MATLAB_Exercise_1

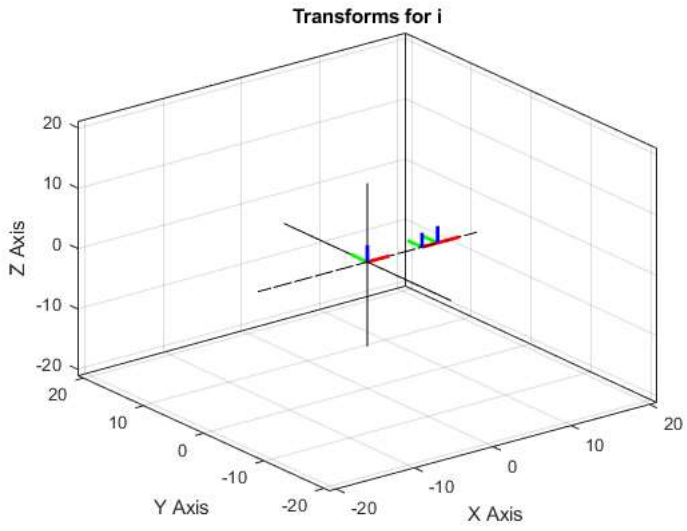
Michael White 3/26/2021

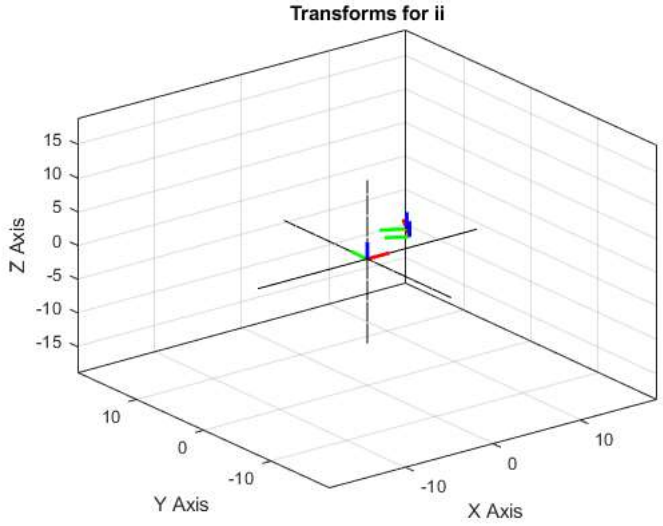
```
clear; clc; close all;

% PART A
    % Define symbolic parameters
    syms theta1 theta2 theta3 L1 L2 L3;

    % Define link parameter table
    linkParamTable = ...
        [0 0 0 theta1;
        0 L1 0 theta2;
        0 L2 0 theta3];

% PART B
    % Identify transformation matrices for each link (I will append the
    % functions I use to the end of this pdf)
    StepTransforms.Transform_01 = functions.links.Link2Transform(linkParamTable(1,:));
    StepTransforms.Transform_12 = functions.links.Link2Transform(linkParamTable(2,:));
    StepTransforms.Transform_23 = functions.links.Link2Transform(linkParamTable(3,:));

    % The transform to point H requires another row to the parameter table,
    % which I will append here:
    linkParamTable(4,:) = [0 L3 0 0];

    % Take new row and find appropriate transform:
    StepTransforms.Transform_3H = functions.links.Link2Transform(linkParamTable(4,:));

% PART C
    % For this part, I can use symbolic MATLAB in my functions to solve for the
    % symbolic transforms, then substitute the actual functions with the
    % replacement symbolics of ci and si with the subs function

    % Derive the Transform_03 and Transform_0H
    Transform_03.trig = functions.links.Link2Transform(linkParamTable(1:3,:));
    Transform_0H.trig = functions.links.Link2Transform(linkParamTable(1:4,:));

    % Convert trig functions to simplified syms
    syms c1 c2 c3 s1 s2 s3;
    Transform_03.simple = subs(Transform_03.trig,...
        [cos(theta1), cos(theta2), cos(theta3),...
         sin(theta1), sin(theta2), sin(theta3)],...
        [c1, c2, c3, s1, s2, s3]);
    Transform_0H.simple = subs(Transform_0H.trig,...
        [cos(theta1), cos(theta2), cos(theta3),...
         sin(theta1), sin(theta2), sin(theta3)],...
        [c1, c2, c3, s1, s2, s3]);

    % Convert trig combinations from angle sum formula to simplified versions
    syms c12 c23 s12 s23;
    Transform_03.simple = subs(Transform_03.simple,...
        [c1*c2-s1*s2, c2*c3-s2*s3, c1*s2+s1*c2, c2*s3+s2*c3],...
        [c12, c23, s12, s23]);
    Transform_0H.simple = subs(Transform_0H.simple,...
        [c1*c2-s1*s2, c2*c3-s2*s3, c1*s2+s1*c2, c2*s3+s2*c3],...
        [c12, c23, s12, s23]);

    % Convert trig combinations from angle sum formula to simplified versions
    syms c123 s123;
    Transform_03.simple = subs(Transform_03.simple,...
        [c3*c12-s3*s12, c3*s12+s3*c12],...
        [c123, s123]);
    Transform_0H.simple = subs(Transform_0H.simple,...
        [c3*c12-s3*s12, c3*s12+s3*c12],...
        [c123, s123]);

    % Display results from simplified transforms
    disp('The simplified transform 03:'); disp(Transform_03.simple);
    disp('The simplified transform 0H:'); disp(Transform_0H.simple);

    % Creating quantified transforms for 03 for scenario i, ii, and iii
    % Also: converting degrees to radians
    Transform_03.i = subs(Transform_03.trig, ...
        [theta1, theta2, theta3, L1, L2, L3], [0 0 0 4 3 2]);
    Transform_03.ii = subs(Transform_03.trig, ...
        [theta1, theta2, theta3, L1, L2, L3], [pi*10/180 pi*20/180 pi*30/180 4 3 2]);
    Transform_03.iii = subs(Transform_03.trig, ...
        [theta1, theta2, theta3, L1, L2, L3], [pi/2 pi/2 pi/2 4 3 2]);

    % Creating quantified transforms for 0H for scenario i, ii, and iii
    Transform_0H.i = subs(Transform_0H.trig, ...
        [theta1, theta2, theta3, L1, L2, L3], [0 0 0 4 3 2]);
    Transform_0H.ii = subs(Transform_0H.trig, ...
        [theta1, theta2, theta3, L1, L2, L3], [pi*10/180 pi*20/180 pi*30/180 4 3 2]);
    Transform_0H.iii = subs(Transform_0H.trig, ...
        [theta1, theta2, theta3, L1, L2, L3], [pi/2 pi/2 pi/2 4 3 2]);
```

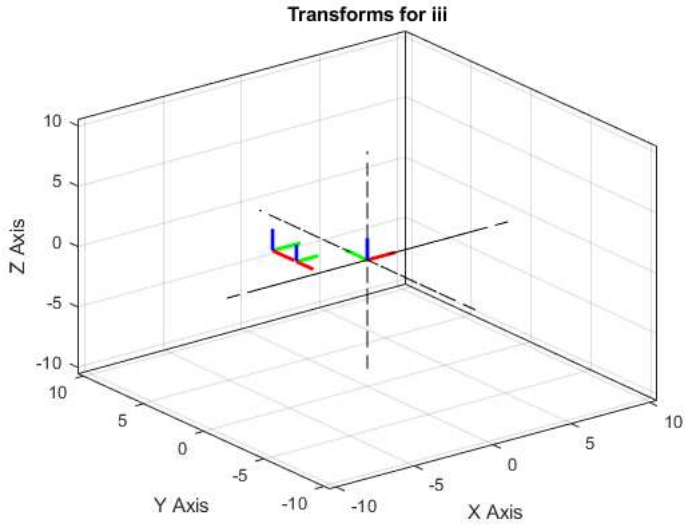
```
    % Display results for i, ii, and iii
    disp('The Transforms for i'); disp(Transform_03.i); disp(Transform_0H.i);
    disp('The Transforms for ii'); disp(Transform_03.ii); disp(Transform_0H.ii);
    disp('The Transforms for iii'); disp(Transform_03.iii); disp(Transform_0H.iii);

    % Plot the 03 and 0H transforms for i
    functions.visual.plotTransform(double(Transform_03.i),1);
    functions.visual.plotTransform(double(Transform_0H.i),1);
    title('Transforms for i');

    % Plot the 03 and 0H transforms for ii
    functions.visual.plotTransform(double(Transform_03.ii),2);
    functions.visual.plotTransform(double(Transform_0H.ii),2);
    title('Transforms for ii');

    % Plot the 03 and 0H transforms for iii
    functions.visual.plotTransform(double(Transform_03.iii),3);
    functions.visual.plotTransform(double(Transform_0H.iii),3);
    title('Transforms for iii');

% PART D: Testing with Corke (compare these to my 3d plots)
    % Scenario i
        L01_i = Link('d',0,'a',0,'alpha',0);
        L12_i = Link('d',0,'a',4,'alpha',0);
        L23_i = Link('d',0,'a',3,'alpha',0);
        L3H_i = Link('d',0,'a',2,'alpha',0);
        bot_i = SerialLink([L01_i L12_i L23_i L3H_i],'name','Scenario i');
        figure;
        bot_i.plot([0 0 0 0]); %Looks good!
    % Scenario ii
        L01_ii = Link('d',0,'a',0,'alpha',0);
        L12_ii = Link('d',0,'a',4,'alpha',0);
        L23_ii = Link('d',0,'a',3,'alpha',0);
        L3H_ii = Link('d',0,'a',2,'alpha',0);
        bot_ii = SerialLink([L01_ii L12_ii L23_ii L3H_ii],'name','Scenario ii');
        figure;
        bot_ii.plot([0 pi/180*10 pi/180*20 pi/180*30]); %Looks good!
    % Scenario iii
        L01_iii = Link('d',0,'a',0,'alpha',0);
        L12_iii = Link('d',0,'a',4,'alpha',0);
        L23_iii = Link('d',0,'a',3,'alpha',0);
        L3H_iii = Link('d',0,'a',2,'alpha',0);
        bot_iii = SerialLink([L01_iii L12_iii L23_iii L3H_iii],'name','Scenario iii');
        figure;
        bot_iii.plot([0 pi/2 pi/2 pi/2]); %Looks good!
```

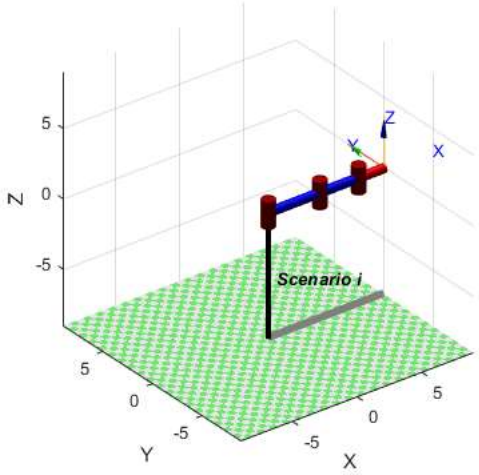
```
The simplified transform 03:
[c123, -s123, 0, L1*c1 + L2*c12]
[s123,  c123, 0, L1*s1 + L2*s12]
[   0,     0, 1,              0]
[   0,     0, 0,              1]

The simplified transform 0H:
[c123, -s123, 0, L1*c1 + L2*c12 + L3*c123]
[s123,  c123, 0, L1*s1 + L2*s12 + L3*s123]
[   0,     0, 1,                        0]
[   0,     0, 0,                        1]

The Transforms for i
[1, 0, 0, 7]
[0, 1, 0, 0]
[0, 0, 1, 0]
[0, 0, 0, 1]

[1, 0, 0, 9]
[0, 1, 0, 0]
[0, 0, 1, 0]
[0, 0, 0, 1]

The Transforms for ii
[(3^(1/2)*(cos(pi/9)*cos(pi/18) - sin(pi/9)*sin(pi/18)))/2 - (cos(pi/9)*sin(pi/18))/2 - (cos(pi/18)*sin(pi/9))/2, (sin(pi/9)*sin(pi/18))/2 - (cos(pi/9)*cos(pi/18))/
[(3^(1/2)*(cos(pi/9)*sin(pi/18) + cos(pi/18)*sin(pi/9)))/2 + (cos(pi/9)*cos(pi/18))/2 - (sin(pi/9)*sin(pi/18))/2, (3^(1/2)*(cos(pi/9)*cos(pi/18) - sin(pi/9)*sin(pi/
[                                                                                                                                                          0,
[                                                                                                                                                          0,

[(3^(1/2)*(cos(pi/9)*cos(pi/18) - sin(pi/9)*sin(pi/18)))/2 - (cos(pi/9)*sin(pi/18))/2 - (cos(pi/18)*sin(pi/9))/2, (sin(pi/9)*sin(pi/18))/2 - (cos(pi/9)*cos(pi/18))/
[(3^(1/2)*(cos(pi/9)*sin(pi/18) + cos(pi/18)*sin(pi/9)))/2 + (cos(pi/9)*cos(pi/18))/2 - (sin(pi/9)*sin(pi/18))/2, (3^(1/2)*(cos(pi/9)*cos(pi/18) - sin(pi/9)*sin(pi/
[                                                                                                                                                          0,
[                                                                                                                                                          0,

The Transforms for iii
[ 0, 1, 0, -3]
[-1, 0, 0,  4]
[ 0, 0, 1,  0]
[ 0, 0, 0,  1]

[ 0, 1, 0, -3]
[-1, 0, 0,  2]
[ 0, 0, 1,  0]
```

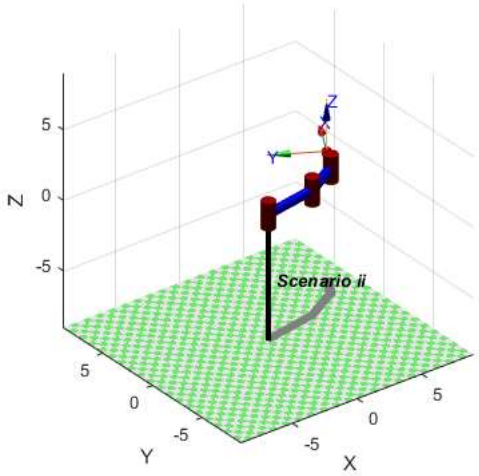
```
[ 0, 0, 0,  1]
```

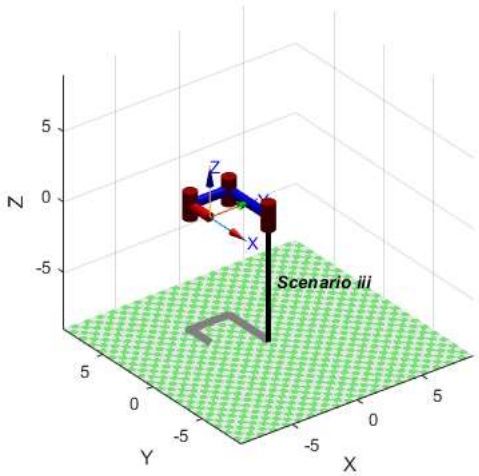

Transforms for i
Z Axis
Y Axis
X Axis


Transforms for ii
Z Axis
Y Axis
X Axis


Transforms for iii
Z Axis
Y Axis
X Axis







Published with MATLAB® R2020b

```
function [outputMatrix] = Link2Transform(linkMatrix)
% This function is intended to convert a link matrix table to an overall
% transform matrix.
%   Detailed explanation goes here
    arguments
        linkMatrix (:,4)
    end

    for i = 1:size(linkMatrix,1)

        alpha_i_minus_1 = linkMatrix(i,1);
        a_i_minus_1 = linkMatrix(i,2);
        d_i = linkMatrix(i,3);
        theta_i = linkMatrix(i,4);

        transformMatrix = ...
            [cos(theta_i),-sin(theta_i),0,a_i_minus_1; ...
            sin(theta_i)*cos(alpha_i_minus_1),cos(theta_i)*cos(alpha_i_minus_1),-sin
(alpha_i_minus_1),-d_i*sin(alpha_i_minus_1); ...
            sin(theta_i)*sin(alpha_i_minus_1),cos(theta_i)*sin(alpha_i_minus_1),cos
(alpha_i_minus_1),d_i*cos(alpha_i_minus_1); ...
            0, 0, 0, 1];
        if i > 1
            tempMatrix = tempMatrix*transformMatrix;
        else
            tempMatrix = transformMatrix;
        end
    end
    outputMatrix = tempMatrix;
end
```

```
function plotTransform(transformMatrix,figureNumber)
%PLOT3D This function looks to take a transform and move a frame, plotting
%the original position and the final position
oldFrame = eye(4);
newFrame = oldFrame;

newFrame = transformMatrix*newFrame;

% Generate old frame translation
oldFrameOrigin = functions.transform.positionFromTransform(oldFrame);

% Generate new frame properties
newFrameOrigin = functions.transform.positionFromTransform(newFrame);

% Generate sizing properties
axesSize(1) = newFrameOrigin(1)+5;
axesSize(2) = newFrameOrigin(2)+5;
axesSize(3) = newFrameOrigin(3)+5;
axesSize = max(abs(axesSize));
frameSize = 0.2*axesSize;
% frameSize = 1;

% Generate old frame rotations
oldFrameX = functions.transform.rotationFromTransform(oldFrame)*[frameSize 0 ↙
0]'+oldFrameOrigin;
oldFrameY = functions.transform.rotationFromTransform(oldFrame)*[0 frameSize ↙
0]'+oldFrameOrigin;
oldFrameZ = functions.transform.rotationFromTransform(oldFrame)*[0 0 ↙
frameSize]'+oldFrameOrigin;

% Generate new frame rotations
newFrameX = functions.transform.rotationFromTransform(newFrame)*[frameSize 0 ↙
0]'+newFrameOrigin;
newFrameY = functions.transform.rotationFromTransform(newFrame)*[0 frameSize ↙
0]'+newFrameOrigin;
newFrameZ = functions.transform.rotationFromTransform(newFrame)*[0 0 ↙
frameSize]'+newFrameOrigin;

% Setup plot
if ~exist('figureNumber','var')
    figure;
else
    figure(figureNumber);
end
axis = [-axesSize,axesSize];
xlim(1.5*axis);ylim(1.5*axis);zlim(1.5*axis);
xlabel('X Axis');ylabel('Y Axis');zlabel('Z Axis');
zero = [0,0];
plot3(axis,zero,zero,'--k');
```

```
hold on;
plot3(zero,axis,zero,'--k');
plot3(zero,zero,axis,'--k');
box on; grid on;

% Plot old coordinate frame
line([oldFrameOrigin(1) oldFrameX(1)],[oldFrameOrigin(2) oldFrameX(2)], ↙
[oldFrameOrigin(3) oldFrameX(3)],'Color','r','Linewidth',2);
line([oldFrameOrigin(1) oldFrameY(1)],[oldFrameOrigin(2) oldFrameY(2)], ↙
[oldFrameOrigin(3) oldFrameY(3)],'Color','g','LineWidth',2);
line([oldFrameOrigin(1) oldFrameZ(1)],[oldFrameOrigin(2) oldFrameZ(2)], ↙
[oldFrameOrigin(3) oldFrameZ(3)],'Color','b','LineWidth',2);

% Plot new coordinate frame
line([newFrameOrigin(1) newFrameX(1)],[newFrameOrigin(2) newFrameX(2)], ↙
[newFrameOrigin(3) newFrameX(3)],'Color','r','LineWidth',2);
line([newFrameOrigin(1) newFrameY(1)],[newFrameOrigin(2) newFrameY(2)], ↙
[newFrameOrigin(3) newFrameY(3)],'Color','g','LineWidth',2);
line([newFrameOrigin(1) newFrameZ(1)],[newFrameOrigin(2) newFrameZ(2)], ↙
[newFrameOrigin(3) newFrameZ(3)],'Color','b','LineWidth',2);
```