Advanced Control Laboratory (085705)
pre-Lab #1: Euclidian Transformations and Pose Definition

Daniel Engelsman # 300546173

1.a

```matlab
function R = RotRzRyRx( phi, theta, psi)
% This function calculates the corresponding rotation matrix from body
% position to global position, in Roll-Pitch-Yaw order

Rx = [  1           0          0       ;
        0        cos(phi)   -sin(phi)  ;
        0        sin(phi)    cos(phi)  ];

Ry = [ cos(theta)   0       sin(theta) ;
        0           1          0       ;
      -sin(theta)   0       cos(theta)];

Rz = [ cos(psi)  -sin(psi)     0       ;
       sin(psi)   cos(psi)     0       ;
        0           0          1      ];

R = Rz*Ry*Rx;

end
```

1.b

```matlab
function [psi, theta, phi] = rpyFromRot( R )
% This function calculates the corresponding euler angles out of an
% rotation matrix input, in Roll-Pitch-Yaw order

if ( R(3,1) ~= 1 && R(3,1) ~= -1 )          %&& R(3,1) != -1 )

    theta_1 = -asin( R(3,1) );
    psi_1 = atan2( R(3,2)/cos(theta_1) , R(3,3)/cos(theta_1) );
    phi_1 = atan2( R(2,1)/cos(theta_1) , R(1,1)/cos(theta_1) );
    theta = theta_1; psi = psi_1; phi = phi_1;

else
    phi = 0; % Initialize;
    if ( R(3,1) == -1 )
        theta = pi/2;
        psi = phi + atan2( R(1,2), R(1,3) );
    else
        theta = -pi/2;
        psi = -phi + atan2( -R(1,2), -R(1,3) );
    end
end
```

1.c

```matlab
%%                     main - Unit Test                     %%
%%
clc; clear all;

phi =    pi/4;
theta = -pi;
psi =   -pi/2;

%%
set = [psi, theta, phi];
my_Rotation = RotRzRyRx( set(3), set(2), set(1) )
matlab_Rotation = eul2rotm(set)

%%
% Matrix 2 Euler Re-Check

[ phi, theta, psi ] = rpyFromRot(matlab_Rotation);
my_results(1:3) = [ psi, theta, phi ]; my_results
matlab_results = rotm2eul(matlab_Rotation)
```

Command Window

```
my_Rotation =

   -0.0000    0.7071   -0.7071
    1.0000    0.0000    0.0000
    0.0000   -0.7071   -0.7071


matlab_Rotation =

   -0.0000    0.7071   -0.7071
    1.0000    0.0000    0.0000
    0.0000   -0.7071   -0.7071


my_results =

    1.5708   -0.0000   -2.3562


matlab_results =

    1.5708   -0.0000   -2.3562
```

As can be seen from right, Rotation matrix obtained from RotRzRyRx are equal to those obtained from Matlab function eul2rotm( ). Same true goes for checking validity of Inverse process, when using rpyFromRot returns same values as Matlab function rot2eul( ).

2.

```
%% 2. Calculate Position of Camera in respect to Global point

l_G = [450, 400, 50]';

R_C_G = [ 0.5363  -0.8440   0;
    0.8440   0.5363   0;
    0        0       1];

t_C_G = [-451.2459 257.0322 400]';

T_C_G = [ R_C_G      t_C_G;
    zeros(1,3)    1  ]

l_C = T_C_G*([l_G; 1])
```

Command Window

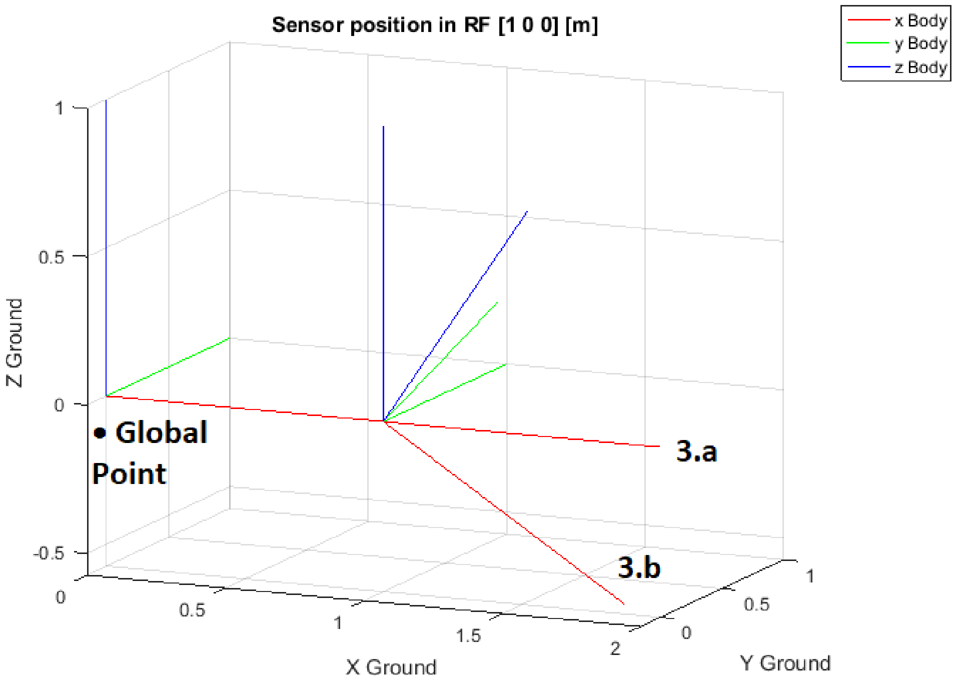
```
l_C =

 -547.5109
  851.3522
  450.0000
    1.0000
```

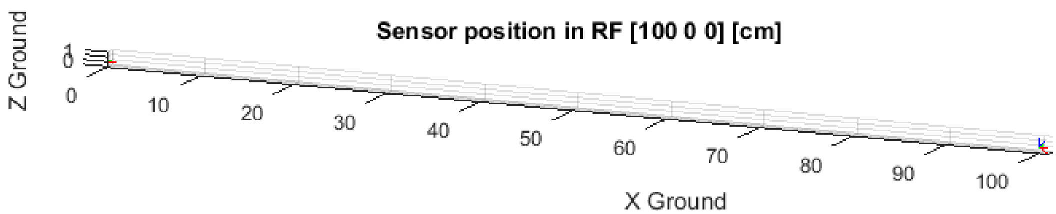
$$T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

inccorect transformation
[-6]

Answer: l_c = [-547.5109   851.3522   450]^T

3.

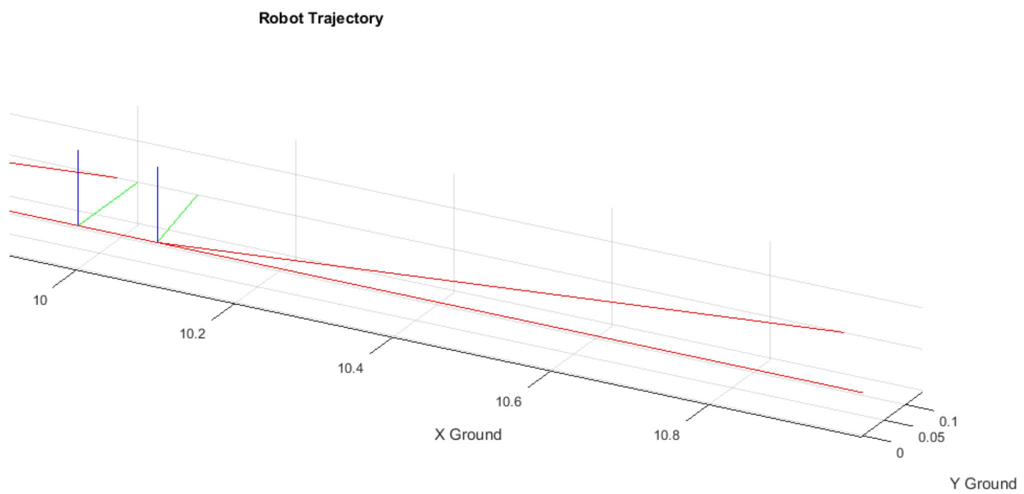




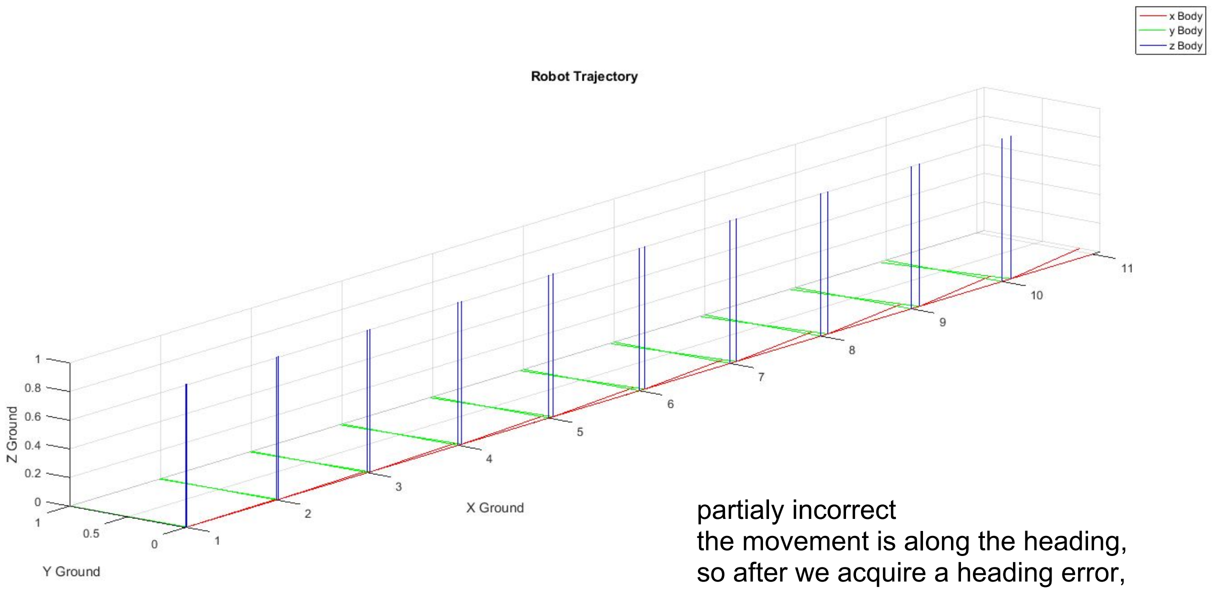
4.a

```
%% 4.a: Dead Rckoning: Recurssive Expression

clc; clear all; close all;

max_steps = 10;
t_err = 0; angle_err = deg2rad(1);  % Errors stemmed from Imperfectness
phi = 0; theta=0; psi=0;
t_s_G = [0 0 0]'; t_err = 0.01;

hold on;
for i=1:max_steps
    psi = psi + angle_err;
    t_s_G = t_s_G + [ (1+t_err) 0 0]';
    R_s_G = RotRzRyRx ( phi, theta, psi );
    T_s_G = [ R_s_G      t_s_G;
        zeros(1,3)    1  ];
    drawPose3(R_s_G, t_s_G, 1)
end
hold off;
```

4.b final Position after error: t’=[10.1 0 0]’ [m] and psi = 10 [deg]. Final Pose :



Actual vs. Commanded robot trajectory:



partialy incorrect
the movement is along the heading,
so after we acquire a heading error,
we should also see movement in the y axis.
[-8]