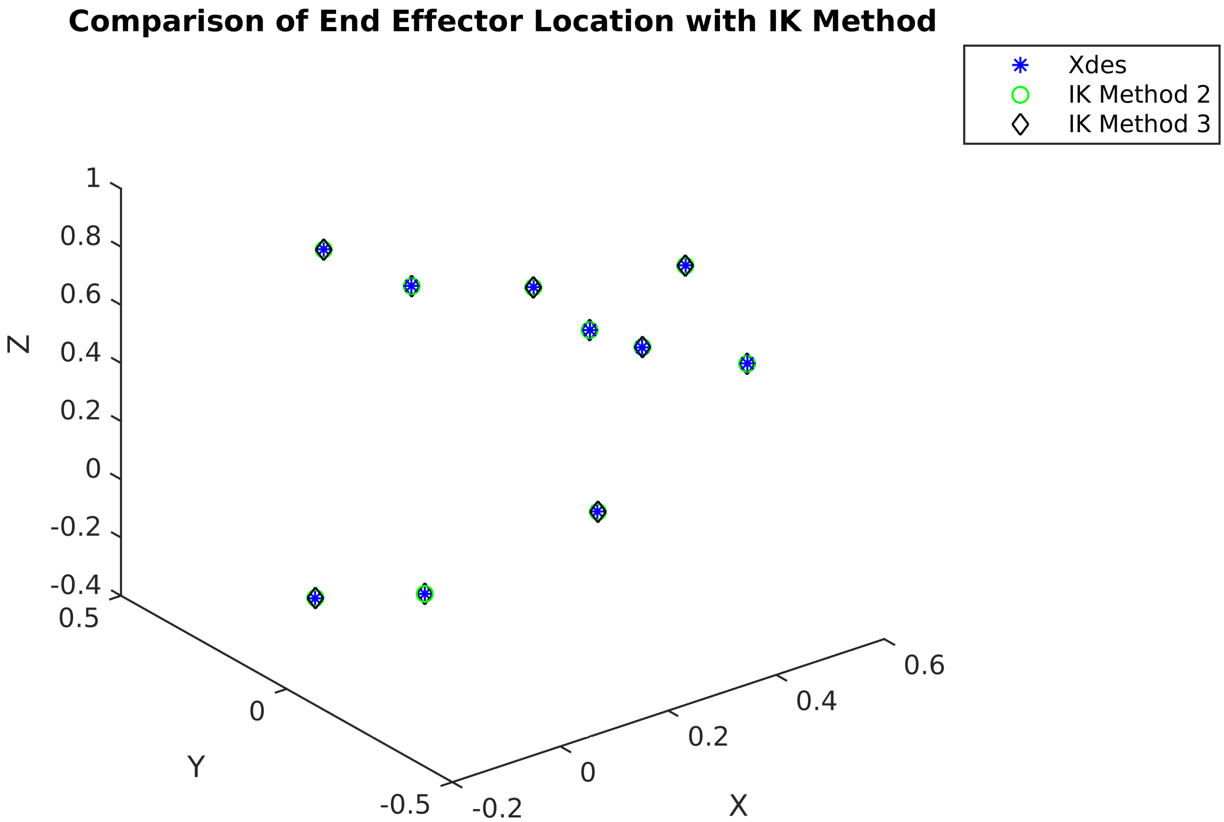
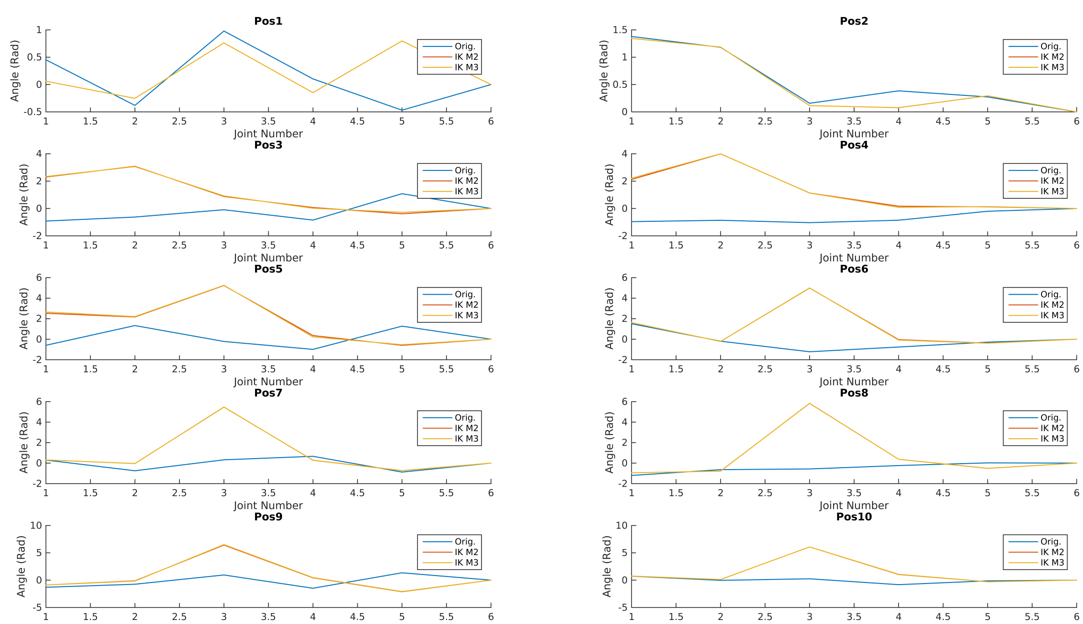
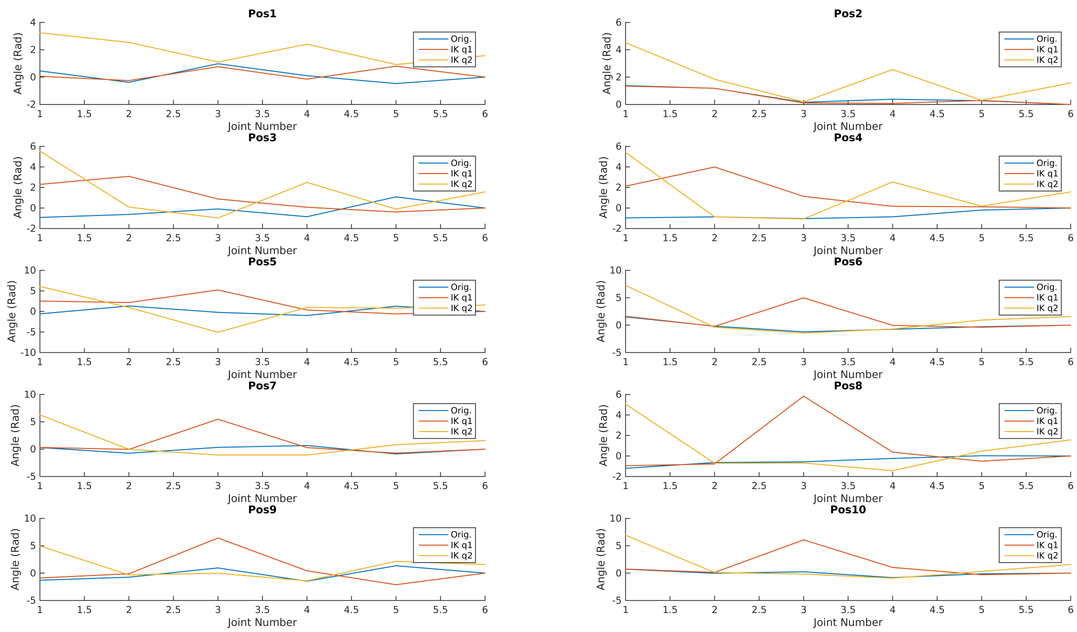
```matlab
%MDL_HW_4
clear all
close all
clc

%Run Robotics Toolbox
%run ~/Desktop/rvctools/startup_rvc.m

%%%%%%%%%%%%% theta, d, a, alpha, revolute or prismatic, offset
L(1) = Link([ 0     0.3     0          -pi/2    0        0], 'standard');
L(2) = Link([ 0     0.      0.3        0        0        0], 'standard');
L(3) = Link([ 0     0.      0         pi/2    0         pi/2], 'standard');
L(4) = Link([ 0     0.2     0        -pi/2    0         pi/2], 'standard');
L(5) = Link([ 0     0.0     0         pi/2    0         0   ], 'standard');
L(6) = Link([ 0     0.2     0          0      0         0], 'standard');


%% defining the robot now
bot = SerialLink(L, 'name', 'HW 4', ...
    'manufacturer', 'Killpack Inc.');

%% Generate 10 random reachable positions using forward kinematics
%position of the origin of the end effector in the end effector frame
end_eff_pos = [0;0;0;1];
a = -pi/2;
b = pi/2;
pos_fk = [];
q_fk = [];
for i = 1:10,
    r = a + (b-a).*rand(5,1);
    H = bot.fkine([r', 0]);
    pos_fk(:,end+1) = H*end_eff_pos;
    q_fk(:,end+1) = ([r', 0]');
    %bot.plot([r',0])
    %pause(0.05)
end
pos_fk = pos_fk(1:3,:);

%% Calculate Joint Angles using Inverse Kinematics Methods 2 & 3
%Initial Starting Point
q1 = [0 0 0 0 0 0];
q2 = [pi/2 pi/2 pi/2 pi/2 pi/2 pi/2];
q = [q1; q2];

%Tuning Parameters
k = 1;
K = eye(3);
q_ik = [];
pos_ik = [];

%Inverse Kinematics
for method = 2:3,
    for j = 1:2,
        qk = q(j,:);
        Ja = bot.jacob0(qk);
        Ja = Ja(1:3,:);
        H = bot.fkine(qk);
        x = H*end_eff_pos;
```

```
        x = x(1:3);
        for i = 1:10,
            xdes = pos_fk(:,i);
            while sum((x-xdes).^2) > 1e-6,
                if method == 2,
                    qdot = Ja.'*inv(Ja*Ja.' + k^2*eye(3))*(K*(xdes - x));
                elseif method == 3,
                    qdot = Ja.'*K*(xdes-x);
                end

                q_k_pl_1 = qk + qdot.';
                H = bot.fkine(q_k_pl_1);
                x = H*end_eff_pos;
                x = x(1:3);
                Ja = bot.jacob0(q_k_pl_1);
                Ja = Ja(1:3,:);
                qk = q_k_pl_1;
            end
            q_ik(:,end+1) = qk.';
            pos_ik(:,end+1) = x;
        end
        if j == 1,
            if method == 2,
                q_ik_m2_q1 = q_ik;
                pos_ik_m2_q1 = pos_ik;
                q_ik = [];
                pos_ik = [];
            else
                q_ik_m3_q1 = q_ik;
                pos_ik_m3_q1 = pos_ik;
                q_ik = [];
                pos_ik = [];
            end
        else
            if method == 2,
                q_ik_m2_q2 = q_ik;
                pos_ik_m2_q2 = pos_ik;
                q_ik = [];
                pos_ik = [];
            else
                q_ik_m3_q2 = q_ik;
                pos_ik_m3_q2 = pos_ik;
                q_ik = [];
                pos_ik = [];
            end
        end
    end
end
% Method 2 Error
end_eff_error_q1_m2 = pos_fk - pos_ik_m2_q1;
joint_ang_error_q1_m2 = q_fk - q_ik_m2_q1;

end_eff_error_q2_m2 = pos_fk - pos_ik_m2_q2;
joint_ang_error_q2_m2 = q_fk - q_ik_m2_q2;

% Method 3 Error
end_eff_error_q1_m3 = pos_fk - pos_ik_m3_q1;
joint_ang_error_q1_m3 = q_fk - q_ik_m3_q1;
```

```
end_eff_error_q2_m3 = pos_fk - pos_ik_m3_q2;
joint_ang_error_q2_m3 = q_fk - q_ik_m3_q2;

% Make Figures
figure(1)
plot3(pos_fk(1,:),pos_fk(2,:),pos_fk(3,:),'b*')
hold on
plot3(pos_ik_m2_q1(1,:),pos_ik_m2_q1(2,:),pos_ik_m2_q1(3,:),'go')
plot3(pos_ik_m3_q1(1,:),pos_ik_m3_q1(2,:),pos_ik_m3_q1(3,:),'kd')
xlabel('X')
ylabel('Y')
zlabel('Z')
legend('Xdes', 'IK Method 2', 'IK Method 3')
title('Comparison of End Effector Location with IK Method')

figure(2)
for i = 1:10,
    subplot(5,2,i)
    hold on
    plot(q_fk(:,i),'LineWidth',1)
    plot(q_ik_m2_q1(:,i),'LineWidth',1)
    plot(q_ik_m3_q1(:,i),'LineWidth',1)
    xlabel('Joint Number')
    ylabel('Angle (Rad)')
    title(strcat('Pos ',num2str(i)))
    legend('Orig.','IK M2','IK M3')
end

figure(3)
for i = 1:10,
    subplot(5,2,i)
    hold on
    plot(q_fk(:,i),'LineWidth',1)
    plot(q_ik_m2_q1(:,i),'LineWidth',1)
    plot(q_ik_m2_q2(:,i),'LineWidth',1)
    xlabel('Joint Number')
    ylabel('Angle (Rad)')
    title(strcat('Pos ',num2str(i)))
    legend('Orig.','IK q1','IK q2')
end


%% Answers to question 1.-(a)-iii.
%Both IK algorithms produce end effector positions that are essentially the
%same as my original end effector positions (my convergence criteria
%ensured this fact). I only generated positions that were reachable in the
%workspace, so neither algorithm ever failed to find the final end effector
%position. The joint angles found varied slightly with IK algorithm (though
%not too much), but greatly varied with the initial starting point.
```


Comparison of End Effector Location with IK Method
Xdes
IK Method 2
IK Method 3
Z
Y
X
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0.5
0
-0.5
-0.2
0
0.2
0.4
0.6


Pos1
Pos2
Pos3
Pos4
Pos5
Pos6
Pos7
Pos8
Pos9
Pos10
Angle (Rad)
Joint Number
Orig.
IK M2
IK M3


Pos1
Pos2
Pos3
Pos4
Pos5
Pos6
Pos7
Pos8
Pos9
Pos10
Angle (Rad)
Joint Number
Orig.
IK q1
IK q2