Prob1. Generate trajectory using JointTrajectory.m

Using matlab generate trajectory by "JointTrajectory,m" in joint space and plot in CoppeliaSim.

- Generate trajectory in joint space
- Show output together with your MATLAB code.
- Plot the motion of Indy 7 manipulator by CoppeliaSim.

First import the matlab library.

```
% 20190348 Jungill Kang
addpath('..\mr')
clear; clc;
```

Observe JointTrajectory.m

```
*** CHAPTER 9: TRAJECTORY GENERATION ***
  Takes thetastart: The initial joint variables,
        thetaend: The final joint variables,
        Tf: Total time of the motion in seconds from rest to rest,
        N: The number of points N > 1 (Start and stop) in the discrete
           representation of the trajectory,
        method: The time-scaling method, where 3 indicates cubic
                (third-order polynomial) time scaling and 5 indicates
                quintic (fifth-order polynomial) time scaling.
  Returns traj: A trajectory as an N x n matrix, where each row is an
                n-vector of joint variables at an instant in time. The
                first row is thetastart and the Nth row is thetaend . The
                elapsed time between each row is Tf/(N - 1).
  The returned trajectory is a straight-line motion in joint space.
  Example Input:

  clear; clc;
  thetastart = [1; 0; 0; 1; 1; 0.2; 0; 1];
  thetaend = [1.2; 0.5; 0.6; 1.1; 2;2; 0.9; 1];
  Tf = 4;
  N = 6;
  method = 3;
  traj = JointTrajectory(thetastart, thetaend, Tf, N, method)

  Output:
  traj =
    1.0000         0         0    1.0000    1.0000    0.2000         0    1.0000
    1.0208    0.0520    0.0624    1.0104    1.1040    0.3872    0.0936    1.0000
    1.0704    0.1760    0.2112    1.0352    1.3520    0.8336    0.3168    1.0000
    1.1296    0.3240    0.3888    1.0648    1.6480    1.3664    0.5832    1.0000
    1.1792    0.4480    0.5376    1.0896    1.8960    1.8128    0.8064    1.0000
    1.2000    0.5000    0.6000    1.1000    2.0000    2.0000    0.9000    1.0000
```

Code for connecting CoppeliaSim Api and matlab.

```
clear all
close all
clc

sim=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
sim.simxFinish(-1); % just in case, close all opened connections
clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)
    disp('Connected to remote API server');

    %joints handles
    h = [0,0,0,0,0,0];
    for i=1:6
    [r, h(i)]= sim.simxGetObjectHandle(clientID, convertStringsToChars("Revolute_joint"+string(i)),
    end

    % n, duration has no big meaning
    n = 100;
    duration = 0.01;

    % You can change this code ----------------------------------------
    joint_pos_mat1 = [linspace(0, pi/2, n); linspace(0, -pi/3, n); linspace(0, pi/4, n);...
                      linspace(0,pi/3,n); linspace(0,pi/2,n); linspace(0,-pi/4,n)];
    joint_pos_mat2 = [linspace(pi/2, 0, n); linspace(-pi/3, 0, n); linspace(pi/4, 0, n);...
                      linspace(pi/3, 0, n); linspace(pi/2, 0, n); linspace(-pi/4, 0, n)];


    while true
        for i=1:n
            tstart = tic;
            for j=1:6
                sim.simxSetJointTargetPosition(clientID, h(j), joint_pos_mat1(j, i), sim.simx_opmod
            end
            dt = toc(tstart);
            if dt<duration
                pause(duration-dt);
            end
        end
        pause(3);

        for i=1:n
            tstart = tic;
            for j=1:6
                sim.simxSetJointTargetPosition(clientID, h(j), joint_pos_mat2(j, i), sim.simx_opmod
            end
            dt = toc(tstart);
            if dt<duration
                pause(duration-dt);
            end
        end
        pause(3);
    end
```

```
        % You can change this code ---------------------------------------
    else
        disp('Failed connecting to remote API server');
    end
    sim.delete(); % call the destructor!
    disp('Program ended');
```

Now create code which create joint trajectory using 3rd order polynomial time scaling.

Starts from initial joint variable (0,0,0,0,0,0) to final joint variable (0, pi/2, 0, 0, 0, 0)

```
thetastart = [0; 0; 0; 0; 0; 0;];
thetaend = [0; pi/2; 0; 0; 0; 0;];
Tf = 4;
N = 100;
method = 3;
traj1 = JointTrajectory(thetastart, thetaend, Tf, N, method)
```

```
traj1 = 100×6
         0         0         0         0         0         0
         0    0.0005         0         0         0         0
         0    0.0019         0         0         0         0
         0    0.0042         0         0         0         0
         0    0.0075         0         0         0         0
         0    0.0116         0         0         0         0
         0    0.0166         0         0         0         0
         0    0.0224         0         0         0         0
         0    0.0291         0         0         0         0
         0    0.0366         0         0         0         0
    ⋮
```

```
traj2 = JointTrajectory(thetaend, thetastart, Tf, N, method)
```

```
traj2 = 100×6
         0    1.5708         0         0         0         0
         0    1.5703         0         0         0         0
         0    1.5689         0         0         0         0
         0    1.5666         0         0         0         0
         0    1.5633         0         0         0         0
         0    1.5592         0         0         0         0
         0    1.5542         0         0         0         0
         0    1.5483         0         0         0         0
         0    1.5417         0         0         0         0
         0    1.5342         0         0         0         0
    ⋮
```

Here, traj1 is trajectroy starts from initial state to final state, and traj2 is trajectory for going back from final state to initial state.

And change the trajectory from above

```
sim=remApi('remoteApi'); % using the prototype file (remoteApiProto.m)
```

```
Note: always make sure you use the corresponding remoteApi library
(i.e. 32bit Matlab will not work with 64bit remoteApi, and vice-versa)
```

```
sim.simxFinish(-1); % just in case, close all opened connections
clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);

if (clientID>-1)
    disp('Connected to remote API server');

    %joints handles
    h = [0,0,0,0,0,0];
    for i=1:6
    [r, h(i)]= sim.simxGetObjectHandle(clientID, convertStringsToChars("Revolute_joint"+string(
    end

    % n, duration has no big meaning
    n = 100;
    duration = 0.01;

    % You can change this code ----------------------------------------
    % traj1 init --> final
    % traj2 final --> init

    % repeat two trajectories
    joint_pos_mat1 = traj1'
    joint_pos_mat2 = traj2'


    while true
        for i=1:n
            tstart = tic;
            for j=1:6
                sim.simxSetJointTargetPosition(clientID, h(j), joint_pos_mat1(j, i), sim.simx_
            end
            dt = toc(tstart);
            if dt<duration
                pause(duration-dt);
            end
        end
        pause(3);

        for i=1:n
            tstart = tic;
            for j=1:6
                sim.simxSetJointTargetPosition(clientID, h(j), joint_pos_mat2(j, i), sim.simx_
            end
            dt = toc(tstart);
            if dt<duration
```

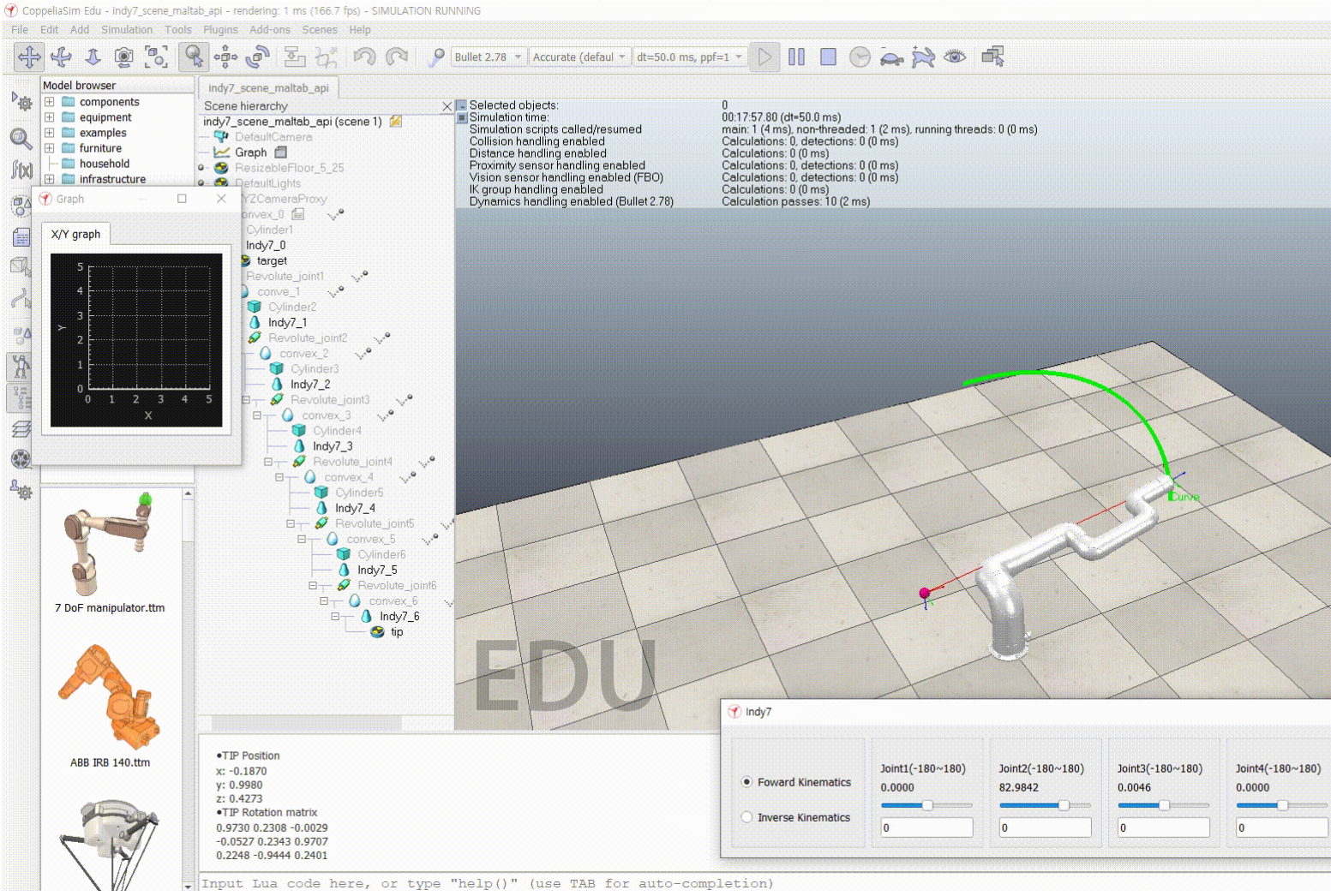
```
                pause(duration-dt);
            end
        end
        pause(3);
    end
    % You can change this code ---------------------------------------
else
    disp('Failed connecting to remote API server');
end
```

```
Failed connecting to remote API server
```

```
sim.delete(); % call the destructor!
disp('Program ended');
```

```
Program ended
```

Results



Since we only change the angle of joint 2 from 0 to pi/2 it draw quarter of circle.