Table of Contents

params and setup

Jesse Wynn Homework 4 Problem 3

```
clc
close all
clear

% load in the 10 random poses
load('random_poses.mat')

% load in the corresponding joint angles
load('q_angles.mat')

% load the robot
robot = robot_6dof();

% two different starting sets of joint angles
q1 = [0.0 0.0 0.0 0.0 0.0 0.0];
q2 = [pi/2 pi/2 pi/2 pi/2 pi/2 pi/2];

q_initials = [q1; q2];

rows = size(q_initials);
rows = rows(1);

% allocate space to hold q data from the different metods
q_original_angles = q_angles;

% for the zero joint angles config
q_method2_angles_0 = zeros(10,6);
q_method3_angles_0 = zeros(10,6);

% for the pi/2 joint angles config
q_method2_angles_pi_2 = zeros(10,6);
q_method3_angles_pi_2 = zeros(10,6);

% IK solution finding methods

% error threshold
error_thresh = 0.015;  % 1.5cm (kinda big but works better)

% time step
dt = 0.01;
```

method 2: Pseudo Inverse

```
error = 100;
count = 0;
max_iter = 1000;

% damped inverse scalar
kd = 0.1;

% error gain
K = 10 .* eye(6);
K = K .* dt;    % multiply by dt

% loop through the 10 random desired poses
for i = 1:10

    T_des = robot.fkine(q_angles(i,:));

    % for the two initial joint configurations
    for j = 1:rows

        % initialize variable q
        q = q_initials(j,:);

        % loop to find IK solution to T_des
        while error > error_thresh && count < max_iter

            % compute Jacobian
            J = robot.jacob0(q);
            %     J = J(1:3,:);    % just get the first 3 rows

            % get current pose
            T_cur = robot.fkine(q);

            % get the error between the current T and desired T
            e = tr2deltabase(T_cur, T_des);

            % compute new q and continue
            q = q + J'/(J*J'+kd^2*eye(6))*(K*e);    % '/' is same as
 inv()

            q = diag(q)';   % just grab the diagonal elements
 (hopefully this works)

            error = norm(e);    % this seems like an OK way to
 quantify error

            if count == max_iter - 1 && j == 1
                dispstrcat("Configuration ", int2str(i)," failed to
 converge using Pseudo Inverse method after ", int2str(max_iter -
 1), " iterations when starting from all zero joint angles")
            end
            if count == max_iter - 1 && j == 2
                disp(strcat("Configuration ", int2str(i)," failed to converge using Pseudo Inverse method after ", int2str(max_iter - 1), " iterations when starting from all pi/2 joint angles"))
            end

            count = count + 1;
%              disp(count)
        end

        % reset count and error
        count = 0;
        error = 100;

        % save some stuff for plots
        if j == 1
            q_method2_angles_0(i,:) = q;
        else
            q_method2_angles_pi_2(i,:) = q;
        end
    end
end
```

method 3: Jacobian Transpose

```
error = 100;
count = 0;
max_iter = 2000;

% scalar tuning param
alpha = 0.5;

% loop through the 10 random desired poses
for i = 1:10

    T_des = robot.fkine(q_angles(i,:));

    % for the two initial joint configurations
    for j = 1:rows

        % initialize variable q
        q = q_initials(j,:);

        % loop to find IK solution to T_des
        while error > error_thresh && count < max_iter

            % compute Jacobian
            J = robot.jacob0(q);
            %     J = J(1:3,:);    % just get the first 3 rows

            % get current pose
            T_cur = robot.fkine(q);
            % get the error between the current T and desired T
            e = tr2deltabase(T_cur, T_des);

            % compute delta_q using Jacobian transpose
            delta_q = alpha .* (J' * e);

            % compute new q and continue
            q = q + delta_q';

            error = norm(e);    % this seems like an OK way to
 quantify error

            if count == max_iter - 1 && j == 1
                disp(strcat("Configuration ", int2str(i)," failed to
 converge using Jacobian Transpose method after ", int2str(max_iter -
 1), " iterations when starting from all zero joint angles"))
            end

            if count == max_iter - 1 && j == 2
                disp(strcat("Configuration ", int2str(i)," failed to
 converge using Jacobian Transpose method after ", int2str(max_iter -
 1), " iterations when starting from all pi/2 joint angles"))
            end

            count = count + 1;
%             disp(count)
%             disp(error)

        end

        % reset count and error
        count = 0;
        error = 100;

        % save some stuff for plots
        if j == 1
            q_method3_angles_0(i,:) = q;
        else
            q_method3_angles_pi_2(i,:) = q;
        end
    end
end
```

```
Configuration 1 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all pi/2 joint angles
Configuration 2 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all pi/2 joint angles
Configuration 4 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all zero joint angles
Configuration 5 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all pi/2 joint angles
Configuration 8 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all zero joint angles
```

```
Configuration 8 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all pi/2 joint angles
Configuration 10 failed to converge using Jacobian Transpose method
 after 1999 iterations when starting from all zero joint angles
```

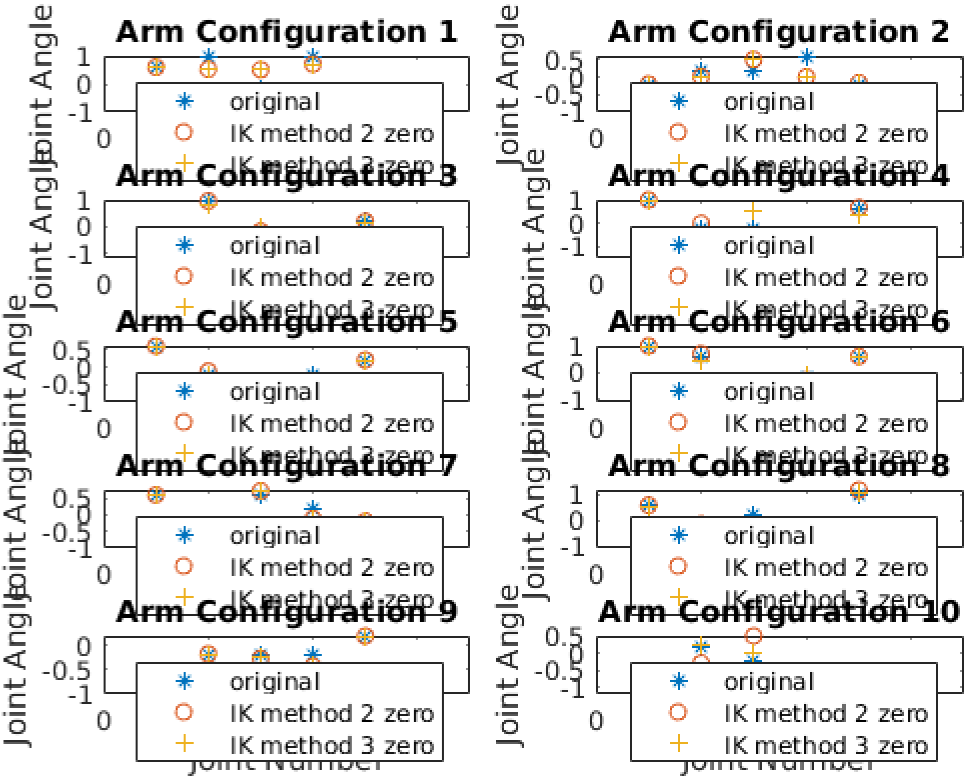
plots

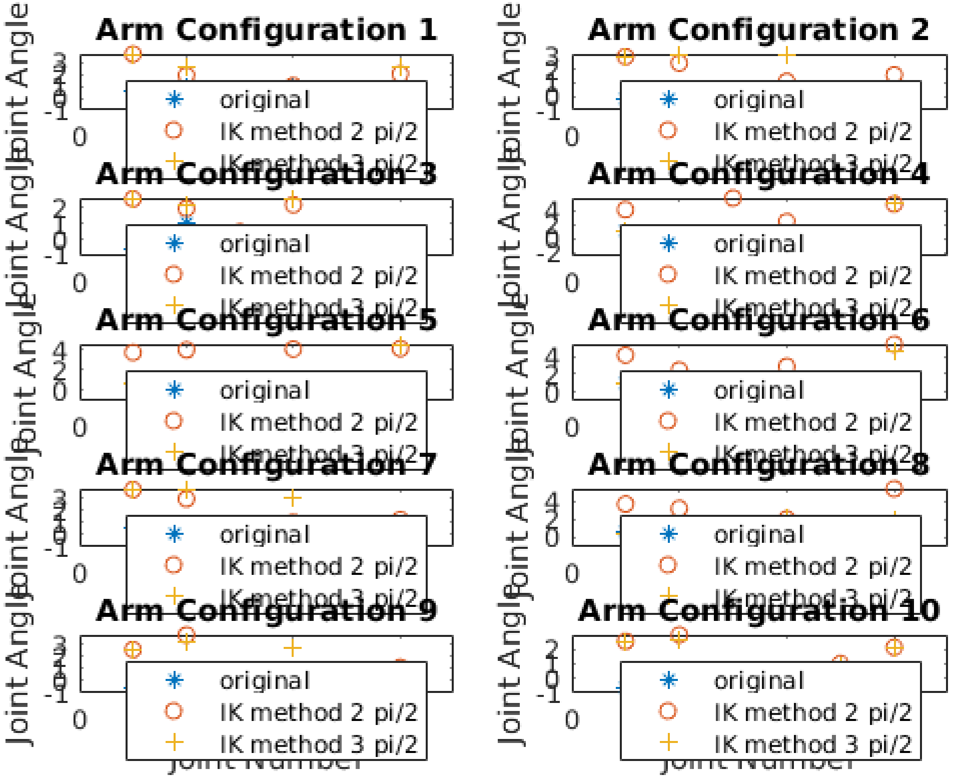
```
joints = [1, 2, 3, 4, 5, 6];

% plots of original q angles vs q angles found by IK methods
figure(1), clf
for i = 1:10
    num = int2str(i);
    subplot(5,2,i)
    plot(joints, q_original_angles(i,:), '*')
    hold on
    plot(joints, q_method2_angles_0(i,:), 'o')
    hold on
    plot(joints, q_method3_angles_0(i,:), '+')
    axis([0 7 -inf inf])
    ylabel('Joint Angle')
    legend('original', 'IK method 2 zero', 'IK method 3 zero')
    title(strcat("Arm Configuration ", num))
    if i >= 9
        xlabel('Joint Number')
    end
end

% plots of original q angles vs q angles found by IK methods (pi/2)
% starting points
figure(2), clf
for i = 1:10
    num = int2str(i);
    subplot(5,2,i)
    plot(joints, q_original_angles(i,:), '*')
    hold on
    plot(joints, q_method2_angles_pi_2(i,:), 'o')
    hold on
    plot(joints, q_method3_angles_pi_2(i,:), '+')
    axis([0 7 -inf inf])
    ylabel('Joint Angle')
    legend('original', 'IK method 2 pi/2', 'IK method 3 pi/2')
    title(strcat("Arm Configuration ", num))
    if i >= 9
        xlabel('Joint Number')
    end
end

% position error
```


Arm Configuration 1
Arm Configuration 2
Arm Configuration 3
Arm Configuration 4
Arm Configuration 5
Arm Configuration 6
Arm Configuration 7
Arm Configuration 8
Arm Configuration 9
Arm Configuration 10
Joint Angle
Joint Number
original
IK method 2 zero
IK method 3 zero
1
0
-1
0.5
-0.5


Arm Configuration 1
Arm Configuration 2
Arm Configuration 3
Arm Configuration 4
Arm Configuration 5
Arm Configuration 6
Arm Configuration 7
Arm Configuration 8
Arm Configuration 9
Arm Configuration 10
Joint Angle
Joint Number
original
IK method 2 pi/2
IK method 3 pi/2
4
2
0
-1
-2

Published with MATLAB® R2017a