Out: January 31, 2023
Due: February 15, 2023

NTNU – Trondheim
Norwegian University of
Science and Technology

TTK4130 Modeling and Simulation

Assignment 4

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:

1. The satellite is a cube of uniform, unitary density, having an edge of 50cm.
2. 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.

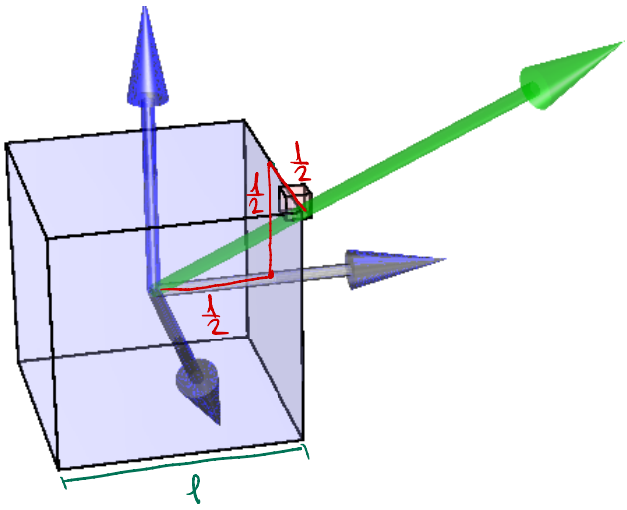


Figure 1: Schematic of the satellite.

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

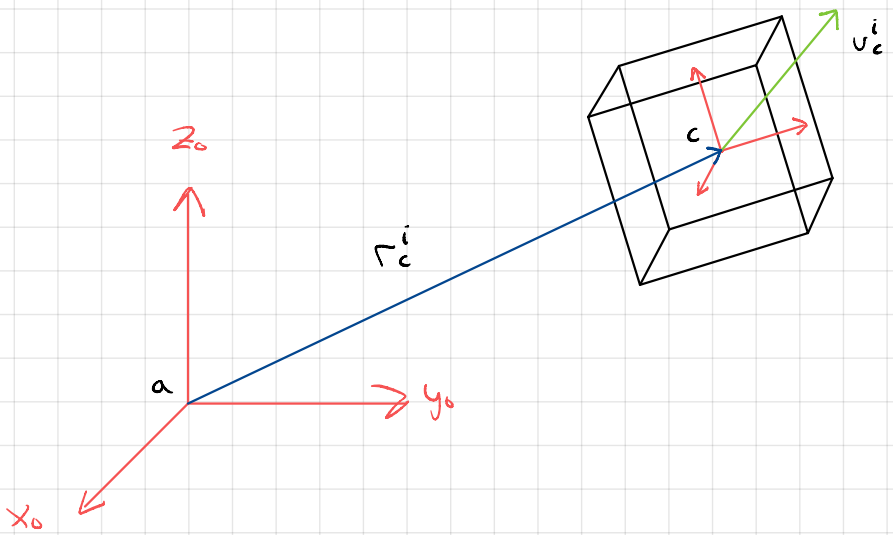
$$\vec{F} = -\frac{G\, m_{\mathrm{T}}\, m}{\|\vec{r}_{\mathrm{c}}\|^2} \cdot \frac{\vec{r}_{\mathrm{c}}}{\|\vec{r}_{\mathrm{c}}\|} \tag{1}$$

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

$$\frac{1}{6} m l^2 I \tag{2}$$

where m is the mass, l is the length of the sides and I is the 3-by-3 identity matrix.

Problem 1



Force of gravity:

$$\vec{F} = -\frac{G \cdot m_T \cdot m}{\|r_c\|^2} \cdot \frac{\vec{r}_c}{\|\vec{r}_c\|} = ma$$

Inertia matrix in the ref. frame attached to the cube:

$$M = \frac{1}{6} \cdot m \cdot l^2 \cdot I$$

m: mass
l: length of the cubes sides
I: 3x3 ident. matrix

a) Use N-E eq. to describe the satellites motion on a state-space model.

States:

- $\vec{r}_c^{\,i}$: Position of the satellites center of mass expressed in the inertial frame
- R_b^i : Rotation matrix from the inertial frame to the body frame.
- $\vec{v}_c^{\,i}$: Velocity of the satellites CoM in inertial frame.
- $\vec{\omega}_{ib}^{\,b}$: Coord. of the angular velocity in the body frame.

State space:

de p. 270 ch. 7.3.2

- $\dot{\vec{r}}_c^{\,i} = \vec{v}_c^{\,i}$
- $\dot{R}_c^i = R_c^i \left(\vec{\omega}_{ib}^{\,b}\right)^\times$

$$\dot{R}_b^a = R_b^a \left(\omega_{ab}^b\right)^\times$$

- $\vec{F} = m\vec{a}_c$, $\dot{\vec{v}}_c^{\,i} = a$

$$\dot{\vec{v}}_c^{\,i} = \frac{\vec{F}}{m} = -\frac{G \cdot m_T \cdot m}{\|r_c\|^2} \cdot \frac{\vec{r}_c}{\|\vec{r}_c\|} = -\frac{G \cdot m_T}{\|r_c\|^2} \cdot \frac{\vec{r}_c}{\|\vec{r}_c\|}$$

- $\vec{T}_c^{\,b} = \vec{M}_{b/i} \cdot \dot{\vec{\omega}}_{ib}^{\,b} + \vec{\omega}_{ib}^{\,b} \times \left(\vec{M}_{b/i} \cdot \vec{\omega}_{ib}^{\,b}\right)$

torque eq. zero since body frame in in center of mass (no arm)

$$0 = \vec{M}_{e/i} \cdot \dot{\vec{\omega}}^{e}_{ie} + \vec{\omega}^{e}_{ie} \times (\vec{M}_{ix} \cdot \vec{\omega}^{e}_{ie})$$

$$\dot{\vec{\omega}}^{e}_{ie} = -(M^{-1}_{e/i} \cdot \vec{\omega}^{e\times}_{ie} \cdot (\vec{M}_{e/i} \cdot \vec{\omega}^{e}_{ie}))$$

b) satelliteDynamics(t, X, parameters)

- parameters: Inertia matrix
- X (vector) :- position x_1 x_2 x_3
  - orientation dot x_4 x_7 x_{10} / x_5 x_8 x_{11} / x_6 x_9 x_{12}
  - velocity x_{13} x_{14} x_{15}
  - angular accel. x_{16} x_{17} x_{18}

6)

```
function [ state_dot ] = SatelliteDynamics( t, x, parameters )

    % Intertia matrix
    M = parameters;
    % Input vector x distributed
    position = x(1:3);
    % Creates a 3x3 matrix from vector element x4-x12
    R = reshape(x(4:12),3,3);
    velocity = x(13:15);
    omega = x(16:18);
    % Skew matrix of omega
    omega_skew = [0        -omega(3)  omega(2);
                  omega(3)  0        -omega(1);
                 -omega(2)  omega(1)  0];
    %Return
    state_dot = [velocity;
                 reshape(R*omega_skew,9,1);      % Calculate R_dot, and reshapes the 3x3 matrix, to a 9x1 vector
                 Gravity_acceleration(position);
                 -inv(M)*omega_skew*M*omega];
    % The code must return in the order you selected, e.g.:
    %    state_dot =  [velocity;
    %                  orientation_dot;
    %                  acceleration (ac);
    %                  angular acceleration (omega dot)];
end

function g = Gravity_acceleration(position)
    G = 6.676e-11;      % Gravitational constant
    M_t = 5,972e+24;      % Earth's mass
    g = -G*M_t*position/norm(position,2)^3;
end
```

```
Not enough input arguments.

Error in SatelliteDynamics (line 4)
    M = parameters;
```

Published with MATLAB® R2022a

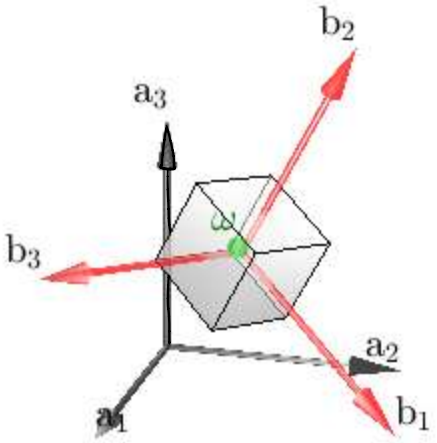
c). Simulation

```
clear all
%close all
clc

% Eart params
earth_radius = 6356e+3;
orbit_height = 36000e+3;
azi = pi/4; %azimuth
dec = pi/4; %declination

position = (earth_radius+orbit_height)*[sin(dec)*cos(azi);sin(dec)*sin(azi);cos(dec)];
omega = [deg2rad(60);deg2rad(80);deg2rad(100)];
velocity = [0;0;0];
% Rotation matrix describing the satellite orientation
r = random('norm',0,1,[3,1]); % Random axis of rotation / angle
orientation = expm([   0,      -r(3),  +r(2);
             +r(3),     0,   -r(1);
             -r(2), +r(1),    0]);

% Define your initial state, e.g. as:
 state = [position;
          reshape(orientation,9,1);
          velocity;
          omega];

% "parameters" allows you to pass some parameters to the "SatelliteDynamics" function
mass = 1;
length = 50e-3;
parameters = 1/6*mass*length^2;

time_final = 6; %Final time

% Simulate satellite dynamics
[time,statetraj] = ode45(@(t,x)SatelliteDynamics(t, x, parameters),[0,time_final],state);

% Here below is a template for a 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 = state_animate(16:18);          % omega



    figure(1);clf;hold on
    % Use the example from "Satellite3DExample.m" to display your satellite
    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,R*omega,FS,SW,'$$\omega$$', 'color', [0,0.5,0])
    DrawRectangle(pos_cm,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
```

```
M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5
```

```
M_t =

     5


M_t =

     5


M_t =

     5


M_t =

     5

Hit a key to start animation
```



The result looks reasonable. It behaves in a reasonable way.

Published with MATLAB® R2022a

The position and ang.velocity coord. are set to simulate the a satellites pos and ang.vel around earth. This seems reasonable. I do not, on the other hand, understand the velocity coord., as the simulation staid the same even though these coord. was changed. ?

For task (a)-(c) consider the case without the added mass (case 1 above)

(a) Considering a frame at the center of mass, apply the Newton-Euler equations to describe the satellite's motion (position and orientation). What is the resulting state-space model?

(b) Complete the function `SatelliteDynamics.m` in the delivered code, and add it to your answer.

(c) Simulate the satellite system using the Matlab ODE integration function `ode45` (seen in previous assignments).

What do you observe? Are the results reasonable?

(d) Now consider the added mass (case 2 above). The added mass will shift the center of mass of the system. Calculate the inertia matrix around this new center of mass and repeat the simulation in (c).

What do you observe? Are the results reasonable? What is the main difference between the two cases? Explain, and comment on both rotation and translation.

Hint: Use the parallel axis theorem

An example code is provided on Blackboard. There you will also find the means to make a 3D animation of your simulation.

Hints:

- *You will find code templates / examples on Blackboard to help you get started. Further hints are provided therein. 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, and describe the different motions.*
- *For parameters and initial values that are not given, you are free to choose reasonable numerical values. For example, Earth's radius is* 6356 km *and its orbital height is* 36 Mm*. The numerical solver will try to capture all dynamics at a certain precision. If the dynamics are "fast", the simulation can not run for long, since the step size will be small. However, we are interested in illustrating the difference of the two cases. What can be different? The thing we want to illustrate must therefore seem fast enough.*

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. See Figure 2.

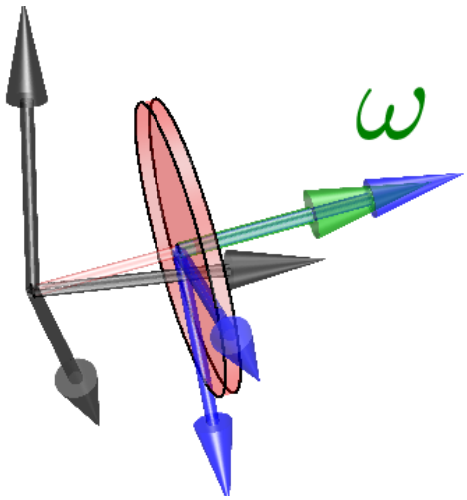


Figure 2: Schematic of the spinner.

d). Consider the added mass (change of CoM).
Calculate the inertia matrix M around the new center of mass.

Parallel axis theorem:

$$M^{b}_{b/o} = M^{b}_{b/c} - m\left(r^{b}_{g}\right)^{\times}\cdot\left(r^{b}_{g}\right)^{\times}$$

$$= M^{b}_{b/c} + m\left[\left(r^{b}_{g}\right)^{2}\cdot \mathbb{I} - r^{b}_{g}\left(r^{b}_{g}\right)^{T}\right]$$

We have to correct the already given inertia matrix, with the added mass.

Inertia matrix with respect to the center of the cube and the added mass, is now:

$$M^{b}_{o} = \underbrace{\frac{1}{6}ml^{2}\cdot\mathbb{I}}_{\text{already given In-matrix}} - \underbrace{m_o\left(r^{b}_{o}\right)^{\times}\cdot\left(r^{b}_{o}\right)^{\times}}_{\text{added mass In-matrix}}$$

m_o : added mass

$$r^{b}_{o} = \frac{1}{2}\begin{bmatrix}1\\1\\1\end{bmatrix}$$

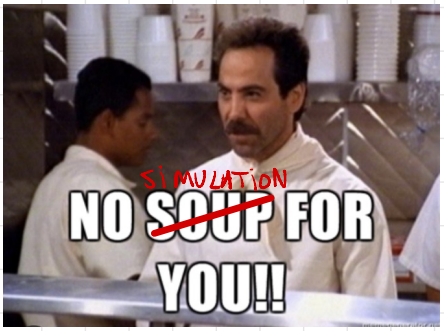
Vector from center of cube, to added mass point.

Also have to find the vector from the "new" center of mass to the center of the cube, $\vec{r}^{\,b}_{s}$.

$$\vec{r}^{\,b}_{s} = -\frac{m_o}{m_o + m}\cdot\vec{r}^{\,b}_{o}$$

Using parallel axis th. to find new inertia matrix.

$$M_c^G = \frac{1}{6} \cdot m \cdot l^2 \cdot \mathbb{I} - m_0 (r_0^G)^x \cdot (r_0^G)^x + (m + m_0) \cdot (r_0^G)^x \cdot (r_s^G)^x$$



Problem 2

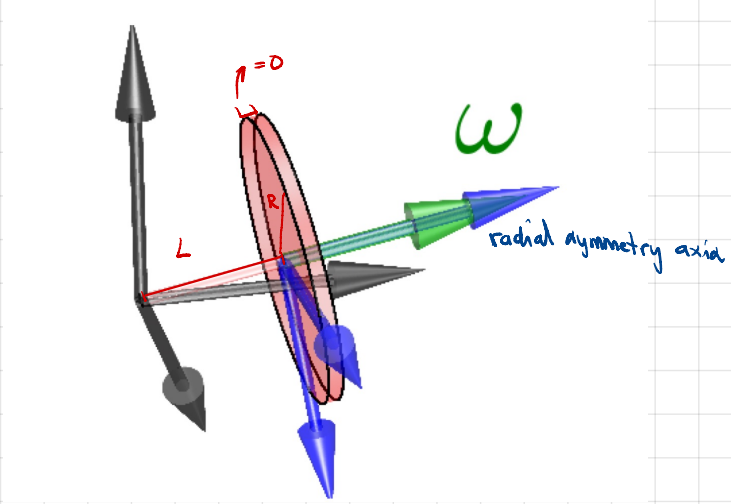
"Spinner" with uniform density:

$R = 1$

$m = 1$

$L = 2$

width = 0



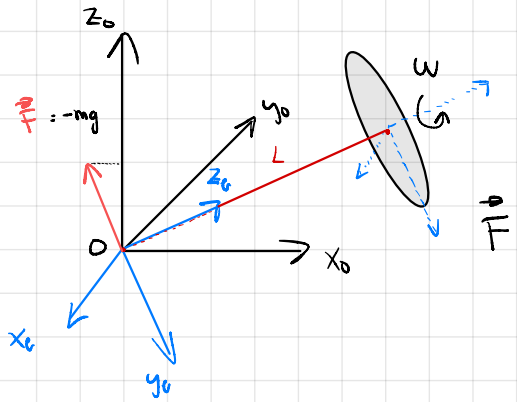
Inertia matrix: (about the disk center of mass)

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

Force of gravity:

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

Choosing same origo for body and inertial ref. frame.



Which leads to only needing these states:

- R_b^i : orientation from spinner frame to inert. ref frame.
- $\vec{\omega}_{ib}^{\,i}$: ang.velocity of the spinner

(world frame)

Must find inertia matrix about the inertial frame origo using:

Parallel axis theorem:

$$M^{b}_{b/o} = M^{b}_{b/c} - m\left(r^{b}_{g}\right)^{\times}\cdot\left(r^{b}_{g}\right)^{\times}$$

↑ point o which we in inertial frame origo

$$= M^{b}_{b/c} + m\left[\left(r^{b}_{g}\right)^{2}\cdot\mathbb{I} - r^{b}_{g}\left(r^{b}_{g}\right)^{T}\right]$$

where $r^{b}_{g_{\nwarrow o?}} = L\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}$

in the vector from the spinner center of mass, to the inertial frame origo.

$$M^{b}_{b/o} = \frac{mR^2}{4}\cdot\begin{bmatrix}1&0&0\\0&1&0\\0&0&2\end{bmatrix} + 1\left(\begin{bmatrix}0\\0\\L\end{bmatrix}^{2}\cdot\mathbb{I} - \begin{bmatrix}0\\0\\L\end{bmatrix}\cdot\begin{bmatrix}0&0&L\end{bmatrix}\right)$$

M^{b}_{c} $\quad r^{b}_{g}$ $\quad r^{b}_{g}$ $\quad r^{b\,T}_{g}$

Can now calculate the spinner dynamics.

State-space:

- $\dot{R}^{i}_{b} = R^{i}_{b}\cdot\left(\vec{\omega}^{i}_{ib}\right)^{\times}$

- $\vec{T}_{bc} = \vec{M}^{b}_{b/o}\cdot\dot{\vec{\omega}}_{ib} + \vec{\omega}_{ib}^{\times}\left(\vec{M}_{b/c}\cdot\vec{\omega}_{ib}\right)$ eq. 7,42 Newton Euler for rigid body

where $\vec{T}_{bc} = \vec{r}^{\,b}_{g}\times\vec{F}$ (torque)

$$\rightarrow \quad \vec{M}^{\mathcal{C}}_{\mathcal{C}/O} \cdot \dot{\omega}_{i\mathcal{C}} = \vec{r}^{\mathcal{C}}_{g} \times \vec{F}^{\mathcal{C}} - \vec{\omega}_{i\mathcal{C}}{}^{\times} \left(\vec{M}_{\mathcal{C}/C} \cdot \vec{\omega}_{i\mathcal{C}} \right)$$

$$\dot{\omega}_{i\mathcal{C}} = \vec{M}^{\mathcal{C}}{}^{-1}_{\mathcal{C}/O} \left(\vec{r}^{\mathcal{C}}_{g} \times \vec{F}^{\mathcal{C}} - \vec{\omega}_{i\mathcal{C}}{}^{\times} \left(\vec{M}_{\mathcal{C}/C} \cdot \vec{\omega}_{i\mathcal{C}} \right) \right) \quad ?$$

θ

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

$$M_c^b = \frac{mR^2}{4} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{3}$$

We will consider that the force of gravity is given by

$$\vec{F} = -m\vec{g} \tag{4}$$

- **Write down the equations:** Select a frame for the spinner and a representation of the $\mathrm{SO}(3)$ Lie group (orientation of the spinner). Then apply correctly the Newton-Euler equations to describe the motion of the spinner.

  What is your resulting state-space model?
- **Implement** your model and simulate it for different angular velocities $\boldsymbol{\omega}$, where

$$\boldsymbol{\omega}_{ab}^b = \omega \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{5}$$

  i.e. $\boldsymbol{\omega}$ is aligned with the radial symmetry axis of the disk. Test e.g. $\omega = \pi, 2\pi, 4\pi, 6\pi \,\mathrm{rad\,s^{-1}}$. When implementing it may be useful to reuse some code that you have already made, and create a function returning the derivative of the states, as we have done before.
- 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.*