INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

PROJECT CT11 REPORT

Design, Sizing and Control of a homemade Segway

Authors:
Baptiste BONNAUD
Jean-Baptiste FLEURY
Adrien MORAINE

INSA Toulouse
5GM-IS

Tutor:
Marc BUDINGER

Université de TOULOUSE
Institut Clément Ader
INSA, Département de Génie Mécanique

xx January 2015

Contents

List of Figures

List of Tables

Introduction

Our project is about designing and making a Segway Personal Transporter (usually named ”Segway”). This Segway is a personal two-wheeled electric vehicle, powered by batteries. The specificity of this system is that the driver controls the Segway by leaning forward to accelerate and leaning backward to slow down. You can see in the picture below a commercialised version of the product.

Figure 1: A person on a Segway PT

In order to create our own Segway, we will use some spare parts from an electric scooter. However this is not the most difficult part of the job to be done here. The Segway is very similar to a standard system analysis and engineering problem known as inverted pendulum. So what we have to do is:

- Analyse the mechanical behaviour of a Segway to deduce a mathematical model
- Design a 3D model of all parts and assemble it to compute the numeric data of the problem
- Choose and adapt a control-command law in order to make the Segway stable
- Code and implement the law on a microcontroller
- Assemble the Segway
- Check, test, improve and add secondary features (e.g.: security, turning...)

In order to use the same vocabulary, we present you a little glossary below (on Fig. 2)

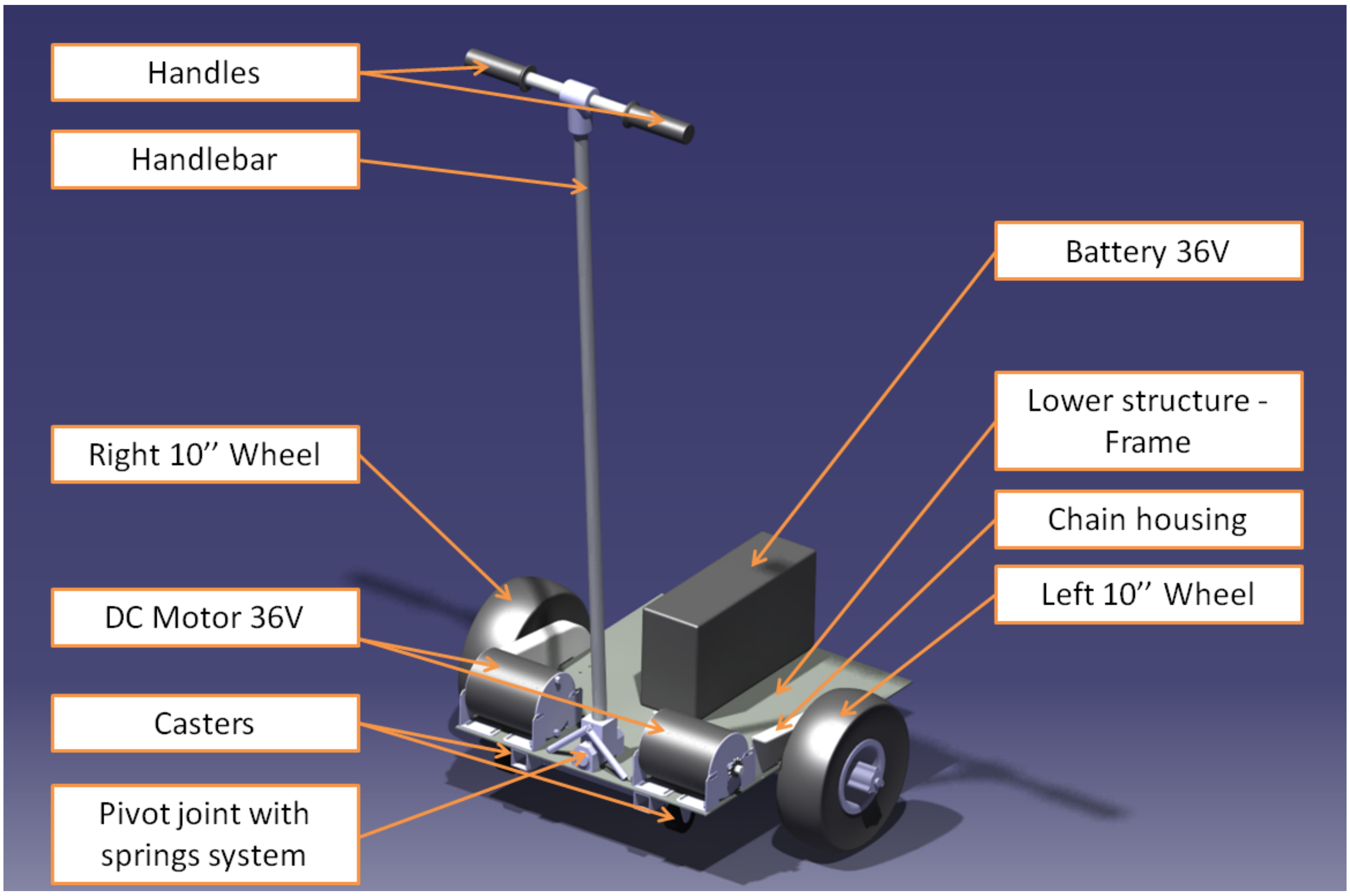


Figure 2: A glossary of a Segway

1 System modelling

1.1 Mathematical modelling

1.1.1 Scheme, variables and hypothesis

We model our Segway as the following system:

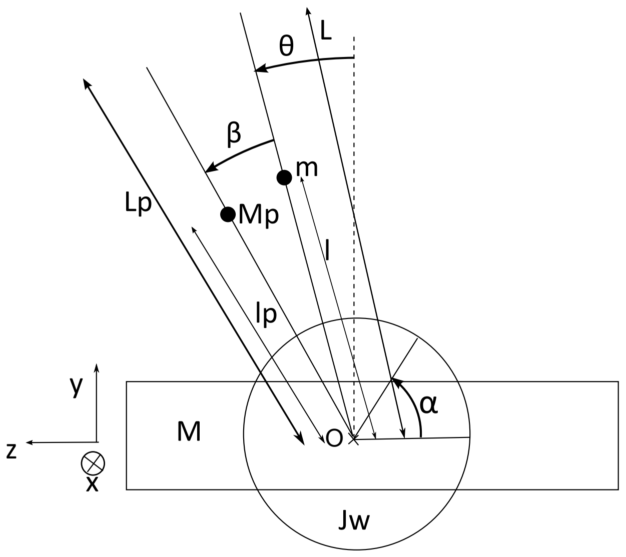


Figure 3: Sketch of the Segway

We define the following parameters:

Variable	Description	Unit
C_{rr}	Rolling resistance coefficient	$[1]$
C_x	Drag coefficient of the person	$[1]$
g	Gravity of the earth	$[m/s^2]$
J_w	Inertia of a wheel and its pinion	$[Kg.m^2]$
J_m	Inertia of a motor with its transmission shaft and pinion	$[Kg.m^2]$
J_h	Inertia of the handlebar	$[Kg.m^2]$
J_p	Inertia of the person	$[Kg.m^2]$
K_m	Torque constant of the DC motor	$[Nm/A]$
K_g	Gearing ratio of the reducer (>1 for a reducer)	$[1]$
L	Length of the handlebar	$[m]$
L_p	Height of the person using the Segway	$[m]$
l	Distance between the pivot O and the center of gravity of the handlebar	$[m]$
l_p	Distance between the pivot O and the center of gravity of the person	$[m]$
L_r	Inductance of the motor's winding	$[H]$
M	Total mass of the lower part of the Segway	$[Kg]$
m	Mass of the handlebar	$[Kg]$
M_p	Mass of the person	$[Kg]$
R	Electrical resistance of the motor's winding	$[\Omega]$
R_w	Radius of the wheel	$[m]$
S	Area of the cross-section of the person in the (Oxy) plane	$[m^2]$
V_g	Speed at the center of gravity of the handlebar	$[m/s]$
V_p	Speed at the center of gravity of the person	$[m/s]$
ρ	Density of the air	$[Kg/m^2]$

Table 1: Parameters of our system

1.1.2 Lagrangian

We know that the Lagrangian is defined as:

$$\mathcal{L} = E_c - E_p \tag{1}$$

Where:

- E_c is the total kinetic energy of the system under study
- E_p is the total potential energy of the system under study

We make the following assumptions:

- The frictional effects are neglected
- The dynamic of β is not considered (we assume that the person moves slowly)

We can write the kinetic energy of our system:

$$\begin{aligned}
E_c &= \frac{1}{2}M\dot{z}^2 + \frac{1}{2}mV_g^2 + \frac{1}{2}J_h\dot{\theta}^2 + 2\frac{1}{2}\left(J_w + K_g^2 J_m\right)\dot{\alpha}^2 + \frac{1}{2}M_pV_p^2 + \frac{1}{2}J_p\dot{\theta}^2 && (2)\\
&= \frac{1}{2}M\dot{z}^2 + \frac{1}{2}mV_g^2 + \frac{1}{2}J_h\dot{\theta}^2 + \frac{\left(J_w + K_g^2 J_m\right)}{R_w^2}\dot{z}^2 + \frac{1}{2}M_pV_p^2 + \frac{1}{2}J_p\dot{\theta}^2 && (3)\\
&= \frac{1}{2}M\dot{z}^2 + \frac{1}{2}mV_g^2 + \frac{1}{2}J_h\dot{\theta}^2 + \frac{1}{2}M_{eq,w}\dot{z}^2 + \frac{1}{2}M_pV_p^2 + \frac{1}{2}J_p\dot{\theta}^2 && (4)
\end{aligned}$$

We can then write the potential energy of our system:

$$E_p = mgl\cos(\theta) + M_pgl_p\cos(\theta+\beta) \tag{5}$$

The langrangian can thus be written as:

$$\begin{aligned}
\mathcal{L} &= E_c - E_p && (6)\\
&= \frac{1}{2}M\dot{z}^2 + \frac{1}{2}mV_g^2 + \frac{1}{2}J_h\dot{\theta}^2 + \frac{1}{2}M_{eq,w}\dot{z}^2 + \frac{1}{2}M_pV_p^2 + \frac{1}{2}J_p\dot{\theta}^2 - mgl\cos(\theta) - M_pgl_p\cos(\theta+\beta) && (7)
\end{aligned}$$

Let O_g be the center of gravity of the handlebar and O_p the center of gravity of the person. We have:

$$\vec{OO_g} = \begin{bmatrix} 0 \\ l\cos(\theta) \\ z + l\sin(\theta) \end{bmatrix}, \vec{OO_p} = \begin{bmatrix} 0 \\ l_p\cos(\theta+\beta) \\ z + l_p\sin(\theta+\beta) \end{bmatrix}$$

By deriving $\vec{OO_g}$ and $\vec{OO_p}$, we can write:

$$\begin{aligned}
V_g^2 &= \left(\dot{z} + l\dot{\theta}\cos(\theta)\right)^2 + \left(-l\dot{\theta}\sin(\theta)\right)^2 && (8)\\
&= \dot{z}^2 + 2l\dot{\theta}\dot{z}\cos(\theta) + l^2\dot{\theta}^2 && (9)\\
V_p^2 &= \left(\dot{z} + l_p\dot{\theta}\cos(\theta+\beta)\right)^2 + \left(-l_p\dot{\theta}\sin(\theta+\beta)\right)^2 && (10)\\
&= \dot{z}^2 + 2l_p\dot{\theta}\dot{z}\cos(\theta+\beta) + l_p^2\dot{\theta}^2 && (11)
\end{aligned}$$

In the end, the langrangian of our system is the following one:

$$\begin{aligned}
\mathcal{L} \quad &= \tfrac{1}{2}M\dot{z}^2 + \tfrac{1}{2}m\left(\dot{z}^2 + 2l\dot{\theta}\dot{z}\cos(\theta) + l^2\dot{\theta}^2\right) + \tfrac{1}{2}J_h\dot{\theta}^2 + \tfrac{1}{2}M_{eq,w}\dot{z}^2 \\
&+ \tfrac{1}{2}M_p\left(\dot{z}^2 + 2l_p\dot{\theta}\dot{z}\cos(\theta+\beta) + l_p^2\dot{\theta}^2\right) + \tfrac{1}{2}J_p\dot{\theta}^2 - mgl\cos(\theta) - M_pl_pg\cos(\theta+\beta)
\end{aligned} \tag{12}$$

1.1.3 Equations of Lagrange

Degree of freedom z:

The equations of Lagrange give us:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{z}}\right) - \frac{\partial\mathcal{L}}{\partial z} = F_{ext,z} \tag{13}$$

We have:

$$\frac{\partial \mathcal{L}}{\partial \dot{z}} = M\dot{z} + m\dot{z} + M_p\dot{z} + M_{eq,w}\dot{z} + ml\dot{\theta}\cos(\theta) + M_pl_p\dot{\theta}\cos(\theta+\beta) \tag{14}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{z}}\right) = M\ddot{z} + m\ddot{z} + M_p\ddot{z} + M_{eq,w}\ddot{z} + ml\ddot{\theta}\cos(\theta) + M_pl_p\ddot{\theta}\cos(\theta+\beta) - ml\dot{\theta}^2\sin(\theta) - M_pl_p\dot{\theta}^2\sin(\theta+\beta) \tag{15}$$

$$\frac{\partial \mathcal{L}}{\partial z} = 0 \tag{16}$$

$$F_{ext,z} = F_m - F_{rr} - F_{aero} - F_{slope} \tag{17}$$

We can write:

$$F_m = \frac{2C_m}{R_w} = \frac{2K_mK_gI}{R_w} \tag{18}$$

$$F_{rr} = (M_p + M + m)\,gC_{rr} \tag{19}$$

$$F_{aero} = \frac{1}{2}\rho C_x S V_p^2 \tag{20}$$

$$F_{slope} = (m + M + M_p)\,g\sin(\gamma) \tag{21}$$

Where γ is the slope's angle.
In the end, we find the following equation:

$$(M + m + M_p + M_{eq,w})\,\ddot{z} + (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))\,\ddot{\theta} - (ml\sin(\theta) + M_pl_p\sin(\theta+\beta))\,\dot{\theta}^2 = F_{ext,z} \tag{22}$$

$$(M_{eq})\,\ddot{z} + (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))\,\ddot{\theta} - (ml\sin(\theta) + M_pl_p\sin(\theta+\beta))\,\dot{\theta}^2 = F_{ext,z} \tag{23}$$

Degree of freedom θ:

The equations of Lagrange give us:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = F_{ext,\theta} \tag{24}$$

We have:

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}} = ml\dot{z}\cos(\theta) + ml^2\dot{\theta} + J_h\dot{\theta} + M_pl_p\dot{z}\cos(\theta+\beta) + M_pl_p^2\dot{\theta} + J_p\dot{\theta} \tag{25}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) = ml\ddot{z}\cos(\theta) - ml\dot{z}\dot{\theta}\sin(\theta) + ml^2\ddot{\theta} + M_pl_p\ddot{z}\cos(\theta+\beta) - M_pl_p\dot{z}\dot{\theta}\sin(\theta+\beta) + M_pl_p^2\ddot{\theta} + J_h\ddot{\theta} + J_p\ddot{\theta} \tag{26}$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = -ml\dot{\theta}\dot{z}\sin(\theta) - M_pl_p\dot{\theta}\dot{z}\sin(\theta+\beta) + mgl\sin(\theta) + M_pl_pg\sin(\theta+\beta) \tag{27}$$

$$F_{ext,\theta} = 0 \tag{28}$$

In the end, we find the following equation:

$$\left(J_h + J_p + M_pl_p^2 + ml^2\right)\ddot{\theta} + (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))\,\ddot{z} = mgl\sin(\theta) + M_pl_pg\sin(\theta+\beta) \tag{29}$$

$$J_{eq}\ddot{\theta} + (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))\,\ddot{z} = mgl\sin(\theta) + M_pl_pg\sin(\theta+\beta) \tag{30}$$

1.1.4 Final set of equations

In the end, we have the following set of equations:

$$\begin{cases} U = RI + L_r\dot{I} + \frac{K_mK_g\dot{z}}{R_w} \\ (M_{eq})\,\ddot{z} + (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))\,\ddot{\theta} - (ml\sin(\theta) + M_pl_p\sin(\theta+\beta))\,\dot{\theta}^2 = F_{ext,z} \\ J_{eq}\ddot{\theta} + (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))\,\ddot{z} = mgl\sin(\theta) + M_pl_pg\sin(\theta+\beta) \end{cases} \tag{31}$$

Let $T_{\cos} = (ml\cos(\theta) + M_pl_p\cos(\theta+\beta))$ and $T_{\sin} = (ml\sin(\theta) + M_pl_p\sin(\theta+\beta))$.
Rearranging the above set of equations, one can find:

$$\dot{I} = \frac{U}{L_r} - \frac{R}{L_r}I - \frac{K_mK_g}{R_wL_r}\dot{z} \tag{32}$$

$$\ddot{\theta} = \tfrac{1}{T_{\cos}^2 - J_{eq}M_{eq}}\left[F_{ext,z}T_{\cos} - T_{\sin}M_{eq}g + T_{\cos}T_{\sin}\dot{\theta}^2\right] \tag{33}$$

$$\ddot{z} = \tfrac{1}{T_{\cos}^2 - M_{eq}J_{eq}}\left[T_{\sin}T_{\cos}g - F_{ext,z}J_{eq} - T_{\sin}J_{eq}\dot{\theta}^2\right] \tag{34}$$

1.2 Identification of the physical parameters

1.2.1 Parameters of the mechanical parts

All physical parameters of the mechanical parts (gears, wheels...) have been identified using either a 3D CAD model of the existing part, or using the design used for the Segway. (See Section 4.2.5).
We assumed a rolling resistance coefficient of the tires of 1%.

1.2.2 Identification of the motor's parameters

We have four different parameters to identify for our motor. The first step was to identify the motor's torque constant. This is quite straightforward, assuming that the motor's torque constant and its speed constant are identical. We can thus deduce the motor's torque constant from the maximum rotational speed, which is 2500rpm, and the maximum voltage, which is 36V. The resistance of the winding was identified using a multimeter.

To identify the motor inertia, we decided to measure the step-response of the motor and deduce from this response the inertia. To do so, we modelled the motor as a first-order. By doing this, we decide to neglect the influence of the inductance of the winding on the dynamic of our system. This is an acceptable assumption as its time constant is usually much smaller than the mechanical time constant. We thus get:

$$\begin{aligned} U &= RI + K_m\omega \\ J\dot{\omega} &= K_m I \end{aligned} \tag{35}$$

$$\Rightarrow \begin{aligned} U &= RI + K_m\omega \\ I &= \tfrac{J\dot{\omega}}{K_{em}} \end{aligned} \tag{36}$$

$$\Rightarrow U = R\frac{J\dot{\omega}}{K_m} + K_m.\omega \tag{37}$$

Going through Laplace transform:

$$\Rightarrow U(s) = \frac{RJ\omega(s)}{K_m} + K_m\omega(s) \tag{38}$$

$$\Rightarrow U(s) = \omega(s)(\frac{RJ}{K_m} + K_m) \tag{39}$$

$$\Rightarrow \frac{\omega}{U}(s) = \frac{1}{\frac{RJ}{K_m}s + K_m} \tag{40}$$

We obtain the following transfer function :

$$\Rightarrow \frac{\omega}{U}(s) = \frac{1/K_m}{\frac{RJ}{K_m^2}s + 1} = \frac{K}{\tau s + 1} \tag{41}$$

By identification:

$$\Rightarrow \tau = \frac{RJ}{K_m^2} \tag{42}$$

$$\Rightarrow J = \frac{K_m^2\tau}{R} \tag{43}$$

This last result enables us to find the motor inertia. We just need to compute (τ) from the measurement shown in Fig.4:

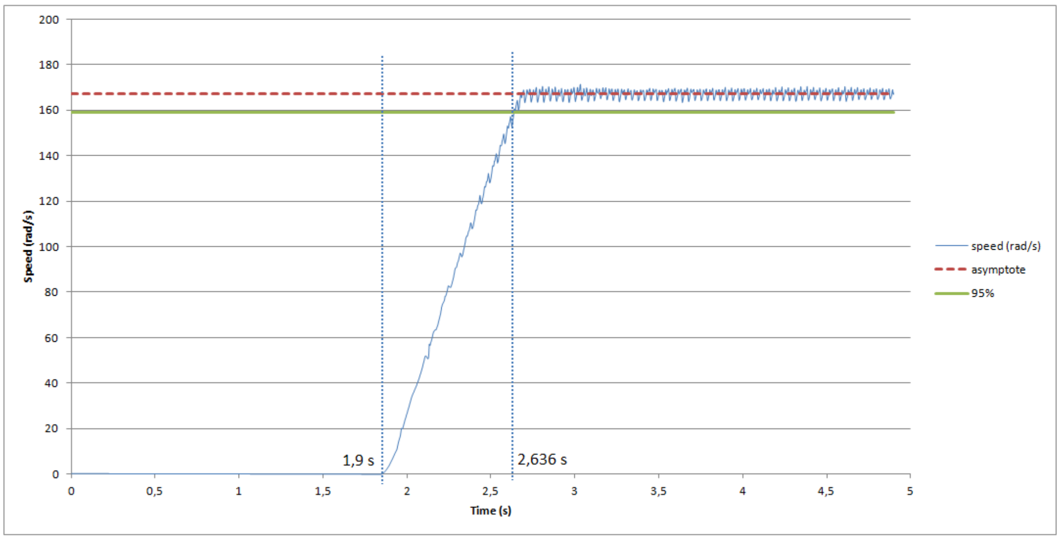


Figure 4: Motor speed response to a 20V step

As, in the case of a first-order system, the 5% response-time correspond to $(3.\tau)$. We can have now the value (τ) with the graph above. All the data are summed up here:

$$\begin{aligned} R &= 1\Omega \\ K_m &= \tfrac{36}{261{,}8} = 0,1375\tfrac{\mathrm{V}}{\mathrm{rad/s}} \\ \tau &= \tfrac{2{,}636-1{,}9}{3} = 0,24533\mathrm{s} \end{aligned}$$

So the numeric application for the motor inertia is:

$$J = \frac{K_{em}^2.\tau}{R} = \frac{0,1375^2 \times 0,24533}{1} \quad (44)$$

$$= 4,64 \times 10^{-4}\mathrm{Kg.m}^2 \quad (45)$$

1.2.3 Identification of the parameters of the driver

We assumed that we had a 1.8m tall driver, weighting 80 kg. We assumed that the center of gravity of the person was at 55% of its height (which in our case, gives us 1m). We also assumed that the inertia of a person was roughly 18 kg.m^2.
For the aerodynamic performances of a person, we assumed that the driver was equivalent to a trapezoid of area 0.8m^2. Thus we assumed that the driver behaved as a plate, therefore we took a drag coefficient of 1.1, which is very conservative.

1.2.4 Summary

To sum up the previous results, we took the following parameters during the control design, and simulation of our Segway:

Variable	Description	Unit	Value
C_{rr}	Rolling resistance coefficient	[1]	0.01
C_x	Drag coefficient of the person	[1]	1.1
g	Gravity of the earth	$[m/s^2]$	9.81
K_m	Torque constant of the DC motor	$[Nm/A]$	0.1375
K_g	Gearing ratio of the reducer (>1 for a reducer)	[1]	5.82
J_w	Inertia of a wheel and its pinion	$[Kg.m^2]$	1.86×10^{-2}
J_m	Inertia of a motor with its transmission shaft and pinion	$[Kg.m^2]$	4.64×10^{-3}
J_h	Inertia of the handlebar	$[Kg.m^2]$	1.1×10^{-1}
J_p	Inertia of the person	$[Kg.m^2]$	18
L	Length of the handlebar	$[m]$	1.2
L_p	Height of the person using the Segway	$[m]$	1.8
l	Distance between the pivot O and the center of gravity of the handlebar	$[m]$	0.7
l_p	Distance between the pivot O and the center of gravity of the person	$[m]$	1
M	Total mass of the lower part of the Segway	$[Kg]$	35
m	Mass of the handlebar	$[Kg]$	2
M_p	Mass of the person	$[Kg]$	80
R	Electrical resistance of the winding of the motor	$[\Omega]$	1
R_w	Radius of the wheel	$[m]$	0.127
S	Area of the cross-section of the person in the (Oxy) plane	$[m^2]$	0.8
ρ	Density of the air	$[Kg/m^2]$	1.2

Table 2: Numerical parameters of our system

1.3 Implementation on Modelica

We have decided to model our system on Modelica in order to test our mathematical modelling and our control design. It is convenient to use that software because our project will lead to a pedagogical tool and the students have some skills in this software.
As it will be a pedagogical tool, it was decided to add a visualizer on the implementation in order to show work of the student in 3D. This implementation is divided into 4 models/blocks. A first block which uses the final set of equations found in the previous part in order to generate a "Segway system" block. A second model which integrates visual components to obtain a 3D model of our simulation. A third block which models the motor. And a fourth one permits us to test the three previous models and simulate our entire system.

1.3.1 "Segway System" Model

The model is the direct implementation of the set of equations 31 found in part 1.1. We can use this set of non-linearized equations because Modelica is able to compute this system easily. This model includes two inputs and four outputs. For the inputs, we have the torque applied on the Segway and the beta angle which represents the inclination between the handlebar and the person. For the three outputs, we have the translational position and velocity of the Segway and also the angle theta and its rotational velocity. The following picture shows the final block with its inputs.

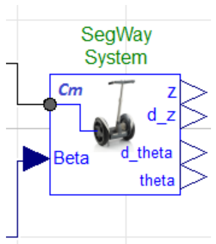


Figure 5: Interfaces of the "Segway System" model

In order to implement the mathematical system studied previously we have had to compute some parameters of our Segway. For example some masses, lengths or inertias of components or sub-assemblies. Those variables have been found by modeling them in CAD software. It was an approximation because the final mechanical design was not finished but it was sufficient to approximate those values correctly. Moreover, we need other values as the gear ratio of

our motors system and in order to compute the external forces applied in our system such as the rolling resistance, the drag coefficient or the frontal area of our system (person+system). More information on these numerical values can be found in section 1.2.
The following table sums up the parameters and variables used in the Modelica implementation of the Segway:

Type Unit	Item	Value	Description
Length	l_handlebar	0.7	Distance between the origin O and the center of gravity of the handlebar
Length	l_person	1	Distance between the origin O and the center of the handlebar
Radius	r_wheel	0.127	Wheel's radius
Mass	m_lower_part	35	Mass of the lower part of the Segway
Mass	m_handlebar	1.5	Mass of the hanledbar
Mass	m_person	75	Mass of the person
Inertia	j_wheel	0.0267	Inertia of the wheel and its pinion/crowns
Inertia	j_gear_motor	0.001	Inertia of the motor and its transmission
Inertia	j_handlebar	0.05	Inertia of the handlebar
Inertia	j_person	20	Inertia of the person
Real	k_g	5.8	Gearing ratio of the reducer
Real	Crr	0.01	Rolling Resistance coefficient
Real	Cx	1.2	aerodynamic coefficient
Real	S	0.8	human surfaces

Table 3: Variables used on the Modelica implementation of the Segway

1.3.2 "Segway Visualizer" Model

The visualizer shows the translational movement of the system on the ground and also the inclination of the handlebar with the ground (the α angle) and the inclination of the person with the handle bar floor (the β angle). Thus we need three inputs for this model, the position z of the system and also the α and β angles. Fig.6 below shows the final block with its inputs.

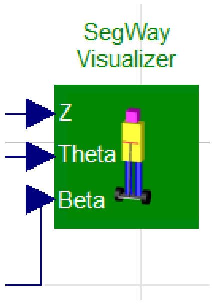


Figure 6: Interfaces of the "Segway Visualizer" model

This model is composed by cylinders and rectangle bodies in order to implement and simplify the 3D model of our system with its user. Fig.7 below shows the structure of this block.

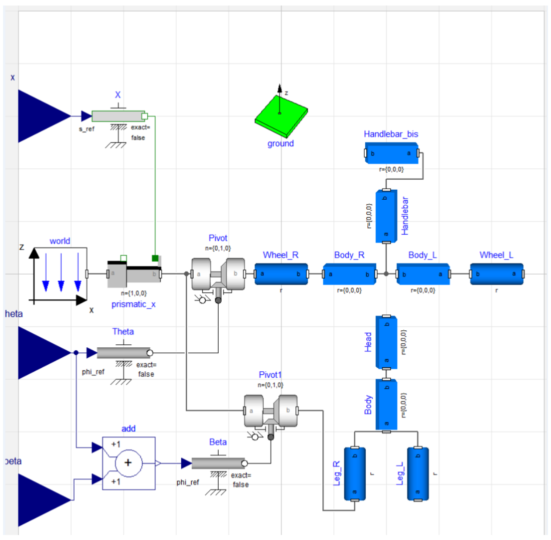


Figure 7: Structure of the "Segway Visualizer" model

1.3.3 "Segway Motor" Model

This motor model takes into account the two motors in one model. This model is very simple because we decide to neglect the coil factor. Thus this model is composed by a resistor and an 'emf' (electromotive force) component. The coefficient K_m of the motor has been obtained with the following equation: $U = k.\Omega$. As we know the voltage (36V) and the nominal velocity (2500tr/min-1), we can find easily: $K_m = 0.1375$.
We can see this model in Fig.8 below:

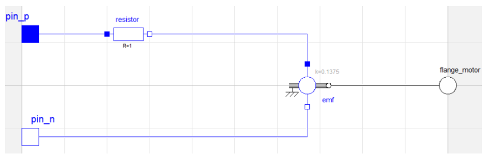


Figure 8: "Segway Motor" Model

1.3.4 "Segway" Model

The last main model is the "Segway" model which contains all previous models in order to simulate our entire system. Moreover some other components are implemented such as the PWM one which control the voltage brought to the motors (TF component in the picture below) and also the basic electronical components as the battery and the ground. One reducer was also added to take into account the gear ratio between the motor and the wheel.

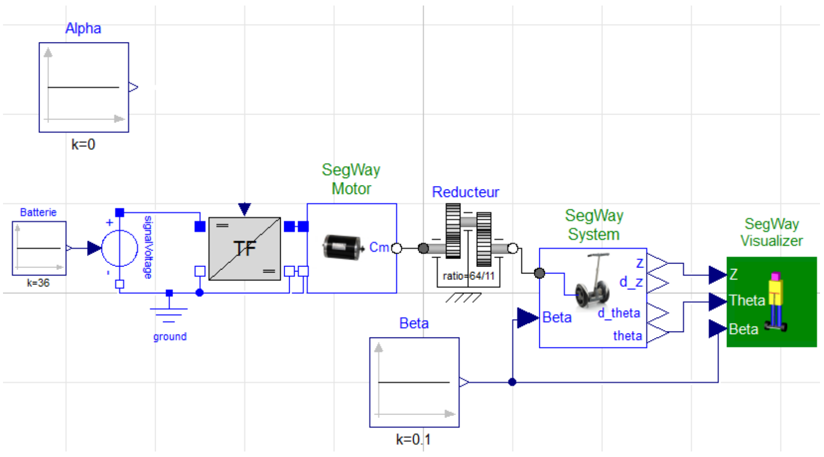


Figure 9: Modelica implementation of the entire system

1.4 Implementation on Simulink

1.4.1 Rationale

On top of the Modelica implementation, we also used a MatLab/Simulink model. This new implementation permits us to simulate the system with new features.

The Simulink model helped us to design the control law with a specific toolbox. This toolbox tunes a PID controller depending of the behaviour that we choose before. The other functionality that we strongly needed is the time discretisation. Sensors and controllers work on discrete time domain, and this adds delays and noises on the system.

After the Modelica model, that gives a first idea of the system's behavior and that is simpler for sizing, the Simulink model is necessary for control design. We describe and explain in this part how we did this implementation.

1.4.2 Schematic description

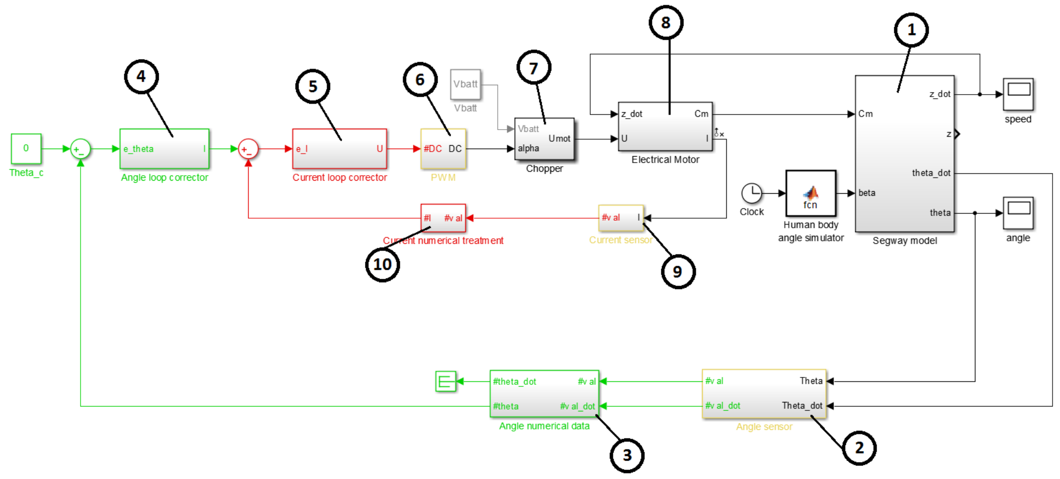


Figure 10: Closed-loop simulink implementation schematic

We can see on Fig. 10 the implementation of the control model on Matlab/Simulink. This model takes into account discretisation effects due to the use of a microcontroller and sensors: the discrete domain is colored in red (fast sampling) or green (slow sampling), the mixed domain (temporal conversion) is in yellow, and continuous domain is represented in black. We describe each block corresponding to the legend in the following parts.

Segway model block

This block contains all the mechanical equations of the Segway. From the motor's torque (C_m) and the position of the driver (β) it computes the speed ($\dot{z}$) and the angle of the Segway (θ).

Angle sensor block

This block aims to simulate the gyroscope and the accelerometer. So it takes the value of theta and its derived term ($\dot{\theta}$) to give the corresponding numerical values.

This block models sensor imperfections and characteristics: time descretisation, values descretisation, values scaling, noises, time response.

Angle numerical treatment block

It contains the software code of the numerical treatment. This numerical treatment is performing filtering and drift correction.

Angle loop corrector block

This block computes the current to inject in the motor depending of the error on the Segway's angle.

Current loop corrector block

This block computes the duty cycle of the chopper in order to have the current specified by the previous block.

Pulse Width Modulation block

This block converts the discretised numerical value of the duty cycle into a continuous time-domain value for the Chopper.

Chopper block

This block doses the batteries tension to apply it at motor's input in function of the duty cycle.

Electrical motor block

This block simulates the electrical motor. With the actual speed of the Segway and the tension applied, it deduces the torque applied to the Segway and the current at motor's input.

Current sensor block

This block aims to simulate the current sensor. So it takes the value of I to give the corresponding numerised and discretised values.

This block models sensor characteristics: time descretisation, values descretisation, values scaling, possible time response.

Current numerical treatment block

This block contains the numerical treatment to rescale the value of current sensor in an understable value.

1.4.3 Time-domain conversion and discretisation

In order to simulate the behaviour of the system as realistic as possible, we used the MatLab/Simulink discretisation capability. This step is very important to validate the control-law because the time discretisation adds delay in control loops and value discretisation of sensors adds noises. These two phenomenons can have important consequences on the system's stability. We explain here what functions we used to implement discretisation.

Sensors: time and value discretisation

The sensors blocks implements inside time and value discretisation as we can see on Fig. 11.

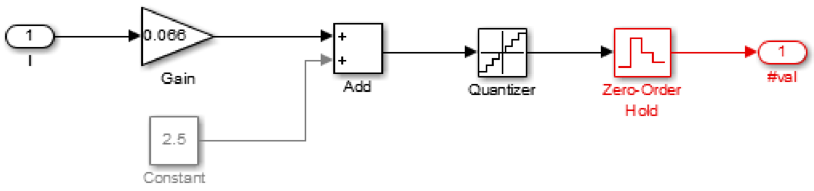


Figure 11: Sensor block implementation

On this figure we can see 3 parts:

- The first one, with the gain and the adder correspond to the scaling part. This will convert the unit of the value measured into a new scale to be numerised.
- The second one is the quantizer. As the sensors are numerical sensors, the value rescaled will be converted into integer, this is value discretisation, to be coded into a finite number of bit.
- The last one is the zero-order hold. The sensor does not give the value in continuous-time, but gives the value at a certain time period. So, to model this, we held the sensor value during the time period.

Signal output: time conversion

At the contrary, the signal output block (in our case the PWM block) implements the conversion back to the time continuous domain.

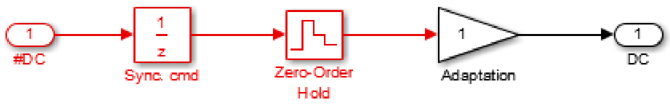


Figure 12: Signal output block implementation

On figure 12 we can see 3 parts:

- The first one is in fact a delay. To model the time gap between the acquisition of values by the microcontroller and the output computed with the control-law, we put this delay corresponding to one acquisition period.
- The second one is the zero-order hold. The microcontroller computes only one output value by period, we put this block to keep this value on the output during the period.
- The last one is just a block to tell Simulink that we are effectively coming back into continuous time-domain from here.

1.4.4 Code implementation on Simulink

During the Simulink implementation we use a specific block named ‘MATLAB Function’. This block is used in two different ways. Firstly to enter easily system equations as classic Matlab functions, this is the case for the Segway model block. Secondly to implement a C-like code as soon as we are ready, as closed as possible to the future implemented code on the microcontroller, this is the case for numerical treatment and implementation of command laws.

The main advantages of this block is the fact that the code will be reusable for the microcontroller. The syntax is not so far from arduino code and with a simple copy-paste and some adaptations, in particular for variables, we obtain the final code. On top of that, with the simulation we can test, compare with the initial blocks (e.g. for PID) and debug our algorithm without electronic parts.

As soon as we do that, the main work which has to be done with the arduino is the configuration and the scheduling of the different tasks.

2 Measuring the attitude of the Segway

2.1 Note on the sensors

2.1.1 Accelerometer

The acceloremer used is a LMS303 sensor. The characteristics of the sensor, as set by the Arduino software, according to the datasheet will be the following ones:

- A sensitivity range of ± 8 g.[1]
- A sensitivity of ± of 4mg/digit, knowing that the value returned is stored on a 12-bit integer, where the highest bit is used for the sign, and the 11 lower bits for value coding.
- A data output rate of 50Hz.

The main source of noise on this sensor is the quantization noise, which we will model directly in the Simulink model.

2.1.2 Compass

The magnetometer used is a LMS303 sensor. The characteristics of the sensor, as set by the Arduino software, according to the datasheet will be the following ones:

- A sensitivity range of ± 4 gauss.[2]
- A sensitivity of ± of 0.16mgauss/digit, knowing that the value returned is stored on a 16-bit integer, where the highest bit is used for the speed sign, and the 15 lower bits for value coding.
- A data output rate of 6.25Hz.

The main source of noise on this sensor is the quantization noise, which we will model directly in the Simulink model.

2.1.3 Gyro

The gyro used is a L3G sensor. The characteristics of the sensor, as set by the Arduino software, according to the datasheet will be the following ones:

- A sensitivity range of $\pm$ 2000 °/s $\Longleftrightarrow$ $\pm$ 34.9 rad/s.[3]
- A sensitivity of ± of 70m°/s/digit, knowing that the value returned is stored on a 16-bit integer, where the highest bit is used for the speed sign, and the 15 lower bits for value coding.
- A data output rate of 100Hz and a high-pass filter cutoff frequency of 8Hz, while the low-pass filter cutoff frequency is 12.5Hz.[4]

The main source of noise on this sensor is the quantization noise, which we will model directly in the Simulink model. We can also add a random noise for taking into account the non-linearity of the sensor, which represents at maximum 4°/s.
The digital zero-rate level is not taken into account as it is corrected every time the board is plugged in.

2.1.4 Current sensor

The current sensor we are going to use is an analog current sensor integrated to the two choppers connected to each motor. The current sensor returns a voltage proportional to the current (between -30A and 30A). It outputs a voltage of 66mV/A, centered at 2.5V. According to the manufacturer's datasheet, the typical error on this measurement is below 1.5%.
This voltage will then be read by the 10-bits ADC of the Arduino to reconstruct the value of the current. This gives us a quantisation interval of 4.88mV, which corresponds to 74mA.

[1]The other available ranges are ± 2 g, ± 4 g, ± 6 g or ± 16 g
[2]The other available ranges are ± 2 gauss, ± 8 gauss or ± 16 gauss
[3]The other available ranges are ± 245 °/s or ± 500 °/s
[4]These values can be changed by the user, see datasheet for more details

2.2 Filter for fusion of data

2.2.1 Rationale

On a Segway, it appears that we only need to estimate the pitch angle (θ) of our system. Therefore, we assume that the roll angle is always null (that is to say, that both Segway's wheels are at the same altitude), and that the measure of the yaw angle is useless. Therefore, we need to be able to measure the angle on only one dimension and only the gyros and the accelerometers will be used on the Inertial Measurement Unit.
The angle of pitch angle can be obtained by two methods:

- By integration of the measurement of the gyros: $\theta_g = \frac{1}{s}\dot{\theta}_{gyro}$
- By measuring the gravity with the accelerometers in near-static situations: $\theta_a = \text{atan}\left(\frac{a_x}{a_z}\right)$

The issue with the first method is that the numerical integration leads to the accumulation of errors, which results in the drift of angle estimation. Moreover, the gyros might have a bias, which will lead to a fast diverging estimation of the angle.
The issue with the accelerometers is that at high dynamics, their measurement is disturbed by all the other accelerations.

2.2.2 Existing filters

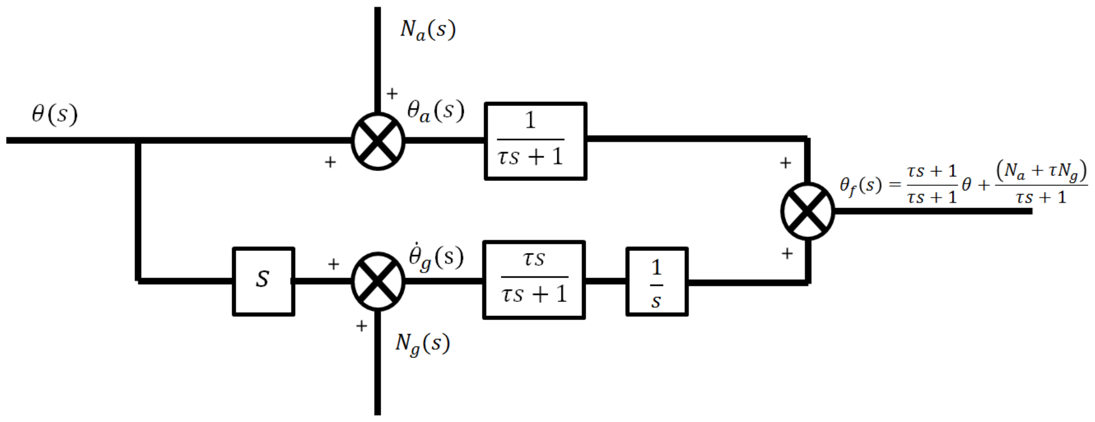
Therefore, the idea is to use the gyros in dynamic situations, and correct the pitch angle estimation at low dynamics with the measurement of the accelerometers. Therefore, there exist different types of filters to make the fusion of both data from the gyros and accelerometers:

- Kalman filter
- Complementary filter

The Kalman filter is a well-known filter, which is optimal. It is used on many applications on inertial measurement units (IMU) but requires high computational effort (although this is not so true if we are in 1D). It is very interesting that this filter is optimal and adaptive, however, it requires knowledge of the covariances of the noise on the estimated states and measured outputs. Moreover, this process is a complicated mathematical process and does not match with the pedagogic objective to implement this project as a fourth year project in systems engineering.
The complementary filter is a much simpler process. The idea is to filter the measurements from the gyros with a high-pass filter, and the measurements from the accelerometers with a low-pass filter. Both filters have the same time constant and the result is a linear transfer function. This solution requires less computational effort, and therefore this is the one we have chosen to implement.

2.2.3 First-order complementary filter

This filter is very simple. It can be summed up by the process below:



$N_a(s)$
$\theta(s)$
+
+
$\theta_a(s)$
$\frac{1}{\tau s+1}$
+
$\theta_f(s) = \frac{\tau s+1}{\tau s+1}\theta + \frac{(N_a + \tau N_g)}{\tau s+1}$
+
S
+
$\dot{\theta}_g(s)$
$\frac{\tau s}{\tau s+1}$
$\frac{1}{s}$
+
$N_g(s)$

Figure 13: First order complementary filter

We can see that the estimated angle θ_f reflects well the real angle, and that the noise from the accelerometers and gyros is filtered.

Another interesting thing is that this filter can also be written:

$$\theta_f = \frac{1}{1+\tau s}\theta_a + \frac{\tau s}{1+\tau s}\frac{1}{s}\dot{\theta}_g \tag{46}$$

$$= \frac{K_p}{K_p+s}\theta_a + \frac{s}{K_p+s}\frac{1}{s}\dot{\theta}_g \tag{47}$$

Further calculations lead to:

$$\theta_f = K_p\frac{1}{s}\left(\theta_a - \theta_f\right) + \frac{1}{s}\dot{\theta}_g \tag{48}$$

We can see that the first order complementary filter can simply be seen as a P-controller where the reference signal would be θ_a and the measured output θ_f. We have $K_p = \frac{1}{\tau}$. This can be represented by the following structure:

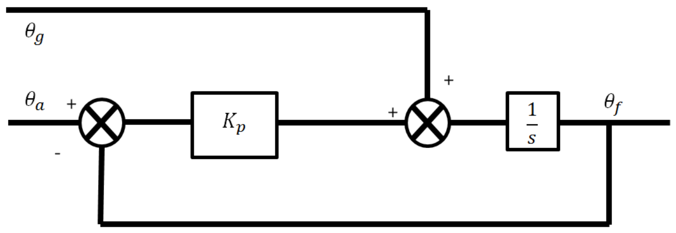


Figure 14: First order complementary filter seen as a P controller

However, it can be noticed that this filter does not explicitly estimate the drift of the gyros.

2.2.4 Second-order complementary filter

To have an explicit estimation of the drift of the gyros, the idea is to introduce a second-order filter. One very simple idea to do it is to use the previous structure, presented in Fig.14 but instead of using a simple P-controller, we use a PI-controller. This gives us the following structure:

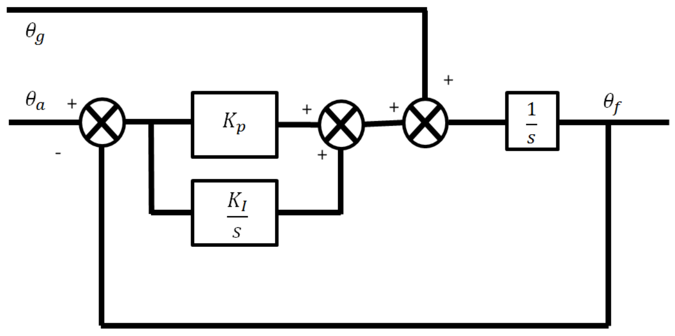


Figure 15: Second order complementary filter seen as a PI controller

This gives us the following function of our filter:

$$\theta_f = \left(K_p + K_i\frac{1}{s}\right)\frac{1}{s}\left(\theta_a - \theta_f\right) + \frac{1}{s}\dot{\theta}_g \tag{49}$$

With further calculations, it can be transformed to the following equation:

$$\theta_f = \frac{1+\frac{K_p}{K_i}s}{1+\frac{K_p}{K_i}s+\frac{1}{K_I}s^2}\theta_a + \frac{\frac{1}{K_I}s^2}{1+\frac{K_p}{K_i}s+\frac{1}{K_I}s^2}\theta_g \tag{50}$$

If we consider, as was done previously, that:

$$\begin{aligned} \theta_a &= \theta_f + N_a \\ s\theta_g &= s\theta_f + N_g \end{aligned} \tag{51}$$

Then, one can obtain:

$$\theta_f = \frac{1+\frac{K_p}{K_i}s+\frac{1}{K_I}s^2}{1+\frac{K_p}{K_i}s+\frac{1}{K_I}s^2}\theta_f + \frac{N_a+\frac{K_p}{K_I}N_a s+\frac{1}{K_I}N_g s}{1+\frac{K_p}{K_i}s+\frac{1}{K_I}s^2} \tag{52}$$

In that way, we can see that the noise is strongly reduced.

2.2.5 Tuning of the complementary filter

From Eq.50 one can deduce the following parameters:

$$\begin{aligned} \omega_w &= \sqrt{K_I} \\ \xi &= \tfrac{K_P}{2\sqrt{K_I}} \end{aligned}$$

Therefore, we just need to choose the cutoff frequency ω_0 and the damping ratio ξ to tune K_P and K_I. Usually we want to avoid oscillations on the output of the filter, therefore we choose a high damping ratio ($\xi \approx 2$).

2.3 Summary

We use the information from the gyros to compute the pitch angle recursively. The angle can also be computed from the accelerometers, still using them only during non-dynamic period, ie $\|a\| \approx 1g$. Let θ_a the pitch angle extracted from the accelerometers.

$$\theta_a = \text{atan}\left(\frac{a_x}{a_z}\right) \tag{53}$$

Now, we can compute the rate of correction ω_a:

$$\epsilon_{(k+1)} = \theta_{c,(k)} - \theta_{a,(k+1)} \tag{54}$$

$$\omega_{a,(k+1)} = \omega_{p,(k+1)} + \omega_{I,(k+1)} \tag{55}$$

$$\omega_{p,(k+1)} = K_p \epsilon_{(k+1)} \tag{56}$$

$$\omega_{I,(k+1)} = \omega_{I,(k)} + K_I \Delta T \epsilon_{(k+1)} \tag{57}$$

The corrected rate of rotation ω_c is then:

$$\omega_{c,(k+1)} = \omega_{a,(k+1)} + \omega_{g,(k+1)} \tag{58}$$

Where ω_g is the rate of rotation measured by the gyros.
In the end, we can compute the pitch angle:

$$\theta_{(k+1)} = \theta_{(k)} - \omega_{c,(k+1)} \Delta T^5 \tag{59}$$

[5] Not that in this part the signs are according to how we have placed the gyros and acceleros on our Segway, and might change with a different configuration

3 Control design

3.1 State-Space representation

3.1.1 Linearisation for control design

The set of equations obtained in Eq.31, in section 1.1, is highly non-linear. To be able to design a corrector for this system, we need to linearise it around its equilibrium point. To that end, we made the following assumptions:

- $\theta \approx 0$
- $\dot{\theta} \approx 0$
- $\theta + \beta \approx 0$

This leads to the following:

- $\cos(\theta) \underset{0}{\sim} 1$
- $\cos(\theta+\beta) \underset{0}{\sim} 1$
- $\sin(\theta) \underset{0}{\sim} \theta$
- $\sin(\theta+\beta) \underset{0}{\sim} \theta+\beta$

Thus we found the following linearised set of equations:

$$\begin{cases} U = RI + L_r\dot{I} + \frac{K_m K_g \dot{z}}{R_w} \\ (M_{eq})\,\ddot{z} + (ml + M_p l_p)\,\ddot{\theta} = F_{ext} \\ J_{eq}\ddot{\theta} + (ml + M_p l_p)\,\ddot{z} = mgl\theta + M_p l_p g\,(\theta+\beta) \end{cases} \tag{60}$$

3.1.2 State-Space model

Equations 60 can be rewritten to put them in a state-space form. We assume that all resisting forces can be considered as perturbations and thus not be considered in the state-space representation of our system. We also assume that the angle β acts as a perturbation on our system. As was mentioned in section 1.2, we also decided to ignore the dynamic induced by the inductance of the motor's winding.
We thus have the following set of equations:

$$\begin{cases} U = RI + \frac{K_m K_g \dot{z}}{R_w} \\ (M_{eq})\,\ddot{z} + (ml + M_p l_p)\,\ddot{\theta} = \frac{2C_m}{R_w} \\ J_{eq}\ddot{\theta} + (ml + M_p l_p)\,\ddot{z} = mgl\theta + M_p l_p g\,(\theta+\beta) \\ C_m = K_g K_m I \end{cases} \tag{61}$$

One can then find the following equations:

$$\ddot{\theta} = \frac{2\,(ml + M_p l_p)\,K_m K_g}{\left((ml + M_p l_p)^2 - J_{eq}M_{eq}\right) R R_w} U - \frac{2\,(ml + M_p l_p)\,K_m^2 K_g^2}{R_w^2 R\left((ml + M_p l_p)^2 - J_{eq}M_{eq}\right)}\dot{z} - \frac{(ml + M_p l_p)\,g M_{eq}}{(ml + M_p l_p)^2 - J_{eq}M_{eq}}\theta \tag{62}$$

$$\ddot{z} = \frac{(ml + M_p l_p)^2\,g}{(ml + M_p l_p)^2 - J_{eq}M_{eq}}\theta + \frac{2J_{eq}K_m^2 K_g^2}{R_w^2 R\left((ml + M_p l_p)^2 - J_{eq}M_{eq}\right)}\dot{z} - \frac{2J_{eq}K_m K_g}{\left((ml + M_p l_p)^2 - J_{eq}M_{eq}\right) R R_w} U \tag{63}$$

We define our state vector: $x = \begin{bmatrix} \dot{z} & \dot{\theta} & \theta \end{bmatrix}^T$, our command vector: $u = \begin{bmatrix} U \end{bmatrix}^T$ and our output vector: $y = [\theta]$. We then find the following state-space form:

$$\begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx + D \end{cases} \tag{64}$$

With the following matrices:

$$\begin{aligned}
A &= \begin{bmatrix} \frac{2J_{eq}K_m^2K_g^2}{R_w^2R\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)} & 0 & \frac{(ml+M_pl_p)^2g}{(ml+M_pl_p)^2-J_{eq}M_{eq}} \\ -\frac{2(ml+M_pl_p)K_m^2K_g^2}{R_w^2R\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)} & 0 & -\frac{(ml+M_pl_p)gM_{eq}}{(ml+M_pl_p)^2-J_{eq}M_{eq}} \\ 0 & 1 & 0 \end{bmatrix}\\
B &= \begin{bmatrix} -\frac{2J_{eq}K_mK_g}{\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)RR_w} \\ \frac{2(ml+M_pl_p)K_mK_g}{\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)RR_w} \\ 0 \end{bmatrix}\\
C &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\\
D &= 0
\end{aligned}$$

3.2 Architecture of the control design

The first step to design the control of our Segway is to analyse the interfaces of our system. Our only input is the voltage U, and the available outputs are θ, $\dot{\theta}$ and I.
If we want to stabilize our Segway, we need to provide a force which is equal to the horizontal force created by the gravity on the Segway as the driver leans forward or backward. Therefore, when the driver leans forward, the Segway will have a constant acceleration, and to maintain a constant speed, the driver will lean back to the vertical. If he wants to stop, he has to lean backward to make the Segway decelerate. Therefore, to control our Segway, we must control the output torque on the motors. We propose to use the following control architecture:

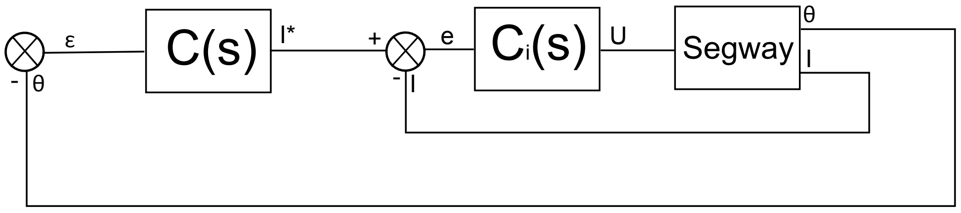


Figure 16: Architecture of the control of the Segway

Therefore, we can see on Fig.16 that we have two control loops:

- A fast control loop regulating the current, and which controls the Segway's motors voltage
- A slower control loop regulating the angle of the Segway, to guarantee the stability of the Segway.

3.3 Design of the position loop

3.3.1 Design in continuous-time

To design the position loop, we assume that the Segway is controlled in torque given to the each wheel (which is closely linked to the intensity provided by the motors as $I = \frac{C_m}{K_mK_g}$). If we make this assumption then we have the following equation:

$$\ddot{\theta} = \frac{2\left(ml + M_pl_p\right)}{\left(\left(ml + M_pl_p\right)^2 - M_{eq}J_{eq}\right)R_w}C_m - \frac{\left(ml + M_pl_p\right)gM_{eq}}{\left(ml + M_pl_p\right)^2 - M_{eq}J_{eq}}\theta \tag{65}$$

We want to find the transfer function between C_m and θ, that is to say $H_{\theta,\mathrm{BO}}(s) = \frac{\theta(s)}{C_m(s)}$.
We can write Eq.65 using the Laplace formalism:

$$C_m(s) = \left(\frac{\left(\left(ml + M_pl_p\right)^2 - M_{eq}J_{eq}\right)R_w}{2\left(ml + M_pl_p\right)}s^2 + \frac{gM_{eq}R_w}{2}\right)\theta(s) \tag{66}$$

Therefore, our transfer function is:

$$\begin{aligned}
H_{\theta,\mathrm{BO}}(s) &= \frac{\frac{2}{gM_{eq}R_w}}{\frac{\left((ml+M_pl_p)^2-M_{eq}J_{eq}\right)R_w}{gM_{eq}R_w(ml+M_pl_p)}s^2 + 1} && (67)\\
&= \frac{K_\theta}{\tau_\theta s^2 + 1} && (68)
\end{aligned}$$

This system is naturally unstable. To stabilise it, we will implement a PID controller. After some tests, we decided to set the performances wanted in closed-loop:

- A rising time of 85ms
- A phase margin of 70°, as our control needs to be very robust to noise.

The structure of the PID controller implemented is the following:

$$C(s) = K_{P,\theta} + K_{I,\theta}\frac{1}{s} + K_{D,\theta}s \tag{69}$$

It is very interesting to notice that, as we can measure directly $\dot{\theta}$, there is no need to derivate numerically θ, thus we reduce the computational effort as no filter on the derivation has to be implemented. The numerical values obtained are:

- $K_{P,\theta} = 426$
- $K_{I,\theta} = 55$
- $K_{D,\theta} = 85$

Thus we have the following control structure:

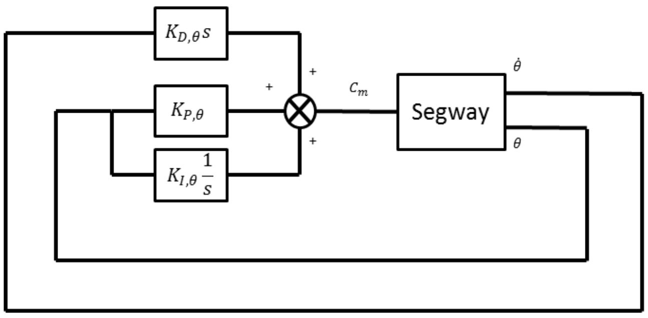


Figure 17: Control loop on the angle of the Segway

In the end, we have the following bode-plot of our system:

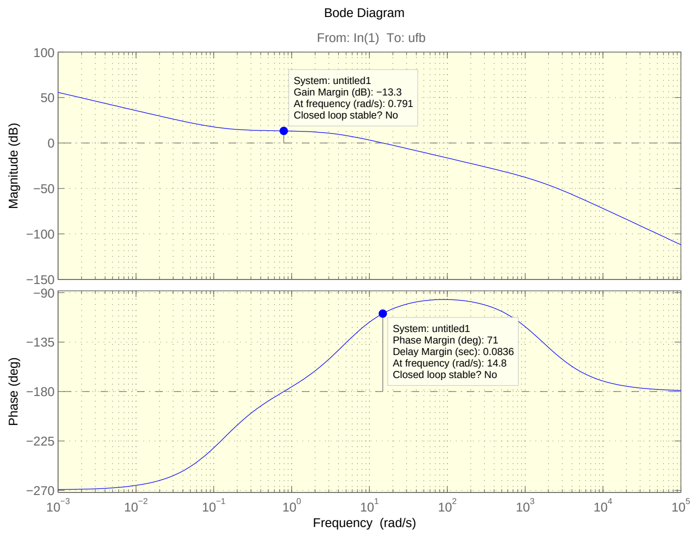


Figure 18: Bode plot of the position loop control

3.3.2 Implementation in discrete-time

As θ and $\dot{\theta}$ can only be sampled at a sampling rate of 50Hz, we need to test our controller in discrete-time. The controller will be implemented on the Arduino using the Tustin's transform, that is to say:

$$s = \frac{2}{T_s}\frac{z-1}{z+1} \tag{70}$$

Where T_s is the sampling rate, and z the delay operator.

Thus the structure of the PID controller will be:

$$C(s) = K_{P,\theta} + K_{I,\theta}\frac{T_s}{2}\frac{z+1}{z-1} + K_{D,\theta}\frac{2}{T_s}\frac{z-1}{z+1} \tag{71}$$

If we now have a look at the bode plot of our system in discrete time, we have the following figure:

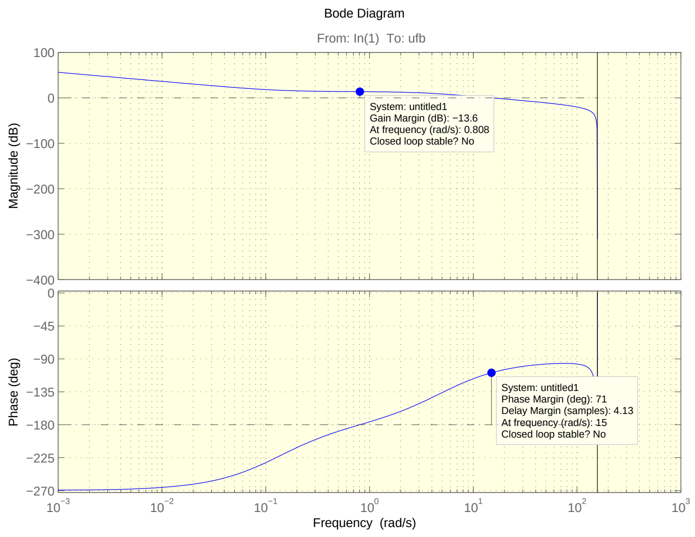


Figure 19: Bode plot of the position loop control in discrete time

We can see that the sampling rate does not affect much our system, which is quite interesting. Tests on the Matlab model were also conducted to be sure that our system in discrete time was sufficiently robust.

3.4 Design of the current loop

3.4.1 Design in continuous-time

For designing the current loop, we want to find the transfer function between the input voltage U and the output current, I. For that, we can write that:

$$I = \frac{U}{R} - \frac{K_m K_g}{R_w R}\dot{z} \tag{72}$$

From the state-space representation found in Section 3.1, we can deduce the following state-space representation:

$$\left\{\begin{array}{rl} \begin{bmatrix}\ddot{z}\\ \ddot{\theta}\\ \dot{\theta}\end{bmatrix} &= \begin{bmatrix} \frac{2J_{eq}K_m^2K_g^2}{R_w^2R\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)} & 0 & \frac{(ml+M_pl_p)^2g}{(ml+M_pl_p)^2-J_{eq}M_{eq}} \\ -\frac{2(ml+M_pl_p)K_m^2K_g^2}{R_w^2R\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)} & 0 & -\frac{(ml+M_pl_p)gM_{eq}}{(ml+M_pl_p)^2-J_{eq}M_{eq}} \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix}\dot{z}\\ \dot{\theta}\\ \theta\end{bmatrix} + \begin{bmatrix} -\frac{2J_{eq}K_mK_g}{\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)RR_w} \\ \frac{2(ml+M_pl_p)K_mK_g}{\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)RR_w} \\ 0 \end{bmatrix}[U] \\ I &= \begin{bmatrix} -\frac{K_mK_g}{R_wR} & 0 & 0 \end{bmatrix} \begin{bmatrix}\dot{z}\\ \dot{\theta}\\ \theta\end{bmatrix} + \left[\frac{1}{R}\right][U] \end{array}\right. \tag{73}$$

As the control loop will target for $\theta = 0$, we can assume that $\theta \approx 0$ and thus that the dynamic on θ can be ignored, and has no influence on $\ddot{z}$. Thus, we can write the set of equations 73 as:

$$\left\{\begin{array}{rl} [\ddot{z}] &= \left[\frac{2J_{eq}K_m^2K_g^2}{R_w^2R\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)}\right][\dot{z}] + \left[-\frac{2J_{eq}K_mK_g}{\left((ml+M_pl_p)^2-J_{eq}M_{eq}\right)RR_w}\right][U] \\ I &= \left[-\frac{K_mK_g}{R_wR}\right]\begin{bmatrix}\dot{z}\\ \dot{\theta}\\ \theta\end{bmatrix} + \left[\frac{1}{R}\right][U] \end{array}\right. \tag{74}$$

Eq.74 can be rewritten in the following form:

$$\left\{\begin{array}{rl} \dot{x} &= Ax + BU \\ Y &= C_Ix + D_IU \end{array}\right. \tag{75}$$

If we use the Laplace Transform, one can write:

$$\begin{cases} sX(s) &= AX(s) + BU(s) \\ Y(s) &= C_I x(s) + D_I U(s) \end{cases}$$

The state equation can also be written

$$\begin{cases} (sI - A)\, X(s) &= BU(s) \\ \iff X(s) &= (sI - A)^{-1}\, BU(s) \end{cases}$$

In the output equation, we can replace $X(s)$ by the above equation:

$$\begin{cases} Y(s) &= \Big(C\,(sI - A)^{-1}\, B + D\Big)\, U(s) \\ \iff H_I(s) &= \frac{Y(s)}{U(s)} \end{cases}$$

Where $H_I(s)$ is a transfer function matrix. In our case, we have a SISO system, therefore the transfer function matrix is one dimensional, and is just the transfer function of our system.
Thus we find the following transfer function:

$$\begin{aligned} H_I(s) &= \tfrac{CB}{s-A} + D \\ &= \tfrac{CB+Ds-DA}{s-A} \end{aligned}$$

One can notice that:

$$\begin{aligned} CB &= \frac{2J_{eq}K_m^2K_g^2}{\big((ml+M_pl_p)^2 - J_{eq}M_{eq}\big)R^2R_w^2} \\ DA &= \frac{2J_{eq}K_m^2K_g^2}{R_w^2R^2\big((ml+M_pl_p)^2 - J_{eq}M_{eq}\big)} \\ CB &= DA \end{aligned}$$

Thus, our transfer function has the form:

$$H_I(s) = \frac{Ds}{s - A} \tag{76}$$

This transfer function has the form of a highpass filter. It has the following Bode plot:

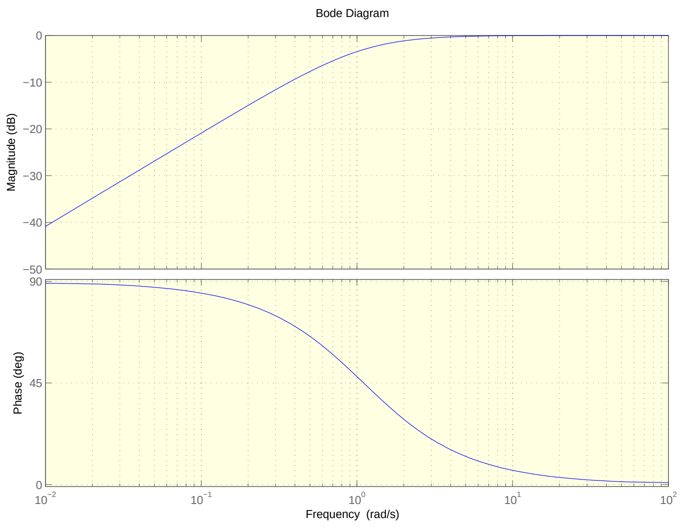


Figure 20: Bode plot of the transfer function of the current

Due to the presence of a derivator at low frequency, the feedback loop will always naturally tend towards 0 when a step in inputted. To cancel this effect, we simply use an I controller with a high gain. We chose $K_I = 100$.

3.4.2 Design in discrete-time

The implementation of a PI controller in discrete time is fairly simple. We decided to have a faster current loop, working at a sampling rate of $T_I = 200\text{Hz}$. Tests gave good results at this sampling rate, implementing our controller with the following structure:

$$C_I(z) = K_I T_I \frac{z}{z-1} \tag{77}$$

Note that this time, we use the backward Euler's discretisation rather than the Tustin's discretisation, which requires lower computational effort while keeping the conditions of stability, as this current loop will run at a faster sampling time.
We now have to take into account the fact that the input voltage will be restricted to [-36V;+36V]. This will induce a saturation at the output of the controller. This will cause the I controller to keep growing, which might then reduce its ability to correct the system with a new input in an acceptable time, and can lead to severe numerical issues when implemented on the Arduino. To avoid this issue, we implement an anti-windup with a gain $K_w = 4K_I$. The result of this process is shown on the figure below:

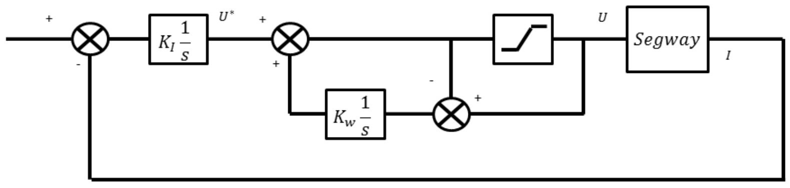


Figure 21: Control Structure of the current loop with anti-windup

3.5 Safety features

3.5.1 Limitation of the motor's temperature

In order to avoid the destruction of the motors, we have a supervision of the temperature of the two motors. As we could not find the thermal time constant of our DC motor, we decided to use a very conservative approach. As most DC motors of this size have a thermal time constant between 20 and 30 min, we assumed that for each of our DC motor, the thermal time constant when the temperature is rising was 10min, and 30min when the temperature decreases. Thus, to monitor the temperature of each of our motor, we measure the current, take its squared value, to represent the Joule's losses in the motor's winding, and applied a filter with a time constant that depends on the sign of the derivative of the measured current. If the square root of the output reaches a value close to the nominal current value for which the motor is rated, we limit the current in our motor to the rated nominal current. A warning will be raised and announced to the driver, via a LED or a buzzer, when we will be reaching 90% of this value, so that the driver can safely stop and let the motors cool for a few minutes. The calculation of the motor's temperature is shown in Fig.22 below:

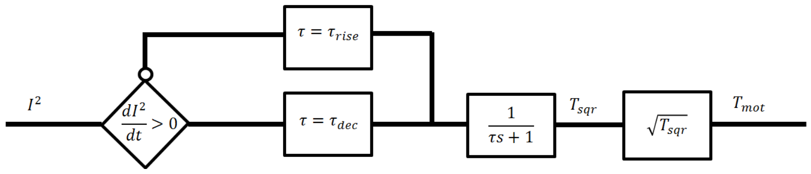


Figure 22: Control of the temperature of the two motors

3.5.2 Limitation of the speed

As our motors are not powerful enough, it is difficult for us to prevent the driver from going over a certain speed. What we have decided to do is thus to raise a warning to the driver when the Segway reached a speed over 2 m/s (7km/h).

3.5.3 Fall of the Segway

In case the Segway would fall (angle over 15 °) we have implemented a function that makes the Segway's wheels go to a full stop within 1s.

3.6 Turning

To turn, we decided to control directly the voltage on both motors. The current control loop thus controls the mean of the current on the two motors it outputs the mean of the voltage on each motor. The voltage on both motors is thus symmetrical with a difference proportional to the steering angle of the driver.
To calculate the gain of the steering, we determine the maximum rotational speed that we want at the maximum steering angle (30rpm at 10°). The $30rpm = \gamma_{max}$ can be converted in a voltage difference. If L_w is the length between the two wheels, then this makes a speed of $V_w = \frac{L_w}{2}\gamma_{max}$. This can be converted into a specific voltage on the motor: $U = K_m K_g \frac{V_w}{R_w} = K_m K_g \frac{\gamma_{max}}{2R_w}$. Thus, the gain is $K_\gamma = K_m K_g \frac{\gamma_{max}}{2R_w \alpha_{steer,max}}$, where $\alpha_{steer,max}$ is the maximum steering angle. In our case, this gives us a gain: $K_\gamma = 1.08V/°$.

4 3D modelling of the Segway

4.1 CAD

Most of the mechanical components were bought before the start of the project. Those come from spare components for electrical scooters. Thus, we began by modelling those components through CAD software: CATIA V5. Those components were the two DC motors with their pinions, the two wheels with their chain rings, the battery and the handles.

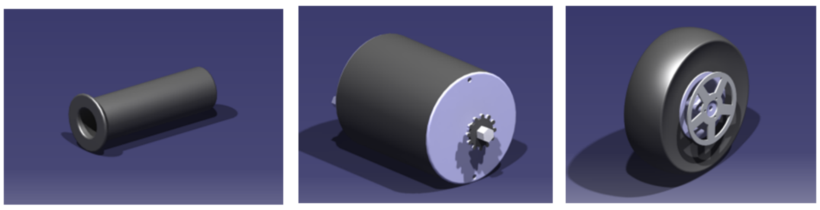


Figure 23: Different components of the Segway

We thus had to make the design and the sizing of the main frame and the handle bar. Those assemblies had to be ergonomic and safe for the user.

We started by searching the best way to arrange these parts in order to reduce the total volume. We obtained the final arrangement shown on Fig.24. The two motors cannot be put below the main frame because otherwise the height between the frame and the ground would not be ergonomic for the user. Indeed, with the motors below, the step would be 22 centimeters high and with this arrangement the step is 15 centimeters high. The battery is between the user's feet. As for the motors, we have had to put it above the frame. It is just fastened with Velcro on the structure.

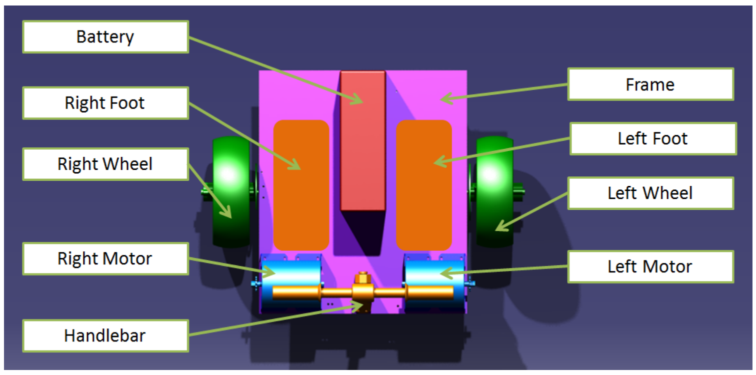


Figure 24: Disposition of the components of the Segway

4.2 Mechanical design

4.2.1 Motors

The transmission between the motor and the wheel is made by pinions and chain with a ratio of 64/11. In order to fasten the motor on the frame we designed 4 metal brackets by motor, with slotted holes to permit to set the tension of the chain. We can see in the next figure those squares. Each motor has four threaded holes to fasten it, two at the front and two at the rear. We can see in Fig.25, the motor with its metal brackets.

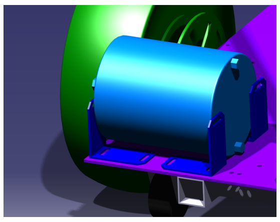
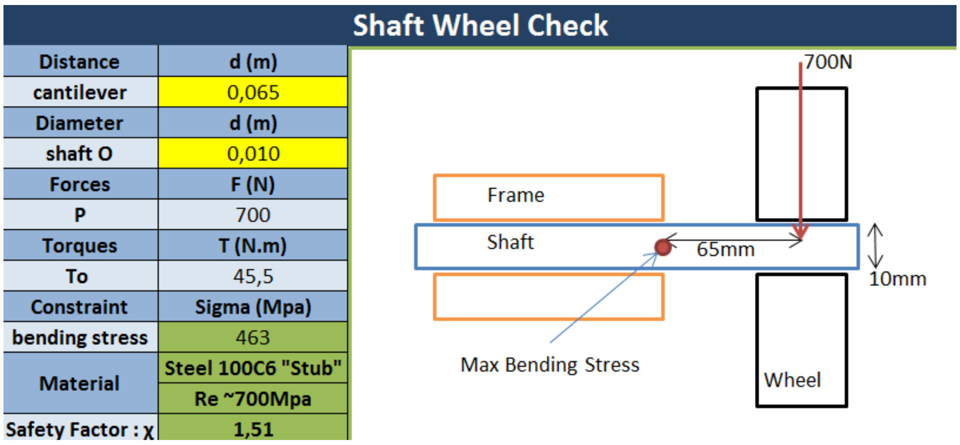


Figure 25: Motor with its metal brackets

4.2.2 Wheels

The wheels we received had already bearings with an inner diameter of 10 mm. But the received shafts were two M10 screws with a length of 150 mm. Their length was too small for our architecture. Indeed, each wheel is fastened on one side and consequently the forces and torques on the shaft near the frame are twice higher. Thus we had to design and manufacture our own shafts. It is simply designed with one shoulder to block the wheel on the first extremity and with a Circlip on the other one. We computed the maximum bending stress near the frame and checked the sizing. You can see those computations on Fig.26 below, we took 700 Newtons for the applied force because we have the lower structure with a mass of 40 kg and we took a maximum mass for the user of 100kg. Our shaft is made with a "‘Stub bar"’ (an alloy steel). We obtained a safety factor of 1.51. We only did a static analysis because our system does not have a high angular speed and moreover it will not be much used.



Shaft Wheel Check

Distance	d (m)
cantilever	0,065
Diameter	d (m)
shaft O	0,010
Forces	F (N)
P	700
Torques	T (N.m)
To	45,5
Constraint	Sigma (Mpa)
bending stress	463
Material	Steel 100C6 "Stub"
	Re ~700Mpa
Safety Factor : χ	1,51

700N
Frame
Shaft
65mm
10mm
Max Bending Stress
Wheel

Figure 26: Wheel shaft check

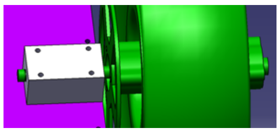


Figure 27: Wheel with its axis and frame

4.2.3 Safety Components

In order to increase the user’s safety we added two components. The first one is a caster on each Segway corner. It ensures a maximum tilt angle of the Segway with the ground of 18°. Those casters are low cost products and can be bought on reseller like "‘RS"’. Fig.28 below shows one of the four casters.

The second part is a housing for the chain on each side of the user’s feet. Indeed each foot is very close to the rolling chain. With 3D printing process, we can obtain an ergonomic and effective protection. We can see it in the figure below.

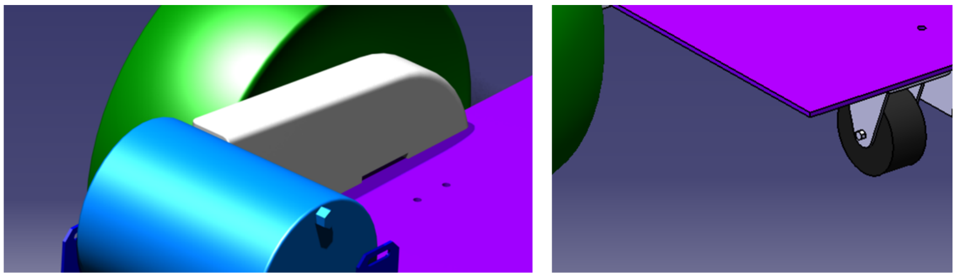


Figure 28: Safety components

4.2.4 Frame

The design of the lower structure is very important as it is the structure which will transmit all the weight of the Segway and the person to the ground via the wheels. Therefore, we have decided to design this structure using a Finite-Element model in order to make sure that our structure would resist the stress induced and also to reduce the weight of this structure as much as possible. The model of the structure we have decided to design is the following one:

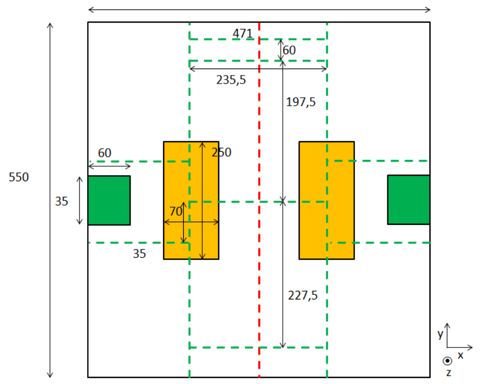


Figure 29: Schematic of the structure

The black square represent an aluminum plate which is the base of our Segway. The dashed green lines represent the positions of reinforcing beams. The green zones represent the zones on which the wheels are attached and the orange ones represent the footprint of the person.
The dashed red lines represent a plane of symmetry. We will make use of this symmetry and thus only model half of our structure in a Finite-Element model, using Patran, as is shown in Fig.30 below:

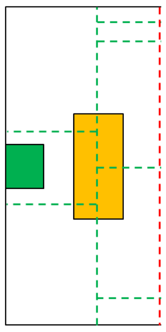


Figure 30: Schematic of the modelled structure

Thus we have meshed this structure in Patran.

Materials:

The plate, represented by the black box on Fig.29 is made of Aluminium. The reinforcing beams will be in steel, due to welding constraints, as we cannot weld Aluminum in our university. The properties of these materials are summed up in the table below:

Name	Young's Modulus (GPa)	Poisson's ratio	Density (Kg/m^3)	R_e (MPa)
Aluminium (5754 O)	70	0.35	2700	80
Steel (S235)	210	0.3	7800	235

Table 4: Materials properties

Modeling elements:

Both the beams and the plate will be designed for bending. Therefore, we will use thin shell elements to model the aluminium plate and beam elements to model the beams. The properties of these elements will vary later to try to optimise our structure.

Boundary conditions:

The green zone being the zone on which the wheels are attached, we assume that the whole green zone is blocked in all translations, that is to say $T_x = T_y = T_z = 0$. On the plane of symmetry along the X axis, we will block the translation along the Y axis, and rotations around the X and Z axis.

Loading:

We assumed that the whole loading is concentrated on the footprint of the driver. This is not totally accurate, but simplifies the model used. We decided to consider that the total weight of the Segway was 150Kg (120kg driver + 30kg structure), which is an ultimate loading scenario. These 150kg will be equally distributed to both orange zones, which approximately gives us a pressure of 0.043 N/mm^2.
We also took into account the bending moment created on the handle bar on the ultimate loading case, which creates a force of ± 3100N on each bearing, and thus on each reinforcing beam.

Meshing:

The next step is to mesh our structure. Due to the simplicity of our structure, it was very easy to use isometric meshing. This gave us the following result:

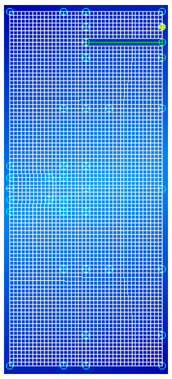
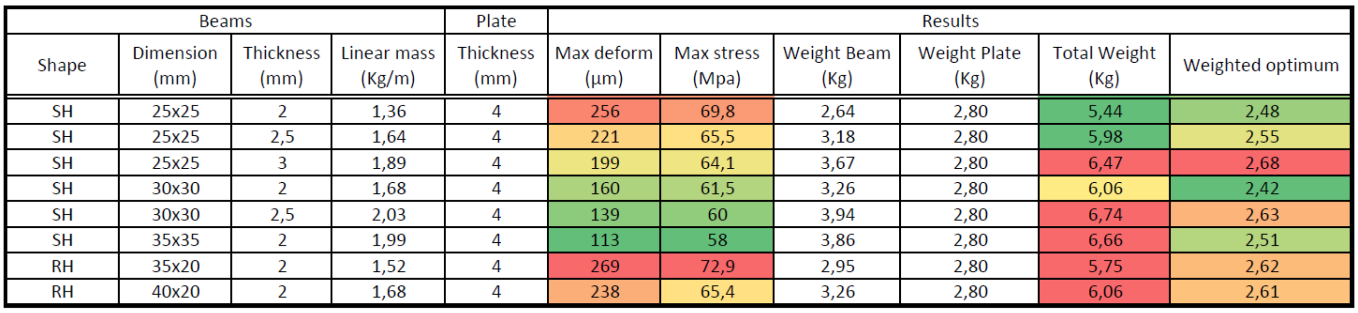


Figure 31: Meshing of our structure

It is interesting to notice that there is a gap of elements between the two upper reinforcing beams. This corresponds to a groove that will be made on the aluminium plate. This groove allows more displacement between the two reinforcing beams and thus reduces the stress induced by the bending moment on the handlebar.

Results:

Our objective was to minimise the mass of our structure while limiting the deformations and reducing the stress of our structure. To that end, we tested different thicknesses of plate and different profiles of beams. The different results are reported in the table below:



Beams				Plate	Results					
Shape	Dimension (mm)	Thickness (mm)	Linear mass (Kg/m)	Thickness (mm)	Max deform (µm)	Max stress (Mpa)	Weight Beam (Kg)	Weight Plate (Kg)	Total Weight (Kg)	Weighted optimum
SH	25x25	2	1,36	4	256	69,8	2,64	2,80	5,44	2,48
SH	25x25	2,5	1,64	4	221	65,5	3,18	2,80	5,98	2,55
SH	25x25	3	1,89	4	199	64,1	3,67	2,80	6,47	2,68
SH	30x30	2	1,68	4	160	61,5	3,26	2,80	6,06	2,42
SH	30x30	2,5	2,03	4	139	60	3,94	2,80	6,74	2,63
SH	35x35	2	1,99	4	113	58	3,86	2,80	6,66	2,51
RH	35x20	2	1,52	4	269	72,9	2,95	2,80	5,75	2,62
RH	40x20	2	1,68	4	238	65,4	3,26	2,80	6,06	2,61

Figure 32: Results obtained for different beams profiles and different plate thickness

The shapes are:

- RH: Rectangular Hollow
- SH: Square Hollow

The maximum stress is the stress found using the Von Mises criterion.
To help in the decision of which plate to choose, we applied a cost function to find the optimal structure:

$$C = \left(\frac{u}{u_{max}}\right)^{\frac{1}{2}} + \left(\frac{\sigma}{\sigma_{max}}\right)^{\frac{1}{2}} + \left(\frac{M}{M_{max}}\right)^{3} \tag{78}$$

In the end, we conclude that the best combination is to use a square hollow section 30x30x1.5mm with a 4mm thick aluminium plate. This design is optimal as it gives us a maximum stress of 61.5MPa and a maximum displacement of 0.160mm, for a total mass of the structure of 6.06 Kg. Thus the safety factor of our structure is $K_r = \frac{R_{e,alu}}{\sigma_{max}} = \frac{80}{61.5} = 1.3$. We also have a maximum stress in the reinforcing beams of 175MPa, thus giving us a safety factor of $K_r = \frac{R_{e,steel}}{\sigma_{max}} = \frac{235}{175} = 1.35$
The results obtained are shown below:

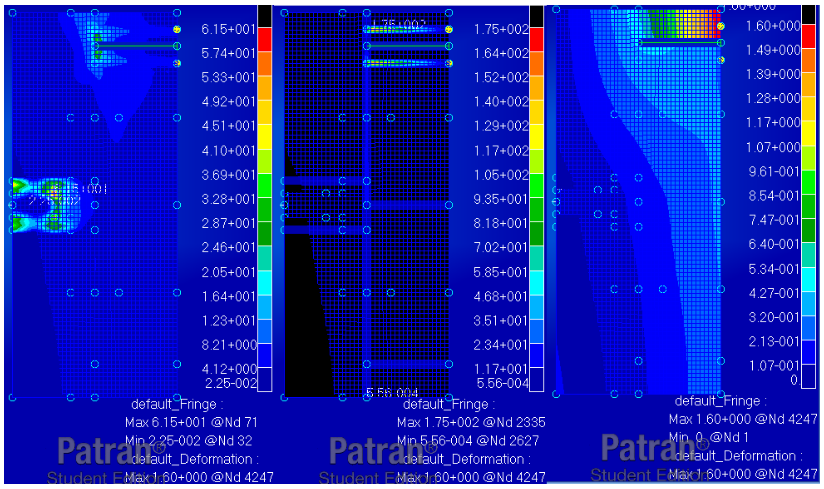
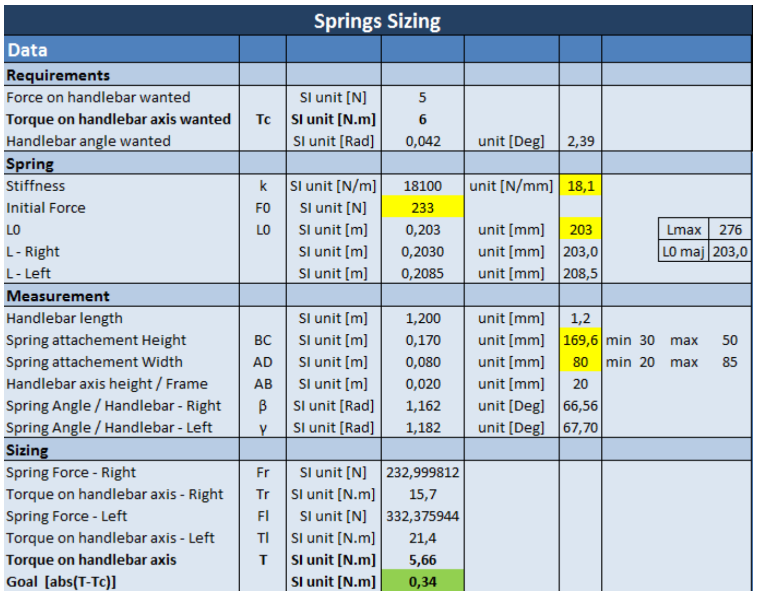


Figure 33: Results obtained for our final choice (left: stress fringe, middle: stress in the reinforcing beams, right: displacement fringe)

4.2.5 Handlebar

We had to design the Segway handlebar with its pivot joint. The main constraint on this assembly was the spring forces repartition. Indeed for an ergonomic driving, one of the requirements was to have a force feedback when the user leans the handlebar to the left or to the right. As a matter of fact, with a gyropode like a Segway we do not turn the handle around the vertical. This requirement imposes a force of 5 Newtons felt by the user when the handles are leant of 5 centimeters to the left or to the right.

Our handlebar is simply composed of two tubes with two handles at the top. These two tubes are fastened with a 3D printing part and glue (epoxy resin). The pivot joint is made with plain bearing in bronze. To make the force feedback, we decided to place two springs on each side of the handlebar. Their arrangement is shown in the figure below. Thanks to computations made on excel, we could size and find the spring which can fulfil the requirement. The goal of this spreadsheet was not only to check the use of one kind of spring, but also to define its position. It was an iteration process in order to find a spring which fits with all constraints. Indeed, we could only obtain 4.7 Newtons compared to the 5N wanted. With those ones we already have a high force, 330N at the maximum, and they are quite big compared to the other parts. They have been found in a French reseller: 'Vanel'. We can see below different table and pictures of our study for this sizing and the result in CAD software.



Springs Sizing						
Data						
Requirements						
Force on handlebar wanted		SI unit [N]	5			
Torque on handlebar axis wanted	**Tc**	**SI unit [N.m]**	**6**			
Handlebar angle wanted		SI unit [Rad]	0,042	unit [Deg]	2,39	
Spring						
Stiffness	k	SI unit [N/m]	18100	unit [N/mm]	18,1	
Initial Force	F0	SI unit [N]	233			
L0	L0	SI unit [m]	0,203	unit [mm]	203	Lmax 276
L - Right		SI unit [m]	0,2030	unit [mm]	203,0	L0 maj 203,0
L - Left		SI unit [m]	0,2085	unit [mm]	208,5	
Measurement						
Handlebar length		SI unit [m]	1,200	unit [mm]	1,2	
Spring attachement Height	BC	SI unit [m]	0,170	unit [mm]	169,6	min 30 max 50
Spring attachement Width	AD	SI unit [m]	0,080	unit [mm]	80	min 20 max 85
Handlebar axis height / Frame	AB	SI unit [m]	0,020	unit [mm]	20	
Spring Angle / Handlebar - Right	β	SI unit [Rad]	1,162	unit [Deg]	66,56	
Spring Angle / Handlebar - Left	γ	SI unit [Rad]	1,182	unit [Deg]	67,70	
Sizing						
Spring Force - Right	Fr	SI unit [N]	232,999812			
Torque on handlebar axis - Right	Tr	SI unit [N.m]	15,7			
Spring Force - Left	Fl	SI unit [N]	332,375944			
Torque on handlebar axis - Left	Tl	SI unit [N.m]	21,4			
Torque on handlebar axis	**T**	**SI unit [N.m]**	**5,66**			
Goal [abs(T-Tc)]		**SI unit [N.m]**	**0,34**			

Figure 34: Spring sizing

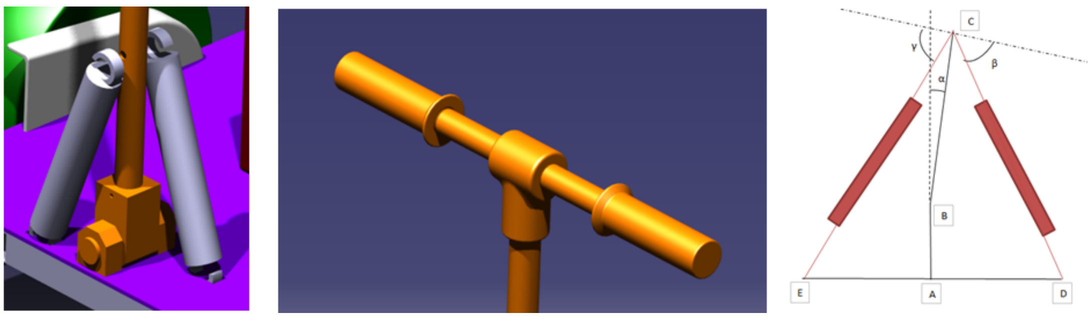
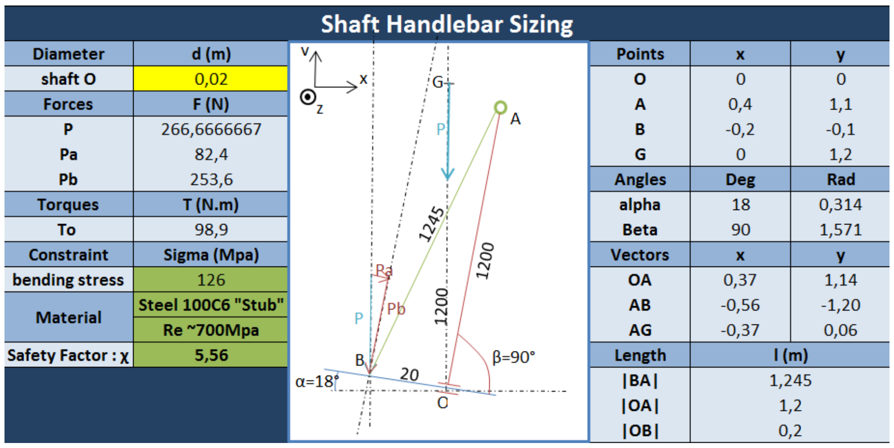


Figure 35: Views of the design of the handlebar

The shaft of the pivot joint had to be designed well because the user put a non-neglectable amount of forces on the top of the handlebar which leads to a high bending stress on the latter. Thus, we modelled the force of the user on the handles when the Segway is stopped on its casters, which implies a tilt angle of 18°with the horizontal axis. Fig.36 below presents our modelling which permits to obtain the maximum bending stress on the shaft. The shaft is made with a 'Stub bar' (an alloy steel) and allows to obtain a high security factor with 20 millimeters on the diameter. We preferred to keep this high value to ensure the robustness of this critical assembly.



Shaft Handlebar Sizing

Diameter	d (m)
shaft O	0,02
Forces	**F (N)**
P	266,6666667
Pa	82,4
Pb	253,6
Torques	**T (N.m)**
To	98,9
Constraint	**Sigma (Mpa)**
bending stress	126
Material	Steel 100C6 "Stub"
	Re ~700Mpa
Safety Factor : χ	5,56

v
x
z
G
P
A
1245
1200
Pa
Pb
P
1200
B
20
β=90°
α=18°
O

Points	x	y
O	0	0
A	0,4	1,1
B	-0,2	-0,1
G	0	1,2
Angles	**Deg**	**Rad**
alpha	18	0,314
Beta	90	1,571
Vectors	**x**	**y**
OA	0,37	1,14
AB	-0,56	-1,20
AG	-0,37	0,06
Length	**l (m)**	
|BA|	1,245	
|OA|	1,2	
|OB|	0,2	

Figure 36: Sizing of the handlebar

5 Performance analysis

In order to analyse the performance of the system, and to confirm the compatibility of the power chain's parts with those performance, we simulate five different scenarios as below:

- Starting test with an offset angle: simulate the Segway's reaction if we turn on the Segway stabilisation with an offset angle.
- Battery life test: test if the battery is big enough to go at 15km/h during 2 hours.
- Slope test: capability to ride up a slope of 5% at 6km/h.
- Abrupt driver command: simulate the Segway's reaction to an abrupt driver command that ask a current of 30A.
- Angle sensor noises and chocks: capability of the system to reject perturbations coming from gyroscope and accelerometer.

As we are using the Simulink model that does not implement the power losses due to the imperfection of the power chain, we will compute in some tests new values for power or current taking into account an efficiency of 60% (the efficiency of the motor is estimated at 70% and the rest globally at 85%).

5.1 Starting test with an offset angle

5.1.1 Objectives and conditions

This test aims to simulate how the Segway will react if the command law for stabilisation is turned on whereas the Segway is not exactly vertical. This is because we are not exactly sure of the verticality when a person takes the Segway and starts it.

- The initial angle of the Segway (θ) = 3°= 0.05236 rad
- Angle of the driver's body (β) = 0°

5.1.2 Means of test

To make this test on the Simulink model, we just have to put a constant block with value '0' on the input beta, and put an initial condition inside the Segway modelling block on the integrator block that gives the theta value.

5.1.3 Results

We obtained the results on the evolution of the Segway's angle and position as below.

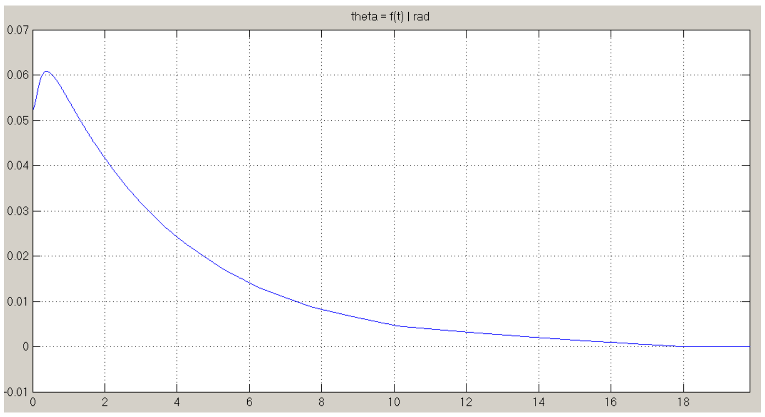


Figure 37: θ (Segway's angle) response in radian function of time in second

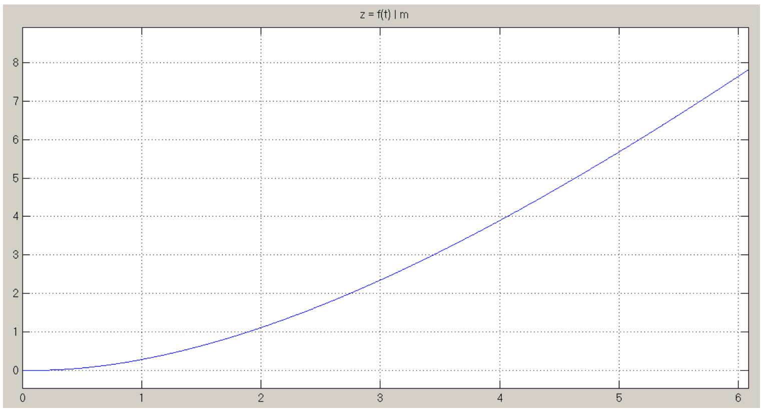


Figure 38: z (Segway position) response in meter function of time in second

We can see on figure 37 that the angle is slowly reduced down to 0°at 18s, and on figure 38 that the Segway will move forward if the driver does not react, starting slowly then going faster.

5.1.4 Conclusion

The behaviour on the theta angle is correct, there is not sudden and abrupt reaction. Talking about the moving forward is a bit more complicated, the Segway will only move of 30cm during the first second. This should be sufficient to take the Segway in hand and begin to drive it.

5.2 Battery life test

5.2.1 Objectives and conditions

This test checks the sizing of our battery, it simulates how much energy we will use in this conditions:

- Speed ($\dot{z}$) = 15km/h = 4.17 m/s
- Slope (γ) = 0 °

5.2.2 Means of test

In order to go at 15km/h, we have to make a little enslavement of beta with a proportional corrector as we can see on figure 39.

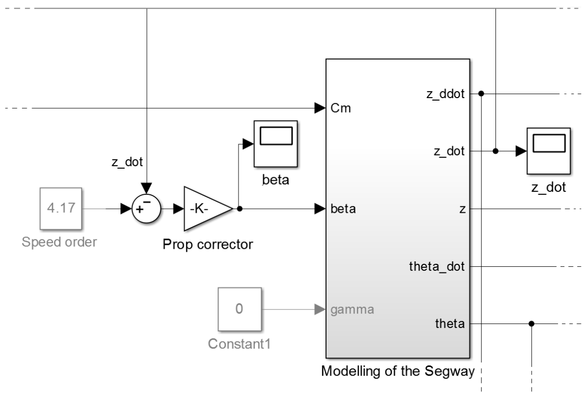


Figure 39: Feedback of speed to have the value of β that gives a speed of 4.17 m/s

After running the simulation, we take the steady state value of the tension and intensity at the input of one motor, compute the power and then compute the energy needed by the system during 2 hours.

5.2.3 Results

We have the following results:

- $Um_{ss} = 30.5V$
- $Im_{ss} = 1.6A$

So the power used is :

$$Pm_{ss} = Um_{ss} \times Im_{ss} = 48.8W$$

Considering the 60% efficiency of the power chain, the power at the output of the battery will be of 81.3W for one motor, and thus 162.6W for two motors. So the energy consumed is:

$$Em_{ss} = Pb_{ss} \times T = 162.6 \times 2 = 325.3Wh$$

5.2.4 Conclusion

All this, with our 36V battery, ask a capacity of $325.3 \div 36 = 9.0Ah$. Our battery has a capacity of 12Ah, so we should be able move at 15km/h during 2 hours.

5.3 Slope test

5.3.1 Objectives and conditions

In this test, we want to check if the Segway is able to ride up a slope that corresponds to an access for disabled people at 6km/h without meet any power or thermal limit of the system. The input datas are:

- Slope angle (γ) = 5% = 2.86°= 0.05 rad
- Speed ($\dot{z}$) = 6km/h = 1.67m/s

5.3.2 Means of test

We proceed in the same way as in the previous test to maintain a speed of 6km/h. After running the simulation, we take the steady state values of power and intensity to check that they will be in the motor's specifications' limits.

5.3.3 Results

We have the following results:

- $Um_{ss} = 16.7V$
- $Im_{ss} = 5.1A$

So the power used in steady state, taking into account the efficiency:

$$Pm_{ss} = \frac{Um_{ss} \times Im_{ss}}{\eta} = \frac{16.7 \times 5.1}{0.6} = 142.0W$$

5.3.4 Conclusion

In steady state the power limit of the motor is 500W and the current limit is 18A, so the Segway is able to ride up this slope. But, a remark has to be made, if the driver tries to accelerate too fastly, the Segway will meet the current's peak limit (fixed at 30A).

5.4 Abrupt driver command

5.4.1 Objectives and conditions

This test aims to know the maximum step order that the driver can give at startup (speed 0km/h, slope 0°). This maximum step order corresponds to the maximum acceleration that the Segway can give from no speed, this means that it reaches the current limit of 30A.

5.4.2 Means of test

We test different step angles for a short time (1 second), and find the one for which we reach the 30A limit but that does not make the system unstable.

5.4.3 Results

The result of this test is 0.21 rad (12°), that gives a peak acceleration of $3.5m/s^2$.

5.4.4 Conclusion

This performance is correct for an home-made Segway. We can notice that abrupt orders are not safe as limits of the system in reality may vary (imperfections, forces and phenomenons not modelled...).

5.5 Angle sensor noises and shocks

5.5.1 Objectives and conditions

In the reality, the signal of the angle sensors (θ) should be disturbed by noises and vibrations. In order to see the reaction of the Segway in this case, we will add noises on the sensors signal.

5.5.2 Means of test

To simulate noises and vibrations, we use white-noises generators that comes at the input of sensors. The test conditions will be an angle beta of 3.4°during 10 seconds that give a steady state speed of 3m/s. As we can see in figure 40.

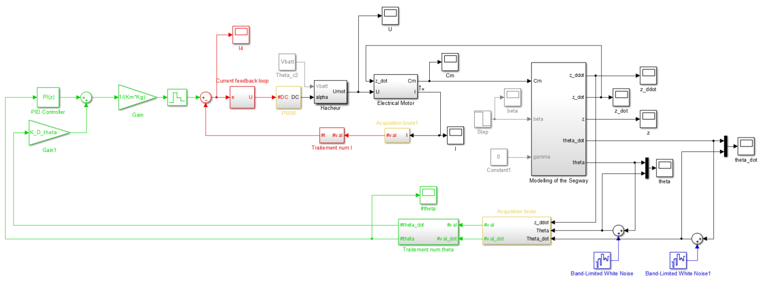


Figure 40: New Simulink schematic with white-noises generators (in blue) and beta step signal

Without noises, the speed and angles are the following ones:

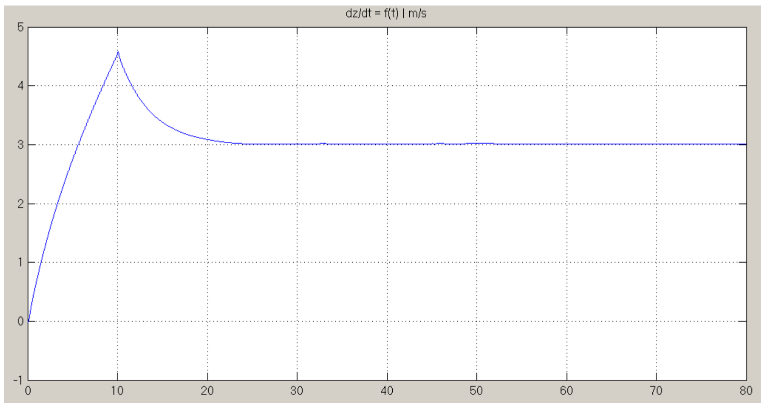


Figure 41: Speed without noises

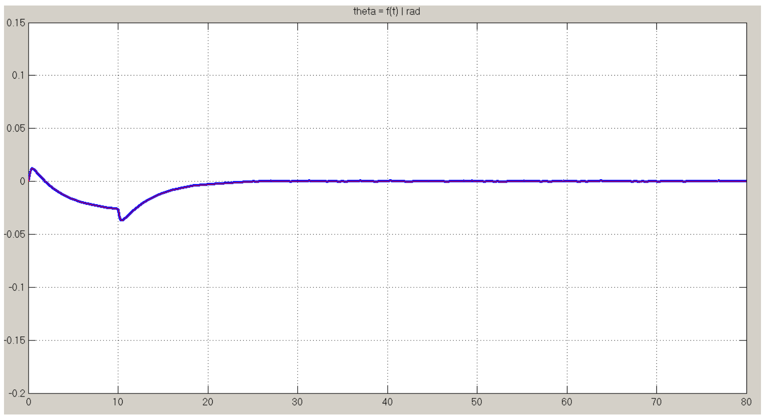


Figure 42: Theta without noises

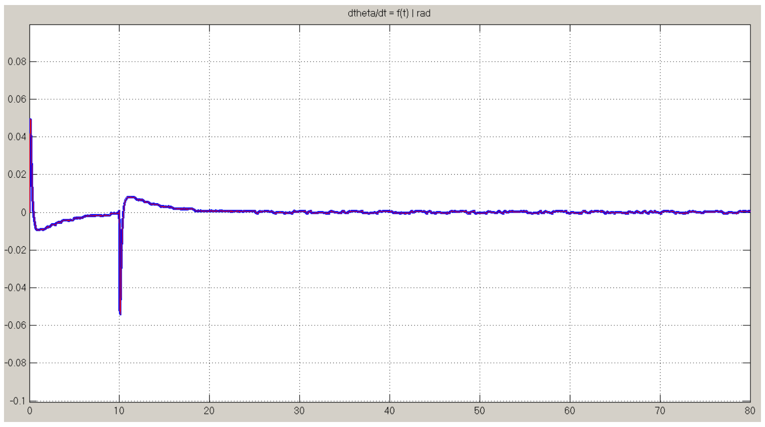


Figure 43: Theta derivative without noises

5.5.3 Results

We now add the white noises, and we obtain following results:

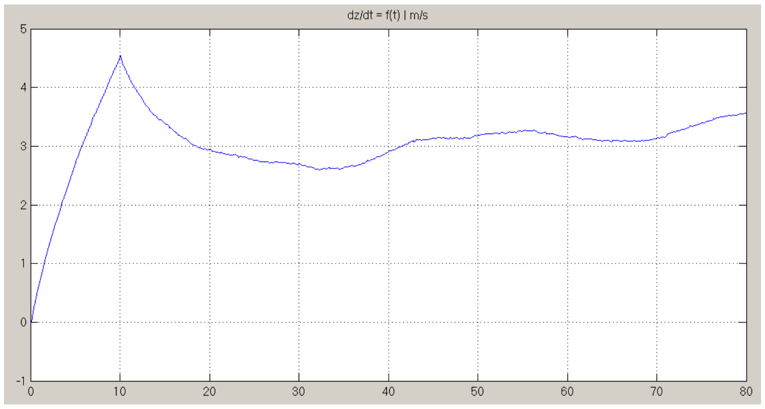


Figure 44: Speed without noises

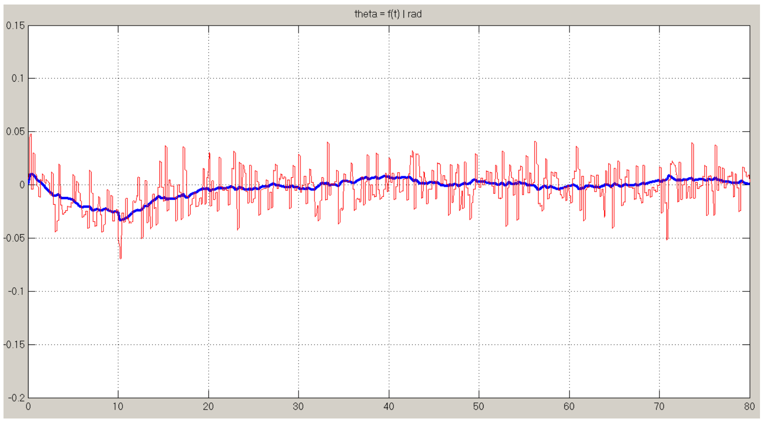


Figure 45: Theta with noises. Theta signal in blue, with noises in red

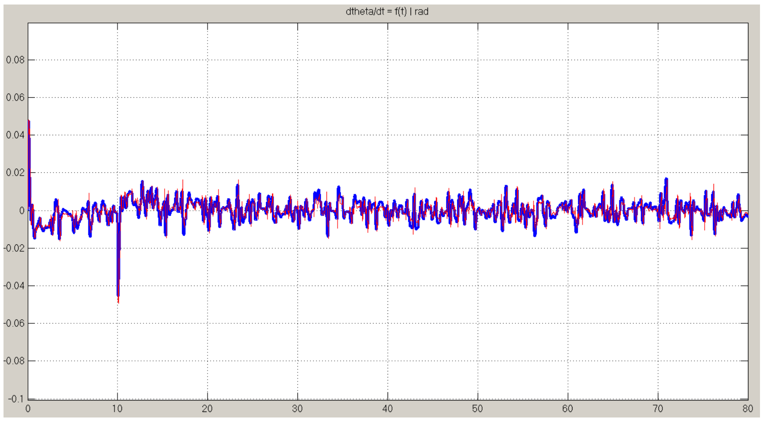


Figure 46: Theta derivative with noises. Theta derivative signal in blue, with noises in red

What we can observe firstly is the absence of stable steady state on figure 44, the system is highly influenced by the added noises.

Secondly we can take a look at figures 45 and 46. We can see for both that they are not the same as previously, but one good thing is that the system rejects relatively well theta's noises (due to numerical treatment). At the contrary, this is not the case for theta derivative because the blue signal fits the noises. This is the main perturbation on the system, another test without perturbations on theta derivative confirms it.

5.5.4 Conclusion

As a result, the behaviour of the system is not satisfactory, even if the driver reacts, it will be difficult to control correctly the Segway. An additional filter is needed at least on theta derivative. But this filter might deteriorate the Segway's performances.

Conclusion

This project aimed to design a virtual prototype of a Segway in order to be used as a teaching project. It was really pleasant and challenging. Indeed we did not have much time and it was a multidisciplinary project, as we have explored many disciplines such as mechanics, electronics or control design.

In order to create the virtual prototype, we began by computing its mathematical model thanks to the Lagrange's equations. Then, the identification of the physical parameters of the components permitted us to use properly the model through simulation software such as Simulink or Modelica. With Modelica we could test our model in 3D and also explore some control design for our system. Simulink allowed to model more precisely our control design and also our electronics implementation such as the sensors. With a CAD software (Catia V5) we did the mechanical design for this prototype.

Our main difficulty stood in the control design. Indeed a Segway system is a complex system to control. Moreover, as the motors and other parts had already been bought, we were limited on the variables to control. With more time we could certainly have made a real prototype. All the electronic implementation was not mentioned and has to be done to build it.

A Fusion of data in a 3D-environment

A.1 The rotation matrix

Before trying to estimate the attitude of our Segway, the first very important step is to define the coordinate frames precisely. The usual representation of the attitude of a solid is the Euler angles. This is represented in Fig.47. On this figure, the coordinate frame $B_g = (x, y, z)$ is the ground reference and $B_s = (x', y', z')$ is the frame attached to the solid that is our IMU in our case.

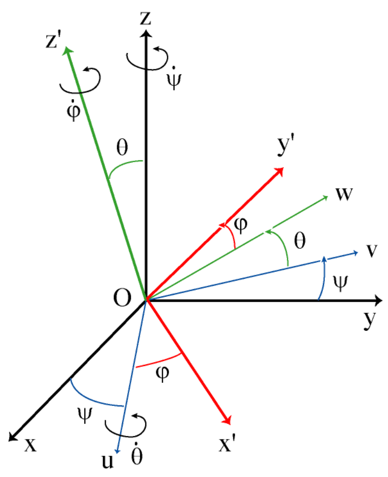


Figure 47: Euler angles [6]

The common Euler angles of a solid to define its attitude are denoted as:

- The pitch angle θ
- The roll angle ϕ
- The yaw angle ψ

Let V be a vector. We denote V_{B_g} the vector V expressed in the coordinate frame B_g and V_{B_s} the vector V expressed in the coordinate frame B_s. A simple relation between these two expressions of the same vector can be found:

$$V_{B_g} = R_{s\to g} V_{B_s} \tag{79}$$

Where $R_{s\to g}$ is referred as the rotation matrix, or direct cosine matrix. It is defined as:

$$R_{s\to g} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{80}$$

It is very important to notice that the rotation matrix is an orthogonal matrix, which means that:

$$R^T_{s\to g} R_{s\to g} = I \tag{81}$$

$$R^{-1}_{s\to g} = R_{g\to s} = R^T_{s\to g} \tag{82}$$

A.2 Updating the rotation matrix

A.2.1 Update from a rotation rate vector

Let assume that ω is our rotation rate vector. Later on, we will see that:

$$\omega = \omega_{gyro} + \omega_{correction} \tag{83}$$

Where:

- ω_{gyro} is the raw measurement of the rotation rate vector given by the gyros

[6] Source : http://commons.wikimedia.org/wiki/File:AngleEuler.png?uselang=fr#globalusage

- $\omega_{correction}$ is a correction term that comes from the complementary filter that realises the fusion of data

Let r be a random vector. Kinematics laws show that:

$$\frac{dr(t)}{dt} = \omega(t) \times r(t) \tag{84}$$

Where $\times$ denotes the cross-vector product.

Thus, it is easy to deduce that:

$$\begin{aligned} r(t) &= r(0) + \int_0^t d\theta(\tau) \times r(\tau) && (85) \\ &= r(0) + \int_0^t \omega(\tau) \times r(\tau) d\tau && (86) \\ & && (87) \end{aligned}$$

Now, if we take a vector of the ground coordinate frame expressed in the coordinate frame of the Segway, we can write that:

$$r_{ground}(t) = r_{ground}(0) + \int_0^t \omega(\tau) \times r_{ground}(\tau) d\tau \tag{88}$$

Using a simple discretisation (backward Euler method) method, we can find:

$$r_{ground}(t + \Delta T) = r_{ground}(t) + \omega(t)\Delta T \times r_{ground}(t) \tag{89}$$

Applying this formula to each axis, one can prove that we obtain:

$$\begin{aligned} R(t + \Delta T) &= R(t) \begin{bmatrix} 1 & -d\theta_z & d\theta_y \\ d\theta_z & 1 & -d\theta_x \\ -d\theta_y & d\theta_x & 1 \end{bmatrix} && (90) \\ d\theta_x &= \omega_x \Delta T && (91) \\ d\theta_y &= \omega_y \Delta T && (92) \\ d\theta_z &= \omega_z \Delta T && (93) \end{aligned}$$

In the end, we have seen how, from the rotation rate vector ω we were able to update recursively the rotation matrix and thus the angles of our Segway defined as:

$$\begin{aligned} \theta &= \operatorname{asin}(X_z) && (94) \\ \phi &= \operatorname{atan}\left(\frac{Y_z}{Z_z}\right) && (95) \\ \psi &= \operatorname{atan}\left(\frac{X_y}{X_x}\right) && (96) \end{aligned}$$

A.2.2 Renormalization

The numerical integration introduced in Eq.89 leads to the accumulation of numerical errors. This accumulation of numerical errors can lead to the loss of the orthonomality of the rotation matrix. These numerical errors are then propagated to the calculation of the angles of the Segway. To solve this issue, the idea is to recreate a rotation matrix at each sample time that is totally orthonormal.
Let $R = \begin{bmatrix} X & Y & Z \end{bmatrix}$.
The principle is fairly simple:
We calculate $e_\perp$ the error of orthogonality by calculating:

$$X.Y = X^T Y \tag{97}$$

We calculate new vectors $X_\perp$ and $Y_\perp$ that are orthogonal:

$$X_\perp = X - \frac{e_\perp}{2} Y \tag{98}$$

$$Y_\perp = Y - \frac{e_\perp}{2} X \tag{99}$$

Finally, we reconstruct the third axis as:

$$Z_\perp = X_\perp \times Y_\perp \tag{100}$$

Now, we have an orthogonal new rotation matrix, however, during the orthogonalisation process, the vectors $X_\perp$, $Y_\perp$ and $Z_\perp$ are no longer normals. To renormalise them, we write:

$$X_N = \frac{X_\perp}{\|X_\perp\|_2} \tag{101}$$

$$= \frac{X_\perp}{\sqrt{X_\perp^T X_\perp}} \tag{102}$$

To avoid costly numerical calculations of square roots and divisions, the key is to notice that $\|X_\perp\|_2$ is always very close to 1. Therefore we proceed to linear approximations:

$$\begin{aligned} X_N &= \frac{X_\perp}{\sqrt{X_\perp^T X_\perp}} \\ &= \frac{X_\perp}{\sqrt{1+\left(X_\perp^T X_\perp - 1\right)}} \\ &\approx \frac{X_\perp}{1+\frac{1}{2}\left(X_\perp^T X_\perp - 1\right)} \\ &\approx X_\perp \left(1 - \tfrac{1}{2}\left(X_\perp^T X_\perp - 1\right)\right) \\ &\approx X_\perp \left(\tfrac{1}{2}\left(3 - X_\perp^T X_\perp\right)\right) \end{aligned}$$

The same can of course be done for the two other axis, and finally we find:

$$X_N = X_\perp \left(\frac{1}{2}\left(3 - X_\perp^T X_\perp\right)\right) \tag{103}$$

$$Y_N = Y_\perp \left(\frac{1}{2}\left(3 - Y_\perp^T Y_\perp\right)\right) \tag{104}$$

$$Z_N = Z_\perp \left(\frac{1}{2}\left(3 - Z_\perp^T Z_\perp\right)\right) \tag{105}$$

In the end, the reconstructed rotation matrix $R_N = \begin{bmatrix} X_N & Y_N & Z_N \end{bmatrix}$ is now completely orthonormal. Therefore, this renormalization process should be repeated at each sample time.

A.2.3 Drift correction

One of the issues of using only gyros is that, even though they are quite efficient in dynamics, they are not in static, and this leads to a drift of the estimated angles. To solve this issues, we will make corrections based on the data obtained from the accelerometers and magnetometers.

Roll & pitch correction

The accelerometers can be used to correct the roll and the pitch quite efficiently. Indeed, when ignoring the phases when the Segway accelerate, the accelerometers provide a good estimation of the gravity vector. The idea is to use this information to correct the drift of the roll and the pitch. If we write $a = \begin{bmatrix} a_{x'} & a_{y'} & a_{z'} \end{bmatrix}^T \equiv g$ we can correct the drift as:

$$\text{Correction}_{RP} = W_{RP} Z_N \times a \tag{106}$$

Where W_{RP} is a weighting used to correct the drift only during static phases, and thus avoid that the measures of the Segway's acceleration influences the measure of the angles.

Yaw correction

Just like the accelerometers, the magnetometers are very good in static. They are able to detect the intensity of the terrestrial magnetic field in each direction. The magnetometers return to us a vector $m_{raw} = \begin{bmatrix} m_{x'} & m_{y'} & m_{z'} \end{bmatrix}^T$. The first step would be to scale these values within the range ± 0.5 to guarantee better numerical calculations.
We can calcultate the tilt compensated magnetic field towards the axis X and Y:

$$M_X = m_x \cos\theta + m_y \sin\phi + \sin\theta + m_z \cos\phi \sin\theta \tag{107}$$

$$M_Y = m_y \cos\phi - m_z \sin\phi \tag{108}$$

Thus, we can deduce the magnetic heading as the angle between the X axis and axis pointing toward the magnetic north:

$$HDG = \text{atan}\left(\frac{-M_Y}{M_X}\right) \tag{109}$$

Knowing this heading, we are therefore able to correct the yaw angle of our Segway. However, it can be noted that this step is optional, as to pilot our Segway we do not require a very accurate measurement of the yaw angle.
The correction plane for the yaw can now be computed as:

$$\text{Correction}_{YAW} = (X_x \sin(HDG) - Y_x \sin(HDG))\, Z_N \tag{110}$$

A.2.4 Fusion of data : Complementary filter

The idea of the second-order complementary filter is to inject the corrections into a PI feedback controller. The principle of this filter is illustrated in Fig.48.

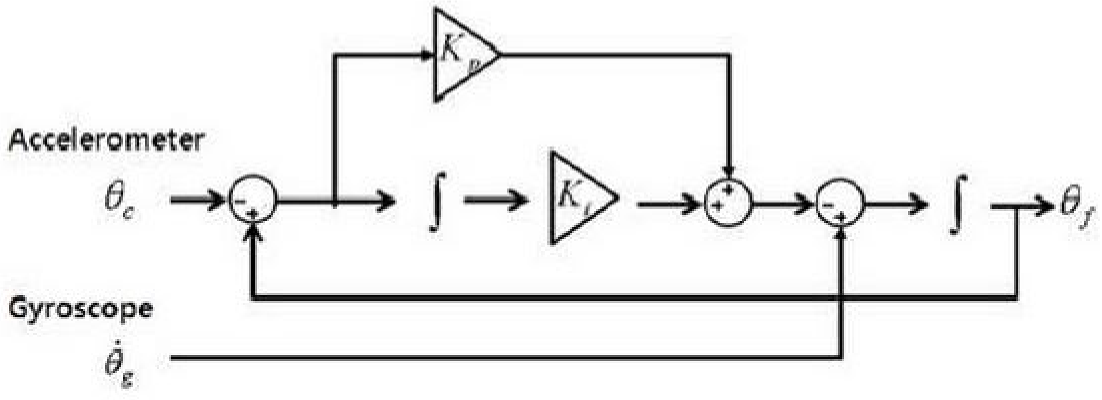


Figure 48: Principle of the complementary filter [7]

Thus we calculate a total correction:

$$\text{Correction} = \text{Correction}_{RP} + \text{Correction}_{YAW} \tag{111}$$

Then, using the PI controller, we can compute $\omega_{correction}$:

$$\omega_{correction} = \omega_{P,correction} + \omega_{I,correction} \tag{112}$$

$$\omega_{P,correction} = K_P \text{Correction} \tag{113}$$

$$\omega_{I,correction} = \omega_{I,correction} + K_I \Delta T \text{Correction} \tag{114}$$

And in the end, we find our new rotation rate vector:

$$\omega = \omega_{gyro} + \omega_{correction} \tag{115}$$

[7]Add source...