Out: January 20, 2020
Due: February 2, 2020

NTNU – Trondheim
Norwegian University of Science and Technology

TTK4130 Modeling and Simulation
Assignment 2

Introduction

Rotation matrices and angular velocity vectors are fundamental for describing the kinematics of a body's orientation. In the context of control theory, these quantities are usually described using parametrizations, such as Euler angles or unit quaternions, as explained in the book. Reasons for this approach are to reduce the computational complexity in real-time systems, and to relate the measurements as directly as possible to the states. However, in the context of modeling and simulation, a theoretically simpler and more direct approach can be used to simulate the rotation of a body, as we will study in Problem 2.

For this problem and the course in general, it will be very useful to make use of the Matlab Symbolic Math Toolbox™. The main commands of this toolbox include:

- `syms x y z real` declares scalar symbolic variables `x,y,z`.
- `A = sym('A',[n,m])` creates a matrix `A` of symbolic variables of size $n \times m$.
- `matlabFunction(expr1,expr2,'File','myfilename','Vars',t,[x y z])` exports a Matlab function called `'myfilename'`, which takes the two arguments specified by `t` and `[x y z]`. The function has (up to) two output arguments, evaluated as the symbolic expressions `expr1` and `expr2`. After the execution of this line, a new file `'myfilename'` will be created on your working folder with the corresponding function. This function can be called using `[expr1,expr2] = myfilename(t,[x,y,z])`, where `t` and `[x,y,z]` are numerical values, and the numerical evaluations of `expr1` and `expr2` are returned.

In order to get used to manipulate this toolbox, you ought to try these different commands.

Problem 1 (Rotation matrices)

(a) Use the definition of SO(3) to find the missing values in the following rotation matrices:

$$\mathbf{R}_1 = \begin{bmatrix} * & 1 & * \\ 1 & * & * \\ * & * & * \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} \frac{5}{13} & * & * \\ * & 1 & * \\ \frac{12}{13} & * & * \end{bmatrix}.$$

Solution: Since $\mathbf{R}_i \in SO(3)$, then the columns of $\mathbf{R}_i$ have norm 1 and are orthogonal to each other (the scalar product is zero). This also applies to the rows of $\mathbf{R}_i$. Furthermore, these orthonormal bases are right-handed, i.e. $\det \mathbf{R}_i = 1$.

These properties give

$$\mathbf{R}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{R}_2 = \begin{bmatrix} \frac{5}{13} & 0 & -\frac{12}{13} \\ 0 & 1 & 0 \\ \frac{12}{13} & 0 & \frac{5}{13} \end{bmatrix}.$$

Let $a = \{O, \vec{a}_1, \vec{a}_2, \vec{a}_3\}$ and $b = \{O, \vec{b}_1, \vec{b}_2, \vec{b}_3\}$ be two reference frames, where O is the common origin, $\vec{a}_1, \vec{a}_2, \vec{a}_3$ are the orthogonal unit vectors that give the axes of frame a, and $\vec{b}_1, \vec{b}_2, \vec{b}_3$ are the orthogonal unit vectors that give the axes of frame b.

Consider the rotation matrix from a to b, $\mathbf{R}_b^a$.

(b) The columns of $\mathbf{R}_b^a$ are the coordinates of some particular vectors in some particular frame. What vectors and what frame are these? Explain.

Solution: The first, second and third column of $\mathbf{R}_b^a$ are the coordinates of $\vec{b}_1$, $\vec{b}_2$ and $\vec{b}_3$ in the frame a, respectively. This follows from the definition of $\mathbf{R}_b^a$ since this matrix is the coordinate transformation matrix from frame b to frame a, i.e. $\mathbf{u}^a = \mathbf{R}_b^a \mathbf{u}^b$.
In symbols, we have that

$$\mathbf{R}_b^a = \begin{bmatrix} \mathbf{b}_1 \cdot \mathbf{a}_1 & \mathbf{b}_2 \cdot \mathbf{a}_1 & \mathbf{b}_3 \cdot \mathbf{a}_1 \\ \mathbf{b}_1 \cdot \mathbf{a}_2 & \mathbf{b}_2 \cdot \mathbf{a}_2 & \mathbf{b}_3 \cdot \mathbf{a}_2 \\ \mathbf{b}_1 \cdot \mathbf{a}_3 & \mathbf{b}_2 \cdot \mathbf{a}_3 & \mathbf{b}_3 \cdot \mathbf{a}_3 \end{bmatrix}.$$

(c) Show that $(\mathbf{u}^a)^T \mathbf{v}^a = (\mathbf{u}^b)^T \mathbf{v}^b$ for all vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$.
What is the geometrical interpretation of this identity?

Solution:

$$(\mathbf{u}^a)^T \mathbf{v}^a = (\mathbf{R}_b^a \mathbf{u}^b)^T (\mathbf{R}_b^a \mathbf{v}^b) = (\mathbf{u}^b)^T (\mathbf{R}_b^a)^T \mathbf{R}_b^a \mathbf{v}^b = (\mathbf{u}^b)^T \mathbf{v}^b.$$

This means that the scalar product of two vectors is invariant under rotations.

(d) For any rotation matrix $\mathbf{R}$ and vector $\mathbf{u}$, we have that $(\mathbf{R}\mathbf{u})^\times = \mathbf{R}\mathbf{u}^\times \mathbf{R}^T$.
Use this identity to show that $\mathbf{u}^a \times \mathbf{v}^a = \mathbf{R}_b^a (\mathbf{u}^b \times \mathbf{v}^b)$.
What is the geometrical interpretation of this identity?

Solution:

$$\begin{aligned} \mathbf{u}^a \times \mathbf{v}^a &= (\mathbf{R}_b^a \mathbf{u}^b) \times (\mathbf{R}_b^a \mathbf{v}^b) = (\mathbf{R}_b^a \mathbf{u}^b)^\times (\mathbf{R}_b^a \mathbf{v}^b) \\ &= \mathbf{R}_b^a (\mathbf{u}^b)^\times (\mathbf{R}_b^a)^T \mathbf{R}_b^a \mathbf{v}^b = \mathbf{R}_b^a (\mathbf{u}^b)^\times \mathbf{v}^b = \mathbf{R}_b^a (\mathbf{u}^b \times \mathbf{v}^b). \end{aligned}$$

This means that the rotation of a cross product is equal to the cross product of the rotations.

Problem 2 (Angular velocities)

In this task, we will simulate the kinematic of a body's orientation for a given angular velocity $\vec{\omega}$. We will simulate the kinematic equations using the Matlab ODE integration function `ode45` (which we will discuss later in the course). You will find code templates on Blackboard to help you get started. `MainKinematic.m` provides a template for building the simulation and a 3D animation of the results. `Kinematics.m` is a Matlab function where the derivative of the state, $\dot{x}$, is to be calculated as a function of the state and other parameters. For more information, see the file `ReadMe.txt`.

(a) Consider an SO(3) representation using Euler angles. More specifically we choose:

$$R_b^a = R_1(\rho) R_2(\theta) R_3(\psi), \tag{1}$$

where

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\rho) & -\sin(\rho) \\ 0 & \sin(\rho) & \cos(\rho) \end{bmatrix}, \quad R_2 = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \quad R_3 = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In order to compute $\dot{\rho}$, $\dot{\theta}$, $\dot{\psi}$ as a function of ρ, θ, ψ and $\vec{\omega}$, complete the code of `SymbolicEuler.m`. The routine `SymbolicEuler.m` builds a Matlab function that delivers $R_b^a = R_b^a(\rho, \theta, \psi)$ and a ma-

trix $M = M(\rho, \theta, \psi)$ that fulfills

$$\boldsymbol{\omega}_{ab}^{b} = M \begin{bmatrix} \dot{\rho} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{2}$$

The parts of the code that are missing are labelled with `complete`.

Finally, implement `Kinematics.m` and modify `MainKinematic.m` in order to simulate the body kinematics using different fixed vectors $\boldsymbol{\omega}_{ab}^{b}$.

Explain your modifications, and add them to your answer.

Try different fixed vectors $\boldsymbol{\omega}_{ab}^{b}$. Do the 3D-simulations give reasonable results? Explain.

Solution:

Implementation of `SymbolicEuler.m`:

```
clear all
close all
clc

syms rho theta psi real
syms drho dtheta dpsi real

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

R{1} = [1        0         0;
        0 cos(A(1)) -sin(A(1));
        0 sin(A(1))  cos(A(1))];

R{2} = [cos(A(2))  0  sin(A(2));
           0       1      0;
       -sin(A(2)) 0  cos(A(2))];

R{3} = [cos(A(3))  -sin(A(3))      0;
        sin(A(3))   cos(A(3))      0;
            0           0          1];

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

dRba = 0;
for k = 1:3
    dRba = dRba + diff(Rba,A(k))*dA(k);
end

Omega = simplify(Rba.'*dRba);

omega = [Omega(3,2);
         Omega(1,3);
         Omega(2,1)];

M = jacobian(omega,dA);    % omega = M*dA
```

```
37
38 % Creates a function that can be called using
39 % "[Rba,M] = Rotations(state);"
40 matlabFunction(Rba,M,'file','Rotations','vars',{A})
```

The dynamic system to be simulated is

$$\begin{bmatrix} \dot{\rho} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = M^{-1}\boldsymbol{\omega}_{ab}^{b}.$$

Hence, the state is $\mathbf{x} = [\rho, \theta, \psi]^T$ and the parameter is $\boldsymbol{\omega}_{ab}^{b}$.
With this in mind, the implementation of `Kinematics.m` becomes

```
function [ state_dot ] = Kinematics( t, state, parameters )
    omega = parameters;
    [R,M] = Rotations(state);
    state_dot = M\omega;
end
```

In `MainKinematic.m`, the initial state values and parameter values are for example:

```
%%%%%% MODIFY. Initial state values and parameter values
state = [0;0;0];
parameters = [1;1;1];
```

While `omega` and `R` are given by

```
    %%%%%% MODIFY THE FOLLOWING LINES TO PRODUCE AN "omega" AND "R"
        FROM YOUR SIMULATION STATE
    omega = parameters;
    R = Rotations(state_animate.');
```

The cube rotates according to the constant angular velocity vector $\boldsymbol{\omega}_{ab}^{b}$.

(b) We will now consider a direct representation of R_b^a, which is often called **Direct Cosine Matrix** (DCM). Here, we will simply carry the entire matrix R_b^a as a state of the system, and proceed to simulate the dynamics:

$$\dot{R}_b^a = R_b^a \left(\boldsymbol{\omega}_{ab}^{b}\right)^{\times} \tag{3}$$

Implement the DCM approach in Matlab by modifying `Kinematics.m` and modify `MainKinematic.m` accordingly, and perform the same simulations as you did in the previous part.

Explain your modifications, and add them to your answer.

Do the 3D-simulations give reasonable results? Explain.

Compare the Euler angles and the Direct Cosine Matrix approaches.

Hint: `ode45` only works with state vectors. Therefore, we recommend to use the Matlab function `reshape` to convert R_b^a back-and-forth between a 3×3 matrix and a 9×1 vector.

Solution:

Implementation of `Kinematics.m`:

```
function [ state_dot ] = Kinematics( t, state, parameters )
```

```
    omega = parameters;
    Omega = [    0,      -omega(3),  +omega(2);
              +omega(3),      0,      -omega(1);
              -omega(2), +omega(1),       0];
    R          = reshape(state,[3,3]);
    state_dot = reshape(R*Omega,[9,1]);
end
```

Modification 1/2 of `MainKinematic.m`:

```
%%%%%% MODIFY. Initial state values and parameter values
state = reshape(eye(3),[9,1]);
parameters = [1;0;1];
```

Modification 2/2 of `MainKinematic.m`:

```
    %%%%%% MODIFY THE FOLLOWING LINES TO PRODUCE AN "omega" AND "R"
        FROM YOUR SIMULATION STATE
    omega = parameters;
    R     = reshape(state_animate,[3,3]);
```

The simulation results are the same as in part a, as expected.

Both approaches can be used for modeling. The DCM approach is more direct and has simpler equations. On the other side, the Euler angles approach is better suited for control since it has less states, and the states can be easily related to measurements in most cases. However, the model equations are complicated and singular at some angle values.

Problem 3 (Angle-axis representation, Sheeperd's method)

Any rotation matrix can be represented as a rotation by an angle θ about an axis $\mathbf{k}$. This is known as the **angle-axis representation**. Moreover, the rotation matrix $\mathbf{R}$ can be written as

$$\mathbf{R} = \mathbf{R}_{\mathbf{k},\theta} = \cos\theta\mathbf{I} + \sin\theta\mathbf{k}^{\times} + (1-\cos\theta)\mathbf{k}\mathbf{k}^{T}. \tag{4}$$

(a) Show that $\mathbf{k} = \mathbf{R}\mathbf{k}$.

What is the geometrical interpretation of this identity?

Solution:

$$\begin{aligned}\mathbf{R}\mathbf{k} = \mathbf{R}_{\mathbf{k},\theta}\mathbf{k} &= \cos\theta\mathbf{k} + \sin\theta\mathbf{k}^{\times}\mathbf{k} + (1-\cos\theta)\mathbf{k}\mathbf{k}^{T}\mathbf{k} \\ &= \cos\theta\mathbf{k} + (1-\cos\theta)\mathbf{k}(\mathbf{k}^{T}\mathbf{k}) = \cos\theta\mathbf{k} + (1-\cos\theta)\mathbf{k} = \mathbf{k}.\end{aligned}$$

This means that $\mathbf{k}$ is an invariant direction of the rotation matrix $\mathbf{R}$. This is expected since $\mathbf{k}$ is the rotation axis of $\mathbf{R}$.

Shepperd's method is an algorithm for calculating the angle-axis representation of a rotation matrix (see section 6.7 in the book).

(b) Implement a Matlab function that calculates the rotation angle θ and the rotation axis $\mathbf{k}$ for an arbitrary rotation matrix $\mathbf{R}$. Add the Matlab script to your answer.

Moreover, find the rotation axis and rotation angle for each of the rotation matrices in part 1.a.

Are the obtained results reasonable? Explain.

Solution:

```
function  [theta, k] = getAngleAxis(R)
    r11 = R(1,1);
    r22 = R(2,2);
    r33 = R(3,3);
    r00 = r11 + r22 + r33;
    [rii,i] = max([r00, r11, r22, r33]); i = i-1;
    if i == 0,
        z0 = sqrt(1 + 2*rii - r00);
        z(1) = (R(3,2)-R(2,3))/z0;
        z(2) = (R(1,3)-R(3,1))/z0;
        z(3) = (R(2,1)-R(1,2))/z0;
    elseif i == 1,
        z(1) = sqrt(1 + 2*rii - r00);
        z0 = (R(3,2)-R(2,3))/z(1);
        z(2) = (R(2,1)+R(1,2))/z(1);
        z(3) = (R(1,3)+R(3,1))/z(1);
    elseif i == 2,
        z(2) = sqrt(1 + 2*rii - r00);
        z0 = (R(1,3)-R(3,1))/z(2);
        z(1) = (R(2,1)+R(1,2))/z(2);
        z(3) = (R(3,2)+R(2,3))/z(2);
    elseif i == 3,
        z(3) = sqrt(1 + 2*rii - r00);
        z0 = (R(2,1)-R(1,2))/z(3);
        z(1) = (R(1,3)+R(3,1))/z(3);
        z(2) = (R(3,2)+R(2,3))/z(3);
    end
    % Euler parameters
    eta = z0/2;
    epsilon = [z(1); z(2); z(3)]/2;
    % Angle axis
    theta= 2*acos(eta);
    k = epsilon/(sin(theta/2));
end
```

$$\theta_1 = 3.1416 \simeq \pi, \quad \mathbf{k}_1 = \begin{bmatrix} 0.7071 \\ 0.7071 \\ 0 \end{bmatrix} \simeq \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{bmatrix}, \quad \theta_2 = 1.1760 \simeq \arccos\left(\frac{5}{13}\right), \quad \mathbf{k}_2 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}.$$

Yes, the results are reasonable.

$\mathbf{R}_2$ is a rotation by $-\arccos\left(\frac{5}{13}\right)$ about the y-axis. Since Shepperd's method only gives rotation angles between 0 and 2π, necessarily $\theta_2 = \arccos\left(\frac{5}{13}\right)$ and $\mathbf{k}_2 = [0, -1, 0]^T$.

The rotation given by $\mathbf{R}_1$ is more difficult to visualize. However, it is immediate to check that $\mathbf{k}_1$ is indeed the invariant direction of $\mathbf{R}_1$, and that $\mathbf{R}_1$ represents a rotation by $\theta_2 = \pi$ about $\mathbf{k}_1$.