POLITECNICO DI TORINO

Facoltà di Ingegneria

Master degree in Mechatronic Engineering

Master Thesis

FlegX

Design and Simulation of a Gravity Compensator with a Noncircular Pulley - Spring Mechanism

Advisor
Prof. Alessandro Rizzo

Company supervisor
Ferdinando Cannella, Ph.D.

Co-advisor
Dr. Cristiano Pizzamiglio
Dr. Daniele Ludovico

Candidate
Francesco Lasagni

April 2019

Acknowledgements

The elaboration of the thesis is a truly motivating challenge to tackle, as it allows to put into practice all the knowledge and skills acquired along the years at university.
And it was an honour and a fortune for me to be able to develop the thesis work at the *Istituto Italiano di Tecnologia* (IIT), one of the most advanced research centres for robotics in Europe.

First of all, I would like to thank Prof. Rizzo for his flawless availability, without whom I could have never had the possibility to undertake this formative activity.
A warm thank you goes to Dr. Cannella, who trusted me and gave me the opportunity not only to develop my thesis, but also to explore the fascinating world of research.
My deepest gratitude goes to Dr. Ludovico, who escorted me from the first days to the conclusion of the thesis. His immense patience, constant support and enthusiasm were essential in guiding me throughout this experience. Thanks also to the rest of the *Advanced Industrial Automation Lab*, for teaching me the importance of kindness and good spirit at the workplace.
Special thanks are due also to Dr. Pizzamiglio, for his all-important remote, yet thoughtful, assistance and his willingness to share his valuable knowledge at literally any time.

I thank my parents, my siblings and Irina: their trust, attentions and support allowed me to reach this milestone, so important to me as it is to them.
I would also like to thank all the teachers encountered along the years, especially Carlo and Mattia, that succeeded in conveying their passion and persistence to leave this world a little better than what we found.
Clearly, I thank all my climbing partners for all the little but precious adventures that we shared: those moments gave me peacefulness and strength to navigate through university.
Finally, I thank Paola for her crucial stubborn and caring support.

Lo svolgimento della tesi è una sfida molto stimolante da affrontare, in quanto dà la possibilità di mettere in pratica tutte le conoscenze e le competenze acquisite negli anni universitari.
Ed è stato un onore e una fortuna poter svolgere il lavoro di tesi presso l'*Istituto Italiano di Tecnologia* (IIT), uno dei più avanzati centri di ricerca per la robotica in Europa.

Desidero innanzitutto ringraziare per la sua impeccabile disponibilità il Prof. Rizzo, senza il quale non avrei mai avuto la possibilità di intraprendere questa esperienza così formativa.
Un caldo ringraziamento va al Dott. Cannella per la sua fiducia e per l'opportunità non solo di svolgere la tesi, ma anche di poter esplorare l'affascinante mondo della ricerca.
La mia gratitudine più sentita va al Dott. Ludovico, che mi ha accompagnato dai primi giorni fino alla conclusione della tesi. La sua immensa pazienza, il costante aiuto ed entusiasmo sono stati essenziali nel guidarmi in questa esperienza. Un ringraziamento va anche al resto dell'*Advanced Industrial Automation Lab*, per avermi insegnato l'importanza di gentilezza e buon umore sul luogo di lavoro.
Un ringraziamento speciale va anche al Dott. Pizzamiglio, per il suo importante supporto accorto e preciso, anche se da remoto, e la disponibilità a condividere le sue preziose conoscenze in qualsiasi momento.

Ringrazio i miei genitori, i miei fratelli e Irina: la loro fiducia, le loro attenzioni e il loro aiuto sono stati per me fondamentali per raggiungere questo importante traguardo.
Desidero ringraziare anche tutti i maestri incontrati negli anni, in particolare Carlo e Mattia, che sono riusciti a trasmettermi la passione e la perseveranza nel lasciare questo mondo migliore di come l'abbiamo trovato.
Doveroso è un ringraziamento a tutti i miei compagni di arrampicata per le piccole ma preziose avventure che abbiamo condiviso: quei momenti mi hanno dato serenità e la forza in questi anni di università.
Infine, ringrazio Paola per il suo fondamentale aiuto ostinato e premuroso.

Contents

List of Figures

List of Tables

Abstract

The purpose of this master thesis is to analyse the effects induced by gravity compensators in robotic systems.

Nowadays robotic research is more focused on attaining energy-efficient and safe solutions. For instance, they are key-aspects of robots that are required to interact with human workers in not confined environments. The introduction of a mechanism that passively compensates the joint torque caused by the weight of the robot may offer a valid solution. Relieving the actuators from generating the torques due to gravity has the consequence of decreasing the power consumption and allowing the possibility to reduce the size and weight of the actuators. Thus, an overall lighter robot is obtained, further reducing the power consumption. Furthermore, in the context of human-robot interaction a lighter robot is potentially less dangerous and can be manoeuvred at higher speeds.
In addition, a gravity compensator allows the robot to hold a static position without the need of an external power source, hence avoiding the risk of collapsing in case of failure of the actuators.

This work is focused on a gravity compensator composed by a noncircular pulley and a spring. This specific gravity compensator features the possibility of generating an arbitrary torque profile, so it is not limited to compensate exclusively the trigonometric torque profiles due to gravity on revolute joints.
First, a virtual prototype of the mechanism applied to an inverted pendulum was developed. This model was initially used to study the static and dynamic behaviour of the mechanism in order to validate its theoretical concept. Following, it was used to guide the design of a physical prototype of the system where the inverted pendulum was actuated by an electric motor. In designing the test bench, particular attention was made to include a system to effectively measure the pretension of the springs.
Once that the test bench had been assembled, a set of experiments was performed to verify the results obtained from the simulations of the virtual model.
Finally, the results are discussed along with problems, limitations and possible improvements of both the virtual model and the physical prototype.

Chapter 1

Introduction

A robot is an artificial machine capable of performing a series of actions to help or replace humans in hard or dangerous jobs. For instance the first industrial robot ever build had the task to handle die castings in order to protect the workers from hot splashes of metal and toxic fumes. It was called Unimate and was designed in the 1950s to work in the assembly lines of General Motor. [2]
Thanks to scientific progress in the fields related to robotics, robots have evolved from being designed exclusively for heavy, monotonous and repetitive tasks in the industrial field to being able to perform more complex routines in numerous fields: medicine, agriculture, exploration and transports, to cite but a few. The introduction of automated systems has resulted in a general enhancement in the production, both in volume and quality, and an improvement of the workers' conditions, yet not without its controversies.

Up to now, most of the robots are designed to be perfectly rigid, in order to withstand high payloads and accelerations, move at high velocity and ensure precision and repeatability of their movements. Achieving these high performances come at the cost of having a bulkier and, above all, heavier structure.
Another prerogative of robots, in particular of robot manipulator, is to have a good manipulability and dexterity, i.e., the ability to reach a desired pose avoiding obstacles. This is obtained through a serial kinematic chain called redundant chain, made of multiple joints. Revolute joints are preferred because of the strength, low friction and reliability of ball bearings. An example is the LBR iiwa by KUKA, which has 7 degrees of freedom (Figure 1.1). A criticality of this type of robots is that the torque due to the weight of the structure increases with the number of degrees of freedom of the mechanism.

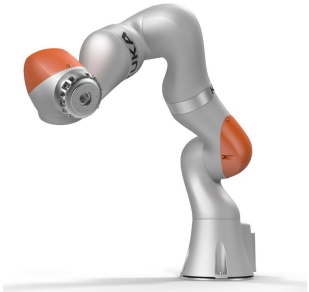


Figure 1.1: LBR iiwa robot.

An increase in the torque due to weight of a heavier, stiffer and longer structure, causes the need of more powerful actuators. In turn a bigger actuator on its own adds weight to the robot and implies a higher power consumption. Moreover, with the increase of circumstances where human-robot cooperation is required, a heavier robots is potentially more dangerous and has to be manoeuvred at lower speeds.

Thus it results clear the benefits of implementing energy-free mechanisms that compensate the torques due to gravity in a revolute joint: the torque needed to move the robot is reduced, which implies a smaller power consumption and the possibility to use a lighter motor to obtain a lighter robot overall. It is remarkable to note that in the case of a perfect weight compensation, the power consumption required to hold a static position is null. Furthermore, such a mechanism would potentially prevent the robot from suddenly collapsing in case of failure of the actuators.
By "energy-free mechanism", it is meant a mechanism that does not require external power and so does not introduce power from outside the system. Hence it compensate the torque due to the weight of the robot by maintaining the potential energy as constant.

1.1 A simple example of gravity compensation

The generic dynamic equation of a multibody system is:

$$H(q)\,\ddot{q}(t) + C(q,\dot{q})\,\dot{q}(t) + B\dot{q} + g(q) = \tau \tag{1.1}$$

The terms used in Equation 1.1 are:

q: generalized coordinates.

$\dot{q}$: generalized velocities.

$\ddot{q}$: generalized accelerations.

$H(q)\,\ddot{q}(t)$: torque due to accelerations.

$C(q,\dot{q})\,\dot{q}(t)$: torque due to centrifugal and Coriolis accelerations.

$B\dot{q}$: torque due to viscous friction.

$g(q)$: torque due to the weight forces.

τ: resultant torque.

In the case of a simplified system that uses bearings, friction can be neglected without compromising the significance of the results. The equation becomes:

$$H(q)\,\ddot{q}(t) + C(q,\dot{q})\,\dot{q}(t) + g(q) = \tau \tag{1.2}$$

A gravity compensation mechanism is designed to compensate the term due to gravity $g(q)$, so that Equation 1.2 becomes:

$$H(q)\,\ddot{q}(t) + C(q,\dot{q})\,\dot{q}(t) = \tau$$

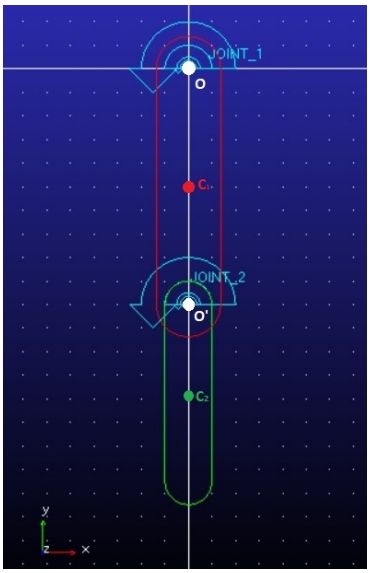


Figure 1.2: Double pendulum example.

To better understand the impact of a weight compensation system, the behaviour of a double pendulum working at different speeds is analysed. The software MSC Adams is used to simulate the dynamic behaviour of the system (Figure 1.2). A perfect weight compensation could be easily introduced by simply deactivating the gravity field.
In the simulation, the link are ideal rigid uniform bodies and the joint are ideal revolute joints. The fact that there is no friction in the system is not a issue as a simplified model is under study.
The goal of the simulation is to compute the torque needed for link 1 to perform a 360° turn at three different speeds $(1.5, 15, 150 degree/s)$, while link 2 rotate at double the speed of link 1.

The main quantities describing the system and the results are reported in Table 1.1 and Figure 1.3.

Table 1.1: Double pendulum quantities.

Mass of link 1	$520\,kg$
Mass of link 2	$212\,kg$
Inertia along z axis of link 1	$4.86\,kgm^2$
Inertia along z axis of link 2	$1.07\,kgm^2$
$\overline{OO'}$: distance between joints	$1.0\,m$
$\overline{OC_1}$: distance between joint 1 and the center of mass of link 1	$0.5\,m$
$\overline{OC_1}$: distance between joint 2 and the center of mass of link 2	$0.375\,m$

Obviously the torques due to gravity $g(q)$ is constant in all three cases, as it does not depend on the speed and acceleration of the system, but only on the position of the centers of mass of the links; and it is approximately equal to the torques of Figure 1.3a, as at low speed the torques due to the accelerations are negligible (Figure 1.3b).

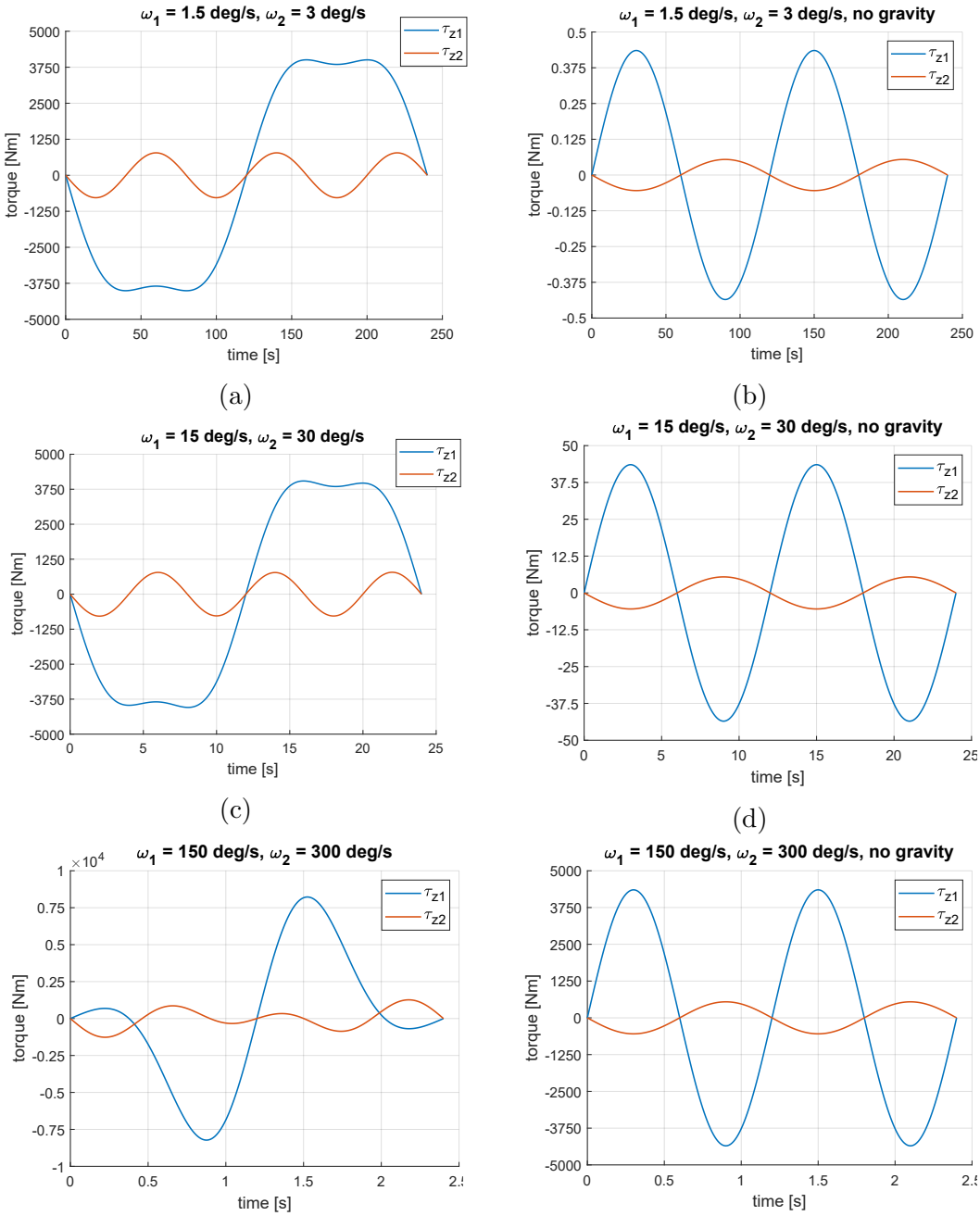


Figure 1.3: Double pendulum simulation's results.

Figure 1.3 shows how the torque due to the acceleration is more and more significant as the angular velocities of the two links increase. Therefore a gravity compensation mechanism is particularly convenient in applications that work at low speed, as the torque due to acceleration and deceleration is sensibly smaller than the torque due to gravity. While in the case of big accelerations and decelerations, the torque required to accelerate is comparable to the one due to gravity and the size and power of the actuators can not be substantially reduced as it is with a lower speed.
Nevertheless, weight compensation could be used advantageously as a safety and energy saver device also in high speeds applications, though not as much as in lower speeds applications.

1.2 Different types of gravity compensators

Different types of weight compensation mechanisms are presented and categorized in [3]. Here only energy-free mechanism for complete gravity compensation on revolute joints are presented.

Counterweight

Using an additional mass and the law of the lever it is possible to let the centre of mass coincide with the joint axis, as shown in Figure 1.4. In this way the gravitational potential energy of the system is constant for every position of the robot, as the centre of mass does not move, and the torque due to gravity is null.
In other words, the counterweight works like an accumulator of the potential energy lost and gain by the robot while moving. This technique allow smooth dynamic behaviours, but as the drawback of increasing the total weight, inertia and volume of the system. Consequently it is not suitable in application that requires movements with high accelerations. The use of counterweight is recommended in applications were the base link can not be rigidly fixed to the ground as in Figure 1.5. [4]

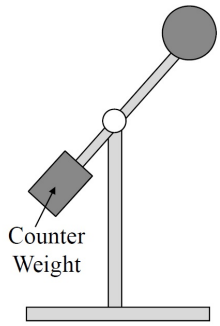


Figure 1.4: Weight compensation using a counterweight.

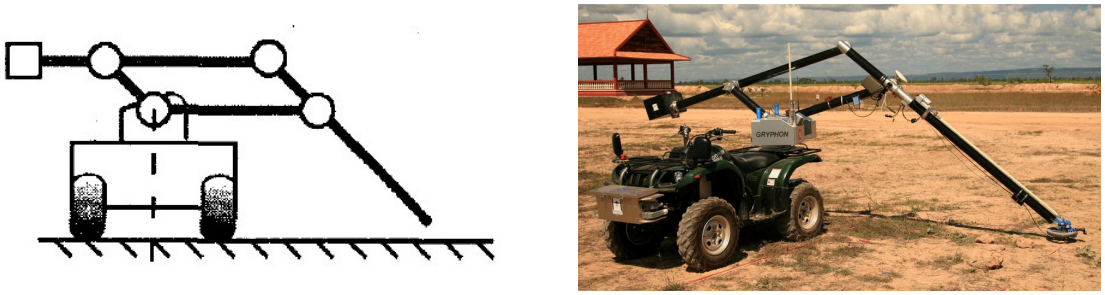


Figure 1.5: pantograph manipulator balanced by a counterweight, mounted on a demininig rover.

Spring

Another method to accumulate the gravitational potential energy is to convert it into elastic potential energy, by substituting the counterweight with a spring. The advantages

of a having a spring are that it adds less mass, inertia and complexity to the system compared to a counterweight. On the other side, it is more difficult to match the nonlinear joint torque induced by gravity and to obtain a smooth dynamic behaviour.
Torsional springs are more rarely compared to axial springs, in view of the fact that it is more difficult to adapt and fine-tune their behaviour to achieve static balance ([5]). On the contrary, the use of axial spring is well documented in literature ([6]).

There are different way to connect the spring to the system to be compensated. The simplest one is to connect the springs directly to the manipulator links as in Figure 1.6a. However, this technique allows perfect compensation only if a zero-free length spring is used, and even in this case, it is difficult to fine-tune the mechanism and obtain a perfect compensation. In addition there is also the risk that the protruded spring may interfere with other parts of the robot.

More elaborated techniques involving cables, additional links, pulleys and cams allow to bypass these two problems. For example, a pulley and a cable can be arranged to simulate a zero-free length spring by storing away the initial length of the spring. Additional links are used to protect the spring in a more compact configuration, but have the disadvantage of adding more weight to the structure.

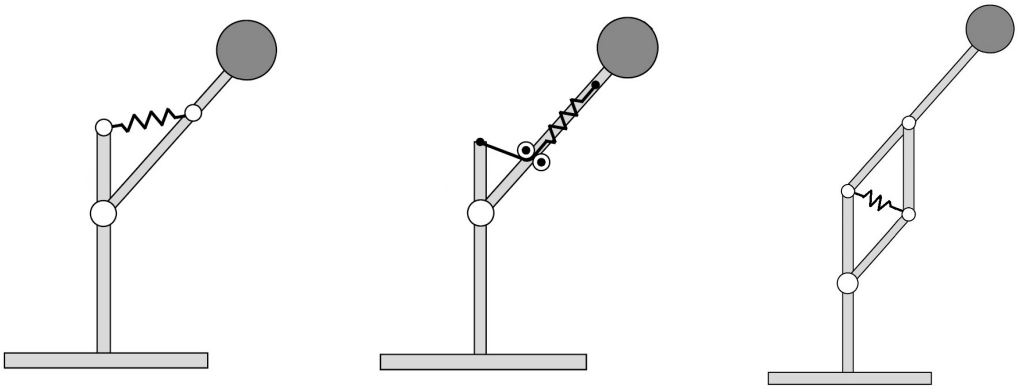


Figure 1.6: Weight compensation mechanisms with springs, cables, additional links and pulleys.

The use of pulleys and cams with noncircular shapes has the benefit to increase the free design parameters of the system, allowing an easier and better optimization of the mechanism of gravity compensation. Figure 1.7 illustrates some of the many examples described in literature. [7, 8, 9, 1]

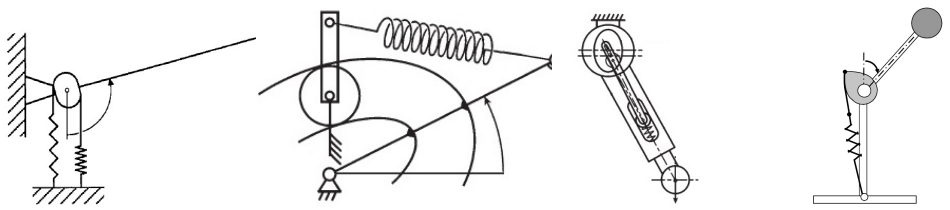
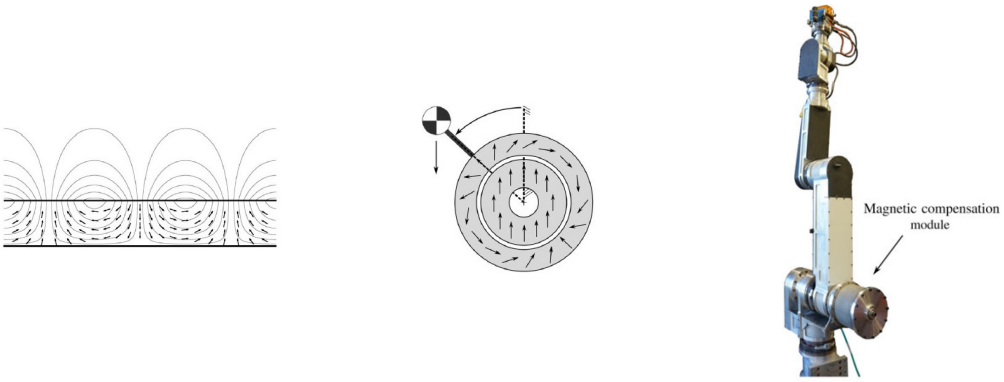


Figure 1.7: Weight compensation mechanisms with springs, cables, cams and pulleys.

Magnets

Recently a series of magnets arranged in a special configuration, called Halbach array, has been used by [10] in a gravity compensator device. It was exploited the fact that two nested Halbach cylinders produce a sinusoidal torque capable of compensating that of a gravitational load over a complete rotation of a revolute joint.
Even if this solution is difficult to fine-tune and it is not possible to obtain a perfect compensation, it has the advantages of being extremely compact, with the cylinders mounted in axis with the joint, and the possibility to be modularize so that several modules can be combined in series to obtain increasing torques.



(a) (b) (c)

Figure 1.8: (a) Linear Halbach array. (b) Nested Halbach cylinders to achieve gravity compensation. (c) Serial manipulator retrofitted with a magnetic compensation module.

Vertical axial joints

By designing a robot with vertical axial joint, parallel to the gravity field, all the weight of the robot is supported by the structure itself and there is no torque due to gravity on the joints. Robots like the SCARA (Figure 1.9) have been build with this notion. However this technique limits considerably the design of the robot and has a worse characteristics in terms of supported payload, accuracy, dexterity as the range of motion increases.

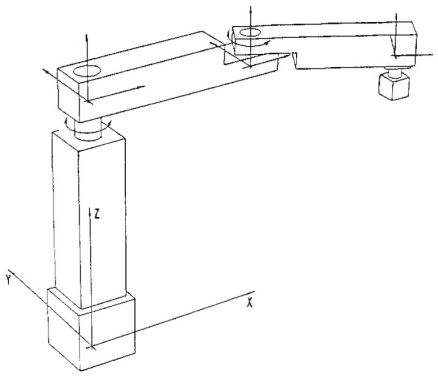


Figure 1.9: SCARA robot.

1.3 FlegX

To deepen the effects of implementing a gravity compensator into a robotic systems, the Italian Institute of Technology (IIT) started a research project connected on the mechanism proposed by [1] (Figure 1.7d). The aim of the project was to build a test bench to apply the gravity compensation mechanism to a simple case: an inverted pendulum with one degree of freedom. In parallel, a model to perform a multibody dynamic simulation of the system was developed using the software MSC.ADAMS®, in order to compare the real and simulated systems. In a second phase of the project, the possibility of implementing such a mechanism into a robotic flexible leg, called FlegX ([11]), will be taken into consideration.
FlegX is a 3 DoF mechanism with two actuated revolute joints, that act as hip and knee of the robotic leg and a translational joint; the upper leg is build with a rigid link, while the lower leg is made out of a flexible component. This robot was build with the intention of examining the advantages of using flexible components into industrial robots.
The reason to apply a gravity compensator to FlegX is to decrease the size of the electric actuators in order to obtain better jumping performances lowering the total mass of the robot.
In this phase the multibody system will be exploited to run some preliminary simulations to better assess the functioning of the gravity compensator in a more complex system.

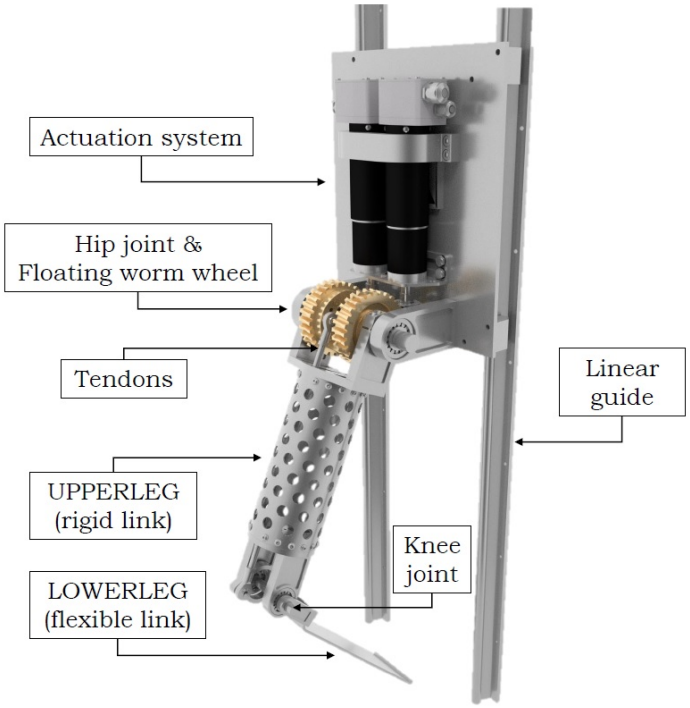


Figure 1.10: FlegX.

The gravity compensator made with a spring and a noncircular pulley, proposed by [1] (Figure 1.11), was preferred among all the possible designs because of its simplicity, compactness and, above all, because it presents the possibility to compensate an arbitrary torque profile on the joint, not just the trigonometric torque profile due to the weight of the structure. This characteristic allows interesting application of this particular gravity

compensator; e.g. it can be adjusted to optimize the specific task of the robot, or it can be used into exoskeleton to generate particular exponential torque typical of human articulations, as reported in [12, 1].

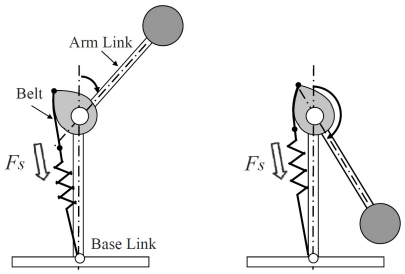
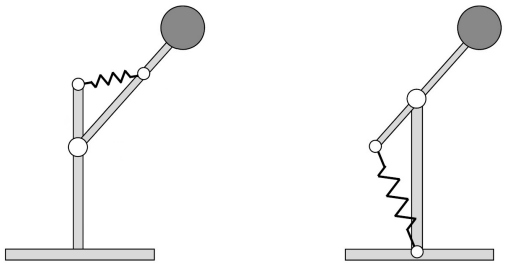


Figure 1.11: Gravity compensation with noncircular pulley and spring.

1.4 Analysis of mechanism with spring connected directly to the link

As a first step in the design of the weight compensation mechanism with the noncircular pulley, the kinematic of the compensator where the spring is directly connected to the manipulator link was analysed. The study of this simpler mechanism had the aim of acquiring a better understanding of the subject. The compensator was applied to a planar 1 DoF inverted pendulum, because it represents an easier system to study and is the same system studied in the further phases of the project.
The two possible configurations illustrated in Figure 1.12 are described by the same kinematic model. While configuration 1.12a is more compact, 1.12b is more convenient in the case of large workspaces and less width robots.



(a) (b)

Figure 1.12: Weight compensation mechanisms with spring connected directly to the link.

Supposing to have an ideal spring, which elongation at rest is null, the equation of the spring is:

$$\begin{cases} F_s = k(x - x_0) \\ x_0 = 0 \end{cases} \Rightarrow F_s = k \cdot x$$

where F_s is the force exerted by the spring, k the spring stiffness, x the elongation and x_0 the elongation of the spring at rest.

The problem is to find, if possible, the value of the spring stiffness k that allows a perfect compensation of the torque due to gravity on the joint. The solution can be obtained

through several procedures. Here two methods are reported: firstly using the equilibrium among the forces and then using and energy approach.
Both solutions are developed in reference to Figure 1.13.

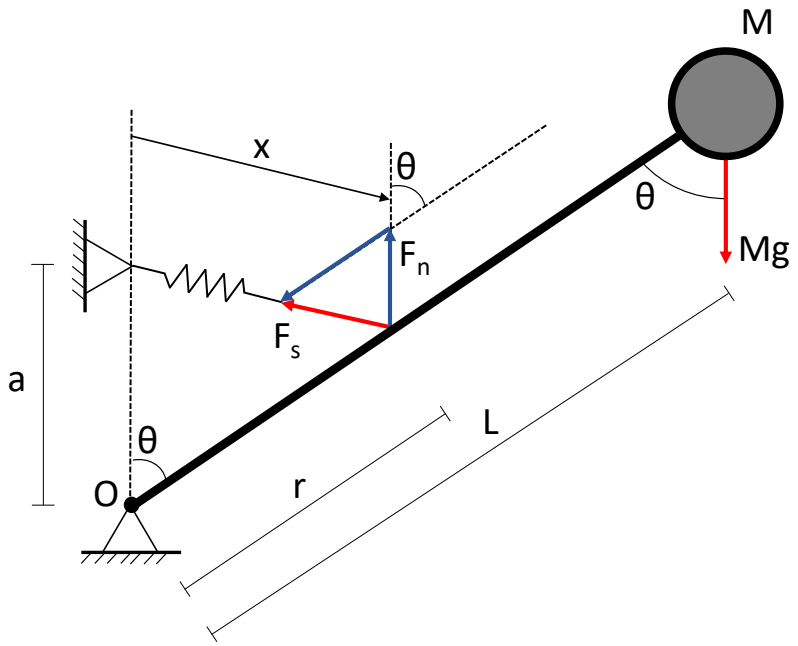


Figure 1.13: Static scheme of the system.

Equilibrium of forces

Force due to mass:

$$F_g = M \cdot g$$

Spring force:

$$F_s = k \cdot x$$

Vertical component of the spring force:

$$F_n = k \cdot a$$

Equilibrium of momentum in O:

$$F_g \cdot L \sin\theta = F_n \cdot r \sin\theta$$

$$MgL \sin\theta = kar \sin\theta$$

$$MgL = kar$$

Spring stiffness that allows perfect gravity compensation:

$$k = \frac{MgL}{ar} \tag{1.3}$$

Constant potential energy

Potential energy of mass M:

$$V_M = MgL\cos\theta$$

Using the law of cosines:

$$x = \sqrt{r^2 + a^2 - 2ar\cos(\theta)}$$

Potential energy of spring k:

$$V_s = \frac{1}{2}kx^2 = \frac{1}{2}k\left(r^2 + a^2\right) - kar\cos\theta$$

Total potential energy of the system:

$$V_{tot} = V_M + V_s = (MgL - kar)\cos\theta + \frac{1}{2}k\left(r^2 + a^2\right)$$

Independence of $Vtot$ from θ is ensured it the related derivative is null for every value θ

$$\frac{\partial V_{tot}}{\partial\theta} = -(MgL - kar)\sin\theta = 0, \forall\theta$$

which is obtained for every value of θ if:

$$k = \frac{MgL}{ar}$$

that is the same expression of Equation 1.3. Therefore perfect gravity compensation is possible by using an ideal spring with stiffness $k = \frac{MgL}{ar}$.

If the case with a real spring is considered, the equation of the spring force is:

$$\begin{cases} F_s = k(x - x_0) \\ x_0 \neq 0 \end{cases}$$

Using the *constant potential energy* solution, it is possible to demonstrate that perfect gravity compensation can not be achieved.
Potential energy of real spring:

$$V_s' = \frac{1}{2}k\left(x - x_0\right)^2 = \frac{1}{2}k\left(r^2 + a^2 - 2ar\cos\theta - 2x_0\sqrt{r^2 + a^2 - 2ar\cos\theta} + x_0^2\right) 4$$

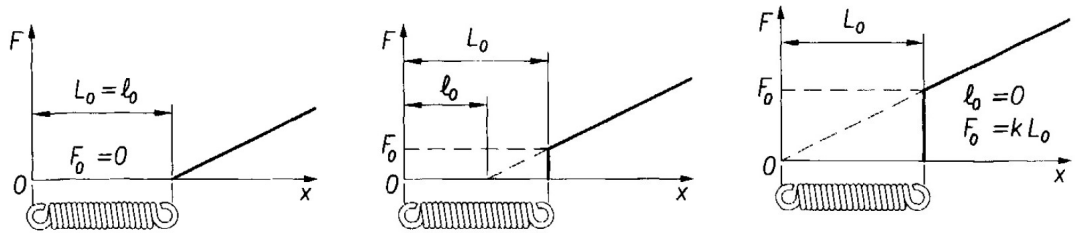
Therefore the total potential energy is:

$$V_{tot}' = (MgL - kar)\cos\theta + \frac{1}{2}k\left(r^2 + a^2 + x_0^2\right) - 2kx_0\sqrt{r^2 + a^2 - 2ar\cos\theta}$$

Independence of V'_{tot} from θ if:

$$\frac{\partial V'_{tot}}{\partial \theta} = -\left(MgL - kar\right)\sin\theta - 2kx_0 \cdot \frac{1}{2}\left(r^2 + a^2 - 2ar\cos\theta\right)^{-\frac{1}{2}} \cdot \left(-2ar\left(-\sin\theta\right)\right) =$$
$$= \left(kar - MgL - \frac{2kx_0 ar}{\sqrt{r^2 + a^2 - 2ar\cos\theta}}\right)\sin\theta = 0, \forall\theta \quad (1.4)$$

Equation 1.4 cannot be verified for every θ and $x_0 \neq 0$, which means that perfect gravity compensation on a revolute joint can not be achieved using a real spring.
A possible alternative solution is the use of pre-loaded spring, tuned to have zero-free length (Figure 1.14); but this method carries two main drawbacks: it cannot be applied if a force smaller than the pre-load of the spring F_0 is required, and this type of spring are difficult to find as off-the -shelf products.



(a) spring with zero initial tension.

(b) spring with some initial tension.

(c) spring with zero-free-length due to an initial tension $F_0 = kx_0$.

Figure 1.14: Initial tension in extension springs.

Alternatively, the initial lenth x_0 of a standard spring can be stored in a different position through pulleys and cables, as illustrated in 1.15.
In these way perfect weight compensation is theoretically possible but it is difficult to realise in practise: once that the geometry of the system is established (M, L, a, r), the quality of the compensation depend solely on the accuracy of the value of the spring stiffness s. Differently, as it will be shown in the following chapter, using the compensator with spring and noncircular pulley, the profile of the pulley can be adjusted to value of the spring, providing an easier possibility of fine-tuning the gravity compensator.

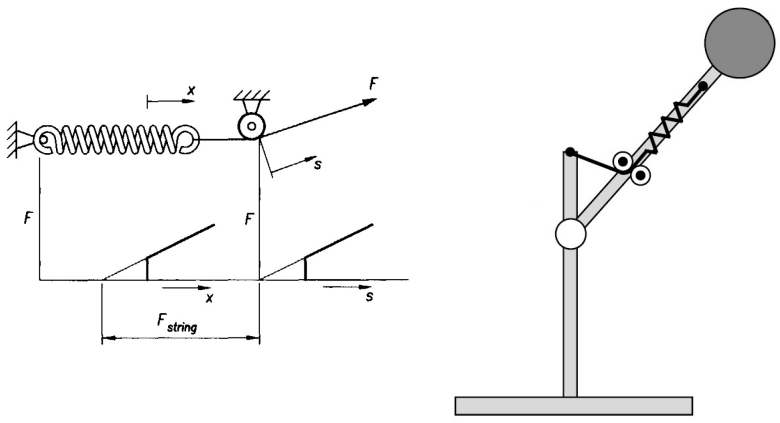


Figure 1.15: Alternatives to zero-free length springs.

Chapter 2

Weight Compensation Mechanism with a Non-Circular Pulley and a Spring

This configuration consists in a linear spring connected to a belt which is laid down on a noncircular pulley. By controlling the elongation of the spring and the variation of the radius of the noncircular pulley, which means controlling the spring force and moment arm, it is possible to control the torque generated on the revolute joint.

In the first part of the current chapter the working principle of the mechanism is analysed in order to obtain the equations that describe the noncircular profile of the pulley. Next, some practical limitations to the shape of the pulley are described and a new procedure via the solution of a convex optimization problem is presented. This procedure allows to include the constraints into the generation of the profile
Afterwards, three topics presented by [1] are reported: a way to take into account some geometrical modifications with respect to the ideal configuration of the mechanism, a procedure to implement an antagonistic springs design to generate a bidirectional torque, and a numerical method to evaluate the torque that can be compensated by a given noncircular pulley.
Finally some examples are reported to clarify the themes introduced in the chapter.

2.1 Working principle

The system represented in Figure 2.1 is used to develop the methodology to synthesis the noncircular pulley profile.
The mechanism is composed by two links, connected by a revolute joint in point O. The noncircular pulley is fixed to the grounded link. A linear extension spring is connected to the mobile link in point R, which will be referred as *insertion point*, and to the grounded link via a cable that wraps around the noncircular pulley. The distance between the axis of revolute joint O and the insertion point R is called *insertion length* and measures L.

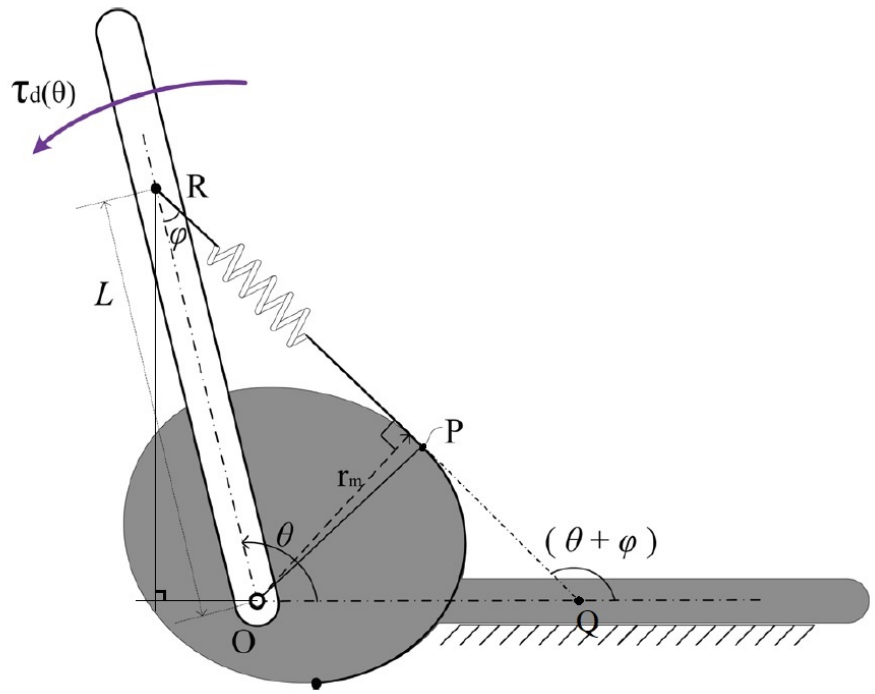


Figure 2.1: Schematic diagram of the system with a spring and a noncircular pulley.

At a joint angle θ, the line of action of the spring passes through point R, the tangency point on the pulley P and point Q on the grounded link. The torque on the joint is determined by the product of the spring force and the moment arm $r_m(\theta)$. If R is fixed, r_m depends solely on the pulley shape.
The goal is to design the profile of the noncircular pulley so that it generates the desired torque profile $\tau_d(\theta)$, given the insertion length L and the spring stiffness k.
In this first phase, it is assumed that the cable has infinite stiffness and negligible radius, the spring is linear, there is no sliding between cable and pulley, there are no friction losses in the system and there is no pulley to redirect the cable in R. Some of these issues will be addressed later on.

There are different procedures to synthesize the pulley profile: [13] proposes a numerical method where each value of $\tau_d(\theta)$ is associated to a point of the pulley profile; [14] and [1] have both developed an analytical method. However the latter was preferred and hereafter described because it takes into account the possibility to consider some practical implementations in the mechanism: the thickness of the cable and the introduction of a circular pulley at the insertion point to redirect the cable.

2.1.1 Moment arm

The first step to synthesize the pulley profile is to find a relation between the desired torque $\tau_d(\theta)$ and the moment arm $r_m(\theta)$, including the effects of the spring elongation . This is done starting from the principle of virtual work: in static equilibrium, the virtual work done by the spring is equal to the one done by the external torque:

$$\tau_d(\theta) \cdot d\theta = F_s(u) \cdot du \tag{2.1}$$

where $F_s(u)$ is the spring force and $u(\theta)$ is the total extended length of the spring.

$$\tau_d(\theta) = F_s(u) \cdot \frac{du}{d\theta} = ku(\theta) r_m(\theta)$$

$$r_m(\theta) = \frac{\tau_d(\theta)}{ku(\theta)} \tag{2.2}$$

The work done by the $\tau_d(\theta)$ from θ_0 to θ is equal to the one done by $F_s(u)$ from u_0 to u. The total work, which takes into account the effects of the cable wrapping around the pulley and the spring elongating, is computed integrating 2.1:

$$\int_{\theta_0}^{\theta} \tau_d(\alpha)\, d\alpha = \int_{u_0}^{u} F_s(\beta)\, d\beta = \frac{1}{2}k\left(u^2 - u_0^2\right)$$

$$u(\theta) = \sqrt{\frac{2}{k}\int_{\theta_0}^{\theta} \tau_d(\alpha)\, d\alpha + u_0^2} \tag{2.3}$$

Equation 2.3 relates the elongation of the spring $u(\theta)$ to $\tau_d(\theta)$. The relation with the moment arm $r_m(\theta)$ is obtained substituting in 2.2:

$$r_m(\theta) = \frac{\tau_d(\theta)}{ku(\theta)} = \frac{\tau_d(\theta)}{k\sqrt{\frac{2}{k}\int_{\theta_0}^{\theta} \tau_d(\alpha)\, d\alpha + u_0^2}} = \frac{\tau_d(\theta)}{\sqrt{2k\int_{\theta_0}^{\theta} \tau_d(\alpha)\, d\alpha + k^2 u_0^2}} \tag{2.4}$$

2.1.2 Pulley shape synthesis

Once the moment arm $r_m(\theta)$ is known for every θ, the pulley profile can be determined. This is accomplished using infinitesimal calculus: the equation of the line $l(\theta)$ passing though R and P is found; then the interception between $l(\theta)$ and $l(\theta + \delta\theta)$ is evaluated, since the interception points coincides with the profile of the noncircular pulley for $\delta\theta$ approaching zero.

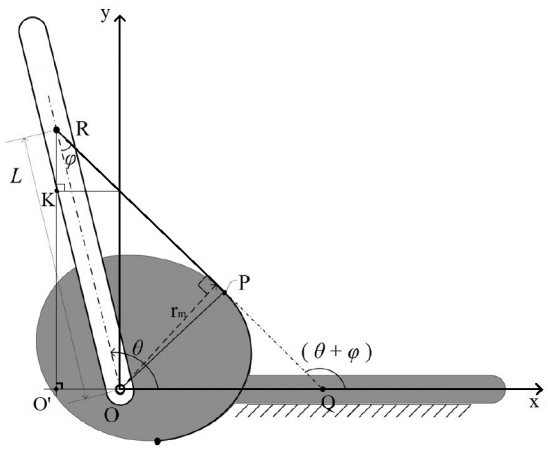


Figure 2.2: Schematic diagram of the system with coordinates system.

Accordingly, the slope of $l(\theta)$ is found:

$$\frac{\overline{RQ}\sin(\theta+\phi)}{\overline{RQ}\cos(\theta+\phi)} = \tan(\theta+\phi)$$

where:

$$\phi = \widehat{ORP}$$

The equation of $l(\theta)$ can be written as:

$$l(\theta) : y = \tan(\theta+\phi)\, x + \overline{O'K}$$

where, in reference to Figure 2.2, $\overline{O'K}$ expressed in terms of θ and ϕ is:

$$\overline{O'K} = \overline{O'R} - \overline{KR} = L\sin\theta - L\cos\theta \cdot \tan(\theta+\phi)$$

Angle ϕ can be written as funcition of θ:

$$\sin\phi = \frac{r_m(\theta)}{L} \Rightarrow \phi = \sin^{-1}\left(\frac{r_m(\theta)}{L}\right)$$

Therefore, the equation of line $l(\theta)$ is function of geometric parameters and θ:

$$l(\theta) : y = \tan\left(\theta + \sin^{-1}\left(\frac{r_m(\theta)}{L}\right)\right) x + L\left[\sin\theta - \cos\theta \cdot \tan\left(\theta + \sin^{-1}\left(\frac{r_m(\theta)}{L}\right)\right)\right] \tag{2.5}$$

$S(\theta)$ and $Y(\theta)$ are introduced to simplify Equation 2.5: $S(\theta)$ is the slope of $l(\theta)$ and $Y(\theta)$ is the y-intercept of the line.

$$\begin{aligned} S(\theta) &= \tan\left(\theta + \sin^{-1}\left(\tfrac{r_m(\theta)}{L}\right)\right) \\ Y(\theta) &= L\left[\sin\theta - \cos\theta \cdot \tan\left(\theta + \sin^{-1}\left(\tfrac{r_m(\theta)}{L}\right)\right)\right] \end{aligned}$$

$$l(\theta) : y = S(\theta)\cdot x + Y(\theta) \tag{2.6}$$

Next, line a and b are defined, which are respectively the lines $\overline{RP}$ for angles θ and and $(\theta+\delta\theta)$; and the coordinates of their intersection $I : (x_I, y_I)$ is found:

$$\begin{aligned} a : y &= S(\theta)\cdot x + Y(\theta) \\ b : y &= S(\theta+\delta\theta)\cdot x + Y(\theta) \end{aligned}$$

$$[S(\theta+\delta\theta) - S(\theta)]\, x_I + Y(\theta+\delta\theta) - Y(\theta) = 0$$

$$\begin{cases} x_I = -\frac{Y(\theta+\delta\theta)-Y(\theta)}{[S(\theta+\delta\theta)-S(\theta)]} \\ y_I = S(\theta)\, x_I + Y(\theta) \end{cases} \tag{2.7}$$

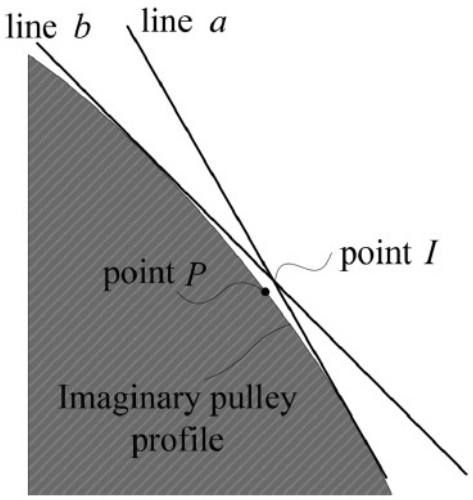


Figure 2.3: Schematic diagram of line a, b and points I, P.

As the the infinitesimal increment $\delta\theta$ decreases, the intersection between line a and b approaches point P on the profile of the pulley. As previously seen, P is the tangency point of the cable on the noncircular pulley for a generic angle θ.
Thus it is possible to find the x-coordinate of point P by taking an infinitesimal increment in θ.

$$x_P = \lim_{\delta\theta\to 0} -\frac{Y(\theta+\delta\theta) - Y(\theta)}{[S(\theta+\delta\theta) - S(\theta)]} = -\frac{dY(\theta)}{dS(\theta)} = -\frac{\frac{dY(\theta)}{d\theta}}{\frac{dS(\theta)}{d\theta}} = -\frac{Y'(\theta)}{S'(\theta)}$$

Finally the y-coordinate is found substituting in Equation 2.7:

$$\begin{cases} x_P(\theta) = -\frac{Y'(\theta)}{S'(\theta)} \\ y_P(\theta) = S(\theta)\, x_P + Y(\theta) \end{cases}$$

Summarising, given the profile of the desired torque to be compensated $\tau_d(\theta)$, the stiffness k and the initial length u_0 of the spring and the insertion length L, the profile of the noncircular pulley is found with the following calculations:

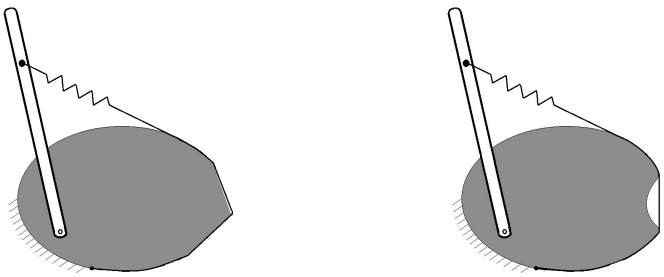
$$\begin{aligned} r_m(\theta) &= \frac{\tau_d(\theta)}{\sqrt{2k\int_{\theta_0}^{\theta}\tau_d(\alpha)d\alpha + k^2u_0^2}} \\ \phi &= \sin^{-1}\left(\frac{r_m(\theta)}{L}\right) \\ S(\theta) &= \tan(\theta+\phi) \\ Y(\theta) &= L[\sin\theta - \cos\theta\cdot\tan(\theta+\phi)] \\ (x_P, y_P) &= \left(-\frac{Y'(\theta)}{S'(\theta)}, Y(\theta) - \frac{Y'(\theta)}{S'(\theta)}S(\theta)\right) \end{aligned} \tag{2.8}$$

2.2 Constraints on Non-Circular Pulley Shape

The noncircular pulley can not have any profile, e.g. in Figure 2.4, and not all combinations of $\tau_d(\theta)$, k, u_0 and L generate a feasible profile. Thus it is necessary to analyse the

constraints to be imposed on the pulley shape and translate them into constraints on the design parameter $(\tau_d(\theta), k, u_0, L)$.
Similar constraints to the one discussed here, had already been discussed in a previous study [15], where noncircular cams are used in band mechanisms to satisfy a given non-circular input-output motion relationship. Given the similar nature of the mechanism, those constraints have been adjusted by [1] to suit the noncircular pulley of the gravity compensator.

First of all, the function of the moment arm must be single valued, continuous and differentiable in all the domain, because the pulley profile must be univocally defined, continuous and without sharpen points, caused by a sudden change of derivative for each angle θ. A sharpen point has the double detriments of damaging the cable and not allowing the cable to perfectly unwrap itself on the pulley (Figure 2.4a).
Also, the moment arm $r_m(\theta)$ must be smaller than the insertion length L, otherwise the mobile link would hit the noncircular pulley and there would be no space to insert the spring.
Lastly, the radius of the pulley contour must not tend to infinite and the shape must be convex. Both of these conditions constraint the curvature of the profile, which must not tend towards 0 to avoid having an infinite radius, and it must not change sign or it would cause interferences between the spring line RP and other sections of the pulley (Figure 2.4b).



(a) (b)

Figure 2.4: Examples of infeasible pulley profile.

Resuming, the constraints on the pulley profile for every $\theta \in [\theta_i, \theta_f]$ are:

1. $r_m(\theta)$ single valued.
2. $r_m(\theta)$ continuous.
3. $r_m(\theta)$ differentiable.
4. $r_m(\theta) < L$.
5. Curvature does not tend to 0.
6. Curvature does not change sign.

where $[\theta_i, \theta_f]$ is the range of angles where the $\tau_d(\theta)$ has to be compensated.

Unluckily, due to the complexity of the relations among the design parameters and the final profile, enclosed in Equations 2.8, it has not been found a way to translate all the aforementioned constraints on the pulley profile into constraint purely on the design parameters $(\tau_d(\theta), k, u_0, L)$.
A solution has been found only for the first three constraints on $r_n(\theta)$: since the expression of r_m is:

$$r_m(\theta) = \frac{\tau_d(\theta)}{\sqrt{2k\int_{\theta_0}^{\theta} \tau_d(\alpha)\, d\alpha + k^2 u_0^2}}$$

$r_m(\theta)$ is single valued, continuous and differentiable if $\tau_d(\theta)$ is single valued, continuous and differentiable.

For fourth contraint $(r_m(\theta) < L)$, the simplest expression found is:

$$\tau_d(\theta) \leq L\sqrt{2k\int_{\theta_0}^{\theta} \tau_d(\alpha)\, d\alpha + k^2 u_0^2}$$

which is not helpful in simplifying the selections of the design parameters.

The same problem appears in the two constraints regarding the curvature of the profile. The expression of the curvature κ of a curve is the derivative of the tangent T with respect to the arc length s, which is equivalent to the ratio between the derivative of the tangent with respect to θ over the inverse of the derivative of the arc lenght with respect to θ:

$$\kappa = \frac{dT(\theta)}{ds(\theta)} = \frac{d(T\theta)}{d\theta}\frac{d\theta}{ds(\theta)} = \frac{T'(\theta)}{s'(\theta)}$$

Therefore imposing:

$$\kappa \neq 0$$

is equivalent to

$$T'(\theta) \neq 0 \tag{2.9}$$

Applying Equation 2.9 to the current case:

$$\frac{d(T\theta)}{d\theta} = \frac{dS(\theta)}{d\theta} \neq 0$$

$$\frac{dS(\theta)}{d\theta} = \frac{d(\tan(\theta+\phi))}{d\theta} = \frac{1}{\cos^2(\theta+\phi)} \cdot \left(1 + \frac{d\phi(\theta)}{d\theta}\right)$$

which implies:

$$1 + \phi' \neq 0 \tag{2.10}$$

Any development of Equation 2.10 has shown to be immune to further analyses.

The second constraint on the curvature of the pulley was not solve directly for the same reason. However if the $r_m(\theta)$ is continuous and 5 is respected, that is that the curvature is not equal to 0 for every θ, it is implied that the curvature can not change sign and 6 is respected.

In conclusion, experience in working with Equations 2.8 and the constraints on the pulley profile has shown that the best way to select the design parameters $(\tau_d(\theta), k, u_0, L)$ is to have a single-valued, continuous, differentiable profile of the desired torque to be compensated τ_d and consequently select the remaining parameters by trial and error.

2.3 Convex optimization approach

An attempt to improve the process of synthesising the profile of the pulley was carried out by finding the moment arm r_m as the solution of an optimization problem in convex form, and then using the previous method presented by [1] to compute the profile. The main advantages of this approach is the possibility of including the constraints on the pulley profile directly as constraints of the optimization problem.

Convex Optimization Problem

First of all, the target function of the optimization problem is determined from the equation of r_m (Equation 2.4).
Angle θ is discretized over the range of motion of the joint, from the initial angle θ_i to the final θ_f:

$$\bar{\theta} \in \mathbb{R}^n, \ \bar{\theta} = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_{n-1} \end{bmatrix}^\intercal$$

Then Equation 2.4 is rewritten and then applied to the discretized angle $\bar{\theta}$:

$$r_m(\theta) \cdot \sqrt{2K \int_{\theta_0}^{\theta} \tau_d(\alpha) d\alpha + k^2 u_0^2} = \tau_d(\theta)$$

$$r_m(\theta_i) \cdot \sqrt{2K \int_{\theta_0}^{\theta_i} \tau_d(\alpha) d\alpha + k^2 u_0^2} = \tau_d(\theta_i), \ \forall \theta_i \in \bar{\theta} \tag{2.11}$$

$$\Rightarrow r_m(\bar{\theta}) \in \mathbb{R}^n$$

Then variable $v(\bar{\theta})$ is introduced

$$v(\theta_i) = \sqrt{2K \int_{\theta_0}^{\theta_i} \tau_d(\alpha) d\alpha + K^2 u_0^2)}$$

and it is used to simplify Equation 2.11:

$$r_m(\theta_i) \cdot v(\theta_i) = \tau_d(\theta_i) \tag{2.12}$$

Then Equation 2.12 is rewritten in matrix form:

$$diag(r_m(\bar{\theta})) \cdot v(\bar{\theta}) = \tau_d(\bar{\theta})$$

r_m can be computed solving a Least Square (LS) problem:

$$\underset{r_m}{\text{minimize}} \quad ||diag(r_m(\bar{\theta})) \cdot v(\bar{\theta}) - \tau_d(\bar{\theta})||_2 \tag{2.13}$$

as $v(\bar{\theta})$ and $\tau_d(\bar{\theta})$ are known matrix defined by the design parameters, and $r_m(\bar{\theta})$ is the decision variable to be optimized.
Remarkably, since a LS problem is known to be constant, problem 2.13 has a unique solution than can be found with very efficient algorithms.

Approximation of r_m with a polynomial of degree m

r_m is approximated py a polynomial of degree m to allow an easier implementations of the constraints:

$$r_m(\theta) = \beta_0 + \beta_1\theta + \beta_2\theta^2 + \ldots + \beta_m\theta^m$$

which can be rewritten in matrix form as:

$$r_m(\bar{\theta}) = \begin{bmatrix} 1 & \bar{\theta} & \bar{\theta}^2 & \ldots & \bar{\theta}^m \end{bmatrix} \cdot \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}, \qquad \bar{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}$$

$$r_m(\bar{\theta}) = \begin{bmatrix} 1 & \bar{\theta} & \bar{\theta}^2 & \ldots & \bar{\theta}^m \end{bmatrix} \cdot \bar{\beta} \tag{2.14}$$

The polynomial approximation is integrated into the optimization problem by adding Equation 2.14 as an affine equality constraint, which preserve the convexity of the problem, and extending the searching variables by including $\bar{\beta}$:

$$\begin{aligned} \underset{r_m,\bar{\beta}}{\text{minimize}} \quad & ||diag(r_m(\bar{\theta})) \cdot v(\bar{\theta}) - \tau_d(\bar{\theta})||_2 \\ \text{subject to:} \quad & r_m(\bar{\theta}) = \begin{bmatrix} 1 & \bar{\theta} & \bar{\theta}^2 & \ldots & \bar{\theta}^m \end{bmatrix} \cdot \bar{\beta} \end{aligned} \tag{2.15}$$

Additional constraints on r_m

The arm moment r_m is constrained to have a positive finite value, smaller than the insertion length L. Therefore it must respect the following relation:

$$0 < r_m < L \tag{2.16}$$

Equation 2.16 can be subdivided into two linear disequality constraints of the LS problem, without depriving the optimization problem from its convexity:

$$\begin{aligned} \underset{r_m,\bar{\beta}}{\text{minimize}} \quad & ||diag(r_m(\bar{\theta})) \cdot v(\bar{\theta}) - \tau_d(\bar{\theta})||_2 \\ \text{subject to:} \quad & r_m(\bar{\theta}) = \begin{bmatrix} 1 & \bar{\theta} & \bar{\theta}^2 & \dots & \bar{\theta}^m \end{bmatrix} \cdot \bar{\beta} \\ & r_m(\bar{\theta}) > 0 \\ & ||r_m(\bar{\theta})||_\infty < L \end{aligned} \tag{2.17}$$

Constraint on the convexity of the pulley

As previously proved, the shape of the pulley is imposed as convex in order to avoid interferences between the cable and other parts of the noncircular pulley. The straightforward way to obtain so is to impose a constraint on the curvature of the pulley, so that it does not change sign.
However, a way to impose convexity on the parametric expression $(x_p(\theta), y_p(\theta))$ was not found; not even by deriving a convex relation $(x_p(\theta), y_p(\theta))$ and the moment arm r_m, due to the complexity of their relations expressed by 2.8.
Even if the proper radius of the pulley $\overline{OP}$ and the moment arm r_m are not to be confused with each other (Figure 2.5), it is considered reasonable to aim for a more conservative solution by imposing convexity on the profile of the moment arm r_m, because of their tight relation. Furthermore, a tendency at benefiting from the use of r_m instead of the proper radius of the pulley was observed in the dissertation of the constraints on the pulley profile in [1], as well.

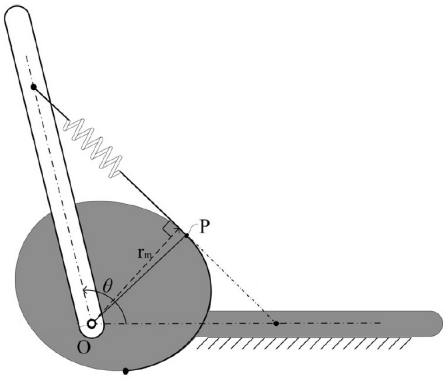


Figure 2.5: Radius $\overline{OP}$ and moment arm r_m.

Thus convexity on the shape is ensured by imposing a positive second derivative of $r_m(\theta)$ with respect to θ:

$$\frac{d^2(r_m(\theta))}{d\theta} \geq 0 \tag{2.18}$$

The polynomial approximation of $r_m(\theta)$ is used to rewrite the constraint as a linear disequality:

$$\frac{d^2(r_m(\theta))}{d\theta} = 2\beta_2 + 6\beta_3\theta + \ldots + m(m-1)\beta_m\theta^{m-2} \geq 0$$

which in matrix form becomes:

$$\begin{bmatrix} 0 & 0 & 2 & 6\bar{\theta} & \dots & m(m-1)\bar{\theta}^{m-2} \end{bmatrix} \cdot \bar{\beta} \geq 0 \tag{2.19}$$

The resulting convex optimization problem is enriched with another linear disequality constraint (Equation 2.19), and therefore retains convexity:

$$\begin{aligned} \underset{r_m,\bar{\beta}}{\text{minimize}} \quad & ||diag(r_m(\bar{\theta})) \cdot v(\bar{\theta}) - \tau_d(\bar{\theta})||_2 \\ \text{subject to:} \quad & r_m(\bar{\theta}) = \begin{bmatrix} 1 & \bar{\theta} & \bar{\theta}^2 & \dots & \bar{\theta}^m \end{bmatrix} \cdot \bar{\beta} \\ & r_m(\bar{\theta}) > 0 \\ & ||r_m(\bar{\theta})||_\infty < L \\ & \begin{bmatrix} 0 & 0 & 2 & 6\bar{\theta} & \dots & m(m-1)\bar{\theta}^{m-2} \end{bmatrix} \cdot \bar{\beta} \geq 0 \end{aligned} \tag{2.20}$$

This final version of the optimization problem includes all the constraints on the shape of the pulley and, as demonstrated, is convex. Hence it can be used to synthesise the profile of the noncircular pulley ensuring to obtain a feasible profile, to the detriment of a perfect compensation of the weight.
In fact, in the cases when the analytical procedure by [1] would produce a profile perfect for weight compensation, yet infeasible to be practically implemented, the convex approach generates a feasible profile, that can not ensure perfect compensation.
In future works, it would be interesting to study whether the feasible but not perfect profile is still convenient to be implemented.
A MATLAB function to synthesise the noncircular profile using the convex approach hare described was reported in Appendix A.4.

2.4 Geometrical corrections

Up this point, it was assumed to have a cable with zero thickness, but in practise the thickness of the cable is never zero and this has the effect of increasing the moment arm with which the force from the spring is applied. In addition, it is generally necessary to reroute the cable through a circular pulley in the insertion point R to have a more convenient arrangement of the mechanism.
[1] suggests how to take into account these two geometrical corrections to achieve a perfect gravity balance.

The thickness of the cable t_c and the radius of the circular pulley r_p change the positions of the contact point R from the insertion point O' to R'. Since the line of action can be assumed to pass through the centre of the cable, $\overline{OS}$ is the difference between $r_m(\theta)$ and $0.5t_c$, and $\overline{O'R'}$ can be obtained from the similarity between triangles OSR' and $O'QR'$. The new slope of the action line l is $\tan(\theta + \phi')$, with ϕ' being the inverse sine of $\overline{O'R'}/(r_m - 0.5t_c)$.
Thus, the corrected equation of line l can be found from $\overline{O'R'}$ and $\tan(\theta + \phi')$, and from it the new profile of the noncircular pulley is computed.

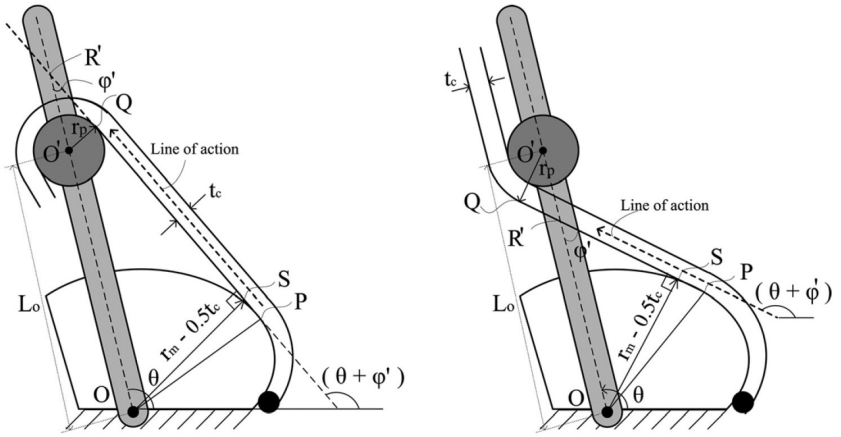


Figure 2.6: Possible configurations of cable routing.

The solution was developed taking into account both possible cable routing configuration (a) and (b), illustrated in Figure 2.6:

Case *a*

$$\begin{aligned} \phi' &= \sin^{-1}\left(\frac{r_m(\theta)-0.5t_c-r_p}{L_O}\right) \\ S(\theta) &= \tan(\theta+\phi') \\ Y(\theta) &= \frac{L_O}{1-r_p/(r_m(\theta)-0.5t_c)}\left[\sin\theta-\cos\theta\cdot\tan(\theta+\phi')\right] \end{aligned} \tag{2.21}$$

Case *b*

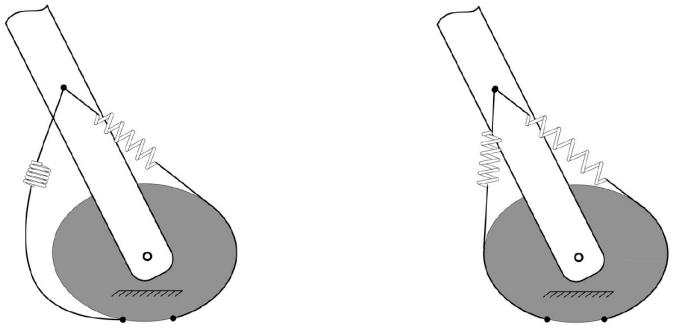
$$\begin{aligned} \phi' &= \sin^{-1}\left(\frac{r_m(\theta)+0.5t_c+r_p}{L_O}\right) \\ S(\theta) &= \tan(\theta+\phi') \\ Y(\theta) &= \frac{(r_m(\theta)-0.5t_c)L_O}{r_m(\theta)+0.5t_c+r_p}\left[\sin\theta-\cos\theta\cdot\tan(\theta+\phi')\right] \end{aligned} \tag{2.22}$$

The expressions of $r_m(\theta)$ and (x_P, y_P) remains unchanged from 2.8.

2.5 Antagonist springs design

In many applications, the gravity compensator is needed to generate a bidirectional torque so that it can be applied to revolute joints that move clockwise and counter-clockwise from a neutral position. One way to achieve this is to use a spring and noncircular pulley compensator on each side of the neutral position. However, in this arrangement the spring must be pretensioned in order to avoid slack in the cable, that could cause the cable to slip from the pulley or to get caught by another moving part (Figure 2.7).

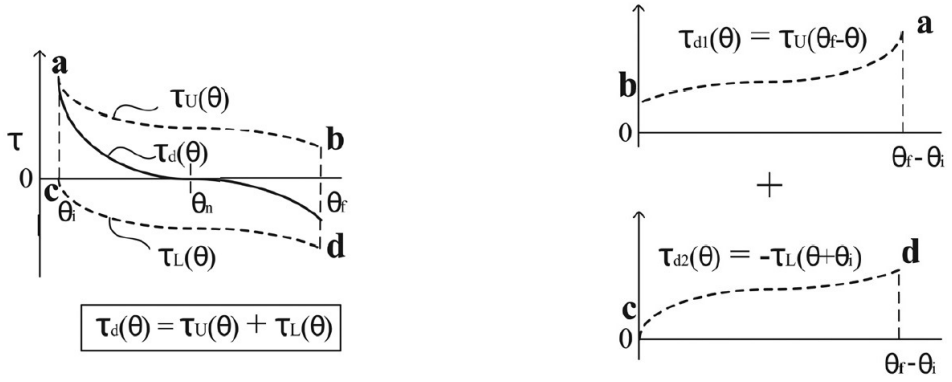
In order to adapt the analytical procedure previously described to this double configuration with pretensioned springs, the desired torque profile $\tau_d(\theta)$ generated by the mechanism is subdivided into two subtorque profiles. Each subtorque is then used to synthesise the profile of one of the two pulley.
The desired torque $\tau_d(\theta)$ over the range of motion from θ_i to θ_f is divided into a positive definite function $\tau_U(\theta)$ and a negative definite function $\tau_D(\theta)$ (Figure 2.8a), whose sum



(a) (b)

Figure 2.7: Antagonistic spring configuration.

equals $\tau(\theta)$. These two subfunctions are then transformed into symmetrical functions ($\tau_{d1}(\theta)$, $\tau_{d2}(\theta)$ in Figure 2.8b) so that their difference equals $\tau_d(\theta)$, since the 2 mechanism have antagonistic action.



(a) (b)

Figure 2.8: Subtorques.

One possible example of the subdivision of $\tau(\theta)$ is the following:

$$\tau_U(\theta) = \gamma\tau_d(\theta) + d_{off} \tag{2.23}$$

$$\tau_L(\theta) = (1-\gamma)\,\tau_d(\theta) - d_{off} \tag{2.24}$$

where γ is a constant value between 0 and 1, and d_{off} is an offset to ensure a positive and negative definite function torque and an initial tension in the spring to avoid slack in the cable. If γ is set equal to 0.5, the two pulley are symmetric.
To avoid slack, the offset d_{off} must respect the following constraint:

$$d_{off} \geq \max\left(\left|\gamma\tau_d\left(\theta\right)_{\min}\right|, \left|(1-\gamma)\,\tau_d\left(\theta\right)_{\max}\right|\right), \quad \forall\theta \in (\theta_i, \theta_f) \tag{2.25}$$

The operation to transform the function of Equation 2.23 and 2.24 into symmetrical forms is the following:

$$\tau_{d1}(\theta) = \tau_U\left(\theta_f - \theta\right) \tag{2.26}$$

$$\tau_{d2}(\theta) = -\tau_L\left(\theta + \theta_i\right), \tag{2.27}$$

with both equations are valid for $\forall\theta \in (0, \theta_f - \theta_i)$.

2.6 Evaluation of a pulley profile without taking into account geometric corrections

The possibility to evaluate the torque produced by a given combination of design parameters and pulley profile is useful as it allows to counter-check an already build configuration of the gravity compensator or to study the torque produced by pulley with a profile easier to manufacture. Furthermore, the process to develop the algorithm hereinafter helps in deepening the understanding of the mechanism.
In [1] only the case with no cable thickness and routing pulley is considered.

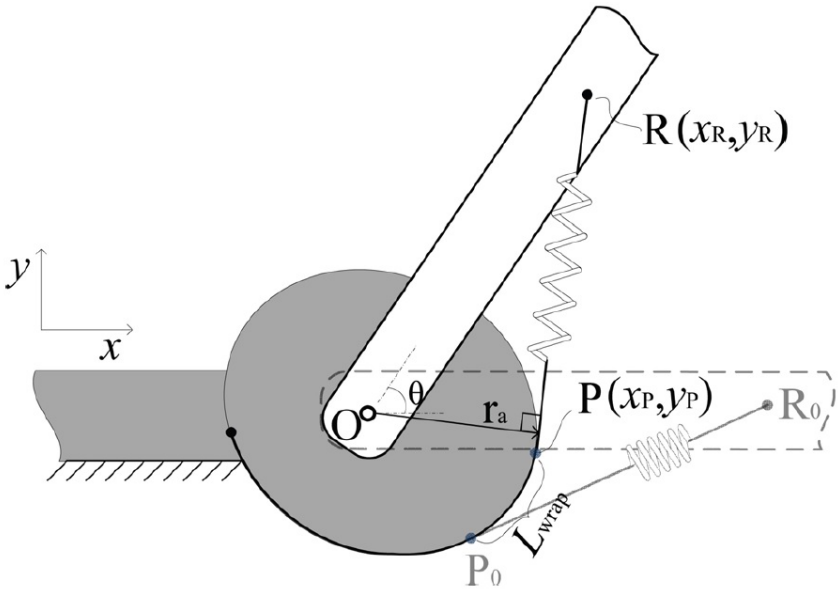


Figure 2.9: Evaluation of the torque generated by the noncircular pulley-spring mechanism.

The solution was developed setting a reference frame on the axis of the revolute joint O. The torque $\tau_{act}(\theta)$ applied in O is the resultant of the product of the spring force $F_s(\theta)$ and the moment arm $r_a(\theta)$:

$$\tau_{act}(\theta) = F_s(\theta) r_a(\theta) = k u(\theta) r_a(\theta) \tag{2.28}$$

The total extended length of the spring $u(\theta)$ is the sum of the length u_0 to apply the initial pretension and the difference between the current length of $L_{wrap} + \overline{RP}$ the initial cable-spring length $\overline{R_0 P_0}$.

$$u(\theta) = L_{wrap}(\theta) + \overline{RP}(\theta) - \overline{R_0 P_0} + u_0 \tag{2.29}$$

L_{wrap} is the arc length of the curve from the initial tangency point P_0 to the current tangency point P. Its computation requires numerical integration, as it is expressed in nonclosed form:

$$L_{wrap} = \int_{\theta_0}^{\theta} \sqrt{\left(\frac{dx_p(\alpha)}{d\alpha}\right)^2 + \left(\frac{dy_p(\alpha)}{d\alpha}\right)^2} d\alpha \tag{2.30}$$

The moment arm $r_a(\theta)$ is the distance from O to the line $l(\theta)$ passing from P and R:

$$r_a(\theta) = \frac{|x_R(\theta)y_P(\theta) - y_R(\theta)x_P(\theta)|}{\sqrt{(x_R(\theta) - x_R(\theta))^2 + (y_R(\theta) - y_R(\theta))^2}} \tag{2.31}$$

Since there is no routing pulley in the insertion point, the coordinates of R during the motion of the link is circular with centre O and radius L, the insertion length:

$$\begin{cases} x_R(\theta) = L\cos(\theta) \\ y_R(\theta) = L\sin(\theta) \end{cases} \tag{2.32}$$

So Equations 2.30, 2.32, 2.29 are used to compute the total extended length of the spring $u_0(\theta)$; Equations 2.32, 2.31 are used to compute the moment arm $r_a(\theta)$; lastly Equation 2.24 generates the produced torque $\tau_{act}(\theta)$.

2.7 Evaluation of a pulley profile taking into account geometric corrections

Since [1] presents a numerical evaluation only for the ideal case, the author of this thesis developed an algorithm to evaluate the profile of a noncircular pulley taking into account the thickness of the cable and the a routing pulley in the insertion point and it is hereunder presented.

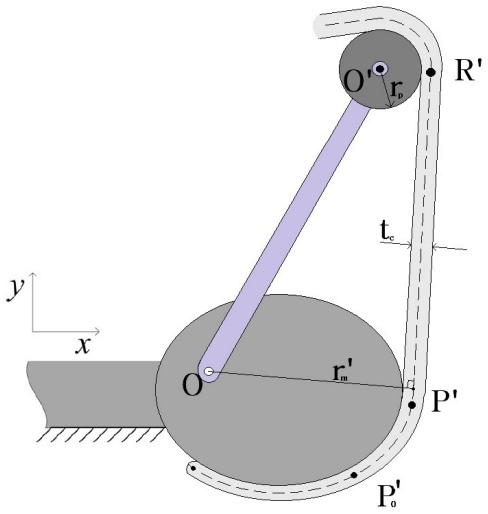
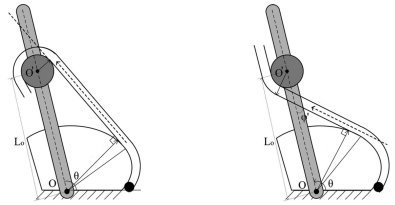


Figure 2.10: Evaluation of the torque generated by the noncircular pulley-spring mechanism.

As shown in Figure 2.10, the contact points between the cables and the two pulley P' and R' are not the same of the previous case. The goal is to find the the expressions of the coordinates of P', R' in order to use them in the solution of [1].
The thickness of the cable t_c is taken into account simply enlarging the radius of the routing pulley and of the noncircular pulley by $0.5t_c$, since the line of action is assumed

to be in the middle of the cable. A MATLAB function to enlarge the profile of the pulley was reported in A.3.
Therefore P' is found moving P of $0.5t_c$ in the direction normal to the profile of the noncircular pulley. R' is computed as the intersection between the enlarged profile c_2 of the circular routing pulley, with centre in the insertion point O' and radius $r_p + 0.5t_c$, and the circumference c_1 with centre in P' and radius $\overline{P'O'}$. The difficult part of the procedure is to implement a robust algorithm able to select the correct intersection between c_1 and c_2, depending on which routing configuration is used ((a) or (b) in Figure 2.11). For example, in Figure 2.12 the intersection corresponding to configuration (a).



(a) (b)

Figure 2.11: Possible configurations of cable routing.

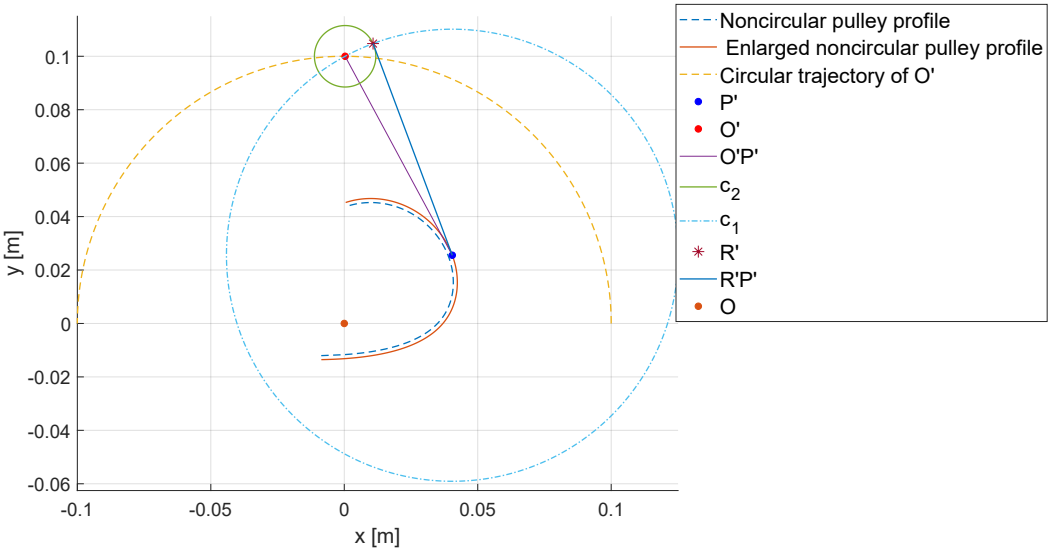


Figure 2.12: Evaluation of the torque generated by the noncircular pulley-spring mechanism.

2.8 Examples

Three examples were analysed in order to test the the algorithms explained in the previous sections: an inverted pendulum, an inverted pendulum with some geometrical corrections to be implemented, a 3 DoF articulate robot.
The software MATLAB® was used to realise several scripts and functions that automate all the algorithms. The scripts of the first two examples were reported in Appendix A.1 and A.2.

2.8.1 Example 1: gravity compensator using the antagonist spring design for an inverted pendulum

The aim of the example reported herein was to synthesise the profile of two noncircular pulleys to compensate the gravity torque of an inverted pendulum. The design parameters are reported in Table 2.1.

Table 2.1: Example 1's parameters.

Gravity	$g = 9.807\, m/s^2$
Mass of pendulum	$M = 3\, kg$
Pendulum length	$l = 1\, m$
Spring stiffness	$k_s = 5 \cdot 10^3\, N/m$
Spring elongation at rest	$x_0 = 22\, mm$
Insertion length	$L = 0.1\, m$
Angular range	$\theta \in [0°, 180°]$
Parameters for the antagonistic subtorques	$\gamma = 0.5,\ d_{off} = 0.55\tau_d(0)$

Since the angular range stretched from 0° to 180°, the gravity compensator had to produce a bidirectional torque. Hence the torque to be compensated $\tau_d(\theta)$ had to be subdivided into two antagonistic subtorques $\tau_{d1}(\theta)$ and $\tau_{d2}(\theta)$, as explained in Section 2.5.

$$\tau_d(\theta) = Mgl\cos\theta$$

$$\begin{aligned}\tau_U(\theta) &= 0.5\tau_d(\theta) + 0.55\tau_d(0) = Mgl\,(0.5\cos\theta + 0.55)\\ \tau_L(\theta) &= (1 - 0.5)\,\tau_d(\theta) - 0.55\tau_d(0) = Mgl\,(0.5\cos\theta - 0.55)\end{aligned}$$

$$\begin{aligned}\tau_{d1}(\theta) &= \tau_U\,(180° - \theta) = -Mgl\,(0.5\cos\theta - 0.55)\\ \tau_{d2}(\theta) &= -\tau_L\,(\theta + 0°) = -Mgl\,(0.5\cos\theta + 0.55)\end{aligned}$$

Figure 2.13 reports the profiles of the subtorques. Since γ was set equal to 0.5, the two subtorques $\tau_{d1}(\theta)$, $\tau_{d2}(\theta)$ that were used to synthesise the noncircular pulley were identical. Consequently the two synthesised noncircular pulleys of the mechanism were symmetrical.

After the subdivision of $\tau_d(\theta)$, the procedure reported in Section 2.1.2 was used to compute the profile of $r_m(\theta)$ and $(x_P(\theta), y_P(\theta))$, whose plots are reported in Figure 2.14.

Finally, the pulley profile was evaluated using the method explained in Section 2.7, in order to check the torque that it would generate.

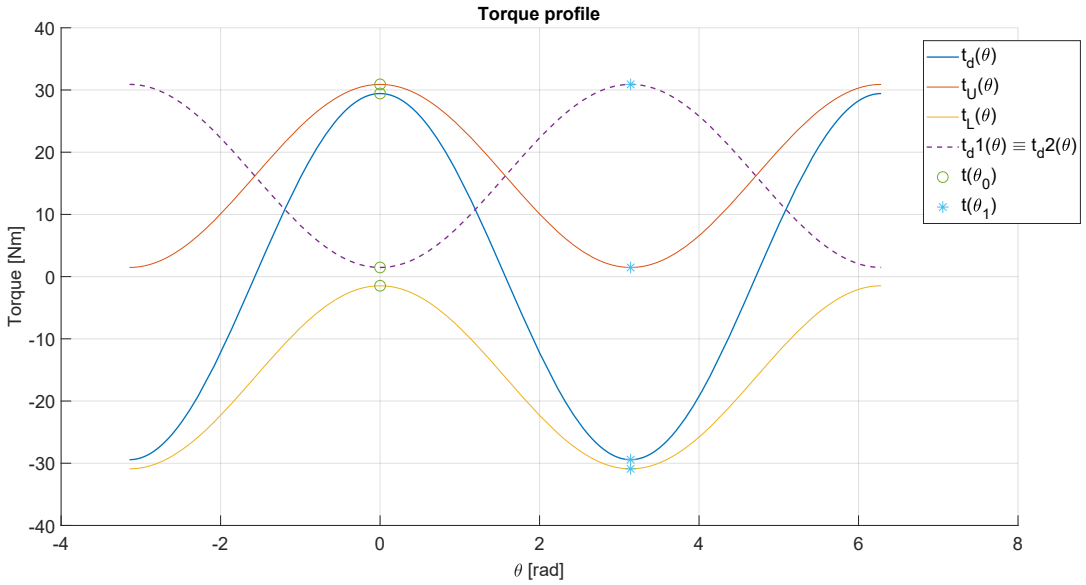
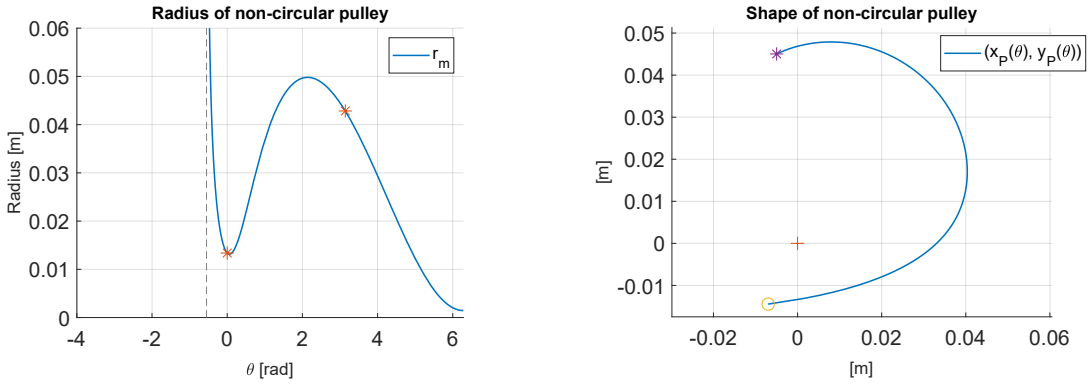


Figure 2.13: Torque subdivision.



(a) (b)

Figure 2.14: $r_m(\theta)$ and $(x_P(\theta), y_P(\theta))$.

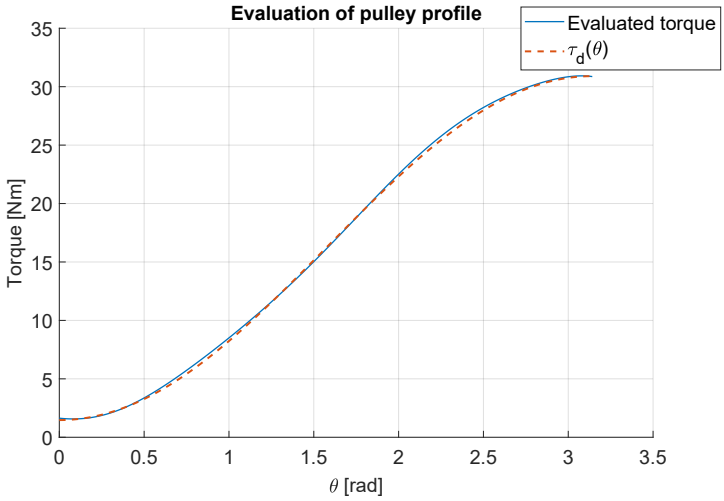


Figure 2.15: Evaluation of torque produced by the pulley profile.

2.8.2 Example 2: Noncircular pulley profile with geometrical corrections

The same inverted pendulum was considered but with the addition of a routing pulley at the insertion point and a cable of not negligible thickness, that was routed to the pulley in configuration (a).

Table 2.2: Geometrical corrections.

Routing pulley's radius	$r_p = 10\,mm$
Cable thickness	$t_c = 7\,mm$

Figure 2.16 reports the two different profiles generated either taking into account the above-mentioned modifications applying the method described in Section 2.4, or not considering them.

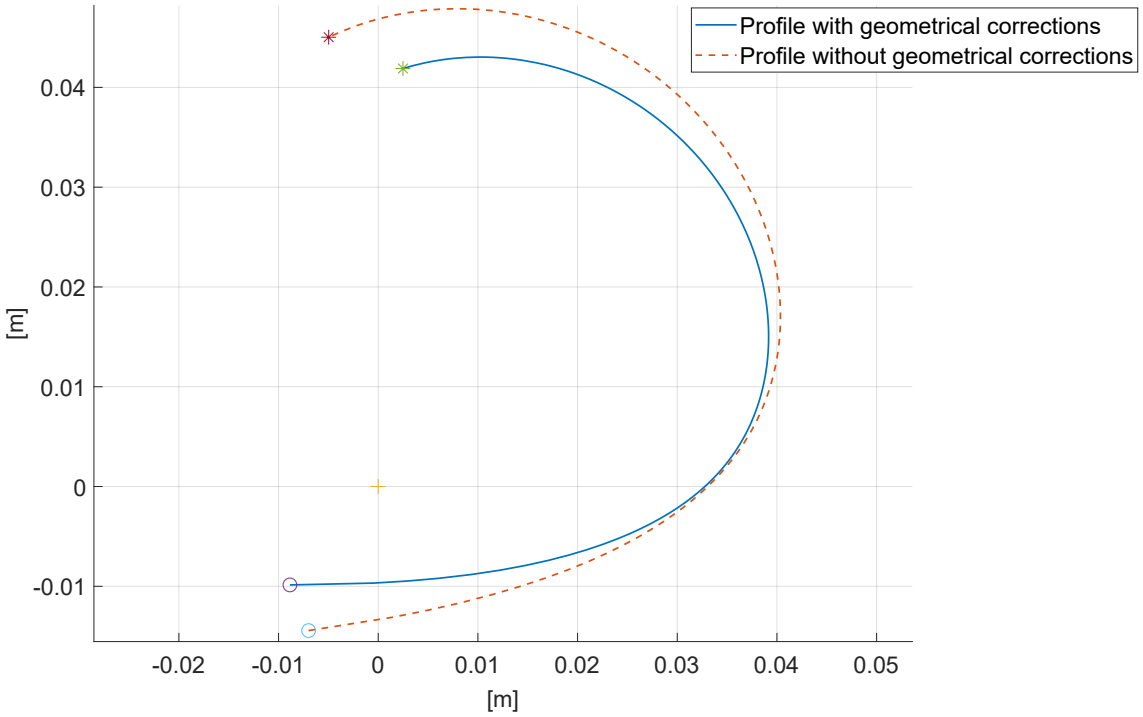
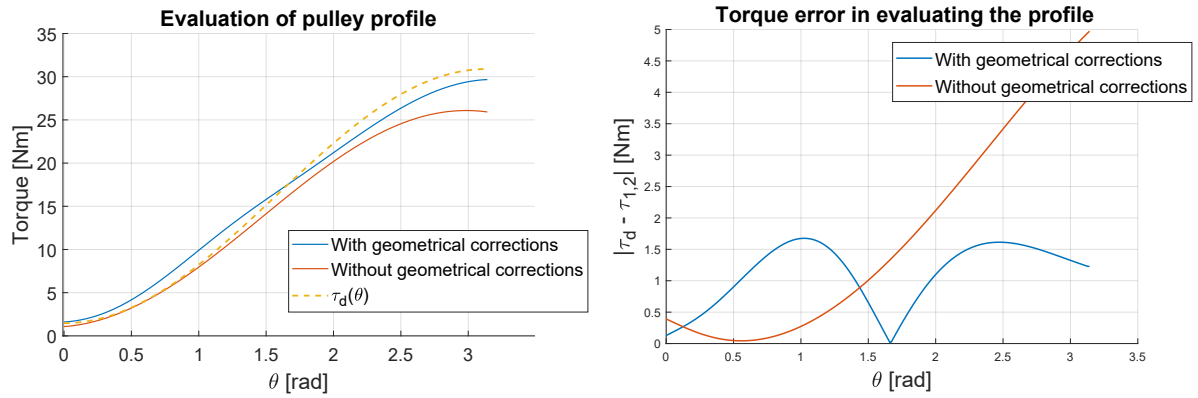


Figure 2.16: Pulley profile taken into considerations the geometry corrections.

This time the pulley profile was evaluated through two different methods: firstly with the method by [1], secondly with the method proposed in Section 2.6 that takes into account the geometry corrections. The results, reported in Figure 2.17, showed how neither of the two methods was able to give a perfect result as in the previous example (Figure 2.15). However, a better result was obtained with the method that takes into account the geometry corrections, suggesting that the general concepts of that method were correct, yet its implementation should be improved.

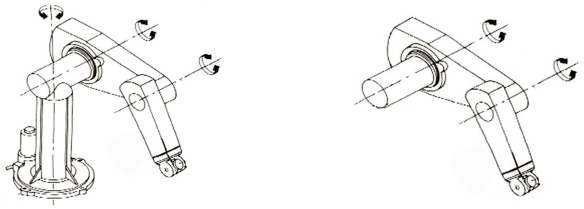


(a) (b)

Figure 2.17: (a) Evaluation of the torque generated by the pulley profile with two different methods. (b) Error between the evaluated and the actual generated torque.

2.8.3 Example 3: 3 DoF articulated robot

Lastly, the case of a 3 DoF articulated robot was examined (Figure 2.18a). Since the first revolute joint axis is perpendicular to the ground, it does not affect the gravitational potential energy of the robot. Hence the case could be simplified to the study of a 2 DoF system, where the two remaining revolute joints have the axis parallel to each other and to the ground, as in Figure 2.18b.



(a) (b)

Figure 2.18: Articulated robot simplified from 3 to 3 DoF.

The key difficulty was to generate on both of the joints a coupled torque, which is function of both the joints' angles. In the following, the solution proposed by [1] is reported. The problem was solved introducing an articulated parallelogram to the structure. In this way a torque that is function of the sum of the 2 joint angle could be generated by the gravity compensator and transferred to the distal joint.

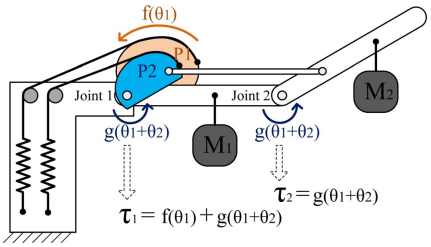


Figure 2.19: Gravity compensation of a 2 DoF robot with articulated parallelogram.

That method was used to generate the profiles of the noncircular pulleys for a simplified model of the 2 DoF articulated parallelogram, depicted in Figure 2.20. The masses of the joints, links and end effector were taken into consideration. The masses of the links could be considered as concentrated in the centres of mass in the middle points of each link, as this simplification does not affect the potential energy of the system.
All the significant parameters of the model were summarised in Table 2.3.

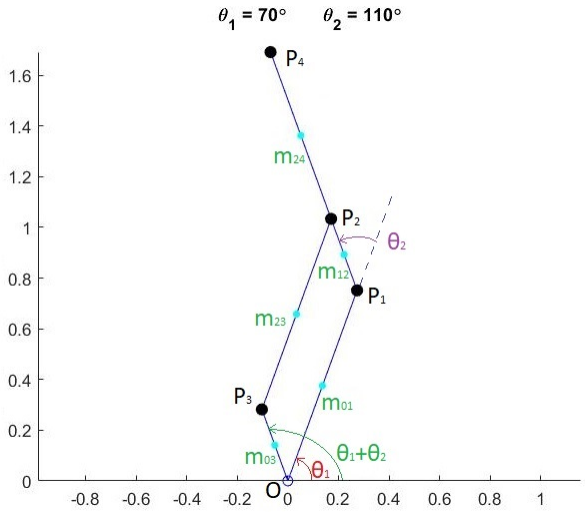


Figure 2.20: Schematic representation of the articulated robot.

Table 2.3: Example 3's parameters.

Length of links l_{01}, l_{23}	$l_1 = 0.8\,m$
Length of links l_{12}, l_{30}	$l_1 = 0.3\,m$
Length of link l_{24}	$l_1 = 0.7\,m$
Mass of P_1	$m_{P_1} = 1\,kg$
Mass of P_2	$m_{P_1} = 1\,kg$
Mass of P_3	$m_{P_1} = 1\,kg$
Mass of P_4	$m_{P_1} = 1\,kg$
Mass per unit length l_{01}, l_{12}, l_{24}	$\rho_1 = 3.0\,kg/m$
Mass per unit length l_{23}, l_{30}	$\rho_2 = 1.5\,kg/m$
Spring stiffness	$k_s = 5 \cdot 10^3\,N/m$
Spring elongation at rest	$x_0 = 22\,mm$
Insertion length	$L = 0.1\,m$
Angular range	$\theta \in [0°, 180°]$
Parameters for the antagonistic subtorques	$\gamma = 0.5,\ d_{off} = 0.55\tau_d(0)$

The first step was the computation of the torques for synthesising the profile of the noncircular pulley.

This was accomplished by finding the contribution of each mass to the torque caused by gravity in O. Thus the expressions of the coordinates of the masses of the joints (Equation 2.33), the values of the masses of the links (Equation 2.34) and their coordinates (Equation 2.35) were computed:

$$\begin{aligned}
P_1 &: \begin{pmatrix} l_1 \cos\theta_1 \\ l_1 \sin\theta_1 \end{pmatrix} \\
P_2 &: \begin{pmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{pmatrix} \\
P_3 &: \begin{pmatrix} l_2 \cos(\theta_1 + \theta_2) \\ l_2 \sin(\theta_1 + \theta_2) \end{pmatrix} \\
P_4 &: \begin{pmatrix} l_1 \cos\theta_1 + (l_2 + l_3) \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + (l_2 + l_3) \sin(\theta_1 + \theta_2) \end{pmatrix}
\end{aligned} \tag{2.33}$$

$$\begin{aligned}
m_{01} &= \rho_1 l_1 = 2.40\,kg \\
m_{12} &= \rho_1 l_2 = 0.90\,kg \\
m_{23} &= \rho_2 l_1 = 1.20\,kg \\
m_{30} &= \rho_2 l_2 = 0.45\,kg \\
m_{24} &= \rho_1 l_3 = 2.10\,kg
\end{aligned} \tag{2.34}$$

$$\begin{aligned}
m_{01} &: \begin{pmatrix} \frac{l_1}{2} \cos\theta_1 \\ \frac{l_1}{2} \sin\theta_1 \end{pmatrix} \\
m_{12} &: \begin{pmatrix} l_1 \cos\theta_1 + \frac{l_2}{2} \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + \frac{l_2}{2} \sin(\theta_1 + \theta_2) \end{pmatrix} \\
m_{23} &: \begin{pmatrix} \frac{l_1}{2} \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ \frac{l_1}{2} \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{pmatrix} \\
m_{30} &: \begin{pmatrix} \frac{l_2}{2} \cos(\theta_1 + \theta_2) \\ \frac{l_2}{2} \sin(\theta_1 + \theta_2) \end{pmatrix} \\
m_{24} &: \begin{pmatrix} l_1 \cos\theta_1 + \left(l_2 + \frac{l_3}{2}\right) \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + \left(l_2 + \frac{l_3}{2}\right) \sin(\theta_1 + \theta_2) \end{pmatrix}
\end{aligned} \tag{2.35}$$

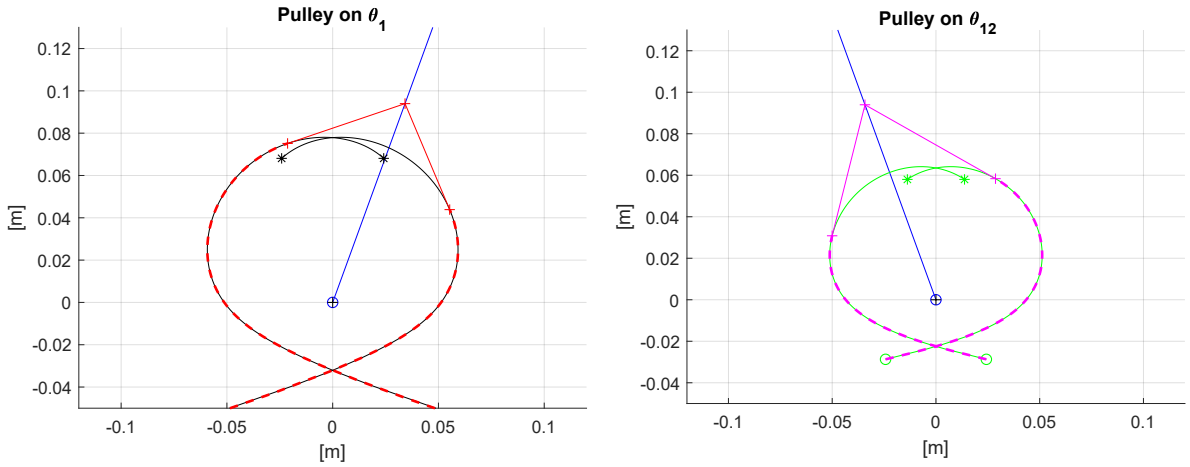
The torque in O due to gravity could be subdivided in the torque τ_1 function of θ_1 and the torque τ_{12} function of to $(\theta_1 + \theta_2)$

$$\tau_1(\theta_1) = \left[(m_{P_1} + m_{P_2} + m_{P_4} l_1) + \left(m_{01} \frac{l_1}{2} + m_{12} l_1 + m_{23} \frac{l_1}{2} + m_{24} l_1 \right) \right] g \cos\theta_1$$

$$\begin{aligned}
\tau_{12}(\theta_1 + \theta_2) = \Big\{ & \left[(m_{P_1} + m_{P_2} + m_{P_4} l_1)\, l_2 + m_{P4} l_3 \right] + \\
& + \left[(m_{12} + m_{30}) \frac{l_2}{2} + m_{23} l_2 + m_{24} \left(l_2 + \frac{l_3}{2} \right) \right] \Big\} g \cos(\theta_1 + \theta_2)
\end{aligned}$$

$\tau_1(\theta_1)$ and $\tau_{12}(\theta_1 + \theta_2)$ were then used to synthesise the profile of two noncircular pulleys, as in the previous examples. The insertion point of the pulley for θ_1 would be on link l_{01},

while the insertion point for the other pulley would be on link l_{30}.



(a) (b)

Figure 2.21: Pulley's profiles to compensate the gravity of the articulated robot.

Figure 2.21 reports the pulleys' profiles obtained. The fact that the noncircular pulleys overlapped could be overcome by mounting the pulley on different planes.

Chapter 3

Virtual dynamic model

In order to test the correct working of the spring-noncircular pulley gravity compensator applied to a revolute joint in the antagonistic springs configuration, a test bench was designed. Since the test bench was used to evaluate the performances of the mechanism, it had to be as simple and neat as possible so that the behaviour of each component could be neatly isolated. It was therefore decided to apply the gravity compensator to a 1 DoF inverted pendulum actuated by a an electric motor.
However, before designing and manufacturing the actual prototype of the inverted pendulum and the relative noncircular pulley-spring mechanism to compensate its gravity, it was decided to realise a virtual prototype of the system. This choice was made to benefit from the several advantages deriving from having a dynamic model of the system during the entire process of realisation of the physical prototype.
First of all it provided the possibility to have a preliminary validation of the theoretical concept. Then, while designing the physical prototype, it allowed to compare different possible designs of the mechanism and to check the forces and mechanical stresses in the structure.
Once that the actual prototype was build and assembled, the physical and virtual prototypes were compared in order to fine-tune the parameters of the virtual model. This process, called *validation*, had the goal to make the virtual model perform more realistically and to deepen the understanding of the static and dynamic behaviour of the system.
At the end of the study, the virtual model could be applied to more complex systems to investigate its interaction with other multibody systems. Furthermore, it could be used to investigate new, more complicated configurations, without the need of manufacturing a new prototype.
To realise the virtual prototype, the multibody dynamics simulation software MSC.ADAMS® (Automated Dynamic Analysis of Mechanical Systems) was used.

When developing the model of a mechanical system, [16] recommends to proceed following the *crawl-walk-run* approach: starting from the easiest possible model, the system is gradually improved with the aim of reaching a model as close to reality as possible. At every step of the process, the system is tested via a dynamic simulation. The following variation to the system is then pondered inspecting the results of the simulation and comparing them with the expected behaviour of the real world system.
There are many indicators that have to be looked at in order to understand whether the

simulation of a model is successful or not, and they vary from case to case. In general, if the computation of the simulation happens smoothly and without crashes, the movements of the parts in the system correspond to the expected ones, and the graphs of the various forces, velocity, accelerations do not have unexpected shapes or values, it is likely that the model is correct; but it really does depend on the specific case that is studied each time.
Anyway, verifying the model via the designated command *Model Verify* before every computation is a good practice and can help foreseeing potential problems.

Driven by the *crawl-walk-run* design approach, it was preferred to report the step-by-step process followed to obtain the final virtual system of the spring-noncircular pulley gravity compensator, instead of simply describing the final definitive model. In this way, the process hereinafter reported could help in giving an idea on how to approach the realisation of an intricate system.

3.1 Early considerations

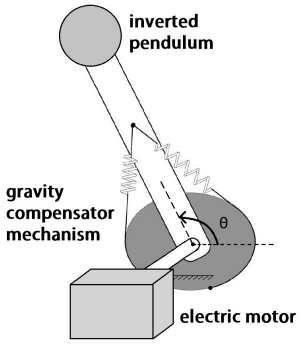


Figure 3.1: Schematic representation of the system.

The essential elements that would form the test bench are the electric motor, the inverted pendulum and the gravity compensator mechanism.
To simplify the system, initially the electric motor could be modelled by either a torque or a *motion* imposed on the joint. In the Adams environment, by *imposing a certain motion* to a joint is meant to force the link of the joint to perform that given movement; e.g. in the pendulum-electric motor case, the rotation of a certain angle at a given angular speed.
The pendulum was simply modelled with a *marker*, i.e. a reference frame, with associated the inertial properties of the pendulum and connected to the *ground* via a revolute joint. However, a *geometry* was associated to the *marker* in order to have a better visual reference of the position and orientation of the pendulum.
The most critical parts to be modelled were the two antagonist spring-cable-noncircular pulley mechanisms: while the springs could be modelled by the respective element present in Adams and a CAD model of the noncircular pulley could be imported into the Adams model, there was not a predefined method to model the cable and its contact with the noncircular pulley.

Hence the main focus of the current chapter is the investigation of an effective way to model the spring-noncircular pulley mechanism.

Initially, the gravity compensator was designed with the noncircular pulley fixed to the ground and the insertion point on the pendulum, as in the examples made by [1]. The cable was attached to the pendulum, so that there was no need of a routing pulley at the insertion point.
Moreover, the two antagonist spring-pulley mechanisms to generate the bidirectional torque were designed to be symmetrical. In this way, once that one of the two spring-pulley-cable had been modelled, the other one could be created simply as the reflection of the first one. This was achieved by setting parameter γ of Equations 2.23, 2.24 equal to 0.5. In addition, angle θ of the pendulum was set equal to 90° in the initial configuration of the simulation, so that not only the two mechanisms were identical, but also the initial arrangements of the cables were symmetrical.
It was thus possible to focus in developing only one of the two spring-pulley mechanisms.

The earliest models were developed imposing to the system the arbitrary design parameters used in the examples of the previous section: the pendulum was set to weight 3 kg, with the mass concentrated in a point distant 1 m from the joint axis; the insertion length was equal to 0.1 m. The profile of the noncircular pulley was synthesised from this parameters and considering an angular range of θ from 0° to 180°, the pretension parameter d_{off} equal to $0.55\tau_d(0)$, the spring stiffness k equal to $5000\,N/m$ and its length at rest x_0 equal to $0.022\,mm$.
At this stage, no geometrical correction was considered to synthesise the pulley profile. The design parameters would be corrected to the one of the physical prototype only in the final definitive model, so that the results could be compared. In this first phase, the exemplifying values of the system were sufficient, as the focus was on the overall functioning of the system, not on the attained values.
The design parameters for the noncircular pulley are reported in Table 3.1.
The profile of the noncircular obtained from the above-mentioned parameters was used to create a CAD model of the noncircular pulley to be imported into the Adams model.

Table 3.1: Design parameters to synthesise the noncircular pulley.

Function of the torque to be compensated	$\tau_d(\theta) = Mgl\sin\theta$
Mass of the pendulum	$M = 3\,kg$
Length of the pendulum	$l = 1\,m$
Gravity	$g = 9.807\,m/s^2$
Angular range	$\theta \in [0°, 180°]$
Insertion length	$L = 0.1\,m$
Parameters for the antagonistic subtorques	$\gamma = 0.5,\ d_{off} = 0.55\tau_d(0)$
Spring's parameters	$k_s = 5000\,N/m,\ x_0 = 0.022\,m$

3.2 Model using Adams' machinery suite

The first attempt at modelling the cable was made exploiting the Adams' simulation suite called *Machinery*. This suite is used to simulate mechanical drive systems like gear, belt, chain, cams and cable transmission system, bearings and motor. In particular the package to simulate a cable and pulleys transmission system was used.
The idea was to discretise the noncircular pulley placing several circular pulley on the profile of the noncircular one, as illustrated in Figure 3.2. The benefits of this method were that the models of the cable and the circular pulleys could be automatically created through the machinery wizard. This implies that the contacts automatically included in the models have optimal values for the type of transmission needed.
In this case the noncircular pulley was used as visual reference: the locations of the circular pulleys were defined using the equation of the profile of the noncircular pulleys for different values of θ.
In order to maintain the same profile, the noncircular pulley had to be shrunk normally to the profile of the dimension of the radius of the circular pulleys. In this way, if the circular pulley were inserted on the profile of the shrunk pulley, they would enter in contact with the cable at the correct distance from the centre of the noncircular pulley, keeping unaltered the moment arm $r_m(\theta)$. The same algorithm used to enlarge the profile of the pulley in section 2.7 was used to shrink the profile, the only difference being the sign of the parameter that determines the entity of the enlargement since in this case the aim is to shrink and not to enlarge.

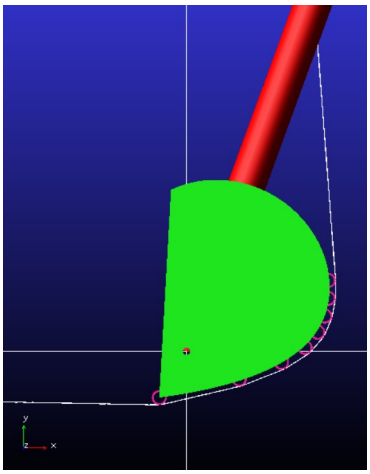


Figure 3.2: Noncircular pulley and cable using the machinery suite.

Unfortunately, despite the advantages of this method, the use of the machinery suite had revealed to be unsuited to model the cable-noncircular pulley system. However powerful and easy to use, the machinery suite is intended for more conventional cases and it is not customisable to cases as specific and unconventional as this one.
The main issue was that the circular pulleys must touch the cable in the initial configuration of the system otherwise they are deactivated during the simulation, letting the cable pass through them.
A possible solution for one of the two antagonist compensators would be to set the initial configuration of the system at the angle at which the cable is engaged with all the circular pulley, i.e. the cable is wrapped on all the noncircular pulley. For example for

$\theta = 180°$, the pulley in Figure 3.2 is fully engaged. The problem is that when one of the two antagonist noncircular pulleys is fully engaged, the specular one is not engaged at all and its circular pulleys would be deactivated during the simulation.
For this reason the solution via the machinery suite was excluded.

3.3 Model using Adams/View Command Language

In alternative to using the machinery suite, it was decided to model the cable from scratch and let it interact with the imported CAD of the noncircular pulley adding a *contact* between the parts to prevent interpenetration.

If the two antagonist mechanisms and the pendulum axis are arranged in the same plane, the exchanges of forces springs-cables and cables-noncircular pulleys all occur in the same plane, e.g. the xy-plane in Figure 3.3. This characteristic of the system was used to simplify the model of the cable.

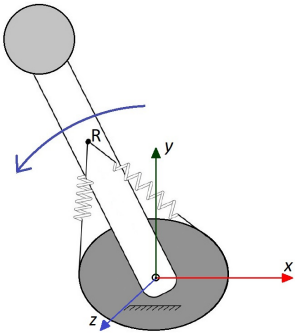
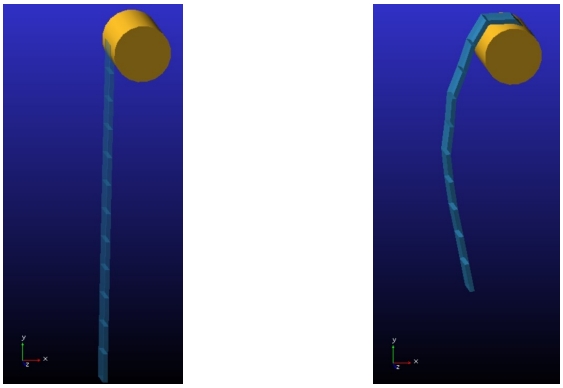


Figure 3.3: Location of the reference frame in the model.

The cable was modelled as a series of small elements, connected in line one to the other. Since all the actions of the model occur in one plane, the cable elements were connected one to the other through revolute joints with the axis perpendicular to the xy-plane, instead of spherical joints. In this way the DoFs of the cable elements were decreased and with it the complexity of the simulation.
Another design choice, made thanks to the fact that the mechanisms are contained in one plane, was to set the section of the cable rectangular instead of circular. This was accomplished by making the cable elements out of parallelepiped, instead of cylinders. In this way the contact area between the single cable element and the noncircular pulley is increased from being punctual to a segment. This choice was made in order to avoid small areas of contacts that could be harder to compute by the solver.
Because of the shape, the cable is hereinafter referred to as belt in the next sections.

Due to the great number of parts that had to be created to model the cable, the plan of building the model entirely through ADAMS graphic interface was not conceivable. Thus it was decided to build the model using the *Adams/View Command Language*, which allows to write macros that can automatise any operation in the Adams environment.

The use of *Adams/View Command Language* is very convenient in our case, because it allows to build elaborated system supported by the benefits of a computer language, e.g. the creation of the cable elements can be iterated using `for` cycles.
Not only the belt, but the entire system was modelled trough macros in order to obtain a more refined model, easier to debug and modify. Moreover, this had the merit to strengthen the knowledge in the new language.



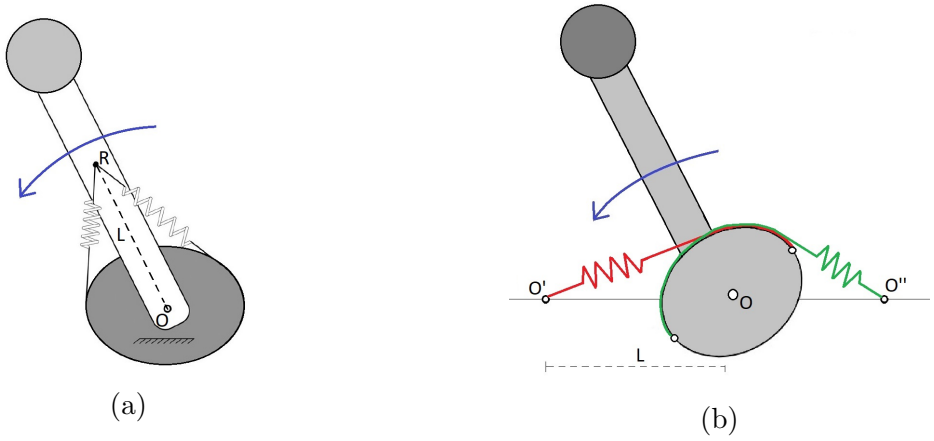
(a) (b)

Figure 3.4: Preliminary belt model.

As a first preliminary attempt, a sample of a belt made out of a dozen elements was created (Figure 3.4a). The belt elements were connected to each other through revolute joints, as previously explained, while the first element of the belt was attached to the cylindrical yellow pulley through a fixed joint. Contacts between each belt elements and the cylinder were defined, using the default values for the contact parameters. Next a motion of rotation was imposed to the cylinder and the model was simulated (Figure 3.4b).
The objectives of this simple model were to become familiar with the geometry of the problem and to have a first test on the contact between the belt and a curved surface.
Once that the simulations had been considered satisfactory, the model of the proper belt was undertaken.

3.4 First belt model

At this stage of the project, it was decided to bring some modifications to the mechanism with the aim of facilitating the design of both the virtual and physical prototype.
The first change was to make the noncircular pulley integral to the pendulum and not to the ground (Figure 3.5). This design decision was made to simplify the realisation of the physical prototype, allowing to lock the noncircular pulley on the shaft of the pendulum. This modification caused the insertion point to be moved from the mobile link to the ground so that the relative motion between the insertion point and the noncircular pulley could remain unaltered. While previously the insertion point on the pendulum would rotate with respect to the noncircular pulley that was fix on the ground, in the new arrangement the noncircular pulley attached to the pendulum rotates with respect to the insertion point, which is fixed on the ground.



(a) (b)

Figure 3.5: Model with fixed or mobile noncircular pulley.

The second change that was brought to the design was to add a routing pulley at the insertion points, as illustrated in Figure 3.6. The reason that motivated this decision was to have a virtual model more similar to the physical one. The inclusion of a routing pulley in the physical prototype was justified by the fact that it would allow for a simpler regulation of the pretension of the spring attached to the cable.
The distance from the axis of the pendulum to centre of the circular pulley must be equal to the insertion length.

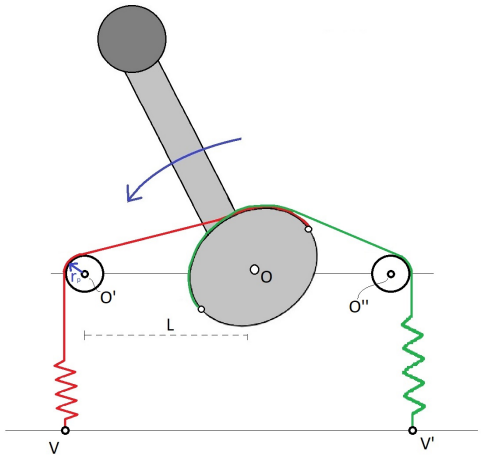


Figure 3.6: Addition of routing pulleys.

For an easier and more efficient placement of the belt elements, the belt was divided into sectors, as reported in Figure 3.7. Every sector is characterised by a different function defining the coordinates of the markers to originate each belt element.

- *Sector* 1 is the portion of belt positioned on the noncircular pulley and goes from point Q to R; where point Q is the point where the belt is fixed to the noncircular pulley, so $(x_P(0°), y_P(0°))$, and R is the tangency point of the belt on the noncircular pulley at the initial position of the model, so $(x_P(90°), y_P(90°))$.

- *Sector* 2 goes from the tangency point of the noncircular pulley R to the tangency

point on the circular pulley S.

- *Sector* 3 is the portion of belt placed on the circular pulley goes from S to T.

- *Sector* 4 is the final vertical portion of the belt that goes from the circular pulley in T to point U, where the belt is attached to the spring.

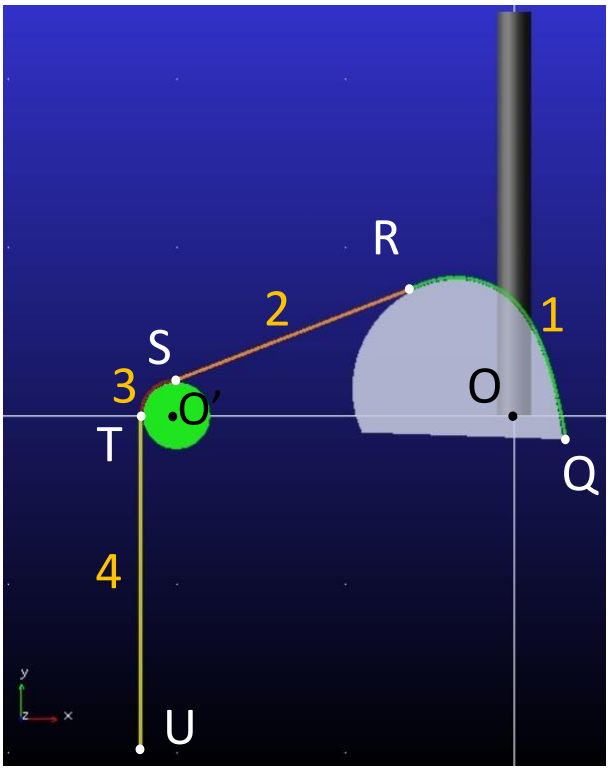


Figure 3.7: Division in sectors.

Regarding the belt elements dimensions, the height and width were set constant for all the sectors. Their values were decided arbitrarily, slightly oversized compared to the actual cable in order to have a bigger surface of contact with the pulleys. The density of mass assigned to the belt elements would be adjusted in a later phase, so that the inertia of the belt elements would still be realistic.
On the contrary, the length of the belt elements differed from sector to sector as their value was a consequence of the number of elements desired in each sector: n_{nonc}, n_{circ}, n_{line}.
n_{nonc} and n_{circ} were the preset numbers of belt elements positioned on the noncircular and circular pulley, n_{line} was the preset number of elements in *sector* 2 and in *sector* 3.

The process to arrange the belt in each sector was to discretise function $\gamma_k(\cdot)$ in the given number of elements, where $\gamma_k(\cdot)$ is the function describing the curve that the belt would follow in that given *sector* k.

This was done by writing the curve in a form $\gamma_k(\cdot)$ such that:

$$\begin{aligned} \gamma_k(i\Delta_k) &= X_i, \quad i \in [0, n_i] \\ \gamma_k(0) &= X_0 \\ \gamma_k(n_i\Delta_k) &= X_1 \\ X_i &\in [X_0, X_1] \end{aligned}$$

where k identifies the number of the sector (here from 1 to 4); X_0, X_1 are the initial and final point of the sector, X_i is a generic point between X_0 and X_1, and Δ_k is the increment used to discretise $\gamma_k(\cdot)$.
It was important to choose the suitable increment Δ_k for each *sector* k, as from it depends the length of the parallelepipedal belt elements.
Each belt element is then created from point X_i to X_{i+1}, where:

$$\begin{aligned} X_i &= \gamma_k(i\Delta_k) \\ X_{i+1} &= \gamma_k\big((i+1)\Delta_k\big) \end{aligned}$$

In the Adams environment, a parallelepiped, called *box* or *block*, is created by defining two opposite vertexes V_i, V'_{i+1}, as illustrated in Figure 3.8. These two points could be derived by offsetting X_i and X_{i+1}: V_i was obtained translating X_i of half the value of the width, while V'_{i+1} translating X_{i+1} of half the value of the width in the opposite direction and then perpendicularly of the height value.
X_i and X_{i+1} were also used to locate the revolute joint in between the belt elements. By not offsetting the revolute joints of half the value of the height, the line of action of the belt was kept on curve $\gamma_k(\cdot)$, simulating an ideal zero thickness cable.

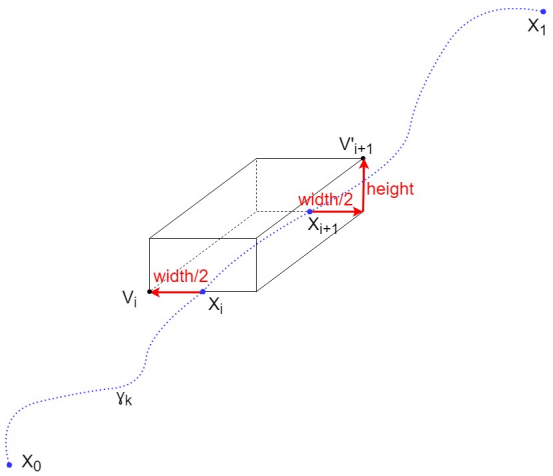


Figure 3.8: Definition of a belt element.

To arrange the belt elements in *sector* 1 and *sector* 3 a further operation was required: since the elements of these sectors were arranged over curved surface, they would interpenetrate the volumes of the pulleys. The interferences, however small, cause problems in the computations of the simulation. Since contacts had to be defined between the belt elements and the pulleys, γ_1 and γ_3 were offset to obtain γ'_1, γ'_3, as illustrated in Figure 3.9. For this reason, a period of time at the beginning of the simulation must be dedicated to let the belt fall and lean on the pulleys. The value of the offset between γ and γ' was set as a value that allows to arrange the desired number of element in that sector, without

interpenetration between the belt elements and the volume. The value was found by trial and error, trying to keep it as small as possible, in order to minimise the fall of the belt at the beginning of the simulation.

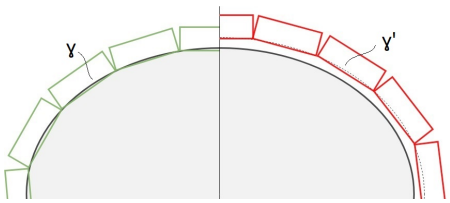


Figure 3.9: Offset curve to avoid interferences.

sector 1

Using the same algorithm to enlarge the profile of the noncircular pulley in Section 2.7, γ_1' was computed in the form of a polynomial in θ, the angle included between the x-axis and the pendulum axis.
Instead of transforming the function in a new form that would allow for a parametrization ensuring a constant length of the belt elements, γ_1 was discretised through regular intervals of θ, computed as:

$$\Delta_1 = \frac{90°}{n_{nonc}}$$

In Figure 3.10 it is possible to note how a constant increment of angle θ generates a particular distribution of the belt elements due to the way the profile of the noncircular profile was parametrised: the length of the elements is greater when the curvature of the profile is smaller. This allowed to have more little belt elements where the curvature is greater and less big elements where it is smaller, allowing to optimise the combination of the number, dimensions and arrangement of the belt parts.
Additionally, a belt element was added to connect the belt to the noncircular pulley. The element was created so that one face touched point $\gamma_1'(0)$, to connect it to the first belt element. The piece can be seen in red in Figure 3.10.

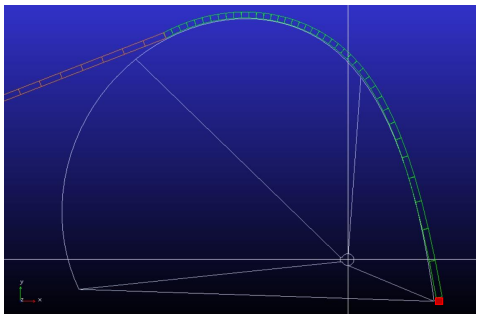


Figure 3.10: Belt elements in *sector* 1.

sector 2

To arrange the belt elements in *sector* 2, the length $\left|\overline{RS}\right|$ and inclination α of the segment from point R to S were computed, taking into account that the two points are offset from

the pulleys' profiles. Then the points X_i to create the belt elements were found starting from R and moving with inclination α of a length l_{line1} for n_{line} iterations; where:

$$l_{line1} = \frac{|\overline{RS}|}{n_{line}} \tag{3.1}$$

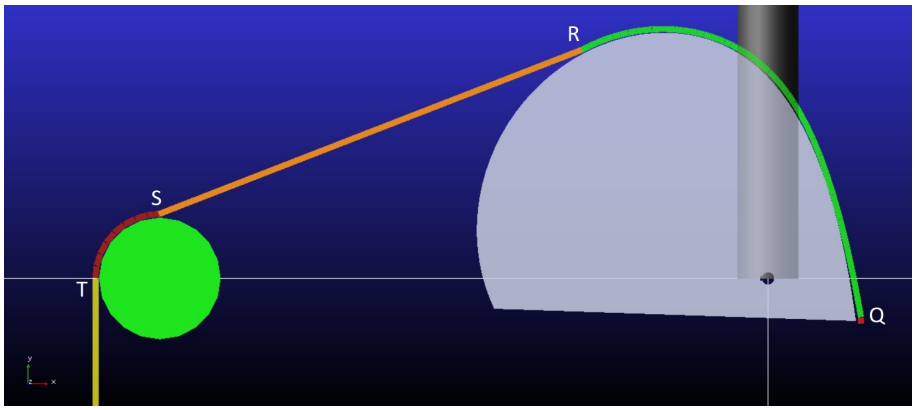


Figure 3.11: Particulars of the belt over the 2 pulleys.

sector 3 and routing pulley

The routing pulley was modelled with a cylinder slightly longer than the width of the belt and connected to the ground with a revolute joint.
Concerning the belt elements of *sector* 3, the coordinates of points X_i were found from the parametric equations of a circumference using angular increment:

$$\Delta_3 = \frac{90°}{n_{circ}} \tag{3.2}$$

In this sector, curve γ_3' has a constant curvature and the belt elements obtained had constant length.
To offset the belt elements from the pulley, the quarter of circumference γ_3' has a radius bigger than the circular pulley.

sector 4

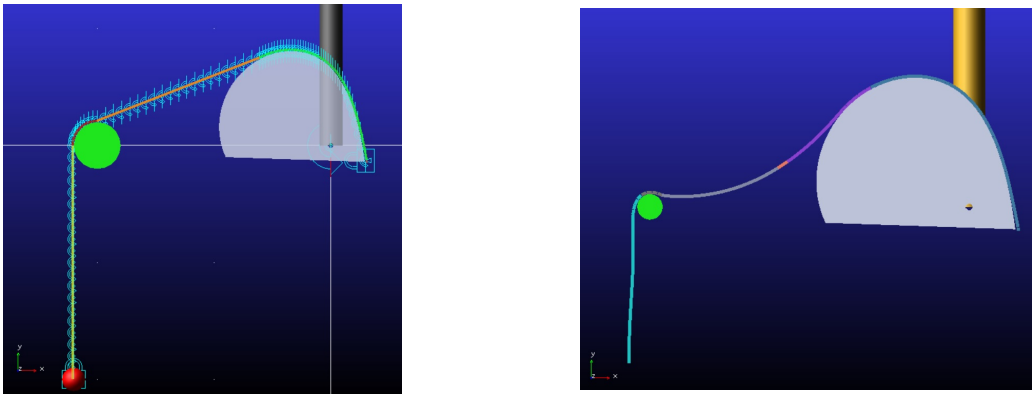
The same method of *sector* 2 was used to locate points X_i in *sector* 4, with the difference that it was not necessary to find the inclination, since the belt elements were arranged on a vertical line. In this phase, the x-coordinate of point U was set to have the same x-coordinates of T, while the y-coordinate was set to obtain belt elements of the same length of *sector* 3.

Final adjustments on the belt

While creating the different sectors, the belt elements were ordered by associating a number to each element, starting from the one connecting the noncircular pulley to the

belt in Q, and finishing with the one connecting the belt to the spring in U. In this way the revolute joints and the contacts with the pulleys could be added within a unique cycle. The revolute joints were added to link each belt element to the next one, while the first belt element was connected to the noncircular pulley through a fixed joint.
Contacts between the belt element and the noncircular pulley were assigned to *sector* 1 and *sector* 2; whereas contacts between the belt element and the circular pulley were assigned to *sector* 2, *sector* 3 and *sector* 4.

Next, a spherical mass was created to tighten the belt. The choice to add a mass instead of the spring was made with the aim of not overcomplicating the model and being able to more easily check the correct functioning of the belt. The mass was connected to the final belt elements trough a fixed joint.
Figure 3.12a reports the mass at one end of the belt and the joints highlighted in light blue.
The belt had to be tightened during the simulation of the intermediate models to avoid it from getting loose and falling in between the two pulleys, as in Figure 3.12b.



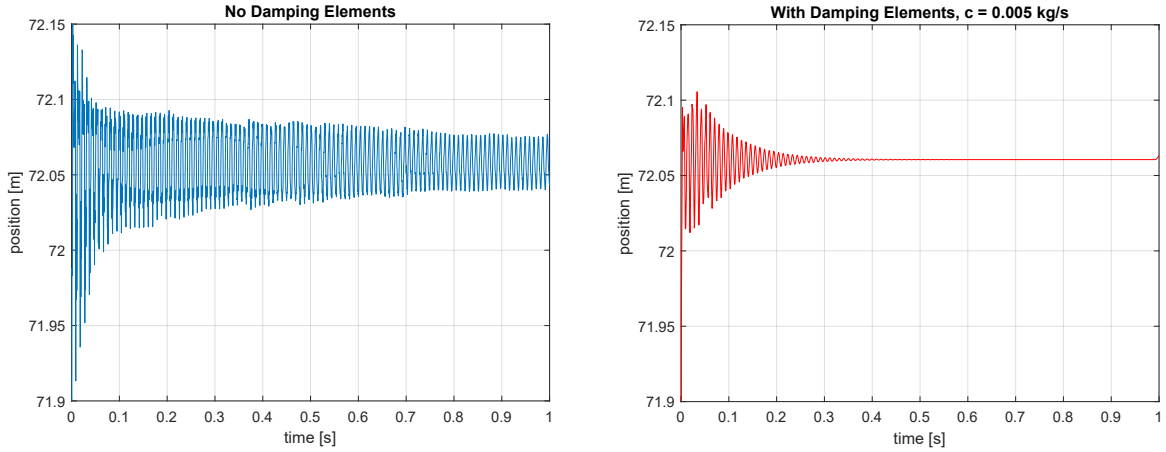
(a) (b)

Figure 3.12: Tighten and loose belt.

The spherical mass was then substituted by a constant single-component force, in order to obtain the same effect of the tightening spherical mass, but avoiding to add further mass and inertia to the belt. The force was located in the final belt element, directed in the opposite direction to the y-axis and with modulus equivalent to that generated by the spherical mass.

Analysing the results of the simulations of these first models, it was noted that the belt would fall and lay on the pulley as expected, but then it would start vibrating almost perpetually. This happened because the energy gained by the belt during the fall would cause it to start vibrating; then the lack of damping in the system would prevent the vibrations from being promptly reduced and cancelled.
Hence a damping element was added to each revolute joint connecting the belt elements. This addition was key as it drastically reduced the computation time of the simulations, highlighting the importance of adding damping to every dynamic model.
Another way to model the damping into the belt would have been adding friction to the revolute joints. However, friction is modelled in a more complex and less robust

manner, that would make the computation of the simulation more time-consuming and less accurate. Consequently, it was preferred to add the damping elements. Figure 3.13 reports the distance from the origin of the centre of mass of a belt element in *sector* 3 in two different cases: without and with damping elements in the belt.
In Figure 3.13b, where there was a damping element at each revolute joints of the belt, it is possible to see how the vibrations were damped of over 99% after a transient of $0.4\,s$. On the contrary, in Figure 3.13a the vibrations have a much longer transient, since the only damping into the system was provided by the friction in the contacts belt-pulleys.



(a) (b)

Figure 3.13: Effect of the damping elements.

3.5 Fine-tuning of the simulation parameters

Considering that the computation of simple simulations of the obtained model were slow and some of the results not as smooth as desired, it was decided to perform a first fine-tuning of some of the parameters of the model rather than carrying on to improve the layout of the belt. This operation aimed at shortening the computation time of the simulation, so that it would be faster to check the correct working of the model after every adjustment, and at obtaining clearer results with less spikes and disturbances.

The design parameters calibrated were:

- thickness and width of the belt elements
- damping value in the revolute joints
- contact parameters (damping, stiffness, static and dynamic friction, force exponent, penetration depth)
- solver's integrator, formulation and error
- step size of the simulation

Since it was not conceivable to assess all the possible combinations of parameters, they were fine-tuned one at a time. Each value was varied several times and tested by computing a simple simulation of the model. Once that the value of the parameter was considered satisfactorily optimized, another parameter was assessed.
To estimate whether the variation of the values brought benefits to the simulations or not, particular attention was brought to checking whether the simulation would crash or not, if the computation time would sensibly decrease, if less spikes appeared in the plot of the results, especially the one reporting the force of contact between pulleys and belt.
The above-mentioned simulation consisted simply in rotating the pendulum at a speed of $1\,deg/s$ after an initial settling time of $1\,s$, during which the pendulum was locked to the ground.
[16] was essential in guiding the process of fine-tuning, also by suggesting the range of possible values of some of the parameters.
The optimal values are summarised in Table 3.2.

Table 3.2

Thickness of belt elements	$0.4\,mm$
Width of belt elements	$5.0\,mm$
Damping value in the revolute joints	$0.005\,kg/s$
Contact damping	$1.0 \cdot 10^4\,kg/s$
Contact stiffness	$1.0 \cdot 10^5\,N/mm^2$
Contact μ_s	0.99
Contact μ_d	$0.9\,m$
Contact force exponent	2.2
Contact penetration depth	$0.1\,mm$
Solver integrator	HHT
Solver formulation	I3
Solver error	$1.0 \cdot 10^{-5}$
Simulation step size	$0.001\,s$

Belt dimensions

There was a slight improvement in the computation time decreasing the dimensions of the belt elements. This event was related to the decrease of the inertia of the belt elements and the relative reduction of the amplitude of the vibrations.
There was no evidence of a alteration of the performance caused by the variation of the contact area. Nevertheless, it was preferred to provisionally leave a bigger contact surface to avoid possible problems, with the intention of attempting to decrease it in a later phase of the design.

Damping value in the revolute joints

The damping value was fine-tuned with the intention of decreasing the computation time and obtaining a transient of the initial vibrations with a reasonable duration. The trade-off was found at a value of $0.005\,ks/s$, with the vibrations reduced of over 99% of their amplitude after a transient of $0.4\,s$.

Contact parameters

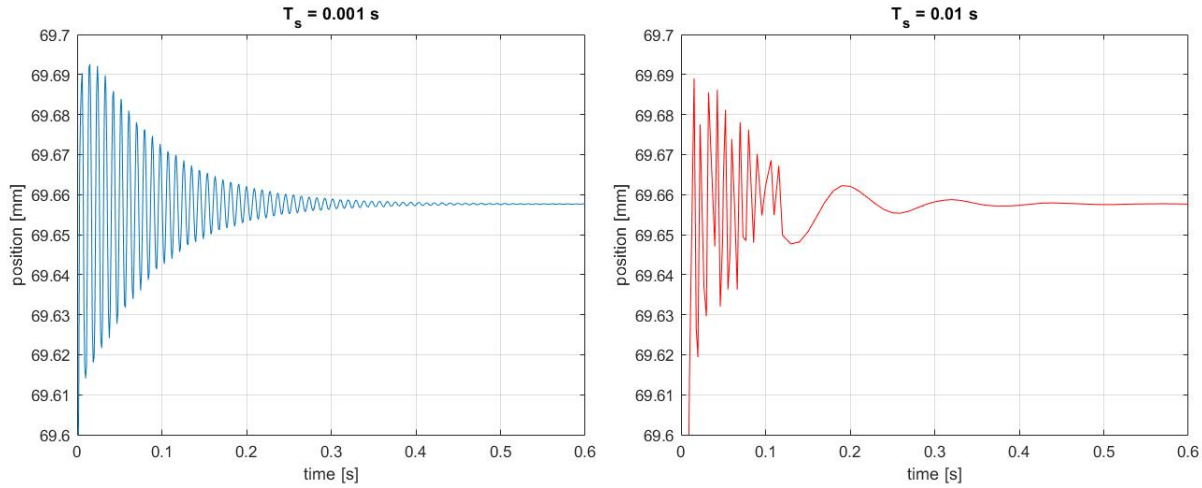
The default value of the stiffness resulted to be satisfactory. Consequently the value of the damping was set at $1 \div 10\%$ of the stiffness, as [16] suggests.
The value of the static and dynamic friction was set very high, so that there would not be any relative motion between the pulleys and the belt.
[16] recommend a force exponent higher than 1.5. An exponent greater than 2.1 is suggested to gain more stability. After the simulations, an exponent at 2.2 was considered optimal.
Regarding the value of the penetration depth, the value suggested by [16] resulted to be adequate.

Solver

Only two solvers were tried: the Gear stiff (GSTIFF) and the Hilber-Hughes-Taylor (HHT). GSTIFF is Adams default solver, and showed to provide acceptable results in the Stabilised Index 2 (SI2) formulation. [16] reccomends this formulation for most contact models as it is more robust and produces less spiky results.
HHT provided slightly less smooth results, but required a shorter computation time, and it was therefore referred. The efficiency of the HHT solver in this type of simulations is reflected also by the fact that is the default solver used by the machinery suite when there are models with numerous contacts, like a gear mechanism.
The default solver error for the HHT solver was kept as it provided a stable computation.

Step size

A bigger step size caused the computation to crash or to provide poor results, as the step size was too big to follow the vibrations of the belt. Moreover in case of model with many contacts, a small step size is suggested. Figure 3.14 reports the distance from the origin of the centre of mass of a belt element in *sector* 3, computed using two different sampling sizes. In 3.14b the sampling time was too big and the simulation was not able to track the movements due to the vibrations.
When the step size was set too low, it proved to slow down the computation and, in more advanced simulations, to produce noisier results. Therefore, a trade-off value was found at $0.001\,s$.



(a) (b)

Figure 3.14: Simulations with different sampling times.

3.6 Second belt model

The preliminary model described in the previous Section had the merit of consolidating the operations to arrange the belt elements and to fine-tune the contact's parameters, but left still room for improvement. Therefore, the macros to arrange the belt elements were rewritten to obtain a more functional disposition of the belt that required less elements.

The criteria used to optimize the design were to:

- minimise the length of the belt
- minimise the number of contacts to be introduced
- customise the length of the belt elements to the curvature of the pulley that they would enter in contact with.

This meant that the spring could be attached to the belt element that would just enter in contact with the circular pulley when the pendulum was at an angle θ of 30°. On the symmetric pulley-belt-spring system, this limited the pendulum to reach a maximum angle θ of 150°. In this way less belt elements were needed with respect to the previous configuration, as there are no redundant belt elements.

The number of contact elements introduced was decreased by assigning a contact only to the belt element that would actually enter in contact with one of the pulley across the angular range. By limiting the angular range of θ to $30° \div 150°$, a sector of the belt resulted to never enter in contact with neither pulley. This section would be made up by a single belt element, with no contacts assigned to it.

The belt elements that would enter in contact with the circular pulley were all created with the same length, to adapted them to the curvature of the pulley.
The length of the belt elements that would touch the noncircular pulley was less straightforward to outline. In the previous model of the belt, only the sector that would touch the noncircular pulley in the initial position had the length customised to the curvature of the pulley, not the entire section of belt that would lay on the pulley across the range. In this second model, a method to customise the length of the entire all the belt elements that would touch the noncircular pulley was developed.

3.6.1 Preparation

First of all, the system composed by pendulum, noncircular pulley, routing pulley, belt and spring was schematised in the initial position of the simulation, and in the two extreme positions of the angular range (30° and 150°), in order to better understand the geometry of the problem.
Figure 3.15a reports some of the parameters that are used in the dissertation: the pendulum angle θ, the reference frame xy used in the model, the insertion length L, that is the distance of the routing pulley from the revolute joint. Moreover, it reports some significant points of the belt: Q is the point where the belt is attached to noncircular pulley, R is the tangency point of the belt on the noncircular pulley, S and T are the tangency points of the belt on the circular pulley, V is the point where the spring is connected to the ground. x_V is the x-coordinate of point V, that is the distance between T and V. c is the length of the circular arch between S and T.
Figure 3.15b, 3.15c, 3.15d report the three significant orientations of the model: with angle θ at 90°, 30° and 150°. The belt was divided into 4 sectors as in the previous configurations, but a different denomination was introduced: l is the sector of belt laying on the noncircular pulley, b is the sector between the two pulleys from R to S, c is the sector laying on the circular pulley, d is the vertical sector after the circular pulley any pulley between point T and U.
Point U is the point where the extremity of the belt is attached to the spring. The length of the segment representing the spring is denominated x, and is situated between point U and V.
The points and sectors that are not constant in θ, were characterised by a subscript indicating the value of the angle θ. Only point S, T, V and sector c are constant for every θ.

The coordinates of point S and T were defined by the geometry of the components:

$$S : \begin{pmatrix} r_{circ} \\ L \end{pmatrix}$$
$$T : \begin{pmatrix} 0 \\ L + r_{circ} \end{pmatrix}$$

where r_{circ} is the radius of the circular pulley.

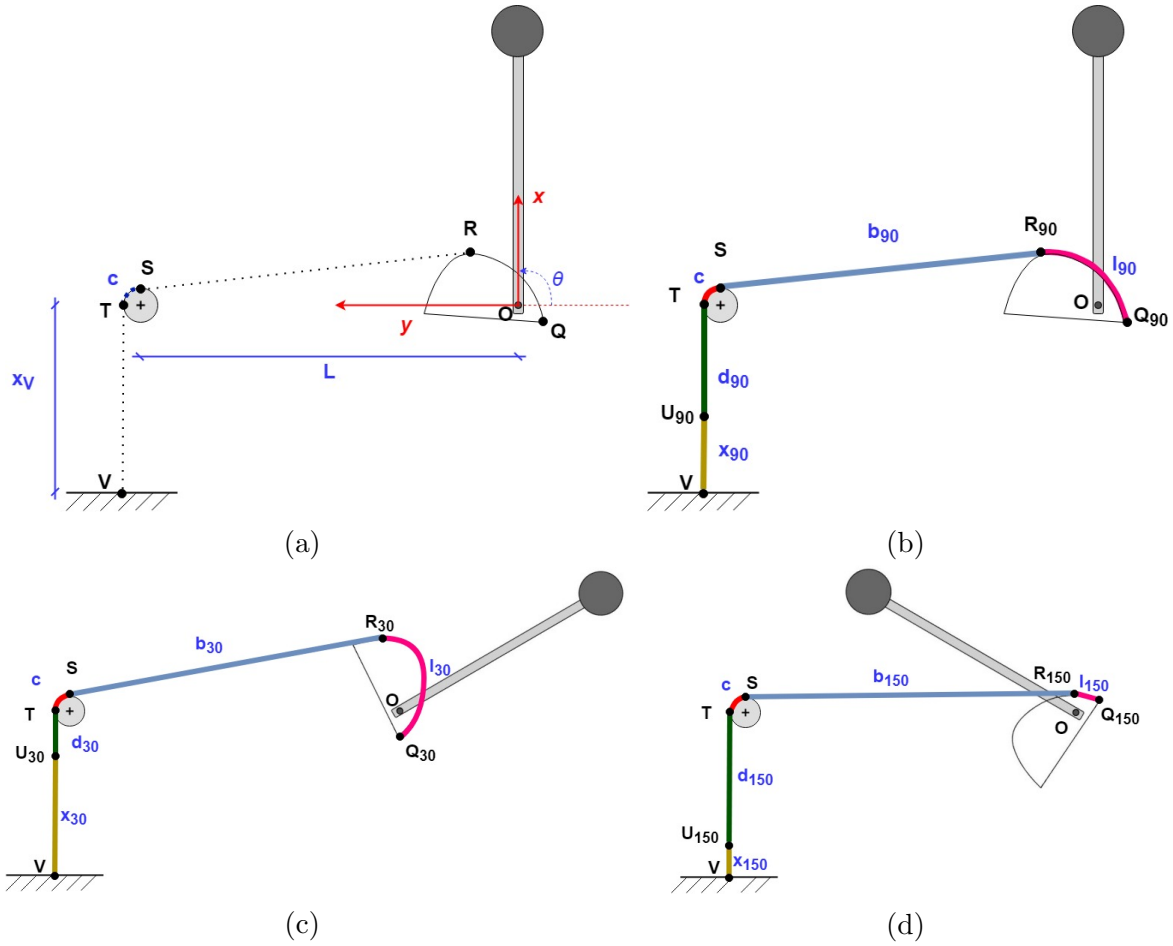


Figure 3.15: Schematic representation of the system across the angular range.

The coordinated of point $Q(\theta)$ were computed from the parametric equations of the non-circular pulley profile $(x_P(\theta); y_P(\theta))$ for $\theta = 0$ and then rotated according to the case:

$$
\begin{aligned}
Q_{90} &: \mathbf{R}(90) \begin{pmatrix} x_P(0) \\ y_P(0) \end{pmatrix} \\
Q_{30} &: \mathbf{R}(30) \begin{pmatrix} x_P(0) \\ y_P(0) \end{pmatrix} \\
Q_{150} &: \mathbf{R}(150) \begin{pmatrix} x_P(0) \\ y_P(0) \end{pmatrix}
\end{aligned}
$$

where $\mathbf{R}(\alpha)$ is the rotation matrix of a generic angle α:

$$
\mathbf{R}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}
$$

Similarly, the coordinates of point $R(\theta)$ were computed from $(x_P(\theta); y_P(\theta))$ and then

rotated of the same angle θ:

$$R_{90} : \mathbf{R}(90) \begin{pmatrix} x_P(90) \\ y_P(90) \end{pmatrix}$$
$$R_{30} : \mathbf{R}(30) \begin{pmatrix} x_P(30) \\ y_P(30) \end{pmatrix}$$
$$R_{150} : \mathbf{R}(150) \begin{pmatrix} x_P(150) \\ y_P(150) \end{pmatrix}$$

The lengths of l_{30}, l_{90}, l_{150} were computed using the formula to estimate the arc length of a curve in parametric form:

$$l_\theta = \int_0^\theta \sqrt{\left(\frac{dx_P(\alpha)}{d\alpha}\right)^2 + \left(\frac{dy_P(\alpha)}{d\alpha}\right)^2} d\alpha$$

Whereas b_{30}, b_{90}, b_{150} were computed as the distance from point R_θ to S.

$$\left|\overline{R_\theta S}\right| = \sqrt{(x_{R_\theta} - x_S)^2 + (y_{R_\theta} - y_S)^2}$$

From these values, the arrangement of the belt in the initial position of the simulation ($\theta = 90°$) was laid out, following the above-mentioned criteria to improve the model.

In order to shorten the belt, the sector of belt d_{30} in Figure 3.15c could be eliminated, as it was not essential. Hence the total length of the belt l_b could be computed as:

$$l_b = l_{30} + b_{30} + c$$

From the length of the belt l_b, it is possible to compute the length of the vertical section of the belt d:

$$d_{90} = l_b - (l_{90} + b_{90} + c)$$
$$d_{150} = l_b - (l_{150} + b_{150} + c)$$

The sector of belt l_{nonc} that would enter in contact with the noncircular pulley coincides with sector l_{30}:

$$l_{nonc} = l_{30}$$

While the length of the sector l_{circ} of belt that would touch the circular pulley across the angular range from 30° to 150° is:

$$l_{circ} = d_{150} + c$$

The section of belt that touches both pulleys l_{both} is the difference between l_{circ} and length of belt that would never touch the noncircular pulley ($b_{30} + c + d_{30}$). As a result:

$$l_{both} = d_{150} - b_{30}$$

If l_{both} is a negative number, that means that there are no elements in contact with both pulleys, but there is a sector long $-l_{both}$ that does not enter in contact with neither pulleys across all the angular range.

The x-coordinate of point V was computed summing up the vertical sector of belt d_{90} and the spring length x_{90}, both at the initial angle of 90°:

$$x_V = d_{90} + x_{90}$$

x_{90} was obtained from the value of the spring force:

$$F_{s_{90°}} = \frac{\tau_{d1}(90°)}{r_m(90°)} = k\,(x_{90} + x_0)$$

$$x_{90} = x_0 + \frac{F_{s_{90°}}}{k}$$

where x_0 is the elongation of the spring at rest.

3.6.2 Deployment of the belt elements

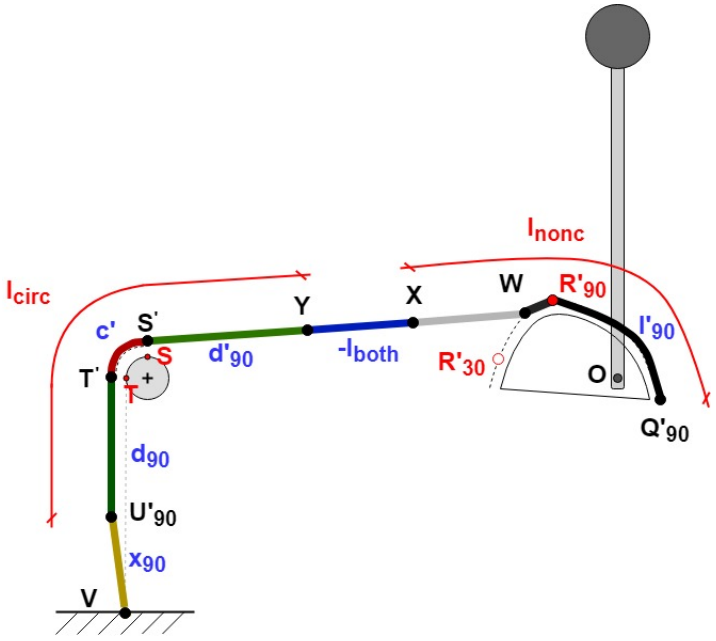


Figure 3.16: Schematic representation of the initial configuration of the belt.

At this point, all the values needed to arrange the belt had been computed, and the new model of the belt could be created. Figure 3.16 reports the draft used to create the belt elements in the final configuration.
As in the previous model of the belt, the sectors on the circular and noncircular pulleys were offset to avoid interpenetration. Points Q'_{90}, R'_{90}, S', T' are the equivalent of Q_{90}, R_{90}, S, T on the new curves. As a consequence, the lengths of the sectors l'_{90} and c', laid

on the two pulley from Q'_{90} to R'_{90} and from S to T, would be slightly longer than l_{90} and c, thought the difference was small and considered negligible.
Also the point of connection between belt and spring U_{90} was moved to U'_{90} in order to align it with V' and simplifying the process of deployment of the belt elements in the segment $\overline{T'U'_{90}}$.

Once again, the belt elements were ordered by associating an increasing number to the name of each part, from point Q'_{90} to U'_{90}.

Noncircular pulley sector

Differently from the previous model, the number of belt elements on the noncircular pulley n_{nonc} was now the number of belt elements that would lay on the pulley across the angular range, that is to say the belt elements in section l_{nonc}.
The angular increment was obtained dividing the maximum angle of the range ($\theta = 150°$) by the number of elements:

$$\Delta_l = \frac{150°}{n_{nonc}}$$

Then, the belt elements were created along the offset curve until 90° was overcome. The point reached by the belt elements, called W, is likely after R'_{90}, due to the discretisation of the curve. This section is represented in black in Figure 3.16.
From point W, the belt elements were arranged on the line from W to S', until the n_{nonc} elements had been created (the grey sector in Figure 3.16). The final point of this section was called X.
The lengths of the belt elements was computed by an algorithm that acted in parallel to the creation of the belt element parts. It worked by continuing to progress along the offset curve with angular increment Δ_l and computing the distance between the points, until the tangency point R'_{30}.
In this way, when the pendulum rotates, the belt elements that wrap the noncircular pulley have the length customised to the curvature of the pulley.

Circular pulley sector

The number of belt elements on the circular pulley n_{circ} was one of the design parameters. The length of the belt elements lb_{circ} was computed so that the quarter of circumference would be divided by the integer number n_{circ}:

$$\frac{lb_{circ}}{2} = r^+ \sin\frac{\Delta_2}{2} = r^+\sqrt{\frac{1-\cos\Delta_2}{2}}$$

$$lb_{circ} = r^+\sqrt{2\,(1-\cos\Delta_2)}$$

where r^+ is the radius of the offset quarter of circumference, and Δ_2 is the angular increment computed as:

$$\Delta_2 = \frac{90°}{n_{circ}}$$

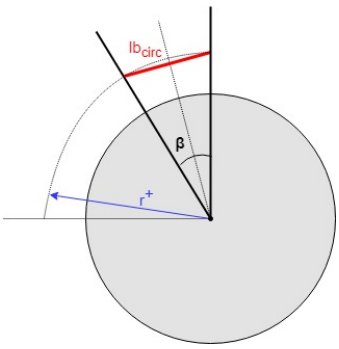


Figure 3.17: Schematic representation of the circular pulley used to compute lb_{circ}.

Straight line sectors

The belt elements in sector d_{90}, from Y to S', and d'_{90}, from T' to U'_{90} were all created with length lb_{circ}, as they would all enter in contact with the circular pulley.
Sector d'_{90} is computed as the difference between the length of belt l_circ that touches the circular pulley across the range sector for θ equal 150°, and the vertical sector at 90° plus c:

$$d'_{90} = l_{circ} - (d_{90} + c) = d_{150} - d_{90} \tag{3.3}$$

Point Y was found progressing along segment $\overline{WS'}$, starting from S', at steps long lb_{circ} and stopping immediately after having covered a distance longer than d'_{90}.
Once that the sectors d'_{90}, c' and d_{90} are defined, the elements in those sectors are created in the same way explained in Section 3.4.

Finally a single element was created in sector l_{both} between X and Y, to connect the two sectors l_{nonc} and l_{circ}. The reason to make up this sector with a single element was that it would never enter in contact with any of the pulleys, and therefore never lay down on a curved surface.

3.6.3 Final adjustments

After having created all the belt elements, the revolute joints connecting them and the contacts were added. A contact with the noncircular pulley was added only to the belt elements between Q'_{90} and W. A contact with the circular routing pulley was added only to the belt elements between Y and U'_{90}.

Next, as in the previous model, a force was added to the terminal point of the belt U'_{90} in order to tighten the belt during the preliminary simulations.

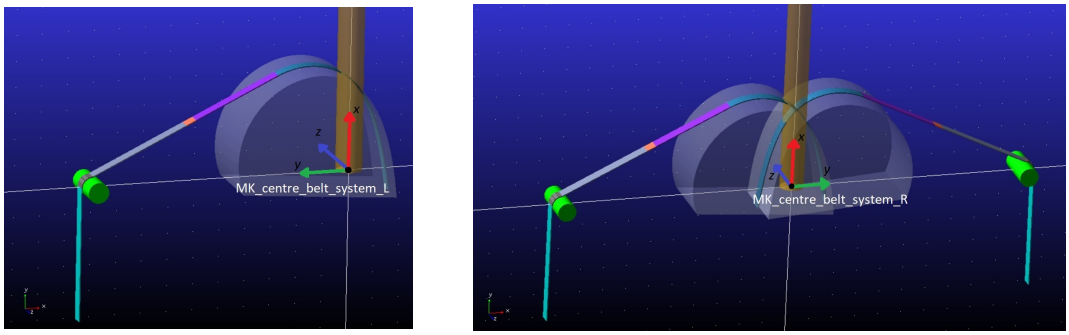
After having tested the model with the tightening force, a spring was added between point U'_{90} and V. The spring was modelled as a single-component force, whose value was defined by the function:

$$F_s = -k_s \cdot (\mathrm{DM}\,(V, U'_{90}) - x_0) - c_s \cdot \mathrm{VM}\,(V, U'_{90})$$

where k_s and c_s are the spring stiffness and damping respectively; DM(A, B) and VM(A, B) are two functions in Adams to compute the distance and relative velocity between two points A and B.
[16] recommends to add some damping to elastic elements in order to stabilise the system. The damping value is usually $1 \div 10\%$ of the stiffness.

3.6.4 Symmetrical mechanism

Once that the left spring-belt-pulleys model was completed, the symmetric right one was created.
With the aim of simplifying this operation, the position and orientation of all the parts making up the left mechanism were defined with respect to a reference frame located in the origin called *MK_centre_belt_system_L*. For the same reasons, the suffix *_L* was added to the name of all the elements making up the left mechanism.
In this way, the right mechanism was easily created by copying all the *macros* used to create the left mechanism, rotate the reference frame *MK_centre_belt_system_L* of 180° along its x-axis (see Figure 3.18) and substitute all the suffixes *_L* with *_R* to all the elements, in order to avoid duplicates.
This operation was possible since the two mechanisms were designed symmetrical to each other by setting the γ parameter equal to 0.5, and the model of the left mechanism was created symmetrical with respect to the xy-axis.
The ease and swiftness with which this operation was accomplished are clear evidences of the advantages in creating complex models using macros and the *Adams/View Command Language*.



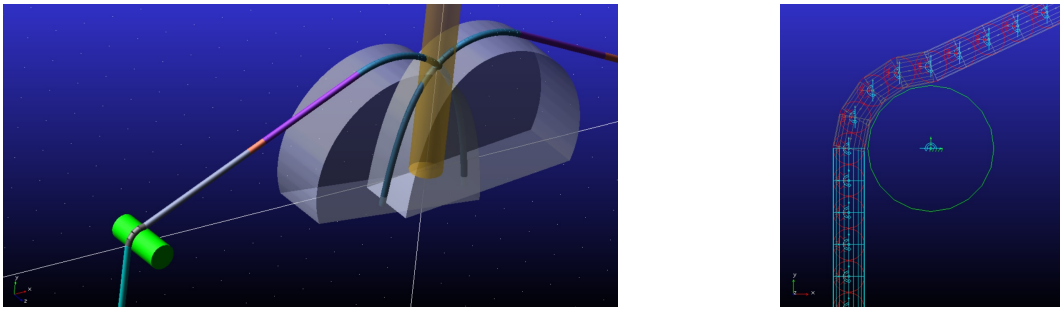
(a) (b)

Figure 3.18: Creation of the symmetric mechanism.

3.6.5 Further improvements

After having successfully tested the complete double mechanism, some final improvements were made to the model. First of all the section of the cable was modified from rectangular to circular to make it more similar to the actual prototype. This was accomplished by creating cylindrical belt elements instead of parallelepipedal ones. In addition the revolute joints between the belt elements were moved to the axis of the cylinders in order to simulate the thickness of the cable. In this way, the line of action on the cable was on the

axis of the cylindrical cable (Figure 3.19b). To do so, the offset curves used to create the elements on the pulleys were further offset of half the thickness of the cylindrical cable. The reduction of the contact areas between the cable and the pulleys did not seem to create problems to the simulation. Nevertheless, an additional fine-tuning of the parameters was carried out, as the one described in Section 3.5, aiming for a further improvements of the simulations. None of the changes to the parameters appeared to bring any improvements and the parameters were hence left unaltered.



(a) (b)

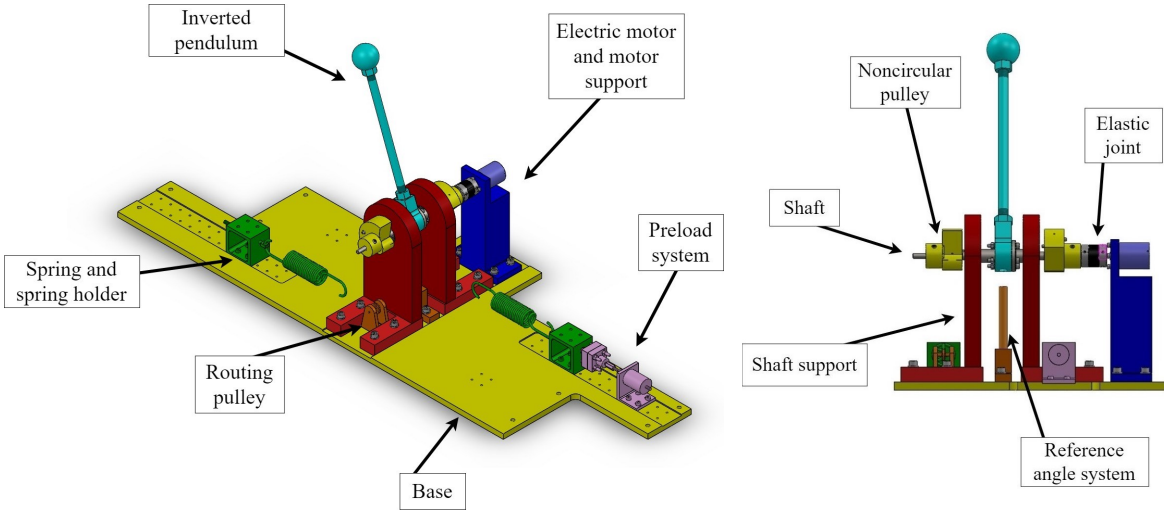
Figure 3.19: Cylindrical cable elements.

Lastly, the values of mass and inertia of the pendulum and the dimensions of the pulleys were substituted by the values of the actual prototype. The profile of the noncircular pulleys were synthesised taking into account the geometrical corrections due to the routing pulleys and cable thickness.

Chapter 4

Physical prototype

After having tested the concepts of the gravity compensator through the dynamic simulations of the Adams model, a 3D model of the prototype was designed. The aim was to build a test bench to verify the effects of the noncircular pulley-spring mechanism on an inverted pendulum in real-world experiments and to collect data to validate the virtual model.
The criteria that guided the design were to realise a sturdy and reconfigurable test bench. A sturdy and robust structure has the merit to avoid vibrations and bending, allowing to collect less noisy results from the tests.
At the same time, the design had to be easy to reconfigure in order to allow to perform tests varying some of the parameters. In particular, it was decided to provide the possibility to substitute the noncircular pulley and to modify all the parameters that affect the synthesis of the noncircular pulley profile; namely, the routing pulleys' radius, the insertion length, the springs and their preload, the mass and inertia of the pendulum.



(a) (b)

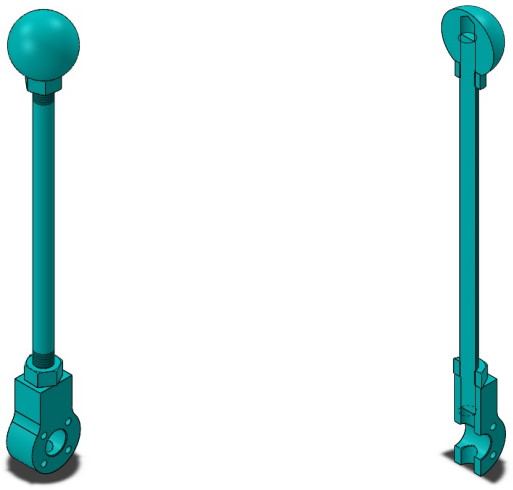
Figure 4.1: 3D model of the test bench, highlighted with illustrative colours.

4.1 General description

The 3D model was created using the solid modelling CAD software SolidWorks®. Despite the numerous iterations needed to accomplish the definitive version of the test bench, only the final design was considered worthy to be described in the following sections.
The dimension drawings of all the non-standard components are reported in Appendix C.

4.1.1 Pendulum

The pendulum design was divided into three components: a base to couple the pendulum to the shaft, a stem with threaded ends, and a spherical mass located at the end of the stem. Both the base and the sphere have a threaded hole to connect them with the stem. The connections were strengthen with nuts.
The bottom base of the pendulum was drilled to allow to couple it with the rod of the *reference angle system*. The hole is visible in Figure 4.2b.
The stem was made out of aluminium so that the mass would be more concentrated towards the extremity of the pendulum. The base and the sphere were made out of steel.
The connection to the shaft was realised through a robust flange coupling to avoid clearance and relative motions between the two parts.
The centre of mass of the pendulum could be modified either by substituting the sphere or the stem.



(a) (b)

Figure 4.2: Assembly and sectional view of the pendulum.

4.1.2 Electric motor and motor support

The maximum torque τ^p_{max} that could be acting on the shaft was computed to select the correct size of the electric motor. τ^p_{max} was set defining an extreme use of the pendulum with a desired maximum angular acceleration α^p_{max} of the pendulum equal to $0.5\,rad/s^2$

when the pendulum is in the horizontal position (either at 0 or $\pi\, rad$). The torque that has to be applied to the shaft is:

$$\tau^p_{max} = I_z \alpha_{max} + m_p l_p = 0.987 + 0.079 = 1.066\, Nm \tag{4.1}$$

The dimensions of the pendulum are summarised in Table 4.1.

Table 4.1: Pendulum's dimensions

Mass	$m_p = 411\, g$
Distance of the center of mass from the shaft's axis	$l_p = 192\, mm$
Moment of inertia along the shaft's axis	$I^p_z = 1.974\, kg\, m^2$

Consequently, the motor-gearbox combination had to be able to generate a torque greater or equal than τ^p_{max}. The value was not multiplied by a safety factor because it had already been intentionally overvalued.
The DC motor with graphite brushes RE 25 in combination to the planetary gearhead GP 32, both by "*Maxon motor Spa*", were selected among the equipment available in the laboratory where the experiments were undertaken. The actuation package was completed by an HEDS-5540 encoder.
Tables 4.2 and 4.3 summarise the main characteristics stated in the datasheets of the motor and the gearhead.

Table 4.2: RE 25 motor data

Motor data	
Nominal Voltage	$48V$
No load speed	$10300rpm$
No load current	$20.1mA$
Nominal speed	$9160rmp$
Nominal torque (max. continuous torque)	$27.7mNm$
Nominal current (max. continuous current)	$0.653A$
Stall torque	$264mNm$
Stall current	$6.03A$
Maximum efficiency	87%
Terminal resistance phase to phase	7.96Ω
Terminal inductance phase to phase	$0.832mH$
Torque constant	$43.8mNm/A$
Speed constant	$218rpm/V$
Speed/torque gradient	$39.6rpm/mNm$
Mechanical time constant	$4.37ms$
Rotor inertia	$10.5gcm^2$

The maximum torque generated by the electric motor is:

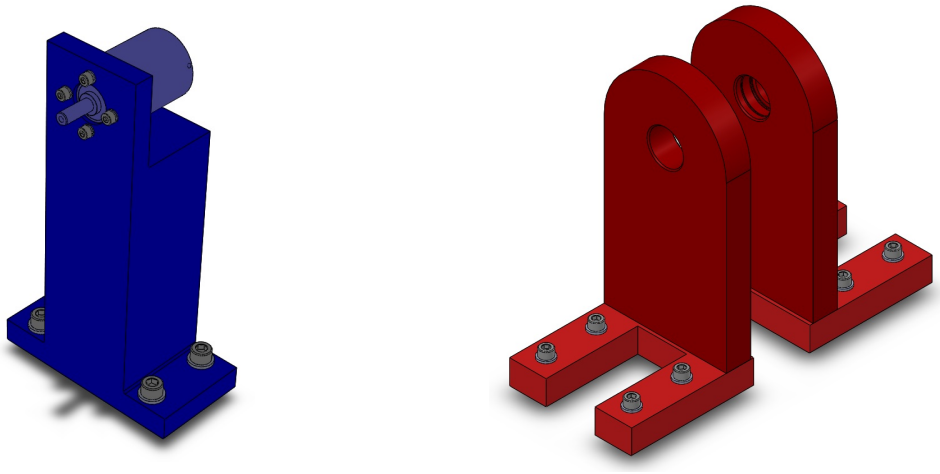
$$\tau^m_{max} = \eta_{gear} k_{gear} t_{motor} = 1.659\, Nm \tag{4.2}$$

$$\tau^m_{max} \geq \tau^p_{max}$$

Table 4.3: GP-42-C gearhead data

Gearhead data	
Reduction	86 : 1
Absolute reduction	14976/175
Number of stages	3
Maximum continuous torque	$6Nm\%$
Maximum intermittent torque	$7.5Nm$
Maximum efficiency	70%
Mass inertia	$0.7gcm^2$

The support that holds the motor has slots to house bolts that lock the support to the base, allowing to regulate the axial distance to the supports of the shaft.



(a) (b)

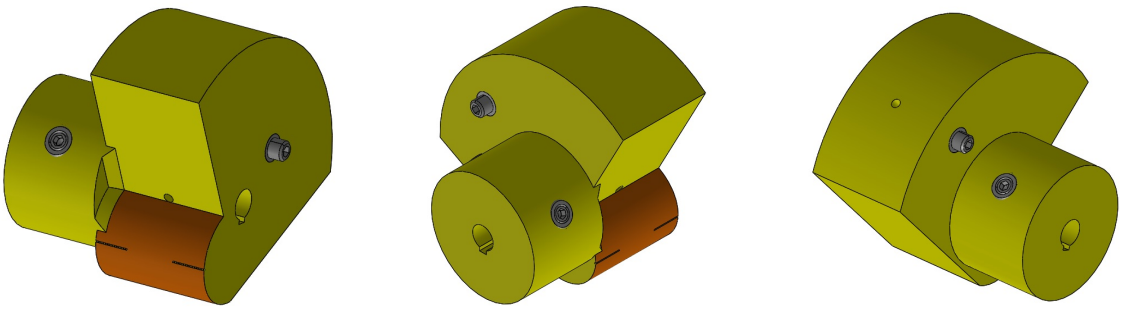
Figure 4.3: (a) Assembly of the electric motor and its support. (b) Assembly of the two shaft supports.

4.1.3 Shaft supports

The shaft supports were designed with a sturdy structure, like the motor support. Each support was divided into two parts to simplify its manufacture and decrease the cost of production.
The two parts were connected by two bolts and two centring pins to ensure their correct relative placement.
Since the two ball bearings were both secured on the shaft with an interference fit, and axially locked with shoulders and Seeger rings, the housings on the shaft supports were designed so that one of the two bearings would be axially locked and the other would be mobile. In this way, eventual inaccuracies in the manufacture or the assembly of the components would not cause damaging stress in the structure.
The supports are fixed to the base with bolts. One of he two support has holes for the bolts, while the other has slots, in order to regulate the distance between the two.

4.1.4 Noncircular pulleys

The 3D model of the noncircular pulleys was subdivided into three parts according to their function: one part to couple to the shaft, on that contains the proper noncircular profile section, and one to secure the cable.
The cylindrical section with the function of coupling to the shaft houses a feather key to transmit the torque and two grub screws at 120° to lock the pulley and avoid any relative motions.
The section with the synthesised profile has two incisions to mark the tangency point of the cable when the pendulum is in vertical position. This section is coloured with a darker shade in Figure 4.4.
The part of the noncircular pulley designed to secure the cable has a hole to house the cable. The hole starts at the tangency point of the lowest angle of the angular range, i.e. 30° in this case, and perforates the pulley. The cable would be inserted and secured with two grub screws perpendicular to the axis of the cable. The two screws are not in line one to the other to ensure a better grip on the cable.
Due to the unconventional shape of the pulley, particularly of the noncircular profile, it was decided to 3D printing the component. This choice also allowed to print the pulley after the manufacture of the other components of the test bench, so that it was possible to properly measure all the parameters needed to synthesise the noncircular profile (insertion length, spring constants,etc.).
In addition 3D printing would allow to manufacture pulleys with different profiles in fairly short times.
The pulley was printed out of ABS, as it was the only material available in the laboratory. A cable with rubber coating was selected in order to better secure the cable in the pulley and to prevent consuming the surface of the pulley that was made out of a softer material. Threaded inserts were heated and forced into the pulley to house the grub screws.

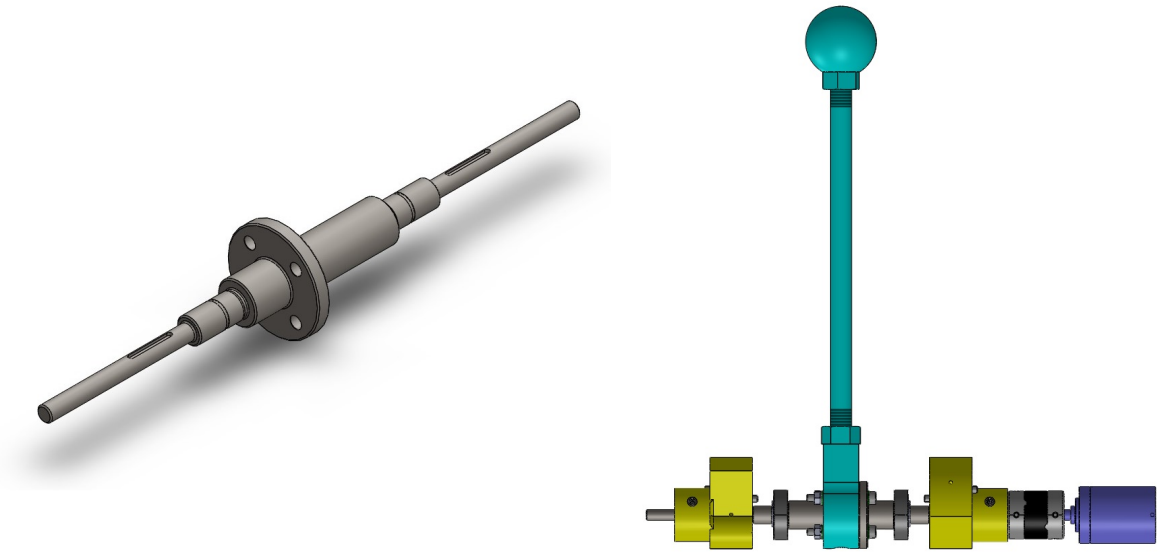


(a) (b) (c)

Figure 4.4: 3D model of the noncircular pulley with threaded inserts and grub screws.

4.1.5 Shaft

The overall shape of the shaft was defined by its couplings with the various components: a flange for the pendulum, shoulders and Seeger ring grooves for the two bearings, keyway slots for the two noncircular pulleys.



(a) (b)

Figure 4.5: (a) 3D model of the shaft. (b) Assembly view of the shaft and all the components coupled to it.

The minimum acceptable diameter for the shaft was sized from the value of τ^m_{max}. According to the Tresca yield criterion, the principal stress in each section is computed as:

$$\sigma_{id} = \frac{1}{W_f}\sqrt{\sigma^2 + 4\tau^2} = \frac{1}{W_f}\sqrt{\left(\frac{M_f}{W_f}\right)^2 + 4\left(\frac{M_t}{W_t}\right)^2} \tag{4.3}$$

where M_f is the bending moment, M_t is the torque, W_f the flexural modulus, W_t the torsion modulus, all relative to a given section of the shaft.

Since:

$$W_t = 2W_f$$

Equation 4.3 could be simplified:

$$\sigma_{id} = \frac{1}{W_f}\sqrt{M_f^2 + M_t^2}$$

Thus the flexural modulus is equal to:

$$W_f = \frac{1}{\sigma_{id}}\sqrt{M_f^2 + M_t^2}$$

The flexural modulus for a solid shaft with circular section is equal to:

$$W_f = \frac{\pi D^3}{32}$$

The greatest admissible stress σ_{adm} was computed dividing the ultimate tensile strength UTS by a safety factor sf:

$$\sigma_{adm} = \frac{UTS}{sf}$$

Hence the minimum admissible diameter in the shaft was computed as:

$$D_{min} = \left(\frac{32}{\pi \sigma_{adm}} \sqrt{M_f^2 + M_t^2} \right)$$

The shaft is made out of C40 steel, whose ultimate tensile strength UTS is $500\,MPa$.
The maximum torque generated by the motor is τ_{max}^m.
The bending moment M_f is null where the shaft has the smallest diameter. With a safety factor sf equal to 5, the minimum admissible diameter is $5.5\,mm$.
Hence the diameter is set to $6\,mm$ to take into account the slots due to the feather keys.

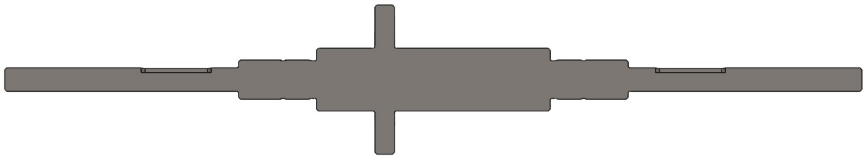
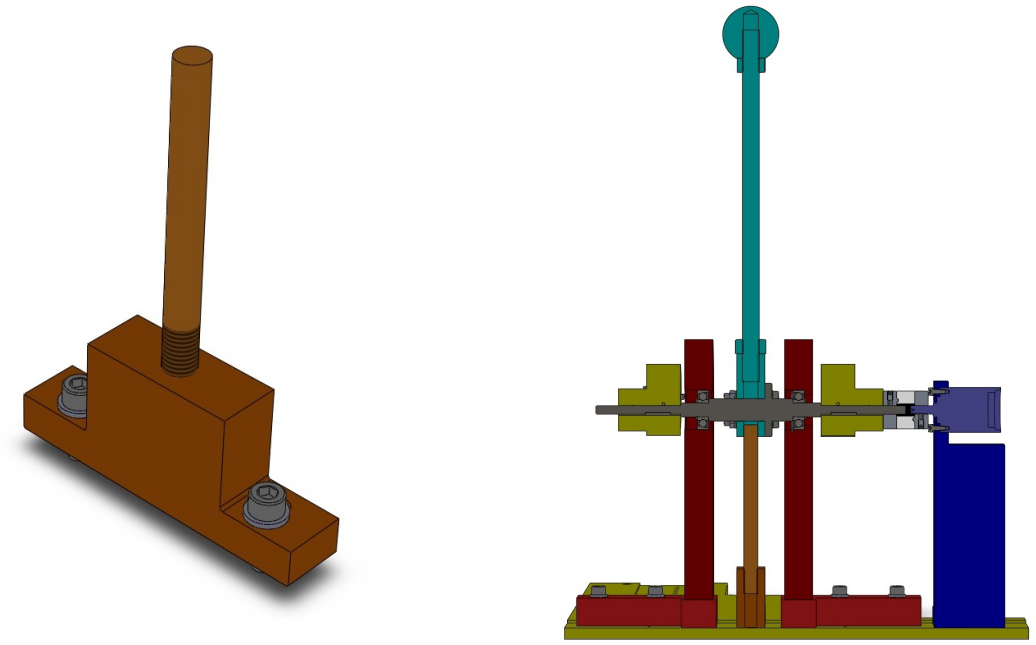


Figure 4.6: Sectional view of the shaft.

The shaft is connected to the electric motor with an elastic joint in order to absorb vibration and correcting eventual misalignment of the axes of the shaft and electric motor. Furthermore, the elastic joint could operate as fuse in case of failure or overload in the transmission.

4.1.6 Reference angle system

One of the features added to the test bench was a system to lock the pendulum in the vertical position, so that the encoder used to measure the angular position of the shaft could be calibrated.
The system, denominated *reference angle system*, consists in a rod that couples to the hole drilled in the base of the pendulum (Figure 4.7b). The bottom part of the rod is threaded so that it can be moved up and down.
The threaded coupling added functionality, but had the disadvantage of adding some clearance, too. However, even if a perfect vertical position is not ensured, the misalignment resulted to be small enough to be neglected.

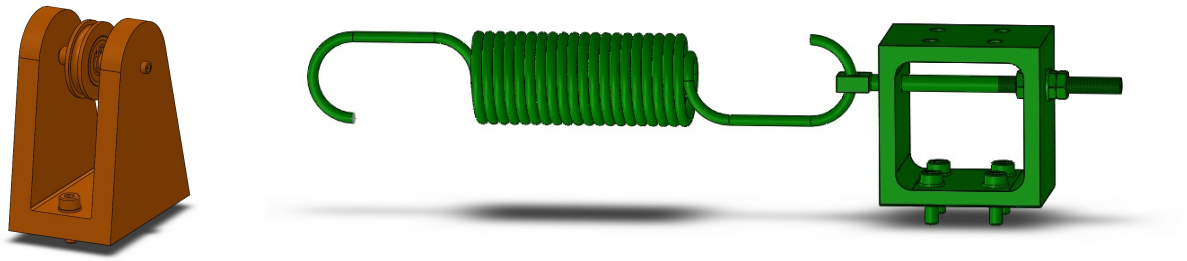


(a) (b)

Figure 4.7: (a) Assembly view of the reference angle system. (b) Sectional view of the test bench.

4.1.7 Routing pulleys

The routing pulleys are composed by a support and a standard circular pulley with ball bearings to decrease energy losses due to friction.
In case the radius of the routing pulley was to be modified, the pulley could be easily disassembled and substituted by a similar standard component.
The insertion length could be regulated by adding spacers beneath the support of the pulley.



(a) (b)

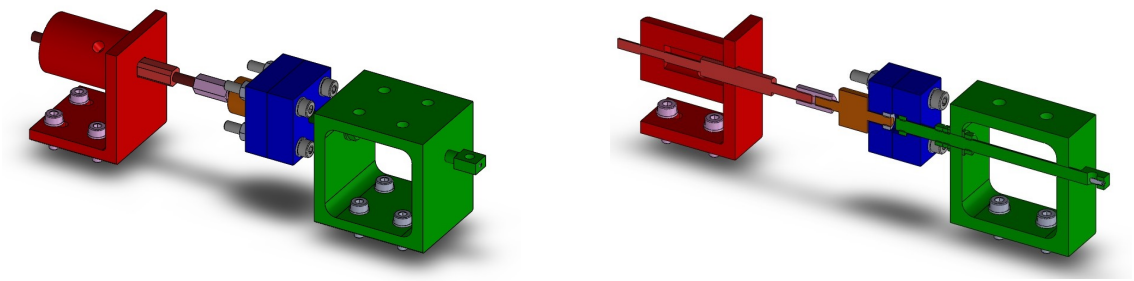
Figure 4.8: (a) Assembly view of the routing pulley. (b) Assembly view of the spring holder.

4.1.8 Spring holder

The components designated to secure the spring to the ground are a rod with an eyelet to attach the axial spring and an hollow parallelepiped to holds the rod.
The upper surface of the parallelepiped was drilled to facilitate the use of the Allen key to mount the piece.
The rod can move backwards and forwards and is locked by nuts. The thread on the rod also allows to attach the mechanism to measure the preload of the spring.
An additional locking nut was added to lock the nut that opposes the force of the spring, after that the operation to regulation of the preload of the spring has been executed.

4.1.9 Preload system

The system to determine and regulate the preload of the spring was composed by a load cell and a mechanism to move backwards and forwards the rod of the spring holder.
The load cell used was the 8417 by *burster*, as it was the only one available in the laboratory. Between the load cell and the rod of the spring holder there is an adapter to connect the two components with different diameters. This was needed because the load cell was selected after the manufacturing of the test bench, so its diameter was unknown at the time of the design. Hence the adapter was 3D printed in a second phase.
The load cell is connected on the other side to a second rod through a long nut. The rod is threaded on the extremes and squared in the middle. The thread on the load cell side is long enough to completely house the nut and free the load cell. This feature is relevant when assembling the mechanism.
On the other side, the rod is screwed to a cylinder with a hole along its axis. A portion of the hole is threaded to couple with the thread of the rod, while the rest was enlarged to house the squared section of the rod.
A hole perpendicular to the axis of the cylinder was added to account for the eventual use of a lever to rotate the cylinder, if a bigger torque was needed.
The support of the rod has a squared hole that couple with the squared section of the rod, preventing it from rotating. In this way, the rod can be moved backwards and forwards by rotating the cylinder.
One preload system is sufficient to tune the preload of both springs, as they can be regulated one at a time.



(a) (b)

Figure 4.9: Assembly and sectional view of the preload system.

4.1.10 Base

The base was made out of aluminium to keep the weight down so that the test bench could be lifted by hand.
The central sector of the base is occupied by a groove that guides the axial alignment of the motor, the shaft holders and the reference angle system. In the groove there are threaded holes to fix the aforementioned components and the two routing pulleys.
On the sides there are two other grooves for a correct assembly of the preload mechanism and spring support. The numerous equidistant threaded holes were made to mount the devices at different distances from the shaft, so that the test bench could employ other springs, too.
In addition the base was provided with eight holes to eventually fix the test bench to the ground (two of which are highlighted in red in Figure 4.10), and eight threaded holes to eventually mount proximity sensors to signal a particular angle of the pendulum to the control system (four of which are highlighted in green).

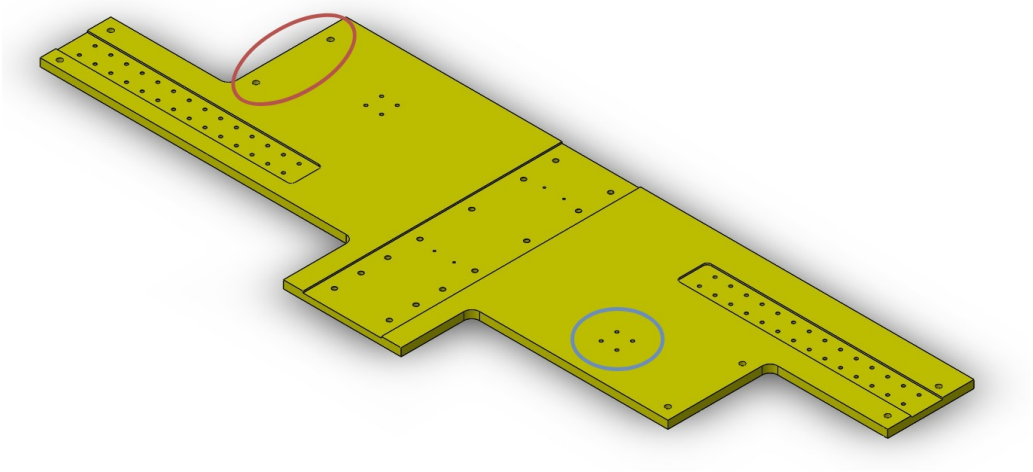


Figure 4.10: 3D model of the base.

4.2 Noncircular pulley profile

Before assembling the test bench, the shaft and all the components that were to be mounted on it were weighted. These values were used to compute the overall mass, the centre of mass and the moments of inertia of the rotating components with the CAD software. These three values were used to synthesise the profile of the noncircular pulley and to characterise the mass properties of the pendulum in the virtual prototype.
For the same reasons, the diameter of the routing pulley, the thickness of the cable and the length of the spring at rest were measured. Also the distance between the routing pulley and the shaft was checked, provisionally assembling the relative components.
Finally, an attempt at evaluating the stiffness of the spring was made, measuring its elongation with different loads applied. However the equipment available in the laboratory did not have the precision required for a valid evaluation, and in conclusion the datasheet value was used.
The values required to synthesise the noncircular profile are all reported in Table 4.4.

Table 4.4: Input parameters to synthesise the noncircular profile.

Mass of the shaft assembly	$900\,g$
Distance of the centre of mass from the axis	$90.1\,mm$
Spring stiffness	$k_s = 410\,N/m$
Spring elongation at rest	$x_0 = 65\,mm$
Insertion length	$L = 108\,mm$
Angular range	$\theta \in [30°, 150°]$
Parameters for the antagonistic subtorques	$\gamma = 0.5,\ d_{off} = 0.55\tau_d(0)$
Routing pulley's radius	$r_p = 6.5\,mm$
Cable thickness	$t_c = 1.5\,mm$
Routing configuration	(a)

After this operations, the noncircular pulley profile could be synthesised (Figure 5.1). Selecting the correct combination of parameters was crucial to realise a functional profile. Particular attention was paid to obtaining a pulley with a short profile, so that the elongation required to the spring was limited. In this way, the spring can maintain a linear behaviour across the entire range.

Ensuring that the noncircular profile would not interfere with the shaft was important, too. Eventually, the problem could be solved with a different design that locates the noncircular pulley on the ground and not on the shaft.
At last, the profile was used to create and print the 3D model of the noncircular pulley.

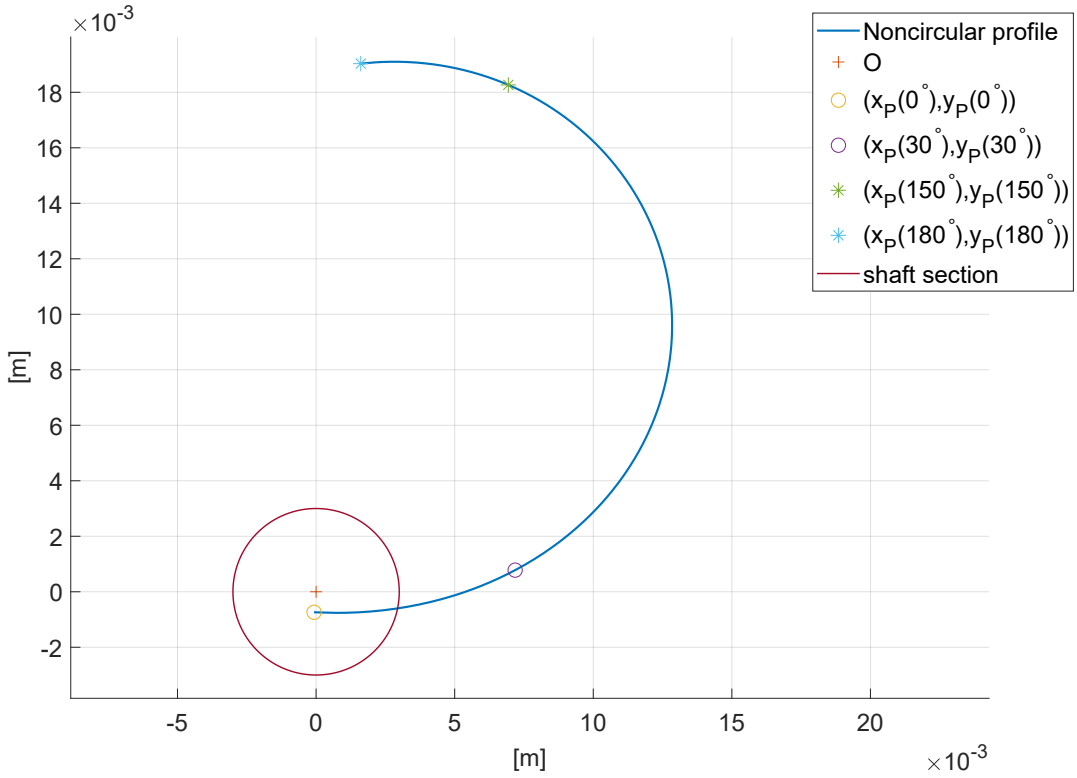


Figure 4.11: Synthesised profile for the noncircular pulley.

4.3 Assembly

The sequence followed to assemble the test bench was:

1. The pendulum was assembled and connected to the shaft with the four bolts. Then, with the help of a press, the ball bearings were mounted on the shaft and locked with the Seeger rings. (Figure 4.12a)

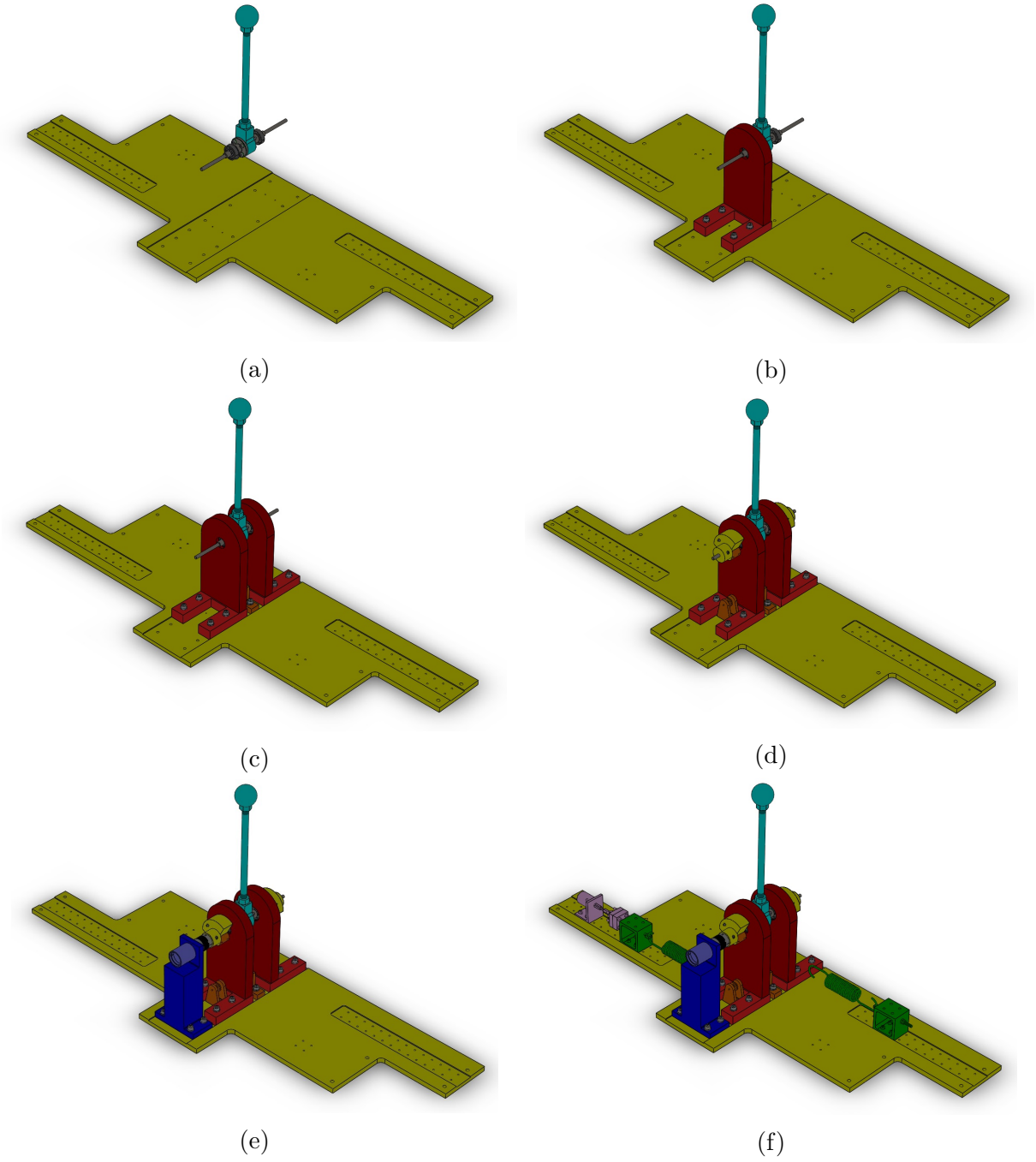
2. The shaft support that axially locks the bearing was assembled and mounted on the base with four bolts. The ball bearings was locked into the support between the shoulder and a Seeger ring. (Figure 4.12b)

3. The second shaft support was assembled. After having inserted the shaft bearing into the designated hole in the support, the distance from the first support was adjusted and then the second support was fixed with four bolts.
   Also the reference angle system was assembled so that is could be used to prevent the pendulum from rotating to facilitate the operations that followed. (Figure 4.12c)

4. The threaded inserts were heated and pressed into the designated holes of the non-circular pulleys, which were then mounted onto the shaft using the feather key and the grub screws.
   The two routing pulleys were assembled and mounted onto the base. (Figure 4.12d)

5. The motor was locked into its support. Then the elastic joint was used to connect shaft and electric motor, and the motor support was fixed to the base with four screws. (Figure 4.12e)

6. The spring supports were mounted onto the base, and the cables were connected to the springs and the noncircular pulleys, passing through the routing pulleys. Finally, the pretension system was used to adjust the preload of each of the two springs. (Figure 4.12f)



(a) (b) (c) (d) (e) (f)

Figure 4.12

Chapter 5

Preliminary tests

5.1 Adams simulation

All the parameters measured or computed in Section 4.2 to synthesise the noncircular profile were used to update the parameters of the virtual prototype. Moreover, the noncircular profile synthesised for the test bench was used to create a simpler 3D model of the noncircular pulley to be added to the virtual model.
Table C.1 reports the values of all the parameters needed to develop the virtual model of the test bench; while the macros used to create the model are all reported in Appendix B. The main reference frame of the model is located at the base of the pendulum (Figure 5.1).
In this particular configurations, it is remarkable how the two long cable elements, with no contacts with neither pulley, succeeded in significantly decreasing the number of elements needed to model the cable (Figure 5.1).

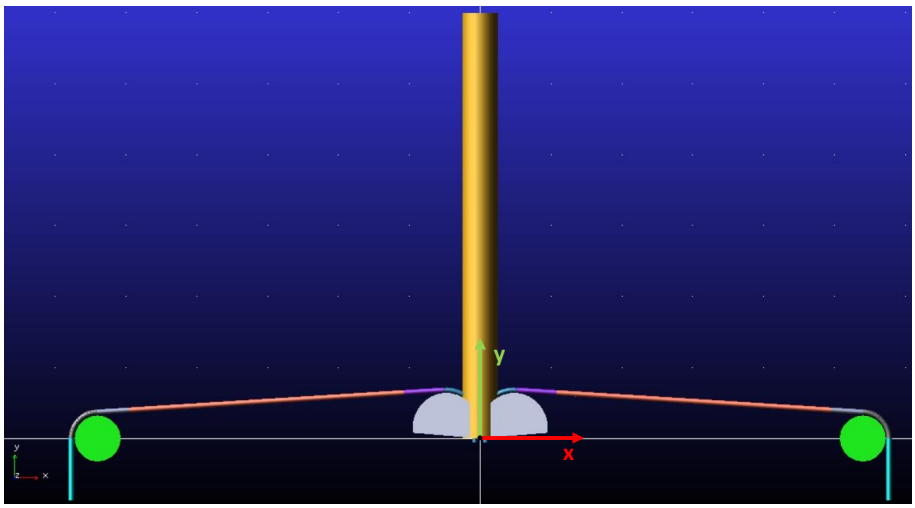


Figure 5.1: Updated model of the test bench.

Table 5.1: Parameters of the final virtual prototype

Pendulum parameters	
Mass of pendulum	$900\,g$
Inertia of pendulum	$\begin{bmatrix} 21513.87 & \cdot & \cdot \\ 0 & 1652.36 & \cdot \\ 86.04 & 0 & 20039.01 \end{bmatrix}\ kg\,mm$
Coordinates of CM of pendulum	$\begin{bmatrix} 0 & 90.06 & 7.45 \end{bmatrix}\ mm$
Routing pulley parameters	
Radius of routing pulleys	$6.5\,mm$
Inertia of routing pulleys	$\begin{bmatrix} i_{xx} = 0.07 & i_{yy} = 0.07 & i_{zz} = 0.13 \end{bmatrix}\ kg\,mm$
Increased radius of routing pulley	$(6.5 + 1.5/2 + 0.2)\,mm$
Cable parameters	
Cable diameter	$1.5\,mm$
Density of cable	$7800\,kg/m^3$
Coefficient of damping elements	$0.005\,kg/s$
Spring parameters	
Stiffness	$410\,N/m$
Damping	$0.005\,kg/s$
Elongation at rest	$65\,mm$
Initial preload	$30.781\,N$
Other parameters to synthesise the noncircular profile	
Angular range	$\theta \in [30°, 150°]$
Insertion length	$108\,mm$
Parameters of the antagonistic subtorques	$\gamma = 0.5,\ d_{off} = 0.55\tau_d(0)$
Routing configuration	(a)
Gravity	$-9.807\,m/s$
Other parameters to arrange the cable elements	
Increased radius for the noncircular pulley	$(r_{nonc} + 1.5/2 + 0.05)\,mm$
n_{nonc}	60
n_{circ}	12
Contact parameters	
Damping	$1.0 \cdot 10^4\,kg/s$
Stiffness	$1.0 \cdot 10^5\,N/mm^2$
μ_s	0.99
μ_d	$0.90\,m$
Force exponent	2.2
Penetration depth	$0.1\,mm$
Solver parameters	
Solver integrator	HHT
Solver formulation	I3
Solver error	$1.0 \cdot 10^{-5}$
Simulation step size	$0.001\,s$

In order to show the behaviour of the model, a simple simulation, with a motion imposed to the pendulum, was performed. The motion, applied to the revolute joint connecting the pendulum to the ground, constraint the angular velocity of the pendulum to track the following profile:

```
"4.0d*(STEP(time, 2, 0 ,2.5 , 1) + STEP(time, 12.0, 0, 12.5, -1)) +
-40.0d*(STEP(time, 15.0, 0,15.1, 1) + STEP(time, 16.9, 0, 17.0, -1))"
```

The end time of the simulation was set at 20 s.

5.1.1 Results exemplifying Adams simulation

Angular position and velocity

Following the motion imposed to the joint, the pendulum remained motionless for 2 s, so that the cables had time to settle on the pulleys. Then it rotated anticlockwise up to 130° (Figure 5.4a) at an angular speed of 4 deg/s, stayed motionless for 2.5 s and rotated clockwise for 2 s at angular speed of 40 deg/s (Figure 5.4b).
The pendulum accelerated four times to reach the given velocity, as reported in Figure 5.3b.

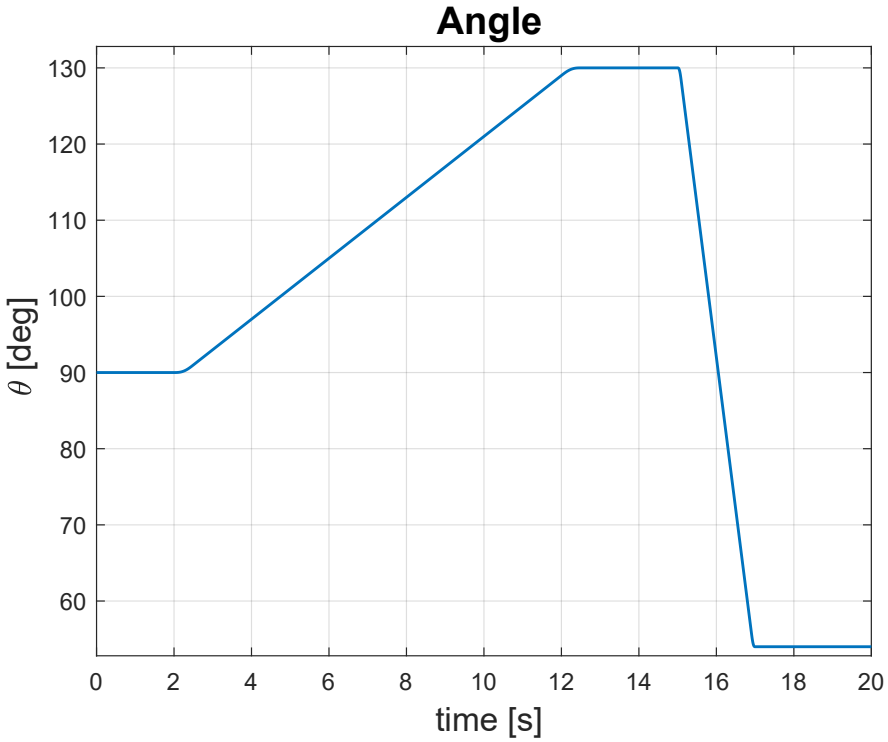
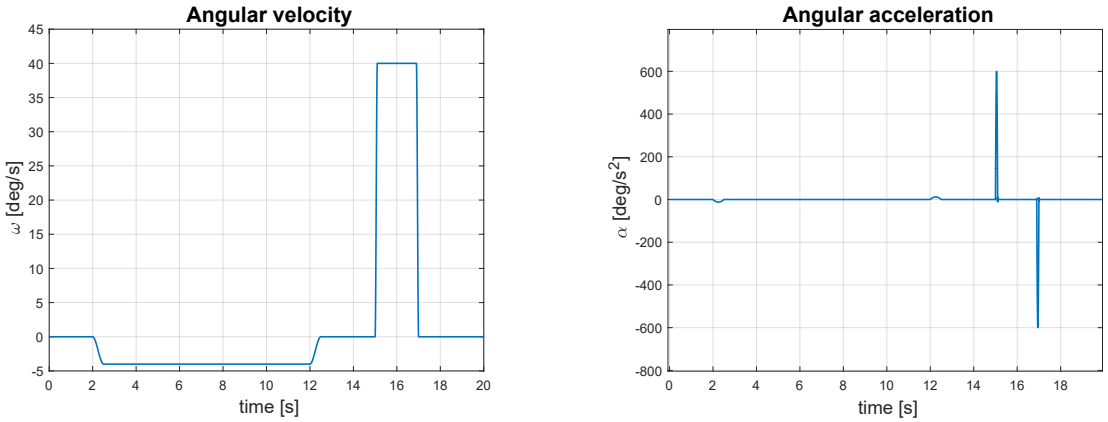
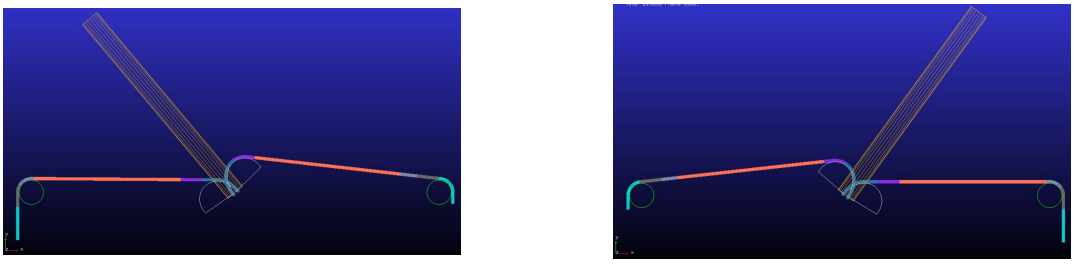


Figure 5.2: Angular position of the pendulum.



(a) (b)

Figure 5.3: Angular velocity and acceleration of the pendulum.

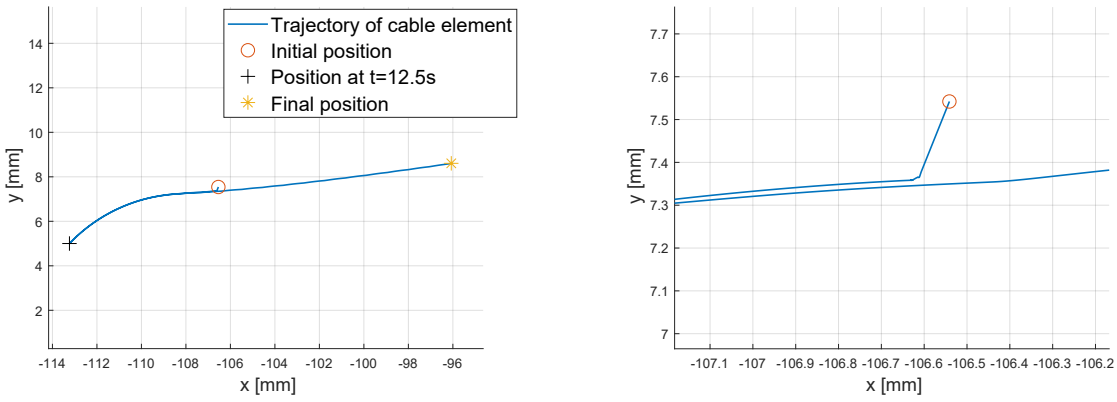


(a) (b)

Figure 5.4: Extreme orientation reached by the pendulum during the simulation.

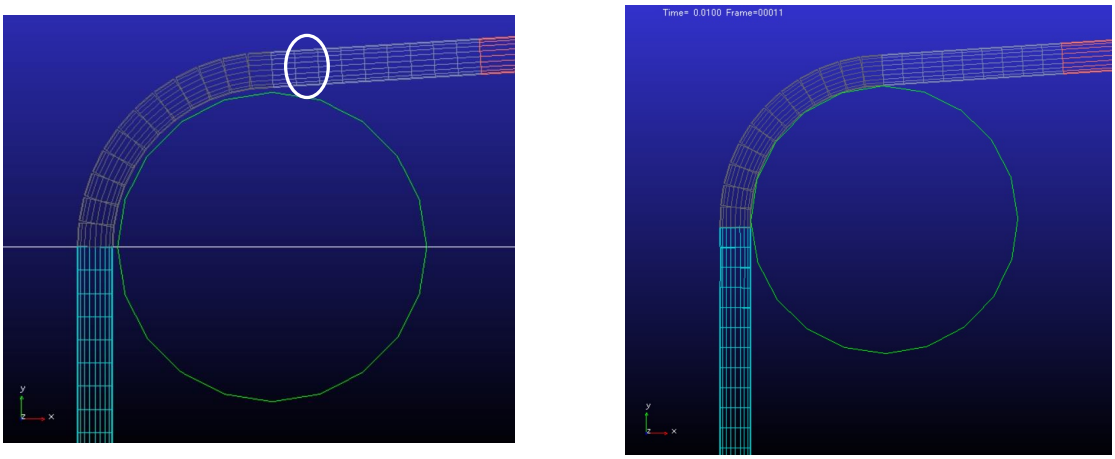
Cable elements and routing pulleys

Figure 5.5a shows the trajectory in the xy-plane of a cable element close to the circular pulley (highlighted in Figure 5.6a). In the enlargement of Figure 5.5b it is possible to observe the initial settling movement of the cable due to the gap with the routing and noncircular pulleys. Figure 5.7 reports the contact force between the same cable element and the routing pulley. The irregularities in the profile were likely due to the discretisation of the cable that caused small vibrations in the cable.



(a) (b)

Figure 5.5: Trajectory of a cable element during the simulation.



(a) (b)

Figure 5.6: Particular of the arrangement of the cable on the routing pulley at the beginning of the simulation (a) and after 0.01 s (b).

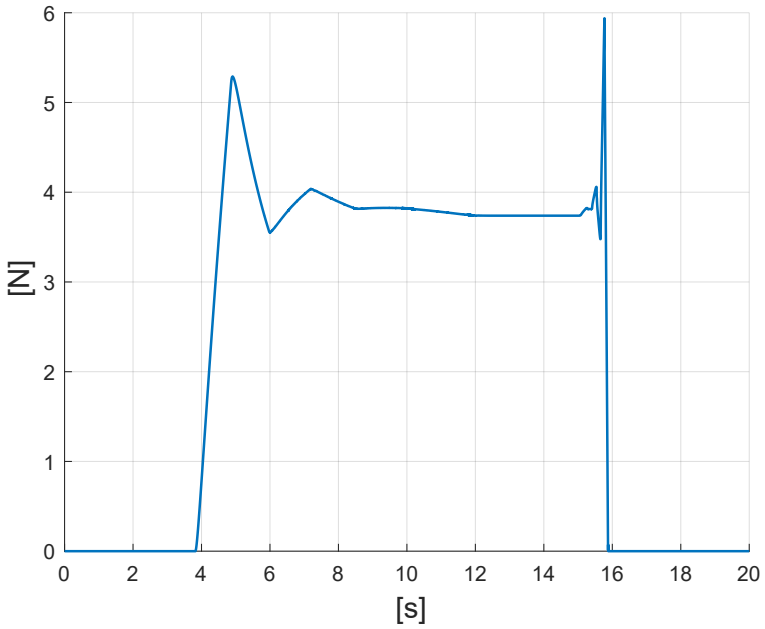


Figure 5.7: Contact force between the cable element highlighted in Figure 5.6 and the routing pulley.

Spring force

The force generated by the springs (Figure 5.8) was characterized by an initial settling of the preload value, as the cables enter in contact with the pulleys.
The desired value of the preload was 30.781 N, while the value assumed after the initial movement was 30.859 N. The difference is small enough to be negligible, as also the preload of the springs in the physical prototype could not have the exact desired value.
However, by fine-tuning the position of the two points connecting the springs to the ground, it would be possible to correct the initial discrepancy of the preload value.
Another peculiarity of the plot in Figure 5.8 is that the two spring forces have similar profiles, but not symmetric values, as the noncircular pulley causes one of the springs to

stretch more than the other, depending on if the angle is greater or smaller than 90°.

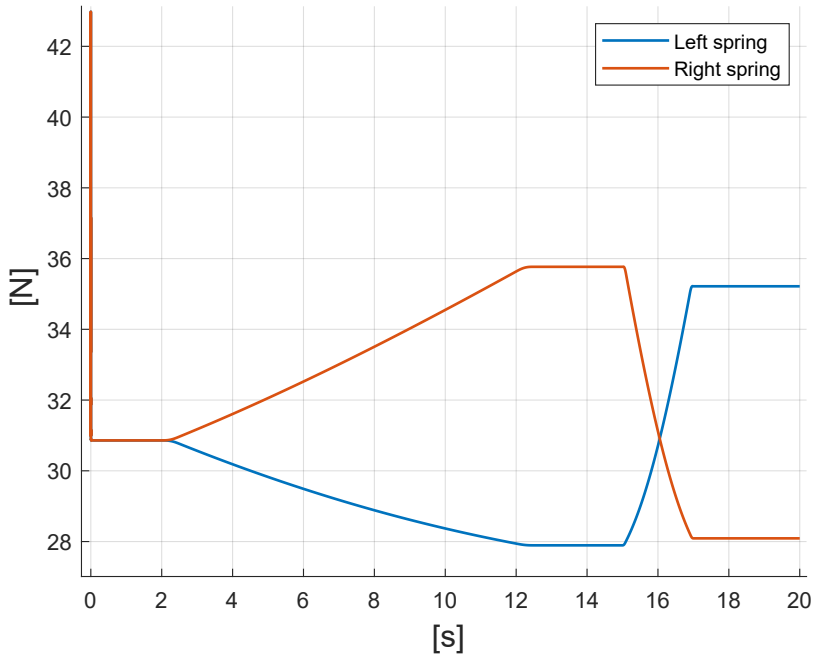


Figure 5.8: Spring forces.

Torque

The most interesting value of the simulation to analyse was the torque required to rotate the pendulum (Figure 5.9).

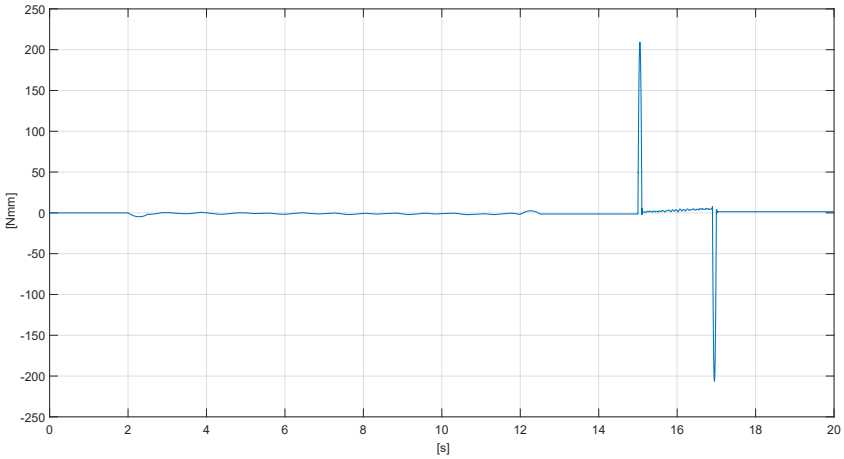


Figure 5.9: Torque generated to rotate the pendulum.

The four peaks at around 2, 12, 15, 17 s are due to the torque needed to accelerate and decelerate the pendulum. In fact they are similar to the peaks in the plot reporting the acceleration of the pendulum (Figure 5.3b), with the first two peakes being smaller since they are related to smaller accelerations.

Between the peaks, from 2 to 12 s and from 15 to 17 s, the pendulum was rotating at a constant speed, hence the torque on the joint is caused solely to gravity and not to inertial accelerations. In these segments the gravity compensator manages to decrease the torque caused by gravity, almost annihilating it.
The vibrations present in the plot were likely caused by the discretisation of the two cable, as the contact on the pulleys is not as smooth as with a continuous cable. It is probable that also the fact that the movement of the pendulum was constraint by the motion contributed to generate the vibrations, as the system could not move smoothly, but it was forced to follow a specific velocity profile.

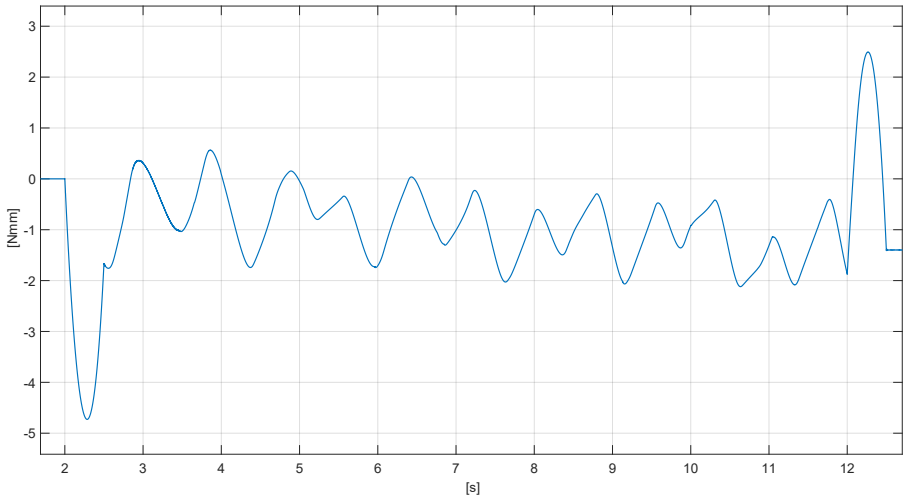


Figure 5.10: Enlargement of the torque profile from 2 to 13 s.

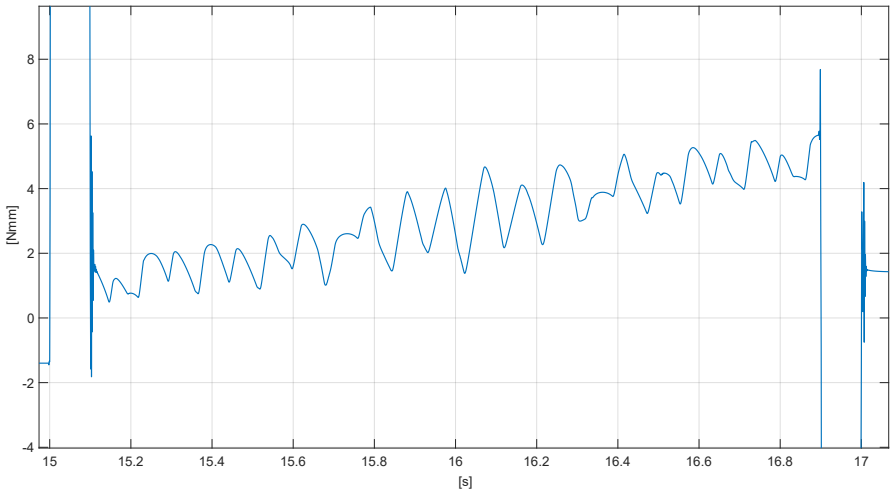


Figure 5.11: Enlargement of the torque profile from 15 to 17 s.

Comparison with models without gravity compensator

In order to better analyse the effect of the gravity compensator, a new model of the cable without the gravity compensator was created (Figure 5.12). It was then used to perform two simulations with the same motion previously imposed to the virtual prototype: in the

first simulation the gravity field was left active to analyse the performances of the pendulum without gravity compensator, while in the other the gravity field was deactivated in order to analyse the behaviour of the pendulum with a perfect gravity compensator.

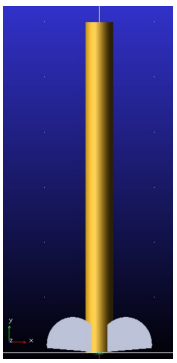


Figure 5.12: Model of the pendulum without the gravity compensator.

Figure 5.13 reports the torque applied to the pendulum to make it follow the imposed motion. It is possible to observe how the gravity compensator significantly reduces the torque needed by the pendulum and has almost the same performances of the case without the gravity field.
As expected, the peaks due to the accelerations are present and are equivalent in all three cases. Excluding those peaks, the maximum torque applied in the case without the gravity compensator is 510.8 Nmm, measured when the pendulum was stopped at 130°. The maximum value of torque applied in the case with the gravity compensator was 5.6 Nmm, again excluding the the peaks of torque due to the accelerations. All in all, it means that using the gravity compensator maximum torque needed by the pendulum was decreased by 98.9%, which is very significant.
Furthermore, the plot shows how at 130° the system with gravity compensator needed a torque of just 1.4 Nmm to hold the position, while without the gravity compensator the maximum torque (510.8 Nmm) needed to be applied to the pendulum.

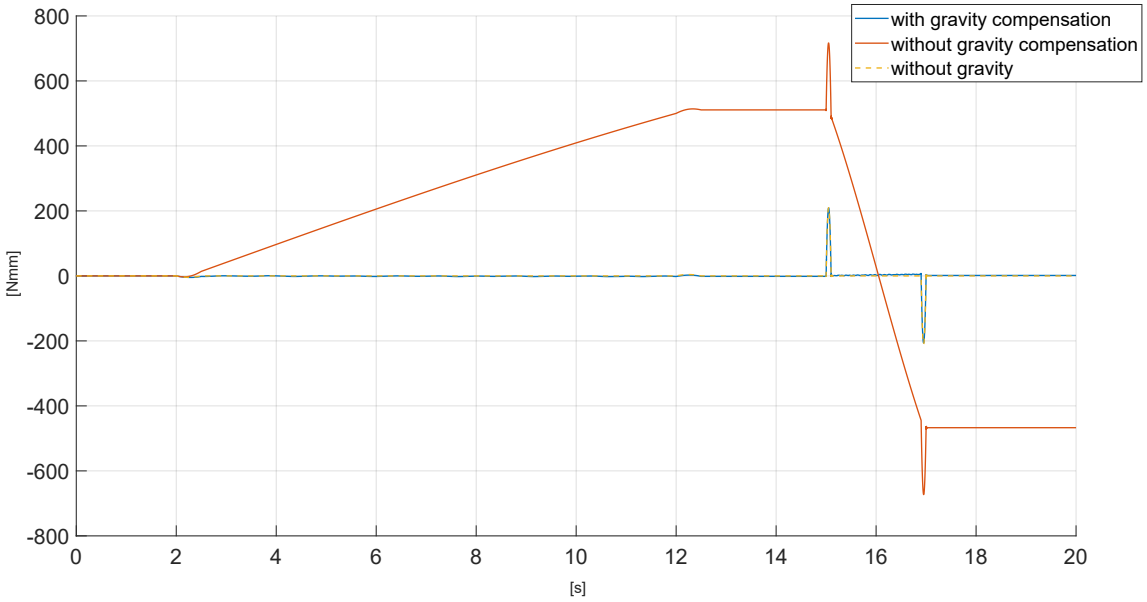


Figure 5.13: Comparison of the torque generated in the three simulations.

Figures 5.14 and 5.15 report the torque in the cases with the gravity compensator and the one without gravity, in the two segments of time when the pendulum is rotating. Apart from the vibrations most likely due to the discretisation of the cable, it is possible to note how the torque with the gravity compensator tended to linearly move away from the theoretical profile of the case without gravity. One possible cause could be that the initial preload of the springs was not the imposed one, as mentioned earlier. The initial error would linearly increase its effect while the pendulum was rotating, as the torque is generated by the spring force multiplied the moment arm of the noncircular pulley. More simulations to investigate this particular issue are planned to be performed in future works.

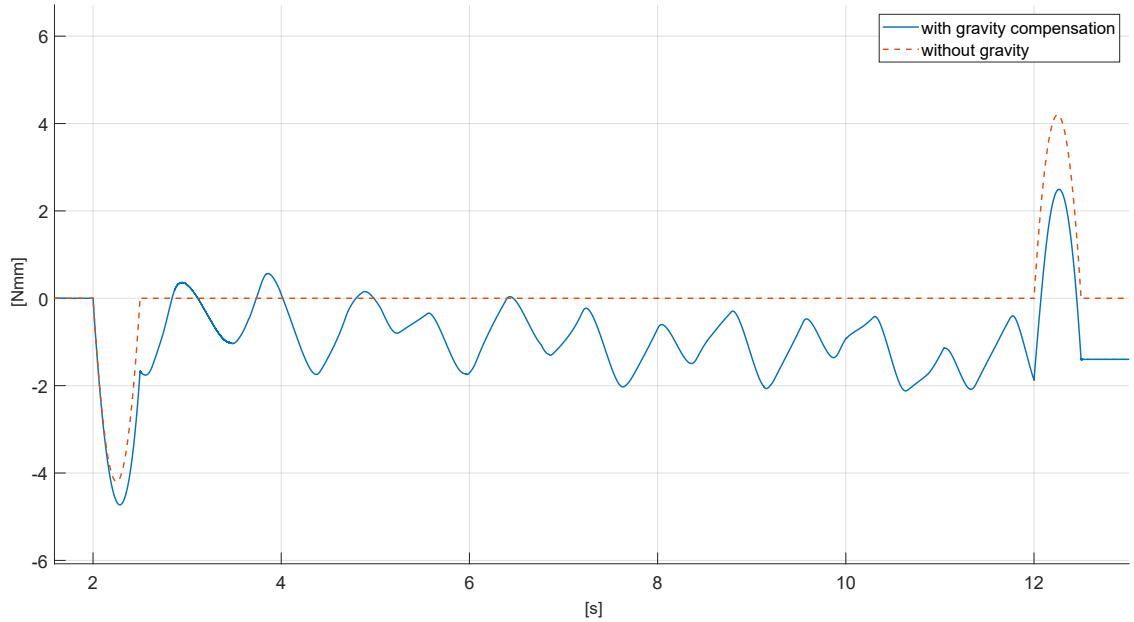


Figure 5.14: Enlargement of the torque profiles from 2 to 13 s.

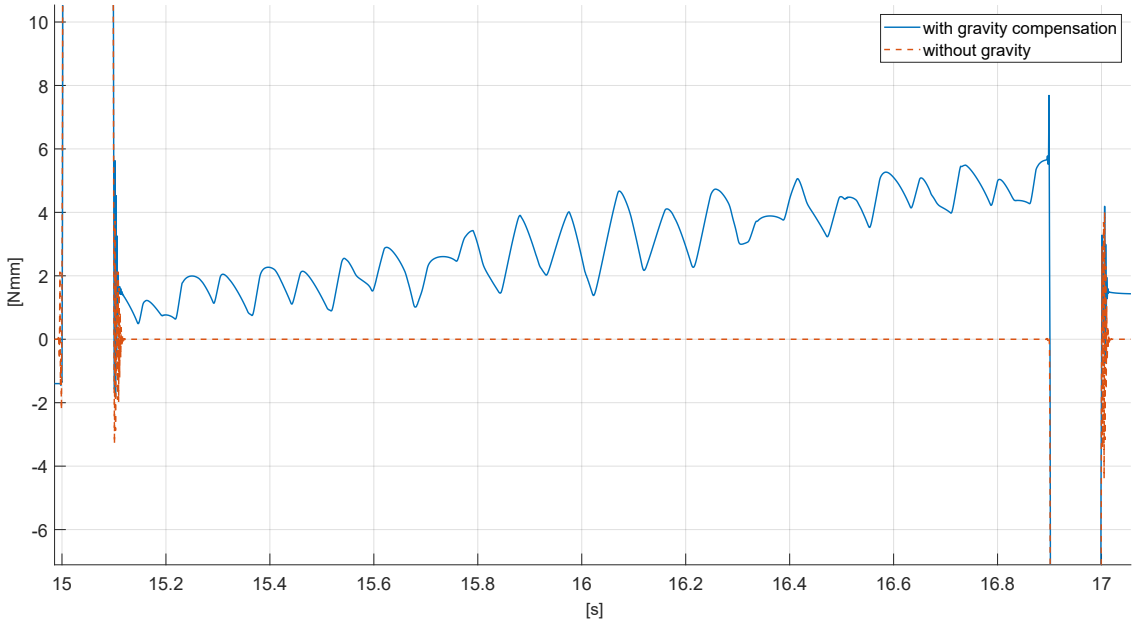


Figure 5.15: Enlargement of the torque profiles from 15 to 17 s.

5.2 Tests on the physical prototype

After assembling the test bench as described in Section 4.3, the prototype was prepared for the first tests.
First of all, the load cell was calibrated by weighting several weights. Afterwards, the preload system was assembled and used to regulate the preload of the two springs with the same value used in the simulations ($30.87\,N$). During this operation, particular attention was paid at carefully tightening both the nuts locking the spring supports, without moving the rod and thus varying the value of the preload. Finally, after that the locking nut was tighten and the preload system was disassembled, the test bench preparation was completed.

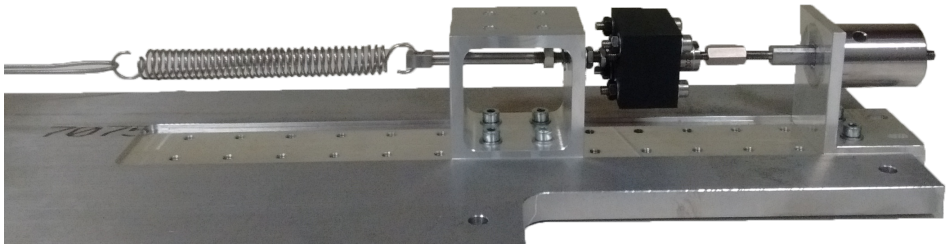


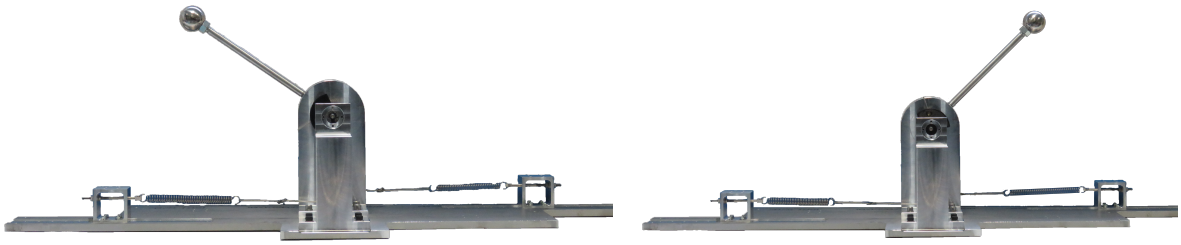
Figure 5.16: Preload system.

These preliminary tests had the aim of checking the general setup of the experiments and collect the first results to perform a first comparison with the virtual prototype.

5.2.1 Static test

This first test consisted in evaluating the behaviour of the test bench without the assistance of the electric motor. For this reason, the motor was disconnected from the shaft by removing the elastic joint.
Initially the pendulum would tended to lean on one of the sides, most likely due to a discrepancy in the preload values of the two springs. So the preload system was reassembled on one of the spring supports, and the spring was regulated so that the pendulum would stand still in the vertical position, without checking the value of the preload force. Then the spring support was locked and the preload system removed.
After that adjustment, the pendulum started to perform as expected: it was set at different angles within the $30° \div 150°$ range and remained balanced in the set positions, with the exception of some small oscillatory movements of a few degrees.
This test gave a first positive feedback on the performances of the gravity compensator, which resulted functioning but not perfect. Moreover, it highlighted the importance of having the same preload tension in the two springs in order to successfully balance the pendulum.
The not perfect regulation of the preload was very likely caused by the use of a load cell without the correct range and precision, and by the reduced dimensions of the preload system that made the operation difficult to perform.
Another factor in the behaviour of the system was the friction presents in the springs,

cables and ball bearings, which had a positive effect on the system: it assisted the gravity compensator in contrasting the torque due to gravity, and allowed to even out the errors due to a not perfect preload. Nevertheless, friction has to be limited, as it potentially increases the energy losses of the system.



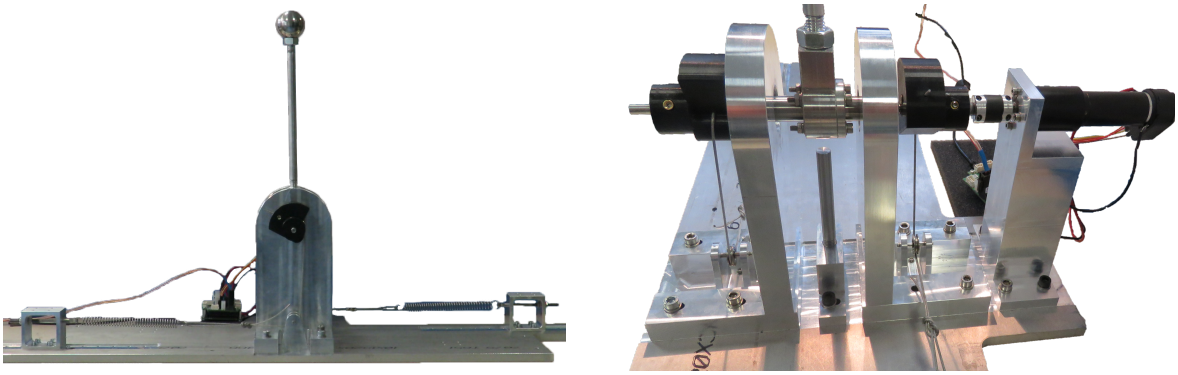
(a) (b)

Figure 5.17: Test bench during the static test.

5.2.2 Dynamic test

After the conclusion of the static test, the electric motor was reconnected to the shaft via the elastic joint to perform a dynamic test, aimed at evaluating the torque required to rotate the pendulum.
The test consisted in rotating the pendulum from 45° to 90° at three different average angular speeds: $\omega_1 = 7\,deg/s$, $\omega_2 = 50\,deg/s$, $\omega_3 = 120\,deg/s$. The test was repeated in two different configurations: one with the gravity compensator on, and the other with the gravity compensator being detached from the shaft by disconnecting the springs from the cables.

Following the test, the virtual model was used to perform three sets of simulations. In each set of simulations, a motion replicating the rotation of the pendulum in the real-world test was imposed to the model, in the configuration with gravity compensator, without gravity compensator and with the gravity field deactivated, as in the simulations of Section 5.1.1.



(a) (b)

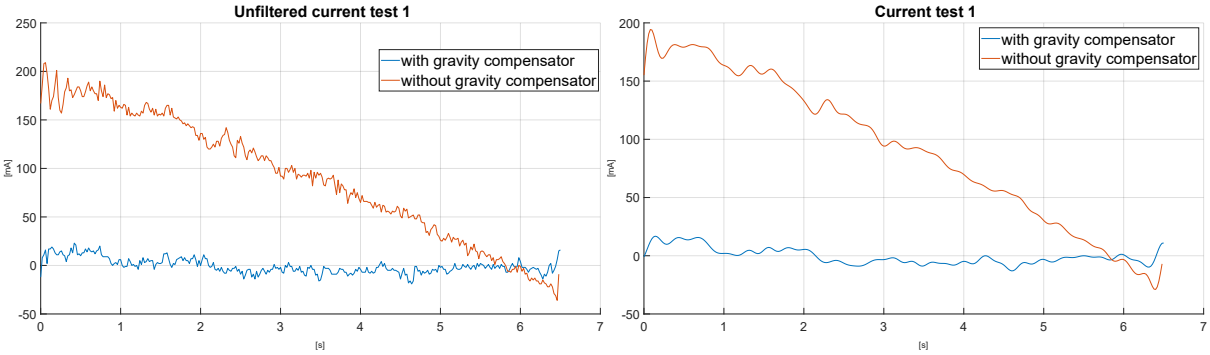
Figure 5.18: Test bench during the dynamic test.

The electric motor was controlled using the EPOS4 50/8 Positioning Controller by "*Maxon motor Spa*".

The current absorbed by the electric motor, the angular velocity and position of the pendulum were collected by the EPOS4 driver. After the test, the recorded signal of the current $i(t)$ was filtered with a LPF to clean it from the noise (Figure 5.19).
The filtered values of the current $i_f(t)$ were then used to estimate the torque $\tau(t)$ applied on the shaft to rotate the pendulum as:

$$\tau(t) = \eta_{gear} \cdot \eta_m \cdot k_{gear} \cdot k_t \cdot i_f(t) \tag{5.1}$$

where η_{gear} and η_m are the datasheet maximum efficiency of the gearbox and electric motor respectively, k_{gear} is the reduction ratio of the gearbox, k_t is the torque constant of the motor.



(a) (b)

Figure 5.19: Unfiltered (a) and filtered (b) current absorbed by the electric motor at ω_1.

Figures 5.20, 5.21 and 5.22 report the plots of the evaluate torques. In all of the three cases, the torque generated by the motor is smaller when the the gravity compensator is used.
Table 5.2 reports the maximum and average value of the torque in each of the cases. Table 5.3 reports the percentage decrease of the maximum and average torque by using the gravity compensator.

Table 5.2: Maximum and average torque generated by the motor.

	Maximum torque [Nmm]	Average torque [Nmm]
$\omega_1 = 7\,deg/s$, with compensator	48	16.1
$\omega_1 = 7\,deg/s$, without compensator	551	243.0
$\omega_2 = 50\,deg/s$, with compensator	155	71.9
$\omega_2 = 50\,deg/s$, without compensator	746	238.9
$\omega_3 = 120\,deg/s$, with compensator	1200	498.2
$\omega_3 = 120\,deg/s$, without compensator	1503	690.9

Table 5.3: Percentage reduction in the torque generated by the motor.

	Reduction in maximum torque	Reduction in average torque
case 1	91.3%	93.4%
case 2	79.2%	69.9%
case 3	20.2%	27.9%

These values are a sign of how advantageous a gravity compensator could be, especially at low velocities. In fact with the increase of the angular velocity, the torque generated by the actuator to accelerate and decelerate the system is more and more important, while the gravity torque becomes less of a factor, as anticipated in Section 1.1.
The oscillations present in the signals were likely due to the controller that accelerated and decelerated the pendulum to track the given signal. They could be decreased by fine-tuning the parameters of the controller.

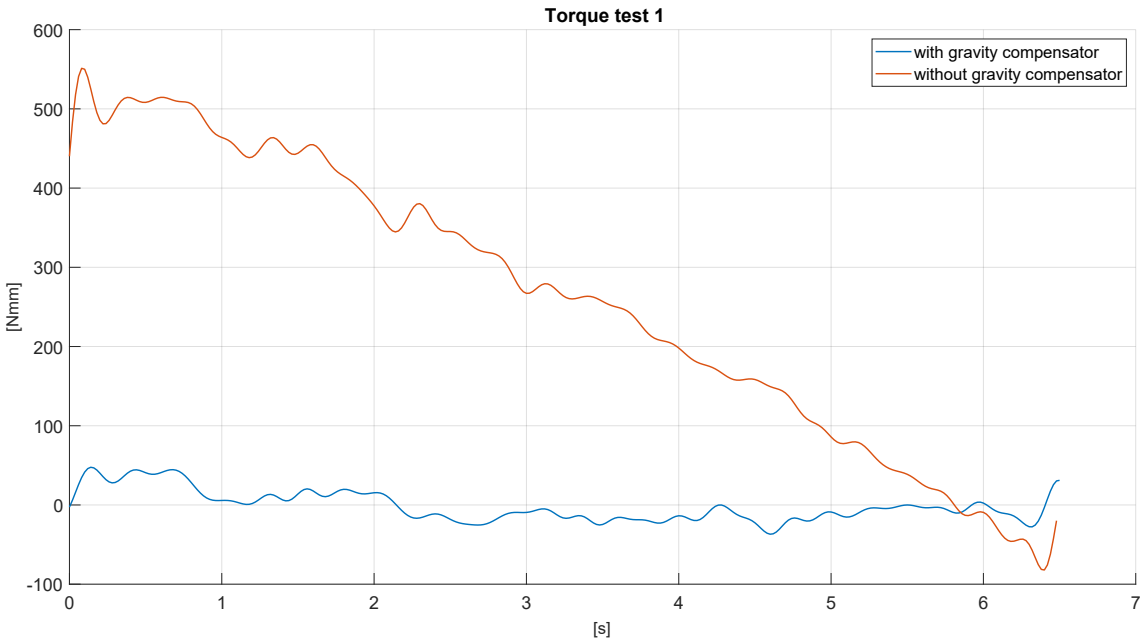


Figure 5.20: Torque absorbed at ω_1,

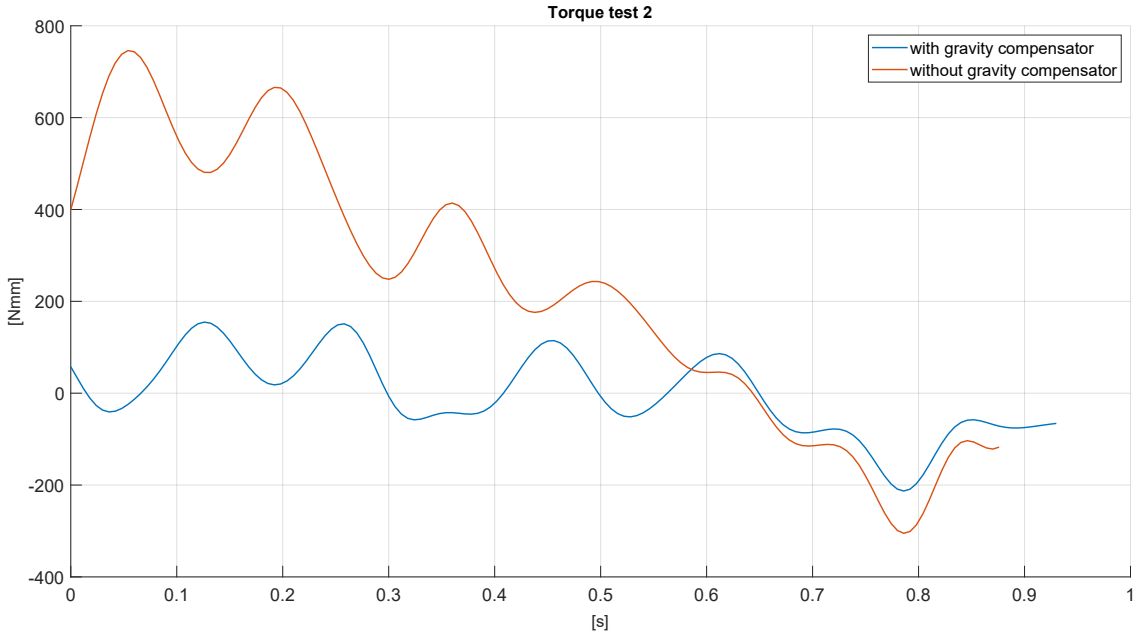


Figure 5.21: Torque absorbed at ω_2,

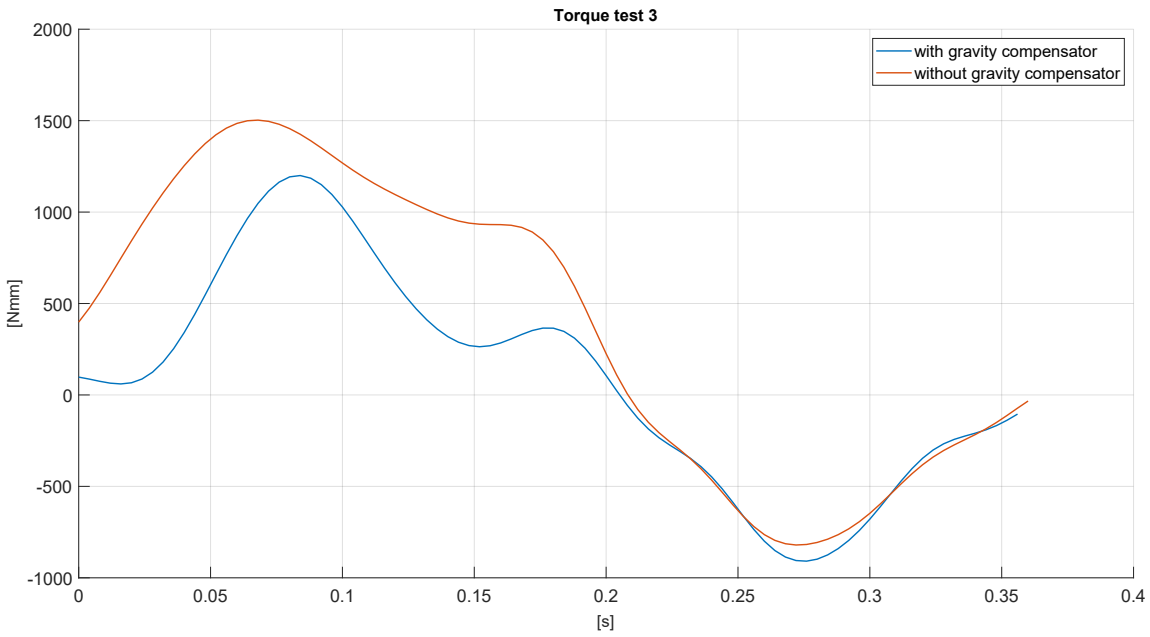


Figure 5.22: Torque absorbed at ω_3,

The angular position and velocity recorded values were used to create three equivalent motions for the virtual model.
Since the rotation performed in the test started from 45°, the motions necessitated to include an initial rotation from 90° to 45° and a small settling time to let the vibrations disappear.
The three computed motions were:

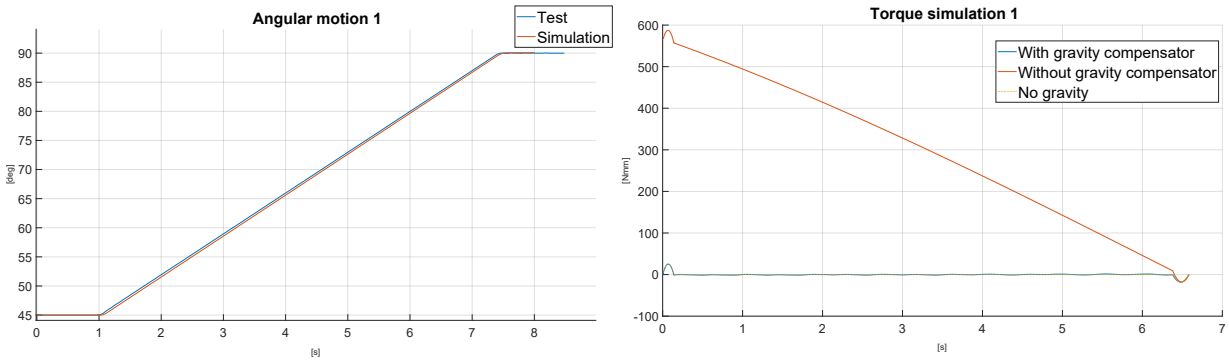
1. ```
   "-18d*(STEP(time, 1.99, 0, 2.00, 1) +
    + STEP(time, 4.49, 0, 4.50, -1)) +
    + 7.03d*(STEP(time, 6.500, 0, 6.640, 1) +
    + STEP(time, 12.880, 0, 13.080, -1))"
   ```

2. ```
   "-45.0d*(STEP(time, 2.0, 0,4.0, 1) +
    + STEP(time, 6.000, 0, 6.882, -1))"
   ```

3. ```
   "-45.0d*(STEP(time, 2.0, 0,4.0, 1) +
    + STEP5(time, 6.000, 0, 6.376, -1))"
   ```

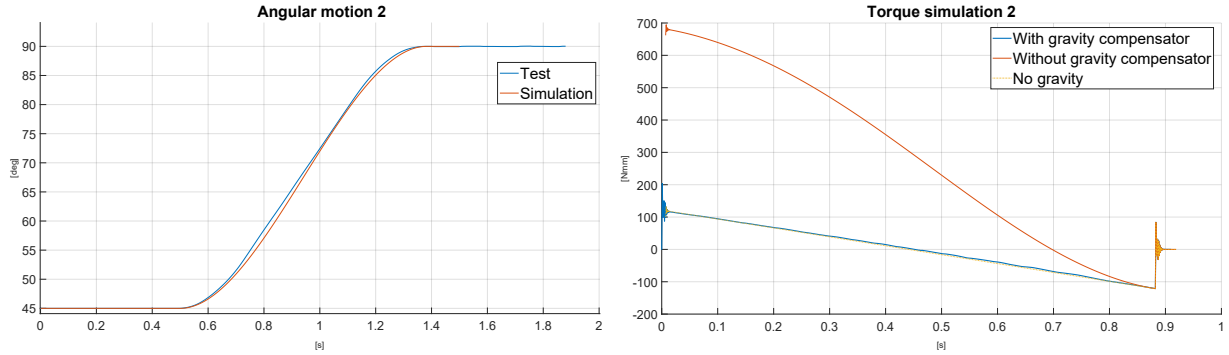
The first motion was imposed to the angular velocity of the pendulum, while the other two were imposed to its angular position.

Figures 5.23a, 5.24a and 5.25a report the comparison of the angle of the pendulum of the dynamic test and the corresponding simulation, only in the segment of time when the pendulum rotates from 45° to 90°.
The simulations performed as expected, in particular the ones with the gravity compensator gave almost the same results of the ones without gravity, as it was the case in Section 5.1.1.
   ```



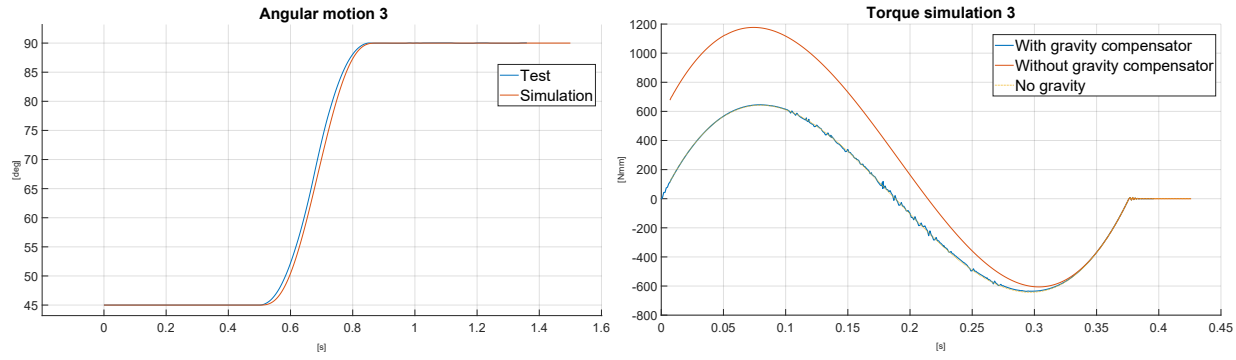
(a) (b)

Figure 5.23: (a) Comparison between the angle of the pendulum in the test and in the simulations. (b) Torques of the simulations. Both for ω_1.



(a) (b)

Figure 5.24: (a) Comparison between the angle of the pendulum in the test and in the simulations. (b) Torques of the simulations. Both for ω_2.



(a) (b)

Figure 5.25: (a) Comparison between the angle of the pendulum in the test and in the simulations. (b) Torques of the simulations. Both for ω_3.

Figures 5.26, 5.27 and 5.28 report the plot comparing the results of the real-world tests and the simulations.
In all the three figures it is possible to see the initial acceleration and final deceleration, which are present also in the cases with no gravity field. In the real-world cases, these peaks are bigger and wider, as the controller took more time to modulate the signal.

Excluding the oscillations due to the controller, the results obtained were very promising, as the profiles of the real-world tests and the simulations had the same trends and the results had similar values; and especially considering all the limitations concerning the tests.

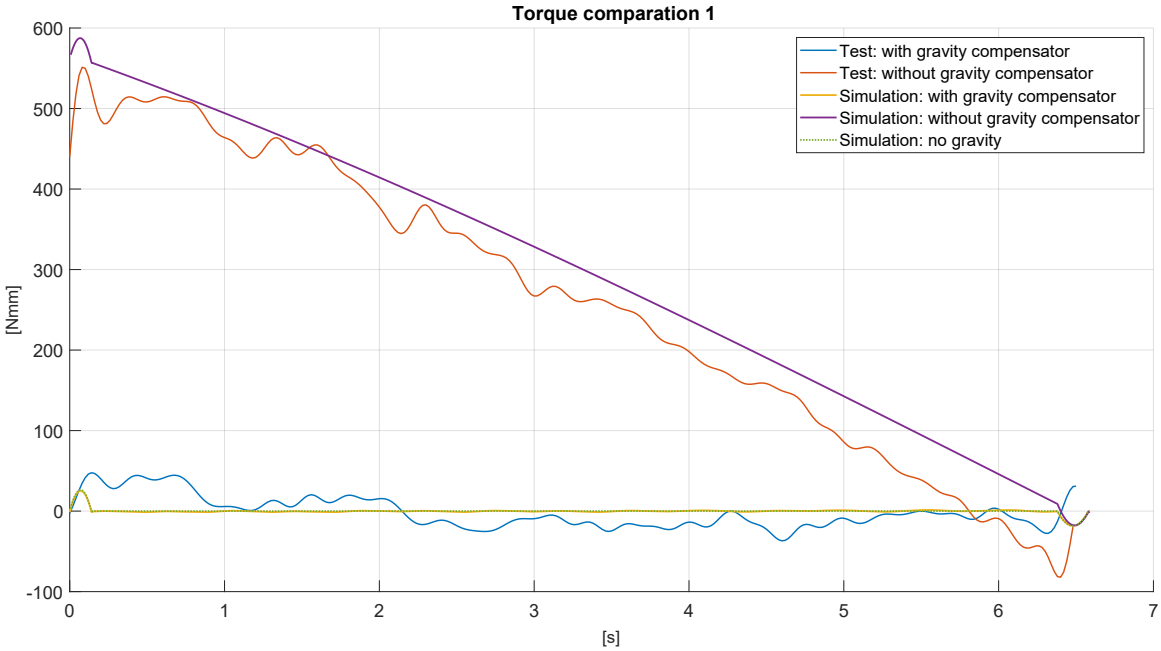


Figure 5.26: Comparison of the torque profiles at ω_1.

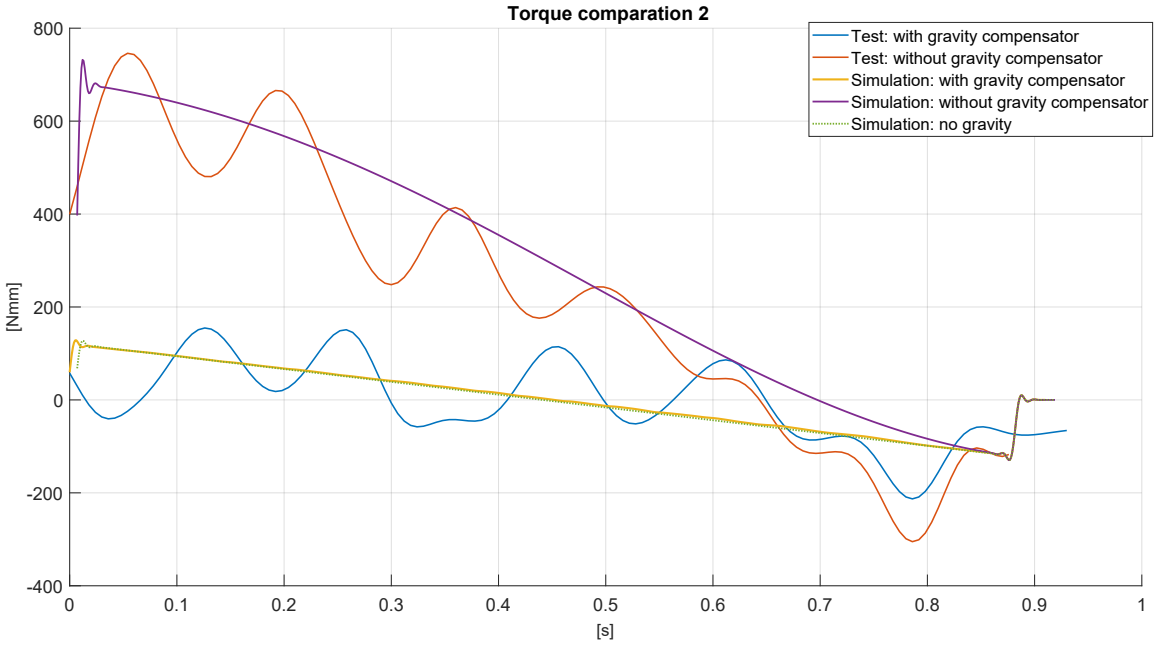


Figure 5.27: Comparison of the torque profiles at ω_2.

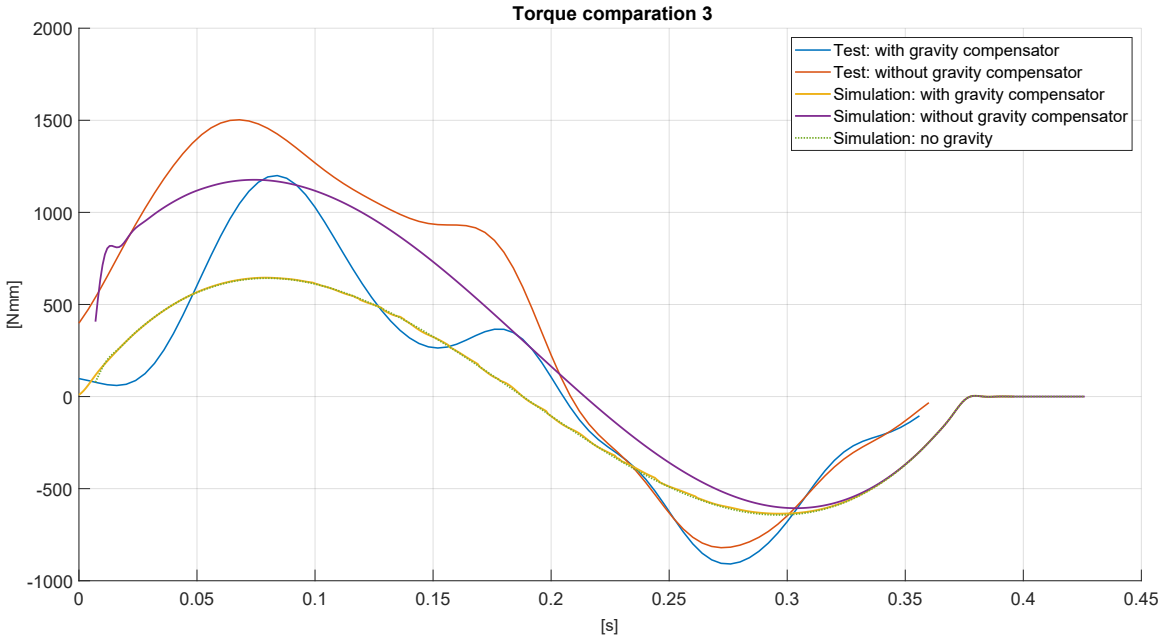


Figure 5.28: Comparison of the torque profiles at ω_3.

Some of the aforementioned limitations could be solved through a better modelling of the components in the virtual prototype, while other were due to not ideal preparation of the physical test bench for the tests and the way they were carried out.

Regarding the virtual prototype, a model of the electric motor and the controller had yet to be developed and used instead of the motion. Possible ways to model the two components are either using the designated machinery suite in Adams, or by developing the model of the controller and electric motor with MATLAB and Simulink and exporting the Adams model into Simulinks.
In addition, friction in the revolute joints of the pendulum and the two routing pulleys had not modelled neither.
Two other possible improvements to the virtual model would be to model the springs as elastic components, and not simply as ideal forces; and substituting the ideal revolute joints present in the cable, in the pendulum and in the routing pulleys with bushing elements, which takes into account the elasticity and damping of a real revolute joint.
All of these modifications would draw the simulations closer the real-world system, but require a careful identification of every parameter using the results of the tests on the test bench.

The limitations of the tests due to inaccuracies in the set-up of the test-bench regarded in particular the preload of the spring, which proved to be incorrect and had to be adjusted already during the static test. Also the losses electric motor should have been better characterised via specific tests and not simply using the datasheet estimated efficiency.
Another factor that should be better analysed in the future would be the precision with which the noncircular profile was manufactured, as at the time of the tests it was yet to be investigated. To this end, the procedure to evaluate the torque generated by a noncircular profile pulley, developed in Section 2.7, could be used.
Finally, more repetitions of the tests should be performed to limit the influence of random uncertainties in the results.

Chapter 6

Conclusions

Nowadays the interest towards more energy-efficient and safer solutions for robotic manipulators is ever increasing, as these are key aspects required to let them interact with human workers in non-confined environments. The introduction of passive mechanisms that compensate the joint torque due to the weight of the robot has proved to be a valid solution to these demands.
In this work the gravity compensator for revolute joints proposed by [1] was examined. This particular mechanism is able to generate an arbitrary torque profile through the action of a spring and a noncircular pulley. Therefore it has the advantage of not being limited to compensate exclusively the trigonometric torque profiles due to gravity on revolute joints.
In order to test the benefits of this gravity compensator, a virtual model of the mechanism applied to an inverted pendulum and a physical test rig of the same system were realised.

The first part of the work is focused on analysing the procedures introduced by [1] to synthesise the noncircular profile of the pulley, create an antagonist spring configuration to generate a bilateral torque, and evaluate the torque generated by a given noncircular pulley. In addition, a new method to synthesise the noncircular profile that exploits convex optimization is proposed, together with an improvement in the procedure to evaluate a noncircular profile, that takes into account also some practical issues related to cable routing. These methods are applied to synthesise the profiles for an exemplary 1 DoF and a 3 DoF systems.
Then, the procedure to develop the virtual model of the noncircular pulley-spring mechanism applied to an inverted pendulum using the software MSC Adams® is reported. The model was created using macros written in Adams/View Command Language, as it proved to be the most effective way to model the cable-noncircular pulley interaction.
The simulations computed with the virtual model allowed to confirm the theory behind the noncircular pulley-spring gravity compensator and analyse the different dynamic behaviours of the pendulum with and without the gravity compensator.
After the realisation of the virtual model, the design and assembly of the physical test bench to test the effects of the gravity compensator is described. A system to regulate the pretension of the springs was included.
Afterwards, the results of the preliminary tests performed on the test rig are reported. They consisted in measuring the torque generated by an electric motor to actuate the

pendulum in two different configurations: one with the gravity compensator acting on the system and the other where the compensator was disconnected. During the tests, the gravity compensator managed to decrease both the average and maximum torque generated by the motor by up to 90%.
In conclusion, the data collected from the test rig were compared to the ones from the simulations of the virtual model. The results were very promising, as they have similar values and highlighted the same dynamic behaviours.

This work succeeded in showing how the introduction of a gravity compensator mechanism could be an effective solution to reduce the energy consumption of a robot and the size of its actuators. But its main merits were to realise two valid and functioning instruments to test the noncircular pulley-spring mechanism proposed by [1], in the virtual model and the test rig.
However, the work still requires to be finalised by performing a proper validation of the virtual model using the test rig. In addition, other improvements could be brought to both the virtual model and the test rig, e.g. the realisation of the models of the electric motor and control system, and a better system to regulate the preload of the springs in the test rig.
Eventually the studies could be deepen by recreating, using the test rig, a case where a more peculiar torque profile has to be compensated, to fully exploit the advantages of the noncircular pulley-spring mechanism. Furthermore, the virtual model could be applied to study the effects of the compensator on more complex system, possibly with multiple DoF systems.
All of these researches could lead to employ this particular compensator in very different fields: from optimising the efficiency of robotic manipulators in executing particulars routines in production lines, to the manufacture of passive exoskeletons that are required to compensate torques with unconventional profiles.

Appendix A

MATLAB scripts to synthesise the noncircular profile

A.1 Example 1

A.1.1 Main

```
clear variables
close all
clc

%% Design Parameters
g = 9.81;

% Pendulum:
m = 3;
l = 1;
k = 5e3;        % spring constant
u_0 = 0.022;    % spring initial elongation
L = 0.1;        % spring insertion point
% motion range:
th_0 = 0;
th_1 = pi;

%% Desired Torque
syms th real;
t_max = m*g*l;
t_D(th) = t_max*cos(th);                % Torque to balanced due to the mass

t_D0 = double(t_D(th_0));
t_U(th) = 0.5*t_D+0.55*t_D0;            % t_L(th) + t_U(th) = t_D(th)
t_L(th) = 0.5*t_D-0.55*t_D0;            % t_L(th),t_U(th): the system uses 2 opposite springs, with initial pretension
t_d(th) = 0.5*t_D(pi-th)+0.55*t_D0;     % Torque profile to be balanced by pulley1

%t_d2(th) = -t_L;      % Torque profile to be balanced by pulley2 (equal to t_d(th), therefore the pulley will be simmetric to pulley1)

%% Profile

[rm,phi,Y,S,x_P,y_P] = analytical_pulley_profile(k,u_0,L,t_d,th,[th_0,th_1]);
save('profile','rm','x_P','y_P');

%% Plot
% Desire torque
figure

hold on
fplot(t_D,[-pi,2*pi],'LineWidth',1.25)
fplot(t_U,[-pi,2*pi])
fplot(t_L,[-pi,2*pi])
fplot(t_d,[-pi,2*pi],'--','LineWidth',1.25)
plot([th_0,th_0,th_0,th_0],[t_D(th_0),t_U(th_0),t_L(th_0),t_d(th_0)],'o')
plot([th_1,th_1,th_1,th_1],[t_D(th_1),t_U(th_1),t_L(th_1),t_d(th_1)],'*')
grid on
axis auto
title('Torque profile')
legend('t_d(\theta)','t_U(\theta)','t_L(\theta)','t_d1(\theta) \equiv t_d2(\theta)','t(\theta_0)','t(\theta_1)')
xlabel('\theta [rad]')
ylabel('Torque [Nm]')
```

```
52 set(gca,'Color','w')
```

A.1.2 Function to synthesise the noncircular profile

```
1  function [rm,phi,Y,S,x_P,y_P] = analytical_pulley_profile(k,u_0,L,t_d,th,theta)
2
3  % Function to generate the profile of an noncircular pulley designed to
4  % conunterbalance a given torque profile.
5
6  % INPUTS
7  % k: spring constant
8  % u_0: spring initial elongation
9  % L: spring insertion point
10 % t_d: torque profile to compensate (symfun)
11 % th: sybolic variable
12 % theta: motion range [th_0;th_1]
13
14 % OUTPUTS
15 % rm(th): radius of the pulley
16 % phi(th)
17 % Y(th)
18 % S(th): slope of the the tangent to the pulley at [x_P(th);y_P(th)]
19 % x_P(th): x_coordinate of the pulley profile
20 % y_P(th): y_coordinate of the pulley profile
21
22     th_0 = theta(1);
23     th_1 = theta(2);
24
25     %% Profile
26
27     rm(th)  = t_d/(2*k*int(t_d)+k^2*u_0^2)^(1/2);
28     phi(th) = asin(rm/L);
29     Y(th) = L*(sin(th)-cos(th)*tan(th+phi));
30     S(th) = tan(th+phi);
31     x_P(th) = -diff(Y,th)/diff(S,th);
32     y_P(th) = S*x_P+Y;
33
34     %% Plot
35
36     figure
37     % Torque profile
38     subplot(1,3,1)
39     grid on
40     hold on
41     fplot(t_d,[-pi,2*pi])
42     plot([th_0,th_1],[t_d(th_0),t_d(th_1)],'*')
43     axis auto
44     title('t_d')
45     xlabel('\theta [rad]')
46     ylabel('Torque [Nm]')
47
48     % Radius over theta
49     subplot(1,3,2)
50     hold on
51     fplot(rm,[-pi,2*pi])
52     plot([th_0,th_1],[rm(th_0),rm(th_1)],'*')
53     grid on
54     title('Radius of non-circular pulley')
55     legend('r_m')
56     xlabel('\theta [rad]')
57     ylabel('Radius [m]')
58
59     % Profile in xy-plane
60     subplot(1,3,3)
61     hold on
62     grid on
63     fplot(x_P,y_P,[th_0,th_1])
64     plot(0,0,'+')
65     plot(x_P(th_0),y_P(th_0),'o')
66     plot(x_P(th_1),y_P(th_1),'*')
67     grid on
68     title('Shape of non-circular pulley')
69     xlabel('[m]')
70     ylabel('[m]')
71     axis equal
72
73 end
```

A.1.3 Function to evaluate the profile as proposed by [1]

```
function t_a = check_profile_torque_num(k,u_0,L,th,x_P,y_P,range,N)

% Function to compute the torque profile generated by the given pulley
% profile, in order to evaluate the quality of the design.

% INPUTS
% k: spring constant
% u_0: spring initial elongation
% L: spring insertion point
% th: symbolic variable
% x_P(th): x_coordinate of the pulley profile
% y_P(th): y_coordinate of the pulley profile
% range: motion range [th_0;th_1]
% N: number of samples used to discretize

% OUTPUTS
% t_a: vector containing the torque profile generated by the pulley profile (x_P(th);y_P(th))

    th_0 = range(1);
    th_1 = range(2);

    x = linspace(th_0,th_1,N)';
    incr = (th_1-th_0)/N;

    % Function used to compute the derivative
    x_P1(th) = x_P(th-incr);
    x_P2(th) = x_P(th+incr);
    y_P1(th) = y_P(th-incr);
    y_P2(th) = y_P(th+incr);

    % Discretization of the functions
    x_P = double(x_P(x));
    y_P = double(y_P(x));
    x_P1 = double(x_P1(x));
    y_P1 = double(y_P1(x));
    x_P2 = double(x_P2(x));
    y_P2 = double(y_P2(x));

    % trajectory of the insertion point (R)
    x_R = L*cos(x);
    y_R = L*sin(x);

    RP = sqrt((x_R-x_P).^2+(y_R-y_P).^2);     % distance between insertion point (R) and tangency point
      on the pulley (P)
    ra = abs(x_R.*y_P-y_R.*x_P)./(((x_P-x_R).^2+(y_P-y_R).^2)).^(1/2);    % distance between centre of
     pulley (O) and tangency point (P)

    %% lenght of wire wrapped around the pulley

    % Derivatives of x_P and y_P
    xPd_1 = (x_P-x_P1)/incr;
    xPd_2 = (x_P2-x_P)/incr;
    xPd = (xPd_1+xPd_2)/2;
    yPd_1 = (y_P-y_P1)/incr;
    yPd_2 = (y_P2-y_P)/incr;
    yPd = (yPd_1+yPd_2)/2;
    L1 = (xPd.^2+yPd.^2).^(1/2);

    % Integral of L1
    L_s = [L1(1)*incr];
    for i = 2:N
        L_s(i,1) = L1(i)*incr+L_s(i-1);
    end

    u = L_s+RP-RP(1)+u_0;     % elongation of the spring
    t_a = k.*u.*ra;           % torque produced by the pulley-spring system

end
```

A.2 Example 2

A.2.1 Function to synthesise the noncircular profile taking into account the geometrical corrections

```
function [rm,phi,Y,S,x_P,y_P] = analytical_pulley_profile_geom_cor(k,u_0,L,t_d,th,theta,geom_corr)

% Function to generate the profile of an noncircular pulley designed to
% conunterbalance a given torque profile.  This function takes into
% account the geometrical corrections meade due to the rope thickness
% and the fact that the insertion point is on a circular pulley.

% INPUTS
% k: spring constant
% u_0: spring initial elongation
% L: spring insertion point
% t_d: torque profile to compensate (symfun)
% th: sybolic variable
% theta: motion range [th_0;th_1]
% geom_corr: [tc,rp,cb]
%           tc = cable thickness
%           rp = pulley radius
%           case of cable routing: case(a) = -1; case(b) = 1

% OUTPUTS
% rm(th): radius of the pulley
% phi(th)
% Y(th)
% S(th): slope of the the tangent to the pulley at [x_P(th);y_P(th)]
% x_P(th): x_coordinate of the pulley profile
% y_P(th): y_coordinate of the pulley profile

    th_0 = theta(1);
    th_1 = theta(2);
    tc = geom_corr(1);
    rp = geom_corr(2);
    cb = geom_corr(3);

    %% Profile
    rm(th)  = t_d/(2*k*int(t_d)+k^2*u_0^2)^(1/2);
    phi(th) = asin((rm+cb*(0.5*tc+rp))/L);
    Y(th) = L*(rm-0.5*tc)/(rm+cb*(0.5*tc+rp))*(sin(th)-cos(th)*tan(th+phi));
    S(th) = tan(th+phi);
    x_P(th) = -diff(Y,th)/diff(S,th);
    y_P(th) = S*x_P+Y;

    %% Plot
    figure

    % Torque profile
    subplot(1,3,1)
    grid on
    hold on
    fplot(t_d,[-pi,2*pi])
    plot([th_0,th_1],[t_d(th_0),t_d(th_1)],'*')
    axis auto
    title('t_d')
    xlabel('\theta [rad]')
    ylabel('Torque [Nm]')

    % Radius over theta
    subplot(1,3,2)
    hold on
    fplot(rm,[-pi,2*pi])
    plot([th_0,th_1],[rm(th_0),rm(th_1)],'*')
    grid on
    title('Radius of non-circular pulley')
    legend('r_m')
    xlabel('\theta [rad]')
    ylabel('Radius [m]')

    % Profile in xy-plane
    subplot(1,3,3)
    hold on
    grid on
    fplot(x_P,y_P,[th_0,th_1])
    plot(0,0,'+')
    plot(x_P(th_0),y_P(th_0),'o')
    plot(x_P(th_1),y_P(th_1),'*')
    grid on
    title('Shape of non-circular pulley')
    xlabel('[m]')
    ylabel('[m]')
    axis equal

end
```

A.2.2 Functions to evaluate the profile as proposed taking into account the geometrical corrections

Main function

```
function t_a = check_profile_torque_num_geom_cor(k,u_0,L,geom_corr,x_P,y_P,range,N)

% Function to compute the torque profile generated by the given pulley
% profile, in order to evaluate the quality of the design. This function
% takes into account the geometrical corrections meade due to the rope
% thickness and the fact that the insertion point is on a circular pulley.

% INPUTS
% k: spring constant
% u_0: spring initial elongation
% L: spring insertion point
% tc: cable thickness
% th: symbolic variable
% x_P(th): x_coordinate of the pulley profile
% y_P(th): y_coordinate of the pulley profile
% range: motion range [th_0;th_1]
% N: number of samples used to discretize

% OUTPUTS
% t_a: vector containing the torque profile generated by the pulley profile (x_P(th);y_P(th))

    th_0 = range(1);
    th_1 = range(2);
  tc = geom_corr(1);
    rp = geom_corr(2);
    cb = geom_corr(3);
    x = linspace(th_0,th_1,N);
    incr = (th_1-th_0)/N;

    [x_P,y_P,x_R,y_R] = correct_PR(L,geom_corr,x_P,y_P,range,N); %  Comnpute new Ps and Rs
    RP = sqrt((x_R-x_P).^2+(y_R-y_P).^2);      % distance between insertion point (R) and tangency point
      on the pulley (P)
    ra = abs(x_R.*y_P-y_R.*x_P)./(((x_P-x_R).^2+(y_P-y_R).^2)).^(1/2);    % distance between centre of
     pulley (O) and tangency point (P)

    %% lenght of wire wrapped around the pulley

    % Derivatives of x_P and y_P
    for i = 2:N-1
        xPd_1(i) = (x_P(i+1)-x_P(i))/incr;
        yPd_2(i) = (y_P(i+1)-y_P(i))/incr;

        xPd_2(i) = (x_P(i)-x_P(i-1))/incr;
        yPd_2(i) = (y_P(i)-y_P(i-1))/incr;

        xPd(i) = (xPd_1(i)+xPd_2(i))/2;
        yPd(i) = (xPd_2(i)+yPd_2(i))/2;
    end
    xPd(1) = (x_P(2)-x_P(1))/incr;
    yPd(1) = (y_P(2)-y_P(1))/incr;
    xPd(N) = (x_P(N)-x_P(N-1))/incr;
    yPd(N) = (y_P(N)-y_P(N-1))/incr;

    L1 = (xPd.^2+yPd.^2).^(1/2);

    % Integral of L1
    L_s = [L1(1)*incr];
    for i = 2:N
        L_s(i) = L1(i)*incr+L_s(i-1);
    end

    u = L_s+RP-RP(1)+u_0;    % elongation of the spring
    t_a = k.*u.*ra;          % torque produced by the pulley-spring system

end
```

Function to evaluate the corrected profile

```
function [x_P,y_P,x_R,y_R] = correct_PR(L,geom_corr,x_P,y_P,range,N)
        % Function to compute the new points P and R to use in the function
        % to find the torque that can be compensated, given the profile of
        % a pulley.

  % INPUTS
    % L: spring insertion point
  % tc: cable thickness
    % th: symbolic variable
    % x_P(th): x_coordinate of the pulley profile
    % y_P(th): y_coordinate of the pulley profile
    % range: motion range [th_0;th_1]
    % N: number of samples used to discretize

    % OUTPUTS
    % [x_P(th),y_P(th)]: new coordinated of P (taking into account the width of the cable)
    % [x_R(th),y_R(th)]: new coordinated of R (taking into account the width of the cable and the
     pulley in the insertion point)
```

```
18
19      th_0 = range(1);
20      th_1 = range(2);
21    tc = geom_corr(1);
22      rp = geom_corr(2);
23      cb = geom_corr(3);
24      x = linspace(th_0,th_1,N);
25      incr = (th_1-th_0)/N;
26
27      %% New Ps:
28      [x_P,y_P] = enlarge_profile(0.5*tc,x_P,y_P,[th_0,th_1],N);
29
30      %% New Rs:
31      x_R_1 = L*cos(x);
32      y_R_1 = L*sin(x);
33
34      RP_1 = sqrt((x_R_1-x_P).^2+(y_R_1-y_P).^2);
35      syms gam ips real;
36      x_R = [];
37      y_R = [];
38      m_flag = zeros(1,N);
39      for i = 1:N
40          sol_1 = solve(x_P(i)+RP_1(i)*cos(gam) == x_R_1(i)+(rp+0.5*tc)*cos(ips),y_P(i)+RP_1(i)*sin(gam)
        == y_R_1(i)+(rp+0.5*tc)*sin(ips),gam,ips);
41          sol_1 = double(sol_1.gam);
42          x_R_2 = x_P(i)+RP_1(i)*cos(sol_1);
43          y_R_2 = y_P(i)+RP_1(i)*sin(sol_1);
44          mR1 = (y_R_2(1)-y_P(i))/(x_R_2(1)-x_P(i));
45          mR2 = (y_R_2(2)-y_P(i))/(x_R_2(2)-x_P(i));
46          if mR1 < mR2
47              x_R(i) = x_R_2(1);
48              y_R(i) = y_R_2(1);
49          else
50              x_R(i) = x_R_2(2);
51              y_R(i) = y_R_2(2);
52          end
53          if (mR1*mR2 < 0)
54              if (i < N)
55                  mP = (y_P(i+1)-y_P(i))/(x_P(i+1)-x_P(i));
56              else
57                  mP = (y_P(i)-y_P(i-1))/(x_P(i)-x_P(i-1));
58              end
59              m_flag(i) = 1;
60              if (i == 1)
61                  x_R(i) = x_R_2(1);
62                  y_R(i) = y_R_2(1);
63              else
64                  if (((mP > 0) & (m_flag(i-1) ~= 1)) | (m_flag(i-1) == 2))
65                      m_flag(i) = m_flag(i) + 1;
66                      if mR1 > mR2
67                          x_R(i) = x_R_2(1);
68                          y_R(i) = y_R_2(1);
69                      else
70                          x_R(i) = x_R_2(2);
71                          y_R(i) = y_R_2(2);
72                      end
73                  end
74              end
75          end
76          if (cb == 1)
77              if ((x_R(i) == x_R_2(1)) & (y_R(i) == y_R_2(1)))
78                  x_R(i) = x_R_2(2);
79                  y_R(i) = y_R_2(2);
80              else
81                  x_R(i) = x_R_2(1);
82                  y_R(i) = y_R_2(1);
83              end
84          end
85      end
86  end
```

A.3 Function to enlarge the profile of a curve

```
function [X,Y] = enlarge_profile(tc,x_P,y_P,range,N)

% Function to enlarge or shrink a profile, along the direction of its
% curvature point by point.

% INPUTS
% tc: magnitude (thickness) of the enlargement(tc>0) or shrinkage  (tc>0)
% [x_P,y_P]: parametric coordinates of the curve
% range: range over which the functions are transformed [th_0;th_1]
% N: number of points to discretize the cruve
%

% OUTPUTS
% [X,Y]: coordinates of the transformed curve

    th_0 = range(1);
    th_1 = range(2);
    x = linspace(th_0,th_1,N)';
    incr = abs((th_1-th_0)/N);
    xp = double(x_P(x))';
    yp = double(y_P(x))';

    %% 1st derivative of (x_P,y_P)
    for i = 2:N-1
        xd1(i) = (xp(i+1)-xp(i))/incr;
        yd1(i) = (yp(i+1)-yp(i))/incr;

        xd2(i) = (xp(i)-xp(i-1))/incr;
        yd2(i) = (yp(i)-yp(i-1))/incr;

        xd(i) = (xd1(i)+xd2(i))/2;
        yd(i) = (yd1(i)+yd2(i))/2;
    end

    % derivative of the extremes
    xd(1) = (xp(2)-xp(1))/incr;
    yd(1) = (yp(2)-yp(1))/incr;

    xd(N) = (xp(N)-xp(N-1))/incr;
    yd(N) = (yp(N)-yp(N-1))/incr;

    %% Curvature of (x_P,y_P)
    for i = 1:N
        % Orientation
        xc(i) = -1/xd(i);
        yc(i) = 1/yd(i);
        % Curvature lenght set equal to tc
        lc = sqrt(xc(i)^2+yc(i)^2);
        xc(i) = tc*xc(i)/lc;
        yc(i) = tc*yc(i)/lc;
        % Direction
        if (xc(i)*yc(i) > 0)
            X(i) = xp(i)+xc(i);
            Y(i) = yp(i)+yc(i);
        else
            X(i) = xp(i)-xc(i);
            Y(i) = yp(i)-yc(i);
        end
    end

end
```

A.4 Function to synthesise via the convex optimization approach

```
function [b,r,cvx_optval] = convex_profile(N,m,range,th,t_d,k,u_0,L)

% Function to generate the profile of an noncircular pulley designed to
% conunterbalance a given torque profile. The solution is obtained by
% solvng a convex optimization problem.

% INPUTS
% N: number of samples
% m: degree of the approssimating polynomial
% range: motion range [th_0;th_1]
% th: sybolic variable
% t_d: torque profile to compensate (symfun)
% k: spring constant
% u_0: spring initial elongation
% L: spring insertion point

% OUTPUTS
% b: coefficients of the polynomimal approssimating the profile of the noncircular pulley
% r: coordinates of the noncircular pulley profile
% cvx_optval: optimal value

    th_0 = range(1);
    th_1 = range(2);
    t_d_int(th) = int(t_d(th));      % integral of the torque design
    t_d_diff(th) = diff(t_d(th));    % derivative of the torque design
    th = linspace(th_0,th_1,N)';     % discretisation of the angular range
    t = double(t_d(th));
    t_int = double(t_d_int(th));
    t_diff = double(t_d_diff(th));

    theta  = [];
    a = [];
    for i = 0:m
        theta = [theta,th.^i];
        a = [a;i*(i-1)];
    end
    a = a(3:end);
    A=[];
    for i = 0:(m-2)
        A = [A, a(i+1).*th.^i];
    end
    v = (2*k*t_int+k^2*u_0^2).^(1/2);
    eps = 1e2*(th_1-th_0)/N;

    cvx_begin
        variables r(N,1) b(m+1,1)
        minimize norm(v.*r-t,2)
        subject to
            L-r >= 0        % Pulley does not interfere with insertion point and radius does not go to
 infinity
            r >= 0;         % radius is positive
            r == theta*b;   % radius is approximated with a polynomial of degree m
            A*b(3:end) <= 0;
    cvx_end

end
```

Appendix B

Macros in Adams/View Command Language

The code to create the virtual model of the test rig was subdivided into several macros to simplify the writing and the debugging operations. In fact, in this way it is possible to gradually create the model and control the outcome at every step.

B.1 Definition of main parameters and creation of the pendulum model

```
!WRAP_IN_UNDO NO

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GRAVITY #

force modify body gravitational gravity = .nog2.gravity &
  x_comp = 0 &
  y_comp = -9806.65 &
  z_comp = 0

force attrib force = .nog2.gravity visibility = no_opinion
  entity attributes entity_name = .nog2.gravity active = on depend=on

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # DESIGN VARIABLES #

!---------
! Mass of the pendulum
variable create variable_name = .nog2.mass_pendulum &
  real = 0.89980 &
  units = mass

!---------
! Cartesian coordinates of the noncircular pulley profile (polynomial of 7th degree)
variable create variable_name = .nog2.x_P &
  real = -0.015651180578848, 0.213652508009337, -1.153939853424818, 3.450332228361171,
     -5.963485230816165, -0.532168316445568, 16.135612371732208, -0.103199694472842 &
  units = no_units

variable create variable_name = .nog2.y_P &
  real = 0.029191532886411, -0.300040172286356, 1.180575742064421, -2.057882976776742,
     -0.604959553102431, 8.357158714679175, -0.919000425196018, -1.534300188872310 &
  units = no_units

!---------
! Angular range of the gravity-compensator mechanism
variable create variable_name = .nog2.range &
  real = (PI/6), (5/6*PI) &
  units = no_units

!---------
! Initial angular position of the pendulum
variable create variable_name = .nog2.th_0 &
```

```
  real = (PI/2) &
  units = no_units

!---------
! Distance of the insertion point from the centre of the pulley
variable create variable_name = .nog2.insertion_length &
  real = 108 &
  units = length

!---------
! radius of circular pulleys at the insertion point
variable create variable_name = .nog2.insertion_pulley_radius &
  real = 6.5 &
  units = length

!---------
! Dimensions of belt's elements
variable create variable_name = .nog2.belt_radius &
  real = (1.5/2) &
  units = no_units

!---------
! radius of circular pulleys at the insertion point increased with the belt radius
variable create variable_name = .nog2.insertion_radius &
  real = (.nog2.insertion_pulley_radius + .nog2.belt_radius) &
  units = length

!---------
! radius of circular pulley increased to place the belt elements not in contact with the pulley
variable create variable_name = .nog2.insertion_radius_increased &
  real = (.nog2.insertion_radius+0.2) &
  units = length

!---------
! coefficient of damping elements in the belt
variable create variable_name = .nog2.belt_c &
  real = 0.005 &
  units = damping

!---------
! CONTACT PARAMETERS
variable create variable_name = .nog2.Cnt_Damp &
  real = 1000.0 &
  units = damping

variable create variable_name = .nog2.Cnt_Stiff &
  real = 1e5 &
  units = stiffness

variable create variable_name = .nog2.Cnt_mus &
  real = 0.99 &
  units = no_units

variable create variable_name = .nog2.Cnt_mud &
  real = 0.9 &
  units = no_units

variable create variable_name = .nog2.Cnt_exp &
  real = 2.2 &
  units = no_units

variable create variable_name = .nog2.Cnt_dmax &
  real = 0.1 &
  units = no_units

!---------
! Spring parameters
variable create variable_name = .nog2.spring_k &
  real = 0.410 &
  units = stiffness

variable create variable_name = .nog2.spring_c &
  real = 0.005 &
  units = damping

! pretension of the spring: td(pi/2)/rm(pi/2) = 0.4372[Nm]/0.014205[m] = 30.781 [N]
variable create variable_name = .nog2.spring_F_0 &
  real = 30.781 &
  units = force

! length of the spring
variable create variable_name = .nog2.spring_x_0 &
  real = 65 &
  units = length

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # MATERIALS #

material create &
  material_name = .materials.AISI_304 &
  adams_id = 1000 &
  youngs_modulus = 2.0E5 &
  poissons_ratio = 0.28 &
  density = (7800 (kg/m**3))


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
! ## CREATE PENDULUM ##

part create rigid_body name_and_position part_name = .nog2.PENDULUM
  part attributes part_name = .nog2.PENDULUM color = Goldenrod name_vis = off

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # MASS PROPERTY OF PENDULUM AND NONCIRCULAR PULLEYS #

!--------
!centre of mass
marker create marker = .nog2.PENDULUM.cm &
  location = 0.0, 90.06, 7.45 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2
marker attributes &
  marker_name = .nog2.PENDULUM.cm &
  visibility = off

!--------
!point in the origin to use as RF for the inertia
marker create marker = .nog2.PENDULUM.MK_inertia_RF &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2
marker attributes &
  marker_name = .nog2.PENDULUM.MK_inertia_RF &
  visibility = off


!--------
!mass properties

part create rigid_body mass_properties part_name = .nog2.PENDULUM &
  mass = (.nog2.mass_pendulum) &
  center_of_mass_marker = .nog2.PENDULUM.cm &
  inertia_marker = .nog2.PENDULUM.MK_inertia_RF &
  ixx = (21513.86950) &
  iyy = (1652.35620) &
  izz = (20039.01384) &
  ixy = (0.0) &
  iyz = (0.00264) &
  izx = (86.04018) &


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # STEM #

marker create marker = .nog2.PENDULUM.MK_pos_stem &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, -90.0, 0.0 relative_to = .nog2

geometry create shape cylinder &
  cylinder_name = .nog2.PENDULUM.Stem &
  length = (120.0mm) &
  radius = (5.00mm) &
  angle = 360.0d &
  center_marker = .nog2.PENDULUM.MK_pos_stem

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # NONCIRCULAR PULLEY L #

file parasolid read &
  file_name = "noncircular_pulley_parasolid.x_t" &
  part_name = .nog2.PENDULUM &
  location = 0.0, 0.0, 0.0 &
  orientation = 90.0, 0.0, 0.0 relative_to = .nog2.PENDULUM

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # NONCIRCULAR PULLEY R #

file parasolid read &
  file_name = "noncircular_pulley_parasolid.x_t" &
  part_name = .nog2.PENDULUM &
  location = 0.0, 0.0, 0.0 &
  orientation = 90.0, 180.0, 0.0 relative_to = .nog2.PENDULUM

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATE REVOLUTE JOINTS BETWEEN GROUND AND PENDULUM#

marker create marker = .nog2.PENDULUM.MK_pendulum_Rev_i &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.PENDULUM

marker create marker = .nog2.ground.MK_pendulum_Rev_j &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0  relative_to = .nog2.PENDULUM

! # CREATE REVOLUTE JOINTS BETWEEN GROUND AND PENDULUM#
constraint create joint Revolute &
  joint_name = .nog2.Rev_Gnd_Pendulum &
  i_marker_name = .nog2.PENDULUM.MK_pendulum_Rev_i &
  j_marker_name = .nog2.ground.MK_pendulum_Rev_j

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # REVOLUTE MOTIONS ON JOINT PENDOLUM-GROUND #

constraint create motion_generator motion_name = .nog2.rotation_03 &
  joint_name = .nog2.Rev_Gnd_Pendulum &
  function = " -18d * (STEP( time , 1.99 , 0 ,2.0 , 1 ) + STEP( time , 4.49 , 0 ,4.5 , -1 )) + 7.03d *
     (STEP( time , 6.5 , 0 ,6.640 , 1 ) + STEP( time , 12.880 , 0 ,13.080 , -1 ))" &
  time_derivative = velocity
```

```

constraint attributes constraint_name = .nog2.rotation_03 &
  active = off

constraint create motion_generator motion_name = .nog2.rotation_07 &
  joint_name = .nog2.Rev_Gnd_Pendulum &
  function = "45.0d *( STEP( time , 2 , 0 ,4.0 , 1 ) + STEP( time , 6.0 , 0 , 6.882 , -1 ))" &
  time_derivative = displacement

constraint attributes constraint_name = .nog2.rotation_07 &
  active = off

constraint create motion_generator motion_name = .nog2.rotation_11 &
  joint_name = .nog2.Rev_Gnd_Pendulum &
  function = "-45.0d *( STEP( time , 2 , 0 ,4.0 , 1 ) + STEP5( time , 6.0 , 0 , 6.376 , -1 ))" &
  time_derivative = displacement

constraint attributes constraint_name = .nog2.rotation_11 &
  active = off

constraint create motion_generator motion_name = .nog2.rotation_00 &
  joint_name = .nog2.Rev_Gnd_Pendulum &
  function = "1.0d * STEP( time , 1.99 , 0 , 2.0 , 1 )" &
  time_derivative = velocity

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATE FIXED JOINTS BETWEEN GROUND AND PENDULUM #

marker create marker = .nog2.PENDULUM.MK_pendulum_Fix_i &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.PENDULUM

marker create marker = .nog2.ground.MK_pendulum_Fix_j &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0  relative_to = .nog2.PENDULUM

constraint create joint fixed &
  joint_name = .nog2.Fixed_Gnd_Pendulum &
  i_marker_name = .nog2.PENDULUM.MK_pendulum_Fix_i &
  j_marker_name = .nog2.ground.MK_pendulum_Fix_j

!------------------------------------------------------------------------
! # DEACTIVATE FIXED JOINTS BETWEEN GROUND AND PENDULUM #

constraint attributes constraint_name = .nog2.Fixed_Gnd_Pendulum &
  active = off

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATE CIRCULAR PULLEY AT THE INSERTION POINT L

part create rigid_body name_and_position &
  part_name = .nog2.INSERTION_pulley_L &
  location = (-.nog2.insertion_length), 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2

part attributes part_name = .nog2.INSERTION_pulley_L color = GREEN name_vis = off

marker create marker = .nog2.INSERTION_pulley_L.MK_insertion_point_L &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.INSERTION_pulley_L

marker create marker = .nog2.INSERTION_pulley_L.MK_pos_insertion_pulley_L &
  location = 0.0, 0.0, -10.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.INSERTION_pulley_L.MK_insertion_point_L

geometry create shape cylinder &
  cylinder_name = .nog2.INSERTION_pulley_L.CYLINDER_insertion_pulley_L &
  length = (20.0mm) &
  radius = (.nog2.insertion_pulley_radius) &
  angle = 360.0d &
  center_marker = .nog2.INSERTION_pulley_L.MK_pos_insertion_pulley_L

!---------
!mass properties

part create rigid_body mass_properties part_name = .nog2.INSERTION_pulley_L &
  mass = (0.00273) &
  center_of_mass_marker = .nog2.INSERTION_pulley_L.MK_insertion_point_L &
  inertia_marker = .nog2.INSERTION_pulley_L.MK_insertion_point_L &
  ixx = (0.06960) &
  iyy = (0.06960) &
  izz = (0.12901) &
  ixy = (0.0) &
  iyz = (0.0) &
  izx = (0.0) &

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # REVOLUTE JOINT INSERTION PULLEY L-GROUND #

marker create marker = .nog2.INSERTION_pulley_L.MK_Insertion_L_Rev_i &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.INSERTION_pulley_L

marker create marker = .nog2.ground.MK_Insertion_L_Rev_j &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0  relative_to = .nog2.INSERTION_pulley_L

```

```
constraint create joint Revolute &
  joint_name = .nog2.Rev_Gnd_Insertion_L &
  i_marker_name = .nog2.INSERTION_pulley_L.MK_Insertion_L_Rev_i &
  j_marker_name = .nog2.ground.MK_Insertion_L_Rev_j

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATE CIRCULAR PULLEY AT THE INSERTION POINT R

part create rigid_body name_and_position &
  part_name = .nog2.INSERTION_pulley_R &
  location = (.nog2.insertion_length), 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2

part attributes part_name = .nog2.INSERTION_pulley_R color = GREEN name_vis = off

marker create marker = .nog2.INSERTION_pulley_R.MK_insertion_point_R &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.INSERTION_pulley_R

marker create marker = .nog2.INSERTION_pulley_R.MK_pos_insertion_pulley_R &
  location = 0.0, 0.0, -10.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.INSERTION_pulley_R.MK_insertion_point_R

geometry create shape cylinder &
  cylinder_name = .nog2.INSERTION_pulley_R.CYLINDER_insertion_pulley_R &
  length = (20.0mm) &
  radius = (.nog2.insertion_pulley_radius) &
  angle = 360.0d &
  center_marker = .nog2.INSERTION_pulley_R.MK_pos_insertion_pulley_R

!--------
!mass properties

part create rigid_body mass_properties part_name = .nog2.INSERTION_pulley_R &
  mass = (0.00273) &
  center_of_mass_marker = .nog2.INSERTION_pulley_R.MK_insertion_point_R &
  inertia_marker = .nog2.INSERTION_pulley_R.MK_insertion_point_R &
  ixx = (0.06960) &
  iyy = (0.06960) &
  izz = (0.12901) &
  ixy = (0.0) &
  iyz = (0.0) &
  izx = (0.0) &

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # REVOLUTE JOINT INSERTION PULLEY R-GROUND #

marker create marker = .nog2.INSERTION_pulley_R.MK_Insertion_R_Rev_i &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.INSERTION_pulley_R

marker create marker = .nog2.ground.MK_Insertion_R_Rev_j &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0  relative_to = .nog2.INSERTION_pulley_R

constraint create joint Revolute &
  joint_name = .nog2.Rev_Gnd_Insertion_R &
  i_marker_name = .nog2.INSERTION_pulley_R.MK_Insertion_R_Rev_i &
  j_marker_name = .nog2.ground.MK_Insertion_R_Rev_j


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.2 Creation of the cable model

B.2.1 Computation of the geometric parameters to arrange the cable

```
!WRAP_IN_UNDO NO

!--------
! perimeter of the noncircular pulley (0,PI/2)
variable create variable_name = .nog2.l_90 &
  real = 19.108 &
  units = length

!--------
! perimeter of the noncircular pulley (0,PI/6)
variable create variable_name = .nog2.l_150 &
  real = 7.504 &
  units = length

!--------
! perimeter of the noncircular pulley (0,5/6*PI)
variable create variable_name = .nog2.l_30 &
  real = 29.638 &
  units = length




!--------
! Point of tangency of belt on the noncircular pulley at angle th_0 = 90
var set var = .nog2.pnt_R_90 &
  real = (eval(x_P[1]*th_0**7+x_P[2]*th_0**6+x_P[3]*th_0**5+x_P[4]*th_0**4+x_P[5]*th_0**3+x_P[6]*th_0
    **2+x_P[7]*th_0**1+x_P[8]*th_0**0)), &
      (eval(y_P[1]*th_0**7+y_P[2]*th_0**6+y_P[3]*th_0**5+y_P[4]*th_0**4+y_P[5]*th_0**3+y_P[6]*th_0**2+
    y_P[7]*th_0**1+y_P[8]*th_0**0))


!--------
! Point of tangency of belt on the noncircular pulley at angle range[1] = 150
var set var = .nog2.pnt_R_150_0 &
  real = (eval(x_P[1]*range[1]**7+x_P[2]*range[1]**6+x_P[3]*range[1]**5+x_P[4]*range[1]**4+x_P[5]*range
    [1]**3+x_P[6]*range[1]**2+x_P[7]*range[1]**1+x_P[8]*range[1]**0)), &
      (eval(y_P[1]*range[1]**7+y_P[2]*range[1]**6+y_P[3]*range[1]**5+y_P[4]*range[1]**4+y_P[5]*range
    [1]**3+y_P[6]*range[1]**2+y_P[7]*range[1]**1+y_P[8]*range[1]**0))

!--------
! Rotate point of +60 deegre
var set var = .nog2.pnt_R_150 &
  real = (eval(pnt_R_150_0[1]*COS(180/PI*(th_0-range[1])) - pnt_R_150_0[2]*SIN(180/PI*(th_0-range[1])))
    ), &
      (eval(pnt_R_150_0[1]*SIN(180/PI*(th_0-range[1])) + pnt_R_150_0[2]*COS(180/PI*(th_0-range[1]))))

!--------
! Point of tangency of belt on the noncircular pulley at angle range[2] = 30
var set var = .nog2.pnt_R_30_0 &
  real = (eval(x_P[1]*range[2]**7+x_P[2]*range[2]**6+x_P[3]*range[2]**5+x_P[4]*range[2]**4+x_P[5]*range
    [2]**3+x_P[6]*range[2]**2+x_P[7]*range[2]**1+x_P[8]*range[2]**0)), &
      (eval(y_P[1]*range[2]**7+y_P[2]*range[2]**6+y_P[3]*range[2]**5+y_P[4]*range[2]**4+y_P[5]*range
    [2]**3+y_P[6]*range[2]**2+y_P[7]*range[2]**1+y_P[8]*range[2]**0))

!--------
! Rotate point of -60 deegre
var set var = .nog2.pnt_R_30 &
  real = (eval(pnt_R_30_0[1]*COS(180/PI*(th_0-range[2])) - pnt_R_30_0[2]*SIN(180/PI*(th_0-range[2])))),
    &
      (eval(pnt_R_30_0[1]*SIN(180/PI*(th_0-range[2])) + pnt_R_30_0[2]*COS(180/PI*(th_0-range[2]))))


!--------
! Point S
var set var = .nog2.pnt_S &
  real = (eval(.nog2.insertion_radius)), &
      (eval(.nog2.insertion_length))

!--------
! Point S_0
var set var = .nog2.pnt_S_0 &
  real = (eval(.nog2.insertion_radius_increased)), &
      (eval(.nog2.insertion_length))

!--------
! Distance b
var set var = .nog2.b_90 &
  real = (eval(((pnt_S[1]-pnt_R_90[1])**2 + (pnt_S[2]-pnt_R_90[2])**2)**(1/2)))

var set var = .nog2.b_150 &
  real = (eval(((pnt_S[1]-pnt_R_150[1])**2 + (pnt_S[2]-pnt_R_150[2])**2)**(1/2)))

var set var = .nog2.b_30 &
  real = (eval(((pnt_S[1]-pnt_R_30[1])**2 + (pnt_S[2]-pnt_R_30[2])**2)**(1/2)))

!--------
! distance l_c
```

```
var set var = .nog2.l_c &
  real = (eval(.nog2.insertion_radius*PI/2))

!--------
! length of belt
var set var = .nog2.belt_length_TOT &
  real = (eval(l_30+b_30+l_c))

!--------
! distance d
var set var = .nog2.d_90 &
  real = (eval(belt_length_TOT-(l_90+b_90+l_c)))

var set var = .nog2.d_150 &
  real = (eval(belt_length_TOT-(l_150+b_150+l_c)))

!--------
! length of belt in contact with NONC
var set var = .nog2.l_cnt_nonc &
  real = (eval(l_30))

!--------
! length of belt in contact with CIRC
var set var = .nog2.l_cnt_circ &
  real = (eval(d_150+l_c))

!--------
! length of belt in contact with both pulleys
var set var = .nog2.l_cnt_both &
  real = (eval(d_150-b_30))

!--------
! initiali extension of the spring
var set var = .nog2.x_90 &
  real = (eval(spring_x_0+spring_F_0/spring_k))

!--------
! Point U_90
var set var = .nog2.pnt_U_90 &
  real = (eval(-d_90)), &
       (eval(insertion_length+insertion_radius))

!--------
! Point V
var set var = .nog2.pnt_V &
  real = (eval(-d_90-x_90)), &
       (eval(insertion_length+insertion_radius))

!--------
! Point T
var set var = .nog2.pnt_T &
  real = 0.0, &
       (eval(insertion_length+insertion_radius))

!--------
! Point T_0
var set var = .nog2.pnt_T_0 &
  real = 0.0, &
       (eval(insertion_length+insertion_radius_increased))

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.2.2 Creation of the cable elements on the left noncircular pulley

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! ## CABLE SYSTEM LEFT ##

marker create marker = .nog2.PENDULUM.MK_centre_belt_system_L &
  location = 0.0, 0.0, 0.0 &
  orientation = 90.0, 0.0, 0.0 relative_to = .nog2.PENDULUM

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # BELT ON NONCIRCULAR PULLEY L #

!--------
! initial angle of the noncircular pulley, different from range[1]
var set var = .nog2.nonc_0 &
  real = 0

```

```
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! Compute angle "th_W" at which the belt is not in contact with the noncircular pulley, but is tighten
    between the 2 pulleys


var set var = .nog2.th_W &
  real = 0

var set var= $_self.ipart &
  integer = 0

while condition=(.nog2.th_W <= .nog2.th_0)
  !--------
  ! counter for the name of the belt elements
  var set var= $_self.ipart &
    integer = (eval($_self.ipart+1))

  !--------
  ! divide the range of the angle defining the noncircular pulley profile
  var set var = .nog2.th_W &
    real = (eval(nonc_0 + $_self.ipart*(range[2]-nonc_0)/$n_bodies_nonc_pulley))
end

!--------
! Point of tangency of belt on the noncircular pulley at angle th_W
var set var = .nog2.pnt_W &
  real = (eval(x_P[1]*th_W**7+x_P[2]*th_W**6+x_P[3]*th_W**5+x_P[4]*th_W**4+x_P[5]*th_W**3+x_P[6]*th_W
    **2+x_P[7]*th_W**1+x_P[8]*th_W**0)), &
      (eval(y_P[1]*th_W**7+y_P[2]*th_W**6+y_P[3]*th_W**5+y_P[4]*th_W**4+y_P[5]*th_W**3+y_P[6]*th_W**2+
    y_P[7]*th_W**1+y_P[8]*th_W**0))

!--------
! Position of part W
var set var = .nog2.part_W &
  integer = (eval(ipart))

!--------
! compute the inclination of segment W-S
var set var = .nog2.alpha &
  real = (eval(ATAN2((.nog2.pnt_W[2]-.nog2.pnt_S_0[2]),(.nog2.pnt_W[1]-.nog2.pnt_S_0[1]))))

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATION OF BELT ELEMENTS ON NONCIRCAR PULLEY#

for var = $_self.npart start = 0 inc = 1 end = (eval($n_bodies_nonc_pulley))

  !--------
  ! counter for the name of the belt elements
  var set var= $_self.ipart &
    integer = (eval(rtoi($_self.npart)))

  !--------
  ! divide the range of the angle defining the noncircular pulley profile
  var set var = $_self.th &
    real = (eval(nonc_0 + $_self.ipart*(range[2] - nonc_0)/$n_bodies_nonc_pulley))

  !--------
  ! save preavious coordinates
  if condition = (eval($_self.ipart) != 0)
    var set var = $_self.pnt_nonc_old &
      real = (eval($_self.pnt_nonc[1])), &
          (eval($_self.pnt_nonc[2]))
  end

  !--------
  ! compute new coordinates
  var set var = $_self.pnt_nonc &
    real = (eval(x_P[1]*$_self.th**7+x_P[2]*$_self.th**6+x_P[3]*$_self.th**5+x_P[4]*$_self.th**4+x_P
      [5]*$_self.th**3+x_P[6]*$_self.th**2+x_P[7]*$_self.th**1+x_P[8]*$_self.th**0)), &
        (eval(y_P[1]*$_self.th**7+y_P[2]*$_self.th**6+y_P[3]*$_self.th**5+y_P[4]*$_self.th**4+y_P[5]*
    $_self.th**3+y_P[6]*$_self.th**2+y_P[7]*$_self.th**1+y_P[8]*$_self.th**0))

  !--------
  ! each belt element goes from the the current point to the preavious one. The first point is used to
     fix the belt onto the noncuircular pulley (see the else part)
  if condition = (eval($_self.ipart) != 0)

    !--------
    ! compute length of each belt element as the distance between 2 points
    var set var = $_self.(eval("length_belt_nonc_"//$_self.ipart)) &
      real = (eval(((($_self.pnt_nonc[1]-$_self.pnt_nonc_old[1])**2+($_self.pnt_nonc[2]-$_self.
    pnt_nonc_old[2])**2)**(1/2)))

    !--------
    ! in the if the elements on the noncircular pulley are created, in the else the elements tighten
    between W and S
    if condition = (eval($_self.th) <= eval(.nog2.th_W))

      !--------
      ! compute the inclination of each belt element
      var set var = $_self.(eval("inclination_belt_"//$_self.ipart)) &
        real = (eval(ATAN2(($_self.pnt_nonc[2]-$_self.pnt_nonc_old[2]),($_self.pnt_nonc[1]-$_self.
    pnt_nonc_old[1]))+90.0))

      !--------
      ! create belt element
      part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
        location = (eval($_self.pnt_nonc[1])),(eval($_self.pnt_nonc[2])),0 relative_to = .nog2.PENDULUM
    .MK_centre_belt_system_L
```

```
110       part modify rigid_body mass_properties part_name = (eval("BELT_L_"//$_self.ipart)) material=.
      materials.AISI_304
111       part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = BLUE_GRAY name_vis = off
112
113       !--------
114       ! create marker for cylinder of each belt element
115       marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
116         location = 0.0,0.0, 0.0 &
117         orientation = (eval("inclination_belt_"//$_self.ipart)), -90.0, 0.0 relative_to = (eval("
      BELT_L_"//$_self.ipart))
118
119       !--------
120       ! create geometry of each belt element
121       geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
122         center_marker = (eval("MK_central_belt"//$_self.ipart)) &
123         angle_extent = 360 &
124         length = (eval("length_belt_nonc_"//$_self.ipart)) &
125         radius = (.nog2.belt_radius)
126
127     else
128
129       if condition = (eval($_self.ipart) != eval(part_W + 1))
130         var set var = $_self.pnt_seg_old &
131           real = (eval($_self.pnt_seg[1])), &
132                  (eval($_self.pnt_seg[2]))
133       else
134         var set var = $_self.pnt_seg_old &
135           real = (eval(pnt_W[1])), &
136                  (eval(pnt_W[2]))
137       end
138
139       var set var = $_self.l_W &
140         real = (eval("length_belt_nonc_"//$_self.ipart))
141       !--------
142       ! compute new coordinates
143       var set var = $_self.pnt_seg &
144         real =  (eval(pnt_seg_old[1] - l_W*COS(.nog2.alpha))), &
145               (eval(pnt_seg_old[2] - l_W*SIN(.nog2.alpha)))
146
147       !--------
148       ! create belt element
149       part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
150         location = (eval($_self.pnt_seg[1])),(eval($_self.pnt_seg[2])),0 relative_to = .nog2.PENDULUM.
      MK_centre_belt_system_L
151       part modify rigid_body mass_properties part_name = (eval("BELT_L_"//$_self.ipart)) material=.
      materials.AISI_304
152       part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = BlueViolet name_vis = off
153
154       !--------
155       ! create marker for cylinder of each belt element
156       marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
157         location = 0.0,0.0, 0.0 &
158         orientation = (eval(.nog2.alpha + 90)), 90.0, 0.0 relative_to = (eval("BELT_L_"//$_self.ipart))
159
160       !--------
161       ! create geometry of each belt element
162       geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
163         center_marker = (eval("MK_central_belt"//$_self.ipart)) &
164         angle_extent = 360 &
165         length = (eval("length_belt_nonc_"//$_self.ipart)) &
166         radius = (.nog2.belt_radius)
167
168     end
169
170   else
171
172     !--------
173     ! create belt element
174     part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
175       location = (eval($_self.pnt_nonc[1])),(eval($_self.pnt_nonc[2])),0 relative_to = .nog2.PENDULUM.
      MK_centre_belt_system_L
176     part modify rigid_body mass_properties part_name=(eval("BELT_L_"//$_self.ipart)) material = .
      materials.AISI_304
177     part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = BLUE_GRAY name_vis = off
178
179     !--------
180     ! create marker for cylinder of first belt element
181     marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
182       location = 0.0,0.0, 0.0 &
183       orientation = 90.0, -90.0, 0.0 relative_to = (eval("BELT_L_"//$_self.ipart))
184
185     !--------
186     ! create geometry of first belt element
187     geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
188       center_marker = (eval("MK_central_belt"//$_self.ipart)) &
189       angle_extent = 360 &
190       length = 1 &
191       radius = (.nog2.belt_radius)
192
193   end
194 end
195
196 !--------
197 ! counter for X: the last belt elements that touches only the noncircular pulley
198
199 var set var = .nog2.pnt_X &
200   real = (eval($_self.pnt_seg[1])), &
201        (eval($_self.pnt_seg[2]))
202
```

```
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self , "variable")))
```

B.2.3 Creation of the cable elements between the left noncircular and routing pulleys

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12
!---------
! compute the length of the belt elements not in contact with NONC
var set var = .nog2.belt_length_circ &
  real = (eval(SQRT(2*insertion_radius_increased**2*(1-COS(90/$n_bodies_circ_pulley)))))

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # FIND POINT Y #

var set var = .nog2.d_150_hat &
  real = (eval(d_150 - d_90))

var set var = .nog2.part_Y &
  integer = 0

while condition = (eval(belt_length_circ*part_Y) < eval(d_150_hat))
  var set var = .nog2.part_Y &
    integer = (eval(part_Y + 1))
end

var set var = nog2.pnt_Y &
  real = (eval(pnt_S_0[1] + part_Y*belt_length_circ*COS(alpha))), &
       (eval(pnt_S_0[2] + part_Y*belt_length_circ*SIN(alpha)))


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # BELT BETWEEN X AND Y #

!---------
! counter for the name of the belt elements
var set var= $_self.ipart &
  integer = (eval($n_bodies_nonc_pulley + 1))

var set var = .nog2.length_XY &
  real = (eval(((pnt_Y[1]-pnt_X[1])**2 + (pnt_Y[2]-pnt_X[2])**2)**(1/2)))

!---------
! create belt element
part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
  location = (eval(pnt_Y[1])),(eval(pnt_Y[2])),0 relative_to = .nog2.PENDULUM.MK_centre_belt_system_L
part modify rigid_body mass_properties part_name = (eval("BELT_L_"//$_self.ipart)) material=.materials.
     AISI_304
part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = coral name_vis = off

!---------
! create marker for cylinder of each belt element
marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
  location = 0.0,0.0, 0.0 &
  orientation = (eval(.nog2.alpha + 90)), 90.0, 0.0 relative_to = (eval("BELT_L_"//$_self.ipart))

!---------
! create geometry of each belt element
geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
  center_marker = (eval("MK_central_belt"//$_self.ipart)) &
  angle_extent = 360 &
  length = (eval(length_XY)) &
  radius = (.nog2.belt_radius)


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # BELT BETWEEN y AND S L #

for var = $_self.npart start = (eval($n_bodies_nonc_pulley + 2)) inc = 1 end = (eval(
     $n_bodies_nonc_pulley + part_Y + 1))

  !---------
  ! counter for the name of the belt elements
  var set var= $_self.ipart &
    integer = (eval(rtoi($_self.npart)))

  if condition = (eval($_self.ipart) != eval($n_bodies_nonc_pulley + 2))
    var set var = $_self.pnt_seg_old &
      real = (eval($_self.pnt_seg[1])), &
           (eval($_self.pnt_seg[2]))
  else
    var set var = $_self.pnt_seg_old &
      real = (eval(pnt_Y[1])), &
           (eval(pnt_Y[2]))
```

```
  end

  !---------
  ! compute new coordinates
  var set var = $_self.pnt_seg &
    real =  (eval(pnt_seg_old[1] - belt_length_circ*COS(.nog2.alpha))), &
          (eval(pnt_seg_old[2] - belt_length_circ*SIN(.nog2.alpha)))

  !---------
  ! create belt element
  part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
    location = (eval($_self.pnt_seg[1])),(eval($_self.pnt_seg[2])),0 relative_to = .nog2.PENDULUM.
     MK_centre_belt_system_L
  part modify rigid_body mass_properties part_name = (eval("BELT_L_"//$_self.ipart)) material=.
     materials.AISI_304
  part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = SlateBlue name_vis = off

  !---------
  ! create marker for cylinder of each belt element
  marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
    location = 0.0,0.0, 0.0 &
    orientation = (eval(.nog2.alpha + 90)), 90.0, 0.0 relative_to = (eval("BELT_L_"//$_self.ipart))

  !---------
  ! create geometry of each belt element
  geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
    center_marker = (eval("MK_central_belt"//$_self.ipart)) &
    angle_extent = 360 &
    length = (eval(belt_length_circ)) &
    radius = (.nog2.belt_radius)


end


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.2.4 Creation of the cable elements between on the left routing pulleys

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # BELT ON CIRCULAR PULLEY L #

!---------
! compute the length of the belt elements not in contact with NONC
var set var = .nog2.belt_length_circ &
  real = (eval(SQRT(2*insertion_radius_increased**2*(1-COS(90/$n_bodies_circ_pulley)))))

for var = $_self.npart start = (eval($n_bodies_nonc_pulley + part_Y + 1 + 1)) inc = 1 end = (eval(
     $n_bodies_nonc_pulley + part_Y + 1 + $n_bodies_circ_pulley))

  !---------
  ! counter for the name of the belt elements
  var set var= $_self.ipart &
    integer = (eval(rtoi($_self.npart)))

  var set var = $_self.th &
      real = (eval(90/$n_bodies_circ_pulley*(ipart - ($n_bodies_nonc_pulley + part_Y + 1))))

  !---------
  ! save preavious coordinates
  if condition = (eval($_self.ipart) != eval($n_bodies_nonc_pulley + part_Y + 2))
    var set var = $_self.pnt_circ_old &
      real = (eval($_self.pnt_circ[1])), &
           (eval($_self.pnt_circ[2]))
  else
    var set var = $_self.pnt_circ_old &
      real = (eval(pnt_S_0))
  end

  var set var = $_self.pnt_circ &
    real = (eval(insertion_radius_increased*COS(th))), &
         (eval(insertion_radius_increased*SIN(th)+ insertion_length))

  !---------
  ! compute the inclination of each belt element
  var set var = $_self.(eval("inclination_belt_"//$_self.ipart)) &
    real = (eval(ATAN2((pnt_circ[2]-pnt_circ_old[2]),(pnt_circ[1]-pnt_circ_old[1]))-90.0))

  !---------
  ! create belt element
```

```
45    part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
46      location = (eval($_self.pnt_circ[1])),(eval($_self.pnt_circ[2])),0.0 relative_to = .nog2.Pendulum.
        MK_centre_belt_system_L
47    part modify rigid_body mass_properties part_name=(eval("BELT_L_"//$_self.ipart)) material=.materials.
        AISI_304
48    part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = DimGray name_vis = off
49
50    !--------
51    ! create marker for cylinder of each belt element
52    marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
53      location = 0.0,0.0, 0.0 &
54      orientation = (eval("inclination_belt_"//$_self.ipart)), 90.0, 0.0 relative_to = (eval("BELT_L_"//
        $_self.ipart))
55
56    !--------
57    ! create geometry of each belt element
58    geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
59      center_marker = (eval("MK_central_belt"//$_self.ipart)) &
60      angle_extent = 360 &
61      length = (eval(belt_length_circ)) &
62      radius = (.nog2.belt_radius)
63
64  end
65
66  ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
67  ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
68  ! # GENERAL ATTRIBUTES #
69
70  constraint attributes constraint_name=.* size_of_icons = 1.0
71  force attributes force_name=.* size_of_icons = 1.0
72  marker attributes marker_name=.* size_of_icons = 1.3
73  entity attributes entity_name=.* name_vis = off
74  simulation single set update = "none"
75  var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.2.5 Creation of the cable elements in the left vertical sector

```
1   !WRAP_IN_UNDO NO
2   ! $n_bodies_nonc_pulley:t=integer:d=60
3   ! $n_bodies_circ_pulley:t=integer:d=12
4
5   ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
6   ! # BELT VERTICAL L #
7
8   var set var = .nog2.part_Z &
9     integer = 0
10
11  while condition = (eval(belt_length_circ*part_Z) < eval(d_90))
12    var set var = .nog2.part_Z &
13      integer = (eval(part_Z + 1))
14  end
15
16  var set var = .nog2.part_Z &
17    integer = (eval(part_Z - 1))
18
19  var set var = .nog2.pnt_Z &
20    real = (eval(pnt_T_0[1] - part_Z*belt_length_circ)), &
21         (eval(pnt_T_0[2]))
22
23
24  for var = $_self.npart start = (eval($n_bodies_nonc_pulley + part_Y + 1 + $n_bodies_circ_pulley + 1))
       inc = 1 end = (eval($n_bodies_nonc_pulley + part_Y + 1 + $n_bodies_circ_pulley + part_Z))
25
26    !--------
27    ! counter for the name of the belt elements
28    var set var= $_self.ipart &
29      integer = (eval(rtoi($_self.npart)))
30
31    if condition = (eval($_self.ipart) != eval($n_bodies_circ_pulley + $n_bodies_nonc_pulley + part_Y +
        2))
32      var set var = $_self.pnt_seg_old &
33        real = (eval($_self.pnt_seg[1])), &
34             (eval($_self.pnt_seg[2]))
35    else
36      var set var = $_self.pnt_seg_old &
37        real = (eval(pnt_T_0[1])), &
38             (eval(pnt_T_0[2]))
39    end
40
41    !--------
42    ! compute new coordinates
43    var set var = $_self.pnt_seg &
44      real =  (eval(pnt_seg_old[1] - belt_length_circ)), &
45            (eval(pnt_seg_old[2]))
46
47    !--------
48    ! create belt element
49    part create rigid name part= (eval("BELT_L_"//$_self.ipart)) &
50      location = (eval($_self.pnt_seg[1])),(eval($_self.pnt_seg[2])),0 relative_to = .nog2.PENDULUM.
        MK_centre_belt_system_L
51    part modify rigid_body mass_properties part_name = (eval("BELT_L_"//$_self.ipart)) material=.
        materials.AISI_304
52    part attributes part_name = (eval("BELT_L_"//$_self.ipart)) color = MedTurquoise name_vis = off
53
54    !--------
55    ! create marker for cylinder of each belt element
```

```
  marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
    location = 0.0,0.0, 0.0 &
    orientation = 90.0, 90.0, 0.0 relative_to = (eval("BELT_L_"//$_self.ipart))

  !--------
  ! create geometry of each belt element
  geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
    center_marker = (eval("MK_central_belt"//$_self.ipart)) &
    angle_extent = 360 &
    length = (eval(belt_length_circ)) &
    radius = (.nog2.belt_radius)

end

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.2.6 Addition of the contact and revolute joints to the left cable elements

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12

var set var = .nog2.part_TOT &
  integer = (eval($n_bodies_nonc_pulley + part_Y + 1 + $n_bodies_circ_pulley + part_Z))

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # MODIFY BELT: CREATE MARKERS FOR JOINTS AND CONTACTS WITH NONCIRCULAR PULLEY L #

for var = $_self.npart start = 0 inc = 1 end = (eval(.nog2.part_TOT))

  !--------
  ! counter for the name of the belt elements
  var set var= $_self.ipart &
    integer = (eval(rtoi($_self.npart)))

  !--------
  ! hide cm of belt elements
  marker attributes &
    marker_name = (eval("BELT_L_"//$_self.ipart//".cm")) &
    visibility = off

  !--------
  ! create marker for joints among the belt elements
  marker create marker = (eval(".nog2.BELT_L_"//$_self.ipart//".MK_Rev_j_"//$_self.ipart)) &
    location = 0.0, 0.0, 0.0  relative_to = (eval("BELT_L_"//$_self.ipart))

  if condition=(eval($_self.ipart) != 0)
     marker create marker = (eval(".nog2.BELT_L_"//$_self.ipart//".MK_Rev_i_"//$_self.ipart)) &
      location = 0.0, 0.0, 0.0  relative_to = (eval("BELT_L_"//($_self.ipart-1)))
    !--------
    !--------
    ! create contacts between belt elements and noncircular pulley
    if condition=(eval($_self.ipart) <= eval($n_bodies_nonc_pulley))
      contact create contact_name = (eval("Cnt_nonc_L_"//$_self.ipart)) &
        type =  solid_to_solid &
        i_geometry_name = (eval(".nog2.BELT_L_"//$_self.ipart//".iWALL_"//$_self.ipart)) &
        j_geometry_name = .nog2.PENDULUM.SOLID1 &
        stiffness = (.nog2.Cnt_Stiff)  &
        damping = (.nog2.Cnt_Damp)  &
        exponent = (.nog2.Cnt_exp)  &
        dmax = (.nog2.Cnt_dmax) &
        coulomb_friction = on  &
        mu_static = (.nog2.Cnt_mus)  &
        mu_dynamic = (.nog2.Cnt_mud)  &
        stiction_transition_velocity = 100.0  &
        friction_transition_velocity = 1000.0
        !entity attributes entity_name = (eval("Cnt_nonc_L_"//$_self.ipart)) active=off depend=off

      display_attributes visibility force force_name = (eval("Cnt_nonc_L_"//$_self.ipart)) &
        visibility = off &
        name_visibility = of
    end

    !--------
    ! create contacts between belt elements and circular insertion pulley
    if condition=(eval($_self.ipart) > ($n_bodies_nonc_pulley + 1))
      contact create contact_name = (eval("Cnt_circ_L_"//$_self.ipart)) &
        type =  solid_to_solid &
        i_geometry_name = (eval(".nog2.BELT_L_"//$_self.ipart//".iWALL_"//$_self.ipart)) &
        j_geometry_name = .nog2.INSERTION_pulley_L.CYLINDER_insertion_pulley_L &
        stiffness = (.nog2.Cnt_Stiff)  &
        damping = (.nog2.Cnt_Damp)  &
        exponent = (.nog2.Cnt_exp)  &
        dmax = (.nog2.Cnt_dmax) &
```

```
          coulomb_friction = on  &
          mu_static = (.nog2.Cnt_mus)  &
          mu_dynamic = (.nog2.Cnt_mud)  &
          stiction_transition_velocity = 100.0  &
          friction_transition_velocity = 1000.0
          !entity attributes entity_name = (eval("Cnt_circ_L_"//$_self.ipart)) active=off depend=off

        display_attributes visibility force force_name = (eval("Cnt_circ_L_"//$_self.ipart)) &
          visibility = off &
          name_visibility = off
      end

    end

end

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATE REVOLUTE JOINTS AMONG BELT ELEMENTS #

for var = $_self.npart start = 1 inc = 1 end = (eval(.nog2.part_TOT))

  !---------
  ! counter for the name of the belt elements
  var set var = $_self.ipart &
    integer = (eval(rtoi($_self.npart)))

  ! revolute joints between a belt element and the previous one
  constraint create joint Revolute &
    joint_name = (eval("Rev_belt_L_"//$_self.ipart)) &
    i_marker_name = (eval(".nog2.BELT_L_"//$_self.ipart//".MK_Rev_i_"//$_self.ipart)) &
    j_marker_name = (eval(".nog2.BELT_L_"//$_self.ipart-1//".MK_Rev_j_"//$_self.ipart-1))

  ! rotational torque between a belt element and the previous one. This is added on order to stop the belt from vibrating
  force create element_like rotational_spring_damper &
    spring_damper_name= (eval("Friction_torque_belt_L_"//$_self.ipart)) &
    damping= (.nog2.belt_c) &
    stiffness= 0 &
    i_marker_name = (eval(".nog2.BELT_L_"//$_self.ipart//".MK_Rev_i_"//$_self.ipart)) &
    j_marker_name = (eval(".nog2.BELT_L_"//$_self.ipart-1//".MK_Rev_j_"//$_self.ipart-1))

end

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATE FIXED JOINTS BETWEEN BELT AND NONCIRCULAR PULLEY L #

constraint create joint fixed &
  joint_name = .nog2.Fix_Belt_Nonc_Pulley_L &
  i_part_name = .nog2.BELT_L_0 &
  j_part_name = .nog2.PENDULUM &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 0.0, 0.0 relative_to = .nog2.BELT_L_0

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.2.7 Creation of the right cable model

The macros to create the model of the right cable are identical to the ones to create the model left cable reported from Appendix B.2.2 to B.2.6, with the only exceptions of rotating of 180° the reference marker `MK_centre_belt_sytem_L` and substituting all the suffixes `_L` with `_R`. Here the first macro is reported as an example.

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! ## CABLE SYSTEM RIGHT ##

marker create marker = .nog2.PENDULUM.MK_centre_belt_system_R &
  location = 0.0, 0.0, 0.0 &
  orientation = 0.0, 180.0, 0.0 relative_to = .nog2.PENDULUM.MK_centre_belt_system_L

! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # CREATION OF BELT ELEMENTS ON NONCIRCAR PULLEY#

for var = $_self.npart start = 0 inc = 1 end = (eval($n_bodies_nonc_pulley))

  !--------
  ! counter for the name of the belt elements
  var set var= $_self.ipart &
    integer = (eval(rtoi($_self.npart)))

  !--------
  ! divide the range of the angle defining the noncircular pulley profile
  var set var = $_self.th &
    real = (eval(nonc_0 + $_self.ipart*(range[2] - nonc_0)/$n_bodies_nonc_pulley))

  !--------
  ! save preavious coordinates
  if condition = (eval($_self.ipart) != 0)
    var set var = $_self.pnt_nonc_old &
      real = (eval($_self.pnt_nonc[1])), &
           (eval($_self.pnt_nonc[2]))
  end

  !--------
  ! compute new coordinates
  var set var = $_self.pnt_nonc &
    real = (eval(x_P[1]*$_self.th**7+x_P[2]*$_self.th**6+x_P[3]*$_self.th**5+x_P[4]*$_self.th**4+x_P[5]*$_self.th**3+x_P[6]*$_self.th**2+x_P[7]*$_self.th**1+x_P[8]*$_self.th**0)), &
         (eval(y_P[1]*$_self.th**7+y_P[2]*$_self.th**6+y_P[3]*$_self.th**5+y_P[4]*$_self.th**4+y_P[5]*$_self.th**3+y_P[6]*$_self.th**2+y_P[7]*$_self.th**1+y_P[8]*$_self.th**0))

  !--------
  ! each belt element goes from the the current point to the preavious one. The first point is used to fix the belt onto the noncuircular pulley (see the else part)
  if condition = (eval($_self.ipart) != 0)

    !--------
    ! compute length of each belt element as the distance between 2 points
    var set var = $_self.(eval("length_belt_nonc_"//$_self.ipart)) &
      real = (eval((($_self.pnt_nonc[1]-$_self.pnt_nonc_old[1])**2+($_self.pnt_nonc[2]-$_self.pnt_nonc_old[2])**2)**(1/2)))

    !--------
    ! in the if the elements on the noncircular pulley are created, in the else the elements tighten between W and S
    if condition = (eval($_self.th) <= eval(.nog2.th_W))

      !--------
      ! compute the inclination of each belt element
      var set var = $_self.(eval("inclination_belt_"//$_self.ipart)) &
        real = (eval(ATAN2(($_self.pnt_nonc[2]-$_self.pnt_nonc_old[2]),($_self.pnt_nonc[1]-$_self.pnt_nonc_old[1]))+90.0))

      !--------
      ! create belt element
      part create rigid name part= (eval("BELT_R_"//$_self.ipart)) &
        location = (eval($_self.pnt_nonc[1])),(eval($_self.pnt_nonc[2])),0 relative_to = .nog2.PENDULUM.MK_centre_belt_system_R
      part modify rigid_body mass_properties part_name = (eval("BELT_R_"//$_self.ipart)) material=.materials.AISI_304
      part attributes part_name = (eval("BELT_R_"//$_self.ipart)) color = BLUE_GRAY name_vis = off

      !--------
      ! create marker for cylinder of each belt element
      marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
        location = 0.0,0.0, 0.0 &
        orientation = (eval("inclination_belt_"//$_self.ipart)), -90.0, 0.0 relative_to = (eval("BELT_R_"//$_self.ipart))

      !--------
      ! create geometry of each belt element
      geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
        center_marker = (eval("MK_central_belt"//$_self.ipart)) &
        angle_extent = 360 &
```

```
78          length = (eval("length_belt_nonc_"//$_self.ipart)) &
79          radius = (.nog2.belt_radius)
80
81      else
82
83        if condition = (eval($_self.ipart) != eval(part_W + 1))
84          var set var = $_self.pnt_seg_old &
85            real = (eval($_self.pnt_seg[1])), &
86                 (eval($_self.pnt_seg[2]))
87        else
88          var set var = $_self.pnt_seg_old &
89            real = (eval(pnt_W[1])), &
90                 (eval(pnt_W[2]))
91        end
92
93        var set var = $_self.l_W &
94          real = (eval("length_belt_nonc_"//$_self.ipart))
95        !--------
96        ! compute new coordinates
97        var set var = $_self.pnt_seg &
98          real =  (eval(pnt_seg_old[1] - l_W*COS(.nog2.alpha))), &
99                (eval(pnt_seg_old[2] - l_W*SIN(.nog2.alpha)))
100
101       !--------
102       ! create belt element
103       part create rigid name part= (eval("BELT_R_"//$_self.ipart)) &
104         location = (eval($_self.pnt_seg[1])),(eval($_self.pnt_seg[2])),0 relative_to = .nog2.PENDULUM.
      MK_centre_belt_system_R
105       part modify rigid_body mass_properties part_name = (eval("BELT_R_"//$_self.ipart)) material=.
      materials.AISI_304
106       part attributes part_name = (eval("BELT_R_"//$_self.ipart)) color = BlueViolet name_vis = off
107
108       !--------
109       ! create marker for cylinder of each belt element
110       marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
111         location = 0.0,0.0, 0.0 &
112         orientation = (eval(.nog2.alpha + 90)), 90.0, 0.0 relative_to = (eval("BELT_R_"//$_self.ipart))
113
114       !--------
115       ! create geometry of each belt element
116       geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
117         center_marker = (eval("MK_central_belt"//$_self.ipart)) &
118         angle_extent = 360 &
119         length = (eval("length_belt_nonc_"//$_self.ipart)) &
120         radius = (.nog2.belt_radius)
121
122     end
123
124   else
125
126     !--------
127     ! create belt element
128     part create rigid name part= (eval("BELT_R_"//$_self.ipart)) &
129       location = (eval($_self.pnt_nonc[1])),(eval($_self.pnt_nonc[2])),0 relative_to = .nog2.PENDULUM.
      MK_centre_belt_system_R
130     part modify rigid_body mass_properties part_name=(eval("BELT_R_"//$_self.ipart)) material = .
      materials.AISI_304
131     part attributes part_name = (eval("BELT_R_"//$_self.ipart)) color = BLUE_GRAY name_vis = off
132
133
134     !--------
135     ! create marker for cylinder of first belt element
136     marker create marker = (eval("MK_central_belt"//$_self.ipart)) &
137       location = 0.0,0.0, 0.0 &
138       orientation = 90.0, -90.0, 0.0 relative_to = (eval("BELT_R_"//$_self.ipart))
139
140     !--------
141     ! create geometry of first belt element
142     geometry create shape cylinder cylinder_name = (eval("iWALL_"//$_self.ipart)) &
143       center_marker = (eval("MK_central_belt"//$_self.ipart)) &
144       angle_extent = 360 &
145       length = 1 &
146       radius = (.nog2.belt_radius)
147
148   end
149 end
150
151 ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
152 ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
153 ! # GENERAL ATTRIBUTES #
154
155 constraint attributes constraint_name=.* size_of_icons = 1.0
156 force attributes force_name=.* size_of_icons = 1.0
157 marker attributes marker_name=.* size_of_icons = 1.3
158 entity attributes entity_name=.* name_vis = off
159 simulation single set update = "none"
160 var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.3 Final adjustments

B.3.1 Creation of the springs model

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12

!--------
! # FORCE REPRESENTING THE SPRING FOR PULLEY L #

!--------
! create marker of spring anchor point on the belt
marker create marker = (eval(".nog2.BELT_L_"//(.nog2.part_TOT))//".MK_end_belt_L") &
  location = 0.0, 0.0, 0.0 relative_to = (eval(".nog2.BELT_L_"//(.nog2.part_TOT)//".MK_Rev_j_"//(.nog2.
    part_TOT)))

marker create marker = .nog2.ground.MK_V_L &
  location = (eval(pnt_V[1])), (eval(pnt_V[2])), 0.0 relative_to = .nog2.PENDULUM.
    MK_centre_belt_system_L

!--------
force create direct single_component_force &
  single_component_force_name = .nog2.Spring_L &
  type_of_freedom = translational &
  function = "-spring_k*(DM(MK_V_L,MK_end_belt_L)-spring_x_0)-spring_c*VM(MK_V_L,MK_end_belt_L,MK_V_L)"
    &
  i_marker_name = (eval(".nog2.BELT_L_"//(nog2.part_TOT))//".MK_end_belt_L") &
  j_marker_name = .nog2.ground.MK_V_L


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # FORCE REPRESENTING THE SPRING FOR PULLEY R #

!--------
! create marker of spring anchor point on the belt
marker create marker = (eval(".nog2.BELT_R_"//(.nog2.part_TOT))//".MK_end_belt_R") &
  location = 0.0, 0.0, 0.0 relative_to = (eval(".nog2.BELT_R_"//(.nog2.part_TOT)//".MK_Rev_j_"//(.nog2.
    part_TOT)))

marker create marker = .nog2.ground.MK_V_R &
  location = (eval(pnt_V[1])), (eval(pnt_V[2])), 0.0 relative_to = .nog2.PENDULUM.
    MK_centre_belt_system_R

!--------
force create direct single_component_force &
  single_component_force_name = .nog2.Spring_R &
  type_of_freedom = translational &
  function = "-spring_k*(DM(MK_V_R,MK_end_belt_R)-spring_x_0)-spring_c*VM(MK_V_R,MK_end_belt_R,MK_V_R)"
    &
  i_marker_name = (eval(".nog2.BELT_R_"//(nog2.part_TOT))//".MK_end_belt_R") &
  j_marker_name = .nog2.ground.MK_V_R


! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
! # GENERAL ATTRIBUTES #

constraint attributes constraint_name=.* size_of_icons = 1.0
force attributes force_name=.* size_of_icons = 1.0
marker attributes marker_name=.* size_of_icons = 1.3
entity attributes entity_name=.* name_vis = off
simulation single set update = "none"
var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

B.3.2 Creation of the tightening forces model

```
!WRAP_IN_UNDO NO
! $n_bodies_nonc_pulley:t=integer:d=60
! $n_bodies_circ_pulley:t=integer:d=12

!-----------------------------------------------------------
! # FORCE TO TIGHTEN BELT L #

marker create marker = (eval(".nog2.BELT_L_"//(.nog2.part_TOT))//".MK_force_L_i") &
  location = 0.0, 0.0, 0.0 &
  orientation = 90.0, -90.0, 0.0 relative_to = (eval(".nog2.BELT_L_"//(.nog2.part_TOT)//".MK_Rev_j_
    "//(.nog2.part_TOT)))

marker create marker = .nog2.ground.MK_force_L_j &
  location = 0.0, 0.0, 0.0 &
  orientation = 90.0, -90.0, 0.0 relative_to = (eval(".nog2.BELT_L_"//(.nog2.part_TOT)//".MK_Rev_j_
    "//(.nog2.part_TOT)))

force create direct single_component_force single_component_force_name = .nog2.tight_L &
  type_of_freedom = translational &
  action_only= on &
  function = (spring_F_0) &
  i_marker_name = (eval(".nog2.BELT_L_"//(.nog2.part_TOT))//".MK_force_L_i") &
```

```
21   j_marker_name = .nog2.ground.MK_force_L_j
22
23
24 !-----------------------------------------------------------
25 ! # FORCE TO TIGHTEN BELT R #
26
27 marker create marker = (eval(".nog2.BELT_R_"//(.nog2.part_TOT))//".MK_force_R_i") &
28   location = 0.0, 0.0, 0.0 &
29   orientation = 90.0, 90.0, 0.0 relative_to = (eval(".nog2.BELT_R_"//(.nog2.part_TOT)//".MK_Rev_j_"//(.
       nog2.part_TOT)))
30
31 marker create marker = .nog2.ground.MK_force_R_j &
32   location = 0.0, 0.0, 0.0 &
33   orientation = 90.0, -90.0, 0.0 relative_to = (eval(".nog2.BELT_R_"//(.nog2.part_TOT)//".MK_Rev_j_
       "//(.nog2.part_TOT)))
34
35 force create direct single_component_force single_component_force_name = .nog2.tight_R &
36   type_of_freedom = translational &
37   action_only= on &
38   function = (spring_F_0) &
39   i_marker_name = (eval(".nog2.BELT_R_"//(.nog2.part_TOT))//".MK_force_R_i") &
40   j_marker_name = .nog2.ground.MK_force_R_j
41
42 !-----------------------------------------------------------
43 ! # DEACTIVATE TGHTENING FORCES #
44
45 force attributes force_name = .nog2.tight_L &
46   active = off
47
48 force attributes force_name = .nog2.tight_R &
49   active = off
50
51 ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
52 ! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
53 ! # GENERAL ATTRIBUTES #
54
55 constraint attributes constraint_name=.* size_of_icons = 1.0
56 force attributes force_name=.* size_of_icons = 1.0
57 marker attributes marker_name=.* size_of_icons = 1.3
58 entity attributes entity_name=.* name_vis = off
59 simulation single set update = "none"
60 var delete var = (eval(DB_CHILDREN($_self, "variable")))
```

Appendix C

Dimension drawings for the test rig

Table C.1: Parameters of the final virtual prototype

Base and supports	
Base	Figure C.1
Base of right shaft support	Figure C.2
Right shaft support	Figure C.3
Base of left shaft support	Figure C.4
Left shaft support	Figure C.5
Motor support	Figure C.6
Pendulum and shaft	
Pendulum base	Figure C.7
Pendulum stem	Figure C.8
Pendulum sphere	Figure C.9
Shaft	Figure C.10
Reference angle system	
Rod	Figure C.11
Base	Figure C.12
Routing pulleys	
Axis	Figure C.13
Base	Figure C.14
Spring holder	
Rod	Figure C.15
Support	Figure C.16
Preload system	
Cylinder	Figure C.17
Support	Figure C.18
Squared rod	Figure C.19

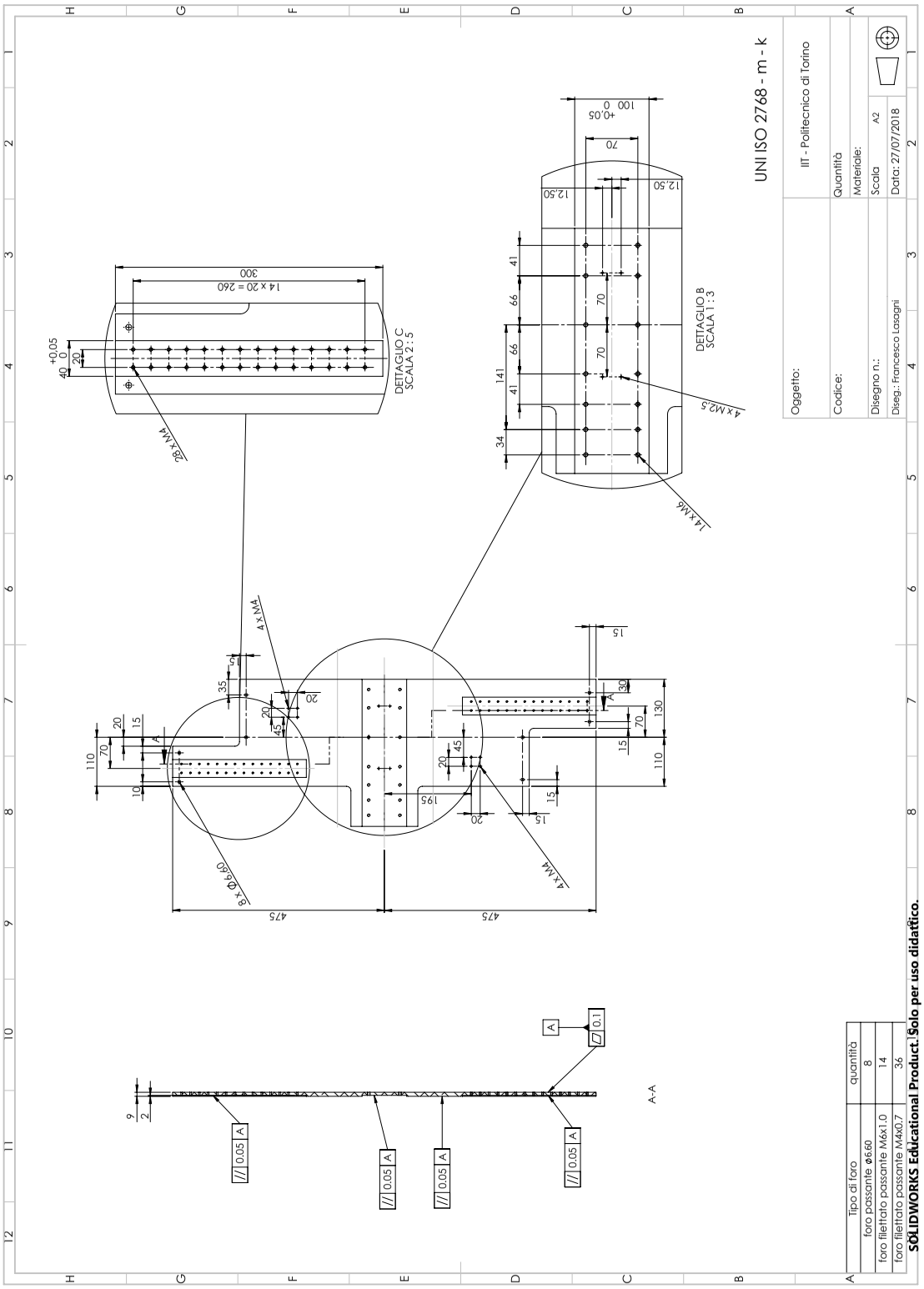

UNI ISO 2768 - m - k
IIT - Politecnico di Torino
Oggetto:
Codice:
Quantità
Materiale:
Disegno n.:
Scala A2
Disegn.: Francesco Lasagni
Data: 27/07/2018
DETTAGLIO C
SCALA 2 : 5
DETTAGLIO B
SCALA 1 : 3
A-A
Tipo di foro | quantità
foro passante ø6.60 | 8
foro filettato passante M6x1.0 | 14
foro filettato passante M4x0.7 | 36
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.1

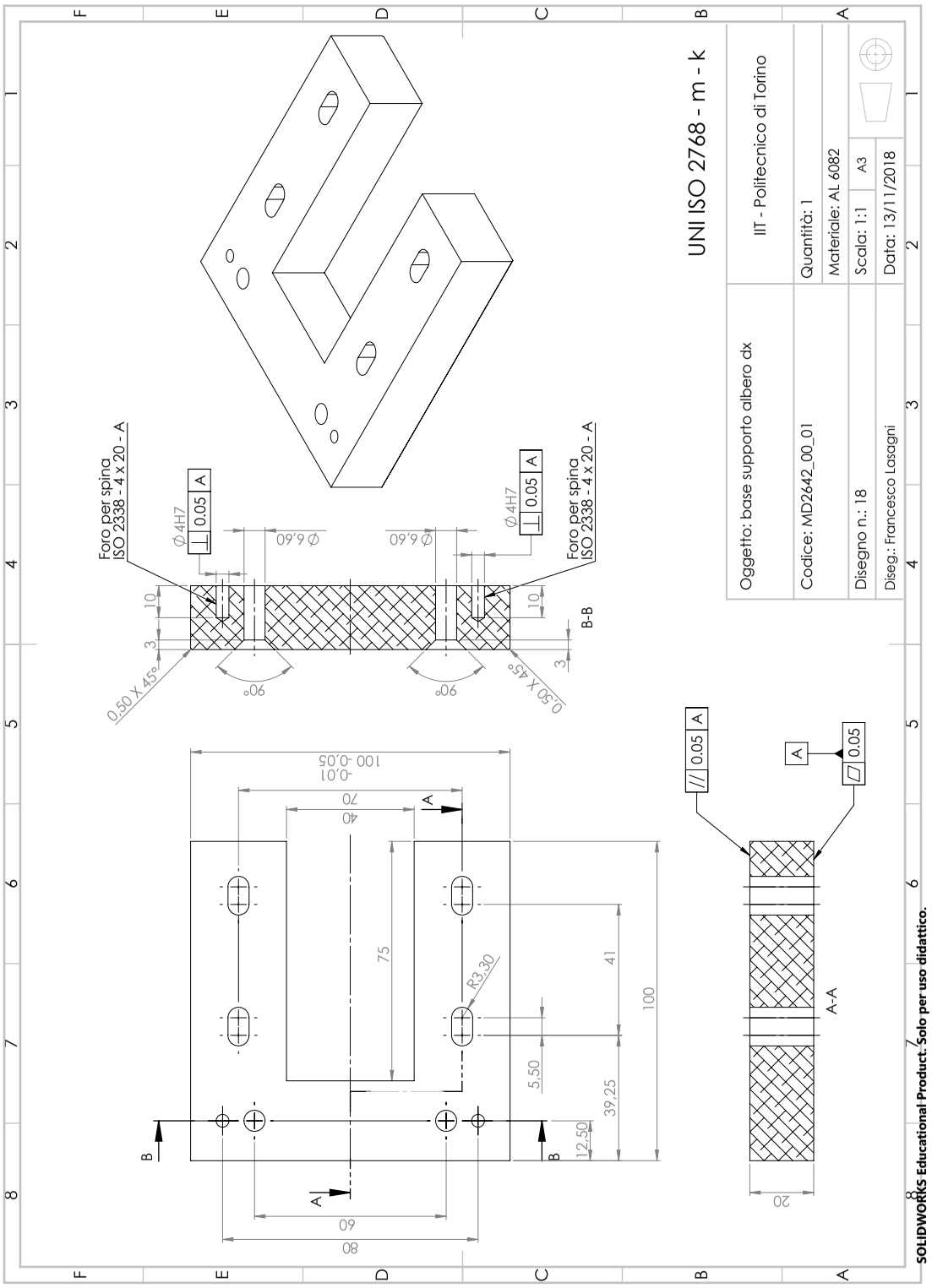

UNI ISO 2768 - m - k
IIT - Politecnico di Torino
Oggetto: base supporto albero dx
Codice: MD2642_00_01
Quantità: 1
Materiale: AL 6082
Scala: 1:1
A3
Disegno n.: 18
Data: 13/11/2018
Diseg.: Francesco Lasagni
Foro per spina ISO 2338 - 4 x 20 - A
Foro per spina ISO 2338 - 4 x 20 - A
B-B
A-A
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.2

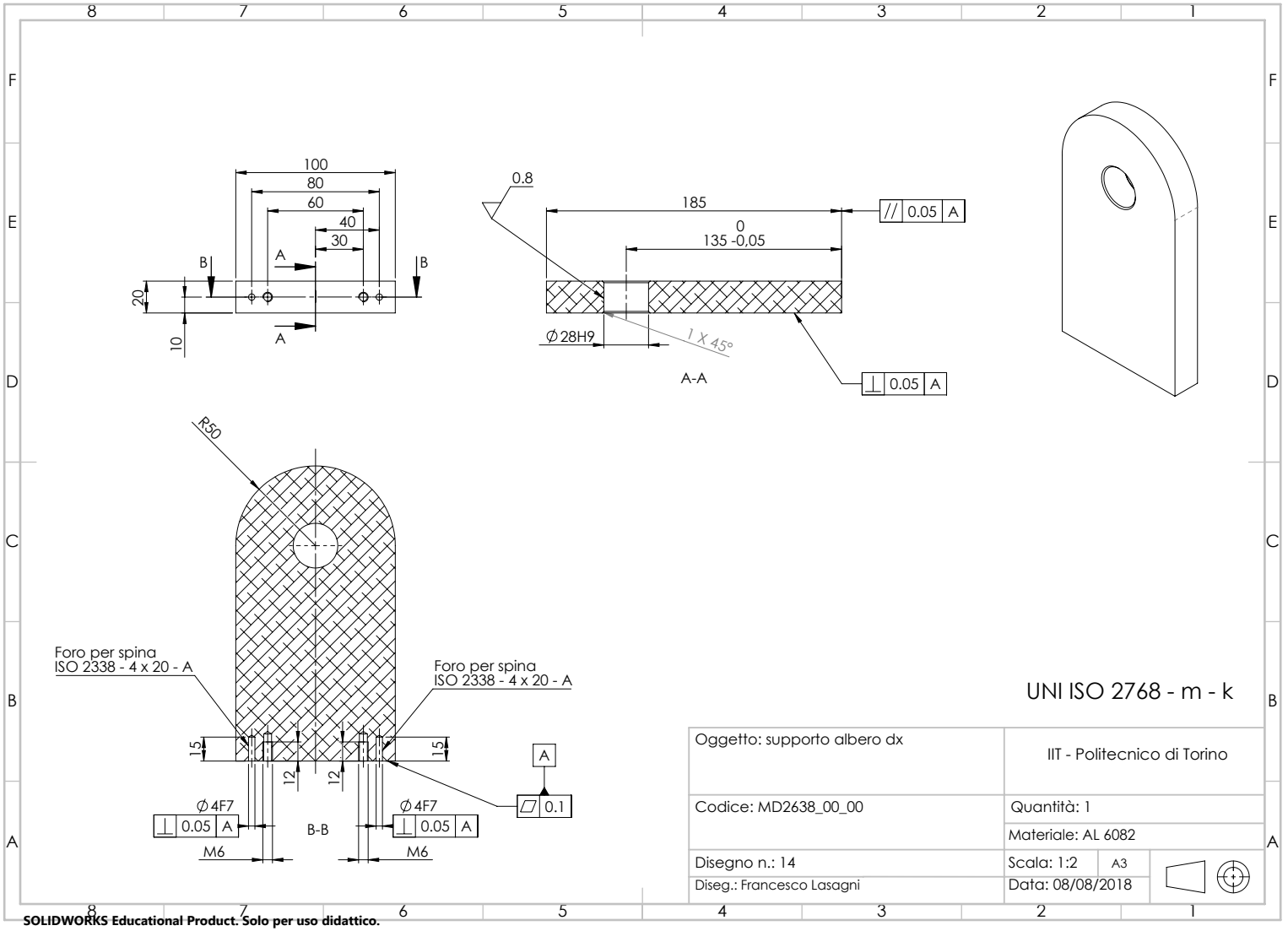

100
80
60
40
30
20
10
A
B
0.8
185
0
135 -0,05
// 0.05 A
Ø28H9
1 X 45°
0.05 A
A-A
R50
Foro per spina
ISO 2338 - 4 x 20 - A
Foro per spina
ISO 2338 - 4 x 20 - A
15
12
12
15
Ø4F7
0.05 A
B-B
Ø4F7
0.05 A
M6
M6
A
0.1
UNI ISO 2768 - m - k
Oggetto: supporto albero dx
IIT - Politecnico di Torino
Codice: MD2638_00_00
Quantità: 1
Materiale: AL 6082
Disegno n.: 14
Scala: 1:2
A3
Diseg.: Francesco Lasagni
Data: 08/08/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.3

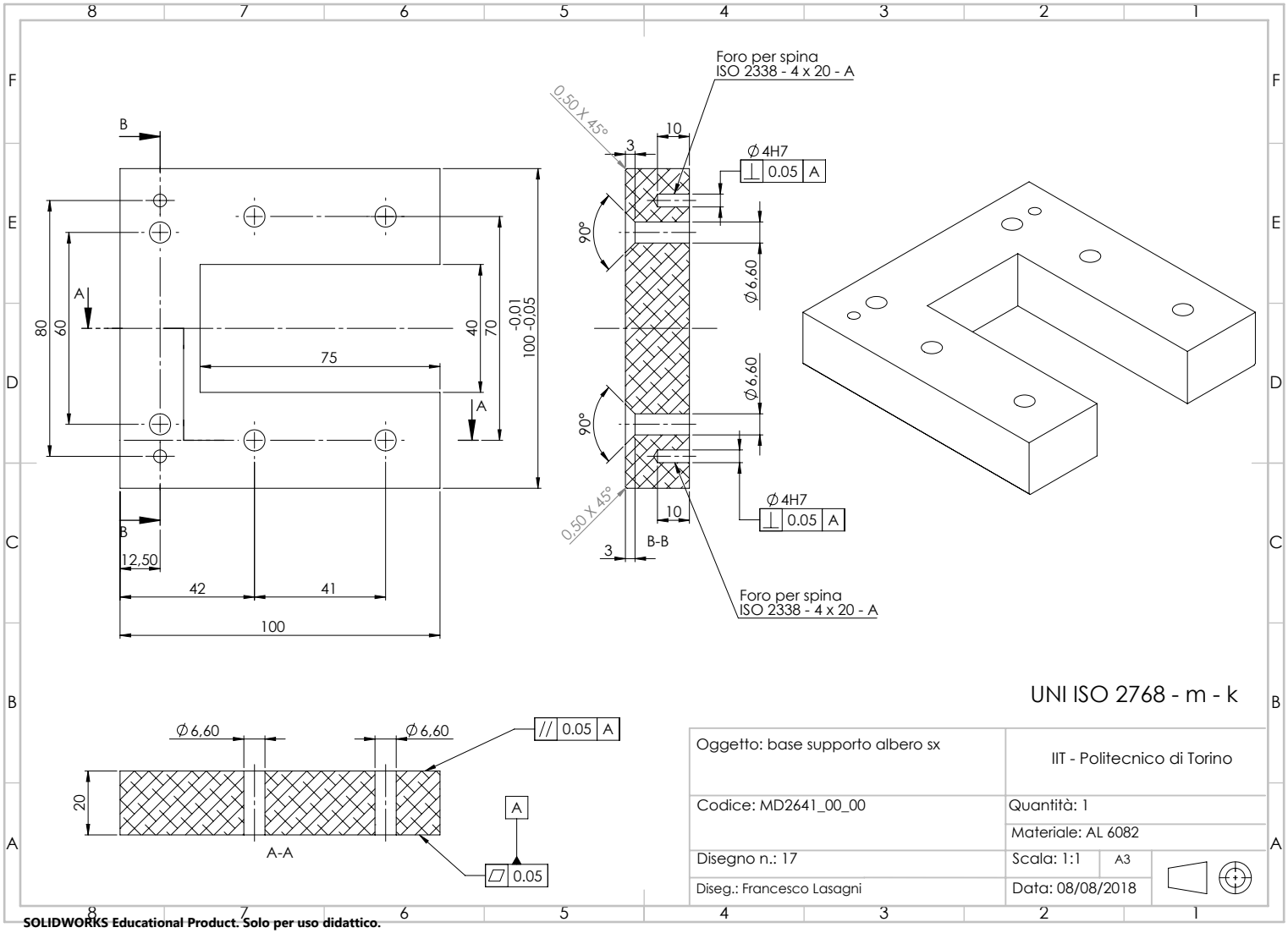

Foro per spina
ISO 2338 - 4 x 20 - A
0.50 X 45°
10
3
Ø 4H7
0.05 A
90°
Ø 6,60
Ø 6,60
90°
Ø 4H7
0.05 A
0.50 X 45°
10
3
B-B
Foro per spina
ISO 2338 - 4 x 20 - A
B
A
80
60
40
70
75
-0,01
100 -0,05
A
B
12,50
42
41
100
UNI ISO 2768 - m - k
Ø 6,60
Ø 6,60
// 0.05 A
20
A
A-A
0.05
Oggetto: base supporto albero sx
IIT - Politecnico di Torino
Codice: MD2641_00_00
Quantità: 1
Materiale: AL 6082
Disegno n.: 17
Scala: 1:1
A3
Diseg.: Francesco Lasagni
Data: 08/08/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.4

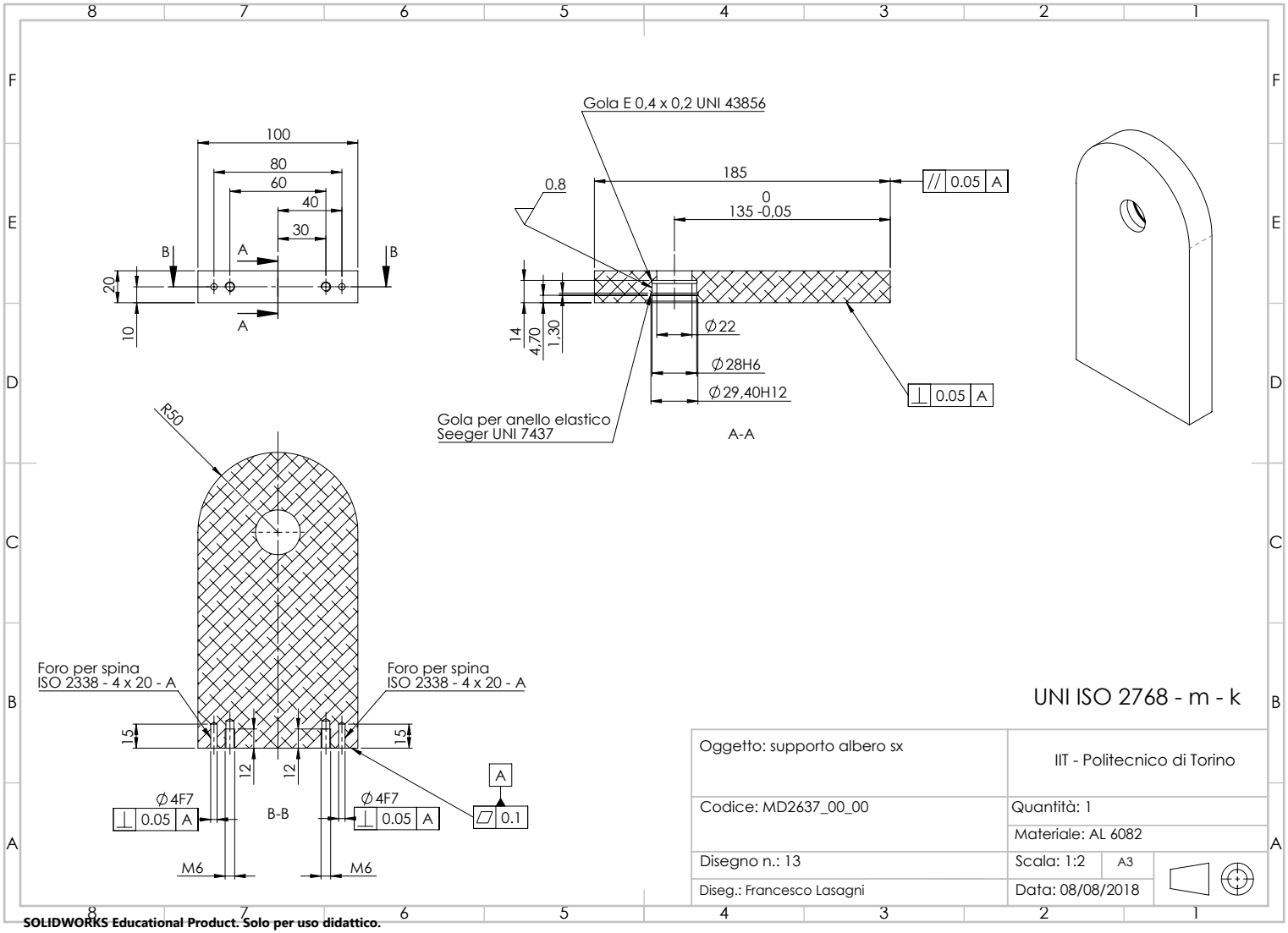

Gola E 0,4 x 0,2 UNI 43856
100
80
60
40
30
185
0
135 -0,05
0.8
// 0.05 A
20
10
14
4,70
1,30
Ø22
Ø28H6
Ø29,40H12
0.05 A
Gola per anello elastico
Seeger UNI 7437
A-A
R50
Foro per spina
ISO 2338 - 4 x 20 - A
Foro per spina
ISO 2338 - 4 x 20 - A
15
15
12
12
Ø4F7
0.05 A
B-B
Ø4F7
0.05 A
A
0.1
M6
M6
UNI ISO 2768 - m - k
Oggetto: supporto albero sx
IIT - Politecnico di Torino
Codice: MD2637_00_00
Quantità: 1
Materiale: AL 6082
Disegno n.: 13
Scala: 1:2
A3
Diseg.: Francesco Lasagni
Data: 08/08/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.5

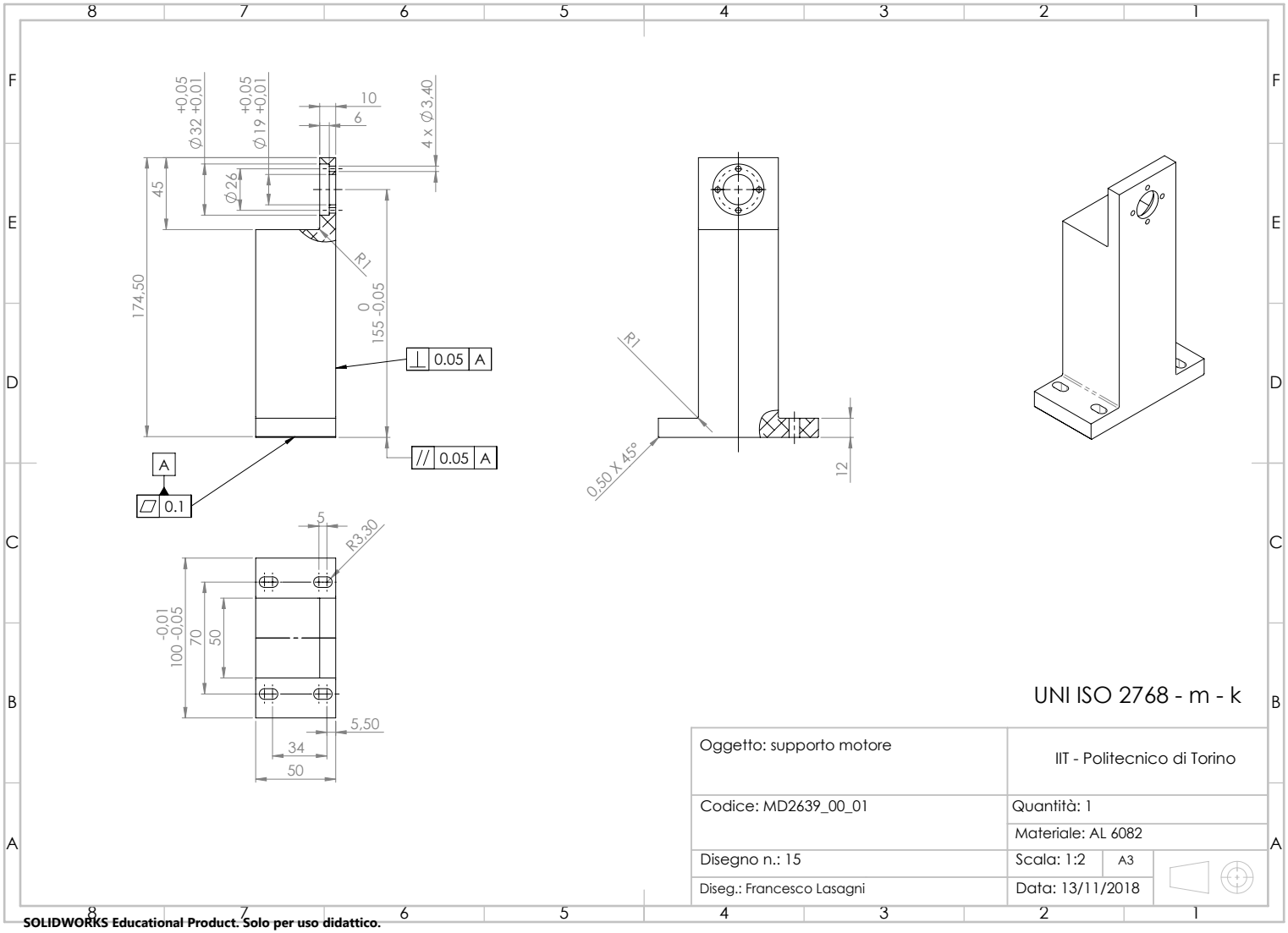

+0,05
Ø32 +0,01
+0,05
Ø19 +0,01
10
6
4 x Ø3,40
45
Ø26
R1
174,50
0
155 -0,05
0.05 A
0.05 A
A
0.1
5
R3,30
-0,01
100 -0,05
70
50
5,50
34
50
R1
0,50 X 45°
12
UNI ISO 2768 - m - k
Oggetto: supporto motore
IIT - Politecnico di Torino
Codice: MD2639_00_01
Quantità: 1
Materiale: AL 6082
Disegno n.: 15
Scala: 1:2
A3
Diseg.: Francesco Lasagni
Data: 13/11/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.6

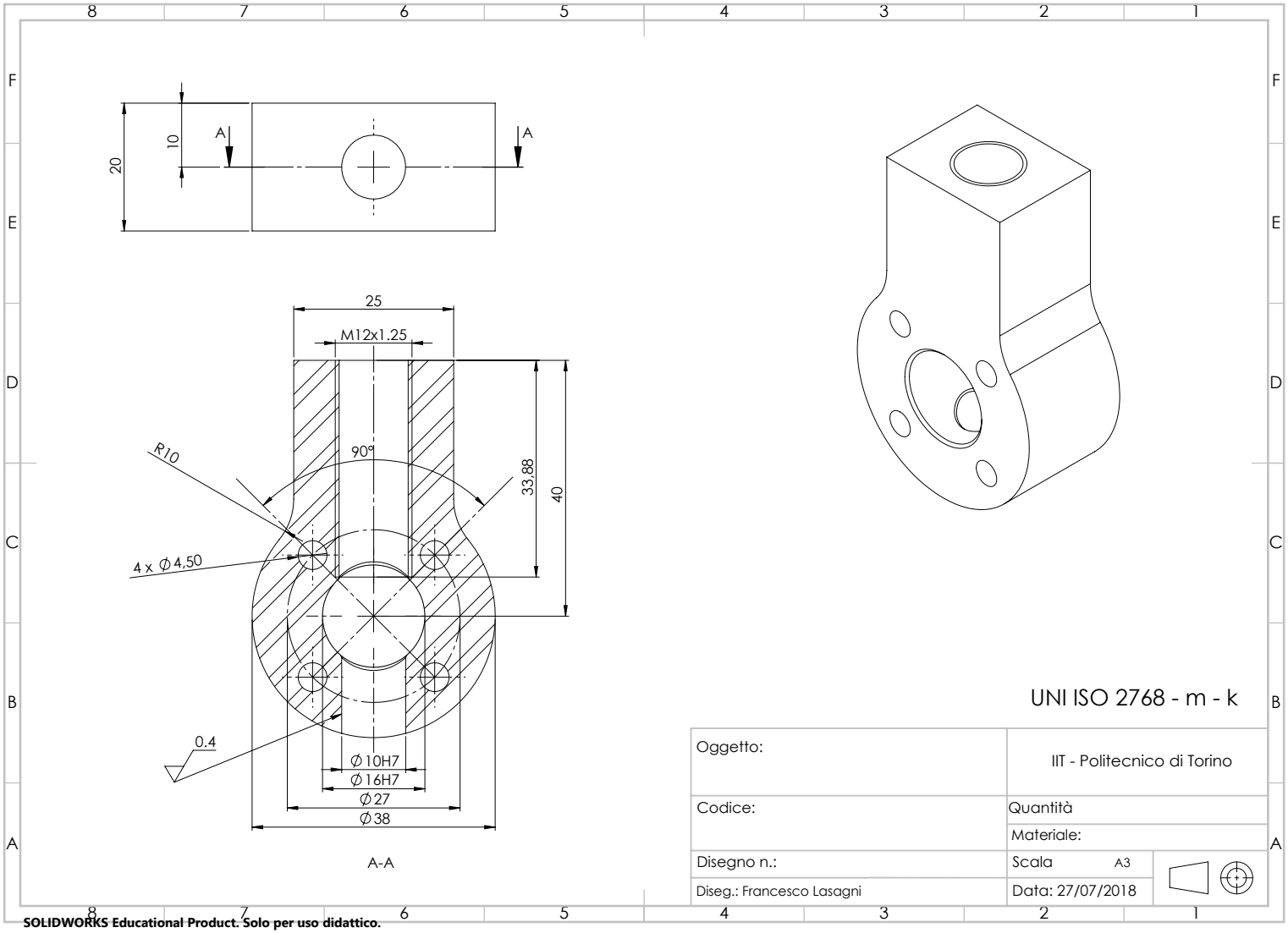

20
10
A
A
25
M12x1.25
R10
90°
33,88
40
4 x ∅4,50
0.4
∅10H7
∅16H7
∅27
∅38
A-A
UNI ISO 2768 - m - k
Oggetto:
IIT - Politecnico di Torino
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.7

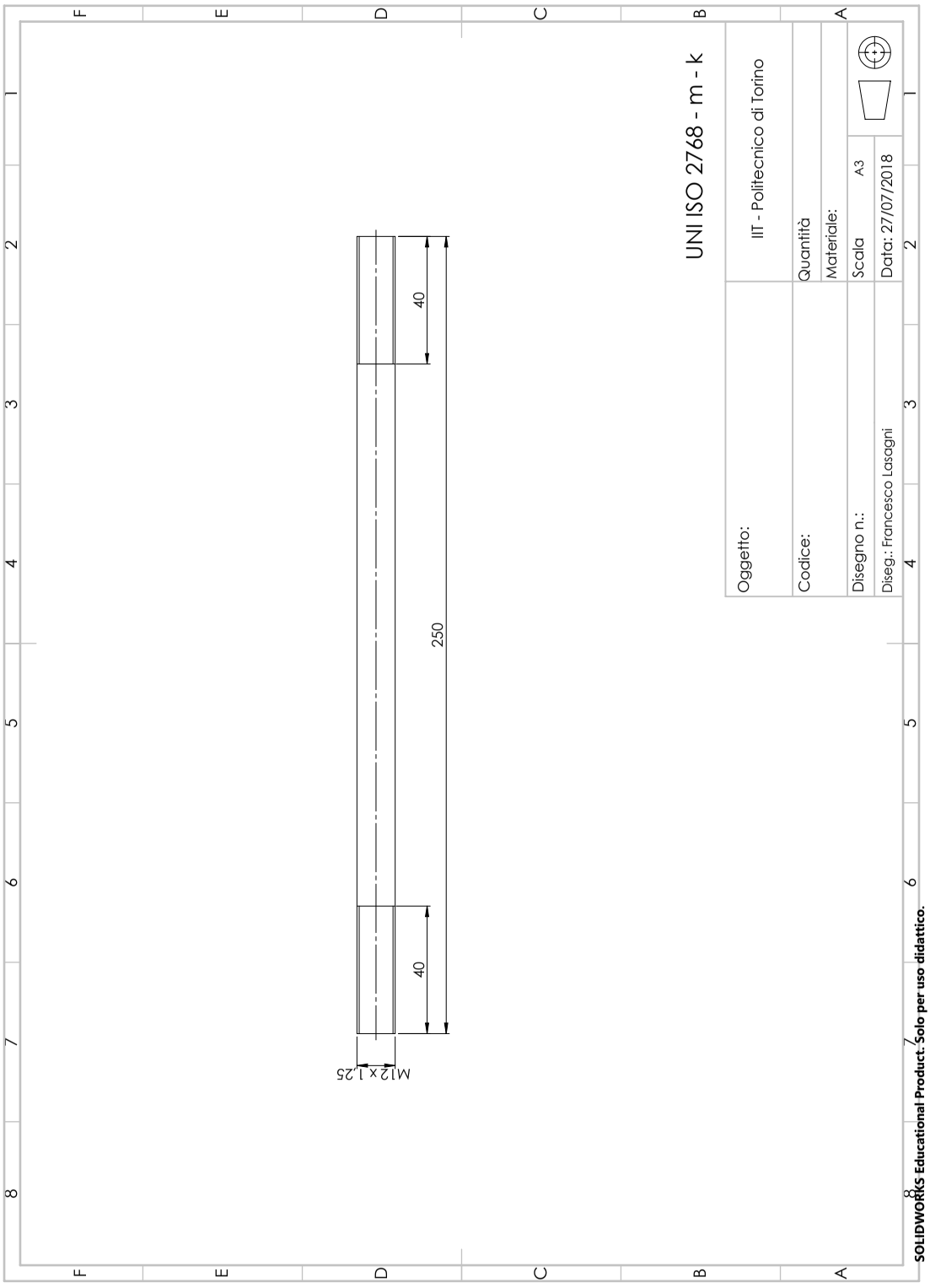

UNI ISO 2768 - m - k
IIT - Politecnico di Torino
Oggetto:
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Data: 27/07/2018
Diseg.: Francesco Lasagni
40
250
40
M12 x 1,25
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.8

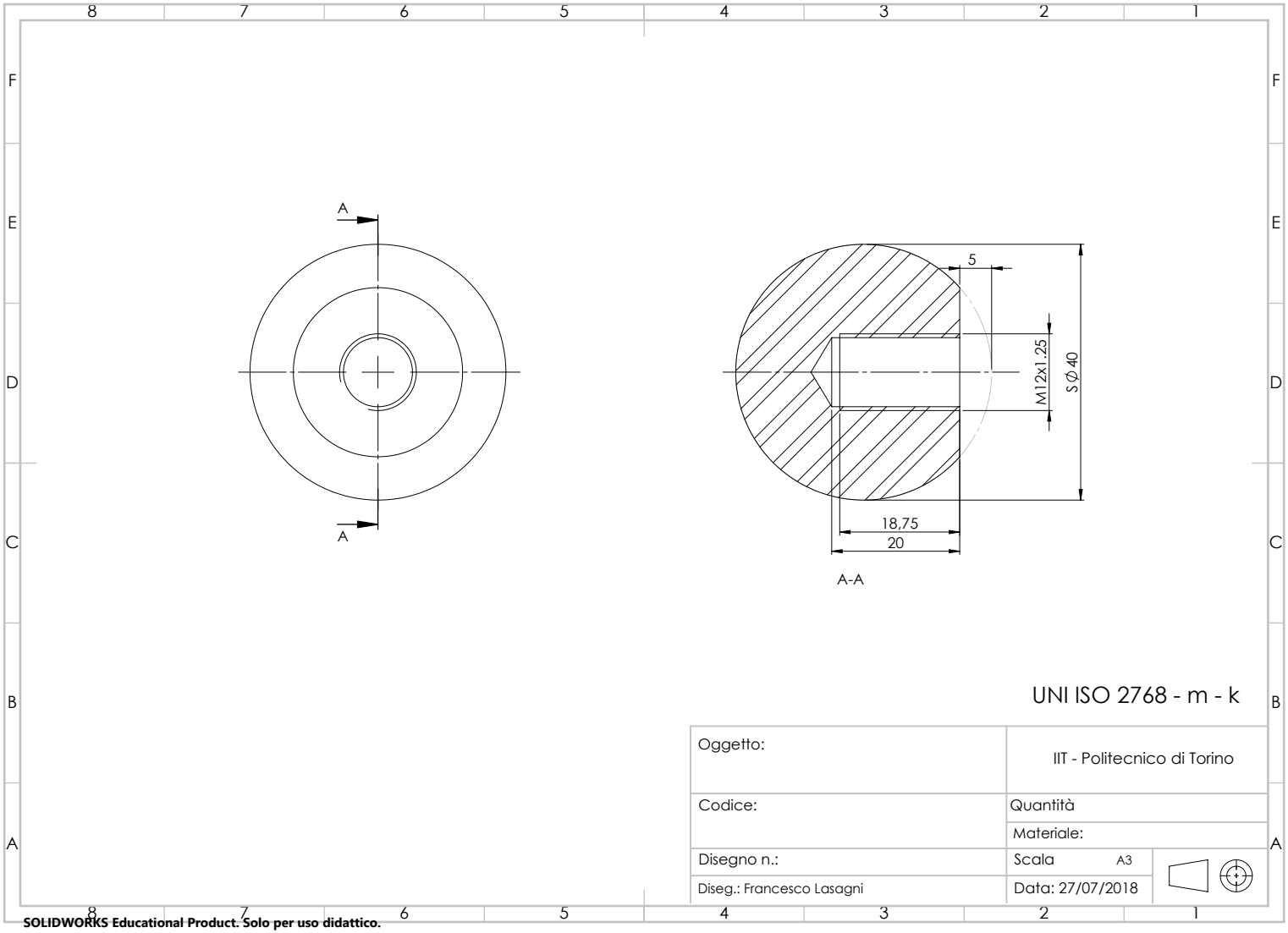

A
A
5
M12x1.25
S Ø 40
18,75
20
A-A
UNI ISO 2768 - m - k
Oggetto:
IIT - Politecnico di Torino
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.9

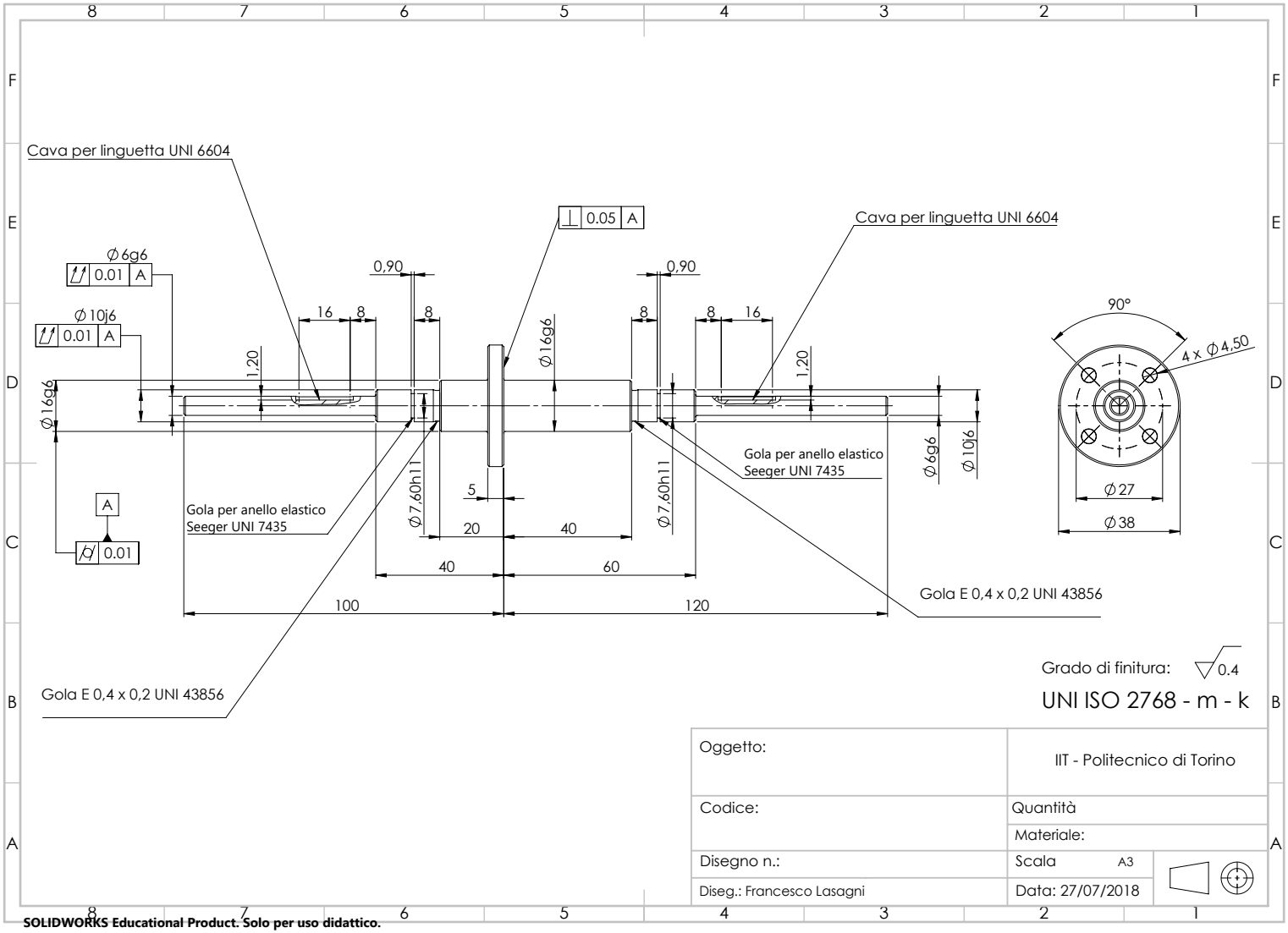

Cava per linguetta UNI 6604
Cava per linguetta UNI 6604
⌀6g6
0.01 A
⌀10j6
0.01 A
0.05 A
0,90
0,90
16
8
8
8
8
16
1,20
1,20
⌀16g6
⌀16g6
⌀7,60h11
⌀7,60h11
5
20
40
40
60
100
120
Gola per anello elastico Seeger UNI 7435
Gola per anello elastico Seeger UNI 7435
Gola E 0,4 x 0,2 UNI 43856
Gola E 0,4 x 0,2 UNI 43856
⌀6g6
⌀10j6
90°
4 x ⌀4,50
⌀27
⌀38
A
0.01
Grado di finitura: 0.4
UNI ISO 2768 - m - k
Oggetto:
IIT - Politecnico di Torino
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.10

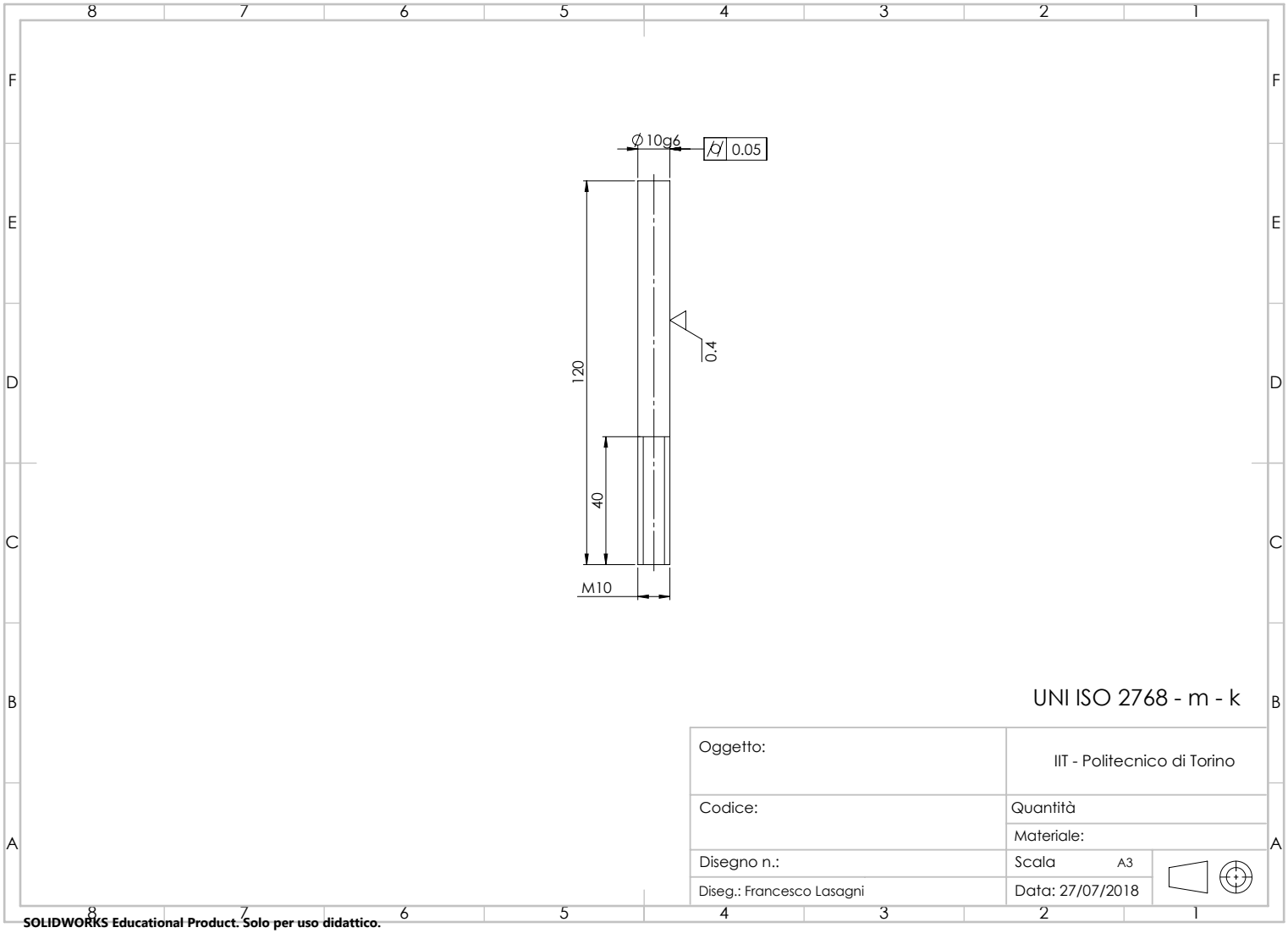

Ø 10g6
0.05
120
0.4
40
M10
UNI ISO 2768 - m - k
Oggetto:
IIT - Politecnico di Torino
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.11

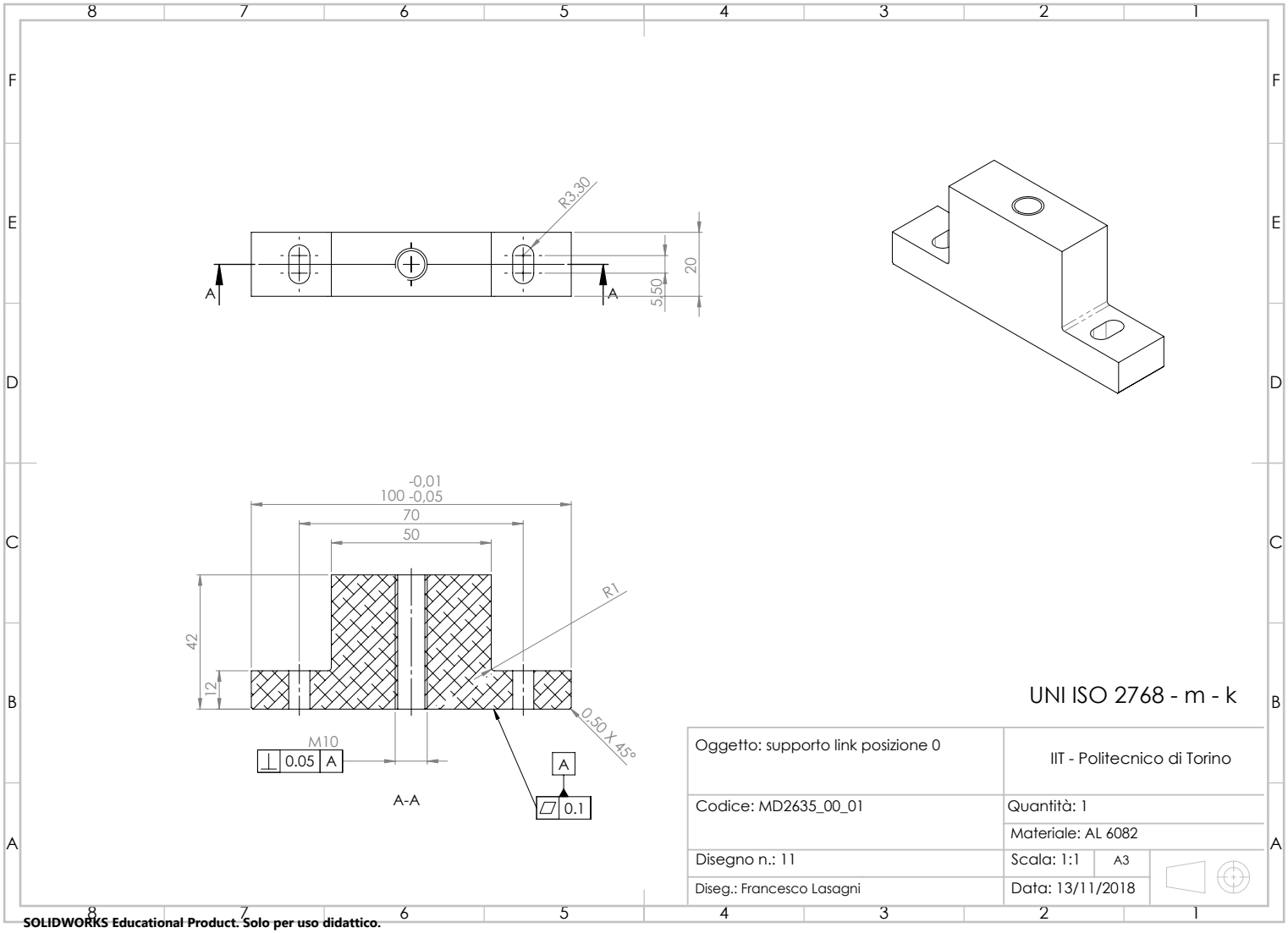

R3,30
20
5,50
A
A
-0,01
100 -0,05
70
50
R1
42
12
0.50 X 45°
M10
0.05 A
A
0.1
A-A
UNI ISO 2768 - m - k
Oggetto: supporto link posizione 0
IIT - Politecnico di Torino
Codice: MD2635_00_01
Quantità: 1
Materiale: AL 6082
Disegno n.: 11
Scala: 1:1
A3
Diseg.: Francesco Lasagni
Data: 13/11/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.12

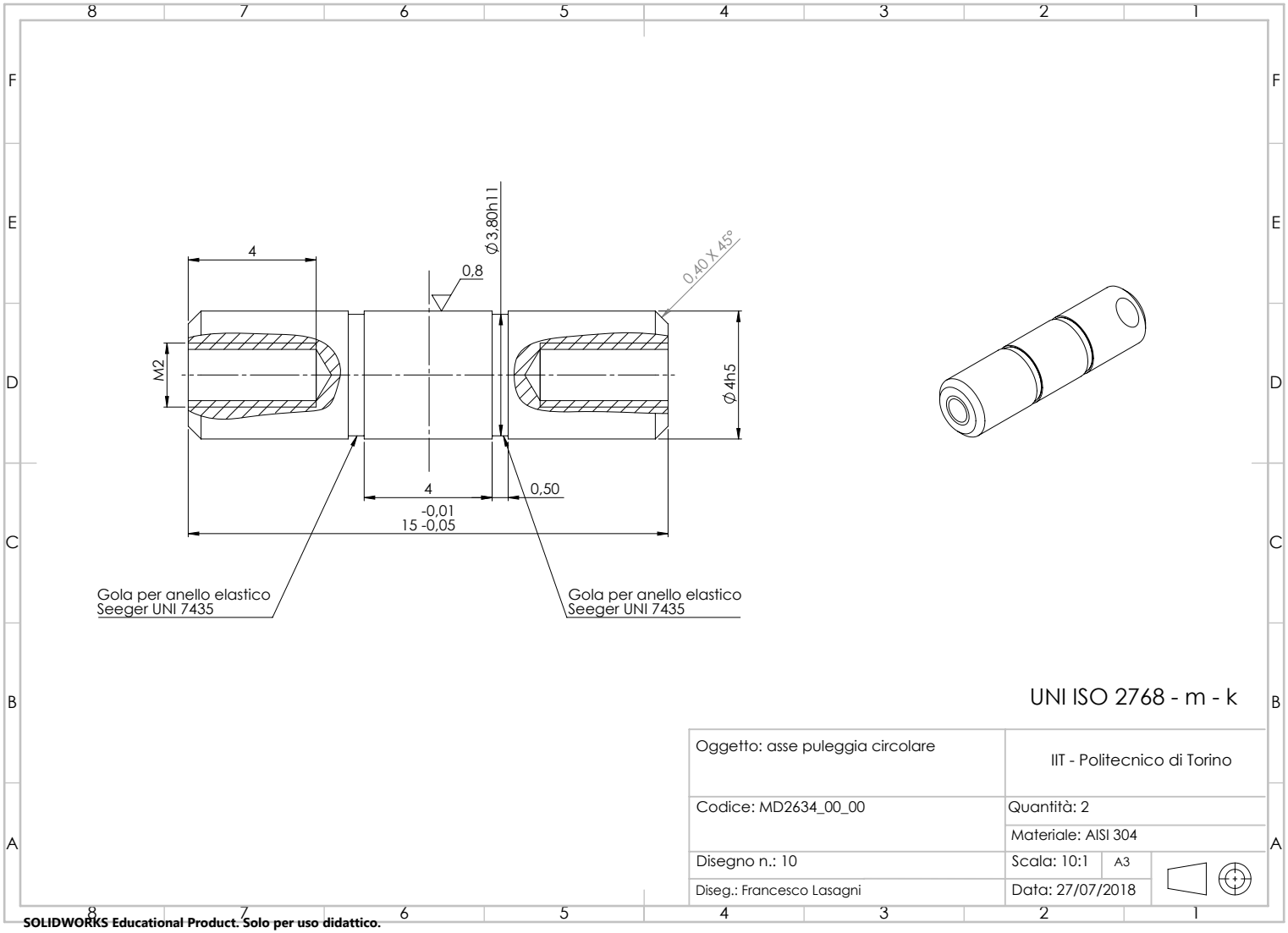

4
0,8
Ø3,80h11
0,40 X 45°
M2
Ø4h5
4
0,50
-0,01
15 -0,05
Gola per anello elastico
Seeger UNI 7435
Gola per anello elastico
Seeger UNI 7435
UNI ISO 2768 - m - k
Oggetto: asse puleggia circolare
IIT - Politecnico di Torino
Codice: MD2634_00_00
Quantità: 2
Materiale: AISI 304
Disegno n.: 10
Scala: 10:1
A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.13

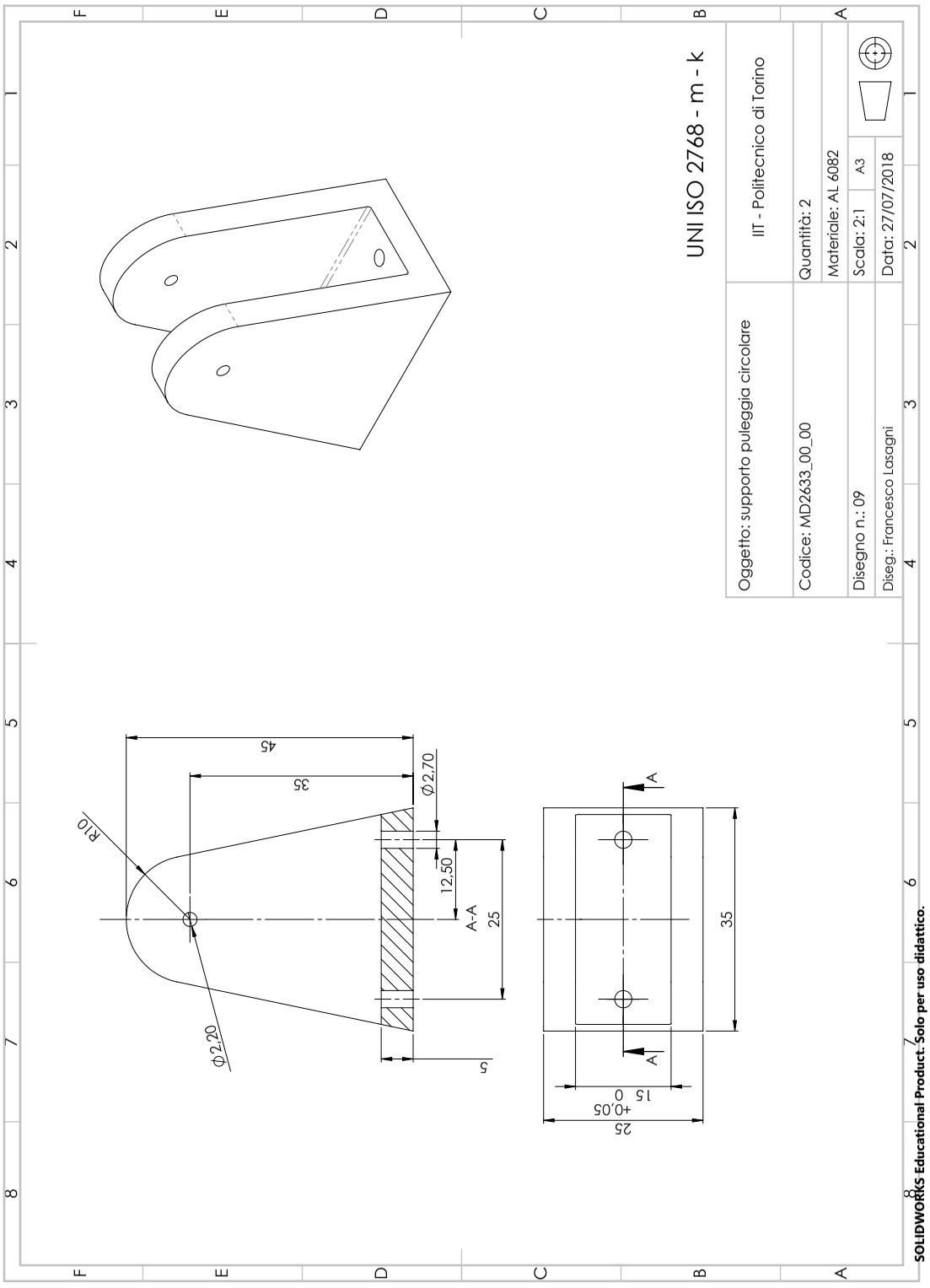

UNI ISO 2768 - m - k
IIT - Politecnico di Torino
Oggetto: supporto puleggia circolare
Codice: MD2633_00_00
Quantità: 2
Materiale: AL 6082
Disegno n.: 09
Scala: 2:1
A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
45
35
Φ2,70
12,50
A-A
25
R10
Φ2,20
5
35
25 +0,05 15 0
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.14

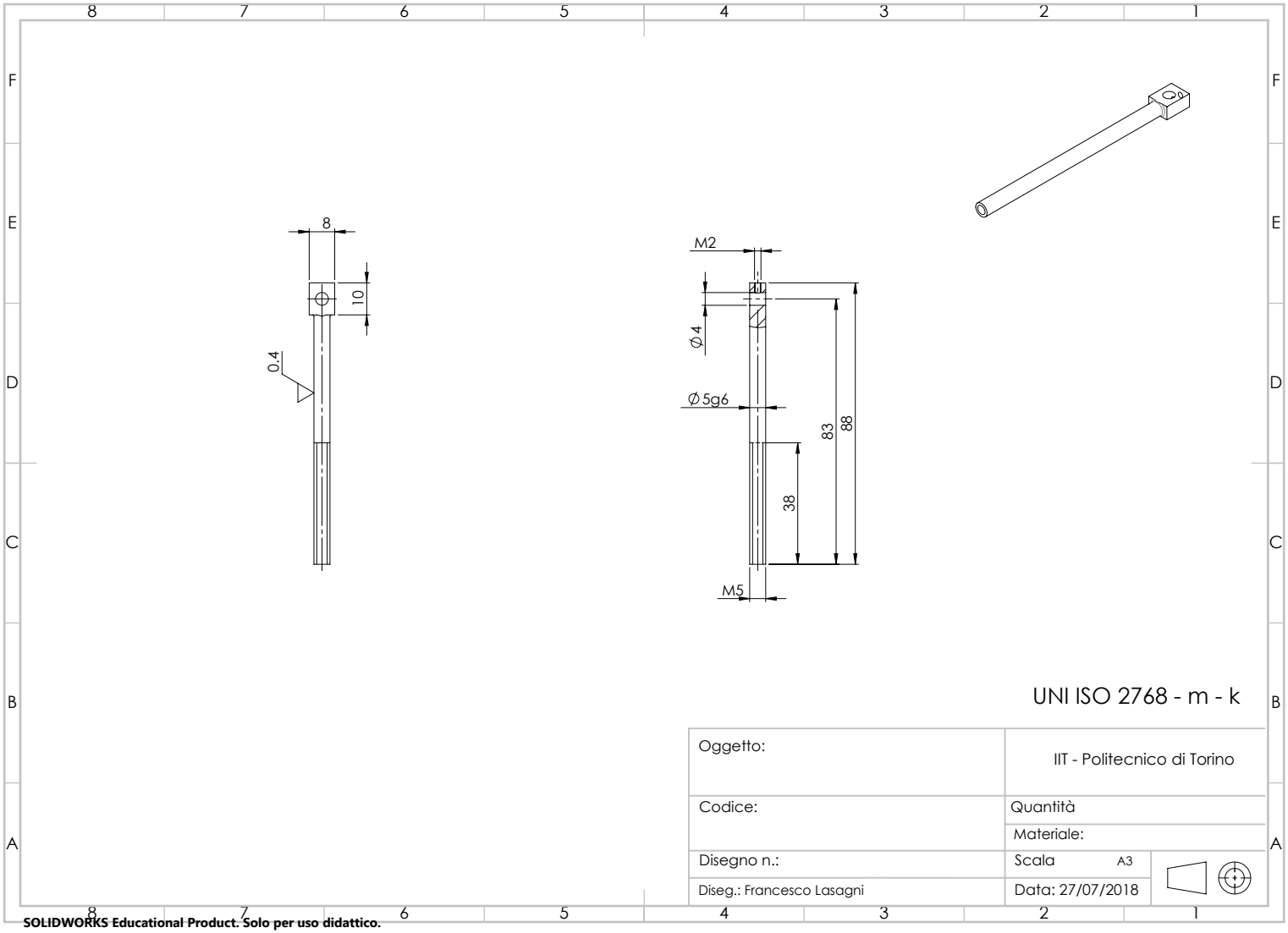

8
10
0.4
M2
Ø4
Ø5g6
83
88
38
M5
UNI ISO 2768 - m - k
Oggetto:
IIT - Politecnico di Torino
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.15

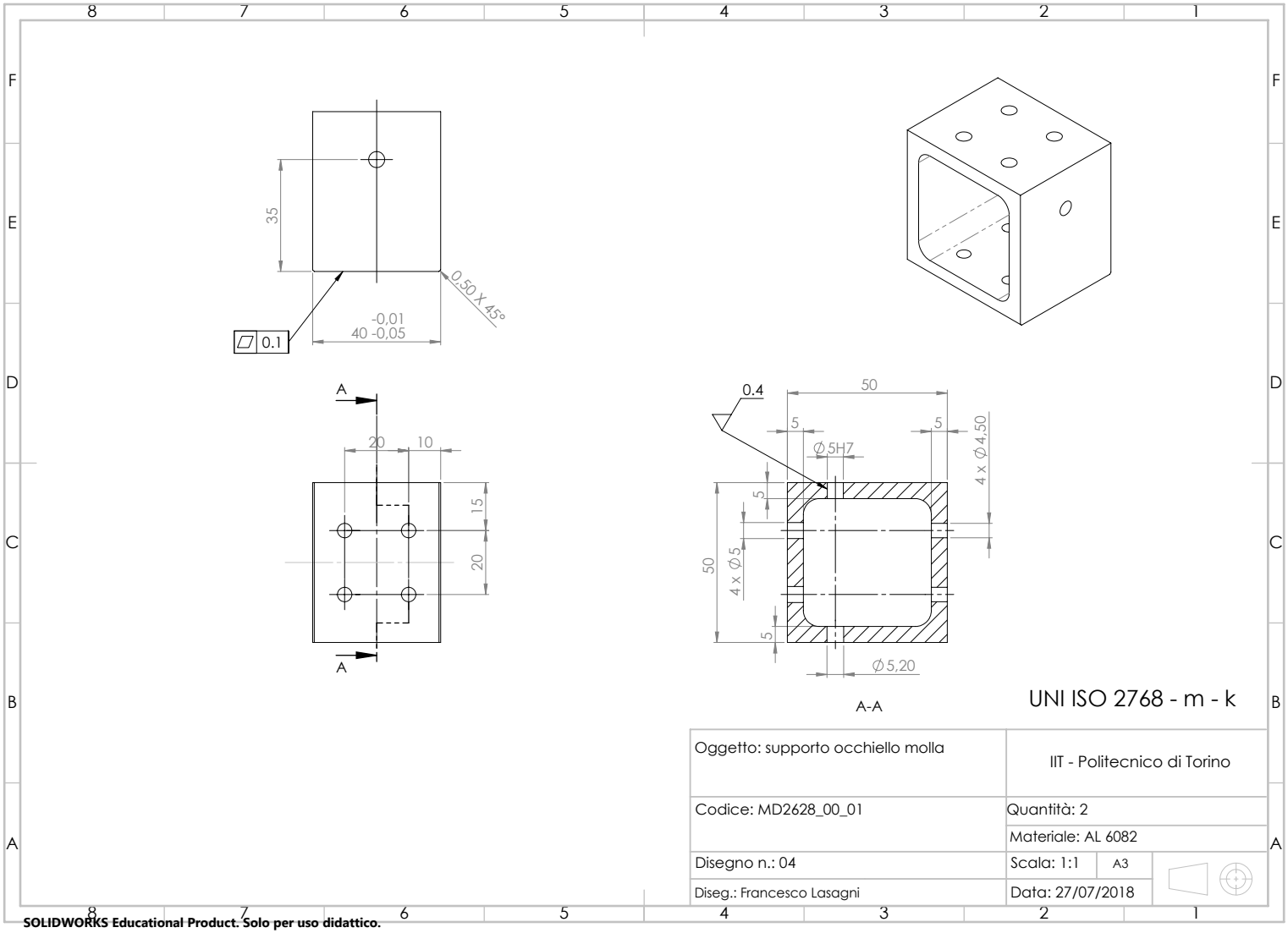

35
0.50 X 45°
-0,01
40 -0,05
0.1
A
20
10
15
20
A
0.4
50
5
5
Ø5H7
4 x Ø4,50
5
50
4 x Ø5
5
Ø5,20
A-A
UNI ISO 2768 - m - k
Oggetto: supporto occhiello molla
IIT - Politecnico di Torino
Codice: MD2628_00_01
Quantità: 2
Materiale: AL 6082
Disegno n.: 04
Scala: 1:1
A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.16

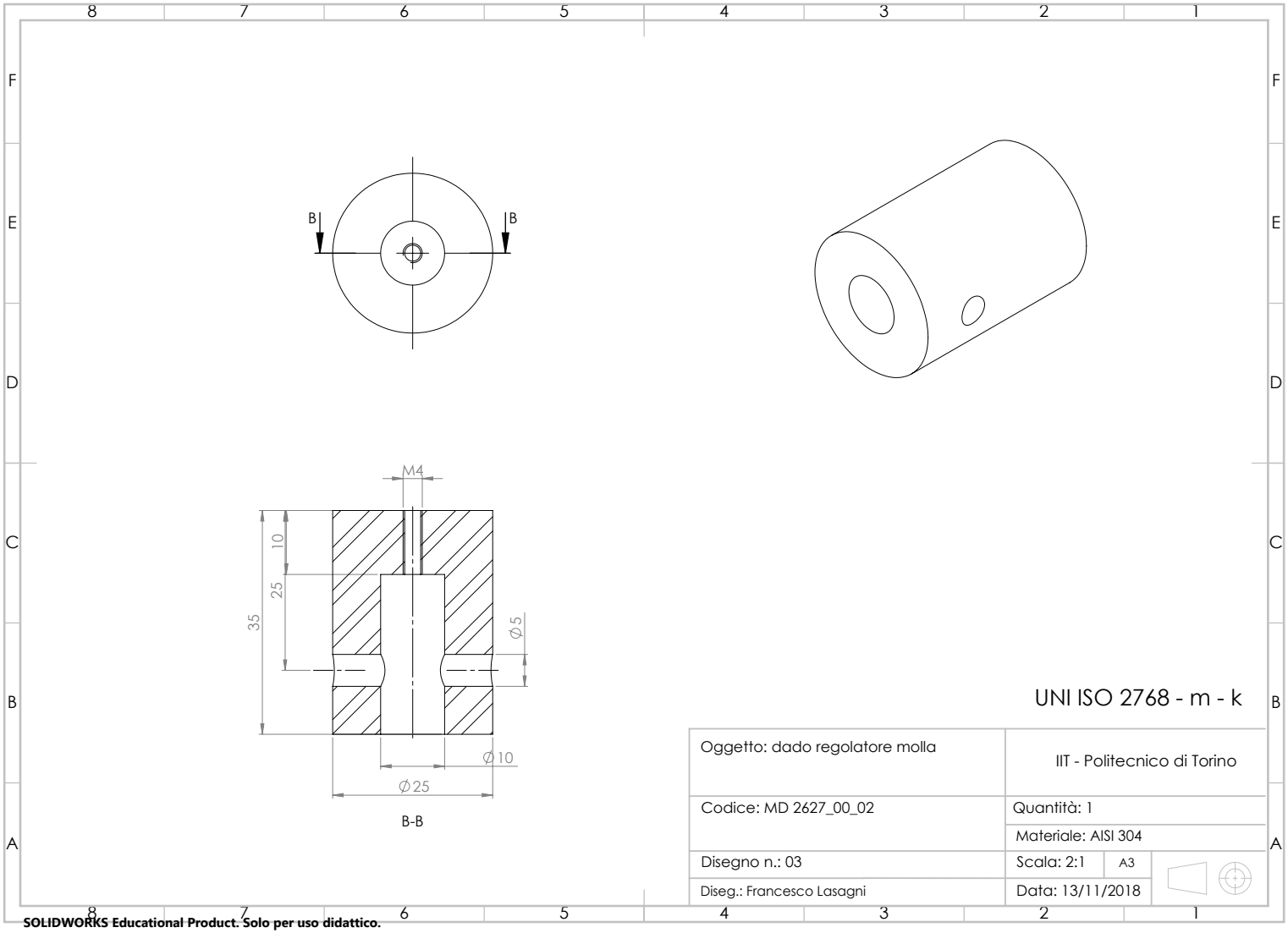

B
B
M4
10
25
35
Ø5
Ø10
Ø25
B-B
UNI ISO 2768 - m - k
Oggetto: dado regolatore molla
IIT - Politecnico di Torino
Codice: MD 2627_00_02
Quantità: 1
Materiale: AISI 304
Disegno n.: 03
Scala: 2:1
A3
Diseg.: Francesco Lasagni
Data: 13/11/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.17

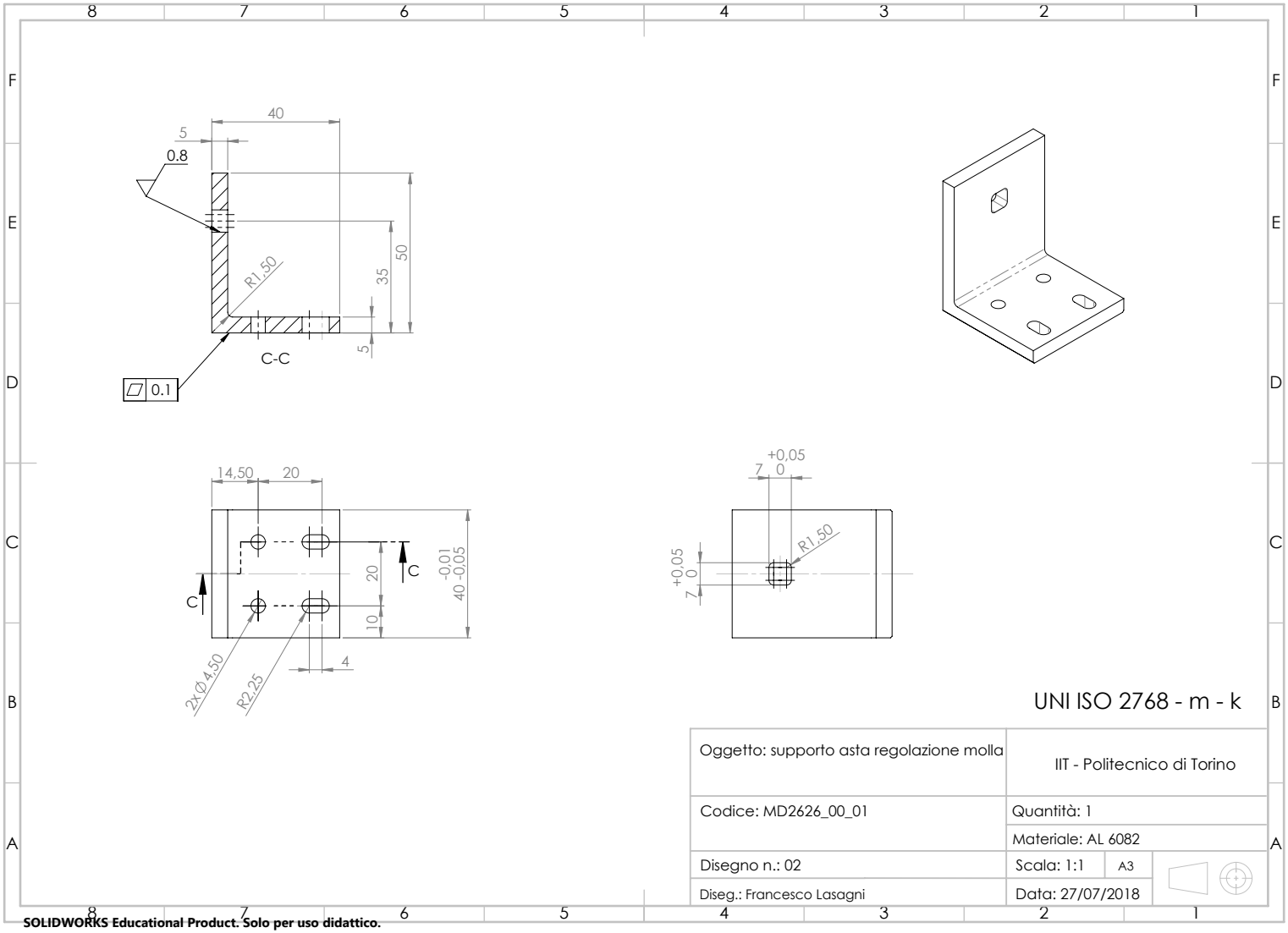

40
5
0.8
R1.50
35
50
5
C-C
0.1
14,50
20
20
10
4
C
C
-0,01
40 -0,05
2xØ4,50
R2,25
+0,05
7 0
+0,05
7 0
R1.50
UNI ISO 2768 - m - k
Oggetto: supporto asta regolazione molla
IIT - Politecnico di Torino
Codice: MD2626_00_01
Quantità: 1
Materiale: AL 6082
Disegno n.: 02
Scala: 1:1
A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.

Figure C.18

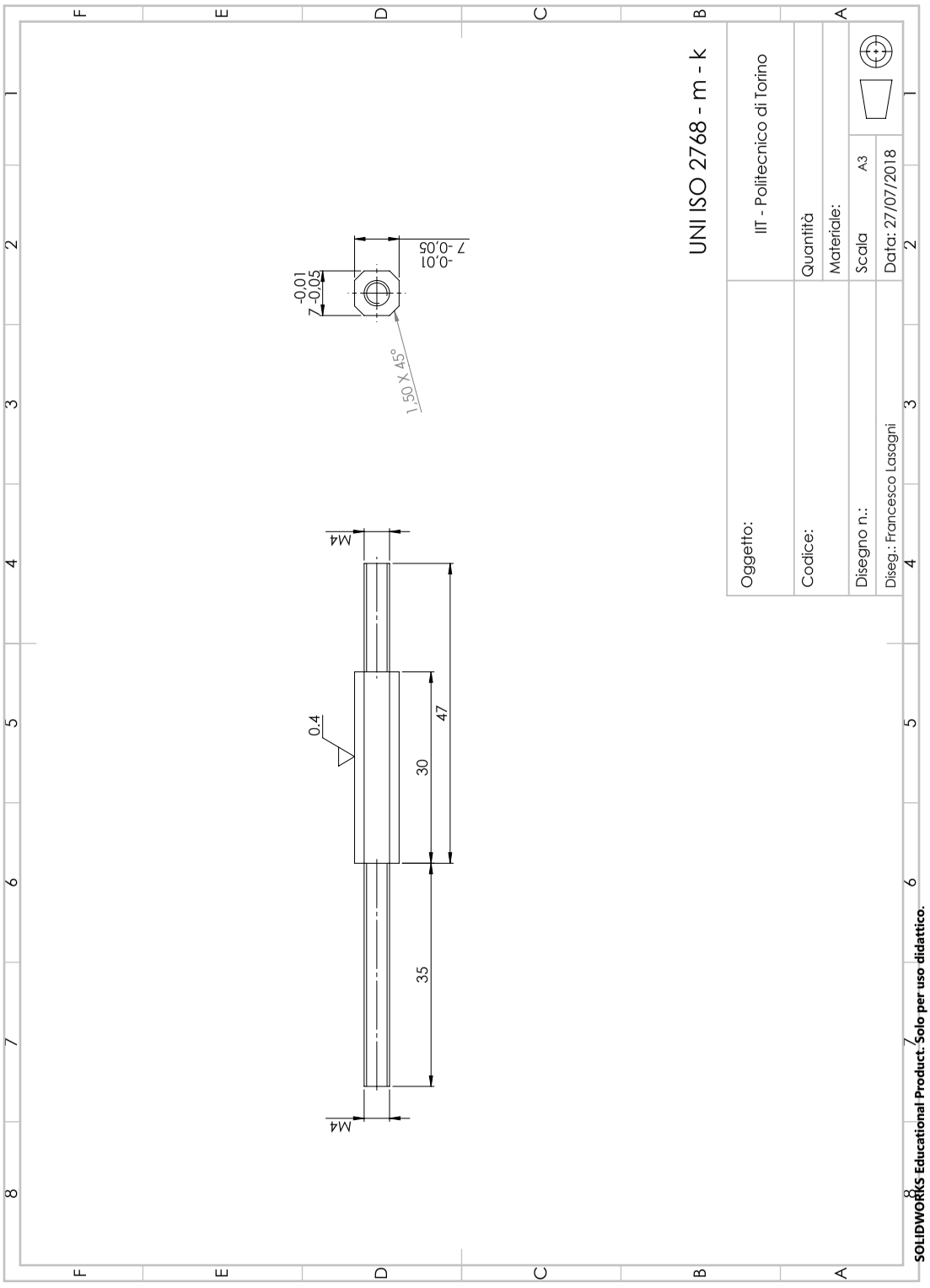

UNI ISO 2768 - m - k
IIT - Politecnico di Torino
Oggetto:
Codice:
Quantità
Materiale:
Disegno n.:
Scala A3
Diseg.: Francesco Lasagni
Data: 27/07/2018
SOLIDWORKS Educational Product. Solo per uso didattico.
M4
M4
0.4
47
30
35
1.50 X 45°
7 -0.05 -0.01
-0.01 -0.05

Figure C.19

Bibliography

[1] B. Kim and A. D. Deshpande, "Design of nonlinear rotational stiffness using a noncircular pulley-spring mechanism," *Journal of Mechanisms and Robotics*, vol. 6, no. 4, p. 041009, 2014.

[2] J. Wallén, *The history of the industrial robot*. Linköping University Electronic Press, 2008.

[3] V. Arakelian, "Gravity compensation in robotics," *Advanced Robotics*, vol. 30, no. 2, pp. 79–96, 2016.

[4] Y. Tojo, P. Debenest, E. F. Fukushima, and S. Hirose, "Robotic system for humanitarian demining," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 2025–2030.

[5] A. Rosyid, B. El-Khasawneh, and A. Alazzam, "Gravity compensation of parallel kinematics mechanism using torsional springs based on potential energy optimization."

[6] J. L. Herder, *Energy-free Systems. Theory, conception and design of statically*, 2001, vol. 2.

[7] N. Ulrich and V. Kumar, "Passive mechanical gravity compensation for robot manipulators," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1536–1541.

[8] I. Simionescu and L. Ciupitu, "The static balancing of the industrial robot arms: Part ii: Continuous balancing," *Mechanism and machine theory*, vol. 35, no. 9, pp. 1299–1311, 2000.

[9] A. Kondrin, L. Petrov, and N. Polishchuk, "Pivoted arm balancing mechanism," *SU Patent*, vol. 1, 1990.

[10] J. Boisclair, P.-L. Richard, T. Laliberté, and C. Gosselin, "Gravity compensation of robotic manipulators using cylindrical halbach arrays," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 457–464, 2017.

[11] M. D'Imperio, D. Ludovico, C. Pizzamiglio, C. Canali, D. Caldwell, and F. Cannella, "Flegx: A bioinspired design for a jumping humanoid leg," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 3977–3982.

[12] P.-H. Kuo and A. D. Deshpande, "Muscle-tendon units provide limited contributions to the passive stiffness of the index finger metacarpophalangeal joint," *Journal of biomechanics*, vol. 45, no. 15, pp. 2531–2538, 2012.

[13] G. Endo, H. Yamada, A. Yajima, M. Ogata, and S. Hirose, "A passive weight compensation mechanism with a non-circular pulley and a spring," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3843–3848.

[14] N. Schmit and M. Okada, "Synthesis of a non-circular cable spool to realize a nonlinear rotational spring," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 762–767.

[15] A. J. McPhate, "Function generation with band mechanisms," *Journal of Mechanisms*, vol. 1, no. 1, pp. 85–94, 1966.

[16] *Adams/Solver help - Adams 2014*, 2014.