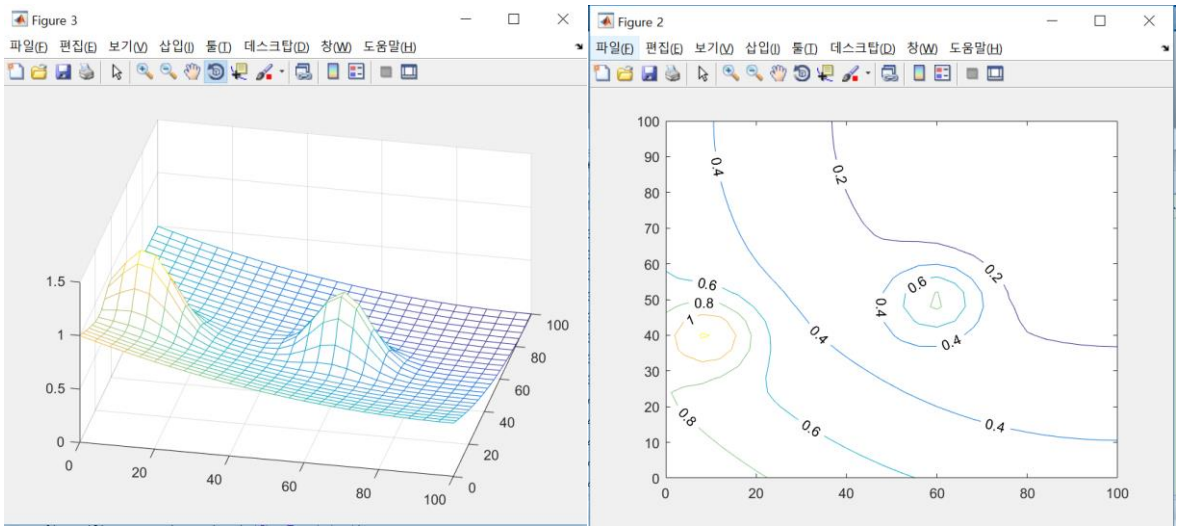
ECE172A
Jiseok Jeong
A14281757

Homwork 2

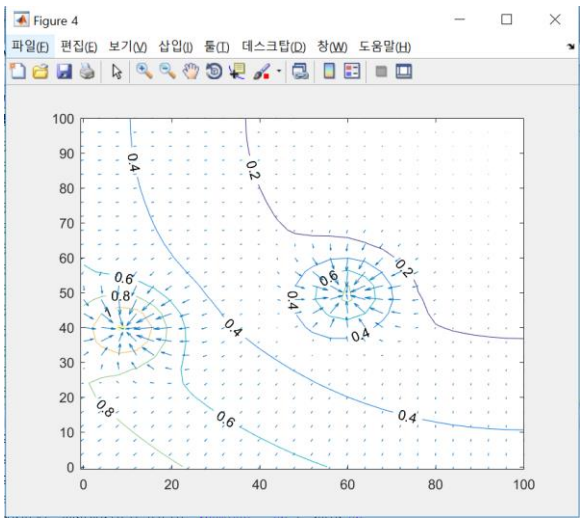
Problem 1. Better Robot Traversal (15 points)

(i) Plotting the potential field

(a)

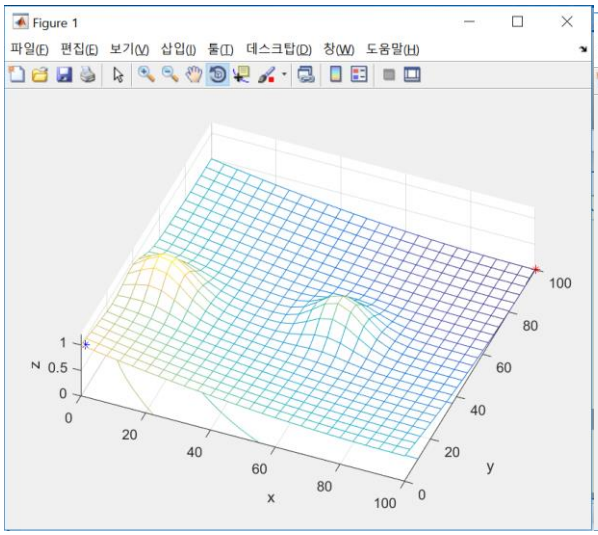


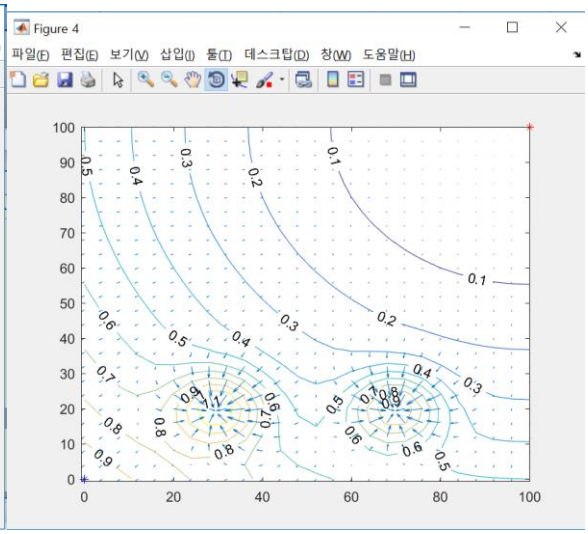
(b)



ECE172A
Jiseok Jeong
A14281757

(c)





•Where are the obstacles in the room?

They are at the (60, 50) and (10, 40)

- ***Why do the obstacles look like they do on the potential field?***

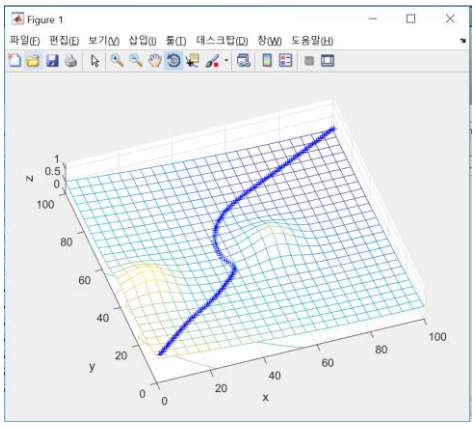
By the symbolic equation, points near obstacles have larger values of potential field.

- ***What happens to the gradient as you approach the end location?***

The closer we are to the end location, the smaller the gradient at the location becomes. This is the reason why we do cur-loc=cur-loc-A*gradient (A=1/|gradient|) to keep the distance of each movement.

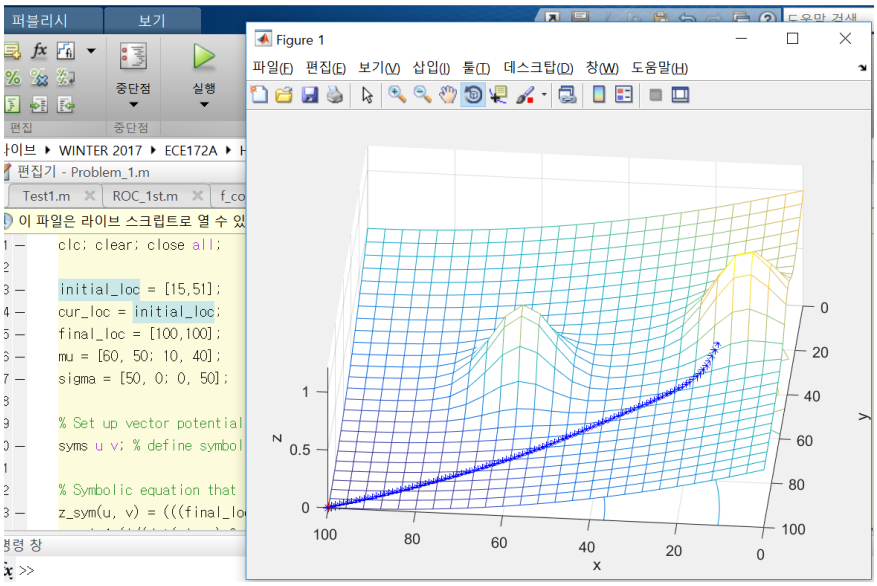
(ii) Gradient descent algorithm

(a) (b)

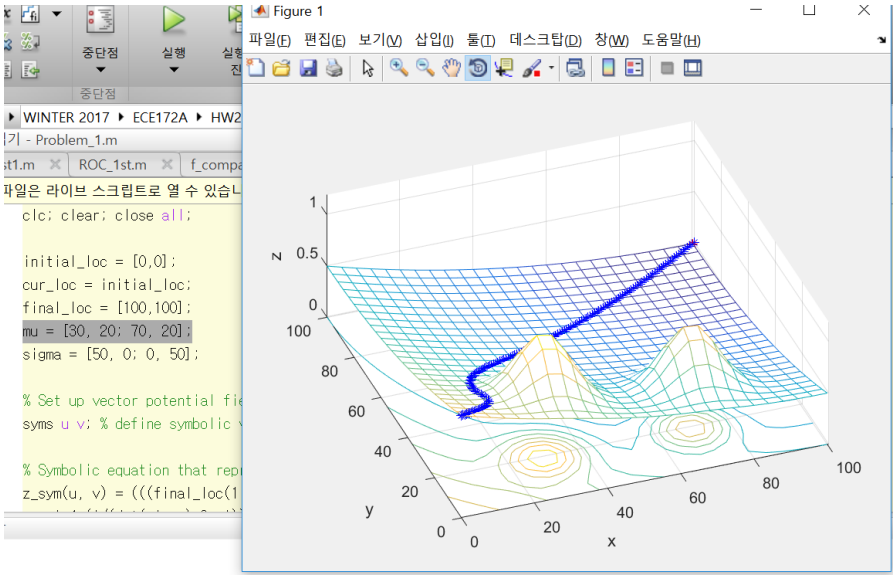


ECE172A
Jiseok Jeong
A14281757

(b)



(c)



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

With this method, we can 'plan' these robots by using potential field rather than make these robots just sense and act.

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

In this algorithm, we use the gradient value at the current location as a parameter used for determining next location. The gradient vectors at the location near the obstacles are towards the obstacles. Therefore we can avoid them by substracting these gradient value from the current location.

ECE172A
Jiseok Jeong
A14281757

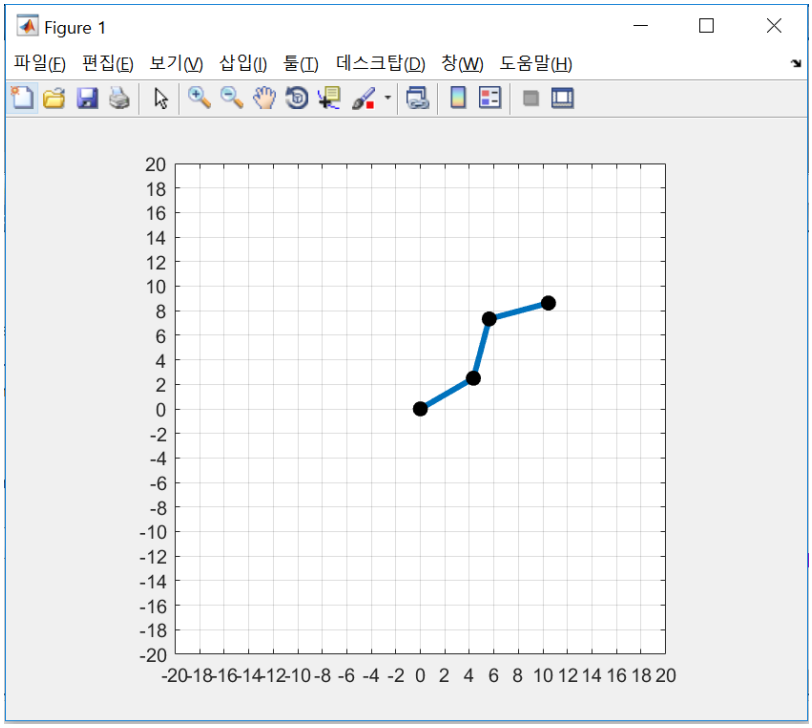
Problem 2. Swarms (15 points)

codes are in Appendix

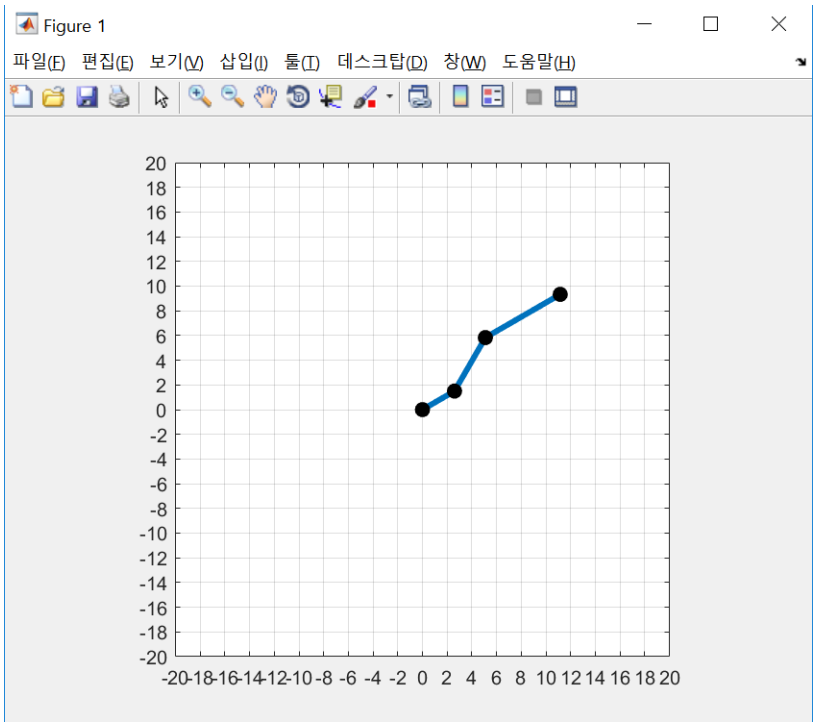
Problem 3. Robot Kinematics (15 points)

(i) Forward Kinematics:

(a) θ0 = π/6, θ1 = π/4, θ2 = −π/3, l1 = 5, l2 = 5, l3 = 5



(b) θ0 = π/6, θ1 = π/6, θ2 = π/6, l1 = 3, l2 = 5, l3 = 7



ECE172A
Jiseok Jeong
A14281757

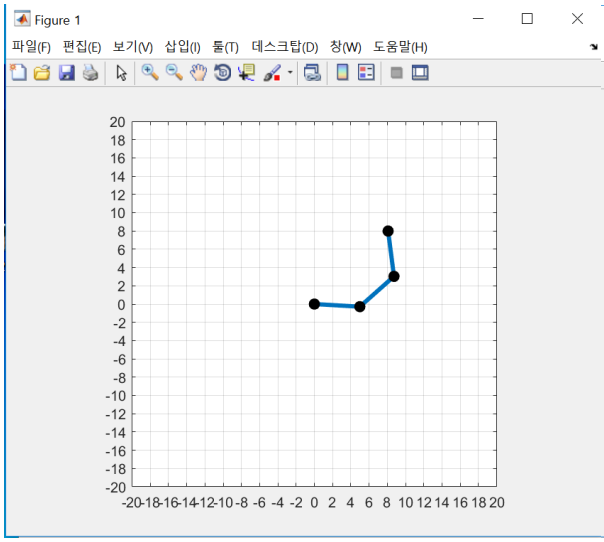
(ii) Inverse Kinematics:

find the values of θ0, θ1 and θ2 for l1 = l2 = l3 = 5 and (xtarget,ytarget) = (8,8)

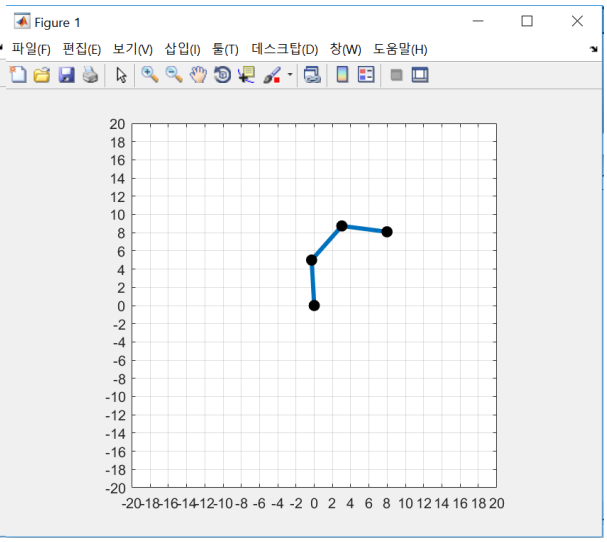
theta0_target	-4.7299
theta1_target	5.6686
theta2_target	5.1254

all scales are expressed as radian scales .

(1)θ0 = π/6, θ1 = 0, θ2 = 0

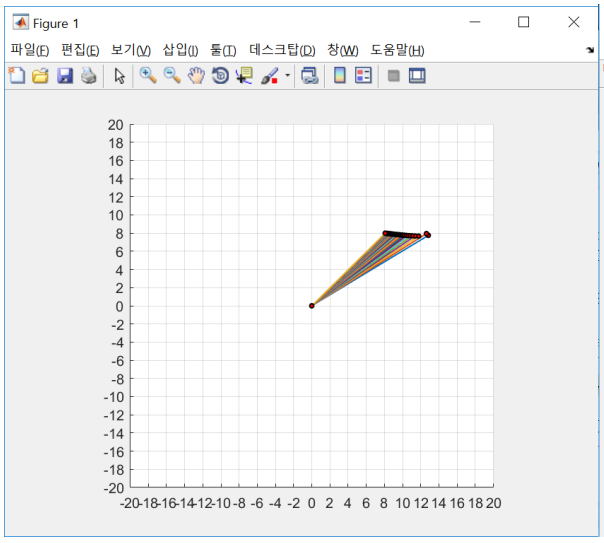


(2) θ0 = π/3, θ1 = 0, θ2 = 0

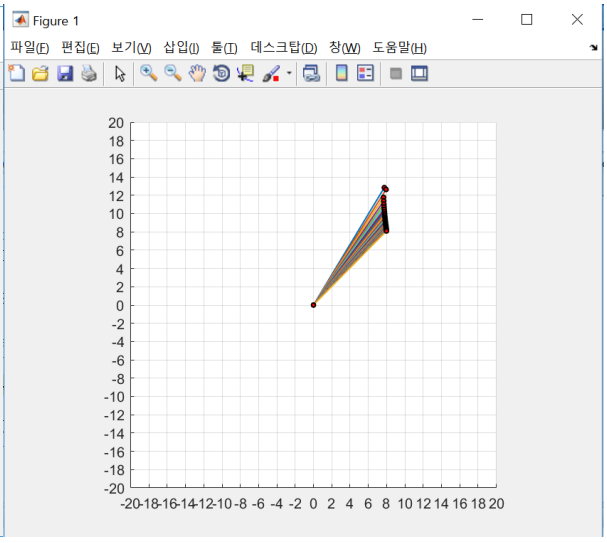


(iii) Plot endpoints of the effector

(a) θ0 = π/6, θ1 = 0, θ2 = 0



(b) θ0 = π/3, θ1 = 0, θ2 = 0.



ECE172A
Jiseok Jeong
A14281757

APPENDIX

Problem 1

```
figure(2), contour(x,y,z(x,y),'ShowText','on') % (i)_(a)

figure(3), mesh(x,y,z(x,y)); % (i)_(a)

figure(4), contour(x,y,z(x,y),'ShowText','on'); hold on; % (i)_(b)


plot3(cur_loc(1,1), cur_loc(1,2), z(cur_loc(1,1), cur_loc(1,2)), '*b'); %(i)_(c)

plot3(final_loc(1,1), final_loc(1,2), z(final_loc(1,1), final_loc(1,2)), '*r'); %(i)_(c)


(ii)

gradient=[dx(cur_loc(1),cur_loc(2)) dy(cur_loc(1),cur_loc(2))];

    A=sqrt(gradient(1)*gradient(1)+gradient(2)*gradient(2));

while ~(cur_loc(1)>100 && cur_loc(2)>100) % to avoid the infinite loop.

    gradient=[dx(cur_loc(1),cur_loc(2)) dy(cur_loc(1),cur_loc(2))];

    A=sqrt(gradient(1)*gradient(1)+gradient(2)*gradient(2));

    cur_loc = cur_loc - (1/A)*gradient;

    %Plot the bots current location on the mesh.

    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

```
function unexplored_areas = get_unexplored_areas(explore_map, UNMAPPED)
index_unmapped = find(explore_map==UNMAPPED);
[I,J] = ind2sub(size(explore_map),index_unmapped(:,1));
unexplored_areas = [I,J];
end


function dest = get_new_destination(curPos, unexplored_areas)
distance = sqrt((curPos(1)-unexplored_areas(:,1)).^2 + (curPos(2)-unexplored_areas(:,2)).^2);
[Y, I] = min(distance);
dest = [unexplored_areas(I,1),unexplored_areas(I,2)];
```

```
end
```

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

```
size_route = size(route);
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
if explore_map(dest(1),dest(2))==UNMAPPED
        explore_map(route(i,1),route(i,2))=PLANNED;
end

end
```

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

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

end
```

Problem 3

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

```
    x_1 = l0*cos(theta0);
    x_2 = x_1+l1*cos(theta0+theta1);
    x_e = x_2+l2*cos(theta0+theta1+theta2);
    y_1 = l0*sin(theta0);
    y_2 = y_1+l1*sin(theta0+theta1);
    y_e = y_2+l2*sin(theta0+theta1+theta2);
    drawRobot(x_1,y_1,x_2,y_2,x_e,y_e)
end
```

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/3;
    theta1 = 0;
    theta2 = 0;
```

ECE172A
Jiseok Jeong
A14281757

```
% 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


    % Calculate the Jacobian matrix for current values of theta:
    % HINT: write a function for doing this
    J = MakeJacobian(l0,l1,l2,theta0, theta1, theta2);



    % Calculate the pseudo-inverse of the jacobian using 'pinv()':
    J_inv=pinv(J);


    % Update the values of thetas by a small step:
    delta_theta = 0.1 * J_inv * [x_e_target-x_e;y_e_target-y_e];
    theta0 = theta0 + delta_theta(1);
    theta1 = theta1 + delta_theta(2);
    theta2 = theta2 + delta_theta(3);

    % 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);


    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
```

function J = MakeJacobian(l0,l1,l2,theta0, theta1, theta2)

ECE172A
Jiseok Jeong
A14281757

```
J = [-l0*sin(theta0) - l1*sin(theta0 + theta1) - l2*sin(theta0 + theta1 + theta2), -l1*sin(theta0 + theta1) - l2*sin(theta0 + theta1 + theta2),-l2*sin(theta0 + theta1 + theta2);l0*cos(theta0) + l1*cos(theta0 + theta1) + l2*cos(theta0 + theta1 + theta2),l1*cos(theta0 + theta1) + l2*cos(theta0 + theta1 + theta2),l2*cos(theta0 + theta1 + theta2)];
end
```

function draw_endpoint(x_e,y_e)

```
plot([0 x_e],[0 y_e],'lineWidth',1,'Marker','o','MarkerSize',3,'MarkerEdgeColor','black','MarkerFaceColor','red');
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
```

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

```
%this one is for a plot showing the end effector positions through all the iterations.


    % Initialize the thetas to some value
    theta0 = pi/3;
    theta1 = 0;
    theta2 = 0;

    % 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);

  figure(1);    %only use this when we use 'draw_endpoint(x_e,y_e);' which
  hold on;      %is in the following 'while' loop

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



        % Calculate the Jacobian matrix for current values of theta:
        % HINT: write a function for doing this
        J = MakeJacobian(l0,l1,l2,theta0, theta1, theta2);



        % Calculate the pseudo-inverse of the jacobian using 'pinv()':
        J_inv=pinv(J);
```

ECE172A
Jiseok Jeong
A14281757

```
        % Update the values of thetas by a small step:
        delta_theta = 0.1 * J_inv * [x_e_target-x_e;y_e_target-y_e];
        theta0 = theta0 + delta_theta(1);
        theta1 = theta1 + delta_theta(2);
        theta2 = theta2 + delta_theta(3);

        % 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); %when we use this, we have to deactivate line
35,36,69 and 84 not to have an error

draw_endpoint(x_e,y_e);      % we activate the following script only when we want to get the
figure of the position of effecter
%choose only either 'drawRobot(x_1,y_1,x_2,y_2,x_e,y_e);' or draw_endpoint(x_e,y_e);' to avoid
error.

        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;


hold off;




end
```