ECE 172A
HW2
Ruomei Ye
A99074215

Problem 1
(i).
1. Where are the obstacles in the room?

The obstacles are at (60,50) and (10,40).

2. Why do the obstacles look like they do on the potential field?

This is because at the obstacles, they are “steeper” and therefore the gradients become larger. Since the potential field reveals the rate of change, it is strong at the obstacles.

3. What happens to the gradient as you approach the end location?

The gradients is closer to 0 as approaching the end.

(ii).

1. Why is this method better than the sense-act paradigm?

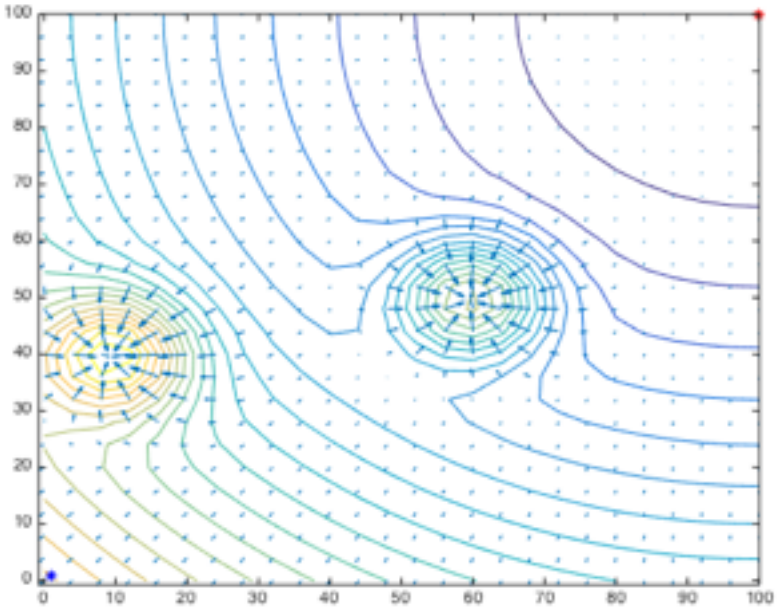
This is because using gradient descent algorithm the bot knows the direction it is aiming at, and therefore will move toward that direction (where gradient is 0). When using sense-act paradigm, it is not designed to move toward to the final destination and there is no attractor for it.

2. Explain what the algorithm is doing and why it works to get the bot across the room while avoiding obstacles.

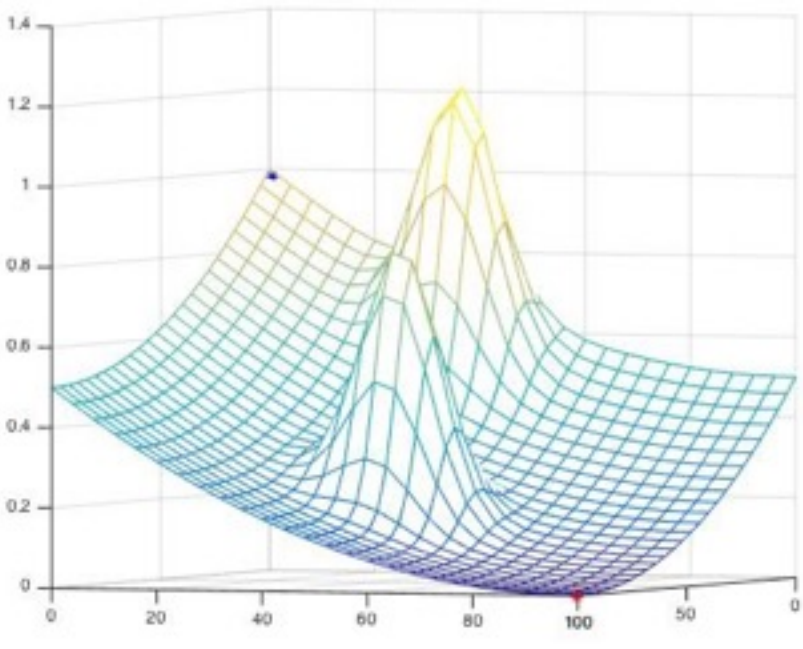
The gradient descent algorithm allows the bot to move the position with lower gradient; since the destination’s gradient is 0, the bot will finally be attracted to the end. Since the gradients of obstacles become higher, the bot is repulsed from obstacles. Also, as the bot becomes closer to end, the scale of its move is smaller and this is also achieved by the algorithm.

Plots:

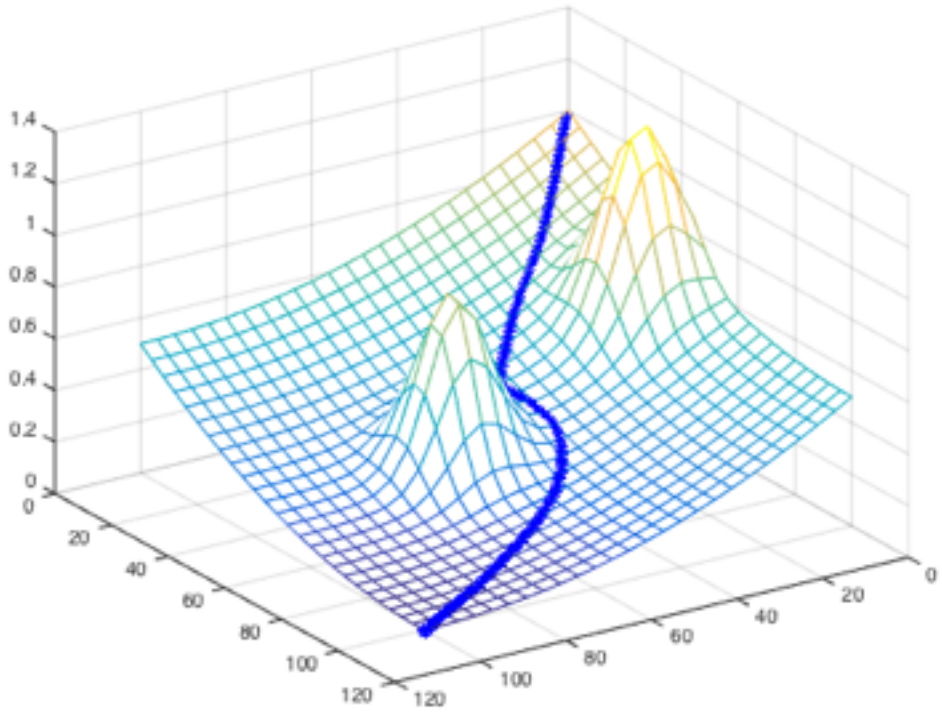
1. contour:



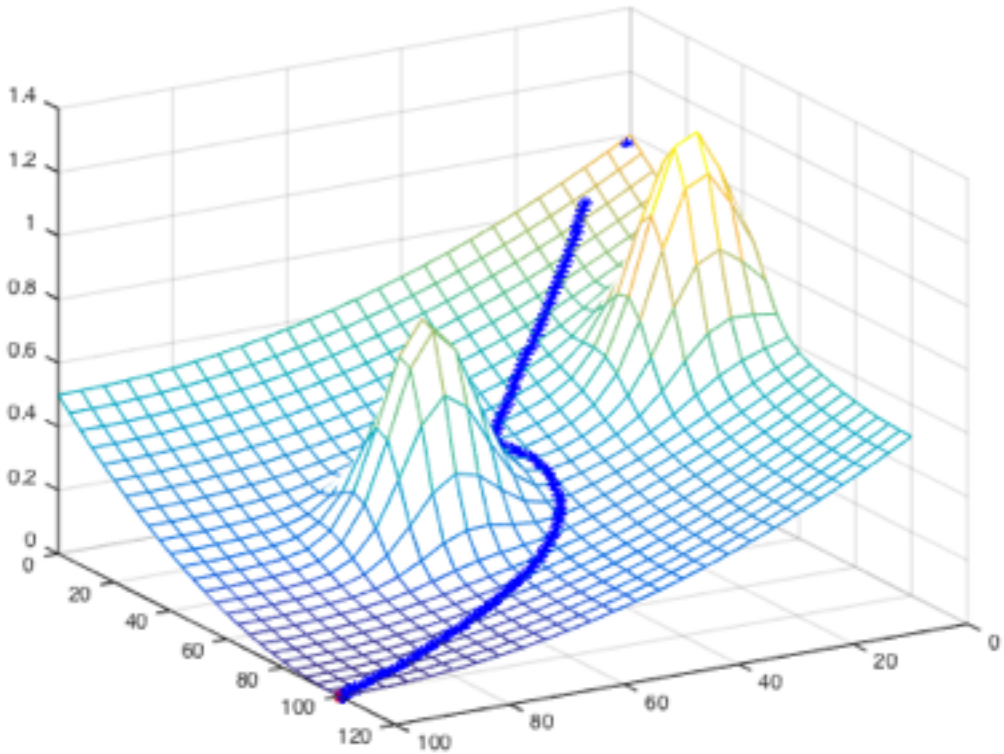

2. mesh:



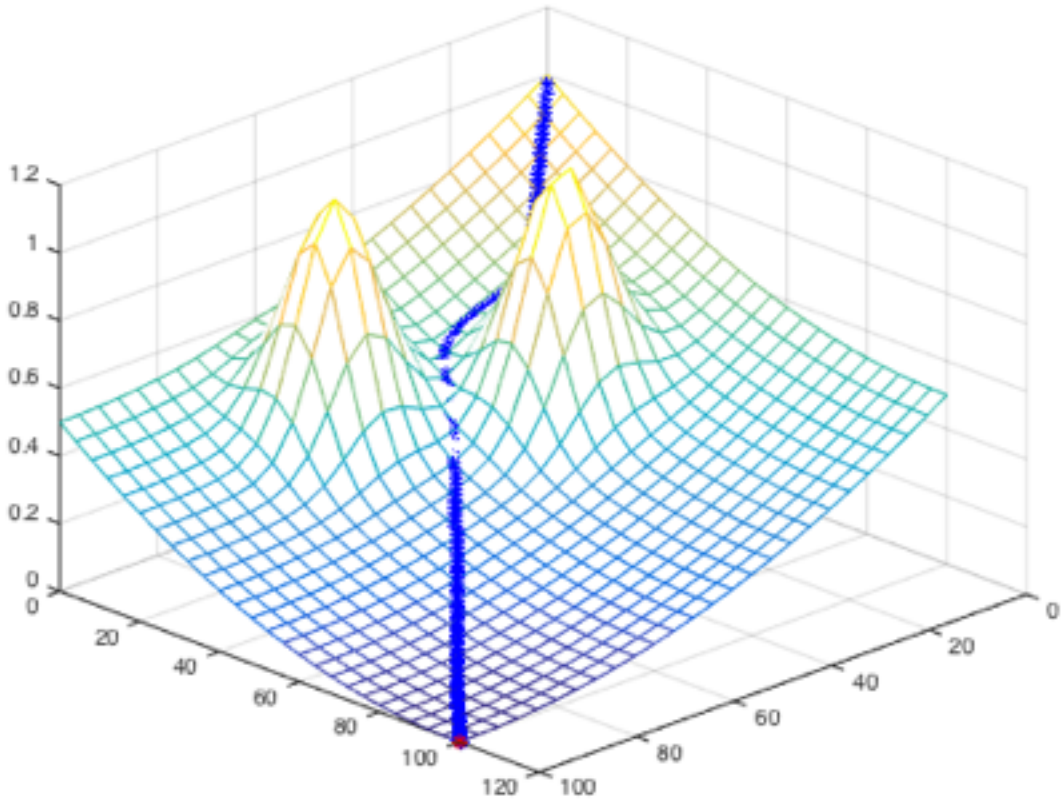
3. (1).




(2). change the initial location to (10.5)



(3). change the obstacles to (30,40),(60,20)



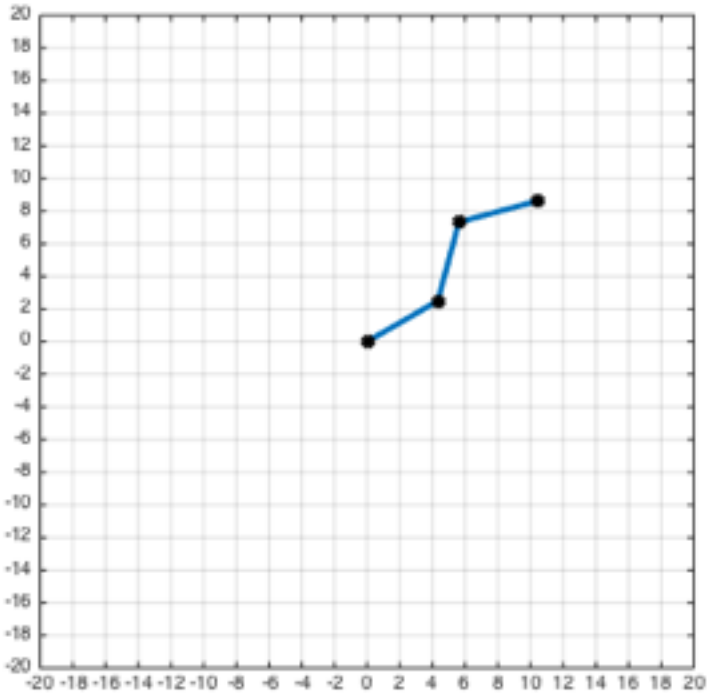

Problem 2

Please see the appendix.

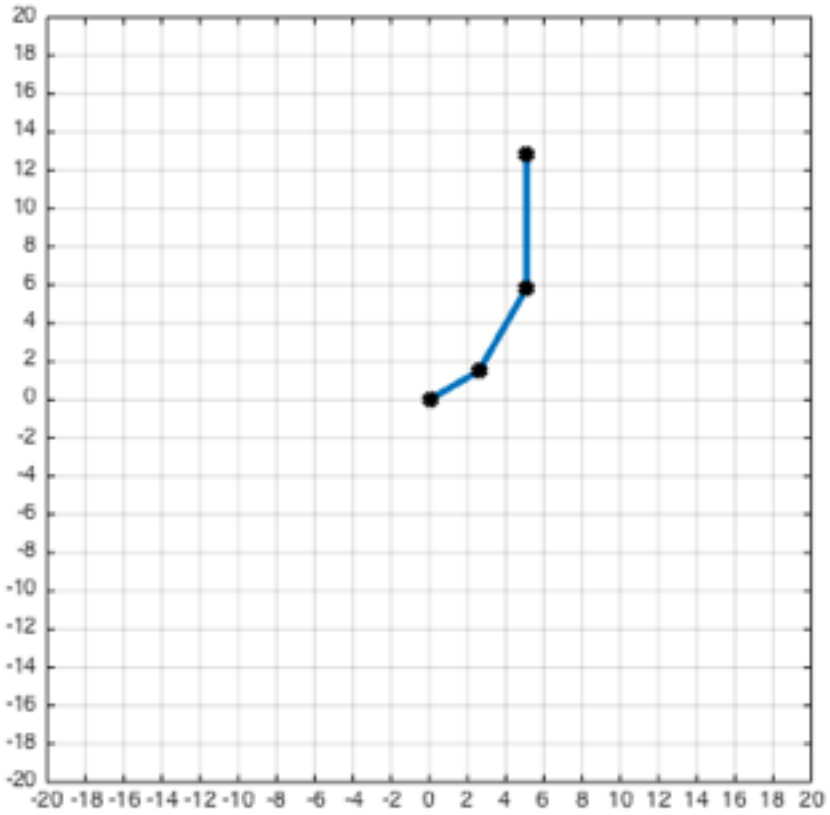
Problem 3

(i). forward:

(a) $\theta_0 = \pi/6, \theta_1 = \pi/4, \theta_2 = -\pi/3, l_1 = 5, l_2 = 5, l_3 = 5$

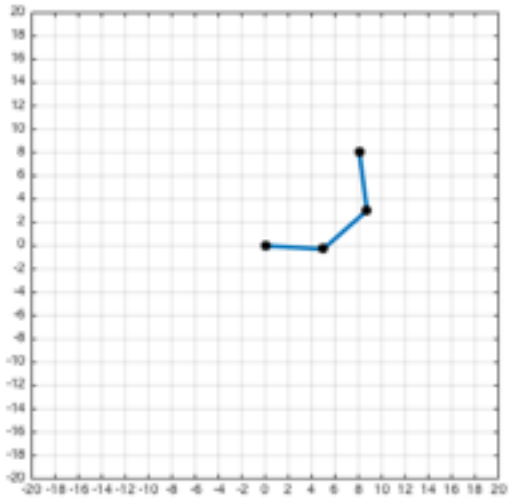


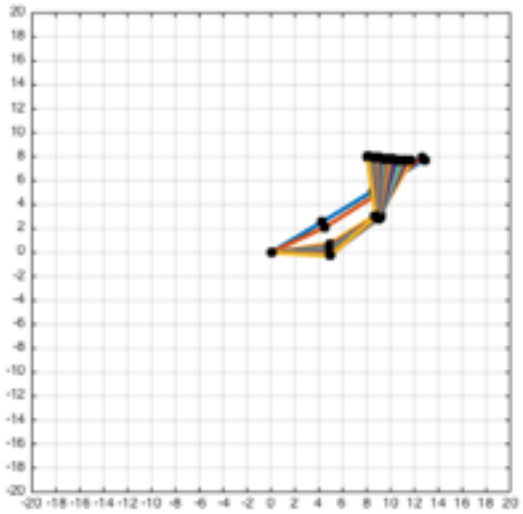

(b) $\theta_0 = \pi/6, \theta_1 = \pi/6, \theta_2 = \pi/6, l_1 = 3, l_2 = 5, l_3 = 7$



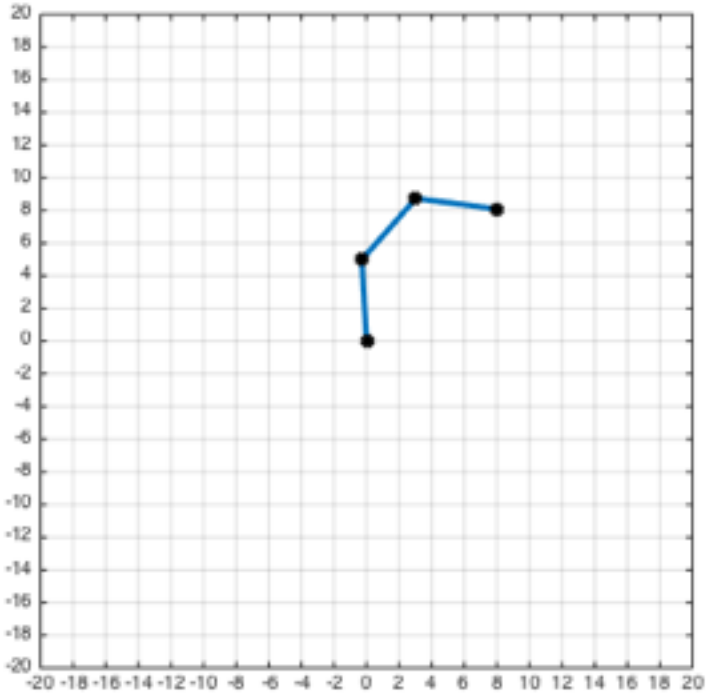
(ii). inverse

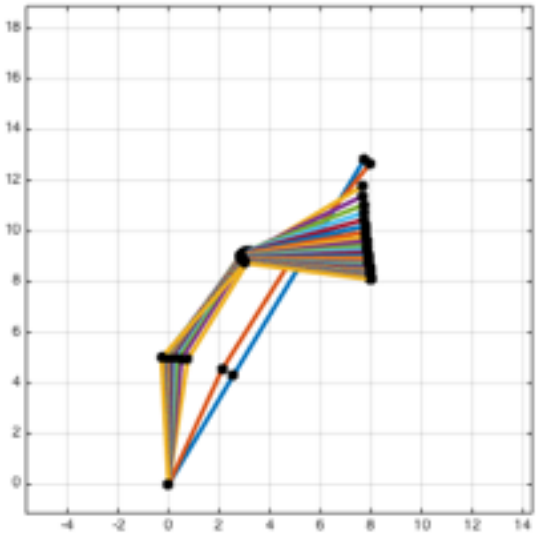
(1) $\theta_0 = \pi/6, \theta_1 = 0, \theta_2 = 0$ ( the right one is with ‘hold on’)





(2) $\theta_0 = \pi/3, \theta_1 = 0, \theta_2 = 0$.





Appendix:

Problem 1

1. code for contour and mesh:

```
contour(x,y,z(x,y),20);hold on;

plot(1,1,'b*');hold on;

plot(100,100,'r*');hold on;

quiver(x,y,dx(x,y),dy(x,y));hold off;

figure

 mesh(x,y,z(x,y));hold on;

 plot3(1,1,z(1,1),'b*');

 plot3(100,100,z(100,100),'r*');
```

2. code for gradient descent:

```
absgradient=sqrt(dx(1,1)^2+dy(1,1)^2);

lambda=1/absgradient;


while (absgradient>0)

   % cur_loc=cur_loc-lambda*[];

   % absgradient=sqrt();


absgradient=sqrt(dx(round(cur_loc(1,1)),round(cur_loc(1,2)))^2+dy(round(cur_loc(1,1)),r
ound(cur_loc(1,2)))^2);

    lambda=1/absgradient;
```

```
cur_loc=cur_loc-
lambda*([dx(round(cur_loc(1,1)),round(cur_loc(1,2))),dy(round(cur_loc(1,1)),round(cur_l
oc(1,2)))]);

   %absgradient=sqrt(dx((cur_loc(1,1)),(cur_loc(1,2)))^2+dy((cur_loc(1,1)),
(cur_loc(1,2)))^2);

   %lambda=1/absgradient;

   plot3(cur_loc(1,1), cur_loc(1,2), z(cur_loc(1,1), cur_loc(1,2)), '*b');

   pause(.1);

end

hold off;
```

Problem 2:

**1. get unexplored_areas:**

```
function unexplored_areas = get_unexplored_areas(explore_map, UNMAPPED)

[r,c]=find(explore_map==UNMAPPED);

if(isempty([r,c]))

   unexplored_areas = [];

end

unexplored_areas = [r,c];

end
```

**2. get new_destination:**

```
function dest = get_new_destination(curPos, unexplored_areas)

difference = sqrt((unexplored_areas(:,1)-curPos(1,1)).^2+(unexplored_areas(:,2)-
curPos(1,2)).^2);

[row,col]=find(difference==min(difference));

row1 = row(1,1);

dest = unexplored_areas(row1,:);

end
```

**3. update explore_map:**

```
function explore_map = update_explore_map(dest, route, explore_map, PLANNED,
UNMAPPED)

for i=1:size(route,1)

    if (explore_map(route(i,1),route(i,2))==UNMAPPED)

    explore_map(route(i,1),route(i,2)) = PLANNED;

    end

end

end
```

**4. update position:**

```
function [curPos, route, dest, explore_map] = update_position(curPos, route, dest,
explore_map, MAPPED)
```

```
curPos= route(1,:);   %1)
explore_map(curPos(1,1),curPos(1,2))=MAPPED;  %2)
if (curPos==dest)      %3)
    dest=[];
end
route(1,:)=[];      %4)
end
```

Problem 3

**forward:**

```
function [x_1,y_1,x_2,y_2,x_e,y_e] = ForwardKinematics(l0,l1,l2, theta0, theta1, theta2)
x_0=0;
y_0=0;
x_1=x_0+l0*cos(theta0);
y_1=y_0+l0*sin(theta0);
x_2=x_1+l1*cos(theta0+theta1);
y_2=y_1+l1*sin(theta0+theta1);
x_e=x_2+l2*cos(theta0+theta1+theta2);
y_e=y_2+l2*sin(theta0+theta1+theta2);
end
```

**Inverse:**

```
function [theta0_target, theta1_target, theta2_target] =
InverseKinematics(l0,l1,l2,x_e_target,y_e_target)


    % Initialize the thetas to some value

    theta0 = pi/6;

    theta1 = 0;

    theta2 = 0;




    l0=5;

    l1=5;

    l2=5;

    x_e_target=8;

    y_e_target=8;

    % Obtain end effector position x_e, y_e for current thetas:

    % HINT: use your ForwardKinematics function


     [x_1,y_1,x_2,y_2,x_e,y_e] = ForwardKinematics(l0,l1,l2, theta0, theta1, theta2);



    while  (sqrt((x_e_target-x_e)^2+(y_e_target-y_e)^2)>0.1)
```

```
%(Replace the '1'  with a condition that checks if your estimated [x_e,y_e] is close to [x_e_target,y_e_target])

% Calculate the Jacobian matrix for current values of theta:

% HINT: write a function for doing this

dxedth0=-l0*sin(theta0)-l1*sin(theta0+theta1)-l2*sin(theta0+theta1+theta2);

dxedth1=-l1*sin(theta0+theta1)-l2*sin(theta0+theta1+theta2);

dxedth2=-l2*sin(theta0+theta1+theta2);


dyedth0=l0*cos(theta0)+l1*cos(theta0+theta1)+l2*cos(theta0+theta1+theta2);

dyedth1=l1*cos(theta0+theta1)+l2*cos(theta0+theta1+theta2);

dyedth2=l2*cos(theta0+theta1+theta2);



J=[dxedth0 dxedth1 dxedth2; dyedth0 dyedth1 dyedth2];

% Calculate the pseudo-inverse of the jacobian using 'pinv()':


jpinv=pinv(J);


% Update the values of the thetas by a small step:


dtheta=0.1*jpinv*[x_e_target-x_e;y_e_target-y_e];

theta0=theta0+dtheta(1,1);
```

```
        theta1=theta1+dtheta(2,1);

        theta2=theta2+dtheta(3,1);



        % Obtain end effector position x_e, y_e for the updated thetas:

         [x_1,y_1,x_2,y_2,x_e,y_e] = ForwardKinematics(l0,l1,l2, theta0, theta1, theta2);


        % Draw the robot using drawRobot( ) : This will help you visualize how the robot
arm moves through the iteration:

        drawRobot(x_1,y_1,x_2,y_2,x_e,y_e);hold on;


        pause(0.00001)  % This will slow the loop just a little bit to help you visualize the
robot arm movement

    end

    % Set the theta_target values:

    theta0_target = theta0;

    theta1_target = theta1;

    theta2_target = theta2;


end
```

**drawRobot:**

```
function drawRobot(x_1,y_1,x_2,y_2,x_e,y_e)

   % uncomment these two lines to plot forward kinematics

   % [x_1,y_1,x_2,y_2,x_e,y_e] = ForwardKinematics(3,5,7, pi/6, pi/6, pi/6);

   %[x_1,y_1,x_2,y_2,x_e,y_e] = ForwardKinematics(5,5,5, pi/6, pi/4, -pi/3);

    figure(1)

    plot([0 x_1 x_2 x_e],[0 y_1 y_2 y_e],'lineWidth',3,'Marker','o','MarkerSize',
5,'MarkerEdgeColor','black','MarkerFaceColor','black');

    pbaspect([1 1 1]);

    xlim([-20 20]);

    ylim([-20 20]);

    set(gca,'Xtick',-20:2:20)

    set(gca,'Ytick',-20:2:20)

    grid on

end
```