yunit = r2 * sin(th) + y2;
plot(xunit, yunit, 'b', LineWidth=1);
xunit = r3 * cos(th) + x3;
yunit = r3 * sin(th) + y3;
plot(xunit, yunit, 'b', LineWidth=1);
plot(x1,y1,'b.', MarkerSize=10);
text(x1,y1,' C1');
plot(x2,y2,'b.', MarkerSize=10);
text(x2,y2,' C2')
plot(x3,y3,'b.', MarkerSize=10);
text(x3,y3,' C3')
jk = ones(2,1);
jk(1) = -4.0;
jk(2) = -4.0;
```

```
options = optimoptions(@fminunc,OptimalityTolerance = 1e-12);
[solorg,fval] = fminunc(@find_norm,jk, options);
name = "./results/prob_4_b";
a = 0.05:0.05:10;
b = 0.05:0.05:10;
b1 = 0.05:0.05:10;
b2 = 0.05:0.05:10;
for jj  = 1:length(a)
   b(jj) = 0;
   for i=0:100
      try
           k = (rand(1)-0.5);
           r1 = 2.5 + a(jj)*k;
           r2 = 5 + a(jj)*k;
           r3 = 3 + a(jj)*k;
           [solmin,fval] = fminunc(@find_norm,[-4.0,-4.0],options);
           err = sqrt((solmin(1)-solorg(1))^2 + (solmin(2)-solorg(2))^2);

           if(b(jj)<err)
               solmax(1) = solmin(1);
               solmax(2) = solmin(2);
               b(jj) = max([b(jj),err]);
           end

       catch
           fprintf('Inconsistent data in iteration skipped.\n');
      end
   end
   plot(solmax(1), solmax(2),'r*')
%     b1(jj) = solmax(1);
%     b2(jj) = solmax(2);
end
plot(solorg(1),solorg(2),'g*', MarkerSize=10);
grid on
axis equal
hold off
saveas(fig_plot,name,'jpg');
%%
name = "./results/4_b_plot";
plot_2bx = figure('visible','off', 'Position', get(0, 'Screensize'));
iter = a;
xx = b;
semilogy(iter, xx,'b','linewidth',2);
xlim([min(iter) max(iter)])
xticks(1:2:max(iter))
set(gca,'FontSize',12)
set(gca,'TickLabelInterpreter','latex');
xlabel('$a$','fontsize',18,'interpreter','latex')
ylabel('$\log_{10} (|max_dist|)$','fontsize',18,'interpreter','latex')
```

```
grid on
saveas(plot_2bx,name,"jpg")
```

Find_norm.m

```
function norm = find_norm(x0)
% fprintf("%s\n",x0);
x = x0(1);
y = x0(2);
x1=0;
y1=0;
x2=5;
y2=5;
x3=2.5;
y3=0;
global r1 r2 r3;
r = [(x-x1)^2+(y-y1)^2-r1^2,(x-x2)^2+(y-y2)^2-r2^2, (x-x3)^2+(y-y3)^2-r3^2]';
norm = r(1)^2 + r(2)^2 + r(3)^2;
```