Out: January 27, 2020
Due: February 9, 2020

NTNU – Trondheim
Norwegian University of
Science and Technology

TTK4130 Modeling and Simulation

Assignment 3

Introduction

The objectives of this assignment are:

- To understand the Newton-Euler equations, and apply them to simple mechanical systems.
- To learn how to select convenient $\mathrm{SO}(3)$ representations and reference frames in order to simplify the associated Newton-Euler equations.
- To understand and apply the parallel axis theorem, also known as Huygens–Steiner theorem.

Problem 1 (Satellite)

In this task, we will consider a satellite orbiting Earth. We define an inertial reference frame with its origin at Earth's center and with an arbitrary and fixed orientation.

We will consider two cases:

(a) The satellite is a cube of uniform, unitary density, having an edge of 50cm.

(b) The satellite is the cube mentioned above, with the addition of a punctual mass of $m_0 = 0.1\text{kg}$ placed at one of the cube's corners.

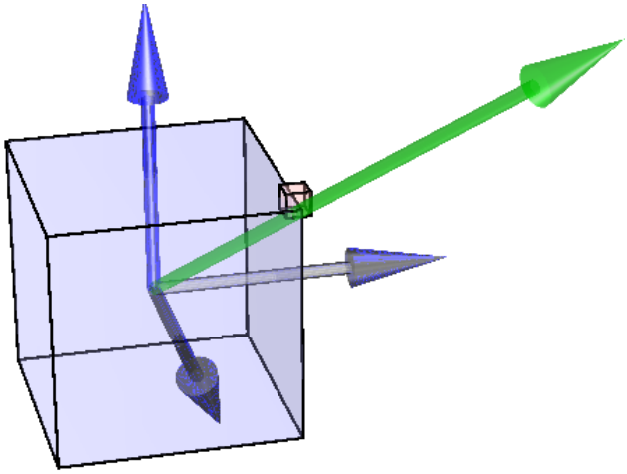

Figure 1: Schematic of the satellite.

We will assume that the force of gravity is given by Newton's law of universal gravitation:

$$\vec{F} = -\frac{G\, m_{\mathrm{T}}\, m}{\|\vec{r}_{\mathrm{c}}\|^2} \cdot \frac{\vec{r}_{\mathrm{c}}}{\|\vec{r}_{\mathrm{c}}\|} \tag{1}$$

The inertia matrix in the reference frame attached to the cube with its origin at the cube's center of mass and with the axes going through the center of the cube's faces is given by $\frac{1}{6}ml^2 I$, where m is the mass, l is the length of the sides and I is the 3-by-3 identity matrix.

For both cases, select a frame for the satellite and a representation of the $\mathrm{SO}(3)$ Lie group (orientation). Then apply correctly the Newton-Euler equations to describe the satellite's motion (position and orientation).

What is your resulting state-space model?

Complete the function `SatelliteDynamics.m` accordingly, and add it to your answer.

Simulate your equations for both cases using the Matlab ODE integration function `ode45` (which we will discuss later in the course). An example code is provided on Blackboard. You will find in the code means to show a 3D animation of your simulation.

What do you observe? Are the results reasonable? What is the main difference between both cases? Explain.

Hints:

- *Note that in case (b), the added punctual mass will shift the position of the center of mass of the satellite from the center of the cube. You may need the parallel axes theorem.*
- *The selection of a convenient body frame and* $\mathrm{SO}(3)$ *representation will make the problem much easier.*
- *You will find code templates / examples on Blackboard to help you get started. See* `Satellite3DTemplate.m` *for a template on how to build the simulation, and* `Satellite3DExample.m` *for tools to do 3D animations. These animations will allow you to assess your simulations.*
- *For parameter and initial values that are not given, you are free to choose reasonable numerical values. For example, Earth's radius is* $6356\,\mathrm{km}$ *and its orbital height is* $36\,\mathrm{Mm}$.

Solution: The satellite moves freely in space. Hence, its movement has all 6 degrees of freedom.

Since the distance between the satellite and Earth is gargantuan compared to the size of the satellite, we can assume that the gravitational force acts on the satellite as it is a point mass.

Since the gravitational force is the only force acting on the satellite, it is a good idea to place the center of the body frame at the satellite's center of mass. Moreover, we use choose the following states:

- $\mathbf{r}_c^i$: The position of the satellite's center of mass in the inertial frame.
- $\mathbf{v}_c^i$: The velocity of the satellite's center of mass in the inertial frame.
- $\mathbf{R}_b^i$: The rotation matrix from the inertial frame to the body frame. This matrix gives the orientation of the satellite.
- $\boldsymbol{\omega}_{ib}^b$: The coordinates of the angular velocity vector in the body frame.

Therefore, Newton-Euler's equations, the definition of velocity and the properties of the rotation matrices give the following differential equations:

$$\begin{aligned}
\dot{\mathbf{r}}_c^i &= \mathbf{v}_c^i \\
\dot{\mathbf{R}}_b^i &= \mathbf{R}_b^i \left(\boldsymbol{\omega}_{ib}^b\right)^\times \\
\dot{\mathbf{v}}_c^i &= \frac{\mathbf{F}^i}{m} = -\frac{G\, m_\mathrm{T}}{\|\mathbf{r}_c^i\|^2} \cdot \frac{\mathbf{r}_c^i}{\|\mathbf{r}_c^i\|} \\
\mathbf{M}_c^b \dot{\boldsymbol{\omega}}_{ib}^b &= -\left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_c^b \boldsymbol{\omega}_{ib}^b,
\end{aligned}$$

where G is the universal constant of gravitation, m_T is Earth's mass and $\mathbf{M}_c^b$ is the inertia matrix of the satellite about the center of mass. Note that the gravitational force does not give a force moment since its act on the center of the body frame.

The implementation of `SatelliteDynamics.m` is then:

```
function [ state_dot ] = SatelliteDynamics( t, x, parameters )
    position = x(1:3);
    R = reshape(x(4:12),3,3);
    velocity = x(13:15);
    omega = x(16:18);
    omega_cross = [    0,       -omega(3),  +omega(2);
                    +omega(3),      0,       -omega(1);
                    -omega(2), +omega(1),       0];
    M = parameters; % inertia matrix
    state_dot = [velocity;
                 reshape(R*omega_cross,9,1);
                 GravityAcceleration(position);
                 -M\omega_cross*M*omega];
end
function g = GravityAcceleration(position)
    G = 6.674e-11;       % universal gravitational constant
    M = 5.972e+24;    % Earth's mass
    g = -G*M*position/norm(position,2)^3;
end
```

In case (a), the inertia matrix is already given, and is $\mathbf{M}_c^b = \frac{1}{6}m\mathbf{I}$.

In case (b), we have to correct this expression. This can be done by adding the contribution of the added mass to the inertia matrix with respect to the center of the cube, and by using the parallel axis theorem to finally get the inertia matrix about the center of mass.

The inertia matrix with respect to the center of the cube is now with the added point mass

$$\mathbf{M}_o^b = \frac{1}{6}ml^2\mathbf{I} - m_0\left(\mathbf{r}_0^b\right)^\times\left(\mathbf{r}_0^b\right)^\times,$$

where m_0 is the added mass, and $\vec{r}_s$ is the vector from the center of mass to the center of the cube, i.e.

$$\mathbf{r}_0^b = \frac{l}{2}\begin{bmatrix}\pm 1\\ \pm 1\\ \pm 1\end{bmatrix} \quad \text{(the signs depend on the corner).}$$

In order to find the inertia matrix with respect to the center of mass, we have to first find the vector from the center of mass to the center of the cube, $\vec{r}_s$. We observe that the addition of a mass m_0 at a distance L in one particular direction, will shift the center of mass $\frac{m_0}{m+m_0}L$ units in that direction. Hence,

$$\mathbf{r}_s^b = -\frac{m_0}{m_0+m}\mathbf{r}_0^b,$$

and the parallel axis theorem gives

$$\mathbf{M}_c^b = \frac{1}{6}ml^2\mathbf{I} - m_0\left(\mathbf{r}_0^b\right)^\times\left(\mathbf{r}_0^b\right)^\times + (m+m_0)\left(\mathbf{r}_s^b\right)^\times\left(\mathbf{r}_s^b\right)^\times.$$

Therefore, an example implementation of the simulation routine is:

```
% define initial state
earth_radius = 6356e+3;
```

```
orbit_height = 36000e+3;
azi = pi/4; %azimuth
dec = pi/4; %declination
pos = (earth_radius+orbit_height)*[sin(dec)*cos(azi);sin(dec)*sin(azi);cos
    (dec)];
r = random('norm',0,1,[3,1]); % random axis of rotation / angle
R = expm([    0,      -r(3),  +r(2);
             +r(3),      0,   -r(1);
             -r(2), +r(1),     0]); % rotation matrix describing the
                satellite orientation
state = [pos;
         reshape(R,9,1);
         zeros(3,1);
         deg2rad(60);deg2rad(80);deg2rad(100)];
% parameters
withAddedMass = true;
l = 0.5;
rho = 1;
m = rho*l^3;
if withAddedMass
    m_added = 0.1;
else
    m_added = 0;
end
pos_added = l/2*[1;1;1];
% shift of center of mass
cm_shift = m_added/(m+m_added)*pos_added;
% inertia matrix of satellite from cube center
M = (m*l^2/6)*eye(3) + m_added*(norm(pos_added,2)^2*eye(3)-pos_added*
    pos_added');
% inertia matrix of satellite from mass center
M = M - (m+m_added)*(norm(cm_shift,2)^2*eye(3)-cm_shift*cm_shift');
parameters = M;
% simulate satellite dynamics
time_final = 120; % final time
[time,statetraj] = ode45(@(t,x)SatelliteDynamics(t, x, parameters),[0
    time_final],state);
% real-time animation
ScaleFrame = 5;   % scaling factor for adjusting the frame size (cosmetic)
FS         = 15;  % fontsize for text
SW         = 0.035; % arrows size
tic; % resets Matlab clock
time_display = 0; % time displayed
while time_display < time(end)
    time_animate = toc; %get the current clock time
    %interpolate the simulation at the current clock time
    state_animate = interp1(time,statetraj,time_animate)';
    pos_cm = state_animate(1:3)*1e-7;   % scaled position of the satellite
    R = reshape(state_animate(4:12),3,3); % orientation
```

```
    omega = R*state_animate(16:18);  % omega
    if withAddedMass
        pos_cube = pos_cm - R*cm_shift;
    else
        pos_cube = pos_cm;
    end
    figure(1);clf;hold on
    MakeFrame(zeros(3,1),eye(3),ScaleFrame,FS,SW,'a', 'color', 'k')
    MakeFrame(pos_cm,R,ScaleFrame,FS,SW,'b', 'color', 'r')
    MakeArrow(pos_cm,omega,FS,SW,'$$\omega$$', 'color', [0,0.5,0])
    DrawRectangle(pos_cube,R,'color',[0.5,0.5,0.5]);
    FormatPicture([0;0;2],0.5*[73.8380  21.0967  30.1493])
    if time_display == 0
        display('Hit a key to start animation')
        pause
        tic
    end
    time_display = toc; %get the current clock time
end
```

In case (a), we observe that the $\boldsymbol{\omega}_{ib}^b$ is constant, i.e. that the satellite rotates in the same direction all the time. This is because the inertia matrix is a multiple of the identity matrix. In more detail, we have that

$$\begin{aligned} \mathbf{M}_c^b \dot{\boldsymbol{\omega}}_{ib}^b &= -\left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_c^b \boldsymbol{\omega}_{ib}^b \\ \Rightarrow \frac{m}{6}\dot{\boldsymbol{\omega}}_{ib}^b &= -\left(\boldsymbol{\omega}_{ib}^b\right)^\times \frac{m}{6}\boldsymbol{\omega}_{ib}^b = 0 \\ \Rightarrow \dot{\boldsymbol{\omega}}_{ib}^b &= 0 \\ \Rightarrow \boldsymbol{\omega}_{ib}^b(t) &= \boldsymbol{\omega}_{ib}^b(t_0), \quad \text{for } t \geq t_0. \end{aligned}$$

In case (b), we observe that the angular velocity oscillates, and that the rotation of the satellite is no longer uniform. This is because the satellite is unbalanced around its center of mass.

Problem 2 (Spinner)

In this task, we will consider a "spinner", i.e. a disk of uniform density, radius $R = 1\,\text{m}$ and mass $m = 1\,\text{kg}$ mounted on a massless rod of length $L = 2\,\text{m}$ connected to a free joint.

We assume that the thickness of the disk is zero. Hence, the inertia matrix of the disk taken in a frame attached at its center with the radial symmetry axis as the third axis, is given by:

$$M_c^b = \frac{mR^2}{4}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{2}$$

Select a frame for the spinner and a representation of the SO(3) Lie group (orientation of the spinner). Then apply correctly the Newton-Euler equations to describe the motion of the spinner.

We will consider that the force of gravity is given by:

$$\vec{F} = -m\vec{g} \tag{3}$$

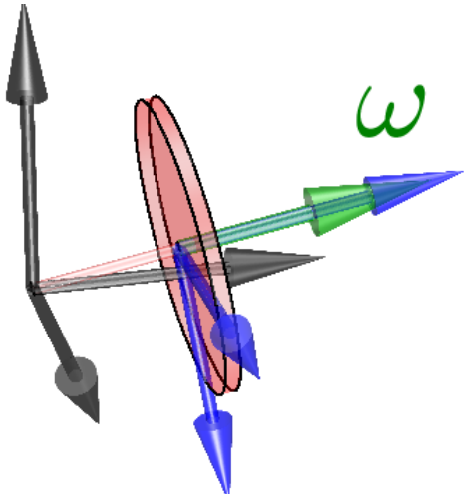

Figure 2: Schematic of the spinner.

What is your resulting state-space model?

Implement your model and simulate it for different angular velocities $\boldsymbol{\omega}$, where

$$\boldsymbol{\omega}_{ab}^{b} = \omega \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{4}$$

i.e. $\boldsymbol{\omega}$ is aligned with the radial symmetry axis of the disk. Test e.g. $\omega = \pi, 2\pi, 4\pi, 6\pi\,\mathrm{rad\,s^{-1}}$.

What do you observe? Are the results reasonable? Explain.

Hints:

- *The selection of a convenient body frame and* $\mathrm{SO}(3)$ *representation will make the problem much easier.*
- *You may need the parallel axes theorem.*
- *You will find code templates / examples on Blackboard to help you. See* `Gyropscope3DExample.m` *and previously delivered codes for tools to do 3D animations. These animations will allow you to assess your simulations.*

Solution: Since the point at the joint does not move, it is a good idea to take this point as the origin of the body frame. Then, we only have to model the orientation of the body. Moreover, we make sure that the 3rd axis of the body frame coincides with radial axial of symmetry of the disc. In that way, we can always find the position of the center of the disc by multiplying L with the 3rd column of the $\mathbf{R}_b^i$.

Therefore, we choose the states $\mathbf{R}_b^i$ and $\boldsymbol{\omega}_{ib}^b$. Hence, Newton-Euler's equations and the properties of the rotation matrices give the following differential equations:

$$\begin{aligned}
\dot{\mathbf{R}}_b^i &= \mathbf{R}_b^i \left(\boldsymbol{\omega}_{ib}^b\right)^\times \\
\mathbf{M}_0^b \dot{\boldsymbol{\omega}}_{ib}^b &= -\left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_0^b \boldsymbol{\omega}_{ib}^b + \mathbf{r}_c^b \times \mathbf{F}^b \\
&= -\left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_0^b \boldsymbol{\omega}_{ib}^b + L\mathbf{e}_3 \times \mathbf{R}_i^b \mathbf{F}^i \\
&= -\left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_0^b \boldsymbol{\omega}_{ib}^b - Lmg\mathbf{e}_3 \times \left(\mathbf{R}_b^i\right)^T \mathbf{e}_3 ,
\end{aligned}$$

where $\mathbf{M}_0^b$ is the inertia matrix with respect to the origin of the body frame, and $\mathbf{e}_3 = [0,0,1]^T$.

Note that the gravity force gives a force moment since the origin of the body frame is not at the center of mass.

This model can be implemented as done in previous tasks. For example, we can define a function that gives the derivative of the state:

```
function [ state_dot ] = GyroscopeDynamics( t, x, parameters )
    e3 = [0;0;1];
    g = 9.81;
    R = reshape(x(1:9),3,3);
    omega = x(10:12);
    omega_cross = [    0,       -omega(3),   +omega(2);
                     +omega(3),       0,       -omega(1);
                     -omega(2), +omega(1),       0];
    L = parameters(1);
    m = parameters(2);
    M = reshape(parameters(3:11),3,3);
    state_dot = [reshape(R*omega_cross,9,1);
                 -M\(omega_cross*(M*omega)-L*m*g*cross(e3,R'*e3))];
end
```

We use the parallel axis theorem to obtain the inertia matrix with respect to the origin of the body frame, $\mathbf{M}_0^b$:

$$\mathbf{M}_0^b = \mathbf{M}_c^b + m\begin{bmatrix} L^2 & 0 & 0 \\ 0 & L^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} = m\begin{bmatrix} \frac{R^2}{4} + L^2 & 0 & 0 \\ 0 & \frac{R^2}{4} + L^2 & 0 \\ 0 & 0 & \frac{R^2}{2} \end{bmatrix}.$$

In the simulations, we observe the phenomenon of precession. Moreover, we observe that the larger the magnitude of the angular velocity, the slower the change in position of the spinner. This as expected due to the larger inertia.