Problem 2

a)

Changes done to `MainKinematics`:

```
 7 %%%%%% MODIFY. Initial state values and parameter values
 8 state = [0;0;0];  % euler angles
 9 omega_ab_in_b = 2 * [1; 1; 1];
10
11 % Simulate dynamics
12 try
13     %%%%%% MODIFY THE FUNCTION "Kinematics" TO PRODUCE SIMULATIONS OF THE SOLID ORIENTATION
14     %%%%%%
15     %%%%%% Hints:
16     %%%%%% - "parameters" allows you to pass some parameters to the "Kinematic" function.
17     %%%%%% - "state" will contain representations of the solid orientation (SO(3)).
18     %%%%%% - use the "reshape" function to turn a matrix into a vector or vice-versa.
19
20     [time,statetraj] = ode45(@(t,x)Kinematics(t, x, omega_ab_in_b),[0, time_final],state);
```

```
46     omega = omega_ab_in_b;
47     R = Rotations(state_animate.');  % .' to avoid complex conjugates
```

b)

Changes done to `MainKinematicsDCM`:

```
 7 %%%%%% MODIFY. Initial state values and parameter values
 8 state = reshape(eye(3), [9,1]);
 9 omega_ab_in_b = 2 * [1; 1; 1];
10
11 % Simulate dynamics
12 try
13     %%%%%% MODIFY THE FUNCTION "Kinematics" TO PRODUCE SIMULATIONS OF THE SOLID ORIENTATION
14     %%%%%%
15     %%%%%% Hints:
16     %%%%%% - "parameters" allows you to pass some parameters to the "Kinematic" function.
17     %%%%%% - "state" will contain representations of the solid orientation (SO(3)).
18     %%%%%% - use the "reshape" function to turn a matrix into a vector or vice-versa.
19
20     [time,statetraj] = ode45(@(t,x)KinematicsDCM(t, x, omega_ab_in_b),[0, time_final],state);
```

```
46     omega = omega_ab_in_b;
47     R = reshape(state_animate, [3,3]);
```

```
function [ state_dot ] = Kinematics( t, state, omega_ab_in_b )
    % state_dot is time derivative of your state.
    % Hints:
    % - "parameters" allows you to pass some parameters to the "Kinematic" ↙
function.
    % - "state" will contain representations of the solid orientation (SO(3)).
    % - use the "reshape" function to turn a matrix into a vector or vice-versa.

    [R,M] = Rotations(state);
    state_dot = M \ omega_ab_in_b;
end

```

```
function [ state_dot ] = KinematicsDCM( t, state, omega_ab_in_b )
    % state_dot is time derivative of your state.
    % Hints:
    % - "parameters" allows you to pass some parameters to the "Kinematic" ↙
function.
    % - "state" will contain representations of the solid orientation (SO(3)).
    % - use the "reshape" function to turn a matrix into a vector or vice-versa.

    % t: time
    % state: reshaped R matrix in 9x1
    % omega_ab_in_b: rotation axis omega_ab in frame b
    %
    % state_dot: derivative of state reshaped to 9x1

    R = reshape(state, [3,3]);
    OmegaX = skewsym3x3(omega_ab_in_b);
    R_dot = R * OmegaX;
    state_dot = reshape(R_dot, [9,1]);
end

```

```
clear all
close all
clc

%%% FILL IN ALL PLACES LABELLED "complete"

syms rho theta psi real
syms drho dtheta dpsi real

A     = [rho;theta;psi];
dA    = [drho;dtheta;dpsi];

% rotation about x
R{1} = [1   0           0;
        0   cos(rho)    -sin(rho);
        0   sin(rho)    cos(rho)];

% rotation about y
R{2} = [cos(theta)  0   sin(theta);
        0           1   0;
        -sin(theta) 0   cos(theta)];

% rotation about z
R{3} = [cos(psi)    -sin(psi)   0;
        sin(psi)    cos(psi)    0;
        0           0           1];

%Rotation matrix
Rba = simplify(R{1} * R{2} * R{3});

%Time deriviatve of the rotation matrix (Hint: use
%the function "diff" to differentiate the matrix w.r.t. the angles
%rho, theta, psi one by one, and form the whole time derivative using the chain rule and summing the deriviatives)
dRba = diff(Rba, rho) * drho + diff(Rba, theta) * dtheta + diff(Rba, psi) * dpsi;

% Use the formulat relating Rba, dRba and OmegaX (skew-symmetric matrix underlying the angular velocity omega)
OmegaX_b = Rba.' * dRba;

% Extract the angular veloticy vector omega (3x1) from the matrix OmegaX (3x3)
omega = [OmegaX_b(3,2); OmegaX_b(1,3); OmegaX_b(2,1)];

% This line generates matrix M in the relationship omega = M*dA
M = jacobian(omega,dA)

% This line creates a Matlab function returing Rba and M for a given A  = [rho; theta;psi], can be called using [Rba,M] = Rotations(state);
matlabFunction(Rba,M,'file','Rotations','vars',{A})

```