TTK4130 Modeling and Simulation

Assignement 2

Ingebrigt Stamnes Reinsborg

1a) The definitions of SO(3):

$R^T R = R R^T = I$

$R^T = R^{-1}$

$\text{Det}(R) = \pm 1$

SO(3) is Non-Abelian

Seeing as R_1 must span $\mathbb{R}^3$ using three unit vectors of length 1, then the residual unknown parameters must be 0, except for a_{33}, which must be 1. This gives us that R_1 has three unit vectors of exact length 1.

$$R_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

R_2 is a bit trickier, but applying the same principles, we can calculate the first vector $\vec{b_1}$.

$$|\vec{b_1}| = \sqrt{\left(\frac{5}{13}\right)^2 + x^2 + \left(\frac{12}{13}\right)^2} = 1$$

$$1 - \left(\frac{5}{13}\right)^2 - \left(\frac{12}{13}\right)^2 = x^2 = 0$$

$$\vec{b_1} = \begin{bmatrix} \frac{5}{13} & 0 & \frac{12}{13} \end{bmatrix}^T$$

and $\vec{b_2}$ must be

$$\vec{b_2} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$$

$\vec{b_3}$ must be the cross-product of $\vec{b_1}$ and $\vec{b_2}$, or orthogonal to both.

$$\begin{bmatrix} \frac{5}{13} \\ 0 \\ \frac{12}{13} \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$$

$$= \begin{bmatrix} b_{12}b_{23} - b_{13}b_{22} \\ b_{13}b_{21} - b_{11}b_{23} \\ b_{11}b_{22} - b_{12}b_{21} \end{bmatrix} = \begin{bmatrix} 0 - \frac{12}{13} \\ 0 - 0 \\ \frac{5}{13} - 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{12}{13} \\ 0 \\ \frac{5}{13} \end{bmatrix}$$

$$\vec{b_3} = \begin{bmatrix} -\frac{12}{13} \\ 0 \\ \frac{5}{13} \end{bmatrix}$$

$$R_2 = \begin{bmatrix} \frac{5}{13} & 0 & -\frac{12}{13} \\ 0 & 1 & 0 \\ \frac{12}{13} & 0 & \frac{5}{13} \end{bmatrix}$$

b)

$$R_1 R_b^a = R_2$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} \frac{5}{13} & 0 & -\frac{12}{13} \\ 0 & 1 & 0 \\ \frac{12}{13} & 0 & \frac{5}{13} \end{bmatrix}$$

$$m_{21} = \frac{5}{13} \qquad m_{11} = 0 \qquad m_{31} = \frac{12}{13}$$

$$m_{22} = 0 \qquad m_{12} = 1 \qquad m_{32} = 0$$

$$m_{23} = -\frac{12}{13} \qquad m_{13} = 0 \qquad m_{33} = \frac{5}{13}$$

$$R_b^a = \begin{bmatrix} 0 & 1 & 0 \\ \frac{5}{13} & 0 & -\frac{12}{13} \\ \frac{12}{13} & 0 & \frac{5}{13} \end{bmatrix} \begin{pmatrix} \approx \vec{b}_2^{\,T} \\ \approx \vec{b}_3^{\,T} \\ \approx \vec{b}_1^{\,T} \end{pmatrix}$$

The columns correspond to the vectors $\vec{b_1}, \vec{b_2}$ and $\vec{b_3}$ in R_2

c) $(u^a)^T v^a = (u^b)^T v^b \quad \forall\, u, v \in \mathbb{R}^3$

The resulting scalars w^a and w^b must be the same number and it relates to the relation between the two vectors regardless of what basis they are in, or the length of the vectors multiplied by the cosine of the angle between them. This does of course not change if one multiplies with a rotational matrix which has been done here.

modsim Ass2 (forts.)

1d)

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

$$= R^a_b u^{b\times} \underbrace{R^{aT}_b \cdot R^a_b}_{I} v^b$$

$$= R^a_b u^{b\times} R^a_b v^b$$

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

$$= u^a \times v^a$$

2a) See picture of script.
The 3D-simulation is fairly reasonable, it spins in exactly the way one might expect

b) DCM and Euler Angles seem to do the same. Euler angles are a bit easier to work with, and DCM was a bit more "complex" I guess.
otherwise, see pics of script.

3a) $R = R_{K,\theta} = \cos\theta\, I + \sin\theta\, K^{\times} + (1-\cos\theta) K K^{T}$

Show that $K = RK$

We are rotating something around K, which means that the rotation won't change anything in terms of the objects position along the K-axis

If $K = RK$, then:

$i \neq Ri$

$j \neq Rj$

(for non-zero rotational matrix R)

notat:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \vec{V}_{\vec{\omega}K} = K \Rightarrow K^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$KK^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_{K,\theta} = \begin{bmatrix} \cos\theta & 0 & 0 \\ 0 & \cos\theta & 0 \\ 0 & 0 & \cos\theta \end{bmatrix} + \begin{bmatrix} 0 & -\sin\theta & 0 \\ \sin\theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1-\cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} (= R_3)(= R_{K,\theta})$$

$$R_{K,\theta} K = K = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \square$$

3b) Made a script, pls have a look.

+4 MainKinematic.m MakeArrow.m MakeFrame.m mArrow3.m SymbolicEuler.m Template3D.m +

```
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 Omega
%(skew-symmetric matrix underlying the angular velocity omega)
Omega = dRba*Rba';

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

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

% This line creates a Matlab function returning
%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})
```

+4 | Kinematics.m | MainKinematic.m | MakeArrow.m | MakeFrame.m | mArrow3.m | SymbolicEuler.m | +

```matlab
function [ state_dot ] = Kinematics( t, state, parameters )
    % state_dot is time derivative of your state.
    %for 2a)
    %state_dot = inv(M)*parameters

    %for 2b)
    omega_skew = [0 -parameters(3) parameters(2);
                    parameters(3) 0 -parameters(1);
                    -parameters(2) parameters(1) 0];
    dRba = reshape(state, [3,3])*omega_skew;
    state_dot = reshape(dRba, 9, 1);
    % 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.

    % Code your equations here...
end
```

unspeakablehorrors.m

```matlab
%Arbitrary rotational matrix
R = [0.788571    0.377143     0.485714;
 -0.337143    0.925714    -0.171429;
 -0.514286  -0.0285714    0.857143];

%Math
r_00 = trace(R);
T = r_00;
r_11 = R(1,1);
r_22 = R(2,2);
r_33 = R(3,3);
r_vec = [r_00;
        r_11;
        r_22;
        r_33];

syms z_0 z_1 z_2 z_3
z_0 = sqrt(1+2*r_00-T);
z_1 = sqrt(1+2*r_11-T);
z_2 = sqrt(1+2*r_22-T);
z_3 = sqrt(1+2*r_33-T);
z_num = [z_0; z_1; z_2; z_3];
r_ii = max(r_vec);
z_i = abs(sqrt(1+2*r_ii)-r_00);

n_withBigDick = z_0 / 2;
epsilon_i = 0.5 * [z_1;
                    z_2;
                    z_3];

%Result
eulerParameters = [n_withBigDick;
                    epsilon_i];
```