TTK4130 Assignment 2 Eirik Falck

Problem 1

a) $SO(3) = \{ R \mid R \in \mathbb{R}^{3\times 3},\ R^T R = I,\ \det(R) = 1 \}$

$$R_1 = \begin{bmatrix} * & 1 & * \\ 1 & * & * \\ * & * & * \end{bmatrix} \qquad R_2 = \begin{bmatrix} 5/13 & * & * \\ * & 1 & * \\ 12/13 & * & * \end{bmatrix}$$

$$R_1^T R_1 = \begin{bmatrix} *' & 1 & *'' \\ 1 & \bar{*} & \bar{\bar{*}} \\ \underline{*} & \underline{\underline{*}} & \underline{\underline{\underline{*}}} \end{bmatrix} \begin{bmatrix} *' & 1 & \underline{*} \\ 1 & \bar{*} & \underline{\underline{*}} \\ *'' & \bar{\bar{*}} & \underline{\underline{\underline{*}}} \end{bmatrix} = \begin{bmatrix} *'^2 + *''^2 + 1 & 0 & 0 \\ 0 & 1 + \bar{*}^2 + \bar{\bar{*}}^2 & 0 \\ 0 & 0 & \underline{*}^2 + \underline{\underline{*}}^2 + \underline{\underline{\underline{*}}}^2 \end{bmatrix}$$

$$\Rightarrow R_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad \det(R_1) = -1 \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} = 1$$

$$R_2^T R_2 = \begin{bmatrix} 5/13 & r_{21} & 12/13 \\ r_{12} & 1 & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} \begin{bmatrix} 5/13 & \underset{0}{r_{12}} & r_{13} \\ r_{21}=0 & 1 & 0=r_{23} \\ 12/13 & \underset{0}{r_{32}} & r_{33} \end{bmatrix} \rightarrow R_2 = \begin{bmatrix} 5/13 & 0 & -\frac{12}{13} \\ 0 & 1 & 0 \\ 12/13 & 0 & \frac{5}{13} \end{bmatrix}$$

$$= \begin{bmatrix} \left(\frac{5}{13}\right)^2 + \cancel{r_{21}^2} + \left(\frac{12}{13}\right)^2 & & \\ \cancel{r_{12} \cdot \frac{5}{13}} + \cancel{r_{21}} + \cancel{\frac{12}{13} r_{32}} & \cancel{r_{12}^2} + \cancel{r_{32}^2} + 1 & \\ \frac{5}{13} r_{13} + \cancel{r_{23} r_{21}} + \frac{12}{13} r_{33} & \cancel{r_{13} r_{12}} + r_{23} + \cancel{r_{33} r_{32}} & r_{13}^2 + \cancel{r_{23}^2} + r_{33}^2 \end{bmatrix} = I_{3\times3}$$

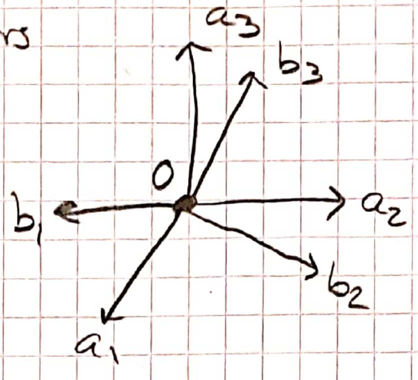
$$\det(R_2) = \frac{5}{13} \begin{vmatrix} 1 & \cancel{r_{23}} \\ \cancel{r_{32}} & r_{33} \end{vmatrix} - \cancel{r_{21} \begin{vmatrix} r_{12} & r_{13} \\ r_{32} & r_{33} \end{vmatrix}} + \frac{12}{13} \begin{vmatrix} \cancel{r_{12}} & r_{13} \\ 1 & \cancel{r_{23}} \end{vmatrix}$$

$$= \frac{5}{13} r_{33} - \frac{12}{13} r_{13} = 1, \quad r_{13}^2 + r_{33}^2 = 1$$

$$\rightarrow r_{13} = -\frac{12}{13} \quad r_{33} = \frac{5}{13}$$

b) $a = \{O, \vec{a}_1, \vec{a}_2, \vec{a}_3\}$ $\quad b = \{O, \vec{b}_1, \vec{b}_2, \vec{b}_3\}$

a_i, b_i : orthonormal vectors



$$R_b^a = [\vec{a}_i \cdot \vec{b}_j]$$

$$= \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix}$$

because a is orthonormal

$$= \begin{bmatrix} a_1 \cdot b_1 & a_1 \cdot b_2 & a_1 \cdot b_3 \\ a_2 \cdot b_1 & a_2 \cdot b_2 & a_2 \cdot b_3 \\ a_3 \cdot b_1 & a_3 \cdot b_2 & a_3 \cdot b_3 \end{bmatrix} = \begin{bmatrix} b_1^a & b_2^a & b_3^a \end{bmatrix}$$

The columns of R_b^a are the basis b represented in frame a coordinates.

c) $v^a = R_b^a v^b$

$$(u^a)^T v^a = (R_b^a u^b)^T R_b^a v^b = (u^b)^T \underbrace{(R_b^a)^T R_b^a}_{I} v^b = \underline{\underline{(u^b)^T v^b}} \quad \square$$

The geometrical interpretation of this is that the scalar product is invariant of the frame it is calculated in.

d) $(Ru)^\times = R u^\times R^T$, $v^a = R^a_b v^b$

$$u^a \times v^a = (u^a)^\times v^a = (R^a_b u^b)^\times R^a_b v^b$$

$$= R^a_b (u^b)^\times \underbrace{R^{a\,T}_b R^a_b}_{I} v^b = R^a_b (u^b)^\times v^b = \underline{\underline{R^a_b (u^b \times v^b)}} \quad \square$$

The interpretation is that the cross-product is dependent on the frame, but can be rotated from frame a to frame b by the rotation matrix R^a_b.

Problem 2

a) $R^a_b = R_1(\rho) R_2(\theta) R_3(\psi)$

$$\dot{R}^a_b = \frac{\partial R^a_b}{\partial \rho}\dot{\rho} + \frac{\partial R^a_b}{\partial \theta}\dot{\theta} + \frac{\partial R^a_b}{\partial \psi}\dot{\psi}$$

$$\dot{R}^a_b = (\omega^a_{ab})^\times R^a_b \rightarrow (\omega^a_{ab})^\times = \dot{R}^a_b (R^a_b)^T$$

$$(\omega^a_{ab})^\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \quad \omega^a_{ab} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = M \begin{bmatrix} \dot{\rho} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

The simulations give sensible results. We can see $\vec{\omega}_{ab}$ is stationary while the red frame b is spinning around it. Larger $\vec{\omega}_{ab}$ also gives faster rotations, as expected.

b) The results of the simulations are also here sensible. The same observations as in a) were observed.

Problem 2

a)

Changes done to `MainKinematics`:

```
%%%%%% MODIFY. Initial state values and parameter values
state = [0;0;0];  % euler angles
omega_ab_in_b = 2 * [1; 1; 1];

% Simulate dynamics
try
    %%%%%% MODIFY THE FUNCTION "Kinematics" TO PRODUCE SIMULATIONS OF THE SOLID ORIENTATION
    %%%%%%
    %%%%%% 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.

    [time,statetraj] = ode45(@(t,x)Kinematics(t, x, omega_ab_in_b),[0, time_final],state);
```

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

b)

Changes done to `MainKinematicsDCM`:

```
%%%%%% MODIFY. Initial state values and parameter values
state = reshape(eye(3), [9,1]);
omega_ab_in_b = 2 * [1; 1; 1];

% Simulate dynamics
try
    %%%%%% MODIFY THE FUNCTION "Kinematics" TO PRODUCE SIMULATIONS OF THE SOLID ORIENTATION
    %%%%%%
    %%%%%% 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.

    [time,statetraj] = ode45(@(t,x)KinematicsDCM(t, x, omega_ab_in_b),[0, time_final],state);
```

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

```matlab
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
```

```matlab
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 velocity 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})
```

Problem 3

$\|k\|^2 = 1$

Axis-angle representation: angle θ around axis k

$$R = R_{k,\theta} = \cos\theta\, I + \sin\theta\, k^\times + (1-\cos\theta) k k^T$$

a) $Rk = (\cos\theta\, I + \sin\theta\, k^\times + (1-\cos\theta)kk^T)k$

$$= \cos\theta\, k + \sin\theta \underbrace{k^\times k}_{0} + (1-\cos\theta)k\underbrace{k^T k}_{\|k\|^2 = 1}$$

$$= \cos\theta\, k + k - \cos\theta\, k = \underline{\underline{k}} \quad \square$$

This implies that a rotation of the axis doesn't change the vector, i.e. the axis k is an eigenvector of the rotation matrix.

b) shepperds(R1) => $\vec{k}_1 = \begin{bmatrix} 0,7071 \\ -0,7071 \\ 0 \end{bmatrix}$, $\theta_k = 3,1416$ rad

shepperds(R2) => $\vec{k}_2 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$, $\theta_2 = 1,1760$ rad

Comparing to compositions of rotation matrices corresponding to the above results, we can conclude that they are reasonable.

```matlab
function [k, theta] = shepperds(R)
% Calculate angle-axis representation of a rotation matrix using
% Shepperd's method, p. 236
%
% R:         rotation matrix
%
% k:         axis
% theta:     angle

%% Setup
T = trace(R);
r00 = T;
temp = num2cell(diag(R));
[r11, r22, r33] = temp{:};
ris = [r00 r11 r22 r33];

%% Step 1
[~, j] = max(ris);

%% Step 2
zii = sqrt(1 + 2 * ris(j) - T);

%% Step 3
% Sign not important for just finding *one* angle-axis representation

%% Step 4
z = zeros(4,1);
z(j) = zii;

if j == 1
    z(2) = (R(3,2) - R(2,3)) / z(1);
    z(3) = (R(1,3) - R(3,1)) / z(1);
    z(4) = (R(2,1) - R(1,2)) / z(1);
elseif j == 2
    z(1) = (R(3,2) - R(2,3)) / z(2);
    z(3) = (R(2,1) + R(1,2)) / z(2);
    z(4) = (R(1,3) + R(3,1)) / z(2);
elseif j == 3
    z(1) = (R(1,3) - R(3,1)) / z(3);
    z(2) = (R(1,3) + R(3,1)) / z(3);
    z(4) = (R(3,2) + R(2,3)) / z(3);
elseif j == 4
    z(1) = (R(2,1) - R(1,2)) / z(4);
    z(2) = (R(1,3) + R(3,1)) / z(4);
    z(3) = (R(3,2) + R(2,3)) / z(4);
end  % if

%% Step 5
n = z(1) / 2;
e = zeros(3,1);
for j = 1:3
    e(j) = z(j+1) / 2;
end  % for

%% Step 6: calculate angle and axis, see p. 231
theta = 2 * acos(n);
k = e / sin(theta / 2);
end  % function
```