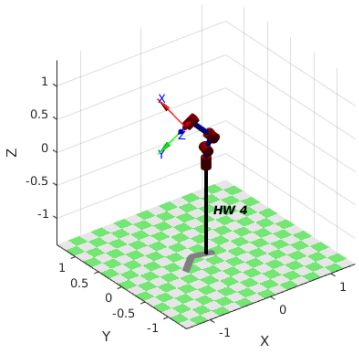
MEEn 537
HW 4
Jae Lee

As far as I observed, there was no specific positions that the robot couldn't get to as long as the desired position is within the workspace of the robot. Depending on starting configuration or method used, it takes longer or shorter time to get to the desired point.

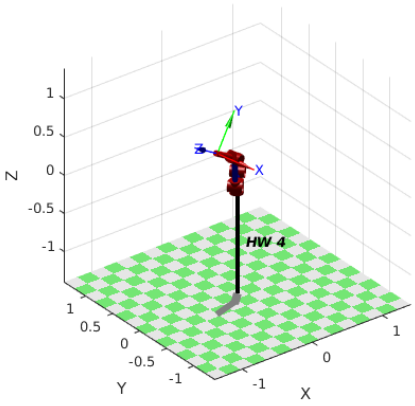
Results

For the desired_position = [-0.5441, -0.2738, 0.7873]
q1 = [0 0 0 0 0 0]
q2 = [pi/2 pi/2 pi/2 pi/2 pi/2 pi/2]
The following is the result for the method 2, 3 and different starting positions(q1, q2).

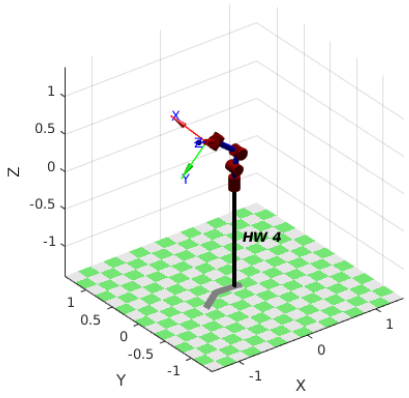
==========Method 2, q1==========
num_iteration = 114
final_joint_angles = [-0.2708 -1.1808 -1.2011 -0.1502 -0.9268 0]



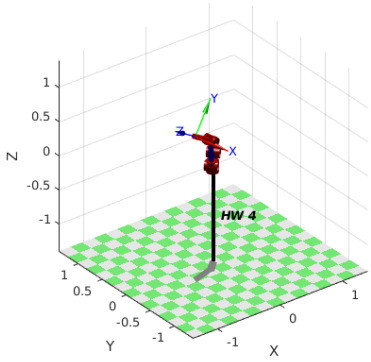
==========Method 2, q2==========
num_iteration = 282
final_joint_angles = [0.7674 2.6279 1.9709 1.0606 0.9027 1.5708]



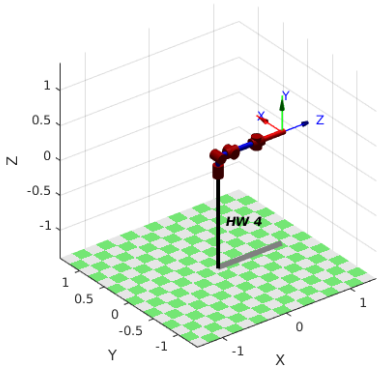
==========Method 3, q1==========
num_iteration = 52
final_joint_angles = [-0.0580 -1.2936 -1.1925 -0.0950 -0.5824 0]



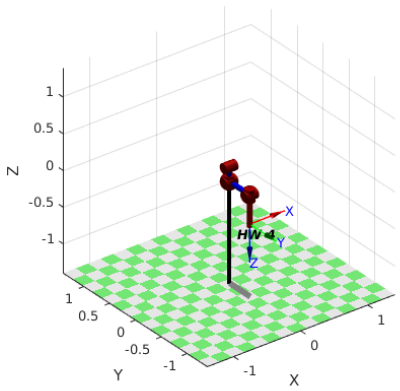
==========Method 3, q2==========
num_iteration = 244
final_joint_angles = [0.7610 2.6184 1.9766 1.0679 0.8942 1.5708]



==========Original q1==========



==========Original q2==========



Code

```
%MDL_HW_2

clear; close all; clc;
clear L

%%%%%%%%%%%% theta, d, a, alpha, revolute or prismatic, offset
L(1) = Link([ 0     0.2     0         -pi/2    0        0], 'standard');
L(2) = Link([ 0     0.      0.2        0       0        0], 'standard');
L(3) = Link([ 0     0.      0         pi/2     0       pi/2], 'standard');
L(4) = Link([ 0     0.4     0        -pi/2     0        pi/2], 'standard');
L(5) = Link([ 0     0.0     0         pi/2     0        0   ], 'standard');
L(6) = Link([ 0     0.4     0         0        0       0], 'standard');


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

% some useful poses
q1 = [0 0 0 0 0 0]; % zero angles, L shaped pose
q2 = [pi/2 pi/2 pi/2 pi/2 pi/2 pi/2];

num_tests = 1;
%generating random joint angles with joint limits
jt_angles = random('uniform', -pi/2, pi/2, 6, num_tests);

%making emtpy vector to store positions
positions = zeros(3, num_tests);

%calculating FK for each set of random joint angles
for i=1:num_tests
    FK = robot.fkine(jt_angles(:,i));
    positions(:, i) = FK(1:3, 4);
end
```

```
%%========== method 2 ==========
% starting configuration with q1
k = 1;
gain = 1;
kappa = [gain*eye(3); zeros(3)];
threshold = 0.1;
time_step = 0.1;
for i=1:num_tests
    q = q1;
    error = 100;
    iteration = 0;
    while error > threshold
        J = robot.jacob0(q);
        J_A = [eye(3) zeros(3); zeros(3) zeros(3)]*J;
        current_fk = robot.fkine(q);
        current_position = current_fk(1:3,4);
        q_dot = J_A'*inv(J_A*J_A' + k^2*eye(6))*(kappa*(positions(:,i)-current_position));
        q = q + q_dot'*time_step;
        robot.plot(q);
        iteration = iteration+1;
        error = norm(current_position-positions(:,i))
    end
    disp('==========Method 2, q1==========');
    desired_position = positions(:,i)
    num_iteration = iteration
    final_joint_angles = q
end

% starting configuration with q2
for i=1:num_tests
    q = q2;
    error = 100;
    iteration = 0;
    while error > threshold
        J = robot.jacob0(q);
        J_A = [eye(3) zeros(3); zeros(3) zeros(3)]*J;
        current_fk = robot.fkine(q);
        current_position = current_fk(1:3,4);
        q_dot = J_A'*inv(J_A*J_A' + k^2*eye(6))*(kappa*(positions(:,i)-current_position));
        q = q + q_dot'*time_step;
        robot.plot(q);
        iteration = iteration+1;
        error = norm(current_position-positions(:,i))
    end
    disp('==========Method 2, q2==========');
    desired_position = positions(:,i)
    num_iteration = iteration
    final_joint_angles = q
end

%%========== method 3 ==========
% starting configuration with q1
gain = 1;
kappa = [gain*eye(3); zeros(3)];
threshold = 0.1;
time_step = 0.1;
for i=1:num_tests
```

```
    q = q1;
    error = 100;
    iteration = 0;
    while error > threshold
        J = robot.jacob0(q);
        J_A = [eye(3) zeros(3); zeros(3) zeros(3)]*J;
        current_fk = robot.fkine(q);
        current_position = current_fk(1:3,4);
        q_dot = J_A'*(kappa*(positions(:,i)-current_position));
        q = q + q_dot'*time_step;
        robot.plot(q);
        iteration = iteration+1;
        error = norm(current_position-positions(:,i))
    end
    disp('==========Method 3, q1==========');
    desired_position = positions(:,i)
    num_iteration = iteration
    final_joint_angles = q
end

% starting configuration with q2
for i=1:num_tests
    q = q2;
    error = 100;
    iteration = 0;
    while error > threshold
        J = robot.jacob0(q);
        J_A = [eye(3) zeros(3); zeros(3) zeros(3)]*J;
        current_fk = robot.fkine(q);
        current_position = current_fk(1:3,4);
        q_dot = J_A'*(kappa*(positions(:,i)-current_position));
        q = q + q_dot'*time_step;
        robot.plot(q);
        iteration = iteration+1;
        error = norm(current_position-positions(:,i))
    end
    disp('==========Method 3, q2==========');
    desired_position = positions(:,i)
    num_iteration = iteration
    final_joint_angles = q
end
```