Robot Simulation: By Max Lemon, Tejas Gulur, Atish Anantharam, and James Farrell

Table of Contents

Setup 1
Anonymous functions 1
Setting robot parameters (constants) 1
Performing Calculations 2
Performing animation 3

This code lets a user to put the amount of degrees (in radians) that a joint experiences for the four joints in the given robot. It will use the PoE method to give the end effector final position as well as show a graphical analysis of the robot moving

Setup

```
close all;  % Close any open figures
clear;  % Clear workspace
clc;    % Clear command window
I = eye(3); % Initialize 3x3 identity matrix
```

Anonymous functions

```
% Anonymous function to caclulate the skew matrix of a vector
skew = @(v) [0 -v(3) v(2); v(3) 0 -v(1); -v(2) v(1) 0];

% Anonymous function to calculate the matrix exponential using Rodrigues
% formula
rod = @(SKEW, V, T) [I+sin(T)*SKEW+(1-cos(T))*(SKEW^2) (I*T+(1-cos(T))*SKEW
+(T-sin(T))*(SKEW^2))*V'; 0 0 0 1];
```

Setting robot parameters (constants)

```
Me = [0 0 1 1; 0 -1 0 0; 1 0 0 0; 0 0 0 1];  % Home position of end-effector
n = 4;  % Number of joints

% Initializing the needed variables
M = cell(n,1);  %
% Home position of the different joints
w = cell(n,1);  % Rotation axes
q = cell(n,1);  % Displacement vectors
v = cell(n,1);  % Linear velocity vectors
theta = cell(n,1);  % Joint variables
del_theta = cell(n,1);  % Incremental change in theta (for use in animation)
s = cell(n,1);  % Skew axes
```

```
S = cell(n,1);  % Skew matrices
e_s = cell(n,1);    % Exponentials of skew matrices

% Home position of joints
M{1} = eye(4);
M{2} = [1 0 0 0; 0 0 -1 0; 0 1 0 0; 0 0 0 1];
M{3} = [0 -1 0 1; 0 0 -1 0; 1 0 0 0; 0 0 0 1];
M{4} = Me;

% Rotation axes of revolute joints
w{1} = [0 0 1];
w{2} = [0 -1 0];
w{3} = [0 -1 0];
w{4} = [0 0 0];

% Displacement of revolute joints
q{1} = [0 0 0];
q{2} = [0 0 0];
q{3} = [1 0 0];

% Direction of prismatic joints
v{4} = [1 0 0];

% Joint variables
theta{1} = pi/2;
theta{2} = pi/2;
theta{3} = pi/2;
theta{4} = pi/2;

% fprintf('Please provide the degree of change joint %d experiences: ', 1)
% theta{1} = input('');
% fprintf('Please provide the degree of change joint %d experiences: ', 2)
% theta{2} = input('');
% fprintf('Please provide the degree of change joint %d experiences: ', 3)
% theta{3} = input('');
% fprintf('Please provide the degree of change joint %d experiences: ', 4)
% theta{4} = input('');
```

Performing Calculations

```
% Finding linear velocity induced by rotation
for i=1:3
    v{i} = -cross(w{i}, q{i});
end

% Calculate skew axes, skew matrices, set each joint variable to pi/2, and
% find the matrix expoential for each joint
for i=1:n
    s{i} = [w{i}'; v{i}'];
    S{i} = [skew(w{i}) v{i}'; 0 0 0 0];
    e_s{i} = rod(skew(w{i}), v{i}, theta{i});
end
```

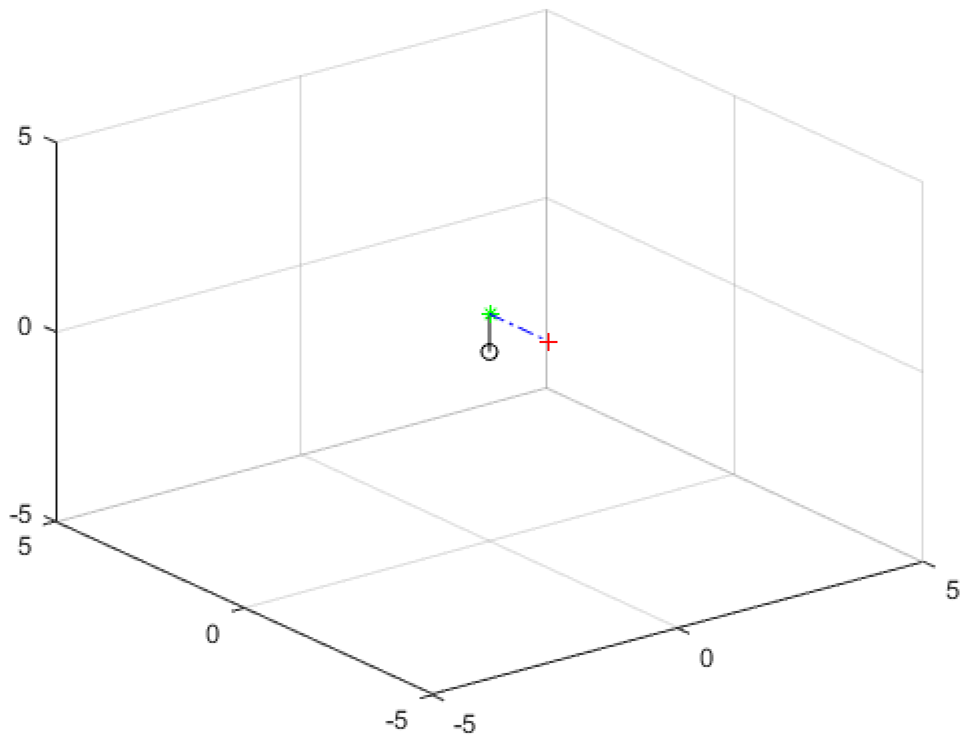
```
% Calculate the final transformation matrix
T = e_s{1}*e_s{2}*e_s{3}*e_s{4}*Me;

disp('End effector configuration: ')
disp(T)
```

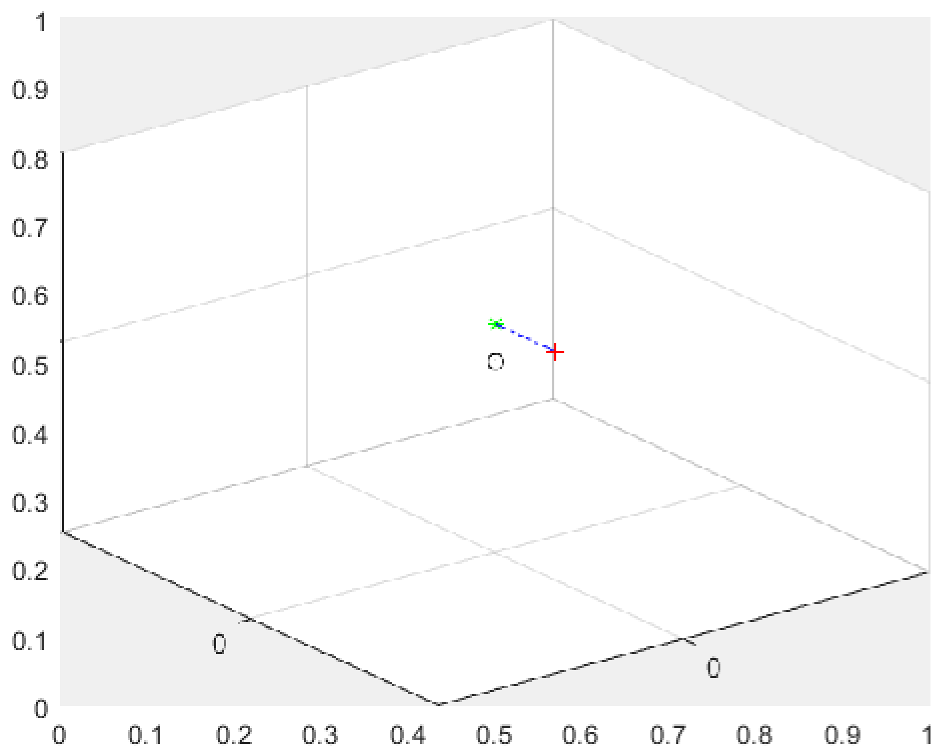
```
End effector configuration:
   -0.0000    1.0000   -0.0000   -0.0000
   -0.0000   -0.0000   -1.0000   -1.5708
   -1.0000         0    0.0000    1.0000
         0         0         0    1.0000
```

Performing animation

```
frames = 120;   % Number of frames for the animation
P = cell(n,1);  % Position of the joints
F(frames) = struct('cdata', [], 'colormap', []);    % Initialize strcut to
 store animation frames

% Joints 1 & 2 are always positioned at the origin
P{1} = [0 0 0];
P{2} = P{1};

% Determine the incremental values
for i=1:n
    del_theta{i} = theta{i}/frames;
end

% Calculate & plot values at each incremental value
for i=1:frames
    % Calculate the matrix exponetial for each joint at each increment
    for j=1:n
        theta{j} = i*del_theta{j};
        e_s{j} = rod(skew(w{j}), v{j}, theta{j});
    end

    % Find the position of joint 3 & 4, and end-effector
    T3 = e_s{1}*e_s{2}*M{3};    %Tranformation matrix for joint 3
    T4 = e_s{1}*e_s{2}*e_s{3}*M{4}; %Transformation matrix for joint 4
    T = e_s{1}*e_s{2}*e_s{3}*e_s{4}*Me;

    % Extract position data from transformation matrices
    P{3} = T3(1:3, 4)';
    P{4} = T4(1:3, 4)';
    Pe = T(1:3, 4)';

    % The line betwen joints 2 & 3
    L1{1} = [P{2}(1) P{3}(1)];  % x-coordinates
    L1{2} = [P{2}(2) P{3}(2)];  % y-coordinates
    L1{3} = [P{2}(3) P{3}(3)];  % z-coordinates

    % The line between joint 3 and the end-effector
```

```
    L2{1} = [P{3}(1) Pe(1)];  % x-coordinates
    L2{2} = [P{3}(2) Pe(2)];  % y-coordinates
    L2{3} = [P{3}(3) Pe(3)];  % z-coordinates

    % Clear the current plot
    clf;

    % Initialize plot parameters
    grid on;
    hold on;
    view(3);
    xlim([-5 5]);
    ylim([-5 5]);
    zlim([-5 5]);

    % Plot the different components
    plot3(0, 0, 0, 'ko')    % The origin, also location of J1 & J2
    plot3(L1{1}, L1{2}, L1{3}, '-k');   % Line between J2 & J3
    plot3(P{3}(1), P{3}(2), P{3}(3), '*g'); % Joint 3/4
    plot3(L2{1}, L2{2}, L2{3}, '-.b');  % Line between  J3 & EE
    plot3(Pe(1), Pe(2), Pe(3), '+r');   % End effector

    % Capture the current plot as a animation frame
    F(i) = getframe();
end

% Play back the animation
figure();
movie(F, 1, 60);
```


5
0
-5
5
0
-5
-5
0
5


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Published with MATLAB® R2022a