UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DESIGN OF A BOSTON DYNAMICS 'HANDLE' ROBOT PROTOTYPE WITH RECURDYN AND MATHEMATICA

AUTHOR: GUILLERMO OLIVER PEIRÓ

SUPERVISOR: JOSÉ LUIS OLIVER HERRERO

Academic year: 2017-18

Agradecimientos

"La realización de este trabajo fin de máster no habría sido posible sin el apoyo y consentimiento de algunas personas. Quiero agradecer a mi padre y tutor por haberme ayudado a lo largo del desarrollo del proyecto. Por supuesto, también darle las gracias a mi novia por todo lo que ha tenido que aguantar, sobre todo en las últimas etapas de desarrollo del proyecto y de la redacción de la memoria, que le ha tenido en vilo hasta el final. Por último, darle las gracias a mi familia, a mis compañeros, y a toda la gente que he conocido durante estos últimos años mientras cursaba el grado y el máster; no habría sido lo mismo sin vosotros."

Abstract

A simplified 2-dimensional model of *Boston Dynamics' Handle* robot will be developed using industry-leading computational tools.

As a starting point, a virtual model will built and designed in the *SolidWorks* environment. Using *COSMOSMotion* add-in, the model will be adequately modified in order to obtain a self-aligned mechanical system with no redundant constraints. *Mathematica* with the *MechanicalSystems* package will provide the necessary tools to build a mathematical model and obtain its kinematic solution, obtaining the expressions of all the generalized coordinates and the inertial properties of the model in terms of its degrees of freedom.

The dynamic equations of motion of the model will be obtained by reducing the model to an inverted pendulum on cart system. The resulting nonlinear model will be linearized around the equilibrium point and closed-loop control schemes will be developed to maintain the balance of the system. PID and state feedback controllers will be developed using pole placement and LQR techniques. The designed controllers will be implemented in *RecurDyn* and *SystemModeler* to compare their performance and the resemblance of the results in both multibody dynamics simulation software.

Keywords: robot, robotics, kinematics, control, dynamics, CAD, CAE, *SolidWorks*, *Mathematica*, *RecurDyn*, *SystemModeler*, *Boston Dynamics*, *Handle*.

Resumen

Un modelo bidimensional simplificado del robot *Handle* de *Boston Dynamics* será desarrollado utilizando herramientas computacionales de amplia aplicación en la industria.

Como punto de partida, se ensamblará un modelo virtual en el entorno de *SolidWorks*. Utilizando el complemento *COSMOSMotion*, el modelo se modificará para obtener un sistema mecánico autoalineador sin restricciones en exceso. *Mathematica* y el complemento *MechanicalSystems* proporcionarán las herramientas necesarias para desarrollar un modelo matemático del cual se podrá obtener la solución cinemática, obteniendo las expresiones de las coordenadas generalizadas y las propiedades inerciales del modelo en función de sus grados de libertad.

Las ecuaciones dinámicas del movimiento del modelo se obtendrán asemejando el modelo a un sistema de péndulo invertido sobre carro. El sistema no lineal obtenido se linearizará alrededor del punto de equilibrio y esquemas de control en bucle cerrado se desarrollarán para mantener el equilibrio del sistema. Controladores PID y de retroalimentación de estados se desarrollarán utilizando técnicas de colocación de polos y LQR. Los controladores diseñados se implementarán en *RecurDyn* y *SystemModeler* para comparar su funcionamiento y la similitud de los resultados en las dos herramientas computacionales de simulación dinámica multicuerpo.

Palabras Clave: robot, robótica, cinemática, control, dinámica, CAD, CAE, *SolidWorks*, *Mathematica*, *RecurDyn*, *SystemModeler*, *Boston Dynamics*, *Handle*.

Contents

List of Figures

Part I

Descriptive Memory

Chapter 1

Introduction

This first chapter will describe the robot this thesis is based on, the different computational tools used in the development of the prototype and the motivation behind all of this.

Robotics has been known in the industry for decades. Industrial robotics for the automation of production lines is a field that has undergone a great amount of development and research in the past. This allowed an increase in the performance of the industry and to free workers from tedious and repetitive jobs with high physical load. Currently, there is a new trend in robotics, which has been stomping hard over the last two decades: social robotics. Social robotics takes a leap away from industrial facilities and develops robots that are able to interact with people and help in the fulfillment of daily activities. Following the line of development of the virtual assistants, through this type of robotics the aim is to make an assistant, this time a tangible one, which is able to render services in the physical world that go way beyond what a cybernetic organism projected on a screen can offer.

A long time has gone by since the presentation by Honda of the humanoid robot *ASIMO* which fascinated the world in the early 2000s. Nowadays, companies such as *Boston Dynamics* or *Agility Robotics* are developing social robotics that only a few years back seemed only possible to conceive in our imagination or in science fiction movies.

The key to success in social robotics projects, and to that extent in any kind of robotics projects, is not easy to obtain. These multidisciplinary projects require of the successful interaction of several fields of engineering. A good knowledge of mechanical design is mandatory in order for the robot to be agile and move efficiently. The design of the associated electronics is also extremely important; power electronics is required to supply the energy needed for motors and actuators, and digital electronics is needed to process information from the outside world and act accordingly. Robust control loops must be designed in order for the robot to maintain its balance and have a dynamic stable behavior. In sum, the development of a social robot depends on the correct synergy between mechanics, electronics, and control theory.

1.1 *Boston Dynamics Handle*

The *Handle* robot was officially introduced in February 27, 2017 when *Boston Dynamics* uploaded a new video titled *Introducing Handle*[1] to their YouTube channel. There was some prior video footage leaked at the end of January when the company founder, Marc Raibert, featured the robot in a presentation to investors and some of those present recorded and uploaded videos to the Internet.

Handle is presented as the best of both worlds, it combines wheels and legs to provide improved movement and agile high-strength mobile manipulation. This research robot stands 2 meters tall, it can travel at speeds of 14.5 kilometers per hour, and jump as high as 1.2 meters vertically. Its main source of energy is electric power, used to operate both electric and hydraulic actuators, with a range of about 15 miles on one battery charge. With a total weight of 105 kg, *Handle* is able to carry a payload of 45 kg. Using depth cameras and computer vision algorithms, it can detect and avoid or even jump over obstacles while moving at relatively high speeds.

According to *Boston Dynamics*, *Handle* uses many of the same dynamics, balance and mobile manipulation principles found in the quadruped and biped robots they build in the past. In this case, the robot has only 10 actuated joints, making it significantly less complex than the previous robots that walk on two legs. The combination of wheels, which are efficient on flat surfaces, and legs, that can go almost anywhere, allow *Handle* to achieve a high mobility and performance in all types of terrains. *Handle* can pick up heavy loads while occupying a small footprint, allowing it to maneuver in tight spaces.

Figure 1.1: Image of the *Handle* robot obtained from the *Boston Dynamics* website[2].

[1] *Boston Dynamics* - **Introducing Handle**: https://www.youtube.com/watch?v=-7xvqQeoA8c
[2] *Boston Dynamics* - **Handle**: https://www.bostondynamics.com/handle

1.2 Computational tools

For the development of this project, several computer aided engineering softwares have been used. A brief presentation of the software and its relevance to the project will be presented below.

1.2.1 Mathematica

Mathematica[3] is a modern technical computing software developed by *Wolfram Research*, an American computer software company, widely used in many technical, scientific, engineering, mathematical, and computer science fields. First introduced in 1988, what makes *Mathematica* unique is its symbolic manipulation kernel, which allows algebraic equation manipulation and symbolic equation solving in terms of explicit variables which can be parametrized or substituted accordingly. It the latest releases of the software many new features have been introduced, including neural networks, machine learning, image processing, Big Data algorithms, 3D visualizations, and many more. All the programming and development in the environment of *Mathematica* is done in *Wolfram Language*, which include thousands of built-in functions developed in the last thirty years.

Mathematica has made possible to express the equations of mechanical motion in terms of different parameters and solve the systems of algebraic equations in order to obtain the symbolic solution for the position and orientation of each body in the mechanical system. The resulting equations, expressed in terms of the system's degrees of freedom and with geometrical parameters of the bodies yet to be determined, can be simplified by substituting the numerical values for the dimensions of the bodies and by applying the convenient *Mathematica* simplification functions.

The symbolic capabilities of *Mathematica* will aid in obtaining the dynamic equations of motion of the system in order to design a controller using two different closed-loop configurations: a classical PID controller and a full state feedback control. The design of these controllers will be done in *Mathematica*, but their performance will be checked with the *RecurDyn* and *SystemModeler* software.

1.2.2 SolidWorks

SolidWorks[4] is a well-known computer-aided engineering (CAE) software for the Windows operating system that is widely used in the industry. Even though it also contains modules for dynamic multibody and electrical circuits simulation, *SolidWorks* is mostly used for product and prototype design as it provides one of the most intuitive and easy to use environments from all the computer-aided design softwares available. The developer behind *SolidWorks* is *Dassault Systèmes*, an European software company with offers other industry leading computer-aided engineering (CAE) and finite element method (FEM) software such as *CATIA* or *ABAQUS*.

The selected version of the *SolidWorks* software is the 2007 edition with the *COSMOSMotion* add-in, a dynamic multibody simulation module. *COSMOSMotion* offers the possibility for the designed system to be simulated in real-world operating conditions, including the mechanical

[3] *Wolfram Research* - **Mathematica**: https://www.wolfram.com/mathematica
[4] *Dassault Systèmes* - **SolidWorks**: https://www.solidworks.com

operations of actuators and motors and the physical forces generated. The version selection is due to the fact that the 2007 edition is the last one to include some key features in the *COSMOSMotion* add-in.

The simplified 2-dimensional model of the *Handle* robot has been developed in *SolidWorks*, making use of parametric equations in order to define the lengths and positions of the different bodies of the mechanical system. Using the tools provided in this CAD software, the complete assembly of the model was developed with the appropriate mechanical joints needed to assemble the model together and making sure the mobility of the model was as expected. The assembled model's mobility is checked using *COSMOSMotion* and its motion will be simulated using this add-in. The theory of self-aligning mechanisms will be taken into account in order to obtain a mechanical configuration with no redundant constraints. The results of the simulation provide a starting point for the dynamics of the model, these results are to be checked in the next phases of the project.

1.2.3 RecurDyn

RecurDyn[5], blended word originating from *Recur*sive and *Dyn*amics, is a computer-aided engineering (CAE) software centered in the dynamic simulation of multibody mechanical systems (MBD). This software is able to simulate rigid and flexible body dynamics by using a combination of traditional rigid multibody algorithms with cutting-edge finite element methodology for modeling of flexible bodies. The developer of *RecurDyn* is *FunctionBay*, a software company based in South Korea, which also offers an add-on multibody dynamics (MBD) module for ANSYS and it is considered a reference in the field of multibody simulation. The computed-aided design capabilities of *RecurDyn* are not as powerful as the ones available in *SolidWorks*, but it also includes an integrated control design and simulation tool, known as *CoLink*, as well as a particle dynamics module. *RecurDyn* also supports co-simulation with various other computer-aided engineering software tools, such as *MATLAB/Simulink* and provides FMI (Functional Mockup Interface) for interaction with compatible software.

By successfully exporting the model from *SolidWorks* using the *Parasolid* binary file format, a tantamount model was obtained when importing in *RecurDyn*. Using the controller design capabilities of *CoLink*, providing feedback from the own CAD model, different closed-loop controllers which allowed the robot to maintain its balance were developed.

1.2.4 SystemModeler

SystemModeler[6] is a modeling and simulation environment based on the *Modelica*[7] language. It was originally developed by the company *MathCore Engineering* under the commercial name of *MathModelica*, but it was acquired by *Wolfram Research* on 2011. Since its acquisition by *Wolfram Research* it has been re-branded as *SystemModeler* and includes a tight integration with their flag-ship software *Mathematica*.

[5] *FunctionBay* - **RecurDyn**: https://www.functionbay.org

[6] *Wolfram Research* - **SystemModeler**: https://www.wolfram.com/system-modeler

[7] *Modelica* is a non-proprietary, object-oriented, multi-domain equation based language to conveniently model complex physical systems (containing mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents, etc.), developed by the *Modelica Association*. https://www.modelica.org

SystemModeler offers a large selection of modeling libraries, including the *Modelica Standard Library*, as well as other self-developed components by *Wolfram Research*, in an intuitive drag-and-drop graphical environment in order to build multidomain models of complete real-world systems. By adding the power of the *Wolfram Language* it provides a fully integrated environment for analyzing, understanding and quickly iterating system designs with a focus on insight, innovation and results. As a new addition other *Modelica*-based software, it also provides a graphical representation of mechanical models when properly defined using supported CAD file formats.

A complete replica of the mechanical system was built in the *Modelica* language, using elements from the *Modelica Standard Library* and other own custom-built components. The bodies from the CAD model were exported as *STL* files[8] in order to provide a graphical representation for the *SystemModeler* simulation results. Using the equations obtained from the *Mathematica* kinematic solution and dynamic model, a closed-loop controller which allowed the robot to maintain its balance was also developed. The results of this simulation are to be compared to the results obtained from the completely numerical simulation of *RecurDyn*.

1.3 Motivation

The realization of the project responds to academic reasons, more specifically to obtain the Master's degree in Industrial Engineering. The choice of the subject is the result of an interest in robotics, in addition to the motivation that implies the desire for learning and the satisfaction of self-realization when something is completed successfully. For my Bachelor's final year project I had the chance of correcting the deficiencies in electronics and reprogramming the embedded systems of an existing robot, and now the challenge consists in tackling a robot design from an earlier stage.

I am sure that *Boston Dynamics'* creations intrigue and fascinate the whole world each time they publish a new video, and I am no exception. Since the first time I saw the *Handle* robot I was amazed by the combination of wheels and legs it used, and I thought it would be interesting to analyze the mechanical configuration behind its motion capabilities. The fact that it used wheels instead of legs made it easier to extract a simplified virtual model of it, but it was still interesting enough for it to be analyzed using the latest computational tools in order to get a glance at the technology enclosed within.

[8] Stereolithography (**.stl*) neutral file format in which three-dimensional bodies are described as their external surface using triangular tessellation with no information about their colors, textures, mass, or any other properties other than their external shape.

Chapter 2

Objectives

The objectives deriving from the fulfillment of this project are the following:

- Obtaining the Master's Degree in Industrial Engineering.
- Acquiring competences in the usage of computational design tools with wide professional application in the field of mechanical engineering.
- Making a first contact with the mechanical and dynamic control design of mobile robots.

Chapter 3

Methodology

This chapter contains the bulk of the project: the development of the different parts will be described, including the methodology and the computational tools behind all the milestones of the project.

3.1 Simplified 2-dimensional model in *SolidWorks*

3.1.1 Theoretical model development

The first step in the development of the project is to carefully analyze the only video footage available for the *Handle* robot, the *Introducing Handle* video at *Boston Dynamics'* YouTube channel, in order to extract all the relevant information about the mechanical systems behind its motion capabilities. A thorough analysis will be carried out to determine the mechanical configuration which allows the relative motion between the different bodies that make up the robot. For the sake of reducing the complexity of the robot analysis to a dimension encompassable by a Master's thesis, considering the limitations in time and length which apply, the objective is to obtain a simplified 2-dimensional model prototype which retains most of the characteristics of its 3-dimensional real counterpart.

Following the conventions of mechanical system analysis, considering the robot as an mechanical entity on a 2-dimensional world, the ground is the first body of the system. It is a fixed body over which the motion of the robot will be defined and it will provide a the surface of mechanical sustenance for it. The wheel is the second body of the system, a relative rolling motion without slipping between it and the ground is allowed. The next two bodies are what make up the leg, being body number three the one between the wheel and the knee, and number four the one which completes the leg by connecting knee and the trunk of the robot. Relative rotational motion is allowed between them and with the trunk and wheels on both ends. The trunk of the robot is the fifth body and it is connected with a rotational joint to the sixth body, which resembles the head of the robot. The arm of the robot are made up of two bodies, numbers seven and eight, with relative rotational motion between them and of body seven with the trunk. A ninth additional body could be considered between the leg and the wheel, this body contains the electric motor which drives the wheel and allows rotational motion with the lower leg.

Once the bodies have been identified, the next thing to consider is what drives the motion of the joints between them, are there electric motors in each of the joints, or is this motion achieved through other mechanical means? The first thought might be to think it the robot contains electric motors for each of the joints, this is the most simple solution but by far the less efficient one. A big electric motor drives the wheel, but are all joints driven by the same configuration? *Boston Dynamics* is known for not going for the easiest solution, but looking for the most efficient mechanical solution. In this case, after analyzing carefully the robot's shapes and the scarce information available online, and taking into consideration configuration of previous robots from the same company, the conclusion was that the joints are driven by hydraulic actuators in a triangle configuration.

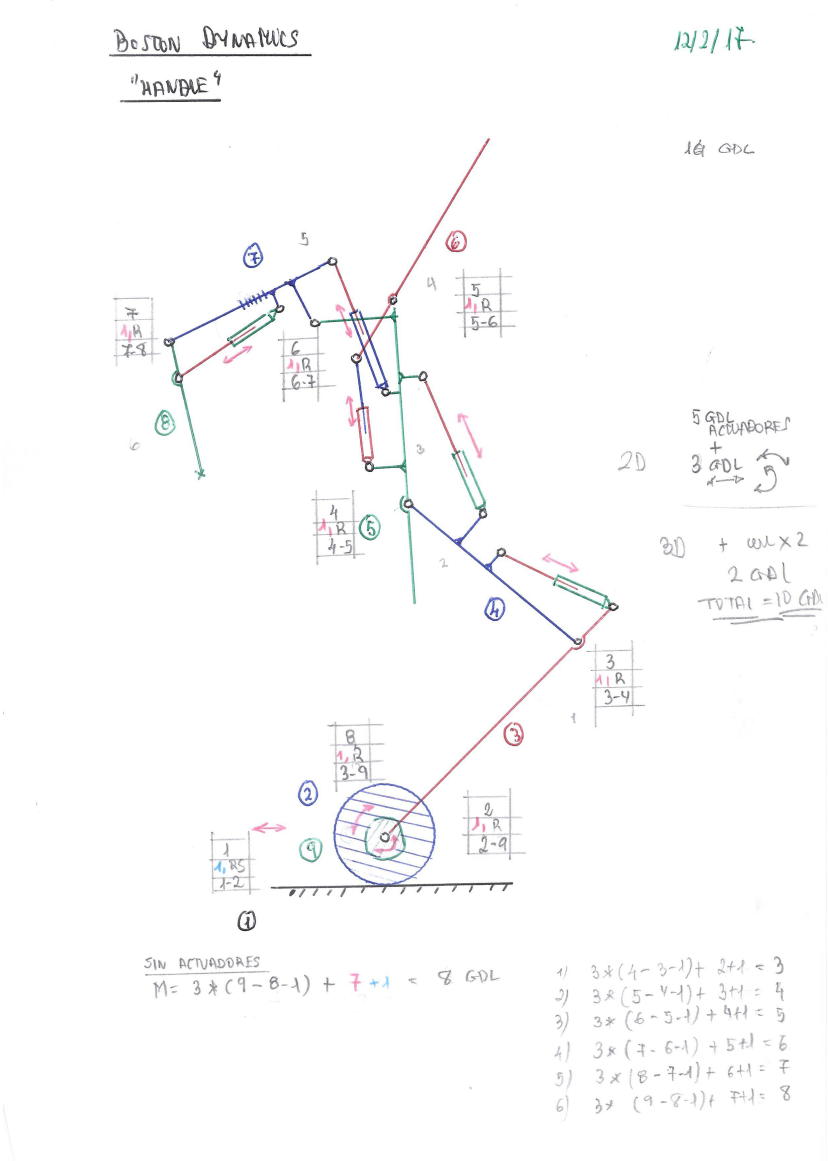


Figure 3.1: Initial kinematic diagram of the *Handle* robot where the different bodies, joints and actuators are defined.

Now, with the robot configuration in mind, a first kinematic diagram of the robot is drawn. This diagram is the starting point for the creation of the 2-dimensional virtual model. The lengths

and other body dimensions are yet to be defined, and the shapes of the bodies and actuator positions are subject to change.

The lengths of bodies and relative starting angle and position values of the robot will be extracted from one of the static photographs of the robot available at the Boston Dynamics website[1].

Figure 3.2: Static image of the *Handle* robot from which the geometrical information is extracted.

The geometrical parameters (lengths, body dimensions, anchor points, etc.) were proven not to be completely definitive. Throughout the process described in the following sections, some fine adjustments had to be done which resulted in modifying slightly some values in order to solve issues regarding incompatible configurations and for the sake of simplifying the resulting systems and equations of motion.

[1] **Boston Dynamics** (2017). *Handle*. URL: `https://www.bostondynamics.com/handle`.

3.1.2 From theory to practice

The complexity of the robot has been reduced to a simplified 2-dimensional model, now it is time to build it using a computer-aided engineering program. The software chosen is *SolidWorks*, an industry-leader CAD software designed to work on 3-dimensional models, therefore we must use solid bodies, instead of 2-dimensional sketches. We will use two of the available *SolidWorks* document types for the creation of the model: parts (**.sldprt*) and assemblies (**.sldasm*). Parts define solid bodies as a whole, there is no relative motion between the internal building blocks of parts. On the other side, assemblies are made up of bodies determined as part documents and allow the relative motion of the bodies by defining joints between them.

The basic building blocks for the model are extrusions of 2-dimensional shapes with a fixed depth, as a simple way of obtaining 3-dimensional bodies from 2-dimensional diagrams. There are three basic shapes (circles, squares and rectangles) with parameterizable dimensions that will make up the model, the shapes can be combined together to form more complex shapes and they will use cylindrical holes as joints in order to define the relative anchors between them. Each body derived from the original diagram will be composed using several of these shapes, and we will define a different color for each one of the bodies, in order for them to be easily distinguishable from one another.

A further simplification is proposed. The wheel element which allows relative rolling motion without slipping will be simplified to the closest equivalent based on blocks. The relative rotational motion is preserved, but the contact will be replaced for a translational joint with allows horizontal displacement along the surface of the ground.

The assembly process will be divided in two parts: the internal formation of bodies and the assembly of the bodies together. The bodies will be constructed by joining the extruded shapes in the necessary combinations to achieve the overall geometrical configuration desired by using fixed mechanical relationships, therefore disallowing relative motion between them, inside part documents. The combination of the part documents which make up the different bodies will be done inside an assembly document, allowing the relative motion of the bodies between them. This document will be at the top of the hierarchy pyramid and it will define our complete mechanical system. *SolidWorks* allows the user to modify part documents and then auto-updating the assembly document in which these bodies are used. This is a feature that will prove very useful in order to tweak some of the parameters of the model with the minimum effort.

Body	Name	Connected to
1	*Ground*	2
2	*Wheel*	1, 3
3	*Lower leg*	2, 4
4	*Upper leg*	3, 5
5	*Trunk*	4, 6, 7
6	*Head*	5
7	*Upper arm*	5, 8
8	*Lower arm*	7

Table 3.1: Bodies of the simplified *Handle* robot model with nomenclature and connections.

Joint	Type	Bodies
1	*Translational*	1, 2
2	*Revolute*	2, 3
3	*Revolute*	3, 4
4	*Revolute*	4, 5
5	*Revolute*	5, 6
6	*Revolute*	5, 7
7	*Revolute*	7, 8

Table 3.2: Joints of the simplified *Handle* robot model with types and connections.

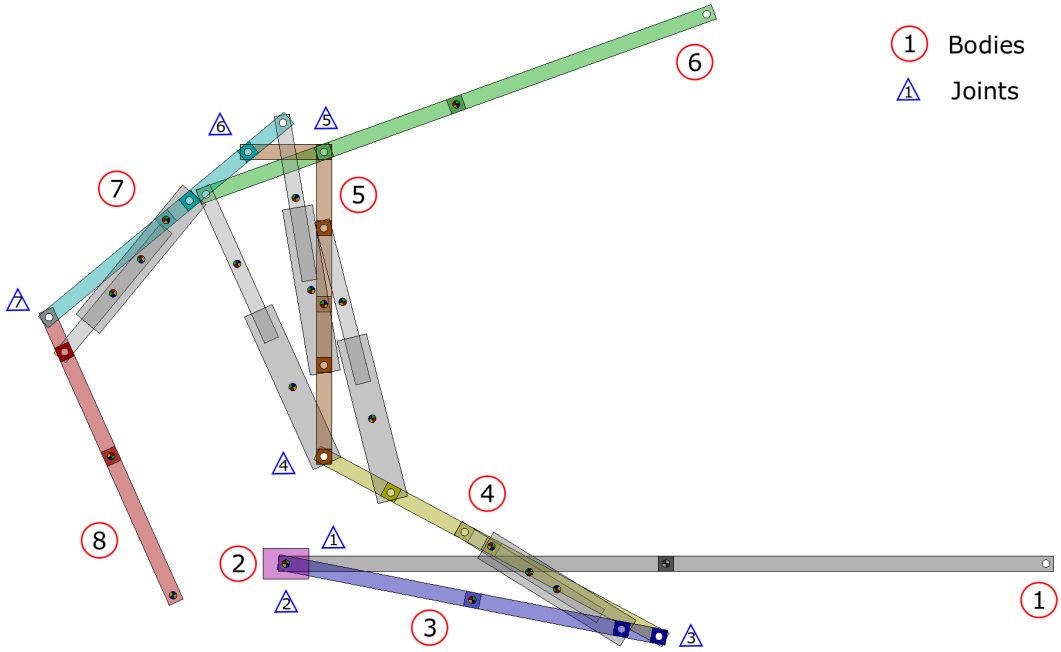


Figure 3.3: Bodies and joints locations on the *Handle* robot.

Once the model is assembled together by inserting bodies and defining the relative constraints or joints (known as *Mates* in the *SolidWorks* environment) between them, it is also necessary to include the drivers and their definition.

The resulting simplified model of the *Handle* robot is an extruded 2-dimensional assembly with 7 degrees-of-freedom (DOF): five linear actuators, a motor and a slider. The linear actuators ranging from La_1 to La_5 are the hydraulic actuators which move the upper body of the robot, these DOF will be precisely controlled by setting their elongation. The motor Θ_3 and slider X_2 (substituting the wheel) will be ultimately controlled using a different closed-loop controller for each one of them.

The end of the lower arm (body number 8) is the *tracer point*. This point is the equivalent of the hand in a humanoid robot, and therefore it is a rather interesting point that must be studied carefully. The equations of motion of this point (position, velocity and acceleration) will be obtained from the latter mathematical model.

Driver	DOF Name	Type	Bodies
1	X_2	*Slider*	1, 2
2	Θ_3	*Motor*	2, 3
3	La_1	*Linear actuator*	3, 4
4	La_2	*Linear actuator*	4, 5
5	La_3	*Linear actuator*	5, 6
6	La_4	*Linear actuator*	5, 7
7	La_5	*Linear actuator*	7, 8

Table 3.3: Drivers of the simplified *Handle* robot model with types and connections.

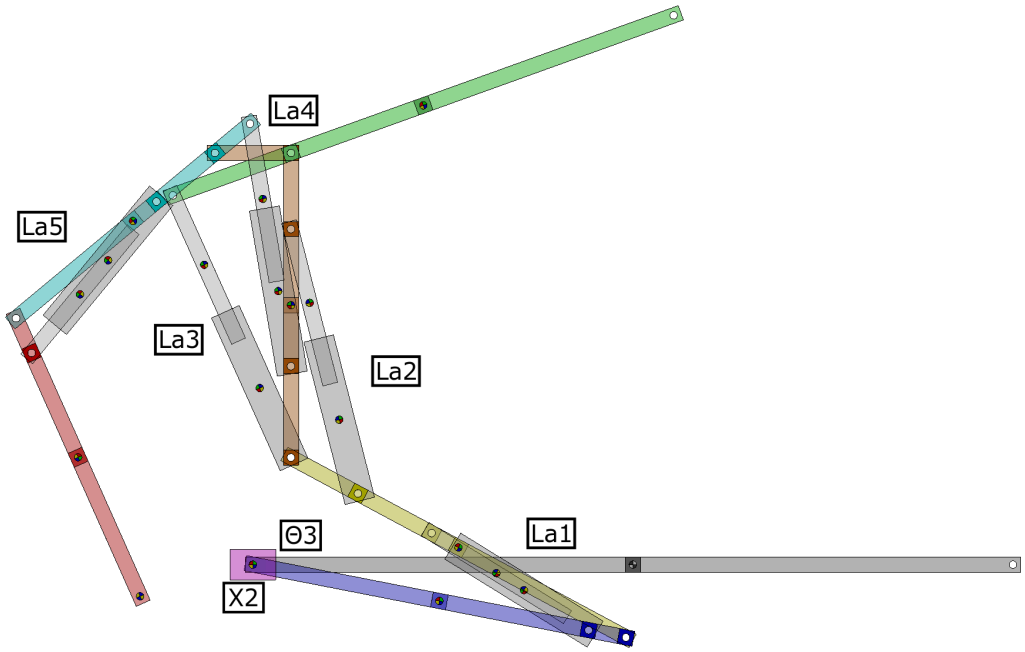


Figure 3.4: Drivers (linear actuators and motors) which control the DOF of the *Handle* robot.

3.1.3 Body definitions

The definitions of the different body parts with their parameterized dimensions are presented. All the bodies have a density defined by the ρ parameter (except body 2 with a density of ρ_2), a height of H (except bodies 2 and the cylinder barrel with a height of two times H), a width of W and the holes present in some bodies have a radius of r. The black and grey mark indicates the origin of the absolute coordinate system located at the middle of the ground body, the four-colored mark indicates the local origin of the body's coordinate system and the blue and yellow mark in body 8 determines the location of the tracer point.

There are eight main bodies and two additional bodies that make up each of the five linear actuators (hydraulic cylinders) in the model. All the relevant points in each body, such as local coordinate system origins and joint positions, have been enumerated accordingly.

According to the specifications of the robot available in the *Boston Dynamics* website, the maximum height of the robot is 2 meters and its weight is 105 kilograms. The dimensions of the different bodies and their densities will be chosen accordingly in order to obtain similar characteristics. All the bodies will have the same density except the wheel body that is considered to be heavier due to the electric motors it contains.

3.1.3.1 Body 1: Ground

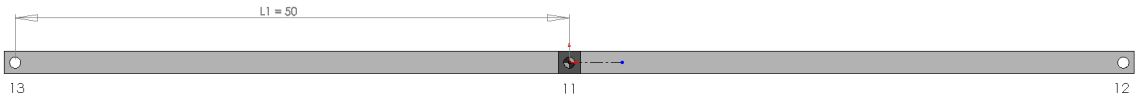


Figure 3.5: Definition of ground body (first body) with parametrized dimensions.

3.1.3.2 Body 2: Wheel

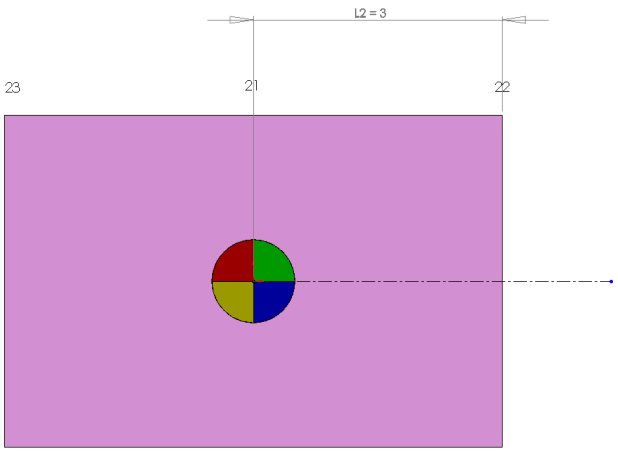


Figure 3.6: Definition of wheel body (second body) with parametrized dimensions.

3.1.3.3 Body 3: Lower leg

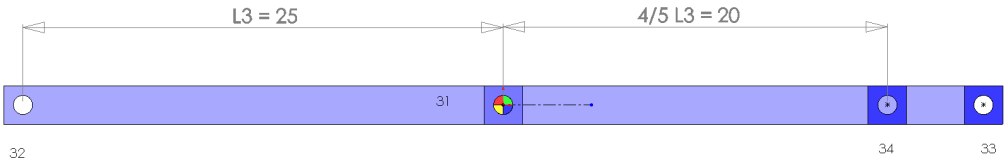


Figure 3.7: Definition of lower leg body (third body) with parametrized dimensions.

3.1.3.4 Body 4: Upper leg

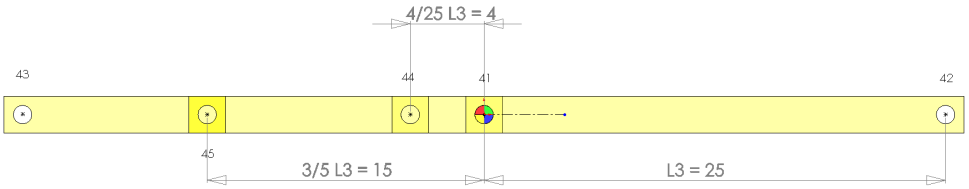


Figure 3.8: Definition of upper leg body (fourth body) with parametrized dimensions.

3.1.3.5 Body 5: Trunk

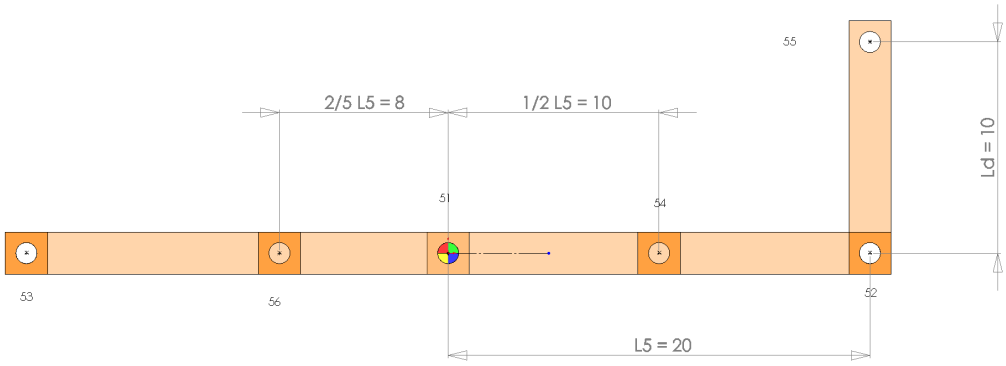


Figure 3.9: Definition of trunk body (fifth body) with parametrized dimensions.

3.1.3.6 Body 6: Head

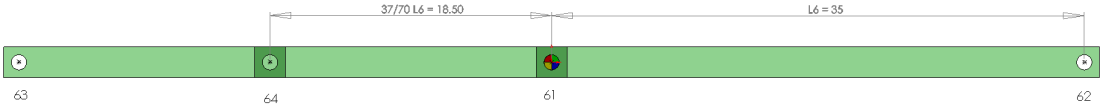


Figure 3.10: Definition of head body (sixth body) with parametrized dimensions.

3.1.3.7 Body 7: Upper arm

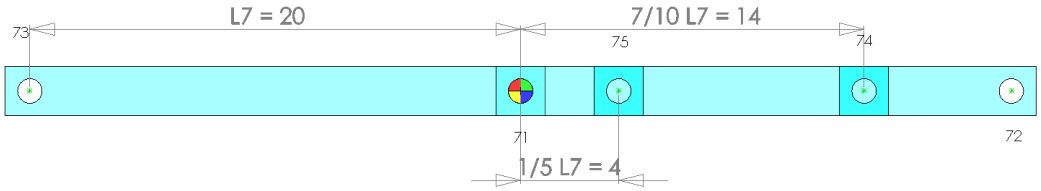


Figure 3.11: Definition of upper arm body (seventh body) with parametrized dimensions.

3.1.3.8 Body 8: Lower arm

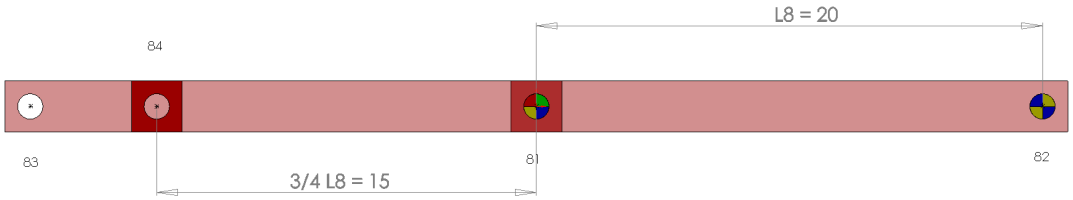


Figure 3.12: Definition of lower arm body (eighth body) with parametrized dimensions.

3.1.3.9 Body: Cylinder barrel

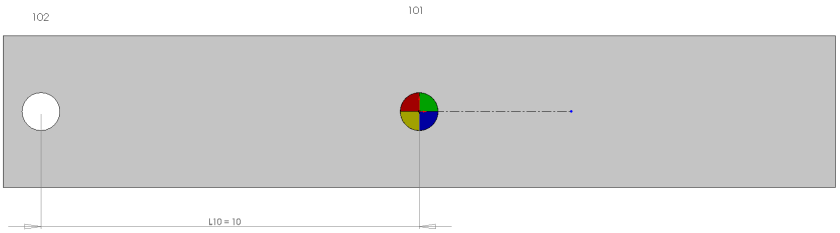


Figure 3.13: Definition of cylinder barrel with parametrized dimensions.

3.1.3.10 Body: Cylinder rod

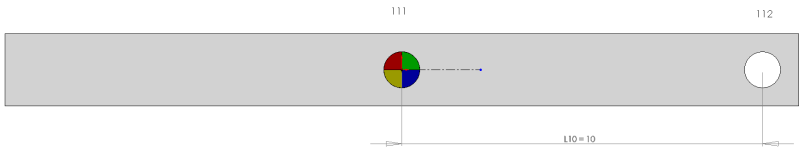


Figure 3.14: Definition of cylinder rod with parametrized dimensions.

3.1.3.11 Parameter values

Parameter	**Value**	**Units**	**Dimension**
L_1	50	cm	length $[L]$
L_2	3	cm	length $[L]$
L_3	25	cm	length $[L]$
L_5	20	cm	length $[L]$
L_6	35	cm	length $[L]$
L_7	20	cm	length $[L]$
L_8	20	cm	length $[L]$
L_{10}	10	cm	length $[L]$
L_d	10	cm	length $[L]$
H	2	cm	length $[L]$
W	1	cm	length $[L]$
ρ	$1 \cdot 10^5$	$\frac{kg}{m^3}$	density $[ML^{-3}]$
ρ_2	$2 \cdot 10^6$	$\frac{kg}{m^3}$	density $[ML^{-3}]$
r	$\frac{1}{2}$	cm	length $[L]$

Table 3.4: Parameters and their dimensions.

3.1.3.12 Inertial properties

The values of the geometrical parameters were chosen to obtain similar characteristics as the real robot. The model is a 2-dimensional approximation with a very small width, which causes the value of the density to be very large in order to match the mass specifications of the parts. This is not a problem as long as mass and inertias have coherent values.

The resulting values for mass, moment of inertia and total length of the bodies considering the parameter values will be recorded. The moment of inertia is calculated for the z axis (perpendicular to the body) because it is considered as a planar body. The total mass and the maximum height are close to the real values of the robot.

Body	**Property**	**Value**	**Units**
2	m_2	48	kg
	I_2	0.0208	$kg \cdot m^2$
	L_{b_2}	6	cm
3	m_3	10.2429	kg
	I_3	0.2248	$kg \cdot m^2$
	L_{b_3}	50	cm
4	m_4	10.2429	kg
	I_4	0.2248	$kg \cdot m^2$
	L_{b_4}	50	cm
5	m_5	10.1644	kg
	I_5	0.1867	$kg \cdot m^2$
	L_{b_5}	40	cm
6	m_6	14.2429	kg
	I_6	0.6033	$kg \cdot m^2$
	L_{b_6}	70	cm
7	m_7	8.2429	kg
	I_7	0.1174	$kg \cdot m^2$
	L_{b_7}	40	cm
8	m_8	8.2429	kg
	I_8	0.1174	$kg \cdot m^2$
	L_{b_8}	40	cm

Table 3.5: Inertial properties and total length of bodies.

	Real	**Model**	**Units**
Mass	105	109.3789	kg
Height	200	193.5	cm

Table 3.6: Comparison of mass and height specifications.

3.1.4 Model definition

For the correct understanding of the following developments, it is recommended to read chapters A and B from the appendices where the theory behind self-aligning mechanisms and variable-sided triangles is carefully detailed.

With all the bodies independently defined and the theory behind self-aligning mechanisms in mind, the next step is to assembly the complete model together. The eight main bodies are assembled together using the kinematic pairs defined in *Table 3.2* and the linear actuators will be temporarily defined using only revolute joints.

3.1.4.1 Initial position values

The model will be assembled in a folded starting position, representing the moment when the robot is at rest lying on the ground. This will be starting position for all the types of subsequent analyses.

This model contains a total of 8 main bodies plus 10 bodies due to linear actuators. The linear actuators are dummy bodies that aid in the motion of the model, therefore they are considered to have a negligible mass compared to the main bodies. These linear actuators are made up of a barrel with the corresponding rod and account for bodies 10 to 19.

All the coordinates are global coordinates, referred to the absolute coordinate system located at the midpoint of the ground body (point 11).

The initial positions and angles of the different parts of the model, as well as the values of the seven degrees of freedom, will be recorded to serve as starting values in the upcoming models.

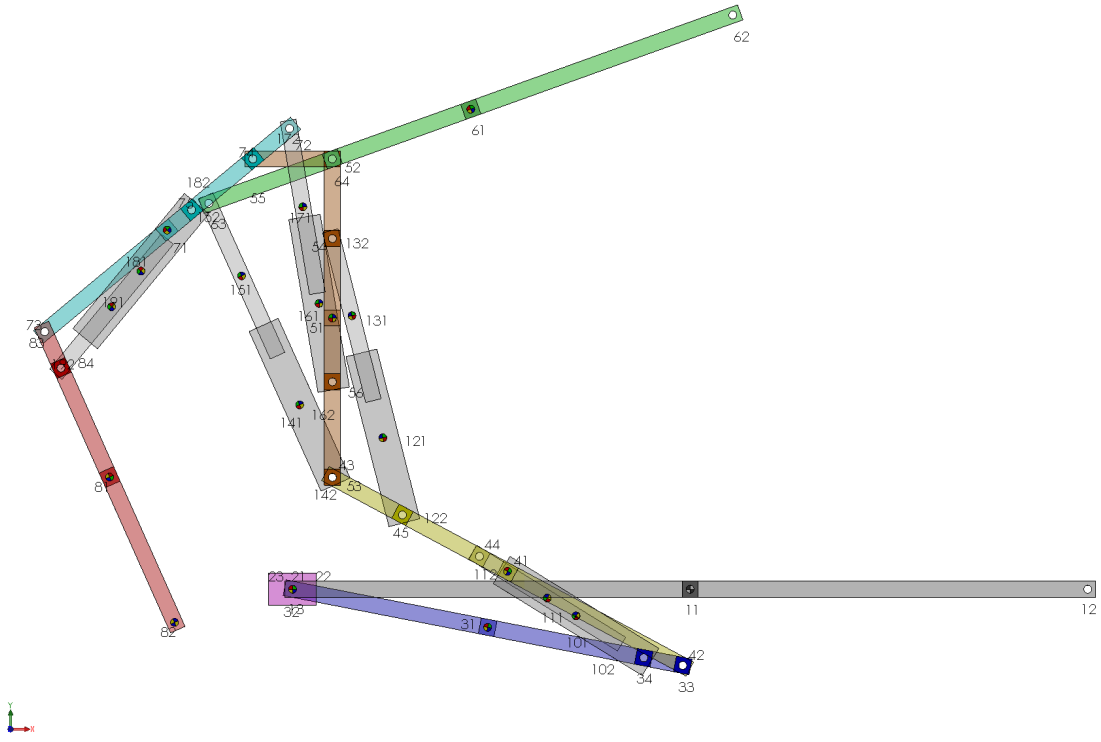


Figure 3.15: *SolidWorks* model of the robot with all the bodies and their relevant points.

Body	Coordinate	Value	Units
	X_2	-50	cm
2	Y_2	0	cm
	Θ_2	0	deg
	X_3	-25.46	cm
3	Y_3	-4.78	cm
	Θ_3	-11.03	deg
	X_4	-22.96	cm
4	Y_4	2.25	cm
	Θ_4	-28.2	deg
	X_5	-44.99	cm
5	Y_5	34.06	cm
	Θ_5	90	deg
	$X6$	-27.58	cm
6	Y_6	60.31	cm
	Θ_6	19.74	deg
	X_7	-65.76	cm
7	Y_7	45.12	cm
	Θ_7	39.68	deg
	X_8	-73	cm
8	Y_8	14.09	cm
	Θ_8	-65.92	deg
	X_{10}	-14.34	cm
10	Y_{10}	-3.36	cm
	Θ_{10}	148.31	deg
	X_{11}	-17.97	cm
11	Y_{11}	-1.11	cm
	Θ_{11}	148.31	deg
	X_{12}	-38.64	cm
12	Y_{12}	19.03	cm
	Θ_{12}	104.24	deg
	X_{13}	-42.53	cm
13	Y_{13}	34.37	cm
	Θ_{13}	104.24	deg
	X_{14}	-49.1	cm
14	Y_{14}	23.18	cm
	Θ_{14}	114.28	deg
	X_{15}	-56.41	cm
15	Y_{15}	39.37	cm
	Θ_{15}	114.28	deg
	X_{16}	-46.66	cm
16	Y_{16}	35.92	cm
	Θ_{16}	99.6	deg
	X_{17}	-48.7	cm
17	Y_{17}	48.03	cm
	Θ_{17}	99.6	deg
	X_{18}	-69.05	cm
18	Y_{18}	39.97	cm
	Θ_{18}	-129.56	deg
	X_{19}	-72.75	cm
19	Y_{19}	35.5	cm
	Θ_{19}	-129.56	deg

Table 3.7: Initial values for the coordinates of all the bodies in the model.

DOF	Value	Units
X_2	-50	cm
Θ_3	-11.03	deg
La_1	24.2722	cm
La_2	35.8221	cm
La_3	37.7705	cm
La_4	32.2810	cm
La_5	25.8034	cm

Table 3.8: Initial values for the seven degrees of freedom of the model.

3.1.4.2 Redundant constraint analysis

Using *COSMOSMotion* to analyze the mechanism returns a result of 15 redundant constraints.

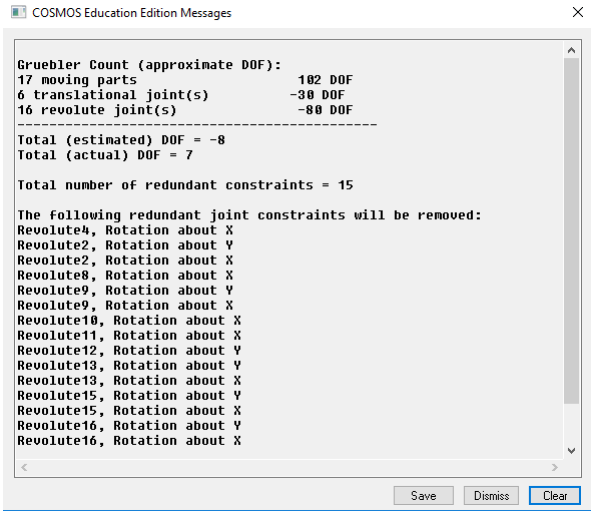


Figure 3.16: *COSMOSMotion* analysis results for the initial configuration.

The triangle configuration the linear actuator defines between itself and the two bodies it connects must be self-aligned in order to avoid these redundant constraints.

3.1.4.3 Self-alignment of the linear actuators

Lets consider the triangle configuration of the linear actuator independently. The mechanism has four bodies: the linear actuator which is made up of the rod and barrel bodies, and the two bodies the actuator connects. If the first body is considered as the fixed one, the model has three moving bodies. This mechanism has one degree of freedom, therefore its apparent mobility is one.

There are four kinematic pairs in this mechanical configuration: three revolute joints (class V, connectivity = 1) and a translational joint (class V, connectivity = 1). The first revolute joint connects the ground body to the second body, the second revolute joint connects the ground with the third body (cylinder barrel), and the third revolute joint connects the second body with the fourth body (cylinder rod). The translational joint allows the displacement of the cylinder rod into the barrel in their longitudinal axis.

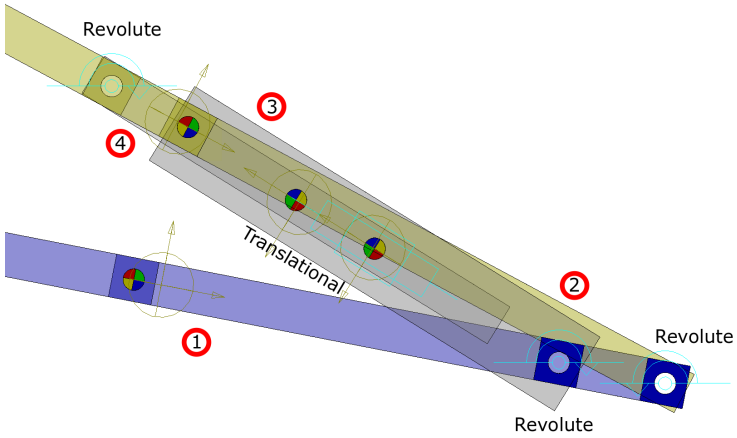


Figure 3.17: Representation of the triangle configuration of the linear actuator.

Applying the Grübler-Kutzbach formula and calculating the number of redundant constraints:

$$\begin{gathered} M_{3D} = 6\,(4-4-1) + (1+1+1+1) = -2 \\ \text{Number of redundant constraints} = 1-(-2) = 3 \end{gathered} \tag{3.1}$$

It is a straightforward deduction the fact that if there are 15 redundant constraints in the model and there are five linear actuators with three redundant constraints, successfully self-aligning the actuators will lead to the self-alignment of the whole model.

In this case the solution is the same as in the four-bar linkage. Three redundant constraints means that some revolute joints must be replaced with joints with a higher connectivity. The sum of the replacement joints' connectivity must be three units higher. From the two possible solutions, the solution adopted in this case is the following: replacing two revolute joints for an spherical (class III, connectivity = 3) and a cylindrical (class VI, connectivity = 2) joint.

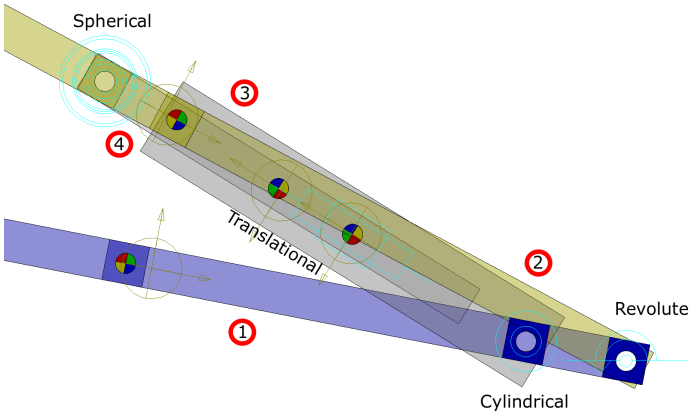


Figure 3.18: Representation of the self-aligned triangle configuration of the linear actuator.

Applying the mobility formula again returns the correct number of DOF:

$$M_{3D} = 6\,(4-4-1) + (1+1+3+2) = 1 \tag{3.2}$$

3.1.4.4 Resulting self-aligned model

Replacing the two revolute joints in each of the five linear actuators with the corresponding spherical and cylindrical joint removes the 15 redundant constraints, resulting in a self-aligned model. Using *COSMOSMotion* to analyze the mechanism again returns the correct number of degrees of freedom and zero redundant constraints.

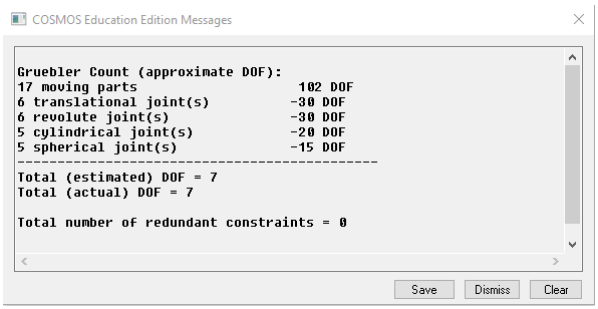


Figure 3.19: *COSMOSMotion* analysis results for the self-aligned model.

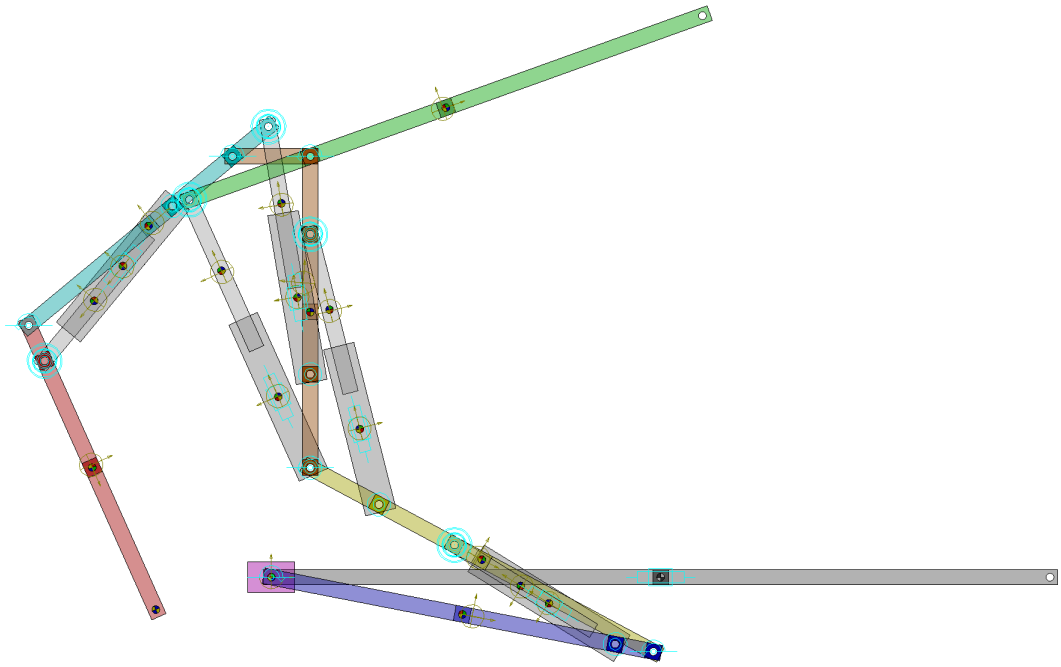


Figure 3.20: *COSMOSMotion* representation of the complete self-aligned model.

Now that the model has seven degrees of freedom and zero redundant constraints, it has been correctly self-aligned and it is ready to be simulated.

3.1.5 Kinematic motion definition

A kinematic simulation using *COSMOSMotion* will be conducted. All the degrees of freedom will be driven in order to make the robot stand up from its resting position. The path, velocity and acceleration of the tracer point will be obtained in order to be compared with the upcoming results of the mathematical model in *Mathematica*.

Using *STEP5* functions, the displacement of the seven degrees of freedom will be defined and the model simulated. The motions are selected to achieve a stand up movement of the robot in the time lapse of one second.

The reason for selecting the *STEP5* function as the interpolating step function for the kinematic motion is detailed in chapter C from the appendices.

3.1.5.1 Slider motion

The slider will describe a positive horizontal motion of 50 centimeters.

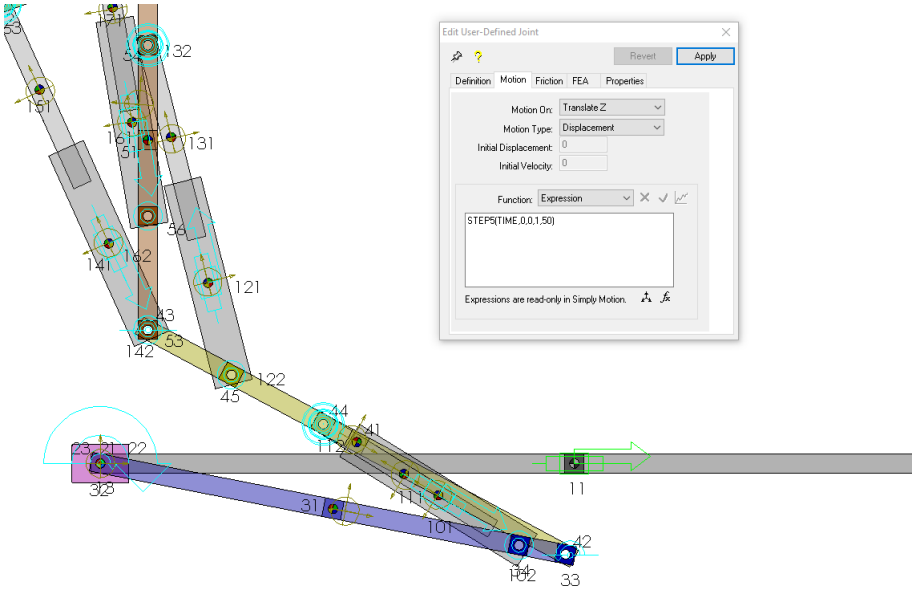


Figure 3.21: STEP5 definition of the slider motion in *COSMOSMotion*.

3.1.5.2 Angular motor motion

The angular motor will describe a positive angular motion of 45 degrees.

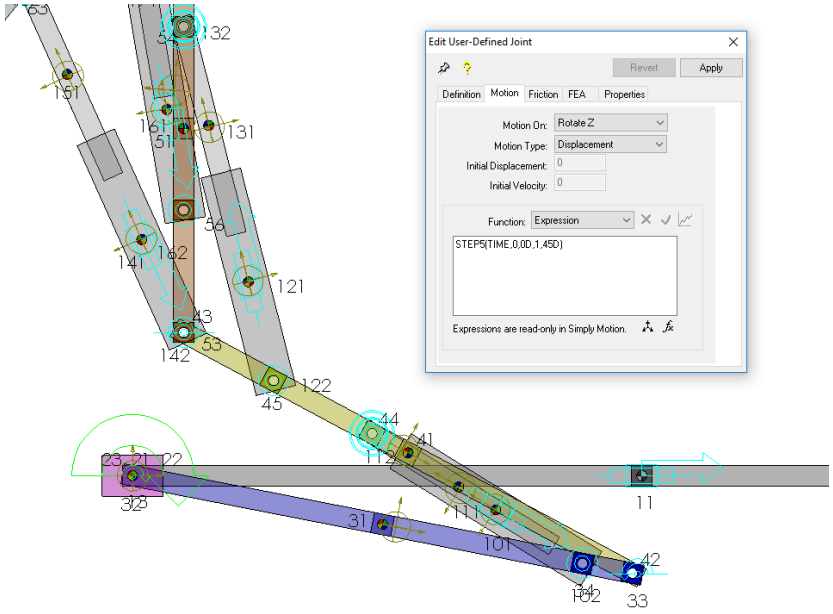


Figure 3.22: STEP5 definition of the angular motor motion in *COSMOSMotion*.

3.1.5.3 First linear actuator motion

The first linear actuator will describe an extension motion of 4 centimeters.

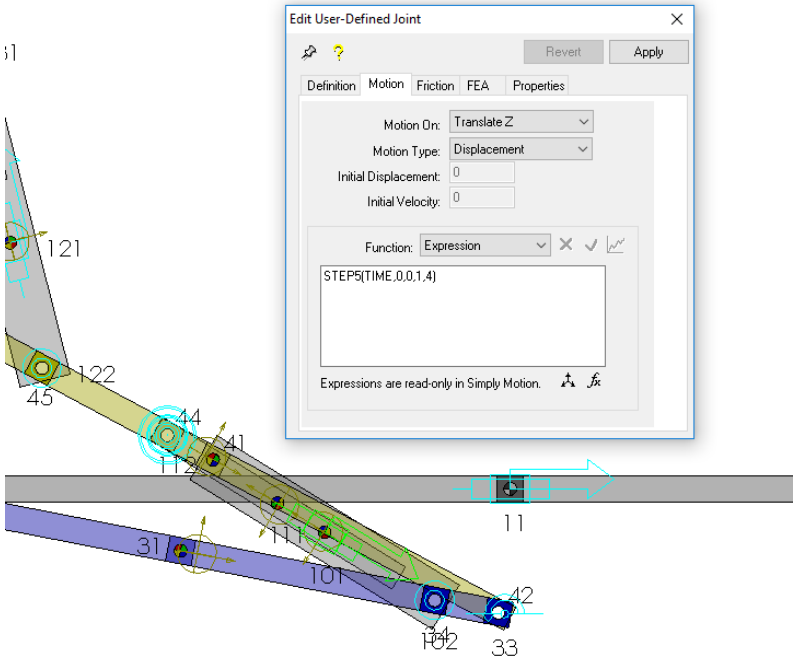


Figure 3.23: STEP5 definition of first linear actuator motion in *COSMOSMotion*.

3.1.5.4 Second linear actuator motion

The second linear actuator will describe an extension motion of 1 centimeter.

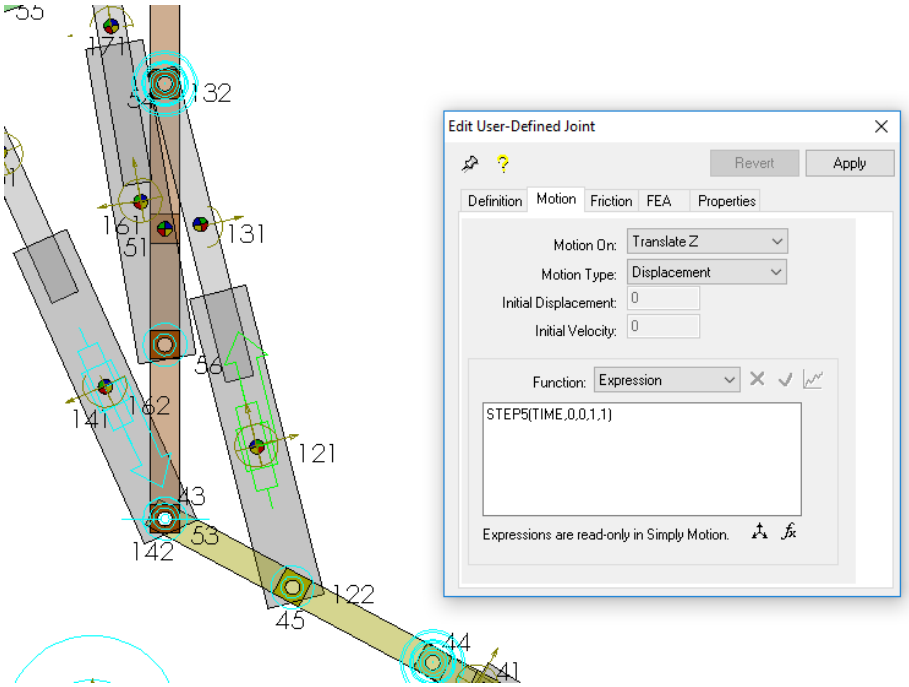


Figure 3.24: STEP5 definition of second linear actuator motion in *COSMOSMotion*.

3.1.5.5 Third linear actuator motion

The third linear actuator will describe a contraction motion of 12 centimeters.

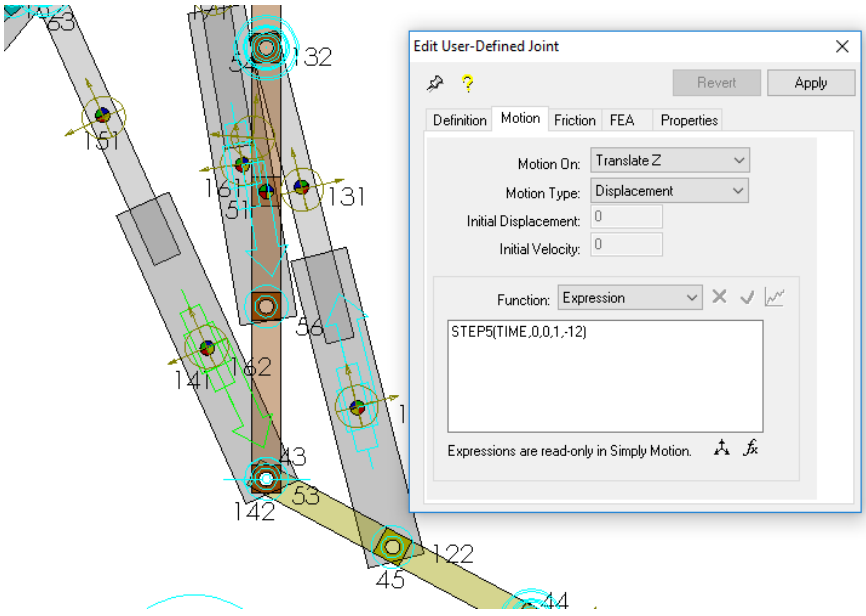


Figure 3.25: STEP5 definition of third linear actuator motion in *COSMOSMotion*.

3.1.5.6 Fourth linear actuator motion

The fourth linear actuator will describe a contraction motion of 3 centimeters.

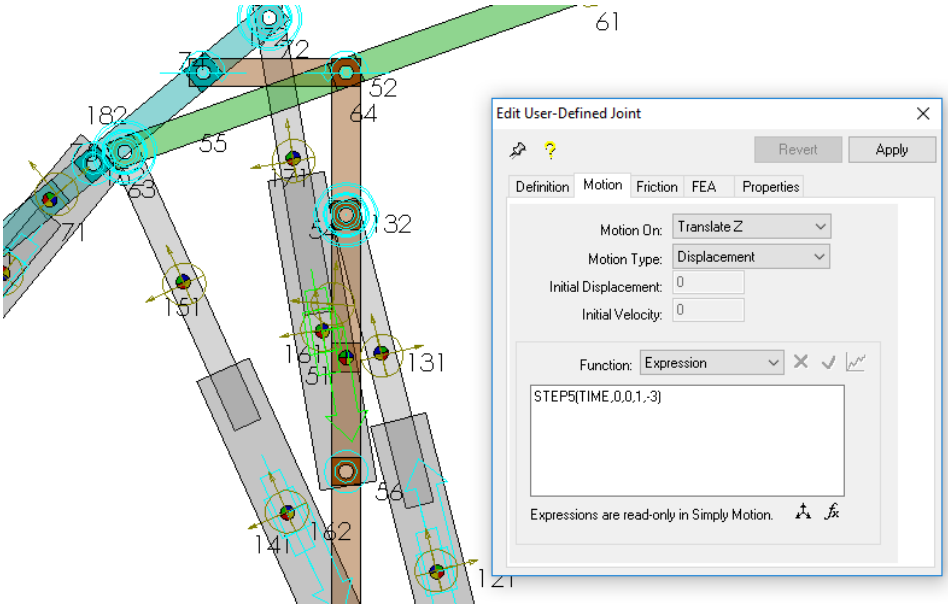


Figure 3.26: STEP5 definition of fourth linear actuator motion in *COSMOSMotion*.

3.1.5.7 Fifth linear actuator motion

The fifth linear actuator will describe an extension motion of 1 centimeter.

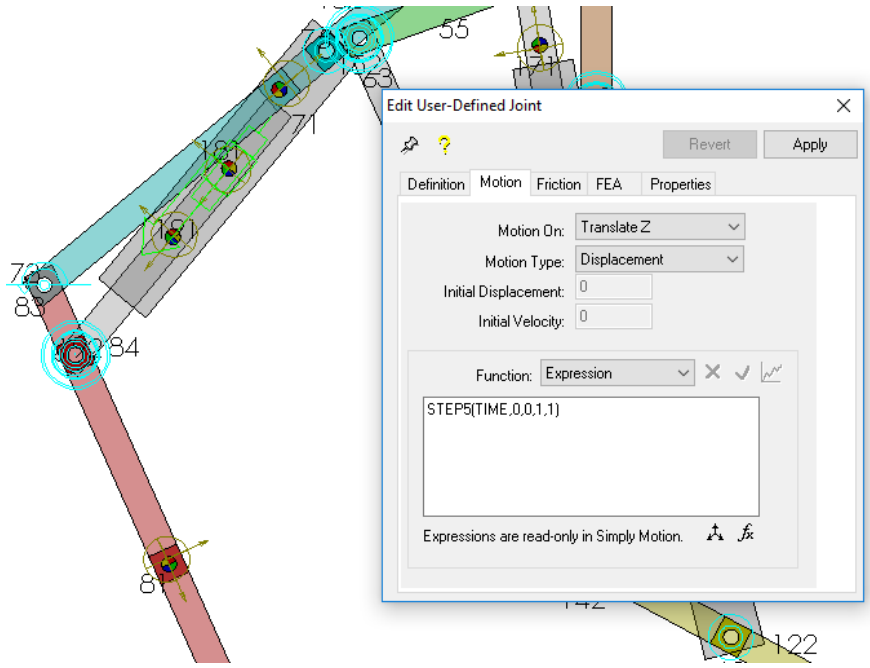


Figure 3.27: STEP5 definition of fifth linear actuator motion in *COSMOSMotion*.

3.2 Symbolic mathematical model in *Mathematica*

Developing a mathematical model of the robot in *Mathematica* provides the opportunity to obtain expressions for the behavior (position, velocity and acceleration) of all the different bodies in terms of the values of its seven degrees of freedom.

Mathematica is a very powerful symbolic environment where the equations of motion can be expressed in terms of the parameters that describe the bodies. By symbolically solving the equations of the model one can obtain the expressions for the behavior of the bodies in a generalized way, and then substitute variables for their numerical values to obtain the behavior for a specific configuration or time period.

The mathematical model is divided in two big parts: kinematics and dynamics. Solving the kinematic model provides expressions for the position, velocity and acceleration of the bodies in terms of the values and first and second derivatives of the drivers (DOF). This is the first model one must obtain when analyzing a mechanical system. Once the kinematic model is solved, the next step is to tackle the dynamic model. The dynamic model arises when leaving some or all of the drivers loose. For each driver left loose, one differential equation that describes the behavior of the degree of freedom appears. The solutions obtained from the kinematic model can be useful when describing and simplifying the dynamic model.

The solution for the kinematic model was obtained in a completely parametrized symbolic manner, in which all the geometrical parameters of the bodies were left unsubstituted, and also in a numerical version, where the value of these parameters are substituted from the beginning and result in much simplified and less computationally expensive expressions. Both of the solutions successfully express the positions of the bodies in terms of the system's degrees of freedom. The numerical expressions will be used to obtain a interactive model in *Mathematica* using the **Manipulate** command, where the values of the drivers can be changed dynamically and the visual representation of the model is updated in real time.

It should be noted that development of the mathematical model was an iterative process. The models had to be successively simplified in order to obtain expressions with a complexity within reasonable levels. Different approaches for solving the problems presented in the model were tried until the optimum and simplest solution was obtained. This was the critical phase for the development of the model, once a simplification was proposed and proven to be valid the previous 2-dimensional model in *SolidWorks* has to be modified accordingly. The final model depicted here is the result of many iterations and it is simplified to the maximum extent possible without losing the essence of the robot.

3.2.1 Mathematical definition of bodies

Once the simplified model of the *Handle* robot is created in the virtual environment of *SolidWorks*, it must be mathematically described in *Mathematica*. The mathematical definition of the mechanical bodies is essentially reduced a series of points and to the inertial properties each one of them has. The points that must be defined are those which determine the position of joints which provide the connection to the surrounding bodies, and the inertial properties are the center of mass (*centroid*), the mass, and the moment of inertia.

In order to obtain the inertial properties of each one of the bodies, the **CompositeInertia** function of the *MechanicalSystems* package is used. This function returns the list of the inertial properties of a composite body made up of a number of specified subcomponents. All of the subcomponents of each body will be manually defined taking into consideration the geometrical features it presents (holes and protruding parts). The holes represent the locations of the mechanical joints between the different bodies.

The ground body does not contribute to the overall center of mass of the model because it only provides the fixed base over which the robot moves, therefore its inertial properties are not relevant. The actuators will be considered to have negligible mass and inertia when compared to the other body parts, therefore not being included in the calculations. The wheel (body number 2) is not included in the center of mass calculated because it only translates horizontally through the ground, this will be explained in the upcoming sections.

The centroid and inertia of the bodies are calculated relative to the origin of their local coordinate system, and all bodies have a height of H (except body number 2), a width of W and a density of ρ (except body number 2). The wheel and the lower leg bodies have a different color than the rest because they contain the drivers X_2 and Θ_3, they are colored with a light green as well as the linear actuators. The orientation and positions of the bodies depicted correspond to the initial position of the robot, in its folded state. The body's coordinate system configuration and the parameters used are the same as in the *SolidWorks* model.

3.2.1.1 Body 2: Wheel

This body contains no holes or protruding parts, its inertial properties are pretty straightforward.

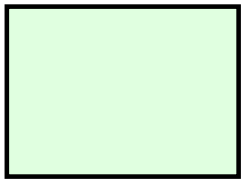


Figure 3.28: Definition of wheel body (second body) in *Mathematica*.

Centroid	$\{0, 0\}$
Mass	$4HL_2\,\rho_2\,W$
Inertia	$\frac{4}{3}HL_2(H^2 + L_2^2)\,\rho_2\,W$

Table 3.9: Inertial properties of wheel body (second body).

3.2.1.2 Body 3: Lower leg

This body contains two holes, the impact of these elements will be taken into account when obtaining its inertial properties.

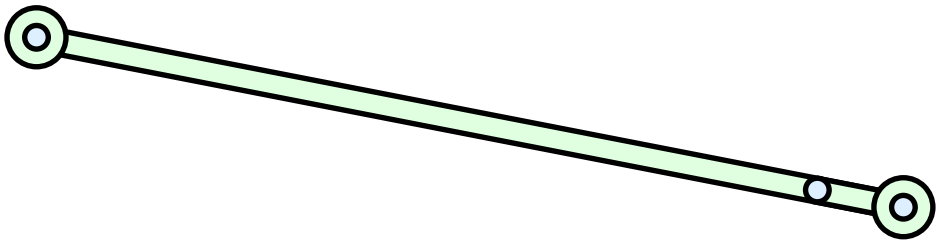


Figure 3.29: Definition of lower leg body (third body) in *Mathematica*.

Centroid	$\{0, 0\}$
Mass	$2\left(H(L_3 + 2r) - \pi r^2\right)\,\rho\,W$
Inertia	$\frac{1}{6}\left(H^3(L_3 + 2r) + 4H(L_3 + 2r)^3 - 6\pi r^2\left(2L_3^2 + r^2\right)\right)\,\rho\,W$

Table 3.10: Inertial properties of lower leg body (third body).

3.2.1.3 Body 4: Upper leg

This body contains two holes, the impact of these elements will be taken into account when obtaining its inertial properties.

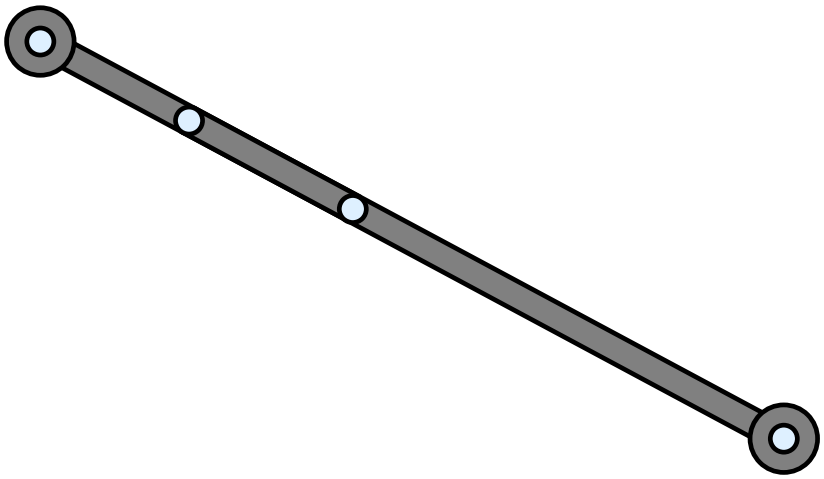


Figure 3.30: Definition of upper leg body (fourth body) in *Mathematica*.

Centroid	$\{0, 0\}$
Mass	$2\left(H(L_3 + 2r) - \pi r^2\right)\,\rho\,W$
Inertia	$\frac{1}{6}\left(H^3(L_3 + 2r) + 4H(L_3 + 2r)^3 - 6\pi r^2\left(2L_3^2 + r^2\right)\right)\,\rho\,W$

Table 3.11: Inertial properties of upper leg body (fourth body).

3.2.1.4 Body 5: Trunk

This body contains three holes and one protruding part, the impact of these elements will be taken into account when obtaining its inertial properties.

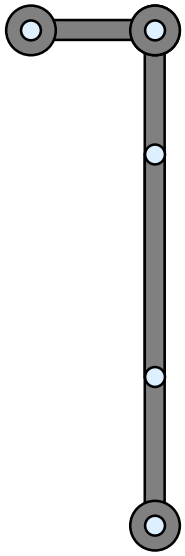


Figure 3.31: Definition of trunk body (fifth body) in *Mathematica*.

Centroid	$\left\{\frac{L_5\left(HL_d-\pi r^2\right)}{H(2L_5+L_d+4r)-3\pi r^2}, \frac{L_d\left(H(H+L_d)-2\pi r^2\right)}{2H(2L_5+L_d+4r)-6\pi r^2}\right\}$
Mass	$\left(H(2L_5+L_d+4r)-3\pi r^2\right)\,\rho\,W$
Inertia	$\frac{1}{12\left(H(2L_5+L_d+4r)-3\pi r^2\right)}\Big(H^4\left(4L_5^2+2L_5(5L_d+8r)+L_d^2+20L_dr+16r^2\right)+$ $6H^3\left(2L_5L_d^2-\pi r^2(L_5+2L_d)+4L_d^2r-2\pi r^3\right)+$ $H^2\left(16L_5^4+32L_5^3L_d+16r\left(8L_5^3+6L_5^2L_d+L_d^3\right)+8L_5L_d^3+\right.$ $\left.6r^2\left(16L_5(4L_5+L_d)-\pi L_d^2\right)+64r^3(8L_5+L_d)+L_d^4+256r^4\right)-$ $6\pi Hr^2\left(8r\left(6L_5^2+L_d^2\right)+2(2L_5+L_d)\left(4L_5^2+L_d^2\right)+3r^2(18L_5+L_d)+44r^3\right)+$ $6\pi^2r^4\left(16L_5^2+4L_d^2+9r^2\right)\Big)\,\rho\,W$

Table 3.12: Inertial properties of trunk body (fifth body).

3.2.1.5 Body 6: Head

This body contains two holes, the impact of these elements will be taken into account when obtaining its inertial properties.

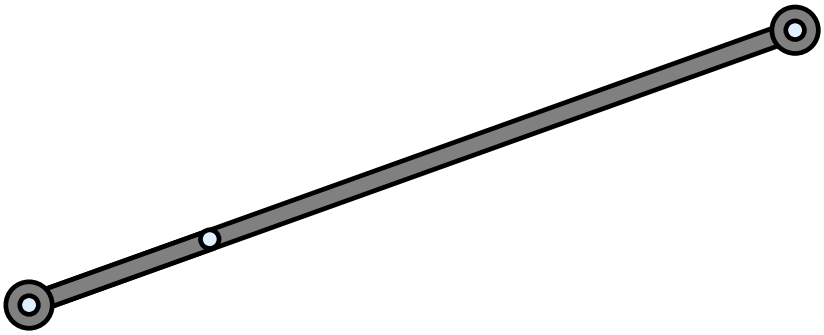


Figure 3.32: Definition of head body (sixth body) in *Mathematica*.

Centroid	$\{0, 0\}$
Mass	$2\left(H(L_6 + 2r) - \pi r^2\right)\,\rho\,W$
Inertia	$\frac{1}{6}\left(H^3(L_6 + 2r) + 4H(L_6 + 2r)^3 - 6\pi r^2\left(2L_6^2 + r^2\right)\right)\,\rho\,W$

Table 3.13: Inertial properties of head body (sixth body).

3.2.1.6 Body 7: Upper arm

This body contains two holes, the impact of these elements will be taken into account when obtaining its inertial properties.

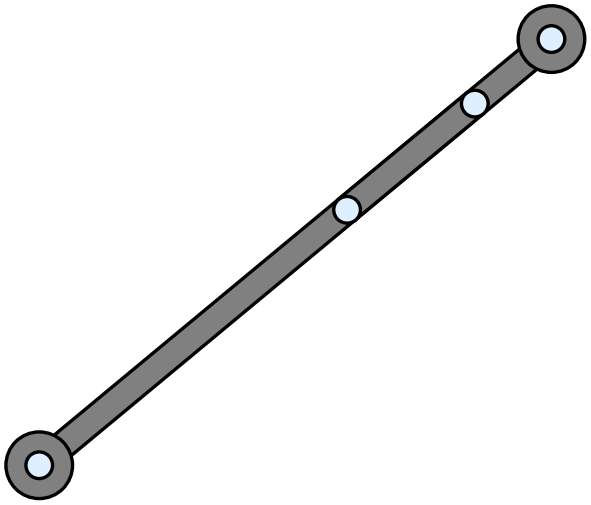


Figure 3.33: Definition of upper arm body (seventh body) in *Mathematica*.

Centroid	$\{0, 0\}$
Mass	$2\left(H(L_7 + 2r) - \pi r^2\right)\,\rho\,W$
Inertia	$\frac{1}{6}\left(H^3(L_7 + 2r) + 4H(L_7 + 2r)^3 - 6\pi r^2\left(2L_7^2 + r^2\right)\right)\,\rho\,W$

Table 3.14: Inertial properties of upper arm body (seventh body).

3.2.1.7 Body 8: Lower arm

This body contains two holes, the impact of these elements will be taken into account when obtaining its inertial properties.

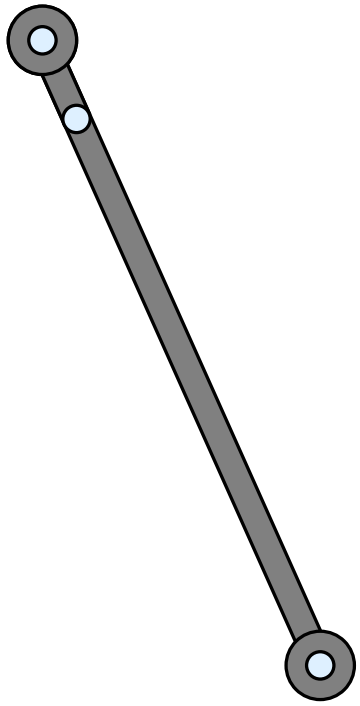


Figure 3.34: Definition of lower arm body (eighth body) in *Mathematica*.

Centroid	$\{0, 0\}$
Mass	$2\left(H(L_8 + 2r) - \pi r^2\right)\,\rho\,W$
Inertia	$\frac{1}{6}\left(H^3(L_8 + 2r) + 4H(L_8 + 2r)^3 - 6\pi r^2\left(2L_8^2 + r^2\right)\right)\,\rho\,W$

Table 3.15: Inertial properties of lower arm body (eighth body).

3.2.1.8 Center of mass calculation

It is possible to obtain the inertias and center of mass of the whole assembly by taking into consideration the masses and inertias of the different bodies it contains. The same function as before, **CompositeInertia**, will be used to calculate the global inertial parameters of the model: center of mass (centroid), total mass, and total inertia.

The center of mass of the total assembly will be calculated considering the relative position of each of the bodies and their relative orientations between each other. The origin of the global coordinate system is located in the midpoint of the ground body (first body), and all the results will be relative to this point. The total inertia will not be presented because of it is a rather monstrous equation that does not fit on the page.

Neither the ground body nor the five linear actuators, that are considered to have negligible mass, will be considered for the calculation of the whole center of mass. The wheel, body number 2, will also be excluded from the center of mass calculation, only considering the bodies that turn around the pivot point defined by the revolute joint driven by Θ_3.

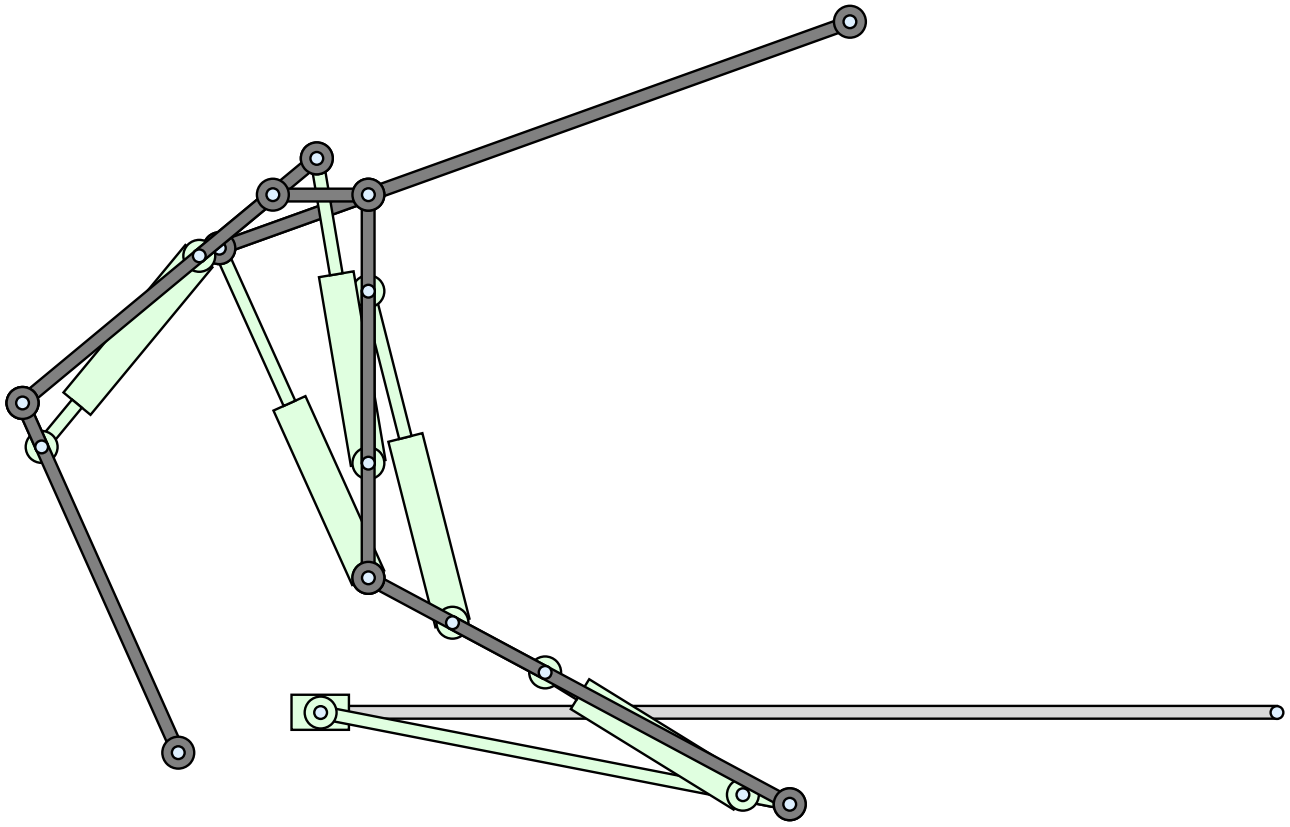


Figure 3.35: Definition of the complete model in *Mathematica* at its initial position.

Mass	$(H(4L_3 + 2L_5 + 2L_6 + 2L_7 + 2L_8 + L_d + 24r) - 13\pi r^2)\ \rho\, W$
Centroid	$\Bigg\{\frac{1}{-4H(2L_3 + L_5 + L_6 + L_7 + L_8) - 2HL_d - 48Hr + 26\pi r^2}$ $\Big(-2H(2L_3(X_3 + X_4) + 2L_5X_5 + 2L_6X_6 + 2L_7X_7 + 2L_8X_8 + L_dX_5 +$ $4r(X_3 + X_4 + X_5 + X_6 + X_7 + X_8)) + 2L_5 \cos[\Theta_5]\left(\pi r^2 - HL_d\right) +$ $L_d \sin[\Theta_5]\left(H(H + L_d) - 2\pi r^2\right) + 2\pi r^2(2X_3 + 2X_4 + 3X_5 + 2(X_6 + X_7 + X_8))\Big),$ $\frac{1}{2H(4L_3 + 2(L_5 + L_6 + L_7 + L_8) + L_d) + 48Hr - 26\pi r^2}$ $\Big(2H(2L_3(Y_3 + Y_4) + 2L_5Y_5 + 2L_6Y_6 + 2L_7Y_7 + 2L_8Y_8 + L_dY_5 +$ $4r(Y_3 + Y_4 + Y_5 + Y_6 + Y_7 + Y_8)) + 2L_5 \sin[\Theta_5]\left(HL_d - \pi r^2\right) +$ $L_d \cos[\Theta_5]\left(H(H + L_d) - 2\pi r^2\right) - 2\pi r^2(2Y_3 + 2Y_4 + 3Y_5 + 2(Y_6 + Y_7 + Y_8))\Big)\Bigg\}$

Table 3.16: Total inertial properties of the complete model, considering bodies 3 through 8.

The solution to the generalized coordinates of the bodies and the trigonometric functions of their angles of orientation will be substituted when the kinematic model is solved in order to obtain an expression that is only dependent on the seven degrees of freedom of the model.

3.2.2 Kinematic model definition

The kinematic model definition in *Mathematica* will be done using the *MechanicalSystems* package with the *Modeler2D* option for planar mechanical systems. There are several steps for the definition of the model and subsequent obtaining of the kinematic equations of motion.

The first thing is to define the bodies using the **Body** function. The input parameters are the following: the points that define the body, which are the same as in the *SolidWorks* definition; the values of the generalized coordinates in the starting point, both Cartesian coordinates and the angle of orientation; and the inertial properties of the body, including mass, inertia, and center of mass.

The constraint equations corresponding to each of the joints will be defined using the 2-dimensional constraint definition functions. In this case, the translational joints will be defined using **Translate2** and revolute joints will be defined using **Revolute2**. The drivers that make up the degrees of freedom will be defined separately. **RelativeX1** will be used for the position of the slider X_2, and **RotationLock1** will define the angular motor Θ_3.

The definition of the linear actuators can be defined in two different ways: as a relative distance with **RelativeDistance1** or as a relative angle between the connecting bodies using **RelativeAngle1**. In the first mathematical models the relative distance constraint was used, but it was finally replaced in favor of the relative angle constraint due to the greater simplicity of the latter. This substitution caused a noticeable decrease in the complexity of the resulting expressions. More insight about this topic is available in chapter B from the appendices.

The resulting constraint equation vector is:

$$\mathbf{\Phi}(\mathbf{q},t) = \begin{pmatrix} 4L_1L_2\sin\Theta_2 \\ 2L_2(L_1+X_2-L_2\cos\Theta_2)\sin\Theta_2 + 2L_2\cos\Theta_2(-Y_2+L_2\sin\Theta_2) \\ X_2 - X_3 + L_3\cos\Theta_3 \\ Y_2 - Y_3 + L_3\sin\Theta_3 \\ X_3 - X_4 + L_3\cos\Theta_3 - L_3\cos\Theta_4 \\ Y_3 - Y_4 + L_3\sin\Theta_3 - L_3\sin\Theta_4 \\ X_4 - X_5 - L_3\cos\Theta_4 + L_5\cos\Theta_5 \\ Y_4 - Y_5 - L_3\sin\Theta_4 + L_5\sin\Theta_5 \\ X_5 - X_6 + L_5\cos\Theta_5 + \frac{37}{70}L_6\cos\Theta_6 \\ Y_5 - Y_6 + L_5\sin\Theta_5 + \frac{37}{70}L_6\sin\Theta_6 \\ X_5 - X_7 + L_5\cos\Theta_5 - \frac{7}{10}L_7\cos\Theta_7 - L_d\sin\Theta_5 \\ Y_5 - Y_7 + L_d\cos\Theta_5 + L_5\sin\Theta_5 - \frac{7}{10}L_7\sin\Theta_7 \\ X_7 - X_8 - L_7\cos\Theta_7 + L_8\cos\Theta_8 \\ Y_7 - Y_8 - L_7\sin\Theta_7 + L_8\sin\Theta_8 \\ X_2 - pos(t) \\ \Theta_3 - ang(t) \\ \sin(\Theta_3 - \Theta_4 - \Theta_{La_1}) \\ \sin(\Theta_4 - \Theta_5 + \Theta_{La_2}) \\ \sin(\Theta_5 - \Theta_6 - \Theta_{La_3}) \\ 5L_d\cos(\Theta_5 - \Theta_7 + \Theta_{La_4}) + 7L_5\sin(\Theta_5 - \Theta_7 + \Theta_{La_5}) \\ \sin(\Theta_7 - \Theta_8 - \Theta_{La_5}) \end{pmatrix} = \mathbf{0} \qquad (3.3)$$

Where Θ_{La_1}, Θ_{La_2}, Θ_{La_3}, Θ_{La_4} and Θ_{La_5} are the relative angles between bodies expressed in terms of the corresponding actuator lengths.

$$\begin{aligned}
\Theta_{La_1} &= \cos^{-1}\left(\frac{\left(\frac{L_3}{5}\right)^2 + \left(\frac{29L_3}{25}\right)^2 - La_1^2}{2(\frac{L_3}{5})(\frac{29L_3}{25})}\right) \\
\Theta_{La_2} &= \cos^{-1}\left(\frac{\left(\frac{2L_3}{5}\right)^2 + \left(\frac{3L_5}{2}\right)^2 - La_2^2}{2(\frac{2L_3}{5})(\frac{3L_5}{2})}\right) \\
\Theta_{La_3} &= \cos^{-1}\left(\frac{(2L_5)^2 + \left(\frac{33L_6}{70}\right)^2 - La_3^2}{2(2L_5)(\frac{33L_6}{70})}\right) \\
\Theta_{La_4} &= \cos^{-1}\left(\frac{\left(\sqrt{\left(\frac{7L_5}{5}\right)^2 + L_d^2}\right)^2 + \left(\frac{3L_7}{10}\right)^2 - La_4^2}{2\left(\sqrt{\left(\frac{7L_5}{5}\right)^2 + L_d^2}\right)(\frac{3L_7}{10})}\right) \\
\Theta_{La_5} &= \cos^{-1}\left(\frac{\left(\frac{6L_7}{5}\right)^2 + \left(\frac{L_8}{4}\right)^2 - La_5^2}{2(\frac{6L_7}{5})(\frac{L_8}{4})}\right)
\end{aligned} \tag{3.4}$$

The constraint equations can be symbolically solved with *Mathematica* to obtain the x and y positions and orientations of all the bodies in terms of the seven degrees of freedom: X_2, Θ_3, La_1, La_2, La_3, La_4 and La_5. The trigonometric functions of the body's angle are also important expressions that are obtained from these equations. These expressions make up the *kinematic solution of the model*. Because of the symbolic nature of the expressions from the kinematic solution, they can be derived to obtain the expression for linear and angular velocity and acceleration of the body in terms of its degrees of freedom.

The kinematic solution allows the center of mass and the moment of inertia of the model to be expressed in terms of its inputs, which are its degrees of freedom. This results can later be used to develop dynamic model or complex control loops.

The visual representation of the model is another important output that is obtained from *Mathematica*. Once the equations that rule the kinematic model are solved and the expressions for the bodies obtained, it is possible to define a visual representation of the model. The visual representation is developed using two built-in graphical functions from *Mathematica*: **Disk** to draw circular shapes, and **Polygon** to define rectangular shapes.

All the bodies are represented in several sections depending on their shapes and the number of holes they have. Linear actuators that make up the variable-sided triangles are not considered bodies, instead they are a result from the relative angle constraints and their definition is done in a simplified way. They are defined between the points they connect using the length of the barrel L_{10} as the fixed one and the length of the rod is obtained from the difference between the corresponding actuator length La and L_{10}.

3.2.2.1 Solution structure

The solution of the kinematic model provides insight into the structure of the solution to the position and orientation of the bodies in terms of the seven degrees of freedom of the model. The coordinates of the bodies and the trigonometric functions of the angle have common traits, both of them have a *canonical form* with shared coefficients. Uppercase letters denote absolute coordinates while lowercase refer to coordinates relative to the position of the slider. The length from the center of mass of the slider to the local origin of coordinates of the body is given the ℓ_i symbol.

The canonical form for the absolute coordinates of body i, where $i \in [3, 8]$, is:

$$\begin{aligned} X_i = X_2 + x_i = X_2 + a_i \cos\Theta_3 - b_i \sin\Theta_3 \\ Y_i = y_i = b_i \cos\Theta_3 + a_i \sin\Theta_3 \end{aligned} \tag{3.5}$$

Using the cosine formula and the tangent half-angle formulas with the relative angle approach to solve the variable-sided triangles formed by the actuators, the angle of orientation of the bodies is obtained as an inverse tangent.

However, the trigonometric functions resulting have a canonical form:

$$\begin{aligned} \cos\Theta_i = c_i \cos\Theta_3 - d_i \sin\Theta_3 \\ \sin\Theta_i = d_i \cos\Theta_3 + c_i \sin\Theta_3 \end{aligned} \tag{3.6}$$

Some important properties of these forms are:

$$\begin{gathered} x_i^2 + y_i^2 = a_i^2 + b_i^2 = \ell_i^2 \\ X_i^2 + Y_i^2 = a^2 + b^2 + X_2^2 + X_2(2a\cos\Theta_3 - 2b\sin\Theta_3) = \ell_i^2 + X_2^2 + X_2(2a\cos\Theta_3 - 2b\sin\Theta_3) \\ \cos^2\Theta_i + \sin^2\Theta_i = c_i^2 + d_i^2 = 1 \end{gathered} \tag{3.7}$$

The same is true for the centroid of the model. The center of mass of the model, considering only bodies 3 through 8, is just another point where its canonical form is the same as the one for the absolute coordinates of the bodies:

$$\begin{aligned} X_G = X_2 + x_G = X_2 + a_G \cos\Theta_3 - b_G \sin\Theta_3 \\ Y_G = y_G = b_G \cos\Theta_3 + a_G \sin\Theta_3 \end{aligned} \tag{3.8}$$

The coefficients a_i, b_i, a_G, b_G, c_i and d_i are functions dependent on the actuator lengths $f(La_1, La_2, La_3, La_4, La_5)$.

The moment of inertia of the body, considering only bodies 3 through 8, has no canonical form, but it is also a function of the actuator lengths:

$$I_G = f(La_1, La_2, La_3, La_4, La_5) \tag{3.9}$$

3.2.2.2 Kinematic simulation

The symbolic kinematic solution is a very powerful tool that can be used to obtain a kinematic simulation of the body. The expressions for linear and angular positions, velocities and accelerations of the bodies in terms of the degrees of freedom can be used to recreate the same results of the *SolidWorks* kinematic model.

If the same type of step functions are defined in *Mathematica*, the expressions for the generalized coordinates and its derivatives of the tracer point, or any other point of any other body, can be obtained and compared with the numerical results of *COSMOSMotion*.

The mathematical definitions for the slider position X_2, the angle Θ_3 and the actuators lengths La_1, La_2, La_3, La_4 and La_5 as a function of time are expressed in terms of their initial value and the STEP5 function. Due to STEP5 function being a class C^2, the expressions for velocity and acceleration of all the generalized coordinates of the bodies can be calculated easily.

In order to have the same inputs as in the *COSMOSMotion* simulation, the drivers will be defined as functions of time:

$$\begin{aligned} X_2 &= X_{2_i} + \Delta X_2 \cdot T = -0.5 + 0.5 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (m) \\ \Theta_3 &= \Theta_{3_i} + \Delta\Theta_3 \cdot T = -11.03 + 45 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (^\circ) \\ La_1 &= La_{1_i} + \Delta La_1 \cdot T = 0.242722 + 0.04 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (m) \\ La_2 &= La_{2_i} + \Delta La_2 \cdot T = 0.358221 + 0.01 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (m) \\ La_3 &= La_{3_i} + \Delta La_3 \cdot T = 0.377705 - 0.12 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (m) \\ La_4 &= La_{4_i} + \Delta La_4 \cdot T = 0.322810 - 0.03 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (m) \\ La_5 &= La_{5_i} + \Delta La_5 \cdot T = 0.258034 + 0.01 \cdot \text{STEP5}[T, 0, 0, 1, 1] \ (m) \end{aligned} \tag{3.10}$$

These definitions will be substituted into the symbolic solutions for the bodies and the tracer point coordinates obtained from the kinematic model. These symbolic expressions will also be derived to obtain velocities and accelerations and compared with the solution from *COSMOSMotion*.

3.2.3 Working range

The model has seven degrees of freedom for which the maximum and minimum values must be specified. The position of the slider X_2 has no limitation on the movement, it can move horizontally in an unlimited range. The angular motor that drives the angle Θ_3 has no theoretical limitation, being able to make a complete 360° degree turn around the pivot point, but there must be a range in order to avoid configurations that would not make sense in the real world. The chosen range will be from $-15°$, which corresponds to the folded initial configuration, to 90°, corresponding to a full extended upwards configuration.

Unlike the other two, the five remaining degrees of freedom do have a mathematical limitation for their value. The relative angle definition of the variable-sided triangles that make up the actuators limit the values for their length.

The relative angle of the bodies connected by the actuator are defined as:

$$\begin{aligned}
\Theta_{La_1} &= \cos^{-1}\left(\frac{\left(\frac{L_3}{5}\right)^2+\left(\frac{29L_3}{25}\right)^2-La_1^2}{2(\frac{L_3}{5})(\frac{29L_3}{25})}\right) \\
\Theta_{La_2} &= \cos^{-1}\left(\frac{\left(\frac{2L_3}{5}\right)^2+\left(\frac{3L_5}{2}\right)^2-La_2^2}{2(\frac{2L_3}{5})(\frac{3L_5}{2})}\right) \\
\Theta_{La_3} &= \cos^{-1}\left(\frac{(2L_5)^2+\left(\frac{33L_6}{70}\right)^2-La_3^2}{2(2L_5)(\frac{33L_6}{70})}\right) \\
\Theta_{La_4} &= \cos^{-1}\left(\frac{\left(\sqrt{\left(\frac{7L_5}{5}\right)^2+L_d^2}\right)^2+\left(\frac{3L_7}{10}\right)^2-La_4^2}{2\left(\sqrt{\left(\frac{7L_5}{5}\right)^2+L_d^2}\right)(\frac{3L_7}{10})}\right) \\
\Theta_{La_5} &= \cos^{-1}\left(\frac{\left(\frac{6L_7}{5}\right)^2+\left(\frac{L_8}{4}\right)^2-La_5^2}{2(\frac{6L_7}{5})(\frac{L_8}{4})}\right)
\end{aligned} \tag{3.11}$$

The maximum and minimum lengths obtained from these equations are:

$$\begin{aligned}
La_1 \in \mathbb{R} &: \frac{24}{25}L_3 \le La_1 \le \frac{34}{25}L_3 \\
La_2 \in \mathbb{R} &: \left(\frac{3}{2}L_5-\frac{2}{5}L_3\right) \le La_2 \le \left(\frac{2}{5}L_3+\frac{3}{2}L_5\right) \\
La_3 \in \mathbb{R} &: \left(2L_5-\frac{33}{70}L_6\right) \le La_3 \le \left(2L_5+\frac{33}{70}L_6\right) \\
La_4 \in \mathbb{R} &: \left(\sqrt{\left(\frac{7L_5}{5}\right)^2+L_d^2}-\frac{3}{10}L_3\right) \le La_4 \le \left(\sqrt{\left(\frac{7L_5}{5}\right)^2+L_d^2}+\frac{3}{10}L_3\right) \\
La_5 \in \mathbb{R} &: \left(\frac{6}{5}L_7-\frac{1}{4}L_8\right) \le La_5 \le \left(\frac{6}{5}L_7+\frac{1}{4}L_8\right)
\end{aligned} \tag{3.12}$$

These values should be compared to check if they exceed the limitations of the lengths of the rods and barrels of the cylinder with a minimum value of $2\,L_{10}$ and a maximum value of $4\,L_{10}$.

The resulting values for the limitation of the actuators considering the numerical values of the parameters are:

$$\begin{aligned} La_1 \in \mathbb{R} : 24 \leq La_1 \leq 34\, cm \\ La_2 \in \mathbb{R} : 20 \leq La_2 \leq 40\, cm \\ La_3 \in \mathbb{R} : 23.5 \leq La_3 \leq 34\, cm \\ La_4 \in \mathbb{R} : 23.732 \leq La_4 \leq 35.732\, cm \\ La_5 \in \mathbb{R} : 20 \leq La_5 \leq 29\, cm \end{aligned} \tag{3.13}$$

Considering the trajectory of the tracer point, the limitations of the actuators define a region that, doing an analogy with an excavator, it could be considered as its *working range*. The position of the slider and the angular motion of the motor can be considered as a pure translation and a rotation of this region around the pivot point of the revolute joint.

The position of the tracer point does not depend on the third actuator, therefore there are four parameters that must be varied in order to obtain the region: La_1, La_2, La_4, La_5.

Mathematica has a function to define regions in terms of varying parameters: **ParametricRegion**. The problem is that the complexity of the function for the position of the tracer point does not allow to obtain the region directly, it must be reduced into simpler regions and then combined using a boolean union. The strategy will be to vary two parameters at a time, maintaining the other two at their maximum or minimum points. The result will be combined to obtain the final region.

The **RegionBoundary** function can obtain the contour of the region, but due to the complexity of the resulting region the algorithm can not obtain the bounds of the region. An alternative approach will be considered. By analyzing the working range region, the lines that define the contour will be obtained by using the **ParametricPlot** function. The contour of the region will be defined in different sections, being necessary to obtain the intersection between curves to obtain a closed set.

3.2.4 Dynamic model definition

For the correct understanding of the following elaborations, it is recommended to read chapter D from the appendices where the necessary theory behind kinematics and dynamics of mechanical systems is carefully detailed.

The complete model has seven degrees of freedom: five linear actuators, a motor, and a slider. The model dynamics will be simplified under the assumption that the linear actuators are position controlled and their length can be precisely determined. This leaves the system with two degrees of freedom, resulting in two differential equations of motion that must be obtained: one for Θ_3 and another one for X_2.

It is trivial to observe that if the actuators are statically determined at every moment, the model is equivalent to an *inverted pendulum on a cart*. The cart is the wheel body, with its position determined by the X_2 slider; and the inverted pendulum is the rest of the model, bodies three to eight, with its rotation determined by the angle Θ_3. As the external action force that drives the model, a translational force F applied to the cart will be considered.

Although this is a really considerable simplification, in this case the equivalent pendulum model is not as trivial as the simple pendulum, where the mass is completely located in a single point; but a *physical pendulum*, where the rigid body that swings around the pivot point has a specific mass distribution and moment of inertia.

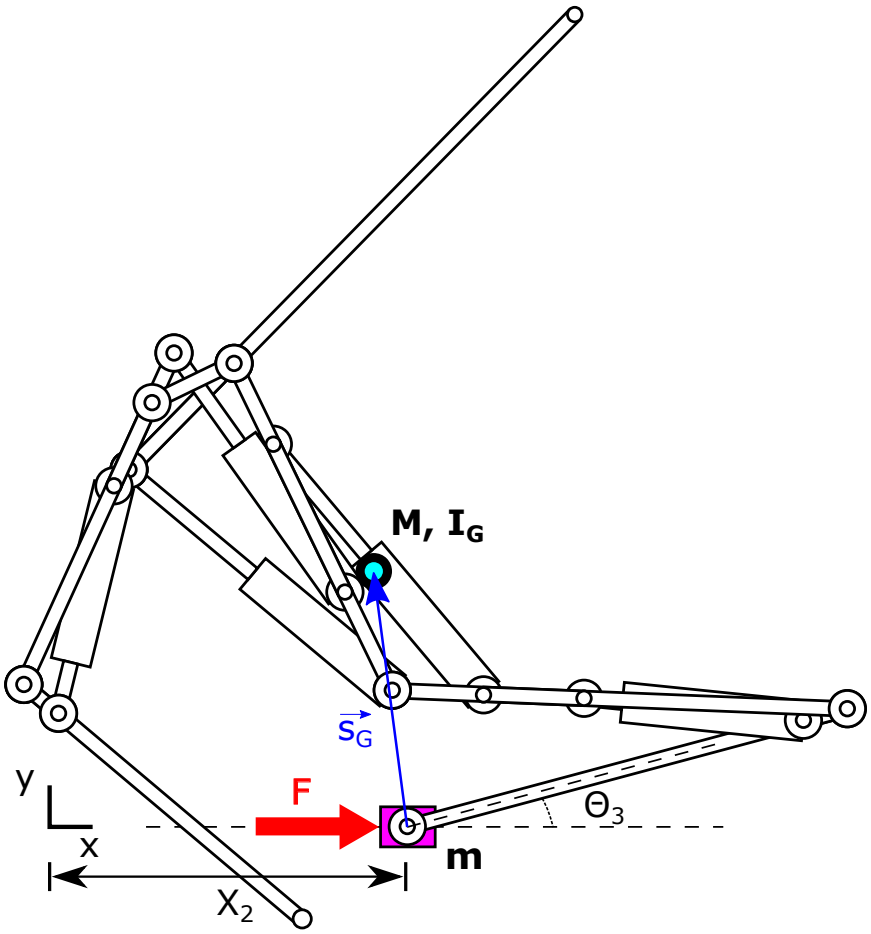


Figure 3.36: Representation of the equivalent inverted pendulum on cart model.

The model can be reduced to a three body system: ground, cart, and pendulum. The ground body is the fixed one which the cart slides on, and the pendulum is attached to the cart allowing it to rotate around the cart's center of gravity. The pendulum's inertial properties must be those of the model: bodies three through eight are replaced by this equivalent pendulum with the same mass, moment of inertia, and center of mass as the combination of the original bodies.

The cart's relevant inertial properties are:

$$\begin{aligned} m = m_2 &= 4HL_2\rho_2 W \\ \mathbf{s_O} &= \{x_O, y_O\}^T = \{0, 0\}^T \end{aligned} \tag{3.14}$$

The pendulum's inertial properties can be defined as:

$$\begin{aligned} M = m_{3,4,5,6,7,8} &= \left(H(4L_3 + 2L_5 + 2L_6 + 2L_7 + 2L_8 + L_d + 24r) - 13\pi r^2\right)\rho W \\ \mathbf{s_G} &= \{x_G, y_G\}^T = \{f(\Theta_3, La_1, La_2, La_3, La_4, La_5), g(\Theta_3, La_1, La_2, La_3, La_4, La_5)\}^T \\ I_G &= h(La_1, La_2, La_3, La_4, La_5) \end{aligned} \tag{3.15}$$

Assuming a strictly rigid body formulation, the masses have constant values for both bodies, they do not depend on the position or orientation of them. The inertia of the pendulum, I_G, depends on the length of the actuators which modify the geometry of the body, but it does not depend on the position of the slider nor the orientation of the pendulum because it is the moment of inertia referred to its center of mass. The $\mathbf{s_G}$ vector is relative to the pendulum's local coordinate origin, located at the place where the revolute joint connects both bodies, and is not dependent on the position of the cart. The mathematical expressions for the moment of inertia and the center of mass of the equivalent pendulum are obtained from the symbolic kinematic model solution.

It is important to note that the angle Θ_3 is not the angle that the center of mass of the pendulum forms with the horizontal, but the angle the original lower leg body forms. Considering the rotation transformation matrix, the coordinates of the local vector of the center of mass, x_G and y_G, will have two addends: a cosine and a sine. The values of the coefficients, a and b, are the coordinates of the same vector calculated with an rotation angle of $\Theta_3 = 0$. The coordinates of the center of mass will have the following expression:

$$\begin{aligned} \mathbf{s_G} &= \{x_G, y_G\}^T \\ x_G &= a\cos\Theta_3 - b\sin\Theta_3 \\ y_G &= b\cos\Theta_3 + a\sin\Theta_3 \end{aligned} \tag{3.16}$$

Being both coefficients $a = f(La_1, La_2, La_3, La_4, La_5)$ and $b = g(La_1, La_2, La_3, La_4, La_5)$ functions dependent on the length of the actuators.

The absolute distance to the center of mass is calculated as the Euclidean norm of the vector, and this distance must not depend on the angle Θ_3. The effect the angle has is the rotation of the center of mass around the anchor point at the joint between the cart, but it does not modify its absolute distance. The norm of the vector must only depend on the coefficients.

$$\begin{aligned}\|\boldsymbol{s_G}\| &= \sqrt{x_G^2 + y_G^2} = \sqrt{(a\cos\Theta_3 - b\sin\Theta_3)^2 + (b\cos\Theta_3 + a\sin\Theta_3)^2} = \\ &= \sqrt{(a^2+b^2)\cos^2\Theta_3 + (a^2+b^2)\sin^2\Theta_3} = \sqrt{(a^2+b^2)} = \ell\end{aligned} \tag{3.17}$$

The norm of the vector, which corresponds to the length from the local origin of the pendulum to its center of mass, will receive the ℓ symbol.

From now on, a simplification in the nomenclature will be introduced, because there are only two variables and one is an angle and another is a distance, there is no possible confusion and therefore, $X_2 = x$ and $\Theta_3 = \Theta$. This will make the expressions easier to follow without unnecessary subscripts.

Interesting properties regarding the partial derivatives of the coordinates will be presented, which will prove to be necessary afterwards:

$$\begin{aligned}\frac{\partial x_G}{\partial\Theta} &= -a\sin\Theta - b\cos\Theta = -y_G \\ \frac{\partial y_G}{\partial\Theta} &= a\cos\Theta - b\sin\Theta = x_G \\ \frac{\partial^2 x_G}{\partial\Theta^2} &= -a\cos\Theta + b\sin\Theta = -x_G \\ \frac{\partial^2 y_G}{\partial\Theta^2} &= -b\cos\Theta - a\sin\Theta = -y_G\end{aligned} \tag{3.18}$$

Throughout the development that follows, the chain rule for first and second order derivatives will be necessary. Derivatives of functions that depend on angle Θ must be calculated and the dependency of this angle with time must be taken into account.

Considering f a function of Θ and Θ is a function of time, this is $f(\Theta(t))$, the chain rule for first and second order derivatives gives the following expressions:

$$\begin{aligned}\frac{df}{dt} &= \frac{\partial f}{\partial\Theta}\frac{\partial\Theta}{\partial t} \\ \frac{d^2 f}{dt^2} &= \frac{\partial^2 f}{\partial\Theta^2}\left(\frac{\partial\Theta}{\partial t}\right)^2 + \frac{\partial f}{\partial\Theta}\frac{\partial^2\Theta}{\partial t^2}\end{aligned} \tag{3.19}$$

The model is therefore further simplified to a well known equivalent model of an inverted pendulum on a cart. The next step is to obtain the dynamic equations of motion of the system, and they will be obtained in three different ways which result in the same expressions.

3.2.4.1 Inverted pendulum using Lagrangian mechanics

Using Lagrangian mechanics the equations of motion of the system can be easily obtained. The generalized coordinates chosen are the absolute position of the cart, X_2, and the relative angle of the pendulum, Θ_3. The generalized external non-conservative forces in this case is only the force that drives the cart, F.

In the inverted pendulum on cart system, there are two bodies that must be taken into consideration. The first body is the cart, which has linear velocity but no rotational motion and therefore no angular velocity. The pendulum has both linear velocity and angular velocity around the joint with the cart. The linear and angular velocity of the pendulum will be taken about its center of mass, but it could be taken in any given point and it would be equivalent, if the moment of inertia is adequately calculated. If the origin of potential energy is defined at the x axis, the only body with potential energy is the pendulum. The angular velocity of the cart is $\omega = \dot{\Theta}$.

The expression for the Lagrangian of the system is:

$$L = \frac{1}{2}mv_1^2 + \frac{1}{2}Mv_2^2 + \frac{1}{2}I_G\dot{\Theta}^2 - My_Gg \tag{3.20}$$

The velocity expressed in this equation is absolute, referring to the global coordinate system. The linear velocity of the cart is equal to the first derivative of the x position. The linear and angular velocity of the pendulum are taken at its center of mass, being the absolute position vector equal to $\mathbf{s_G}$ plus the x position of the cart in the x coordinate.

$$v_1^2 = \dot{x}^2 \tag{3.21}$$

$$v_2^2 = \left(\frac{d}{dt}(x + x_G)\right)^2 + \left(\frac{d}{dt}y_G\right)^2 = \left(\frac{dx}{dt} + \frac{\partial x_G}{\partial\Theta}\frac{\partial\Theta}{\partial t}\right)^2 + \left(\frac{\partial y_G}{\partial\Theta}\frac{\partial\Theta}{\partial t}\right)^2 =$$

$$= \left(\dot{x} + \frac{\partial x_G}{\partial\Theta}\dot{\Theta}\right)^2 + \left(\frac{\partial y_G}{\partial\Theta}\dot{\Theta}\right)^2 = \dot{x}^2 + 2\frac{\partial x_G}{\partial\Theta}\dot{\Theta}\dot{x} + \left(\frac{\partial x_G}{\partial\Theta}^2 + \frac{\partial y_G}{\partial\Theta}^2\right)\dot{\Theta}^2$$

The velocity of the second body can be simplified because the term $\left(\frac{\partial x_G}{\partial\Theta}^2 + \frac{\partial y_G}{\partial\Theta}^2\right)$ is equal to the square of the distance from the joint to the center of mass of the pendulum, ℓ. This is due to the nature of the $\mathbf{s_G}$ vector in which the coefficients are independent of the angle Θ and the norm of the vector is therefore only dependent on the length of the actuators.

The resulting Lagrangian of the system is:

$$L = \frac{1}{2}(M + m)\dot{x}^2 + M\frac{\partial x_G}{\partial\Theta}\dot{\Theta}\dot{x} + \frac{1}{2}M\ell^2\dot{\Theta}^2 + \frac{1}{2}I_G\dot{\Theta}^2 - My_G\,g \tag{3.22}$$

Once the Lagrangian has been obtained, the next step is to obtain the equations of motion particularizing the Euler-Lagrange equation for the degrees of freedom (generalized coordinates) of the system.

For the $q = x$ generalized coordinate, the Euler-Lagrange equation is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = F \tag{3.23}$$

Differentiating the Lagrangian the two terms can be obtained:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left((M+m)\dot{x} + M\frac{\partial x_G}{\partial \Theta}\dot{\Theta}\right) = (M+m)\ddot{x} + M\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 + M\frac{\partial x_G}{\partial \Theta}\ddot{\Theta} \tag{3.24}$$

$$\frac{\partial L}{\partial x} = 0$$

Resulting in the first equation of motion:

$$\boxed{(M+m)\ddot{x} + M\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 + M\frac{\partial x_G}{\partial \Theta}\ddot{\Theta} = F} \tag{3.25}$$

Taking $q = \Theta$ into the Euler-Lagrange equation, in this case with no external torque applied:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\Theta}}\right) - \frac{\partial L}{\partial \Theta} = 0 \tag{3.26}$$

Proceeding in the same way as before, differentiating the Lagrangian:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\Theta}}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left((I_G + M\ell^2)\dot{\Theta} + M\frac{\partial x_G}{\partial \Theta}\dot{x}\right) = M\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}\dot{x} + M\frac{\partial x_G}{\partial \Theta}\ddot{x} + (I_G + M\ell)\ddot{\Theta} \tag{3.27}$$

$$\frac{\partial L}{\partial \Theta} = M\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}\dot{x} - M\frac{\partial y_G}{\partial \Theta}g$$

Substituting into the equation and simplifying, the second equation of motion is obtained:

$$\boxed{M\frac{\partial x_G}{\partial \Theta}\ddot{x} + (I_G + M\ell^2)\ddot{\Theta} + M\frac{\partial y_G}{\partial \Theta}g = 0} \tag{3.28}$$

3.2.4.2 Inverted pendulum using Newton's laws

The same result can be obtained using Newton's laws, but a deeper analysis of the forces in the system must be performed. When following this method, a free-body diagram for each body must be done, identifying and obtaining all the reaction forces in each of the bodies.

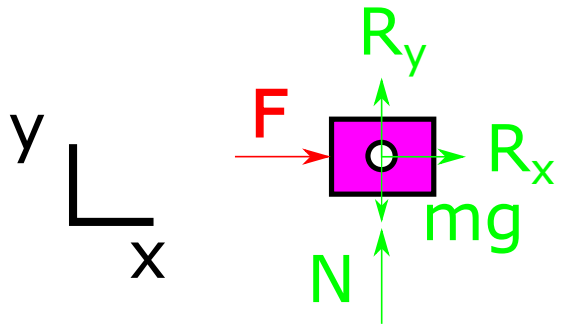


Figure 3.37: Free-body diagram of the first body (cart).

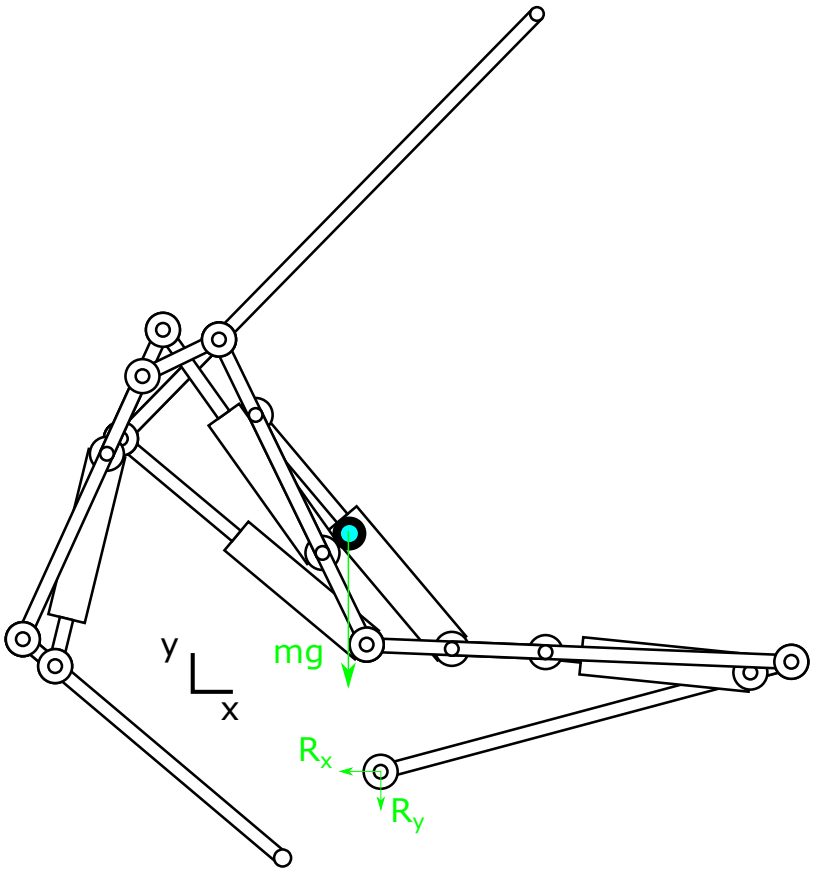


Figure 3.38: Free-body diagram of the second body (pendulum).

Applying Newton' laws to the first body:

$$\begin{aligned} F + R_x &= m a_{x_1} \\ F_y - mg + N &= m a_{y_1} \\ 0 &= I_o \alpha_1 \end{aligned} \tag{3.29}$$

Applying Newton' laws to the second body:

$$\begin{aligned} -R_x &= M a_{x_2} \\ -Mg - R_y &= M a_{y_2} \\ R_y x_G - R_x y_G &= I_G \alpha_2 \end{aligned} \tag{3.30}$$

From the equations of the second body:

$$\begin{aligned} R_x &= -M a_{x_2} \\ R_y &= -M(a_{y_2} + g) \end{aligned} \tag{3.31}$$

Substituting these terms into the first equation of the first body:

$$F = m a_{x_1} + M a_{x_2} \tag{3.32}$$

The acceleration term of the first body is directly the second derivative of the x position, while the second derivative of the second body can be calculated using the chain rule for second order derivatives:

$$\begin{aligned} a_{x_1} &= \frac{d^2}{dt^2} x = \ddot{x} \\ a_{x_2} &= \frac{d^2}{dt^2}(x + x_G) = \ddot{x} + \frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 + \frac{\partial x_G}{\partial \Theta}\ddot{\Theta} \end{aligned} \tag{3.33}$$

Which yields the first equation of motion again:

$$\boxed{(M+m)\ddot{x} + M\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 + M\frac{\partial x_G}{\partial \Theta}\ddot{\Theta} = F} \tag{3.34}$$

Substituting the reaction force terms into the third equation of the third body:

$$-M(a_{y_2} + g)x_G + M a_{x_2} y_G = I_G \alpha_2 = I_G \ddot{\Theta} \tag{3.35}$$

Calculating the term of the acceleration in the y axis for the first body:

$$a_{y_2} = \frac{d^2}{dt^2} y_G = \frac{\partial^2 y_G}{\partial \Theta^2}\dot{\Theta}^2 + \frac{\partial y_G}{\partial \Theta}\ddot{\Theta} \tag{3.36}$$

Substituting the acceleration terms and reorganizing the equation:

$$(I_G - M(\frac{\partial x_G}{\partial \Theta} y_G - \frac{\partial y_G}{\partial \Theta} x_G))\ddot{\Theta} - M y_G \ddot{x} + M x_G g - M\dot{\Theta}^2(\frac{\partial^2 x_G}{\partial \Theta^2} y_G - \frac{\partial^2 y_G}{\partial \Theta^2} x_G) = 0 \tag{3.37}$$

Using the properties of the partial derivatives of the coordinates with respect of Θ:

$$(\frac{\partial^2 x_G}{\partial \Theta^2} y_G - \frac{\partial^2 y_G}{\partial \Theta^2} x_G) = (-x_G) y_G - (-y_G) x_G = 0 \tag{3.38}$$

$$(\frac{\partial x_G}{\partial \Theta} y_G - \frac{\partial y_G}{\partial \Theta^2} x_G) = (-y_G) y_G - (x_G) x_G = -(y_G^2 + x_G^2) = -\ell^2$$

$$y_G = -\frac{\partial x_G}{\partial \Theta}$$

$$x_G = \frac{\partial y_G}{\partial \Theta}$$

The substitution of these terms in the equation yields the second equation of motion again:

$$\boxed{M\frac{\partial x_G}{\partial \Theta}\ddot{x} + (I_G + M\ell^2)\ddot{\Theta} + M\frac{\partial y_G}{\partial \Theta} g = 0} \tag{3.39}$$

This proves that Lagrangian mechanics and Newton's laws are equivalent methods for obtaining the equations of motion of a mechanical system.

3.2.4.3 Inverted pendulum using constraint equations

The third way of obtaining the equations of motion might seem more complicated than the previous ones, but it is more easily automatized and more adequate for mechanical systems that have many different bodies. In this case the constraint equations that reign the interaction between bodies must be taken into account.

The simplified model contains two relevant bodies that will be defined using their masses, inertias, and centers of gravity. No parameters will be substituted, everything will be evolved using a symbolic approach with the help of *Mathematica* and its *MechanicalSystems* package. The resulting equations should match those of the two previous sections.

The local origin of the slider is located in its center of mass; while the local origin of the pendulum is located in the junction point with the slider, which is a different point than its center of mass. The constraints between the bodies must be defined. The first constraint is a translational joint between the ground body and the slider, which introduces two constraint equations. The second constraint is the revolute joint between the slider and the pendulum, which again introduces two constraint equations. The x position of the second body (slider) and the angle of rotation of the third body (pendulum) will be driven constraints. The constraint equation vector for the system is:

$$\mathbf{\Phi}(\mathbf{q},t) = \begin{pmatrix} 4L_1L_2\sin\Theta_2 \\ 2L_2(L_1+X_2-L_2\cos\Theta_2)\sin\Theta_2 + 2L_2\cos\Theta_2(-Y_2+L_2\sin\Theta_2) \\ X_2 - X_3 \\ Y_2 - Y_3 \\ X_2 - pos(t) \\ \Theta_3 - ang(t) \end{pmatrix} = \mathbf{0} \tag{3.40}$$

The mass matrix of the system will have terms out of the diagonal due to the eccentricity of the pendulum's center of mass:

$$\mathbf{M} = \begin{pmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & Io & 0 & 0 & 0 \\ 0 & 0 & 0 & M & 0 & M(-x_G\sin(\Theta_3) - y_G\cos(\Theta_3)) \\ 0 & 0 & 0 & 0 & M & M(x_G\cos(\Theta_3) - y_G\sin(\Theta_3)) \\ 0 & 0 & 0 & M(-x_G\sin(\Theta_3) - y_G\cos(\Theta_3)) & M(x_G\cos(\Theta_3) - y_G\sin(\Theta_3)) & I_G + M(x_G^2 + y_G^2) \end{pmatrix} \tag{3.41}$$

The generalized coordinates, generalized acceleration and the Lagrange multiplier vectors are:

$$\mathbf{q} = \begin{pmatrix} X_2 \\ Y_2 \\ \Theta_2 \\ X_3 \\ Y_3 \\ \Theta_3 \end{pmatrix} \qquad \ddot{\mathbf{q}} = \begin{pmatrix} \ddot{X}_2 \\ \ddot{Y}_2 \\ \ddot{\Theta}_2 \\ \ddot{X}_3 \\ \ddot{Y}_3 \\ \ddot{\Theta}_3 \end{pmatrix} \qquad \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \end{pmatrix}$$

The Jacobian of the constraint equations is:

$$\mathbf{\Phi_q}(\mathbf{q},t)=\begin{pmatrix} 0 & 0 & 4L_1L_2\cos\Theta_2 & 0 & 0 & 0\\ 2L_2\sin\Theta 2 & -2L_2\cos\Theta_2 & 2L_2^2\cos^2\Theta_2+2L_2\cos\Theta_2(L_1+X_2-L_2\cos\Theta 2)+2L_2^2\sin^2\Theta 2-2L_2\sin\Theta 2(-Y_2+L_2\sin\Theta 2) & 0 & 0 & 0\\ 1 & 0 & 0 & -1 & 0 & 0\\ 0 & 1 & 0 & 0 & -1 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.42}$$

The centrifugal forces and the generalized applied forces vectors are:

$$\mathbf{C}=\begin{pmatrix} 0\\ 0\\ 0\\ M(\dot\Theta_3)^2(-x_G\cos\Theta_3+y_G\sin\Theta_3)\\ M(\dot\Theta_3)^2(-x_G\sin\Theta_3-y_G\cos\Theta_3)\\ 0 \end{pmatrix} \qquad \mathbf{Q^A}=\begin{pmatrix} F\\ -gm\\ 0\\ 0\\ -gM\\ gM(-x_G\cos\Theta_3+y_G\sin\Theta_3) \end{pmatrix} \tag{3.43}$$

The only external applied forces are the gravitational field and the control force, F. The eccentricity of the pendulum's center of mass creates centrifugal forces and a torque due to the gravitational pull.

From the constraint equation vector, the kinematic solution for two generalized coordinates can be obtained: $\Theta_2=0$, $Y_2=0$, and $Y_3=0$. Substituting the values of these generalized coordinates and their derivatives, which are equal to 0, will simplify the equations significantly. The Lagrange multipliers designated for the driven equations of motion, which are the last two equations of the constraint equation vector, are also equal to 0 because these are the unconstrained degrees of freedom of the system, therefore $\lambda_5=0$ and $\lambda_6=0$.

The result from applying the general Lagrange multiplier form of the constrained equations of motion is:

$$\begin{gathered} m\ddot X_2+\lambda_3=F \\ \lambda_4-2L_2\lambda_2=-gm \\ 4L_1L_2\lambda_1+(2L_2^2)+2(L_1-L_2+X_2)L_2)\lambda_2=0 \\ -M(x_G\cos\Theta_3-y_G\sin\Theta_3)(\dot\Theta_3)^2+M\ddot X_2-\lambda_3+M\ddot\Theta_3(-x_G\sin\Theta_3-y_G\cos\Theta_3)=0 \\ -M(x_G\sin\Theta_3+y_G\cos\Theta_3)(\dot\Theta_3)^2-\lambda_4+M\ddot\Theta_3(x_G\cos\Theta_3-y_G\sin\Theta_3)=-gM \\ (I_G+M(x_G^2+y_G^2))\ddot\Theta_3+M\ddot X_2(-x_G\sin\Theta_3-y_G\cos\Theta_3)=gM(-x_G\cos\Theta_3+y_G\sin\Theta_3) \end{gathered} \tag{3.44}$$

Considering the expressions of the partial derivatives of the local position vector of the center of mass of the pendulum, the last equation of this set of equations is directly the second equation of motion:

$$(I_G+M(x_G^2+y_G^2))\ddot\Theta_3+M\ddot X_2(-x_G\sin\Theta_3-y_G\cos\Theta_3)=gM(-x_G\cos\Theta_3+y_G\sin\Theta_3) \tag{3.45}$$

$$\boxed{M\frac{\partial x_G}{\partial\Theta}\ddot x+(I_G+M\ell^2)\ddot\Theta+M\frac{\partial y_G}{\partial\Theta}g=0}$$

The first equation of motion is obtained by obtaining the expression of λ_3 from the first equation and substituting in the fourth equation.

$$\lambda_3 = F - m\ddot{X}_2 \tag{3.46}$$

$$-M(x_G \cos\Theta_3 - y_G \sin\Theta_3)(\dot{\Theta}_3)^2 + M\ddot{X}_2 - F + m\ddot{X}_2 + M\ddot{\Theta}_3(-x_G \sin\Theta_3 - y_G \cos\Theta_3) = 0$$

$$\boxed{(M+m)\ddot{x} + M\frac{\partial^2 x_G}{\partial\Theta^2}\dot{\Theta}^2 + M\frac{\partial x_G}{\partial\Theta}\ddot{\Theta} = F}$$

As expected, the three methods used for obtaining the equations of motion yielded the same results.

3.2.4.4 Space state model

Considering the equations of motion of the inverted pendulum system with condensed coefficients in the form:

$$\begin{aligned} a_1\ddot{x} + b_1\ddot{\Theta} + c_1\dot{\Theta}^2 &= F \\ a_2\ddot{x} + b_2\ddot{\Theta} + d_2 &= 0 \end{aligned} \tag{3.47}$$

Where:

$$\begin{aligned} a_1 &= (M+m) \qquad b_1 = \frac{\partial x_G}{\partial \Theta}M \qquad c_1 = \frac{\partial^2 x_G}{\partial \Theta^2}M \\ a_2 &= \frac{\partial x_G}{\partial \Theta}M \qquad b_2 = (\ell^2 M + I_G) \qquad d_2 = M\frac{\partial y_G}{\partial \Theta}g \end{aligned} \tag{3.48}$$

This system is non-linear and time-variant, and its space state equations are:

$$\begin{aligned} \dot{x}_1 &= \dot{x} = x_2 \\ \dot{x}_2 = \ddot{x} &= \frac{b_1 d_2 + b_2(F - c_1\dot{\Theta}^2)}{a_1 b_2 - a_2 b_1} = \frac{M^2 \frac{\partial x_G}{\partial \Theta}\frac{\partial y_G}{\partial \Theta}g + (\ell^2 M + I_G)(F - M(\frac{\partial^2 x_G}{\partial \Theta^2})\dot{\Theta}^2)}{(M+m)(\ell^2 M + I_G) - (\frac{\partial x_G}{\partial \Theta})^2 M^2} \\ \dot{x}_3 &= \dot{\Theta} = x_4 \\ \dot{x}_4 = \ddot{\Theta} &= \frac{-a_1 d_2 + a_2(c_1\dot{\Theta}^2 - F)}{a_1 b_2 - a_2 b_1} = \frac{-(M+m)M\frac{\partial y_G}{\partial \Theta}g + M\frac{\partial x_G}{\partial \Theta}(M(\frac{\partial^2 x_G}{\partial \Theta^2})\dot{\Theta}^2 - F)}{(M+m)(\ell^2 M + I_G) - (\frac{\partial x_G}{\partial \Theta})^2 M^2} \\ y_1 &= x_1 = x \\ y_2 &= x_3 = \Theta \end{aligned} \tag{3.49}$$

The non-linear space state equations can be written in general space state form:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ \mathbf{y}(\mathbf{t}) &= \mathbf{C}\,\mathbf{x}(t) \end{aligned} \tag{3.50}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ \Theta \\ \dot{\Theta} \end{pmatrix} \qquad \dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\Theta} \\ \ddot{\Theta} \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} F \\ 0 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, t) = \begin{pmatrix} x_2 \\ \frac{gM^2\frac{\partial x_G}{\partial x_3}\frac{\partial y_G}{\partial x_3} - Mx_4^2(I_G+\ell^2 M)(\frac{\partial^2 x_G}{\partial x_3^2})}{(m+M)(I_G+\ell^2 M) - M^2(\frac{\partial x_G}{\partial x_3})^2} + F\frac{(I_G+\ell^2 M)}{(m+M)(I_G+\ell^2 M) - M^2(\frac{\partial x_G}{\partial x_3})^2} \\ x_4 \\ \frac{-gM(m+M)\frac{\partial y_G}{\partial x_3} + M^2 x_4^2\frac{\partial x_G}{\partial x_3}(\frac{\partial^2 x_G}{\partial x_3^2})}{(m+M)(I_G+\ell^2 M) - M^2(\frac{\partial x_G}{\partial x_3})^2} + F\frac{-M\frac{\partial x_G}{\partial x_3}}{(m+M)(I_G+\ell^2 M) - M^2(\frac{\partial x_G}{\partial x_3})^2} \end{pmatrix}$$

The system state equations must be linearized before proceeding with further analysis.

3.2.4.5 Operating point

The linearization of a non-linear model is a linear approximation around a given operating point, and it is only valid in a small region around this point. The selected operating point is usually the equilibrium point of the system, around which we want our system to work.

The system has two equilibrium points, one stable and the other unstable. The stable equilibrium point is when the pendulum is hanging down, and the unstable equilibrium point represents the pendulum in its upwards position. In both of them the derivatives of the state variables are equal to 0 and the x position of the cart can take any value.

The equilibrium is reached when the relative x coordinate of the center of mass of the pendulum is equal to 0, that is when the center of mass of the pendulum is in the horizontal from the cart's center of mass. In order to find the equilibrium points the following equation must be solved:

$$x_G = a\cos\Theta - b\sin\Theta = 0 \tag{3.51}$$

Using the tangent half angle formulas this equation is converted to a second order polynomial equation:

$$\begin{gathered} x = \tan(\frac{\Theta}{2}) \\ \frac{a\left(1-x^2\right)}{x^2+1} - \frac{2bx}{x^2+1} = 0 \\ a(1-x^2) - 2bx = 0 \\ x_1 = \frac{-b-\sqrt{a^2+b^2}}{a} \qquad x_2 = \frac{-b+\sqrt{a^2+b^2}}{a} \\ \Theta_1 = 2\tan^{-1}(x_1) = 2\tan^{-1}\left(\frac{-b-\ell}{a}\right) \\ \Theta_2 = 2\tan^{-1}(x_2) = 2\tan^{-1}\left(\frac{-b+\ell}{a}\right) \end{gathered} \tag{3.52, 3.53}$$

Because $a = f(La_1, La_2, La_3, La_4, La_5)$ and $b = g(La_1, La_2, La_3, La_4, La_5)$ it is possible to find the equilibrium angle at any moment. The angle corresponding to the unstable equilibrium point is the positive and close to zero one:

$$\Theta_{eq} = 2\tan^{-1}\left(\frac{-b+\ell}{a}\right) \tag{3.54}$$

There is another way to obtain the equilibrium point, this case using the expression of the y coordinate of the center of mass. This alternative way allows to distinguish between the upwards and downwards position from the beginning. In the upwards position the y coordinate is equal to the length to the local origin, $y_G = \ell$; while in the downwards position the y coordinate is equal to the length with a negative sign. The equation to be solved is:

$$y_G = b\cos\Theta + a\sin\Theta = \sqrt{a^2+b^2} = \ell \tag{3.55}$$

Proceeding in the same way as before:

$$x = \tan(\frac{\Theta}{2}) \tag{3.56}$$

$$\begin{gathered}
\frac{b\left(1-x^2\right)}{x^2+1}+\frac{2ax}{x^2+1}=\sqrt{a^2+b^2} \\
b(1-x^2)+2ax=(x^2+1)\sqrt{a^2+b^2} \\
x_1=\frac{a}{b+\sqrt{a^2+b^2}} \qquad x_2=\frac{a}{b+\sqrt{a^2+b^2}} \\
\Theta=2\tan^{-1}(x)=2\tan^{-1}\left(\frac{a}{b+\ell}\right)
\end{gathered} \tag{3.57}$$

This solution yields the correct equilibrium angle directly, but with a different expression.

$$\Theta_{eq}=2\tan^{-1}\left(\frac{a}{b+\ell}\right) \tag{3.58}$$

As expected, the equilibrium point only depends on the actuator lengths.

Lets consider the dynamic equations of motion again:

$$M\frac{\partial x_G}{\partial\Theta}\ddot{x}+(I_G+M\ell^2)\ddot{\Theta}+M\frac{\partial y_G}{\partial\Theta}g=0 \tag{3.59}$$

$$(M+m)\ddot{x}+M\frac{\partial^2 x_G}{\partial\Theta^2}\dot{\Theta}^2+M\frac{\partial x_G}{\partial\Theta}\ddot{\Theta}=F \tag{3.60}$$

It is trivial to observe from these equations that, in a ideal mechanical formulation, if the system is as its equilibrium point, where $x_G = 0$ and $y_G = \ell$, and all the velocities and accelerations are equal to zero, the resulting force needed to maintain this position is $F = 0$.

3.2.4.6 Linearization

If the system is expressed using linearized space state representation around the operating point determined by $\mathbf{x_o}$ and $\mathbf{u_o}$:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}(\mathbf{x_o}, \mathbf{u_o})\,\mathbf{x}(t) + \mathbf{B}(\mathbf{x_o}, \mathbf{u_o})\,\mathbf{u}(t) \tag{3.61}$$
$$\mathbf{y(t)} = \mathbf{C}\,\mathbf{x}(t)$$

Where:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \\ \Theta \\ \dot{\Theta} \end{pmatrix} \qquad \mathbf{x_o} = \begin{pmatrix} 0 \\ 0 \\ \Theta_{eq} \\ 0 \end{pmatrix} \qquad \mathbf{u} = \begin{pmatrix} F \\ 0 \end{pmatrix} \qquad \mathbf{u_o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.62}$$
$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Taking into consideration the expression of the relative coordinates of the center of mass the pendulum x_G and y_G and the expression of their derivatives, the values of the state variables and their derivatives, and that at the equilibrium $x_G = 0$ and $y_G = \ell$, the linearized matrices of coefficients around the operating point are:

$$\mathbf{A}(\mathbf{x_o}, \mathbf{u_o}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{g\ell^2M^2}{I_G(m+M)+\ell^2mM} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g\ell M(m+M)}{I_G(m+M)+\ell^2mM} & 0 \end{pmatrix} \qquad \mathbf{B}(\mathbf{x_o}, \mathbf{u_o}) = \begin{pmatrix} 0 & 0 \\ \frac{I_G+\ell^2M}{I_G(m+M)+\ell^2mM} & 0 \\ 0 & 0 \\ \frac{\ell M}{I_G(m+M)+\ell^2mM} & 0 \end{pmatrix}$$

It is quite important to notice that the linearized system is done for the angle Θ_{eq} which depends on the configuration of the model (the lengths of the actuators), but the results are independent of this angle.

The length of the actuators are considered in $\ell = \sqrt{x_G^2 + y_G^2} = f(La_1, La_2, La_3, La_4, La_5)$ and $I_G = h(La_1, La_2, La_3, La_4, La_5)$, the linearized system is valid for any combination of the actuator lengths while the ℓ, I_G, and Θ_{eq} parameters are correctly updated.

From this linearized model it is easy to obtain the transfer functions of the system. Applying the Laplace transform to the space state equations, obtaining the state vector from the state equation and substituting into the output equation, the transfer functions are:

$$\frac{\Theta(s)}{\mathbb{F}(s)} = \frac{\ell M}{(I_G(m+M)+\ell^2mM)s^2 - g\ell M(m+M)} \left[\frac{rad}{N}\right] \tag{3.63}$$
$$\frac{\mathbb{X}(s)}{\mathbb{F}(s)} = \frac{s^2(I_G+\ell^2M) - g\ell M}{(I_G(m+M)+\ell^2mM)s^4 - g\ell M(m+M)s^2} \left[\frac{m}{N}\right]$$

The poles of the linearized system can be obtained in order to check the stability of the open-loop configuration. The condition for stability that must verify is that the poles s_i have a negative real part, $Re(s_i) < 0$, this would mean that the system without any controller is stable.

The poles are the eigenvalues of the linearized system matrix $\mathbf{A}$ which are the solution to the following equation:

$$\det(s\,\mathbf{I} - \mathbf{A}) = 0 \tag{3.64}$$

$$\begin{vmatrix} s & 1 & 0 & 0 \\ 0 & s & -\frac{g\ell^2M^2}{I_G(m+M)+\ell^2mM} & 0 \\ 0 & 0 & s & 1 \\ 0 & 0 & -\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2mM} & s \end{vmatrix} = 0$$

$$s^4 - s^2\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2mM} = 0$$

$$s_1 = 0 \qquad s_2 = 0 \qquad s_3 = \sqrt{\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2mM}} \qquad s_4 = -\sqrt{\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2mM}}$$

As expected, the system has a pole on the positive real semi-plane and therefore it is unstable in open-loop configuration.

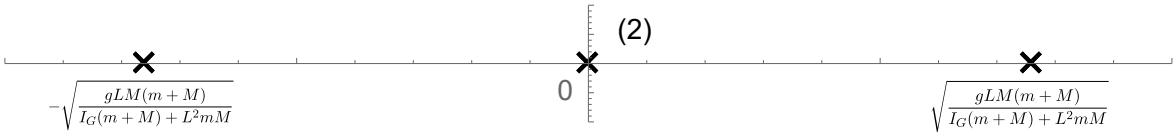


Figure 3.39: Root locus plot for the inverted pendulum system.

To make sure the system is correctly defined, the controllability and observability of the system will be checked. For the inverted pendulum system, the controllability matrix $\mathbf{R}$ and the observability matrix $\mathbf{O}$ are:

$$\mathbf{R} = \begin{pmatrix} 0 & \frac{M\ell^2+I_G}{mM\ell^2+I_G(m+M)} & 0 & \frac{g\ell^3M^3}{(mM\ell^2+I_G(m+M))^2} \\ \frac{M\ell^2+I_G}{mM\ell^2+I_G(m+M)} & 0 & \frac{g\ell^3M^3}{(mM\ell^2+I_G(m+M))^2} & 0 \\ 0 & \frac{\ell M}{mM\ell^2+I_G(m+M)} & 0 & \frac{g\ell^2M^2(m+M)}{(mM\ell^2+I_G(m+M))^2} \\ \frac{\ell M}{mM\ell^2+I_G(m+M)} & 0 & \frac{g\ell^2M^2(m+M)}{(mM\ell^2+I_G(m+M))^2} & 0 \end{pmatrix} \quad \mathbf{O} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g\ell^2M^2}{mM\ell^2+I_G(m+M)} & 0 \\ 0 & 0 & \frac{g\ell M(m+M)}{mM\ell^2+I_G(m+M)} & 0 \\ 0 & 0 & 0 & \frac{g\ell^2M^2}{mM\ell^2+I_G(m+M)} \\ 0 & 0 & 0 & \frac{g\ell M(m+M)}{mM\ell^2+I_G(m+M)} \end{pmatrix} \tag{3.65}$$

Both matrices have a rank of 4, the same as the number of states, therefore the system is controllable and observable.

The linearized system depends on six parameters: g, ℓ, m, M, I_G, and Θ_{eq}. The masses (m, M), and the acceleration of gravity are constants, while ℓ, I_G and Θ_{eq} depend on the actuator lengths and are subject to change during the simulation.

3.2.4.7 Dynamic simulation

In order to check if the results of the simplified inverted pendulum model are correct, the method of constraint equations with the complete eight body system will be solved numerically. These constraint equations are the same ones that were used to obtain the symbolic kinematic solution of the model.

Once the inertial properties of bodies, constraints, and generalized loads of the system are defined, the **SolveMech** function of the *MechanicalSystems* add-in returns the numerical solution to the equations. This numerical solution includes the generalized coordinates and their derivatives, as well as the values of the Lagrange multipliers.

Substituting the numerical values into the two equations obtained from the simplified model and solving the differential equations numerically using the **NDSolve** function in *Mathematica* should yield the same solution as the numerical solution of the complete system obtained with **SolveMech**.

It is also possible to obtain the equations of motion of this complete eight body system by setting the actuator lengths to constant values and only considering variations in the slider position X_2 and angle Θ_3. If the general Lagrange multiplier form of the constrained equations of motion is applied to this system, two equations of motion can also be obtained that must yield the same results as the ones obtained from the simplified inverted pendulum on cart model.

The dynamic simulation performed in *Mathematica* will be a free motion movement under the action of gravitational force. The actuator lengths will be set at their initial position and the slider position and the pendulum angle will be let free to swing. The results will be compared with the same simulation in *RecurDyn*. Further dynamic simulations using controllers designed with the linearized model obtained will be performed in *RecurDyn* and *SystemModeler*.

3.3 Dynamic control in *RecurDyn*

The dynamic analysis of the model will be performed using *RecurDyn*, an industry-leader computer aided engineering software in the field of multibody dynamics simulation. This software allows importing geometry from other proprietary or open source CAD formats as well as creating the model in its own engine from the ground up. *RecurDyn* is primarily focused toward simulation and it also provides a built-in toolbox, *CoLink*, to develop control oriented solutions with a high integration between them. All the relevant mathematical operations for the development of the controllers will be done symbolically with the aid of *Mathematica*.

For the correct understanding of the following section, it is recommended to read chapter E from the appendices where the necessary theory behind controller design is carefully detailed.

3.3.1 Importing model into RecurDyn

The starting point is the *SolidWorks* model, which is stored in its own proprietary format. In order to achieve a flawless import, this model should be exported from the *SolidWorks* environment as a *Parasolid* binary file (**.x_b*).

A new model is created in *RecurDyn* and the *Parasolid* file imported. The import process provides only the geometry of the bodies that make the model up, without any reference to the connections or joints between them. All the joints must be defined again in this environment in order to obtain a self-aligned model, the solution for the joint selection obtained in the *SolidWorks* environment is valid in *RecurDyn*.

First, a dynamic free motion simulation will be performed where the model swings subject to the force of gravity. These results will be compared to the ones obtained in the same dynamic simulation made with *Mathematica*. Using the dynamic analysis from the previous chapter, a PID controller and a state feedback will also be implemented.

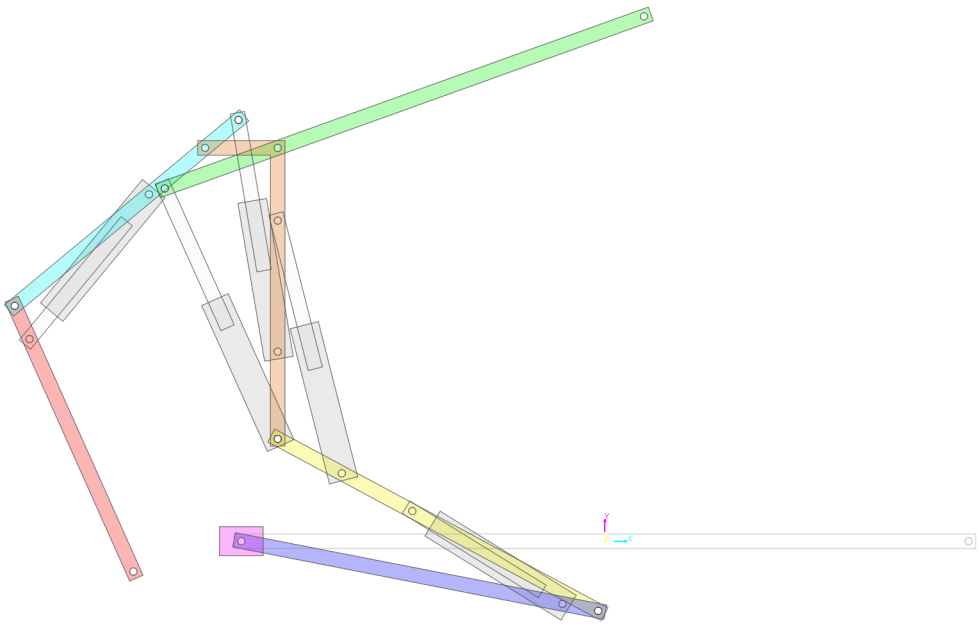


Figure 3.40: Model of the *Handle* robot in *RecurDyn* as imported from *SolidWorks*.

3.3.2 Obtaining the center of mass

In order to create a dynamic control which endows the model to maintain its balance, it is imperative to be able to know the position of the center of mass of the system at any moment during runtime.

The model is made up of a discrete number of bodies from which their inertial properties are known, therefore it is equivalent to a discrete mass system. In terms of inertial properties calculation, each body can be replaced by a particle with the same mass and inertia as the body and located at its center of mass. In a system comprised of a series of particles, what is known as a point mass system, the center of mass is calculated using the following equation:

$$\mathbf{r}_{\text{cm}} = \frac{\sum_i m_i \mathbf{r}_i}{\sum_i m_i} = \frac{1}{M} \sum_i m_i \mathbf{r}_i \tag{3.66}$$

Being M the total mass of the system, m_i the mass of the i-th particle, and $\mathbf{r}_i$ the position vector of the particle expressed in the desired reference system. In this case, $\mathbf{r}_{\text{cm}}$ is a three dimensional vector; the z-coordinate is known and equal to half the width of the bodies, but the x- and y-coordinates must be calculated during simulation time using the center of mass equation.

CoLink is the appropriate environment for the calculation of the center of mass, but it must receive the position of the center of mass of the different bodies as an input. This platform is independent from the *RecurDyn* modeling platform, therefore data flow must be defined using *Plant Input* for variables transferred from *CoLink* to *RecurDyn* and *Plant Output* for the other way around.

First, the position of the center of mass of the bodies must be taken as output from the *RecurDyn* environment into *CoLink*. This is done by defining outputs using the absolute position of the *CM marker* of the bodies in both x-coordinate,using the *DX()* function, and the y-coordinate, using the *DY()* function. The expressions of the *Plant Outputs* take the center of mass marker *BdR_handle_part_0X_2017.CM* for bodies ranging from 2 to 8 as the input.

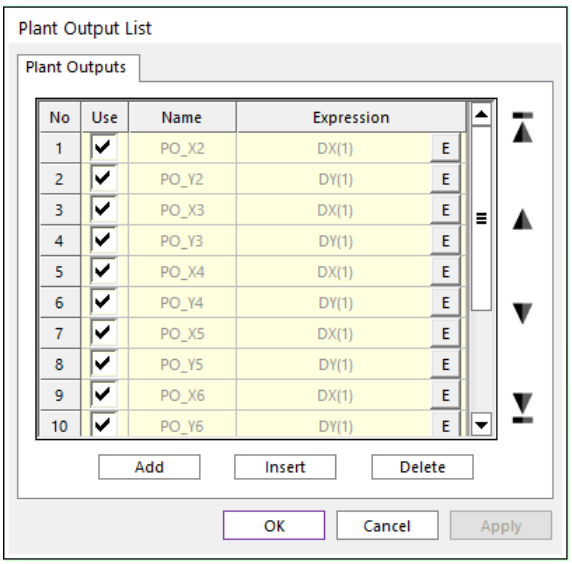


Figure 3.41: List of Plant Outputs for the model.

Even though body number 2 has no effect in the center of mass of the pendulum, it is important to obtain its center of mass position to be able to subtract its x coordinate from the x coordinate of the center of mass of the pendulum in order to obtain the relative position of the center of mass with respect to the revolute joint which connects it with the cart.

The next step is to define the block structure that reproduces the center of mass calculation within the *CoLink* environment. This is done using a *RecurDyn plant block* (which provides the plant inputs and outputs), muxers, demuxers, constant values, and multipliers. The coordinates of each of the bodies obtained from the modeling environment are multiplied by their masses and then added together and divided by the total mass of the system. The masses of the different bodies and the total mass of the system do not change during the simulation, therefore they are defined as constant values. This procedure provides the coordinates of the center of mass at any moment during the simulation as an output, and the x coordinate of the wheel body will also be taken as an output to be used in further developments.

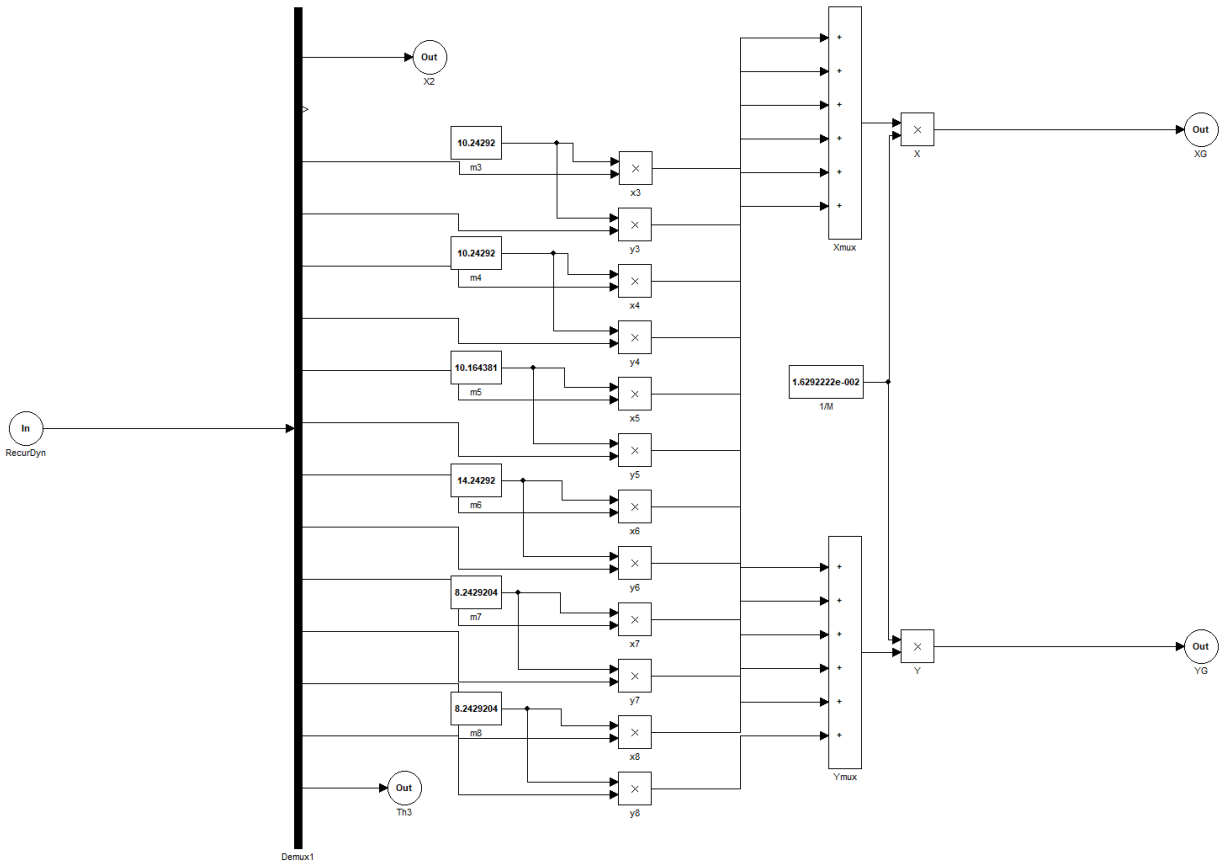


Figure 3.42: *CoLink* subroutine to calculate the center of mass of the model.

An auxiliary body, with the shape of a sphere, will serve as the visual representation of the center of mass. Using general constraint equations available in *RecurDyn*, the resulting coordinates of the center of mass $\{x_G, y_G\}$ obtained from *CoLink* will be forced to be the position of the sphere in every moment using a general constraint condition $\Phi = 0$.

3.3.3 Obtaining the equilibrium angle

From the kinematic model, the angle of the lower leg (body number 3) that makes the relative center of mass be $\mathbf{s_G} = \{x_G, y_G\} = \{0, \ell\}$ has been obtained:

$$\Theta_{3_{eq}} = 2\tan^{-1}(\frac{a}{b+\ell}) \tag{3.67}$$

This value is dependent on the length of the actuators and it must be obtained in real time during simulation. The symbolic expressions of the a and b coefficients have been obtained from the kinematic model, but their complexity is beyond what is reasonably suitable for calculation in a continuous process. They must be obtained in another way, using the canonical form:

$$\begin{aligned} \mathbf{s_G} &= \{x_G, y_G\}^T \\ x_G &= a\cos\Theta_3 - b\sin\Theta_3 \\ y_G &= b\cos\Theta_3 + a\sin\Theta_3 \end{aligned} \tag{3.68}$$

The numerical values of x_G and y_G are calculated during runtime therefore available at any moment, but the coefficients are unknown. The easiest way to obtain this coefficients is by reversing the planar rotation of angle Θ_3. The angle is also available, which means that the inverse rotation can be done to obtain the numerical values of the coefficients. Multiplying by the inverse rotation matrix, $\mathbf{A^{-1}}$, the coefficients can be obtained:

$$\mathbf{A^{-1}}\,\mathbf{s_G} = \begin{pmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} a\cos\Theta_3 - b\sin\Theta_3 \\ b\cos\Theta_3 + a\sin\Theta_3 \end{pmatrix} = \tag{3.69}$$

$$= \begin{pmatrix} (a\cos\Theta_3 - b\sin\Theta_3)\cos\Theta_3 + (b\cos\Theta_3 + a\sin\Theta_3)\sin\Theta_3 \\ -(a\cos\Theta_3 - b\sin\Theta_3)\sin\Theta_3 + (b\cos\Theta_3 + a\sin\Theta_3)\cos\Theta_3 \end{pmatrix} =$$

$$\begin{pmatrix} a\cos^2\Theta_3 - b\sin\Theta_3\cos\Theta_3 + b\sin\Theta_3\cos\Theta_3 + a\sin^2\Theta_3 \\ b\cos^2\Theta_3 - a\sin\Theta_3\cos\Theta_3 + a\sin\Theta_3\cos\Theta_3 + b\sin^2\Theta_3 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \tag{3.70}$$

With these coefficients now it is easy to obtain the length $\ell = \sqrt{a^2 + b^2}$ and calculate the equilibrium angle $\Theta_{3_{eq}}$ in real time in order to feed it as the reference value to the controller. This method of dynamically calculating the angle allows the reference to adapt to the possible changes in the geometry resulting from changing the lengths of the actuators.

Using the same procedure as with the center of mass, the equilibrium angle will be calculated in *CoLink*. The subroutine for the center of mass calculation will be modified in order to obtain the coefficients by multiplying the coordinates $\{x_G, y_G\}$ of the center of mass by the inverse rotation matrix. These coefficients will be combined to obtain the length from the slider to the center of mass ℓ, an finally the equilibrium angle $\Theta_{3_{eq}}$ will be obtained.

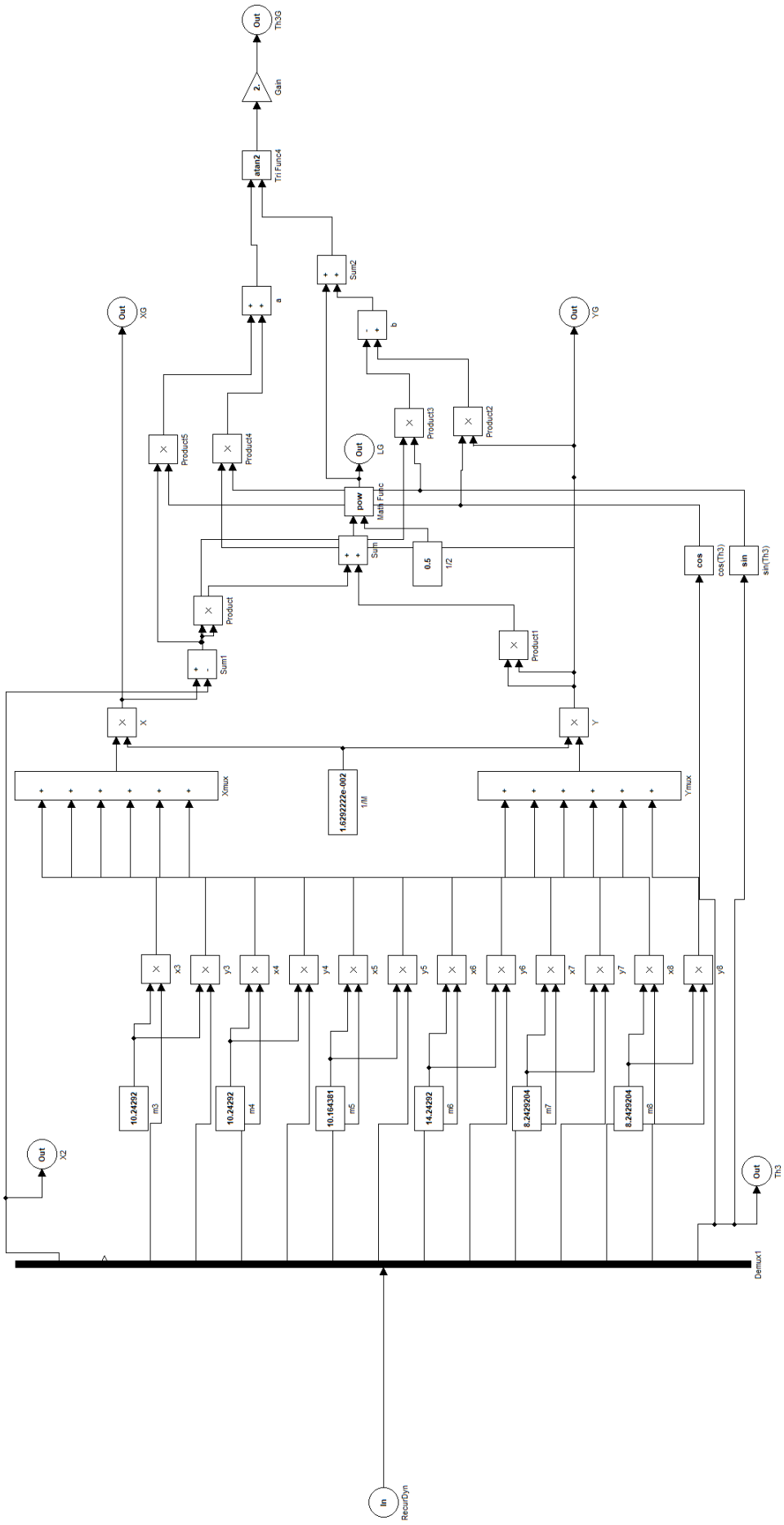


Figure 3.43: *CoLink* subroutine to calculate the center of mass of the model and the equilibrium angle.

3.3.4 PID controller design

The first approach is to design a double closed-loop control structure with two PID controllers, one for the angle of the pendulum and another one for the position of the cart.

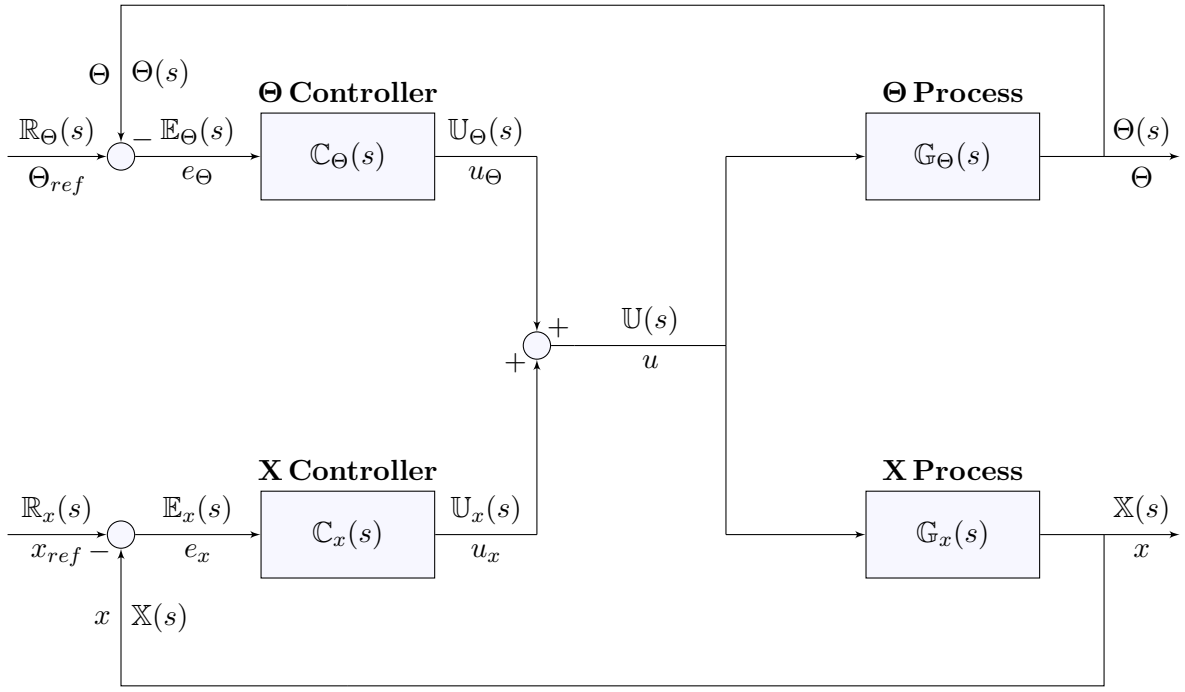


Figure 3.44: Double closed-loop controller design.

3.3.4.1 Pendulum angle controller

As a first approximation, only the PID controller of the pendulum angle is going to be designed. This will be done heuristically and by the pole placement method. In both cases the transfer function will be used. The manipulated or control variable will be a translational force applied to the center of mass of the cart and the assumption that no saturation occurs will be made.

The inverted pendulum model resulting from the dynamic analysis is unstable in open-loop, therefore only closed-loop heuristic methods can be applied. The chosen method will be the Ziegler-Nichols ultimate gain method. Because the transfer function is available, the parameters of the method will be obtained theoretically and they will be checked with the virtual model in *RecurDyn*. This will also serve as proof that the linearized model has been correctly obtained.

Considering the open-loop transfer function of the pendulum angle regarding the control action:

$$\mathbb{G}_\Theta(s) = \frac{\Theta(s)}{\mathbb{U}(s)} = \frac{\ell M}{(I_G(m+M) + \ell^2 m M)s^2 - g\ell M(m+M)} \tag{3.71}$$

The closed-loop transfer function of the system, considering no controller in the cart position section is easily obtained:

$$\mathbb{T}_\Theta(s) = \frac{\Theta(s)}{\mathbb{R}_\Theta(s)} = \frac{\mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s)}{1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s)} \tag{3.72}$$

Therefore, the characteristic equation of the system is:

$$1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) = 0 \tag{3.73}$$

In order to obtain the ultimate gain parameter for the Ziegler-Nichols method, the controller considered must only have the proportional term, $\mathbb{C}_\Theta(s) = K_p$. Developing the characteristic equation of the system and obtaining the roots of the characteristic polynomial:

$$1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) = 1 + K_p \frac{\ell M}{(I_G(m+M) + \ell^2 mM)s^2 - g\ell M(m+M)} = 0 \tag{3.74}$$

$$(I_G(m+M) + \ell^2 mM)s^2 - g\ell M(m+M) + K_p\,\ell M = 0$$

$$s_{1,2} = \pm\sqrt{\frac{g\ell M(m+M) - K_p\,\ell M}{(I_G(m+M) + \ell^2 mM)}}$$

This solution gives a set of complementary real poles that are positive and negative, therefore resulting in an unstable system, if the number inside the square root is positive. It is possible to obtain an stable oscillatory system with two conjugate imaginary poles if the numerator is negative, making the condition to obtain a stable system: $K_p > g(m+M)$. The ultimate gain parameter K_u is exactly the value of the proportional gain which satisfies the equation: $K_u = g(m+M)$.

To obtain the critical period parameter T_u, the characteristic equation must be solved with the substitution $s = i\,\omega$:

$$-(I_G(m+M) + \ell^2 mM)\omega^2 + \ell M\,(K_p - g(m+M)) = 0 \tag{3.75}$$

$$\omega = \pm\sqrt{\frac{\ell M\,(K_p - g(m+M))}{(I_G(m+M) + \ell^2 mM)}}$$

(3.76)

Considering that the relationship between frequency and period is $\omega = 2\pi f = 2\pi/T$, the period of the oscillation is:

$$T = \frac{2\pi}{\sqrt{\frac{\ell M\,(K_p - g(m+M))}{(I_G(m+M) + \ell^2 mM)}}} \tag{3.77}$$

Due to the nature of this system, with no damping terms, the critical frequency resulting when $K_p = K_u$ is $\omega = 0$ which results in a non-oscillatory system with both poles at $s = 0$. It might seem that the Ziegler-Nichols method is not suitable for this specific system.

Taking advantage that the symbolic expression of the transfer function of the model is available, it is possible to apply the pole position method. This method provides the parameters of the PID controller according to previously defined specifications.

Considering a PID controller with the transfer function:

$$\mathbb{C}_\Theta(s) = \frac{K_d \cdot s^2 + K_p \cdot s + K_i}{s} \tag{3.78}$$

Grouping the coefficients of the transfer function in order to simplify calculations:

$$\mathbb{G}_\Theta(s) = \frac{c_3}{d_1 s^2 + d_2} \tag{3.79}$$

Where:

$$d_1 = I_G(m+M) + \ell^2 mM \qquad d_2 = -g\ell M(m+M) \qquad c_3 = \ell M \tag{3.80}$$

The characteristic equation of the closed-loop system is:

$$1 + \mathbb{G}_\Theta(s)\mathbb{C}_\Theta(s) = 1 + \frac{K_d \cdot s^2 + K_p \cdot s + K_i}{s} \frac{c_3}{d_1 s^2 + d_2} = 0 \tag{3.81}$$

The characteristic polynomial, arranged in coefficient form:

$$\lambda(s) = s^3 + \frac{c_3\, K_d}{d_1}\, s^2 + \frac{d_2 + c_3\, K_p}{d_1}\, s + \frac{c_3\, K_i}{d_1} = 0 \tag{3.82}$$

The resulting system is of third degree, the crafted polynomial should have two dominant complex conjugate poles and a third real pole with a faster dynamics $p_1 = -k_1\sigma$.

$$\begin{gathered} \lambda'(s) = (s^2 + 2\xi\omega_n s + \omega_n^2)(s - p_1) = \\ s^3 + (-p_1 + 2\xi\omega_n)s^2 + (-2p_1\xi\omega_n + \omega_n^2)s + (-p_1\omega_n^2) = 0 \end{gathered} \tag{3.83}$$

By comparing the coefficients of the equations, the solution for the parameters of the PID in terms of the damping ξ and natural frequency ω_n can be obtained:

$$K_i = \frac{-d_1 p_1 \omega_n^2}{c_3} \qquad K_p = \frac{-d_2 - 2d_1 p_1 \xi\omega_n + d_1\omega_n^2}{c_3} \qquad K_d = \frac{-d_1(p_1 - 2\xi\omega_n)}{c_3} \tag{3.84}$$

3.3.4.2 Pendulum angle and cart position controller

Finally, the whole double closed-loop configuration with two PID controllers will be considered. The parameters of the two controllers can be obtained using the pole placement method.

The transfer functions to be considered are both for the pendulum angle and for the cart position:

$$\mathbb{G}_\Theta(s) = \frac{\Theta(s)}{\mathbb{U}(s)} = \frac{\ell M}{(I_G(m+M)+\ell^2 mM)s^2 - g\ell M(m+M)} = \frac{c_3}{d_1 s^2 + d_2} \tag{3.85}$$
$$\mathbb{G}_x(s) = \frac{\mathbb{X}(s)}{\mathbb{U}(s)} = \frac{s^2(I_G+\ell^2 M) - g\ell M}{(I_G(m+M)+\ell^2 mM)s^4 - g\ell M(m+M)s^2} = \frac{c_1 s^2 + c_2}{s^2(d_1 s^2 + d_2)}$$

Where:

$$d_1 = I_G(m+M)+\ell^2 mM \qquad d_2 = -g\ell M(m+M) \tag{3.86}$$
$$c_1 = I_G + \ell^2 M \qquad c_2 = -g\ell M \qquad c_3 = \ell M$$

The closed-loop characteristic equation is more complicated due to the double closed-loop, but it can also be calculated using the matrix equation, considering unity feedback: $\det(\mathbf{I}+\mathbf{C}(s)\mathbf{G}(s)) = 0$. Where the controller and process matrix are:

$$\mathbf{C}(s) = \begin{pmatrix} \mathbb{C}_\Theta(s) \\ \mathbb{C}_x(s) \end{pmatrix} \qquad \mathbf{G}(s) = \begin{pmatrix} \mathbb{G}_\Theta(s) \\ \mathbb{G}_x(s) \end{pmatrix}^T \tag{3.87}$$

In this case the matrix are of dimensions 2×1 because there are two output variables with feedback and only one control variable, the resulting is a 2×2 square matrix.

$$\det(\mathbf{I}+\mathbf{C}(s)\mathbf{G}(s)) = \left| \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} \mathbb{C}_\Theta(s) \\ \mathbb{C}_x(s) \end{pmatrix} \cdot \begin{pmatrix} \mathbb{G}_\Theta(s) & \mathbb{G}_x(s) \end{pmatrix} \right| = \tag{3.88}$$
$$\left| \begin{pmatrix} 1+\mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) & \mathbb{C}_\Theta(s)\mathbb{G}_x(s) \\ \mathbb{C}_x(s)\mathbb{G}_\Theta(s) & 1+\mathbb{C}_x(s)\mathbb{G}_x(s) \end{pmatrix} \right| = 1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) + \mathbb{C}_x(s)\mathbb{G}_x(s) = 0$$

If two PID controllers with following transfer functions are considered:

$$\mathbb{C}_\Theta(s) = \frac{K_{d_\Theta} \cdot s^2 + K_{p_\Theta} \cdot s + K_{i_\Theta}}{s} \qquad \mathbb{C}_x(s) = \frac{K_{d_x} \cdot s^2 + K_{p_x} \cdot s + K_{i_x}}{s} \tag{3.89}$$

Expanding the characteristic equation, the characteristic polynomial is obtained:

$$\lambda(s) = 1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) + \mathbb{C}_x(s)\mathbb{G}_x(s) = \tag{3.90}$$
$$s^5 + \frac{c_3 K_{d_\Theta} + c_1 K_{d_x}}{d_1} s^4 + \frac{d_2 + c_3 K_{p_\Theta} + c_1 K_{p_x}}{d_1} s^3 + \frac{c_2 K_{d_x} + c_3 K_{i_\Theta} + c_2 K_{i_x}}{d_1} s^2 + \frac{c_2 K_{p_x}}{d_1} s + \frac{c_2 K_{i_x}}{d_1} = 0$$

The resulting polynomial equation is of fifth degree, therefore the pole placement method must set five poles. Two of them will be the complex conjugate poles and the three remaining will be negative real poles with faster dynamics.

$$\lambda'(s) = (s^2 + 2\xi\omega_n s + \omega_n^2)(s - p_1)(s - p_2)(s - p_3) = \tag{3.91}$$
$$s^5 + (-p_1 - p_2 - p_3 + 2\xi\omega_n)s^4 + (p_1p_2 + p_1p_3 + p_2p_3 - 2p_1\xi\omega_n - 2p_2\xi\omega_n - 2p_3\xi\omega_n + \omega_n^2)s^3 +$$
$$+(-p_1p_2p_3 + 2p_1p_2\xi\omega_n + 2p_1p_3\xi\omega_n + 2p_2p_3\xi\omega_n - p_1\omega_n^2 - p_2\omega_n^2 - p_3\omega_n^2)s^2 +$$
$$+(-2p_1p_2p_3\xi\omega_n + p_1p_2\omega_n^2 + p_1p_3\omega_n^2 + p_2p_3\omega_n^2)s - p_1p_2p_3\omega_n^2 = 0$$

If the crafted and the characteristic polynomial equations are compared, there are five coefficients and six controller parameters. This means that one of the parameters can be arbitrarily selected and all the others will be calculated according to this value. If the K_{d_Θ} parameter is the one chosen to be set, the expressions for the other parameters are:

$$K_{p_\Theta} = \frac{-c_1 d_1 \omega_n(-2p_1p_2p_3\xi + p_2p_3\omega_n + p_1(p_2+p_3)\omega_n) + c_2(-d_2 + d_1(p_1p_2 + p_1p_3 + p_2p_3 - 2(p_1+p_2+p_3)\xi\omega_n + \omega_n^2))}{c_2c_3}$$
$$K_{i_\Theta} = \frac{c_1^2 d_1 p_1p_2p_3\omega_n^2 + c_1c_2d_1(-p_1p_2p_3 + 2(p_2p_3 + p_1(p_2+p_3))\xi\omega_n - (p_1+p_2+p_3)\omega_n^2) + c_2^2(c_3K_{d_\Theta} + d_1(p_1+p_2+p_3-2\xi\omega_n))}{c_1c_2c_3}$$
$$K_{p_x} = \frac{d_1\omega_n(-2p_1p_2p_3\xi + p_2p_3\omega_n + p_1(p_2+p_3)\omega_n)}{c_2}$$
$$K_{d_x} = -\frac{c_3K_{d_\Theta} + d_1(p_1 + p_2 + p_3 - 2\xi\omega_n)}{c_1}$$
$$K_{i_x} = -\frac{d_1p_1p_2p_3\omega_n^2}{c_2} \tag{3.92}$$

Using this approach, the control action consists of two terms, one dependent on the reference of the first controlled variable and the other dependent on the reference of the second controlled variable:

$$\mathbb{U}(s) = \mathbb{C}_\Theta(s)(\mathbb{R}_\Theta(s) - \Theta(s)) + \mathbb{C}_x(s)(\mathbb{R}_x(s) - \mathbb{X}(s)) \tag{3.93}$$

The closed-loop output of the system for the cart position and the pendulum angle can be obtained from analyzing the control loop. There are cross terms in the equations, but the dynamics of the system is determined by the poles of the function. The denominator is the same for both output functions, and the poles of that expression have been placed using this method.

$$\Theta(s) = \frac{\mathbb{G}_\Theta(s)(\mathbb{C}_\Theta(s)\mathbb{R}_\Theta(s) + \mathbb{C}_x(s)\mathbb{R}_x(s))}{1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) + \mathbb{C}_x(s)\mathbb{G}_x(s)} \tag{3.94}$$
$$\mathbb{X}(s) = \frac{\mathbb{G}_x(s)(\mathbb{C}_\Theta(s)\mathbb{R}_\Theta(s) + \mathbb{C}_x(s)\mathbb{R}_x(s))}{1 + \mathbb{C}_\Theta(s)\mathbb{G}_\Theta(s) + \mathbb{C}_x(s)\mathbb{G}_x(s)}$$

3.3.4.3 Implementation in CoLink

The *CoLink* environment will also serve as the place to implement the controller. For each time step within a controlled simulation, *RecurDyn* feeds *CoLink* the specified inputs (body positions), *CoLink* performs the programmed operation (force inputs for the drivers), *CoLink* outputs the results, and then *RecurDyn* applies the relevant command for the next time step.

In this case, the motion is driven during the first period of time, the moment of switching is defined as a parameter in *RecurDyn* named *TimeSwitch*, which makes the starting point of the control loop have a delay. This might cause problems with the derivative and integral term. Regarding the derivative term, because the equilibrium angle is dynamically calculated in each iteration, it is not necessary to implement an algorithm for anti-derivative kick. On the other side, actions must be taken in the integral term, the integration must be disabled until the control starts in order to avoid integrator windup due to the accumulation of error.

The derivative term of the error will be obtained using a *derivative block* that is equivalent to a $G(s) = s/(N\,s + 1)$ transfer function. The linearization parameter N is used to filter high frequency noise.

The PID controller will be implemented as a subroutine. The parameters of the proportional, integral and derivative terms of the PID controllers will be defined as parametric values in *RecurDyn*. This will make it easier to change their values without having to modify the subroutines in *CoLink*, being able to be modified directly from the *RecurDyn* environment. Scopes will be attached to each of the terms in order to see their contribution to the total control action.

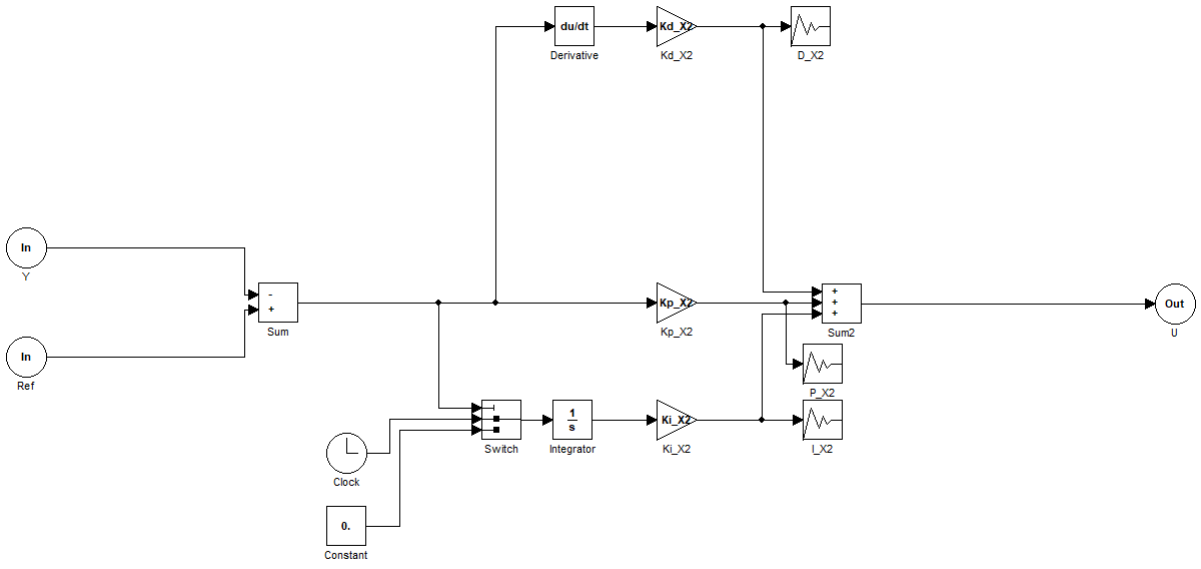


Figure 3.45: *CoLink* subroutine to implement a PID controller.

3.3.5 State feedback controller design

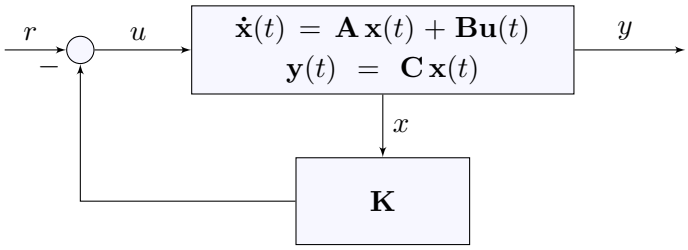


Figure 3.46: Full state feedback control system.

The alternative to the PID controller design is to create a state feedback control. Considering the linearized dynamic system of the inverted pendulum and due to the fact that all components of the state vector are available in the *RecurDyn* environment, a linear full state feedback control that stabilized the system can be designed.

The space state system of the open-loop linearized model is:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\,\mathbf{x}(t) \end{aligned} \tag{3.95}$$

The system has four state variables: horizontal position x and velocity $\dot{x}$ of the cart, and angle Θ and angular velocity $\dot{\Theta}$ of the pendulum. The control action is only the translational force F applied horizontally to the cart.

The feedback control action will be: $\mathbf{u}(t) = -\mathbf{K}\,\mathbf{x}(t)$. The reference value of the control action is zero. Because it is a linearized system, the proportional matrix $\mathbf{K}$ must be really applied to the deviation state vector $\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_o(t)$.

$$\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_o(t) = \begin{pmatrix} x - x_{eq} \\ \dot{x} - \dot{x}_{eq} \\ \Theta - \Theta_{eq} \\ \dot{\Theta} - \dot{\Theta}_{eq} \end{pmatrix} \qquad \mathbf{u}(t) = \begin{pmatrix} F \end{pmatrix} \qquad \mathbf{K} = \begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \end{pmatrix} \tag{3.96}$$

The resulting equations for the closed-loop linearized system with the feedback control are:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= (\mathbf{A} - \mathbf{B}\,\mathbf{K})\,\mathbf{x}(t) \\ \mathbf{y}(t) &= \mathbf{C}\,\mathbf{x}(t) \end{aligned} \tag{3.97}$$

Where:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{g\ell^2M^2}{I_G(m+M)+\ell^2mM} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g\ell M(m+M)}{I_G(m+M)+\ell^2mM} & 0 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 0 & 0 \\ \frac{I_G+\ell^2M}{I_G(m+M)+\ell^2mM} & 0 \\ 0 & 0 \\ \frac{\ell M}{I_G(m+M)+\ell^2mM} & 0 \end{pmatrix} \quad \mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The poles of the resulting closed-loop system are the eigenvalues of the coefficient matrix that goes with the state vector, that matrix in this case is: $\mathbf{A} - \mathbf{B}\,\mathbf{K}$.

$$\lambda(s) = \det(s\mathbf{I} - (\mathbf{A} - \mathbf{B}\,\mathbf{K})) = 0 \tag{3.98}$$

$$\lambda(s) = s^4 + \frac{I_G k_{12} + k_{14}\ell M + k_{12}\ell^2 M}{\ell^2 mM + I_G(m+M)} s^3 + \frac{I_G k_{11} + k_{13}\ell M + k_{11}\ell^2 M - g\ell M(m+M)}{\ell^2 mM + I_G(m+M)} s^2 + \\ + \frac{-g k_{12}\ell M}{\ell^2 mM + I_G(m+M)} s + \frac{-g k_{11}\ell M}{\ell^2 mM + I_G(m+M)} = 0$$

The next step is to determine the parameters of the $\mathbf{K}$ matrix considering the characteristic equation of the closed-loop system $\lambda(s)$.

It must be noted that the linear velocity and angular velocity equilibrium values are 0, as well as the value for the slider position. The value of the angle at equilibrium is calculated using the formula obtained previously, which is the same as the reference input to the PID controller.

$$x_{eq} = 0 \qquad \dot{x}_{eq} = 0 \tag{3.99}$$
$$\Theta_{eq} = 2\tan^{-1}\left(\frac{a}{b+\ell}\right) \qquad \dot{\Theta}_{eq} = 0$$

3.3.5.1 Pole placement method

The procedure to follow is the same as in the PID controller, a specially crafted polynomial with the desired poles will be constructed, in this case the characteristic polynomial is of fourth degree:

$$\lambda'(s) = (s^2 + 2\xi\omega_n s + \omega_n^2)(s - p_1)(s - p_2) = \tag{3.100}$$
$$s^4 + s^3(-p_1 - p_2 + 2\xi\omega_n) + s^2(p_1 p_2 - 2p_1\xi\omega_n - 2p_2\xi\omega_n + \omega_n^2) + \tag{3.101}$$
$$+ s(2p_1 p_2 \xi\omega_n - p_1\omega_n^2 - p_2\omega_n^2) + p_1 p_2 \omega_n^2 = 0$$

The values of the parameters can be obtained by comparing the coefficients polynomials such that $\lambda(s) = \lambda'(s)$, as done before with the PID control. In this case, because the system is controllable, applying Ackermann's formula the parameters are easily obtained:

$$k_{11} = \frac{-(\ell^2 mM + I_G(m+M))p_1 p_2 \omega_n^2)}{g\ell M} \tag{3.102}$$
$$k_{12} = \frac{(\ell^2 mM + I_G(m+M))\omega_n(-2p_1 p_2 \xi + (p_1 + p_2)\omega_n)}{g\ell M}$$
$$k_{13} = g(m+M) + \frac{(I_G + \ell^2 M)(\ell^2 mM + I_G(m+M))p_1 p_2 \omega_n^2}{g\ell^2 M^2} + \frac{(\ell^2 mM + I_G(m+M))(p_1 p_2 - 2(p_1+p_2)\xi\omega_n + \omega_n^2)}{\ell M}$$
$$k_{14} = \frac{-1}{g\ell^2 M^2}(\ell^2 mM + I_G(m+M))((I_G + \ell^2 M)\omega_n(-2p_1 p_2 \xi + (p_1+p_2)\omega_n) + g\ell M(p_1 + p_2 - 2\xi\omega_n))$$

This is the same formula *Mathematica* uses to calculate the feedback values when using the **StateFeedbackGains** function.

3.3.5.2 Linear-Quadratic Regulator

Considering quadratic cost function subject to the system dynamics:

$$J_{LQR} = \int_0^\infty \left(\mathbf{x}(t)^T \, \mathbf{Q} \, \mathbf{x}(t) + \mathbf{u}(t)^T \, \mathbf{R} \, \mathbf{u}(t) \right) dt \tag{3.103}$$

The solution to this problem is given by the time-invariant linear state feedback control law:

$$\mathbf{u}(t) = -\mathbf{K_{LQR}} \, \mathbf{x}(t) \tag{3.104}$$

$$\mathbf{K_{LQR}} = \begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \end{pmatrix} \tag{3.105}$$

The values for the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$, will be tuned using *Bryson's rules*.

$$\mathbf{Q} = \begin{pmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{pmatrix} \qquad \mathbf{R} = \rho \, r_1 \tag{3.106}$$

$$q_{ii} = \frac{\alpha_i^2}{(x_i)_{max}} \qquad r_1 = \frac{1}{(u_1)_{max}}$$

The α, β and ρ parameters and the maximum admissible values $(x_i)_{max}$ and $(u_1)_{max}$ will be tuned in order to obtain a satisfactory controller.

In order to obtain the optimum results for the dynamics of the system, the method for setting the values of the parameters is iterative. Different values for the parameter will be tried and the $\mathbf{K_{LQR}}$ matrix will be calculated using *Mathematica* function **LQRegulatorGains**. This results will be tested on the system and tuned to obtain the desired response.

3.3.5.3 Implementation in CoLink

In the *CoLink* environment a space feedback controller can easily be implemented, considering that the state variables are accessible in any moment of the simulation.

As well as in the previous case, the motion is driven during the first period of time, the moment of switching is defined as a parameter in *RecurDyn* named *TimeSwitch*, which makes the starting point of the control loop have a delay. This is not a problem for a state feedback controller. Both the pole placement method and the LQR controller share the same control scheme, it is the values of the proportional feedback gain what changes.

The derivative values of the state variables will be obtained using a *derivative block* that is equivalent to a $G(s) = s/(N\,s + 1)$ transfer function. The linearization parameter N is used to filter high frequency noise.

The parameters of the proportional terms of the state feedback controller will be defined as parametric values in *RecurDyn*. This will make it easier to change their values without having to modify the subroutines in *CoLink*, being able to be modified directly from the *RecurDyn* environment. Scopes will be attached to each of the terms in order to see their contribution to the total control action.

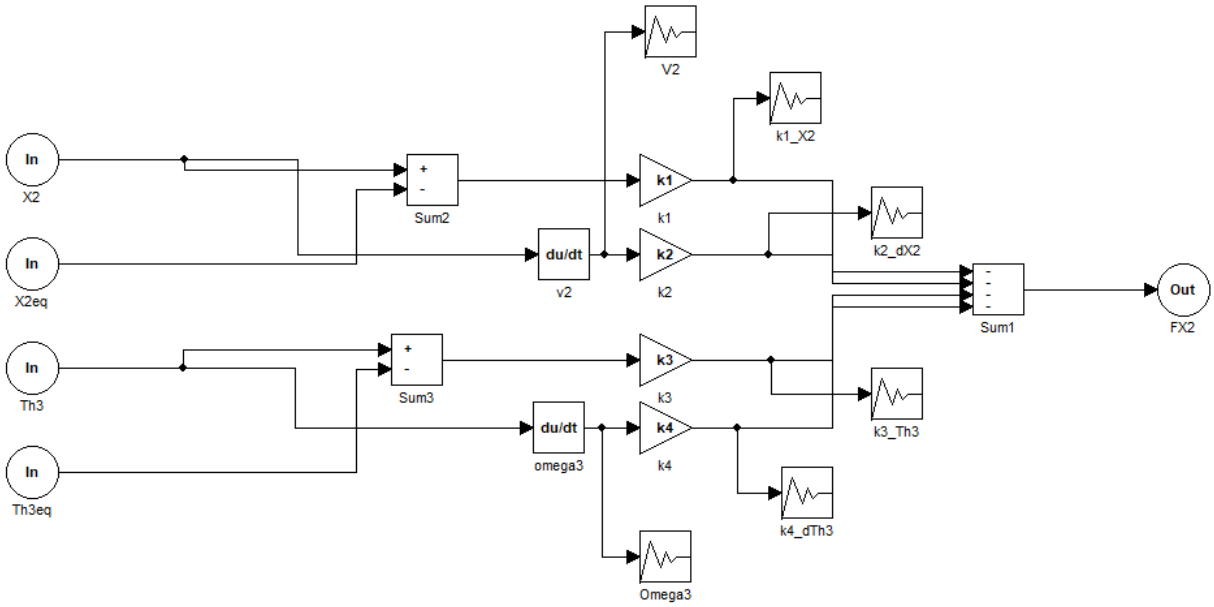


Figure 3.47: *CoLink* subroutine to implement a state feedback controller.

3.3.6 Dynamic simulation

In this case, the dynamic simulation will be of the closed-loop configuration using the controllers designed to maintain the balance of the robot. The robot will start in its initial configuration and it will be driven to the initial starting dynamic position using the following motions:

$$\begin{aligned} X_2 &= X_{2_i} + \Delta X_2 \cdot T = -0.5 + 0.5 \cdot \text{STEP5}[T, 2, 0, 4, 1]\ (m) \\ \Theta_3 &= \Theta_{3_i} + \Delta\Theta_3 \cdot T = -11.03 + 45 \cdot \text{STEP5}[T, 0, 0, 2, 1]\ (^\circ) \\ La_1 &= La_{1_i} + \Delta La_1 \cdot T = 0.242722 + 0.04 \cdot \text{STEP5}[T, 0, 0, 2, 1]\ (m) \\ La_2 &= La_{2_i} + \Delta La_2 \cdot T = 0.358221 + 0.01 \cdot \text{STEP5}[T, 3, 0, 5, 1]\ (m) \\ La_3 &= La_{3_i} + \Delta La_3 \cdot T = 0.377705 - 0.12 \cdot \text{STEP5}[T, 0, 0, 2, 1]\ (m) \\ La_4 &= La_{4_i} + \Delta La_4 \cdot T = 0.322810 - 0.03 \cdot \text{STEP5}[T, 0, 0, 2, 1]\ (m) \\ La_5 &= La_{5_i} + \Delta La_5 \cdot T = 0.258034 + 0.01 \cdot \text{STEP5}[T, 0, 0, 2, 1]\ (m) \end{aligned} \tag{3.107}$$

The *TimeSwitch* parameter will determine when the controller action starts, and it will be set to $TimeSwitch = 5\ s$. The configuration has been chosen in order for the center of mass to be away from the vertical line over the slider, causing the control loop to correct this situation.

After the first initial response has settled, two actuator lengths will be forced to change.

$$\begin{aligned} La_1 &= La_{1_d} + \Delta La_1 \cdot T = 0.282722 - 0.02 \cdot \text{STEP5}[T, 10, 0, 15, 1]\ (m) \\ La_5 &= La_{5_d} + \Delta La_5 \cdot T = 0.268034 + 0.01 \cdot \text{STEP5}[T, 20, 0, 25, 1]\ (m) \end{aligned} \tag{3.108}$$

This will cause the ℓ and I_G parameters to change, as well as the equilibrium point Θ_{eq}, causing the model to be altered. This changes are meant to cause a significant change in the values of the linearized model causing the closed-loop poles to be altered from the original design as well as a change in the equilibrium point. The dynamic response to these changes of the system implemented will be observed in order to test the robustness of the control.

Changing from the driven motion in the interval of time $0 < T < TimeSwitch$ to the free motion determined by the dynamic of the system and the controllers for $T > TimeSwitch$ is not a trivial task in *RecurDyn*.

3.3.6.1 Driven motion to free motion

In *RecurDyn*, motions, defined as expressions, can be assigned to a joint in order to make this joint driven. If no motion is defined, the joint is free to move under the dynamics of the system, but if a motion is defined this joint will be driven. A value of zero in the expression of the motion will make the joint to remain fixed, it will not make it have free motion. There is no way of changing the joint from driven and free during simulation.

The only way of addressing this issue is to create a dummy body and a force or torque, depending if it is a translational or a rotational motion what should be defined. A dummy body with a negligible weight is created at the location of the body that must perform the motion. An axial or rotational axial force is created between the dummy body and the target body. The same joint as the one the target body has is defined between the dummy body and the base body.

The desired motion is applied to the dummy body's joint, which makes the target body to be connected to the base body through the dummy force or torque. The expression of this virtual action is defined using an expression and it is disconnected with a conditional function, e.g. the *IF()* function, when a certain condition is satisfied. Once the force keeping the dummy body and the target body together is set to 0, the connection between the two bodies is broken completely and the target body is free to move.

The expressions for the virtual forces and torques are:

$$\begin{aligned} F &= \text{IF}(\text{TIME} - TimeSwitch : K_F \cdot (DX(Marker_1, Marker_2)), 0, 0) \\ \tau &= \text{IF}(\text{TIME} - TimeSwitch : K_\tau \cdot (PSI(Marker_1, Marker_2)), 0, 0) \end{aligned} \tag{3.109}$$

Where K_F and K_τ should be very big constants to minimize the deviation from the desired motion, the sign of this constants depend on the direction of the motion. The distance function can be in any direction and magnitude: *DX()*, *DY()*, *DZ()*, *DM()*; and the angular function also: *PSI()*, *THETA()*, *PHI()*, *YAW()*, *PITCH()*, *ROLL()*. The first marker is the position of the target body to be driven while the second marker is the position of the dummy body which drives. The force is disconnected when the *TimeSwitch* threshold is reached.

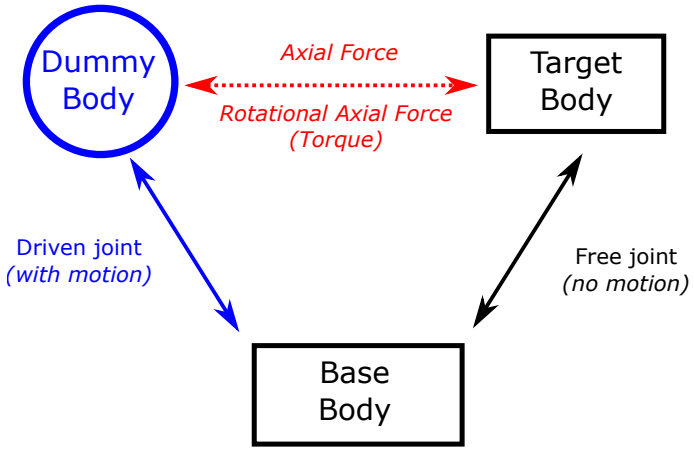


Figure 3.48: Diagram of the driven to free motion technique implementation.

This technique will allow a body to perform a driven motion and change to a free motion when certain conditions apply, e.g. after a certain time or when a sensor has activated, during simulation. The application of this method is not limited to *RecurDyn*, it can be applied in any other multibody dynamics software and it will be applied in *SystemModeler* to obtain the same kind of driven to free motion results.

3.4 *Modelica*-based model in *SystemModeler*

Modelica is an object-oriented and equation based language used to model and simulate the behavior of physical systems in many engineering fields, e.g. mechanical, electrical, hydraulic, control or process engineering. There is a large set of libraries with many components and examples available, being the most important the *Modelica Standard Library (MSL)* with components from many domains. The language is non-proprietary and it is maintained by the *Modelica Association*, a non-profit organization with members all around the world.

There are different software environments implemented that can be used to develop systems in the *Modelica* language: some of them are free and open source, such as *OpenModelica*; and others are proprietary, e.g. *Wolfram SystemModeler*, *MapleSim* and *Dymola*.

The *Modelica* language is based on what is known as *models*, which can be standalone or be part of another model. If the model is an implementation of a very specific feature or element, it will be considered as a *component*. The models and components are usually organized into *packages*, that might contain many models and even other packages. At the lowest level of the hierarchy, *functions* and *blocks* are the simplest implementations that contain no subcomponents. Inside a model, other models, blocks, functions and variables can be instantiated, as well as *parameters* that define values that might me subject to change depending on the application. In models with components, the different elements can be connected with each other using *connectors*. In order to be able to simulate a model, it must be *balanced*, which means that the complete set of model components an its connections leads to a set of differential, algebraic and discrete equations where the number of variables is equal to the number of equations to solve.

The *Modelica Standard Library* has a built-in package in order to model 3-dimensional mechanical systems, the *Modelica.Mechanics.MultiBody* library. This library contains all kinds of components: forces, bodies, sensors, joints, etc. Probably the most interesting feature of this library is the built-in animation properties of all components, such as joints, forces, bodies, sensors, allowing an easy visual check of the constructed model. Animation of every component can be switched off via a parameter, and the animation of a complete system can be switched off via one parameter in the world model. If animation is switched off, all equations related to animation are removed from the generated code. This is especially important for real-time simulation. The animation features are defined through the *Visualizer* package containing components to visualize 3-dimensional shapes: boxes, spheres, cylinders, cones, tori, pipes, beams, gears, springs, wheels, axes, arrows, planes, and surfaces.

In order to check the results obtained from the other software, the *Modelica* language was used to create a model in *SystemModeler*. The results from the kinematic solution from *Mathematica* were used to symbolically define masses and moments of inertia for the different bodies and for the coordinates of the center of mass.

For the correct understanding of the following section, it is recommended to read chapter F from the appendices where some concepts that must be considered during the *Modelica* model definition are carefully detailed.

3.4.1 Custom components

During the development of the equivalent model in *SystemModeler*, there were some components that could not be found in the libraries and therefore they had to be customly defined. Everyone who has some experience developing custom engineering solutions, regardless of the field, knows that libraries (proprietary or open source) can come in quite handy, but there is always something that is not yet implemented in them and it must be developed *ex profeso* for the desired application. In this case some components had to be modified in order to obtain the desired behavior and others had to be created from scratch.

The components customly created or modified are:

- *Step5* component
- *PrismaticMotion* component
- *RevoluteMotion* component
- *VectorTranslation* component

3.4.1.1 Step5 component

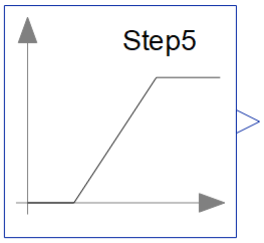


Figure 3.49: Modelica Step5 component.

The first component that must be created is the *Step5* component, a quintic polynomial interpolation of the step function as defined in chapter C from the appendices. The *MSL* contains a step component, but it is an exact implementation of the unit step function with a discontinuity at the moment of the step. The *Step5* component uses *Equation C.3* to create a sigmoid transition from the h_0 value at time t_0 to the h_1 value at time t_1.

Name	Units	Description
h0	*Real*	Initial value
h1	*Real*	Final value
t0	s	Starting time
t1	s	Ending time

Table 3.17: Parameters of the *Step5* component.

3.4.1.2 PrismaticMotion component

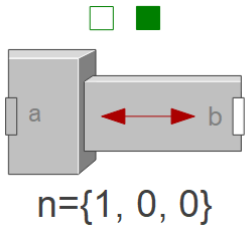


Figure 3.50: *Modelica PrismaticMotion* component.

The *MultiBody* library from the *MSL* contains a prismatic joint component to create a connection between two bodies allowing a relative linear motion between them. Just as it happened in *RecurDyn*, this component can be allowed to move freely or forced to be driven using a mechanical flange by enabling the *useAxisFlange* parameter, but it not possible to switch between these states. For the development of the model it is necessary to tweak this component further in order to allow a switch from driven to free motion at a certain time.

A switchable prismatic joint, the *PrismaticMotion* component, will be created in order to obtain this configurable behavior. First, a virtual dummy body is created and it starts in the same position as the second body, with a fixed relative distance from the first body. The switching between free and driven motion is implemented by enabling or disabling a force that attracts the virtual dummy body and the second body together being proportional to the relative distance between them. The proportional constant of the force is very large, causing the bodies to be together while it is enabled. The location of the virtual dummy body relative to the first body is driven using the mechanical flange and therefore the second body follows this same motion. At a certain time, the force joining the bodies together is disabled causing the virtual dummy body to stay at its relative location from the first body and the second body to move freely.

Name	**Type**	**Units**	**Description**
Frame_a	*Connector*	-	Connection to first body
Frame_b	*Connector*	-	Connection to second body
Flange_a	*Connector*	-	Mechanical flange connection
Flange_b	*Connector*	-	Mechanical flange support
X	*Parameter*	m	Initial position of body
timeSwitch	*Parameter*	s	Time to switch from driven to free motion
n	*Parameter*	*Real[3]*	Axis of traslation resolved in frame_a
r	*Parameter*	*Real[3]*	Vector from frame_a to frame_b

Table 3.18: Parameters and connectors of the *PrismaticMotion* component.

3.4.1.3 RevoluteMotion component

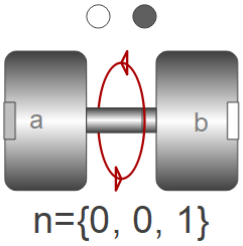


Figure 3.51: *Modelica RevoluteMotion* component.

The *MultiBody* library from the *MSL* contains a revolute joint component to create a connection between two bodies allowing a relative angular motion between them. Just as it happened in *RecurDyn*, this component can be allowed to move freely or forced to be driven using a mechanical flange by enabling the *useAxisFlange* parameter, but it not possible to switch between these states. For the development of the model it is necessary to tweak this component further in order to allow a switch from driven to free motion at a certain time.

A switchable revolute joint, the *RevoluteMotion* component, will be created in order to obtain this configurable behavior. First, a virtual dummy body is created and it starts in the same position as the second body, with a fixed relative angle from the first body. The switching between free and driven motion is implemented by enabling or disabling a torque that attracts the virtual dummy body and the second body together being proportional to the relative angle between them. The proportional constant of the torque is very large, causing the bodies to have the same orientation while it is enabled. The orientation of the virtual dummy body relative to the first body is driven using the mechanical flange and therefore the second body follows this same motion. At a certain time, the torque joining the bodies together is disabled causing the virtual dummy body to stay at its relative orientation from the first body and the second body to move freely.

Name	**Type**	**Units**	**Description**
Frame_a	*Connector*		Connection to first body
Frame_b	*Connector*		Connection to second body
Flange_a	*Connector*		Mechanical flange connection
Flange_b	*Connector*		Mechanical flange support
Th	*Parameter*	°	Initial angle of body
timeSwitch	*Parameter*	s	Time to switch from driven to free motion
n	*Parameter*	*Real[3]*	Axis of rotation resolved in frame_a
r	*Parameter*	*Real[3]*	Vector from frame_a to frame_b

Table 3.19: Parameters and connectors of the *RevoluteMotion* component.

3.4.1.4 VectorTranslation component

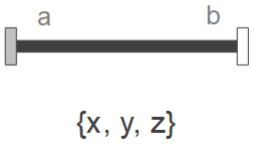


Figure 3.52: *Modelica VectorTranslation* component.

The symbolic expression of the center of mass of the model was obtained using *Mathematica* and it would be interesting to show a graphical representation of its spatial location while the motion of the model occurs. A blue sphere will determine the visual representation of the center of mass, but a new component that allows it to be positioned according to the values provided by the symbolic expression must be created.

The resulting *VectorTranslation* component is a slightly modification of the *FixedTranslation* component included in the *MSL*. It has been modified to allow the relative position of the body connected to the *flange_b* connector to be determined by a 3-dimensional vector *{x,y,z}*. In this case the first body is the ground and the second body will be the spherical center of mass representation. The modification was necessary due to the limitation of the *FixedTranslation* component that did not allow the original vector that defined the relative position between *frame_b* and *frame_a* to be modified at runtime.

Name	**Type**	**Units**	**Description**
Frame_a	*Connector*		Connection to first body
Frame_b	*Connector*		Connection to second body
xyz	*Parameter*	*Real[3]*	Vector from frame_a to frame_b

Table 3.20: Parameters and connectors of the *VectorTranslation* component.

3.4.2 Model definition

Even though it may seem simple, this model is quite complex due to high number of closed loops derived from the usage of variable-sided triangles in the linear actuator configuration. It is very important to choose state variables correctly and to provide the correct initial values for these variables.

After several attempts, by a trial and error methodology, the initialization problem of the model was determined correctly, successfully obtaining the desired initial position. It was necessary to define self-aligning variable-sided triangle configurations using cylindrical and spherical joints in substitution for two revolute joints. This is exactly the same procedure described when defining the model in *SolidWorks*. The selected state variables were angular positions and velocities of all revolute joints and the linear position and velocity of the translational (prismatic) joint with the ground. The translational joint of the slider driver and the revolute joint of the angular driver between body 2 and 3 had a fixed initial value for their initial values of positions and velocities. All the remaining revolute joints had fixed initial angular velocity values set to 0.

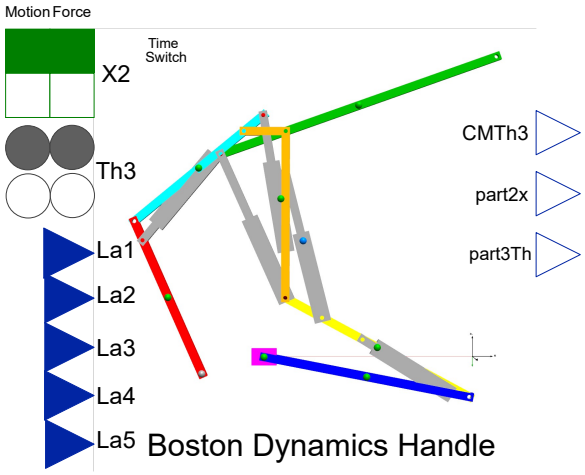


Figure 3.53: *Modelica Handle* component.

The resulting model has been implemented as a standalone component in *SystemModeler* with several inputs and outputs. The inputs of the model are separated in two groups: parameters and actual connected inputs. The parameters of the model include geometrical and inertial properties of all the models, including their initial positions. The connected inputs are the degrees of freedom of the model. The actuator lengths, which make up five of the seven degrees of freedom, are input variables that are controlled from outside the model. The remaining two degrees of freedom are part of the dynamic control group. The positions of the slider and the angle of body number 3 can be driven directly, using the connectors under the *motion* label; or dynamically controlled using a force or torque, using the connectors under the *force* label. The *TimeSwitch* parameter determines in what moment of time the control changes from driven to dynamically controlled. The outputs of the model are the position of the slider X_2, the angle of body 3 Θ_3 and the equilibrium angle for the selected configuration of actuator positions $\Theta_{3_{eq}}$.

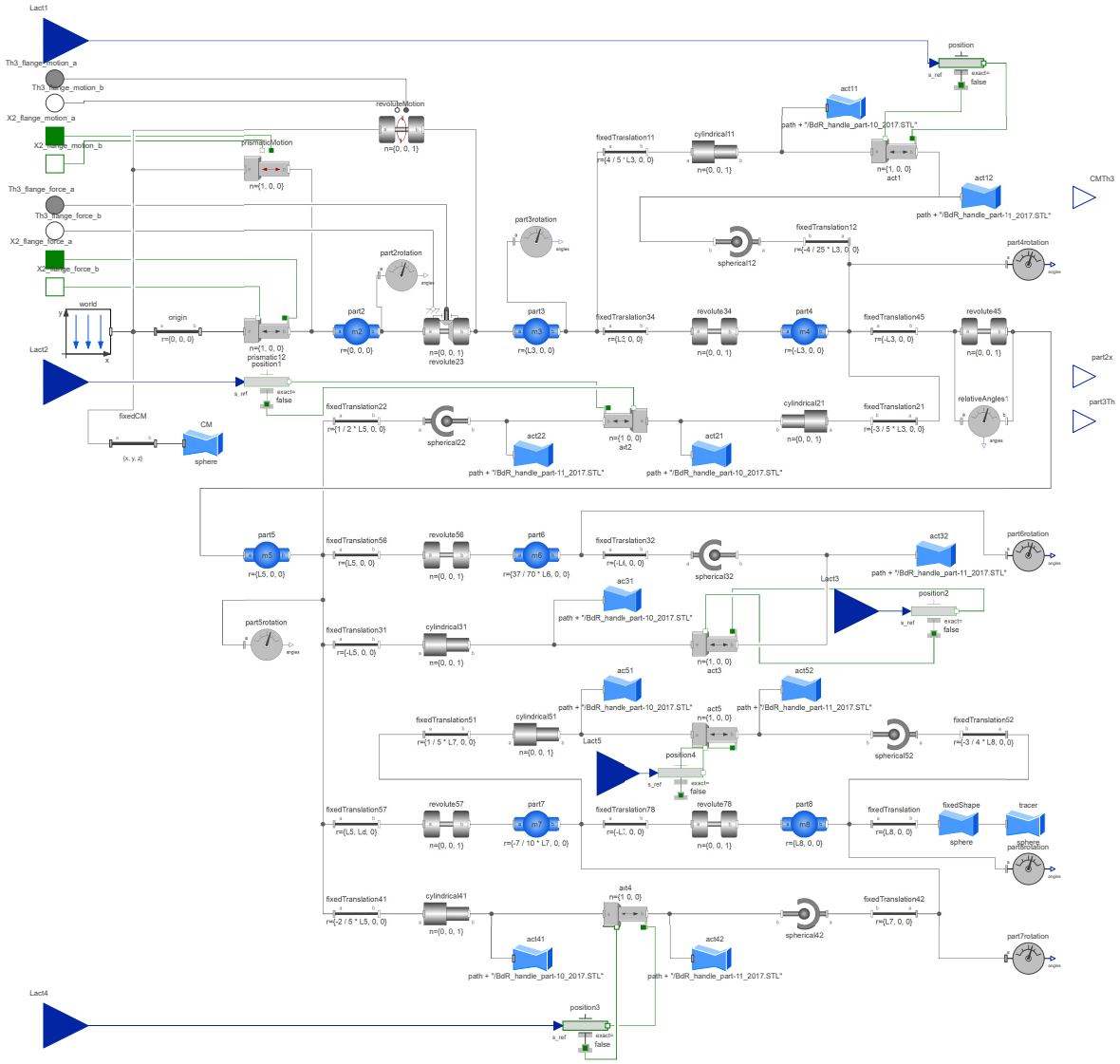


Figure 3.54: Internal structure of the *Handle* component.

The visual representation of the bodies will be achieved by importing the *SolidWorks* files in the *STL* format, which is one of the supported by the *Visualizer* package in the *MSL*. These surface geometry files will be attached to the body definitions using the *FixedShape* component, allowing the model to have a graphical representation that shows the behavior of the system during the simulation.

3.4.3 PID controller implementation

Following the same procedure as in *RecurDyn*, a closed-loop must be implemented using a PID controller and the feedback from the model's X_2 and Θ_3 variables and their set-points. In order to avoid integrator windup, the PID controller has the integral term deactivated until the dynamic control starts with *TimeSwitch*. The output is the force applied on the slider F. The parameters of the component are the proportional, integral and derivative gains for the X_2 and Θ_3 variables.

The derivative term will be obtained using a *derivative block* that is equivalent to a $G(s) = s/(N\,s + 1)$ transfer function. The linearization parameter N is used to filter high frequency noise.

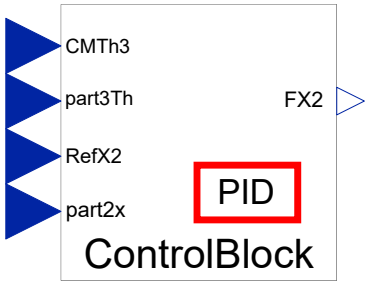


Figure 3.55: *Modelica PIDControl* component.

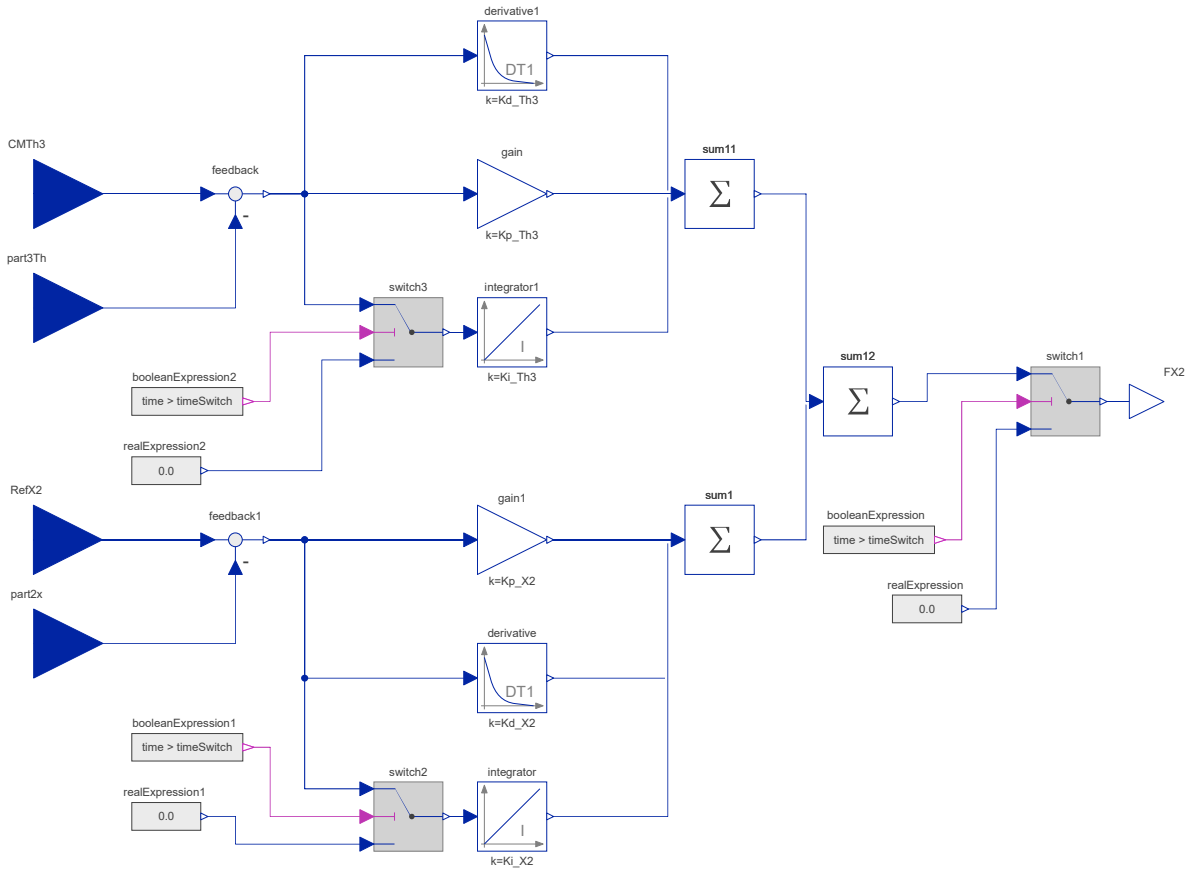


Figure 3.56: Internal structure of the *PIDControl* component.

3.4.4 State feedback controller implementation

Following the same procedure as in *RecurDyn*, a closed-loop must be implemented using state feedback from the model's X_2 and Θ_3 variables and their set-points. The output is the force applied on the slider F. The parameters of the component are the feedback gains for the state variables.

The derivative values of the state variables will be obtained using a *derivative block* that is equivalent to a $G(s) = s/(N\,s + 1)$ transfer function. The linearization parameter N is used to filter high frequency noise.

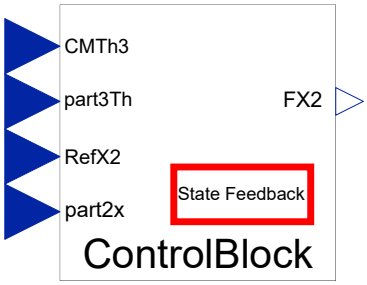


Figure 3.57: *Modelica StateFeedbackControl* component.

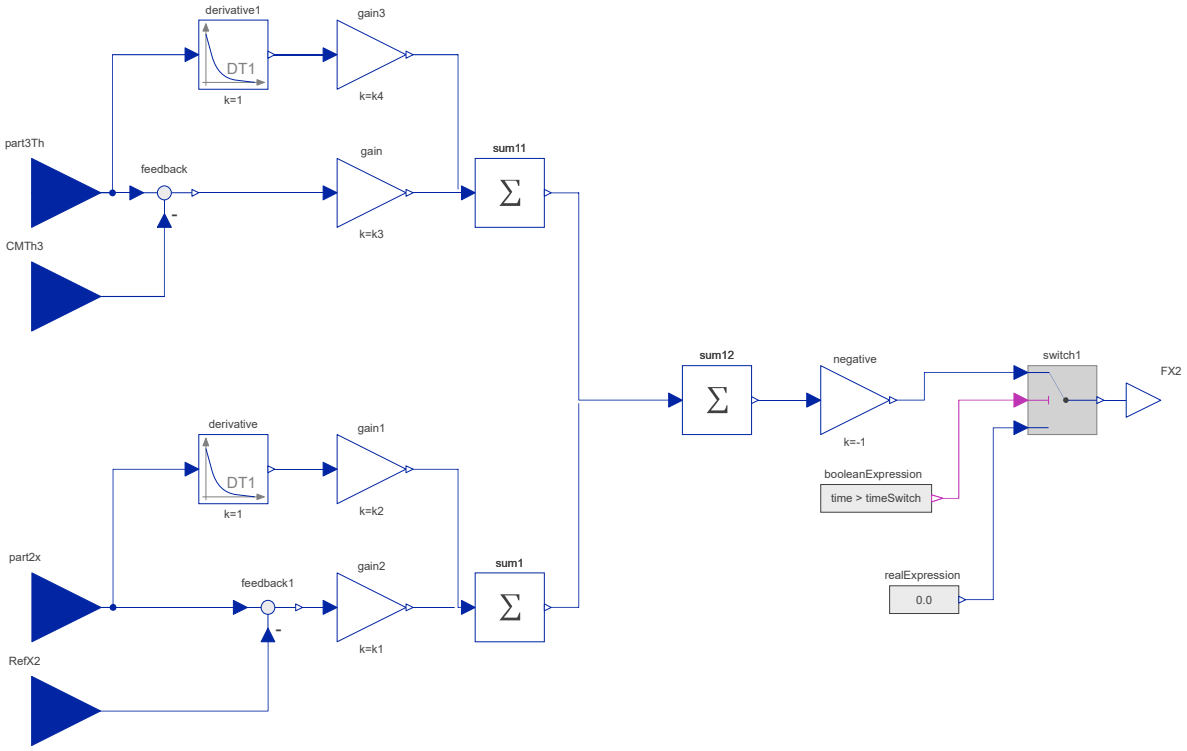


Figure 3.58: Internal structure of the *StateFeedbackControl* component.

Chapter 4

Presentation and Analysis of Results

The results derived from the methods described in the previous section will be presented and analyzed in this chapter. The output of the computational tools used and the different approaches considered will be compared to highlight the differences and similarities of the data obtained.

4.1 Introduction

The simplified virtual models obtained in the different software environments will be presented. The first model is defined in the *SolidWorks* environment using the *COSMOSMotion* add-in, including a graphical representation of the tracer point position when driving the different degrees of freedom in what is known as a kinematic simulation. The symbolic kinematic equations obtained in *Mathematica* for the position of the bodies will be presented with the numerical values of the parameters substituted. The results from the kinematic motion simulation of *SolidWorks COSMOSMotion* add-in will be compared to the data obtained when applying the same input to the drivers in the kinematic equations obtained from *Mathematica*. The results of the position, velocity and acceleration of the tracer point will be compared side by side.

The dynamic model developed in *Mathematica* together with the *MechanicalSystems* add-in will allow a dynamic simulation to be done. The first simulation will show how the model behaves when the position of the slider and the first revolute joint are left free to move under the action of gravity. The scenario considered will have no damping elements nor numerical damping. A model in the multibody dynamics software *RecurDyn* will be developed, and the same simulation will be performed, comparing the results for the trajectories of the tracer point of both softwares.

Classical PID controllers and state feedback controllers to stabilize the model have been developed for the *RecurDyn* model using the control capabilities of the *CoLink* add-on. Dynamic calculation of the center of mass allows the stabilization of the model to maintain its center of mass along the vertical axis of the slider. An equivalent model has been defined in *SystemModeler*, where implementation of controllers is also available. The two types of controllers, PID and state feedback, will also be implemented in this software, and the results of both of them compared. This will conclude the analysis of the *Handle* robot prototype.

The model has undergone many changes since it started with the initial hand-made kinematic diagram of Figure 3.1. It seemed necessary to create a new kinematic diagram with the final configuration of the model. In this case the graphics obtained in the *Mathematica* model were used along with the image editing software *Inkscape* to create a vector graphic version of the kinematic diagram.

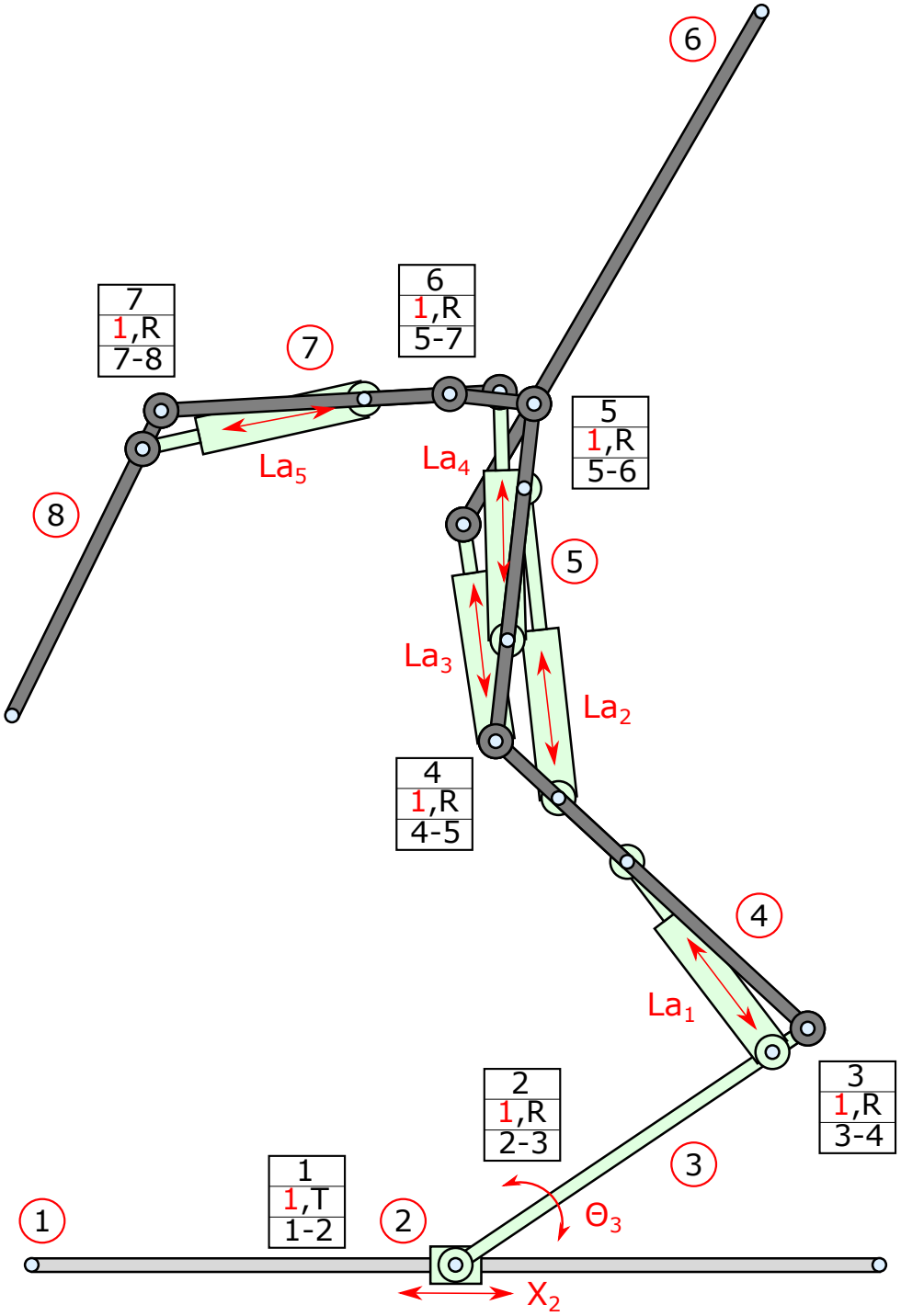


Figure 4.1: Kinematic diagram of the final version of the model.

4.2 *SolidWorks* and *COSMOSMotion*

The *SolidWorks* model is the starting point for all of the subsequent models. This model will be exported to *RecurDyn* and *SystemModeler* through an intermediary neutral format. The model consists of eight main bodies and ten more bodies that make up the visual representation of the linear actuators.

Once the different bodies are geometrically defined, the *SolidWorks* environment is used to assembly the model from these building blocks. The objective is to obtain a simplified virtual equivalent model of the theoretical model. The joint configuration of the *SolidWorks* model will be analyzed in order to obtain a self-aligned model with the aid of the Grübler mobility formula from the *COSMOSMotion* add-in.

There have been further simplifications from the theoretical model to the 2-dimensional virtual model, some of them resulting from necessary simplifications for the mathematical model's complexity to stay within reasonable limits.

A kinematic simulation using the *COSMOSMotion* add-in will be performed obtaining position, velocity and acceleration of the tracer point. These results will later be compared to the kinematic expressions obtained from *Mathematica*.

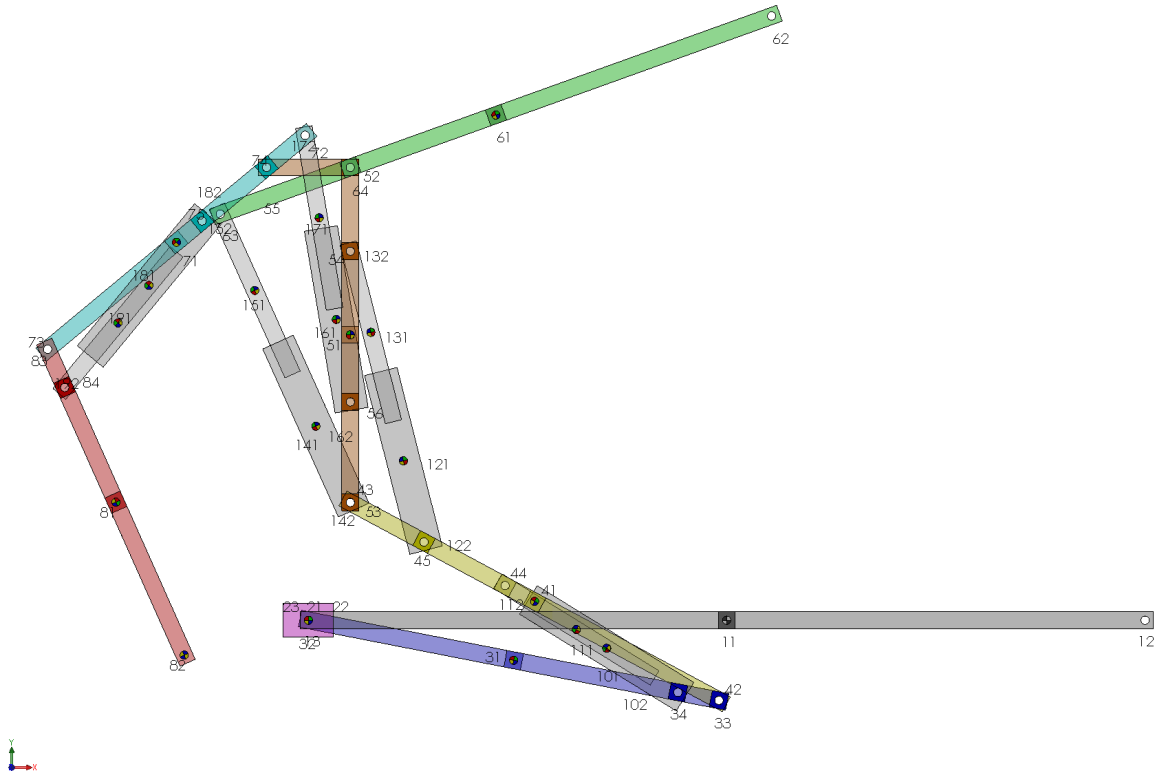


Figure 4.2: *SolidWorks* model of the robot with all the bodies and their relevant points.

4.2.1 Self-aligned model

A complete self-aligned model was obtained using *SolidWorks* and *COSMOSMotion*, where all the joints between the bodies have been adequately selected in order to obtain a model with no redundant constraints.

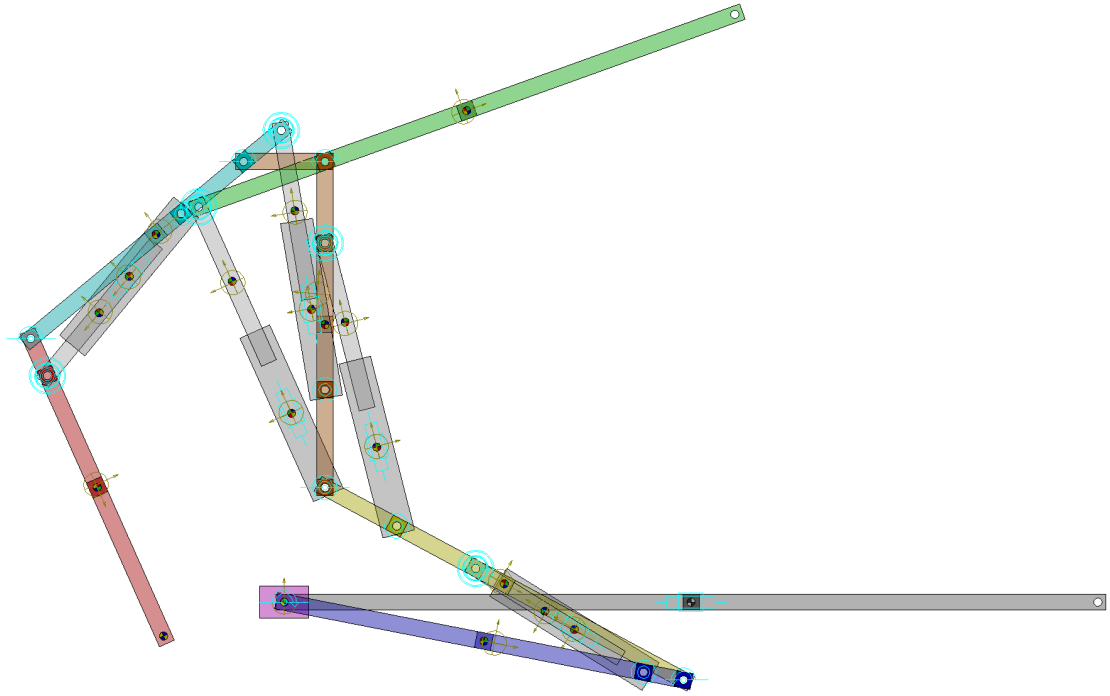


Figure 4.3: *SolidWorks COSMOSMotion* representation of the complete self-aligned model where the joint position and types are depicted.

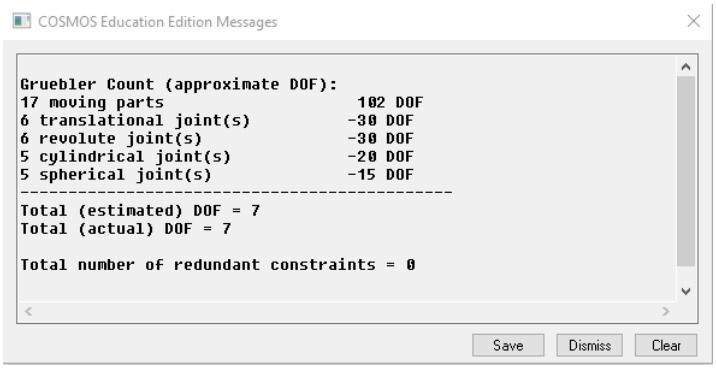


Figure 4.4: COSMOSMotion analysis results for the self-aligned model.

4.2.2 COSMOSMotion kinematic motion results

A series of motions are applied to the degrees of freedom of the model, making the tracer point follow a given path. The coordinates of the position, velocity and acceleration of the tracer point's trajectory will be exported for further analysis.

4.2.2.1 Position results

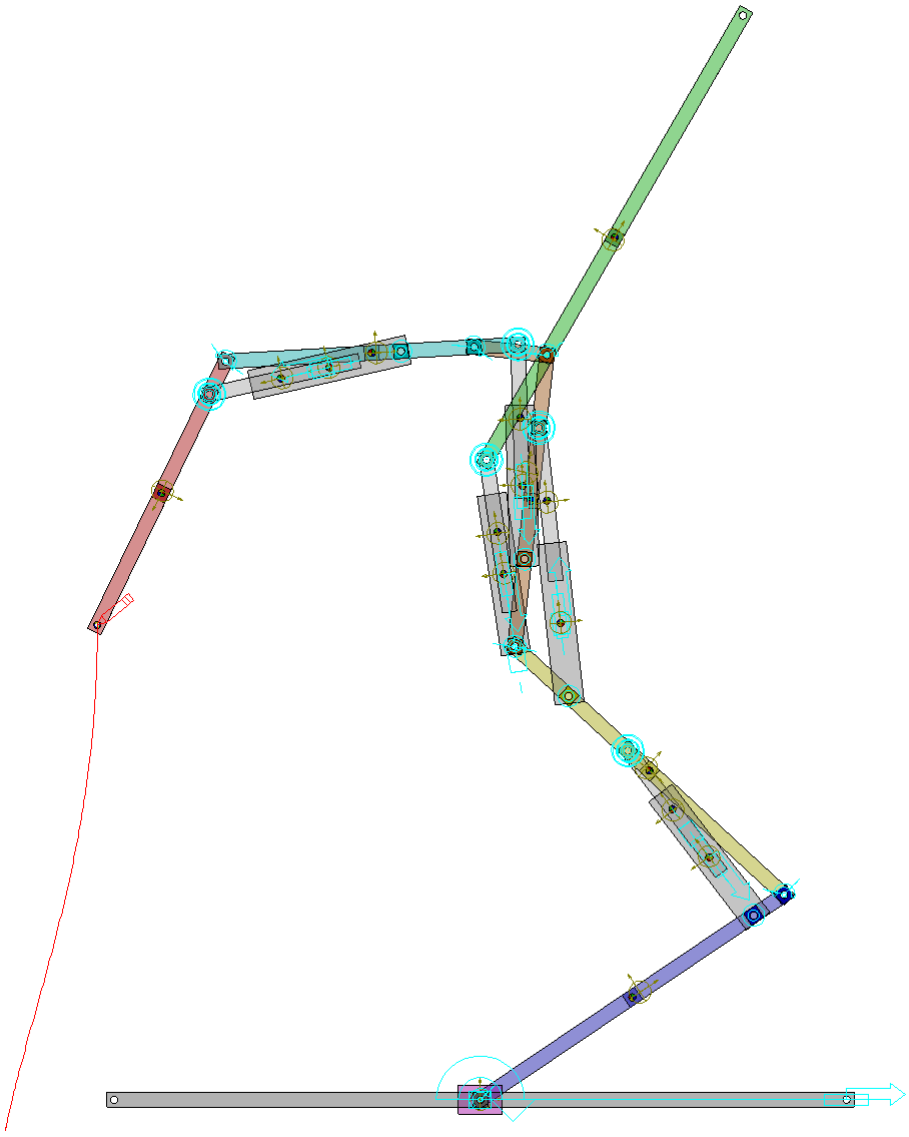
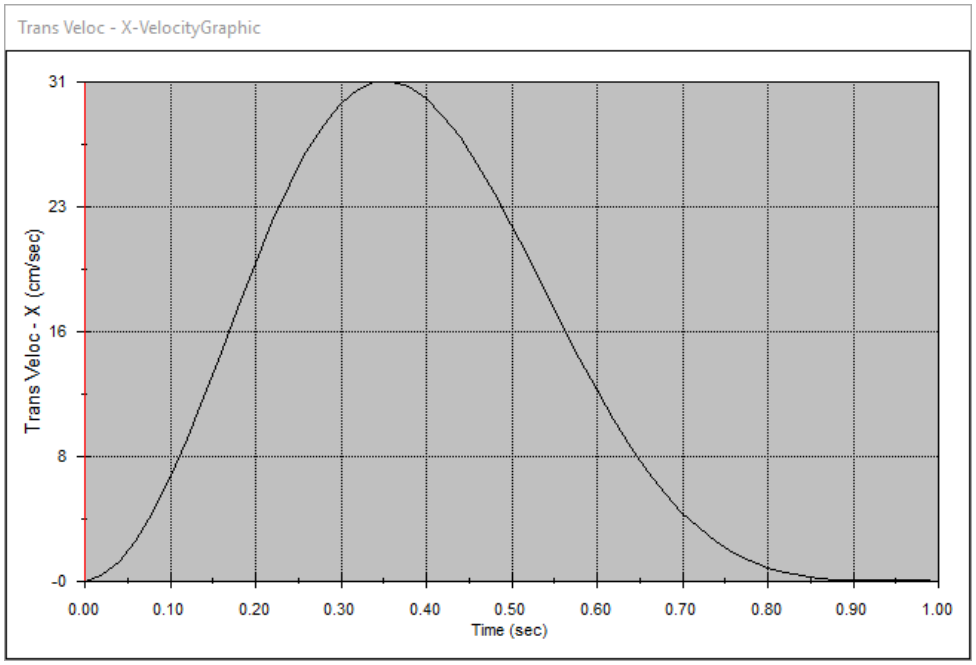
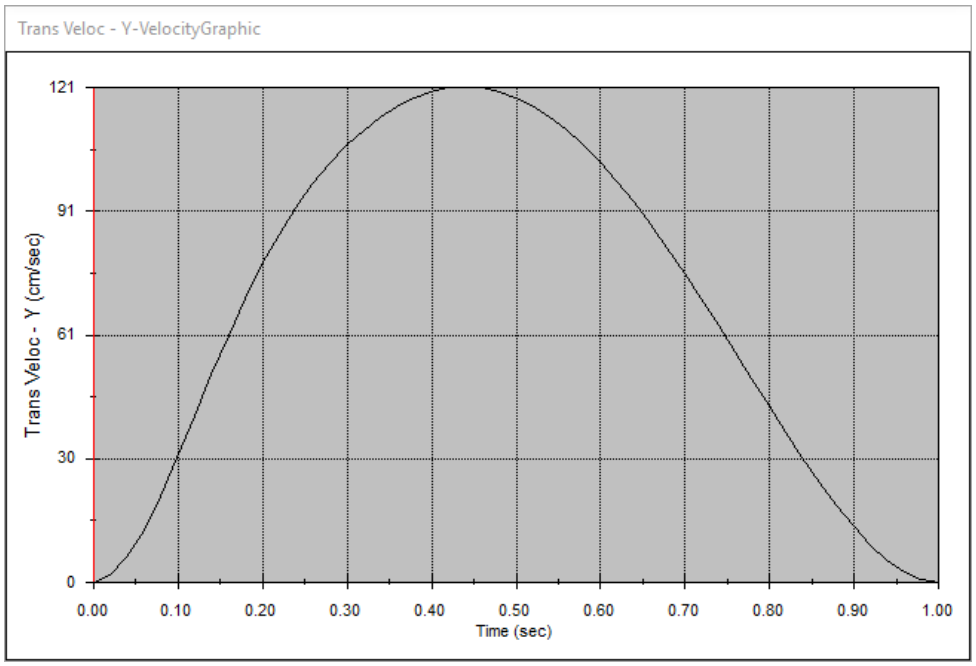


Figure 4.5: Path followed by the tracer point in COSMOSMotion.

4.2.2.2 Velocity results



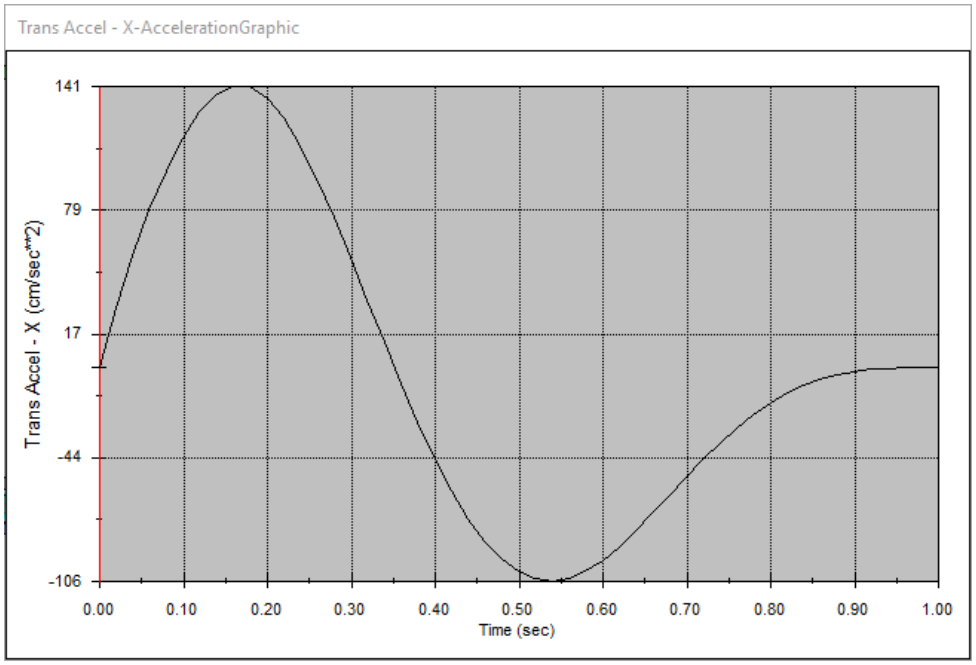
(a) Velocity x-coordinate.



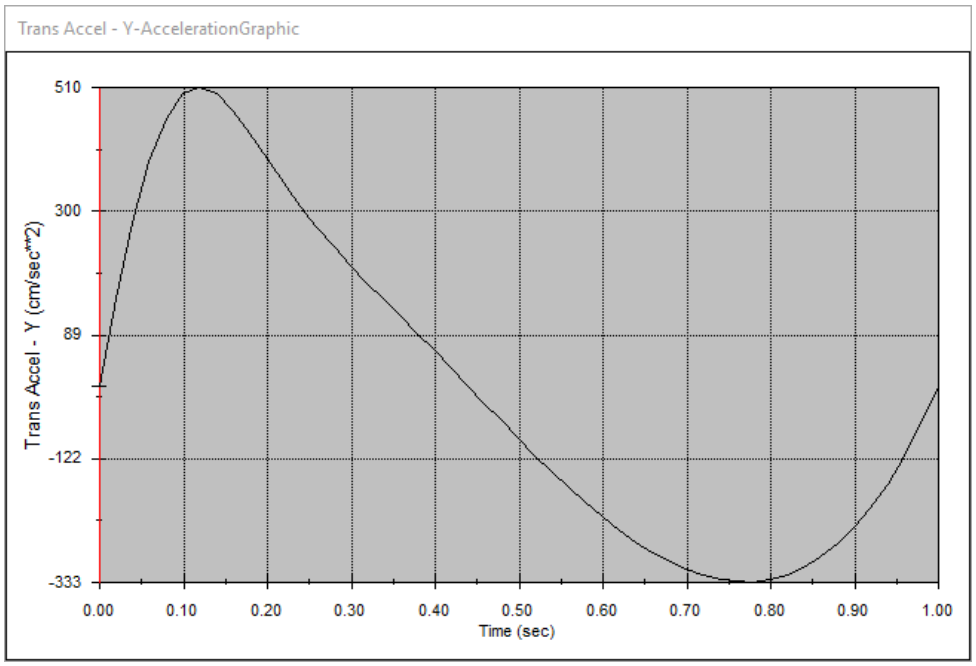
(b) Velocity y-coordinate.

Figure 4.6: Velocity of the tracer point during the motion in COSMOSMotion.

4.2.2.3 Acceleration results



(a) Acceleration x-coordinate.



(b) Acceleration y-coordinate.

Figure 4.7: Acceleration of the tracer point during the motion in COSMOSMotion.

4.3 *Mathematica*

4.3.1 Kinematic model

4.3.1.1 Equations of motion

The canonical forms for the position and trigonometric functions of body $i \in [3, 8]$ are:

$$\begin{gathered} X_i = X_2 + x_i = X_2 + a_i \cos\Theta_3 - b_i \sin\Theta_3 \\ Y_i = y_i = b_i \cos\Theta_3 + a_i \sin\Theta_3 \\ \cos\Theta_i = c_i \cos\Theta_3 - d_i \sin\Theta_3 \qquad \sin\Theta_i = d_i \cos\Theta_3 + c_i \sin\Theta_3 \end{gathered} \tag{4.1}$$

The expression for the center of mass of bodies 3 through 8 also presents a canonical form:

$$\begin{gathered} X_G = X_2 + x_G = X_2 + a_G \cos\Theta_3 - b_G \sin\Theta_3 \\ Y_G = y_G = b_G \cos\Theta_3 + a_G \sin\Theta_3 \end{gathered} \tag{4.2}$$

The moment of inertia for the grouping of bodies 3 through 8 is a function of the actuator lengths:

$$I_G = f(La_1, La_2, La_3, La_4, La_5) \tag{4.3}$$

The coefficients of the canonical forms have been obtained symbolically, in terms of the parameters defined in Table 3.4, but they will be presented with the numerical value of the parameters substituted, which result in simpler expressions. All the expressions of the canonical forms are in terms of the degrees of freedom of the model: X_2, Θ_3, La_1, La_2, La_3, La_4, and La_5. The expression for the moment of inertia is too large to fit in a single page, therefore it will not be presented here.

The parameters are defined as rational numbers allowing *Mathematica* to express the resulting expressions without any accuracy loss due to the use of real numbers with finite precision.

Body 2

$$\begin{gathered} X_2 = f(T) \\ Y_2 = 0 \\ \Theta 2 = 0 \end{gathered} \tag{4.4}$$

Body 3

$$\begin{gathered} a_3 = \frac{1}{4} \\ b_3 = 0 \\ \Theta_3 = f(T) \end{gathered} \tag{4.5}$$

Body 4

$$a_4 = \frac{1}{580}\left(5000La_1^2 - 143\right) \tag{4.6}$$

$$b_4 = \frac{1}{145}\sqrt{289 - 2500La_1^2}\sqrt{625La_1^2 - 36}$$

$$c_4 = \frac{1}{145}\left(433 - 5000La_1^2\right)$$

$$d_4 = -4\frac{1}{145}\sqrt{289 - 2500La_1^2}\sqrt{625La_1^2 - 36}$$

Body 5

(4.7)

$$a_5 = \frac{12500La_1^2\left(20La_2^2+1\right) + 8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4} - 21650La_2^2 + 5}{2175}$$

$$b_5 = \frac{2\Big(100\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2 - 5000La_1^2\sqrt{-625La_2^4+125La_2^2-4} + 5\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2} + 433\sqrt{-625La_2^4+125La_2^2-4}\Big)}{2175}$$

$$c_5 = \frac{1}{435}\left(\left(433 - 5000La_1^2\right)\left(5 - 50La_2^2\right) + 8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\right)$$

$$d_5 = \frac{1}{435}\left(20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(10La_2^2-1\right) + 2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}\right)$$

Body 6

(4.8)

$$a_6 = \tfrac{500}{87}La_1^2\left(20La_2^2+1\right) + \tfrac{1}{435}\left(5000La_1^2-433\right)\left(10La_2^2-1\right) + \frac{37(7489-40000La_3^2)\Big((433-5000La_1^2)(5-50La_2^2)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)}{459360000} +$$
$$+\frac{37\sqrt{(12769-40000La_3^2)(40000La_3^2-2209)}\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2+2(433-5000La_1^2)\sqrt{-625La_2^4+125La_2^2-4}-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\Big)}{459360000} + \frac{16\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}}{2175} - \tfrac{866La_2^2}{87} + \tfrac{1}{435}$$

$$b_6 = -\frac{37\Big((7489-40000La_3^2)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}(5-50La_2^2)+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\Big)+\sqrt{(12769-40000La_3^2)(40000La_3^2-2209)}\Big(25000La_1^2(10La_2^2-1)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)\Big)}{459360000} +$$
$$+\frac{20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}(10La_2^2-1)+2(433-5000La_1^2)\sqrt{-625La_2^4+125La_2^2-4}}{2175} + \frac{2\Big(100\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2-5000La_1^2\sqrt{-625La_2^4+125La_2^2-4}+5\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+433\sqrt{-625La_2^4+125La_2^2-4}\Big)}{2175}$$

$$c_6 = \frac{(7489-40000La_3^2)\Big((433-5000La_1^2)(5-50La_2^2)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)+\sqrt{(12769-40000La_3^2)(40000La_3^2-2209)}\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2+2(433-5000La_1^2)\sqrt{-625La_2^4+125La_2^2-4}-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\Big)}{2296800}$$

$$d_6 = \frac{(7489-40000La_3^2)\Big(-\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}(5-50La_2^2)+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big)-\sqrt{(12769-40000La_3^2)(40000La_3^2-2209)}\Big(25000La_1^2(10La_2^2-1)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)}{2296800}$$

Body 7

(4.9)

$$a_7 = \frac{1}{961350}\Big(5525000La_1^2\left(20La_2^2+1\right)-44200\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}La_2^2+442\left(433-5000La_1^2\right)\left(5-50La_2^2\right)-\frac{14}{3}\left(1750La_4^2+\sqrt{-390625La_4^4+71875La_4^2-2809}-161\right)\Big(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)+$$

$$-\frac{7}{15}\Big(-6250La_4^2+28\sqrt{-390625La_4^4+71875La_4^2-2809}+575\Big)\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\Big)+442\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}+7072\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+4420\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}-9569300La_2^2+2210\Big)$$

$$b_7 = \frac{1}{961350}\Big(11050\left(5000La_1^2-433\right)La_2^2+8840\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(10La_2^2-1\right)+\frac{14}{3}\Big(1750La_4^2+\sqrt{-390625La_4^4+71875La_4^2-2809}-161\Big)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(5-50La_2^2\right)+2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{7}{15}\Big(-6250La_4^2+28\sqrt{-390625La_4^4+71875La_4^2-2809}+575\Big)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)+$$

$$-884\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}+1768\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+884\Big(100\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2-5000La_1^2\sqrt{-625La_2^4+125La_2^2-4}+5\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+433\sqrt{-625La_2^4+125La_2^2-4}\Big)-1105\left(5000La_1^2-433\right)\Big)$$

$$c_7 = \frac{1}{663}\Big(\frac{2}{87}\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)\Big(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{1}{435}\Big(-6250La_4^2+28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}+575\Big)\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\Big)\Big)$$

$$d_7 = \frac{1}{663}\Big(-\frac{2}{87}\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(5-50La_2^2\right)+2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)-\frac{1}{435}\Big(-6250La_4^2+28\sqrt{-390625La_4^4+71875La_4^2-2809}+575\Big)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)\Big)$$

Body 8

(4.10)

$$a_8 = -\frac{1}{87}4\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}La_2^2+\frac{1}{435}\left(5000La_1^2-433\right)\left(10La_2^2-1\right)+\frac{\left(\frac{2}{87}\Big(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)+\frac{1}{435}\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big(-6250La_4^2+28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}+575\Big)\Big)\left(601-10000La_5^2\right)}{795600}+$$

$$+\frac{2}{435}\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+\frac{\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}}{2175}+\frac{8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}}{2175}+\frac{12500\left(20La_2^2+1\right)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+5}{2175}-\frac{\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big(-6250La_4^2+28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}+575\Big)}{848250}+$$

$$\frac{\Big(\frac{2}{87}\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)+\frac{1}{435}\Big(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big(6250La_4^2-28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-575\Big)\Big)\sqrt{841-10000La_5^2}\left(10000La_5^2-361\right)}{795600}-\frac{\left(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\right)}{84825}$$

$$b_8 = \frac{1}{87}\left(5000La_1^2-433\right)La_2^2+\frac{4}{435}\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(10La_2^2-1\right)+\frac{\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)-2809}-161\right)\left(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}(5-50La_2^2)+2(5000La_1^2-433)\sqrt{-625La_4^2+125La_2^2-4}\right)}{84825}+$$

$$-\frac{\left(601-10000La_5^2\right)\Big(\frac{2}{87}\Big(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)-2809}-161\Big)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(5-50La_2^2\right)+2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{1}{435}\Big(-6250La_4^2+28\sqrt{-390625La_4^4+71875La_4^2-2809}+575\Big)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)\Big)}{795600}-\frac{\sqrt{841-10000La_5^2}\left(10000La_5^2-361\right)\Big(\frac{1}{435}\Big(6250La_4^2-28\sqrt{3125La_4^2(23-125La_4^2)-2809}-575\Big)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(5-50La_2^2\right)+2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{2}{87}\Big(1750La_4^2+\sqrt{-390625La_4^4+71875La_4^2-2809}-161\Big)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)\Big)}{795600}+$$

$$+\frac{\left(-6250La_4^2+28\sqrt{-390625La_4^4+71875La_4^2-2809}+575\right)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)}{848250}+\frac{4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}}{2175}+\frac{2\Big(100\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2-5000La_1^2\sqrt{-625La_2^4+125La_2^2-4}+5\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+433\sqrt{-625La_2^4+125La_2^2-4}\Big)}{2175}-\frac{2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}}{2175}+\frac{1}{870}\left(433-5000La_1^2\right)$$

$$c_8 = \frac{1}{210}\Big(\frac{1}{663}\left(601-10000La_5^2\right)\Big(\frac{2}{87}\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)\Big(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{1}{435}\Big(-6250La_4^2+28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}+575\Big)\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\Big)\Big)+$$

$$\frac{1}{663}\sqrt{\left(841-10000La_5^2\right)\left(10000La_5^2-361\right)}\Big(\frac{2}{87}\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)\Big(200\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}La_2^2+2\left(433-5000La_1^2\right)\sqrt{-625La_2^4+125La_2^2-4}-20\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\Big)+\frac{1}{435}\Big(6250La_4^2-28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-575\Big)\Big(\left(433-5000La_1^2\right)\left(5-50La_2^2\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}\Big)\Big)\Big)$$

$$d_8 = \frac{1}{210}\Big(-\frac{1}{663}\left(601-10000La_5^2\right)\Big(\frac{2}{87}\Big(1750La_4^2+\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-161\Big)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(5-50La_2^2\right)+2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{1}{435}\Big(-6250La_4^2+28\sqrt{-390625La_4^4+71875La_4^2-2809}+575\Big)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)\Big)+$$

$$-\frac{1}{663}\sqrt{\left(841-10000La_5^2\right)\left(10000La_5^2-361\right)}\Big(\frac{1}{435}\Big(6250La_4^2-28\sqrt{3125La_4^2\left(23-125La_4^2\right)-2809}-575\Big)\Big(4\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\left(5-50La_2^2\right)+2\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\Big)+\frac{2}{87}\Big(1750La_4^2+\sqrt{-390625La_4^4+71875La_4^2-2809}-161\Big)\Big(25000La_1^2\left(10La_2^2-1\right)+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}-21650La_2^2+2165\Big)\Big)\Big)$$

Center of mass

(4.11)

$$a_G = -\frac{1}{130(192-\pi)}\Bigg(-\frac{1}{87}291200000La_2^2La_1^2+\frac{1040000La_1^2}{87}+\frac{500000}{29}La_2^2\pi La_1^2+\frac{252179920La_2^2}{87}+\frac{37\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)(40000La_3^2-7489)}{79750}+\frac{56000\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(1750La_4^2+\sqrt{31125La_4^2(23-125La_4^2)}-2809-161\right)La_3^2}{57681}+$$

$$+\frac{5600\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(6250La_4^2-28\sqrt{3125La_4^2(23-125La_4^2)}-2809-575\right)La_3^2}{57681}-\frac{37\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)(40000La_3^2-7489)\pi}{22968000}-\frac{1000\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)La_3^2\pi}{173043}+$$

$$+\frac{200\left(5000\sqrt{-625La_2^4+125La_2^2-4}La_1^2-100\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}La_2^2+10\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}-433\sqrt{-625La_2^4+125La_2^2-4}\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)La_3^2\pi}{173043}-\frac{37\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\sqrt{(113-200La_3)(200La_3+113)}(40000La_3^2-2209)\pi}{22968000}+\frac{5\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)\pi}{346086}+$$

$$+\frac{361\left(-3000\sqrt{-625La_2^4+125La_2^2-4}La_1^2+100La_2^2\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}-10\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+433\sqrt{-625La_2^4+125La_2^2-4}\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)\pi}{865250}-\frac{28\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(6250La_4^2-28\sqrt{3125La_4^2(23-125La_4^2)}-2809-575\right)\pi}{1440025}+\frac{\left(-5000\sqrt{-625La_2^4+125La_2^2-4}La_1^2+100La_2^2\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}-10\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+433\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)\sqrt{841-10000La_4^2}\sqrt{10000La_4^2-361}\pi}{946345}+$$

$$-\frac{\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)\sqrt{841-10000La_4^2}\sqrt{10000La_4^2-361}\pi}{17304300}+\frac{960}{29}\left(10La_2^2-1\right)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+\frac{4}{87}\left(5-50La_2^2\right)\pi\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}-\frac{96}{29}\left(5000La_1^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}-\frac{46592}{435}\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}\sqrt{-625La_2^2+125La_2^2-4}+\frac{16}{29}\pi\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+$$

$$+\frac{2}{87}\left(5000La_1^2-433\right)\pi\sqrt{-625La_2^4+125La_2^2-4}+\frac{2}{435}(-80+\pi)\left(25000\left(10La_2^2-1\right)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)+\frac{74\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}(1-10La_2^2)\sqrt{(12769-40000La_2^2)(40000La_2^2-2209)}}{7975}+\frac{37(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\sqrt{(12769-40000La_2^2)(40000La_2^2-2209)}}{39875}-\frac{140\left(20000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)}{57681}+$$

$$-\frac{14\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(6250La_4^2-28\sqrt{3125La_4^2(23-125La_4^2)}-2809-575\right)}{57681}+\frac{28\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)\sqrt{29-100La_3}\sqrt{100La_3+29}\sqrt{10000La_3^2-361}}{288405}+\frac{14\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)}{1442025}+\frac{43300La_3^2\pi}{29}-\frac{1085344}{87}+60\pi\Bigg)$$

$$b_G = \frac{1}{130(192-\pi)}\Bigg(\frac{120000000La_2^2La_1^2}{29}-\frac{66560000}{87}\sqrt{-625La_2^4+125La_2^2-4}La_1^2+\frac{12000}{29}\pi\sqrt{-625La_2^4+125La_2^2-4}La_1^2-\frac{1200000La_1^2}{29}-\frac{250000}{87}La_2^2\pi La_1^2+\frac{25000\pi La_1^2}{87}-\frac{37\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)(7489-40000La_3^2)}{79750}+\frac{56000\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)La_3^2}{57681}+$$

$$+\frac{5600\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)La_3^2}{57681}+\frac{37\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)(7489-40000La_3^2)\pi}{22968000}-\frac{1000\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)La_3^2\pi}{173043}+$$

$$-\frac{100\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)La_3^2\pi}{173043}+\frac{37\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\sqrt{(12769-40000La_2^2)(40000La_2^2-2209)}\pi}{22968000}+\frac{16\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}(10La_2^2-1)\sqrt{3125La_4^2(23-125La_4^2)}-2809\pi}{3393}+\frac{8(433-5000La_2^2)\sqrt{-625La_2^4+125La_2^2-4}\sqrt{3125La_4^2(23-125La_4^2)}-2809\pi}{16965}+$$

$$+\frac{433\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)\pi}{1730430}+\frac{\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)\pi}{692172}+\frac{\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)\sqrt{841-10000La_4^2}\sqrt{10000La_4^2-361}\pi}{1730430}+$$

$$+\frac{\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(6250La_4^2-28\sqrt{3125La_4^2(23-125La_4^2)}-2809-575\right)\sqrt{841-10000La_4^2}\sqrt{10000La_4^2-361}\pi}{17304300}+\frac{133120}{87}La_2^2\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+\frac{9152}{87}\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+\frac{282368(10La_2^2-1)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}}{1131}+\frac{28000(10La_2^2-1)La_1^2\pi\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}}{3393}+\frac{156800(5000La_1^2-433)La_2^4\sqrt{-625La_2^4+125La_2^2-4}}{3393}+$$

$$+\frac{2882048}{405}\sqrt{-625La_2^4+125La_2^2-4}+\frac{4384}{87}\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+\frac{47465000La_1^2(-433)\pi\sqrt{-625La_2^4+125La_2^2-4}}{3393}-\frac{37\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\sqrt{(12769-40000La_2^2)(40000La_2^2-2209)}}{79750}+\frac{896\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}(1-10La_2^2)\sqrt{3125La_4^2(23-125La_4^2)}-2809}{1131}+\frac{448\left(5000La_2^2-433\right)\sqrt{-625La_2^4+125La_2^2-4}\sqrt{3125La_4^2(23-125La_4^2)}-2809}{5655}-\frac{12124\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)}{288405}+$$

$$-\frac{14\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(-6250La_4^2+28\sqrt{3125La_4^2(23-125La_4^2)}-2809+575\right)}{57681}-\frac{28\left(25000(10La_2^2-1)La_1^2-21650La_2^2+8\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}\sqrt{-625La_2^4+125La_2^2-4}+2165\right)\left(1750La_4^2+\sqrt{3125La_4^2(23-125La_4^2)}-2809-161\right)\sqrt{(841-10000La_5^2)(10000La_5^2-361)}}{288405}+$$

$$-\frac{14\left(4(5-50La_2^2)\sqrt{625La_1^2-36}\sqrt{289-2500La_1^2}+2(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}\right)\left(6250La_4^2-28\sqrt{3125La_4^2(23-125La_4^2)}-2809-575\right)\sqrt{(841-10000La_5^2)(10000La_5^2-361)}}{1442025}-\frac{1039200La_3^2}{29}+\frac{103920}{29}-\frac{2165\pi}{87}-\frac{1568000\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}(10La_2^2-1)La_1^2}{1131}-\frac{2800(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}La_4^2\pi}{3393}+$$

$$-\frac{4760\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}(10La_2^2-1)\pi}{3393}-\frac{141184(5000La_1^2-433)\sqrt{-625La_2^4+125La_2^2-4}}{5655}-\frac{240}{29}\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36La_2^2}\pi+\frac{21650La_2^2\pi}{87}-\frac{16}{29}\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}\pi-\frac{5196}{145}\sqrt{-625La_2^4+125La_2^2-4}\pi-\frac{8}{87}\sqrt{289-2500La_1^2}\sqrt{625La_1^2-36}\sqrt{-625La_2^4+125La_2^2-4}\pi\Bigg)$$

4.3.1.2 Comparison between Mathematica and COSMOSMotion kinematic results

The same kinematic simulation as in *COSMOSMotion* was performed using the symbolic kinematic solution in *Mathematica*, obtaining the same results for the position, velocity, and acceleration of the tracer point for the specified motion.

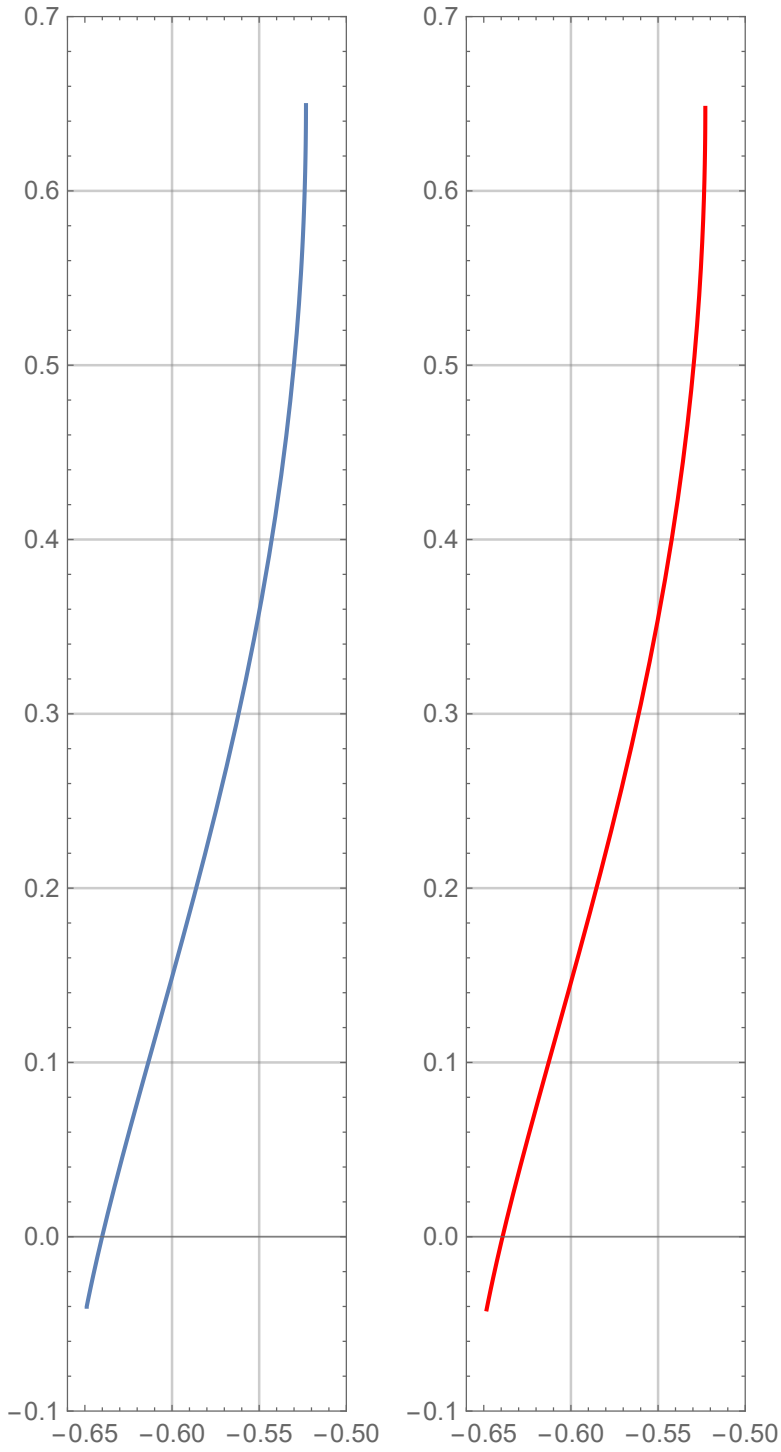


Figure 4.8: Comparison of tracer point trajectories during the kinematic simulation between *Mathematica* symbolic solution (blue) and *COSMOSMotion* numeric solution (red). Y vs. X coordinate parametric plot in m.

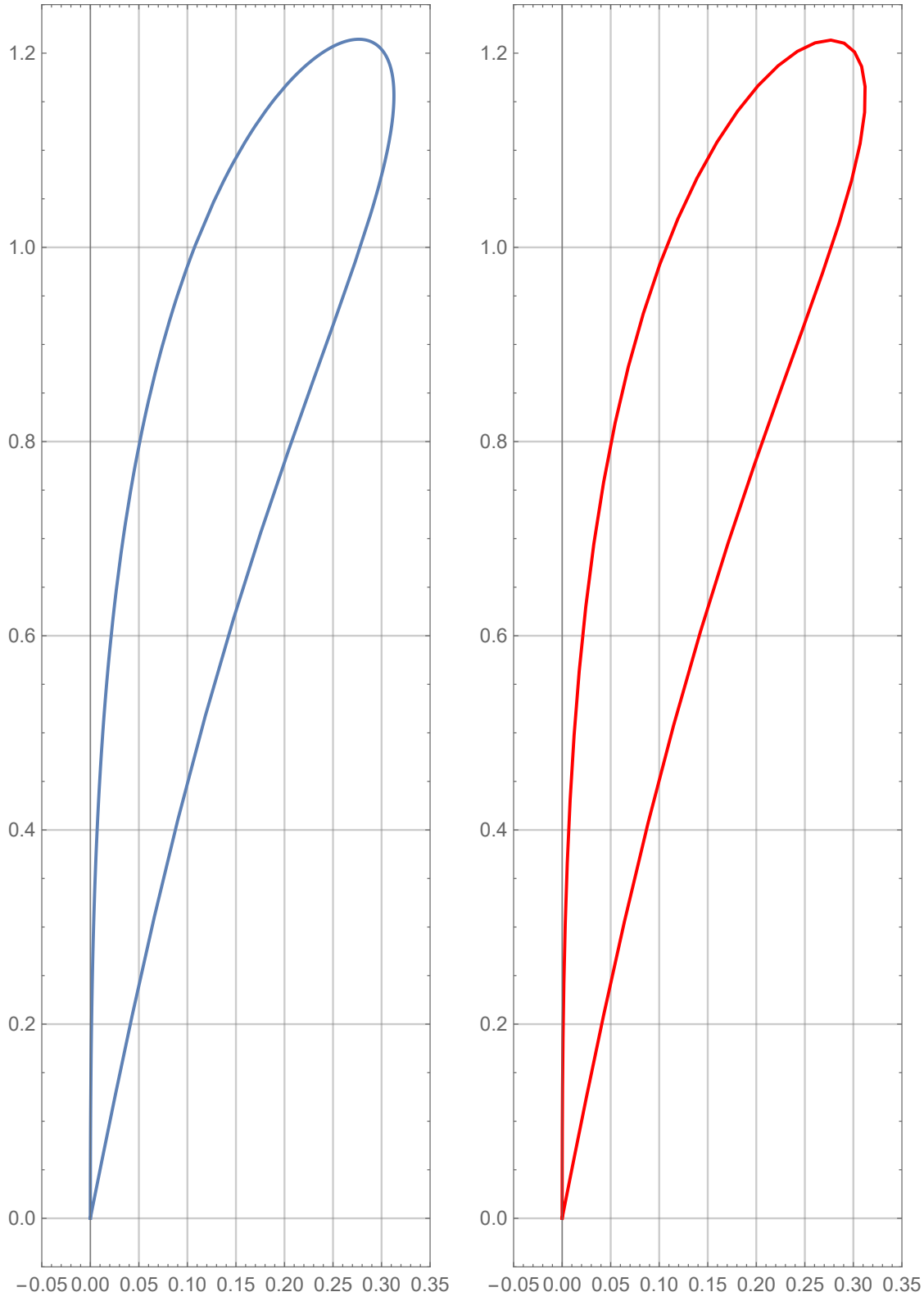


Figure 4.9: Comparison of tracer point velocities during the kinematic simulation between *Mathematica* symbolic solution (blue) and *COSMOSMotion* numeric solution (red). Y vs. X coordinate parametric plot in m/s.

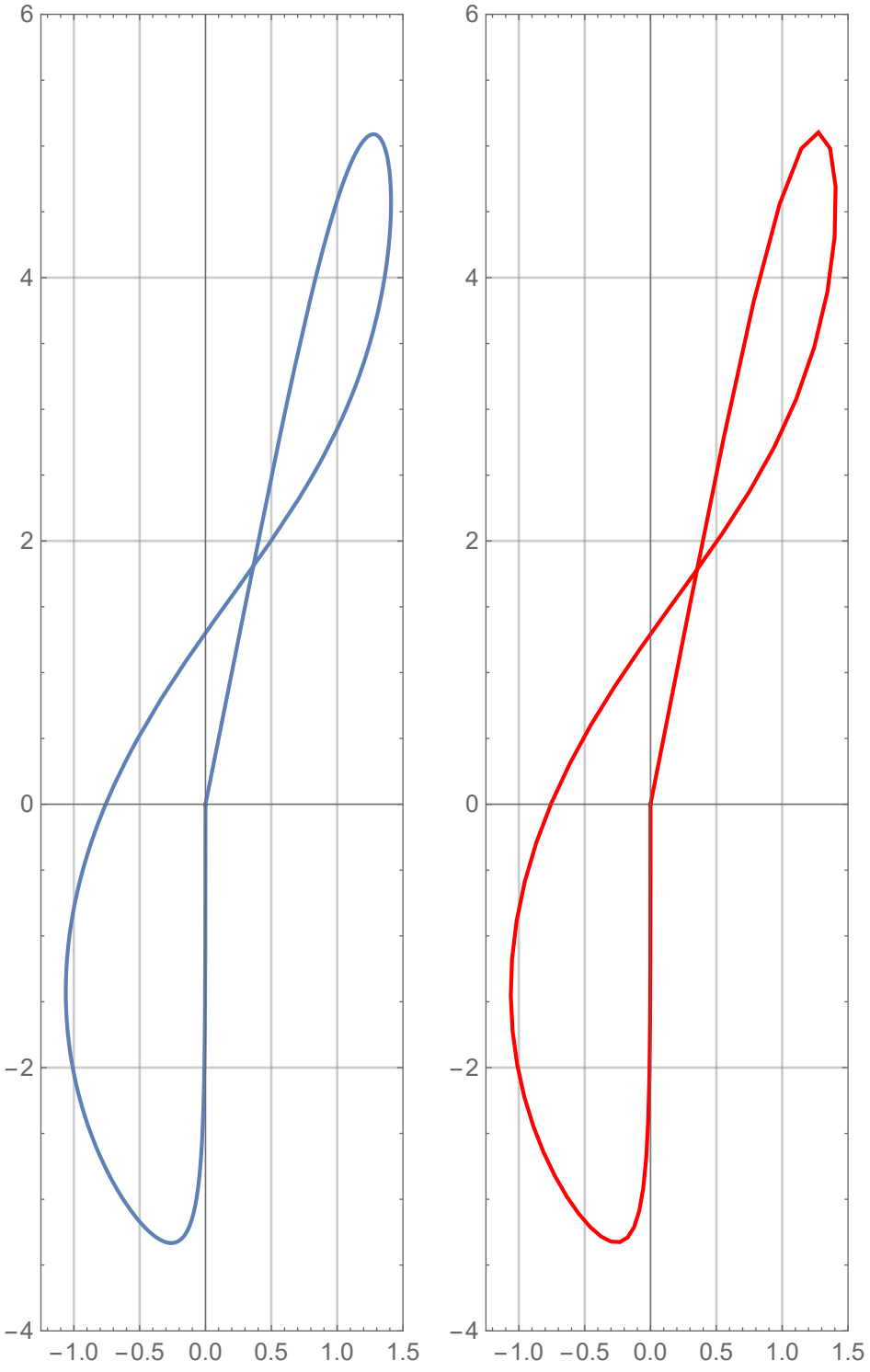


Figure 4.10: Comparison of tracer point accelerations during the kinematic simulation between *Mathematica* symbolic solution (blue) and *COSMOSMotion* numeric solution (red). Y vs. X coordinate parametric plot in m/s^2.

4.3.2 Working range

The working range for the tracer point considering the range of variation of the lengths of the actuators is a two dimensional area that has been obtained for the initial value of Θ_3. The movement of the slider X_2 is simply a translation of this region and the rotation of the Θ_3 will rotate the region around the pivot point of the revolute joint.

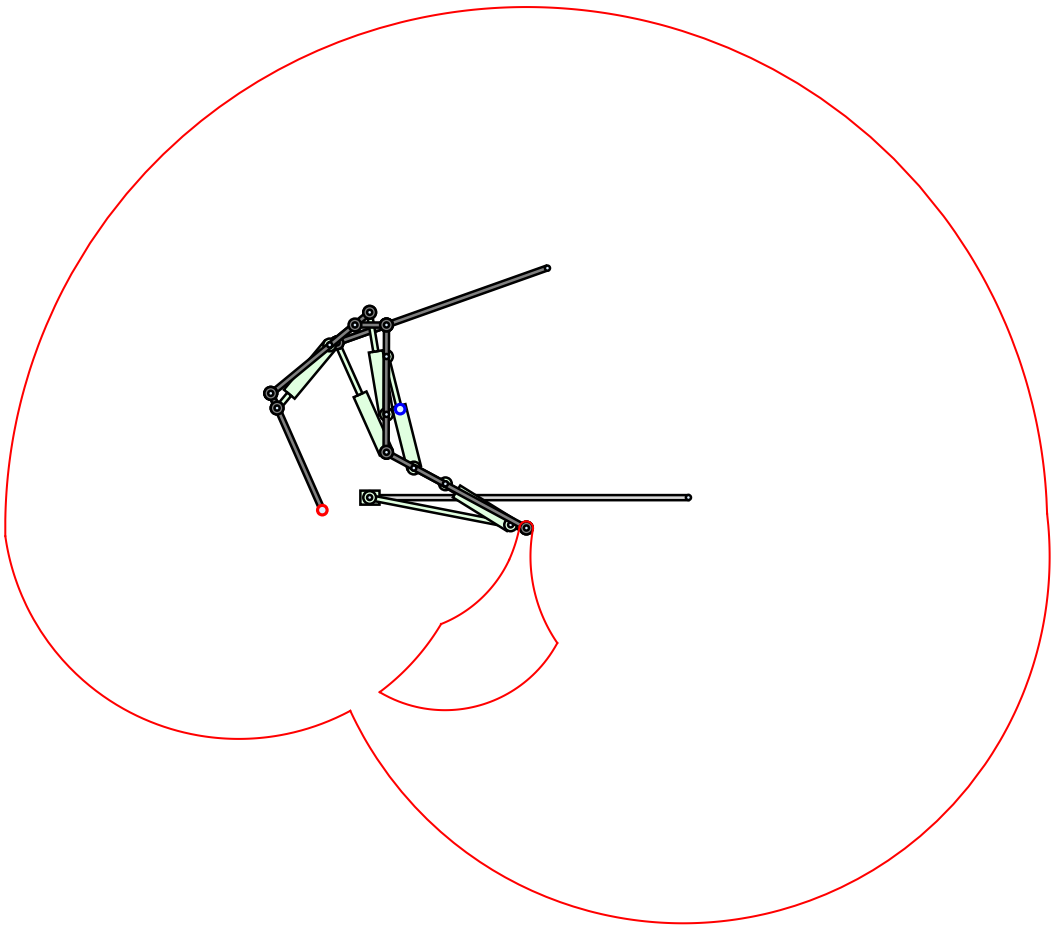


Figure 4.11: Working range of the tracer point for variations in La_1, La_2, La_4, and La_5 for the initial position of X_2 and Θ_3.

4.3.3 Interactive visual representation

An interactive visual representation was developed using the **Manipulate** function in *Mathematica*. This has been possible due to the kinematic expressions obtained for the position of the bodies and to the working range analysis to obtain the maximum and minimum values for the actuator lengths.

The relative angle formulation for the actuator lengths yielded symbolic kinematic expressions that were simplified enough for the model to be recalculated in real time when the values of the drivers are changed dynamically using the sliders in the interactive controls.

The values of all the degrees of freedom of the model can be modified inside a determined minimum and maximum range. There is also an option to overlap the working range and a reset button to return to the initial position.

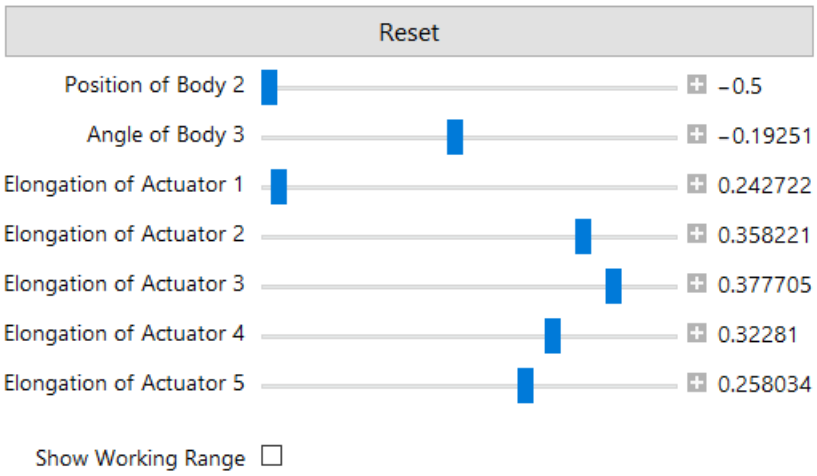


Figure 4.12: Interactive controls in *Mathematica*.

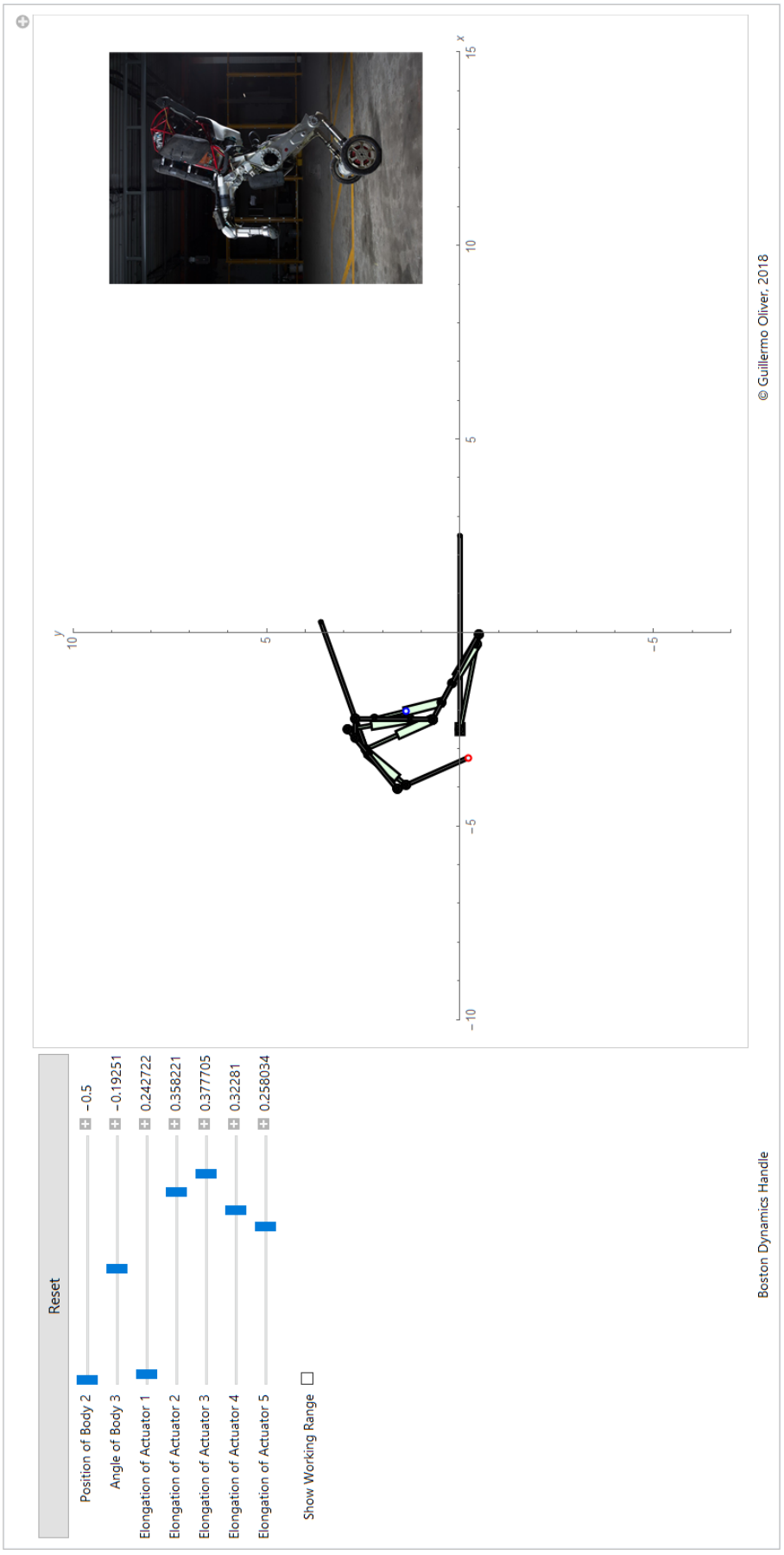


Figure 4.13: Interactive visual representation in *Mathematica*.

4.3.4 Dynamic model

The dynamic model for the *Handle* robot has been reduced to a inverted pendulum on cart system with the following equations of motion:

$$\boxed{M\frac{\partial x_G}{\partial \Theta}\ddot{x} + (I_G + M\ell^2)\ddot{\Theta} + M\frac{\partial y_G}{\partial \Theta}g = 0} \tag{4.12}$$

$$\boxed{(M+m)\ddot{x} + M\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 + M\frac{\partial x_G}{\partial \Theta}\ddot{\Theta} = F} \tag{4.13}$$

Where:

$$\mathbf{s_G} = \{x_G, y_G\} = \{a\cos\Theta_3 - b\sin\Theta_3, b\cos\Theta_3 + a\sin\Theta_3\}$$

The parameters a, b, I_G, and ℓ are functions of the actuator lengths:

$$\begin{aligned} a &= f(La_1, La_2, La_3, La_4, La_5) \qquad & b &= g(La_1, La_2, La_3, La_4, La_5) \\ I_G &= h(La_1, La_2, La_3, La_4, La_5) \qquad & \ell &= k(La_1, La_2, La_3, La_4, La_5) \end{aligned} \tag{4.14}$$

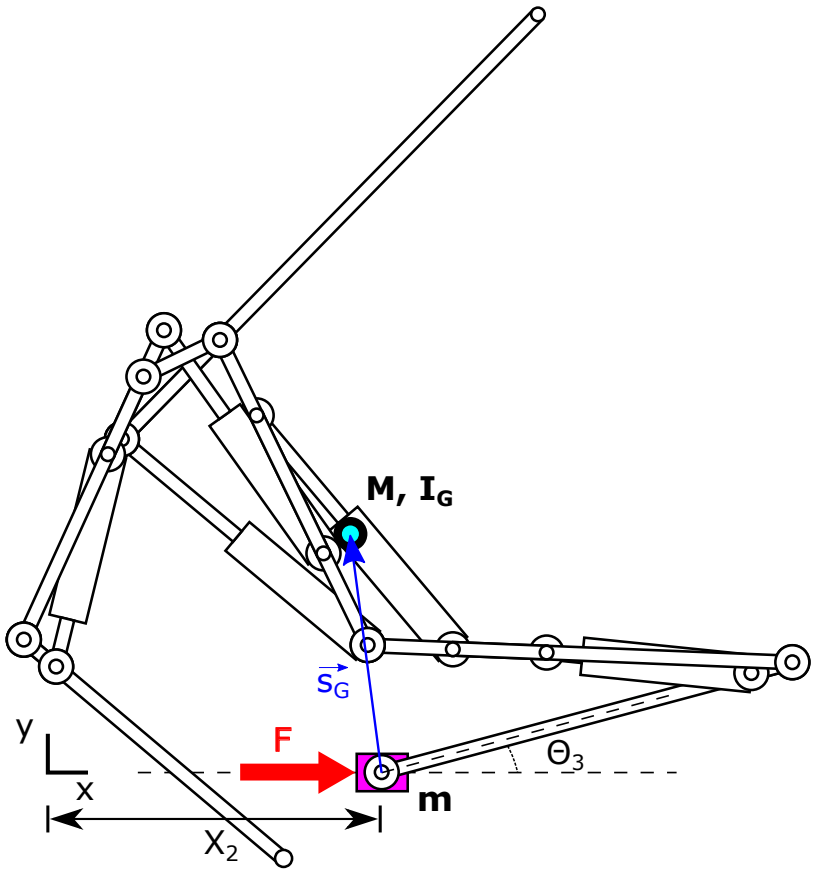


Figure 4.14: Representation of the equivalent inverted pendulum on cart model.

4.3.4.1 Free motion results

A dynamic simulation of the model in its starting position swinging under the action of the force of gravity has been performed in *Mathematica* and *RecurDyn*.

In *Mathematica*, the numerical free motion results from the simplified inverted pendulum on cart model are the same as the numerical solution of the complete eight body system. The eight body system was solved using the general Lagrange multiplier form of the constrained equations of motion and, even though the two equations of motion obtained were not as simplified as the ones of the inverted pendulum model, they yielded the same results as the simplified model. This confirms that the simplified equivalent model has been correctly developed.

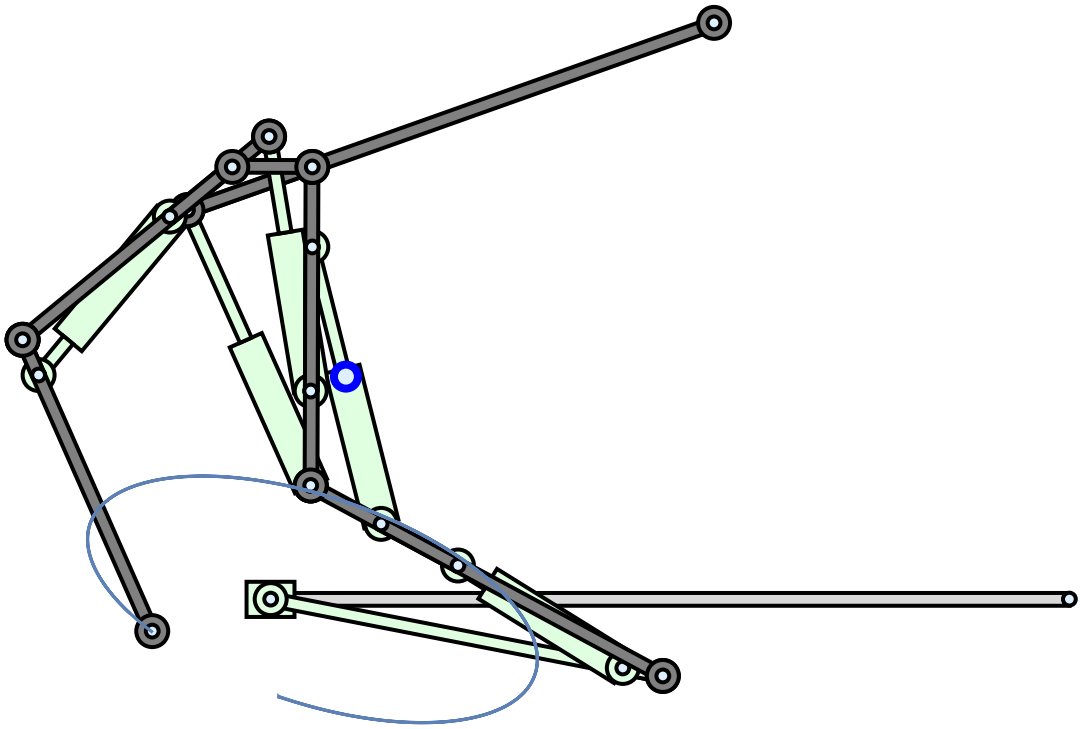


Figure 4.15: *Mathematica* model with the trajectory of the tracer point during free motion.

4.3.5 Comparison between RecurDyn and Mathematica dynamic free motion results

The free motion dynamic simulation of the model in its starting position swinging under the action of the force of gravity yielded the same results in *Mathematica* and *RecurDyn*.

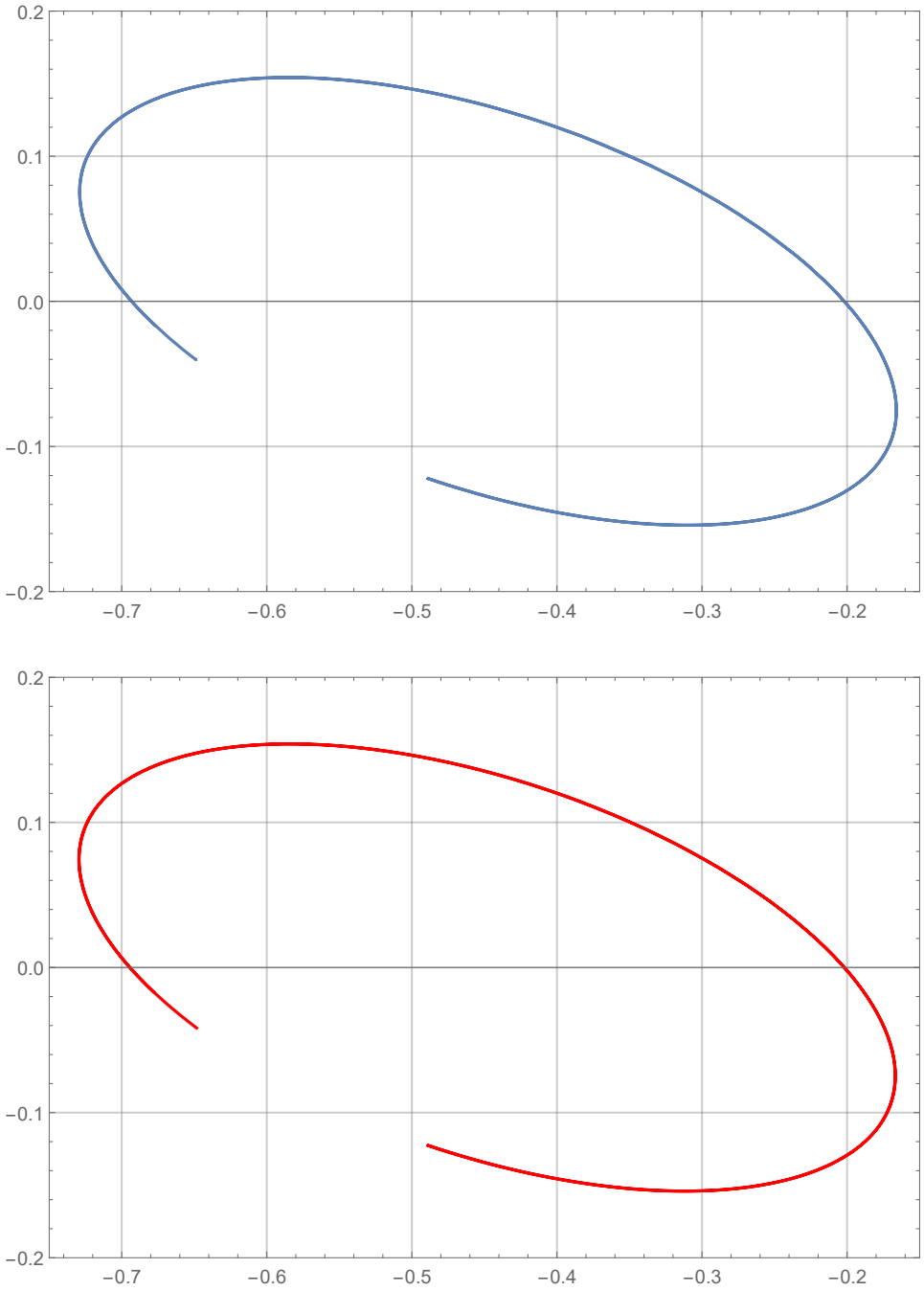


Figure 4.16: Comparison of tracer point trajectories for the free motion dynamic simulation between *Mathematica* symbolic solution (blue) and *RecurDyn* numeric solution (red). Y vs. X coordinate parametric plot in m.

4.4 *RecurDyn*

4.4.1 Model

The model starts from the initial value of Θ_3 and it is driven until the dynamic system's initial position. This initial position represents a *standing up* configuration, and it is at that moment when the controllers kick in. The initial position has a small deviation from the equilibrium point, where the center of mass of the upper part of the model should be right on top of the slider's center of mass.

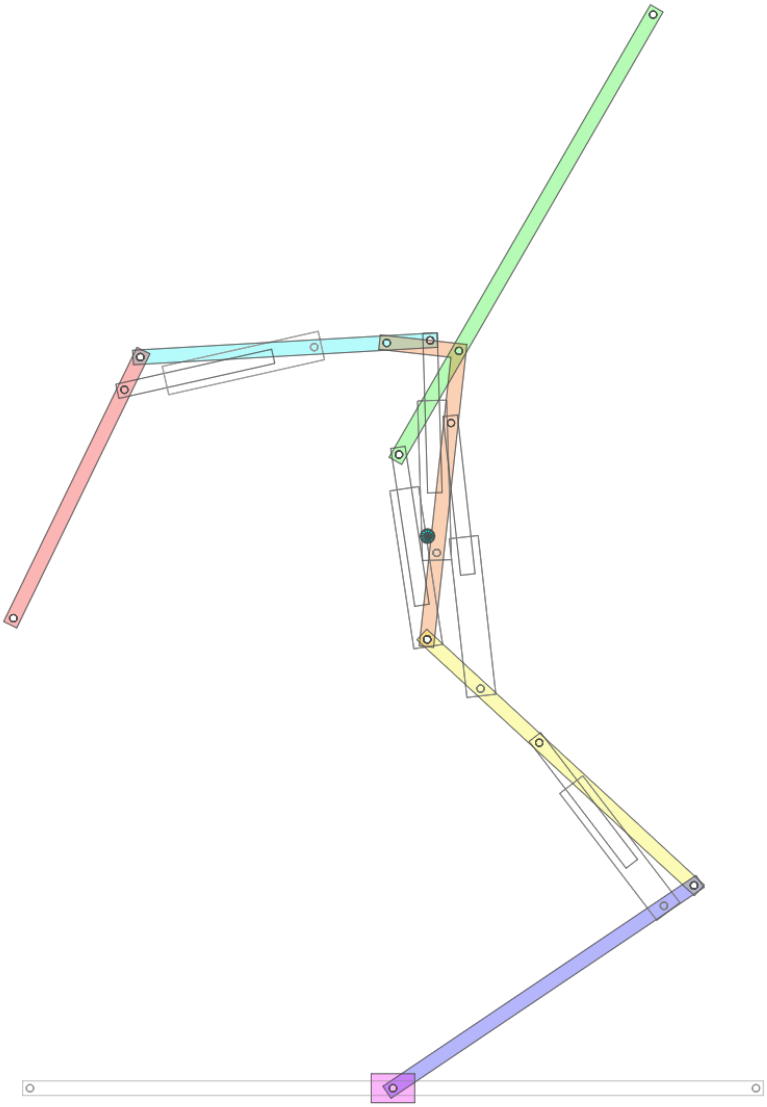


Figure 4.17: Dynamic system's starting position depicted in *RecurDyn*.

The linearized model will be considered for the design of the controllers, where the equilibrium point is:

$$\Theta_{eq} = 2\tan^{-1}(\frac{a}{b+\ell}) \tag{4.15}$$

The values for the degrees of freedom at the dynamic system's starting position are:

$$\begin{gathered} X_2 = 0\,m \qquad \Theta_3 = 33.97^\circ \\ La_1 = 0.282722\,m \qquad La_2 = 0.368221\,m \qquad La_3 = 0.257705\,m \\ La_4 = 0.29281\,m \qquad La_5 = 0.268034\,m \end{gathered} \tag{4.16}$$

The resulting parameters for those values of actuator lengths are:

$$\begin{gathered} \ell = 76.2733 \cdot 10^{-2}\,m \qquad I_G = 12.6727\,kg \cdot m^2 \qquad \Theta_{eq} = 37.5884^\circ \\ m = 48\,kg \qquad M = 61.379\,kg \qquad g = 9.80665\,\frac{m}{s^2} \end{gathered} \tag{4.17}$$

The mass parameters are invariants, but the ℓ, I_G, and Θ_{eq} parameters must be calculated according to the length of the actuators in every given point.

The open-loop transfer functions of the linearized model are:

$$\begin{aligned} \frac{\Theta(s)}{\mathbb{F}(s)} &= \frac{\ell M}{(I_G(m+M)+\ell^2 mM)s^2 - g\ell M(m+M)}\left[\frac{rad}{N}\right] \\ \frac{\mathbb{X}(s)}{\mathbb{F}(s)} &= \frac{s^2(I_G+\ell^2 M) - g\ell M}{(I_G(m+M)+\ell^2 mM)s^4 - g\ell M(m+M)s^2}\left[\frac{m}{N}\right] \end{aligned} \tag{4.18}$$

The open-loop poles of the Θ transfer function are:

$$s_1 = \sqrt{\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2 mM}} = 4.0247 \qquad s_2 = -\sqrt{\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2 mM}} = -4.0247$$

The open-loop poles of the X transfer function are:

$$s_1 = 0 \qquad s_2 = 0 \qquad s_3 = \sqrt{\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2 mM}} = 4.0247 \qquad s_4 = -\sqrt{\frac{g\ell M(m+M)}{I_G(m+M)+\ell^2 mM}} = -4.0247$$

4.4.2 Controllers

The controllers will be designed using the linearized model obtained and implemented in *RecurDyn* and *SystemModeler*.

4.4.2.1 Ziegler-Nichols closed-loop method

The root locus plot of the closed-loop transfer function for Θ shows that the system switches from unstable to oscillatory from a certain value of K_p.

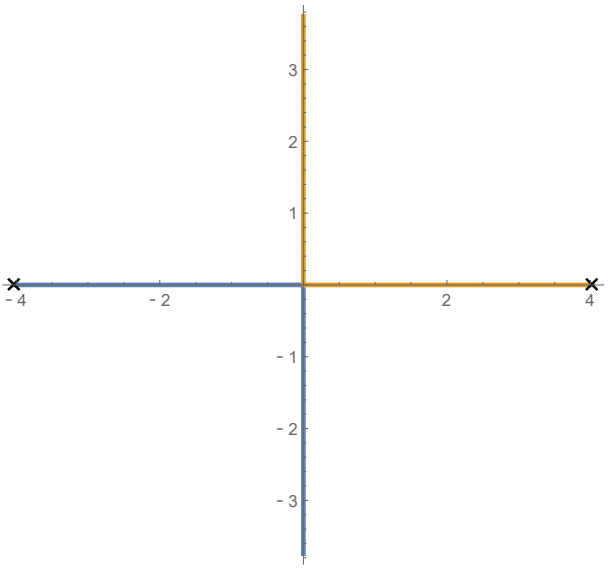


Figure 4.18: Root locus representation of the closed-loop transfer function of Θ.

Considering the numerical values of the parameters, the Ziegler-Nichols method returns the following value for the critical gain:

$$K_u = g(m + M) = 1072.642 \tag{4.19}$$

Due to the characteristics of the root locus plot, the critical period T_u for that specific value of K_p is zero. This means that the heuristic Ziegler-Nichols closed-loop method is not applicable for the system.

However, for a close value such as $K_p = 1100$, the period can be calculated:

$$T = \frac{2\pi}{\sqrt{\frac{\ell M\,(K_p - g(m+M))}{(I_G(m+M)+\ell^2 mM)}}} = 9.78\,s \tag{4.20}$$

Using *RecurDyn* it is possible to check that the value of K_u is the stability frontier and that a certain value of K_p produces oscillations with the resulting period. This shows that the linearized model is a reliable representation of the virtual model.

4.4.2.2 PID controller for pendulum angle

Using the pole placement method the solution for the parameters of a PID controller for the angle of the pendulum have been obtained:

$$K_i = \frac{-d_1 p_1 \omega_n^2}{c_3} \qquad K_p = \frac{-d_2 - 2d_1 p_1 \xi \omega_n + d_1 \omega_n^2}{c_3} \qquad K_d = \frac{-d_1(p_1 - 2\xi\omega_n)}{c_3} \tag{4.21}$$

Where:

$$d_1 = I_G(m+M) + \ell^2 mM \qquad d_2 = -g\ell M(m+M) \qquad c_3 = \ell M \tag{4.22}$$

$$\xi = \frac{-ln\frac{PO}{100}}{\sqrt{\pi^2 + ln^2\frac{PO}{100}}}$$

$$\omega_n = \frac{4}{T_{set}\,\xi}$$

Considering the numeric values of the coefficients, the settling time T_{set} and the percentage overshoot PO, as well as a fast negative pole, the coefficients are easily calculated:

$$T_{set} = 2s \qquad PO = 15\% \tag{4.23}$$
$$K_i = 19824.8 \qquad K_p = 7361.42 \qquad Kd = 1589.26$$

Where the dominant poles of the closed-loop are:

$$s_1 = -2 - 3.31196i \qquad s_2 = -2 + 3.31196i \tag{4.24}$$

The negative real pole with fast dynamics is:

$$s_3 = -20 \tag{4.25}$$

4.4.2.3 Double PID controller for pendulum angle and cart position

Using the pole placement method the solution for the parameters of a PID controller for the angle of the pendulum have been obtained:

$$K_{p_\Theta} = \frac{-c_1 d_1 \omega_n(-2p_1p_2p_3\xi+p_2p_3\omega_n+p_1(p_2+p_3)\omega_n)+c_2(-d_2+d_1(p_1p_2+p_1p_3+p_2p_3-2(p_1+p_2+p_3)\xi\omega_n+\omega_n^2))}{c_2c_3}$$
$$K_{i_\Theta} = \frac{c_1^2 d_1 p_1p_2p_3\omega_n^2+c_1c_2d_1(-p_1p_2p_3+2(p_2p_3+p_1(p_2+p_3))\xi\omega_n-(p_1+p_2+p_3)\omega_n^2)+c_2^2(c_3K_{d_\Theta}+d_1(p_1+p_2+p_3-2\xi\omega_n))}{c_1c_2c_3}$$
$$K_{p_x} = \frac{d_1\omega_n(-2p_1p_2p_3\xi + p_2p_3\omega_n + p_1(p_2+p_3)\omega_n)}{c_2}$$
$$K_{d_x} = -\frac{c_3K_{d_\Theta} + d_1(p_1+p_2+p_3-2\xi\omega_n)}{c_1}$$
$$K_{i_x} = -\frac{d_1p_1p_2p_3\omega_n^2}{c_2} \tag{4.26}$$

Where:

$$d_1 = I_G(m+M)+\ell^2mM \qquad d_2 = -g\ell M(m+M) \tag{4.27}$$
$$c_1 = I_G+\ell^2M \qquad c_2 = -g\ell M \qquad c_3 = \ell M$$

The derivative term of the Θ controller can take any arbitrary value, it will be set to $K_{d_\Theta} = 100$.

Considering the numeric values of the coefficients, the settling time T_{set}, the percentage overshoot PO, and three fast negative poles, the coefficients are calculated:

$$T_{set} = 4s \qquad PO = 5\% \tag{4.28}$$
$$K_{p_\Theta} = 8487.84 \qquad K_{d_\Theta} = 100 \qquad K_{i_\Theta} = 23644.2$$
$$K_{p_x} = -1658.31 \qquad K_{d_x} = 928.473 \qquad K_{i_x} = -907.425$$

Where the dominant poles of the closed-loop are:

$$s_1 = -1-1.04869i \qquad s_2 = -1+1.04869i \tag{4.29}$$

The negative real poles with fast dynamics are:

$$s_3 = -2 \qquad s_4 = -4 \qquad s_5 = -8 \tag{4.30}$$

4.4.2.4 State feedback control using pole placement

Using the pole placement method the solution for the parameters of a state feedback controller for the angle of the pendulum have been obtained:

$$k_{11} = \frac{-(\ell^2 mM + I_G(m+M))p_1 p_2 \omega_n^2)}{g\ell M} \tag{4.31}$$

$$k_{12} = \frac{(\ell^2 mM + I_G(m+M))\omega_n(-2p_1 p_2 \xi + (p_1+p_2)\omega_n)}{g\ell M}$$

$$k_{13} = g(m+M) + \frac{(I_G+\ell^2 M)(\ell^2 mM + I_G(m+M))p_1 p_2 \omega_n^2}{g\ell^2 M^2} + \frac{(\ell^2 mM + I_G(m+M))(p_1 p_2 - 2(p_1+p_2)\xi\omega_n + \omega_n^2)}{\ell M}$$

$$k_{14} = \frac{-1}{g\ell^2 M^2}(\ell^2 mM + I_G(m+M))((I_G+\ell^2 M)\omega_n(-2p_1 p_2 \xi + (p_1+p_2)\omega_n) + g\ell M(p_1+p_2-2\xi\omega_n))$$

Considering the numeric values of the coefficients, the settling time T_{set}, the percentage overshoot PO, and two fast negative poles, the coefficients are calculated:

$$T_{set} = 4s \qquad PO = 5\% \tag{4.32}$$

$$K_{11} = -453.713 \qquad k_{12} = -602.301 \qquad k_{13} = 5388.84 \qquad k_{14} = 1549.5$$

Where the dominant poles of the closed-loop are:

$$s_1 = -1 - 1.04869i \qquad s_2 = -1 + 1.04869i \tag{4.33}$$

The negative real poles with fast dynamics are:

$$s_3 = -4 \qquad s_4 = -8 \tag{4.34}$$

4.4.2.5 State feedback control using a Linear-Quadratic Regulator

The values for the feedback gain matrix K will be obtained from the **LQRegulatorGains** function in *Mathematica*.

In order to keep it simple, the relative weighting between state variables will not be considered, making $\alpha_i = 1$, and the penalty parameter between state and control variables will be considered as $\rho = 1$.

The maximum admissible values for the state and control variables selected are:

$$(x_1)_{max} = x_{max} = 0.00001\,m \qquad (x_2)_{max} = \dot{x}_{max} = 0.01\,m/s$$
$$(x_3)_{max} = \Theta_{max} = 0.001\,rad \qquad (x_4)_{max} = \dot{\Theta}_{max} = 0.01\,rad/s$$
$$(u_1)_{max} = F_{max} = 1\ N$$

The resulting weighting matrices **Q** and **R** are:

$$\mathbf{Q} = \begin{pmatrix} 100000 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 100 \end{pmatrix} \qquad \mathbf{R} = (1) \tag{4.35}$$

(4.36)

The coefficients obtained as the solution to the problem are:

$$K_{11} = -316.228 \qquad k_{12} = -435.351 \qquad k_{13} = 4009.89 \qquad k_{14} = 1148.47 \tag{4.37}$$

The resulting poles for the closed-loop system are:

$$s_1 = -3.98836 + 0.247558i \qquad s_2 = -3.98836 + 0.247558i \tag{4.38}$$
$$s_3 = -1.286264 + 1.13062i \qquad s_4 = -1.286264 + 1.13062i$$

The resulting dominant dynamics is similar to the one obtained using pole placement.

4.4.2.6 CoLink closed-loop control structures

Two different closed-loop control structures were implemented, one for the PID controllers and another one for the state feedback controllers. The single PID controller uses the same structure as the double PID controller with the parameters of the second controller set to zero. The state feedback controller designed with pole placement and using the linear-quadratic regulator approach share the same structure.

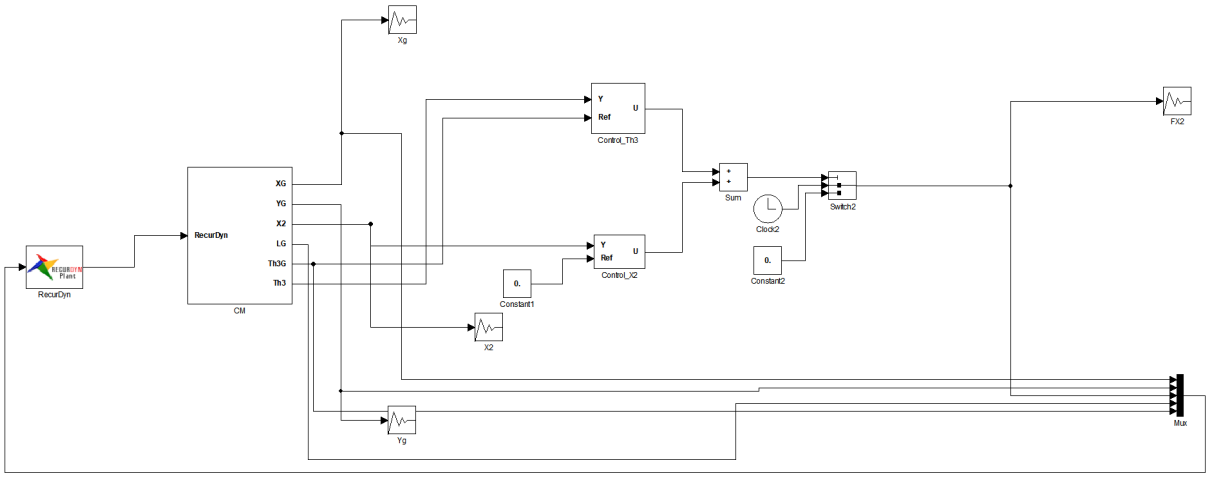


Figure 4.19: Structure of the single and double PID closed-loop control configuration in *CoLink*.

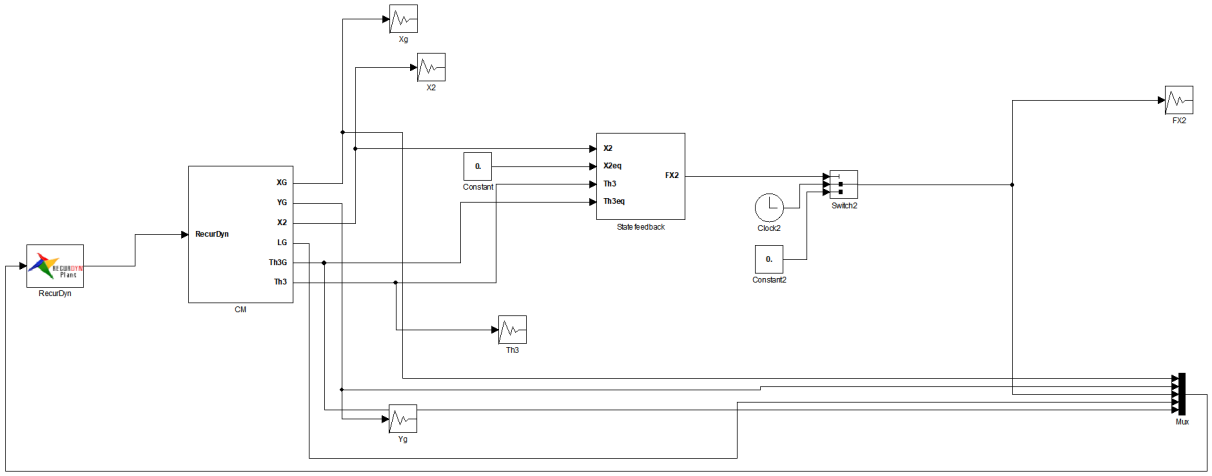


Figure 4.20: Structure of the state feedback (pole placement and LQR) control configuration in *CoLink*.

4.5 *SystemModeler*

4.5.1 Model

The complete model in *SystemModeler* is a closed-loop control structure with the *Handle* component and a control block, which can be either a PID control scheme or a state feedback configuration.

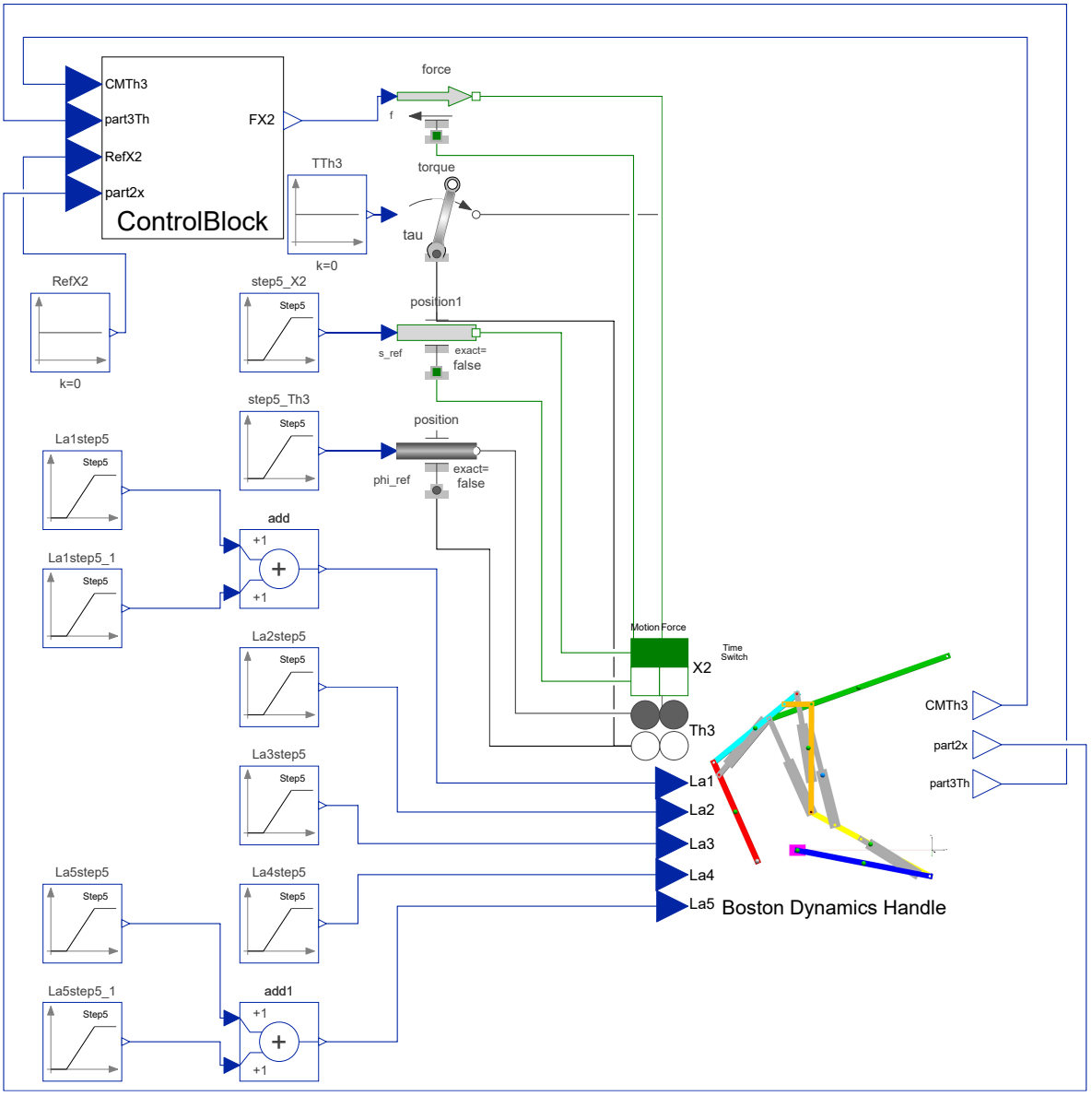


Figure 4.21: Complete model implemented in *SystemModeler*.

The validation of the model yields that the complete model is balanced, which is necessary to perform a simulation. When using a PID control block, the model is globally balanced with 10331 equations and 10331 variables, where 3012 of these are non-trivial equations. When using a state feedback control block, the model is globally balanced with 10319 equations and 10319 variables, where 3009 of these are non-trivial equations.

4.5.2 Controller comparison between SystemModeler and RecurDyn

The controllers calculated in the previous section are going to be simulated in both *RecurDyn* and *SystemModeler* and the outputs of the simulation for the state variables Θ_3, and X_2, the equilibrium angle $\Theta_{3_{eq}}$, and the translational force F will be compared between them.

As a first result, the period for the closed-loop system using only a proportional gain will be checked. The simulated results should be in agreement with the calculated values. In this case, the Ziegler-Nichols closed-loop method is not applicable for the system, but it can be used to check the resemblance of the mathematical model to the virtual one.

Once the first check is done, four different simulations will be performed: PID controller for pendulum angle only, double PID controller for pendulum angle and cart position, state feedback control using pole placement, and state feedback control using the LQR approach. The simulations consists of two stages. In the first stage, the position of the slider X_2 and the angle Θ_3 are driven using the *motion* connectors of the component. At the time determined by the *TimeSwitch* parameter the second stage begins, where the motion of the slider and the angle are dynamically changed using a translational force F obtained from the control block until the equilibrium point is reached. During both stages the actuator lengths are driven by STEP5 functions. This is the same configuration as the one used in *RecurDyn*, the results of the simulations between these platforms will be compared.

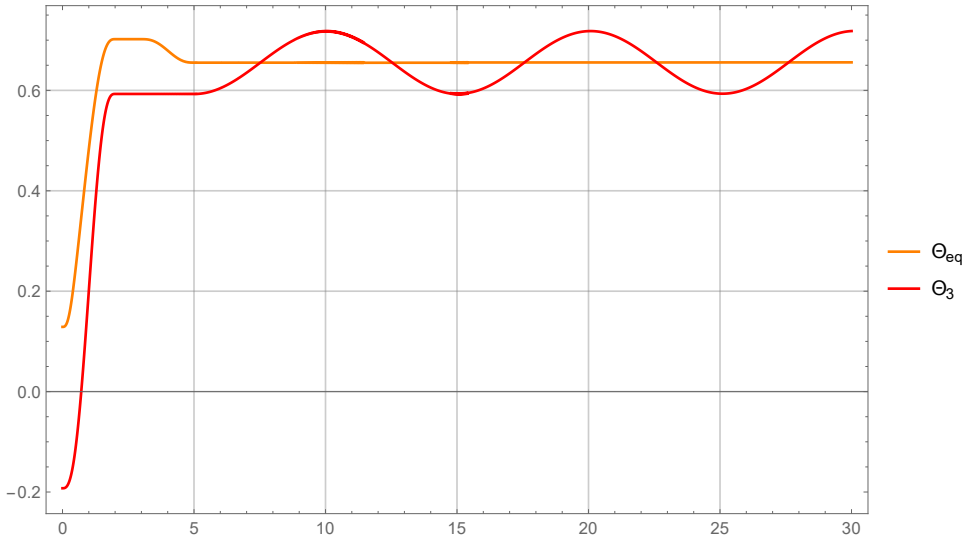
Simulation stage	**Starting time (s)**	**Ending time (s)**
Driven kinematic motion	0	5
Dynamic equilbrium settle	5	10
Modification of actuator length	10	15
Dynamic equilibrium settle	15	20
Modification of actuator length	20	25
Dynamic equilibrium settle	25	30

Table 4.1: Time table of the dynamic simulation stages with PID and state feedback control

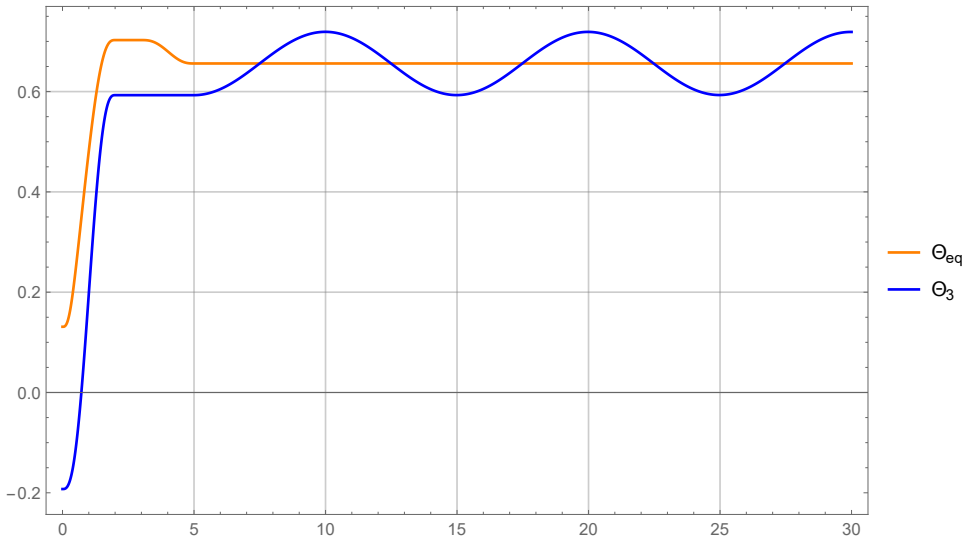
In terms of simulation settings, the solver used for *SystemModeler* was the default *Differential/Algebraic System Solver* (DASSL) with default settings (adaptive step size and a tolerance of 10^{-6}) and 2000 steps. The *RecurDyn* solver was the Implicit General Alpha (IMGALPHA) solver with no numerical damping and an error tolerance of 10^{-5} and 2000 steps. Both in *SystemModeler* and *RecurDyn*, the linearization parameter of the derivative block transfer function is set to $N = 0.01$ instead of the default value $N = 0.001$ in order to filter the high frequency noise of the numerical values of the system.

4.5.2.1 Ziegler-Nichols closed-loop method

For a proportional gain of $K_p = 1100$, the calculated value for the oscillation period is $T = 9.78\,s$.



(a) *RecurDyn* results.



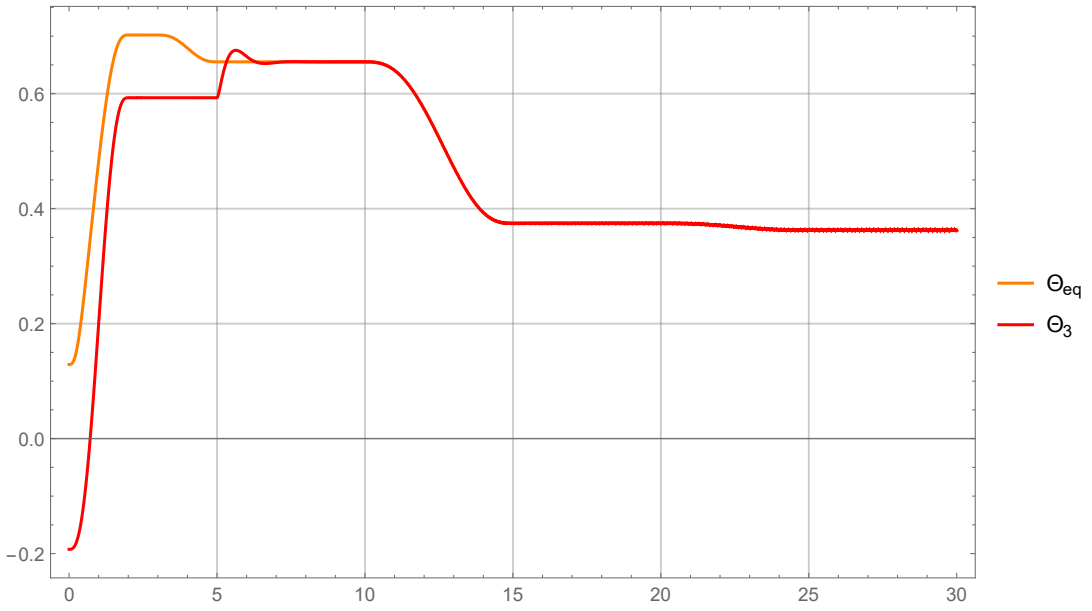
(b) *SystemModeler* results.

Figure 4.22: Oscillatory closed-loop response to proportional gain $K_p = 1100$.

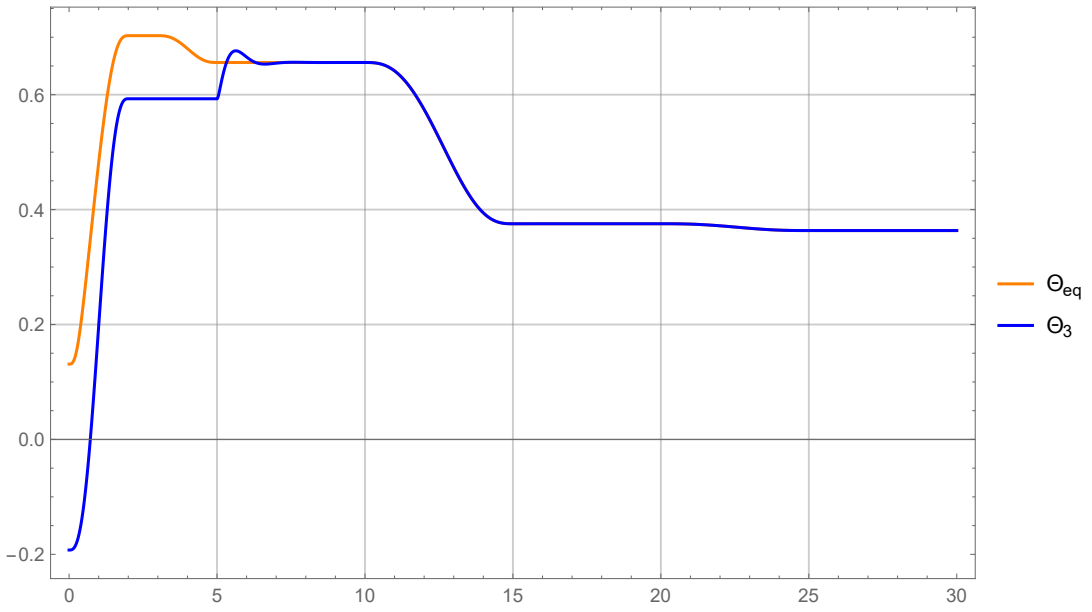
The simulation results in both softwares match. The oscillation period obtained from the graphs is $T = 10\,s$, which confirms that the linearized mathematical model is accurate to the virtual model. This verification gives the necessary support to develop the controller using the simplified linearized model.

4.5.2.2 PID controller for pendulum angle

A dynamic simulation using a single PID controller for the pendulum angle was performed in *RecurDyn* and *SystemModeler*. The behavior of the relevant variables were extracted and plotted using *Mathematica*.

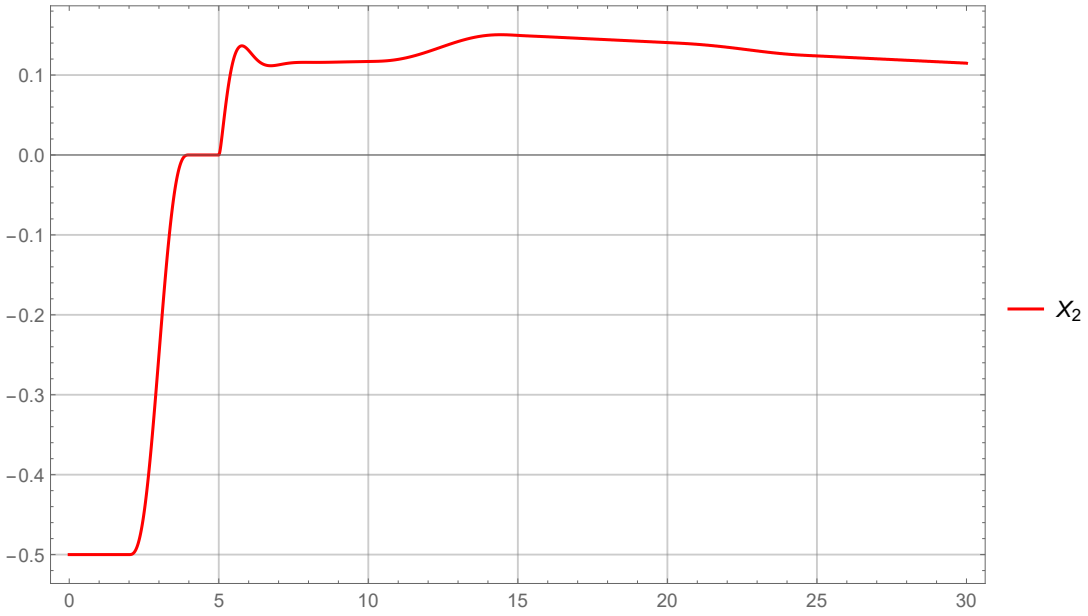


(a) *RecurDyn* results.

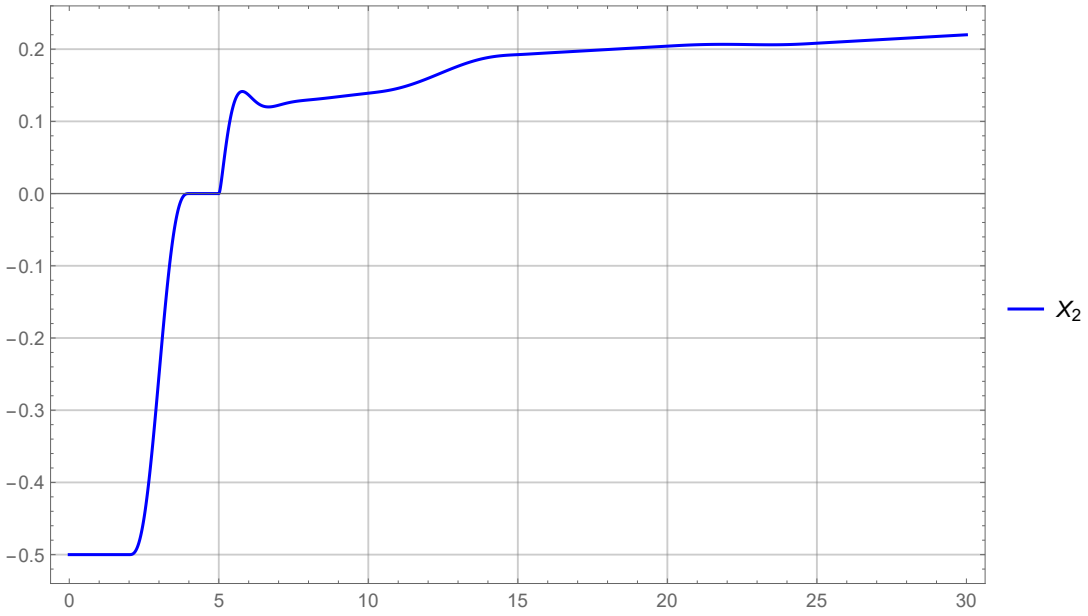


(b) *SystemModeler* results.

Figure 4.23: Θ_3 response for the closed-loop control with one PID controller for the pendulum angle.

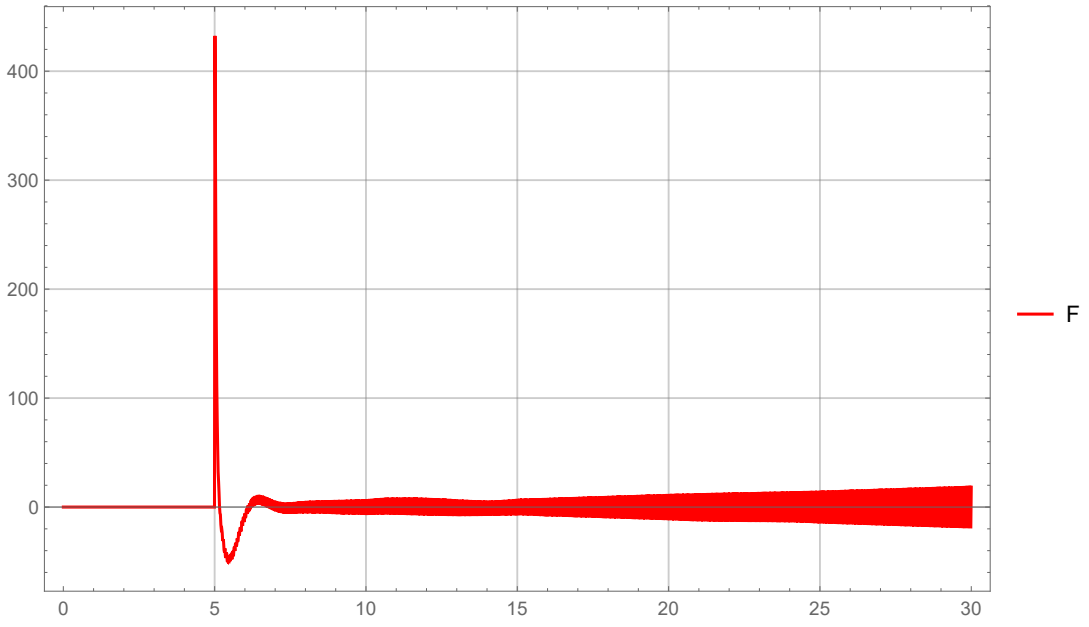


(a) *RecurDyn* results.

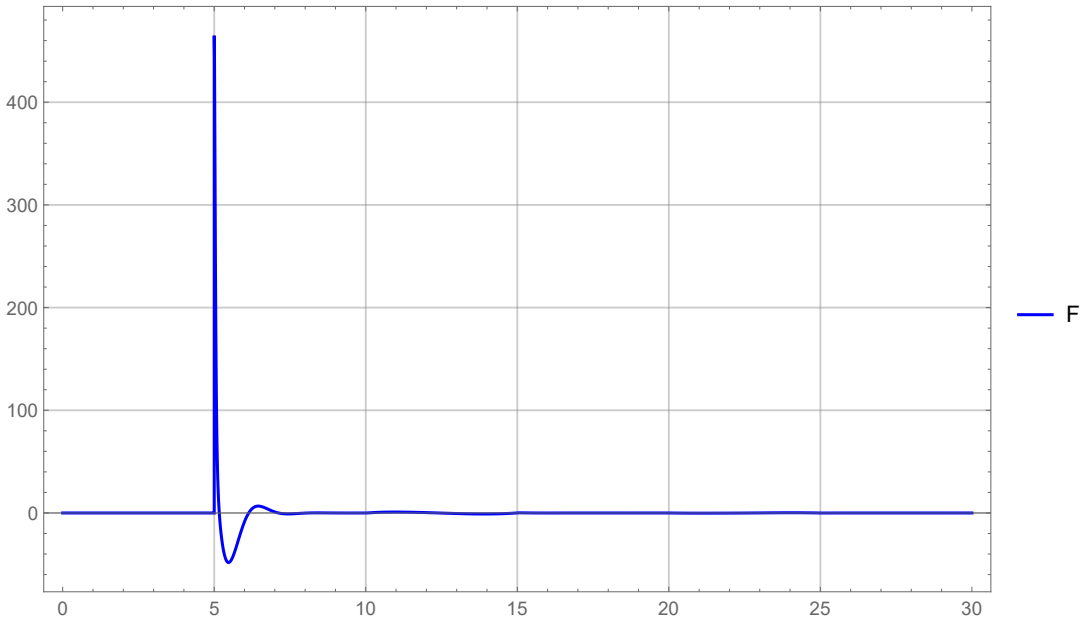


(b) *SystemModeler* results.

Figure 4.24: X_2 response for the closed-loop control with one PID controller for the pendulum angle.



(a) *RecurDyn* results.

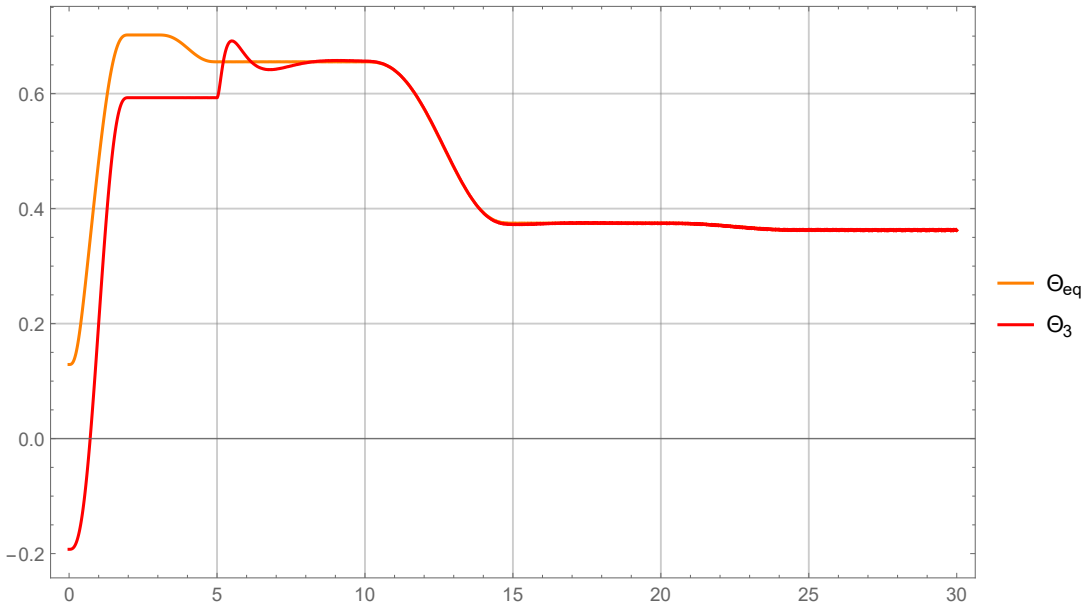


(b) *SystemModeler* results.

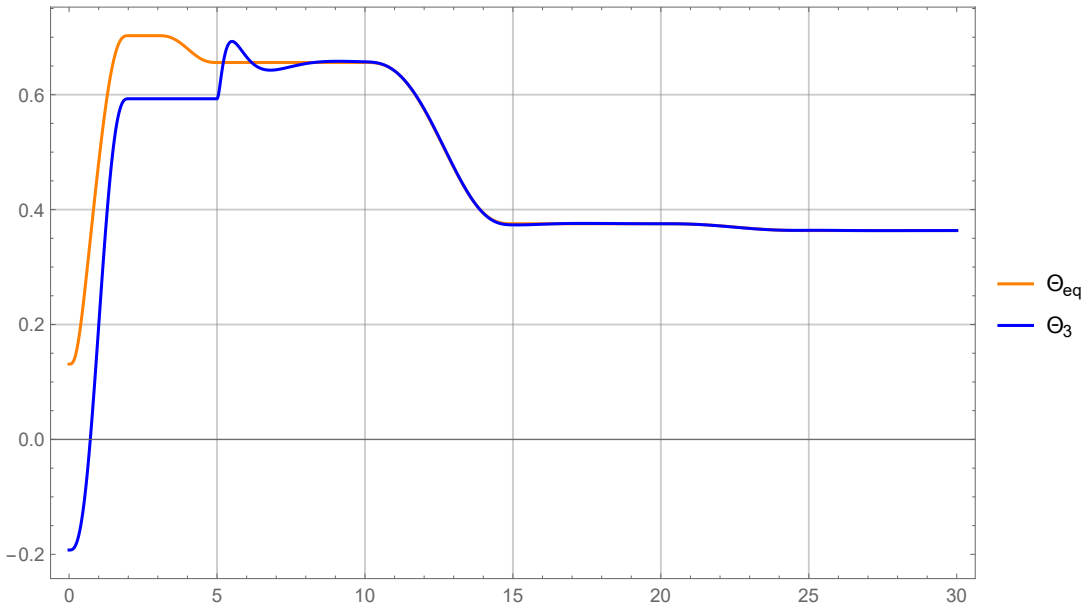
Figure 4.25: F action for the closed-loop control with one PID controller for the pendulum angle.

4.5.2.3 Double PID controller for pendulum angle and cart position

A dynamic simulation using a double PID controller for the pendulum angle and cart position was performed in *RecurDyn* and *SystemModeler*. The behavior of the relevant variables were extracted and plotted using *Mathematica*.

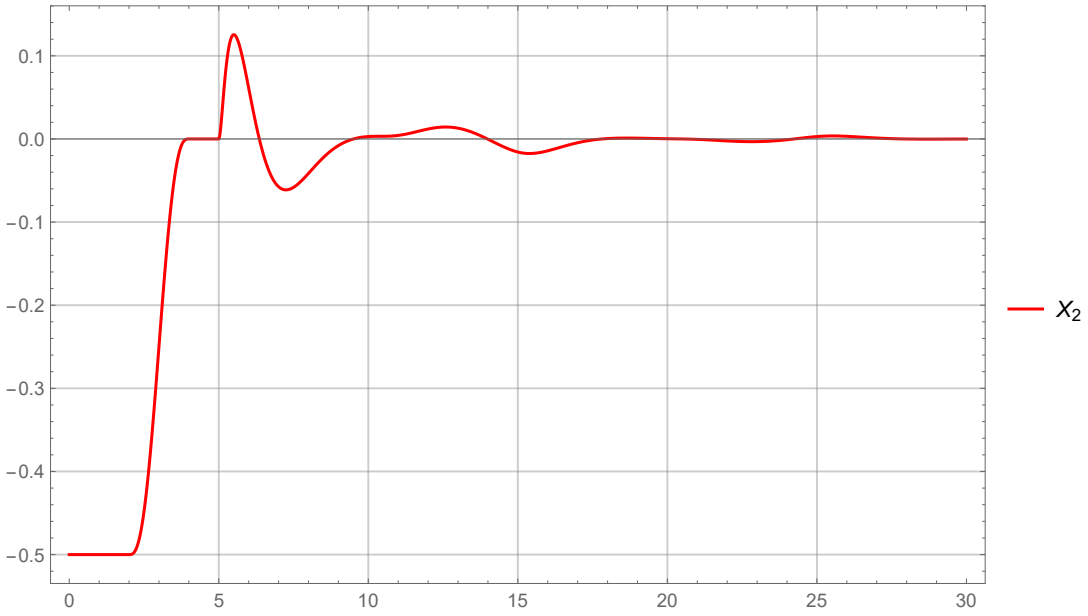


(a) *RecurDyn* results.

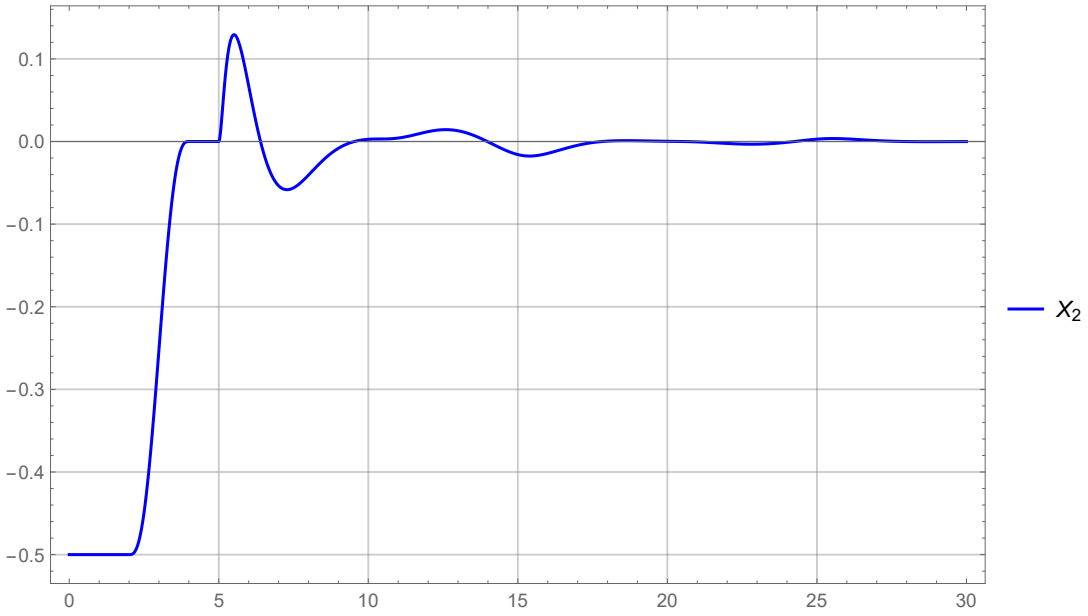


(b) *SystemModeler* results.

Figure 4.26: Θ_3 response for the closed-loop control with double PID controller for pendulum angle and cart position.

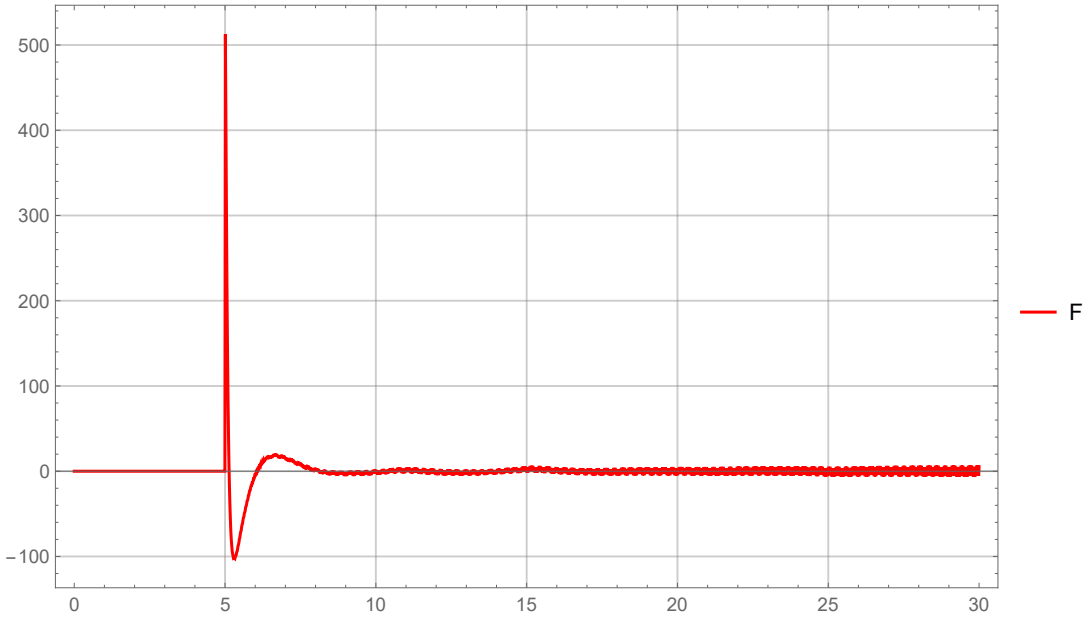


(a) *RecurDyn* results.

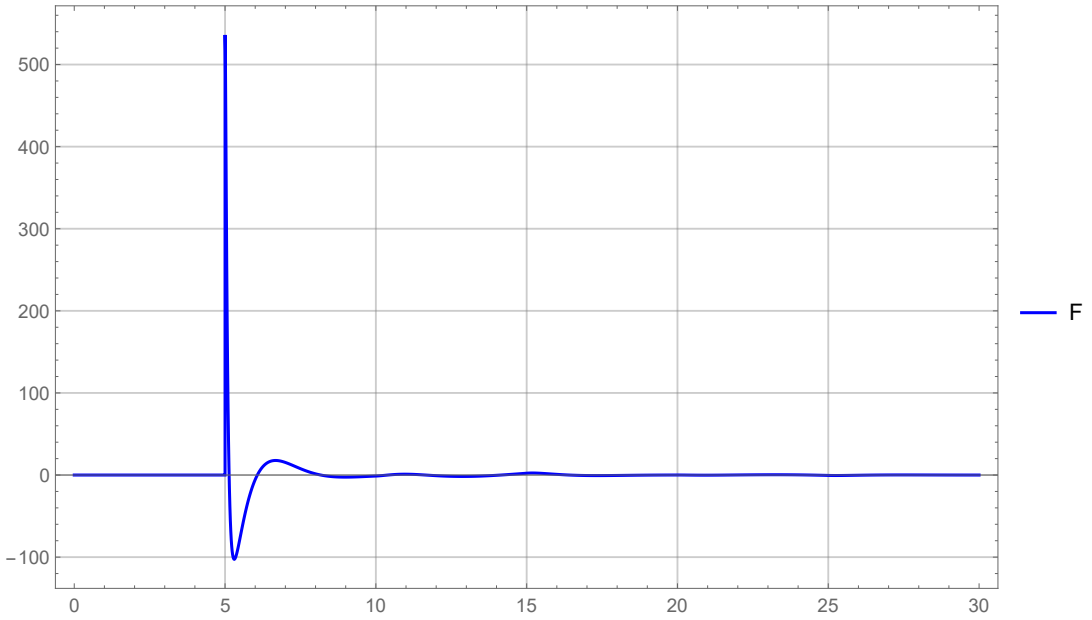


(b) *SystemModeler* results.

Figure 4.27: X_2 response for the closed-loop control with double PID controller for pendulum angle and cart position.



(a) *RecurDyn* results.

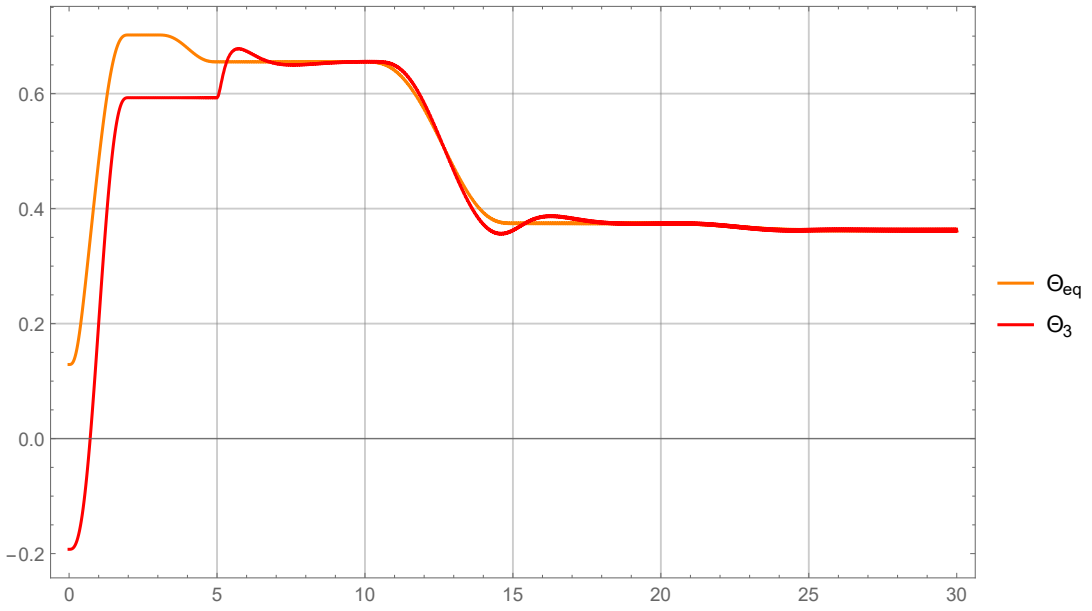


(b) *SystemModeler* results.

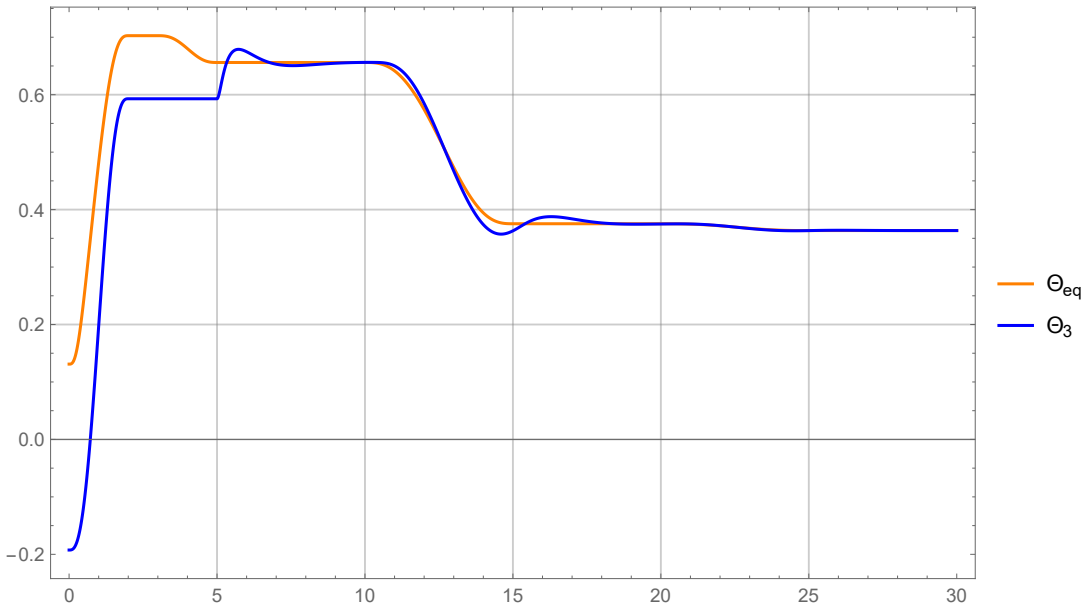
Figure 4.28: F action for the closed-loop control with double PID controller for pendulum angle and cart position.

4.5.2.4 State feedback control using pole placement

A state feedback control was designed using pole placement and the corresponding dynamic simulation was performed in *RecurDyn* and *SystemModeler*. The behavior of the relevant variables were extracted and plotted using *Mathematica*.

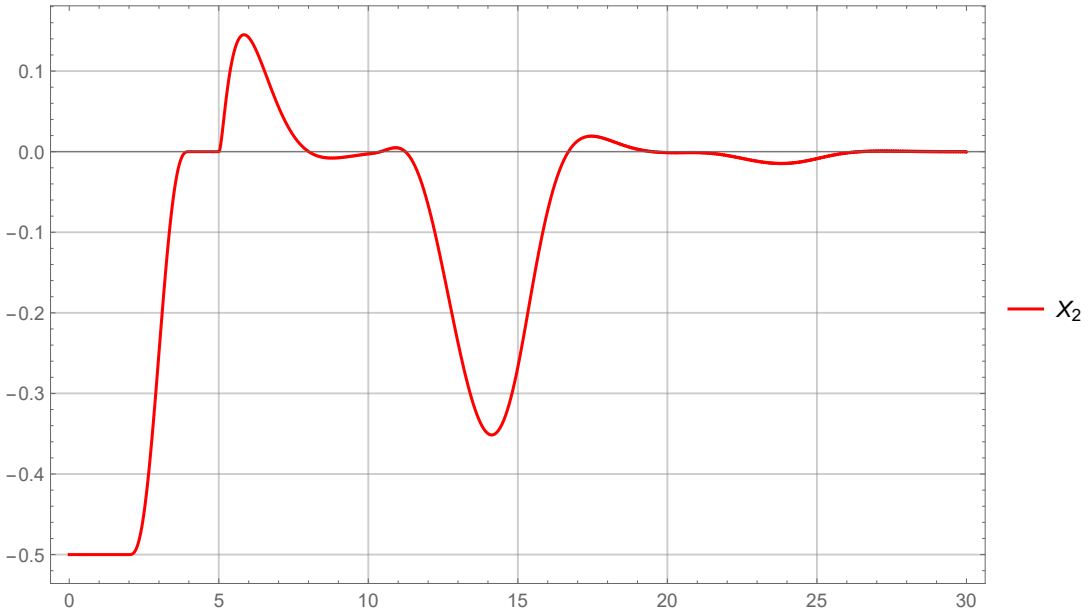


(a) *RecurDyn* results.

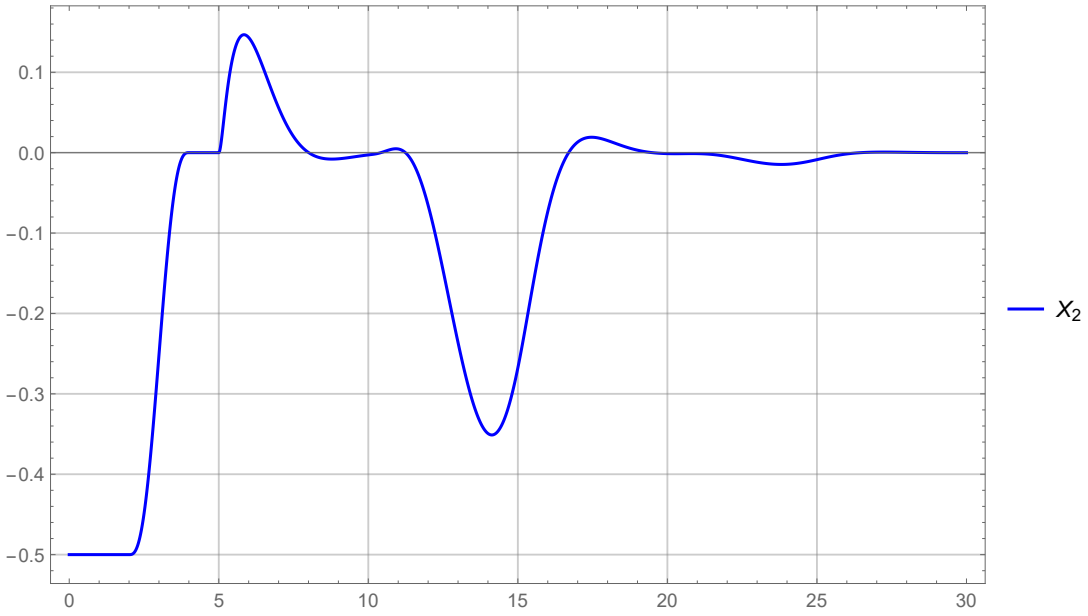


(b) *SystemModeler* results.

Figure 4.29: Θ_3 response for the closed-loop control with state feedback control designed using pole placement.

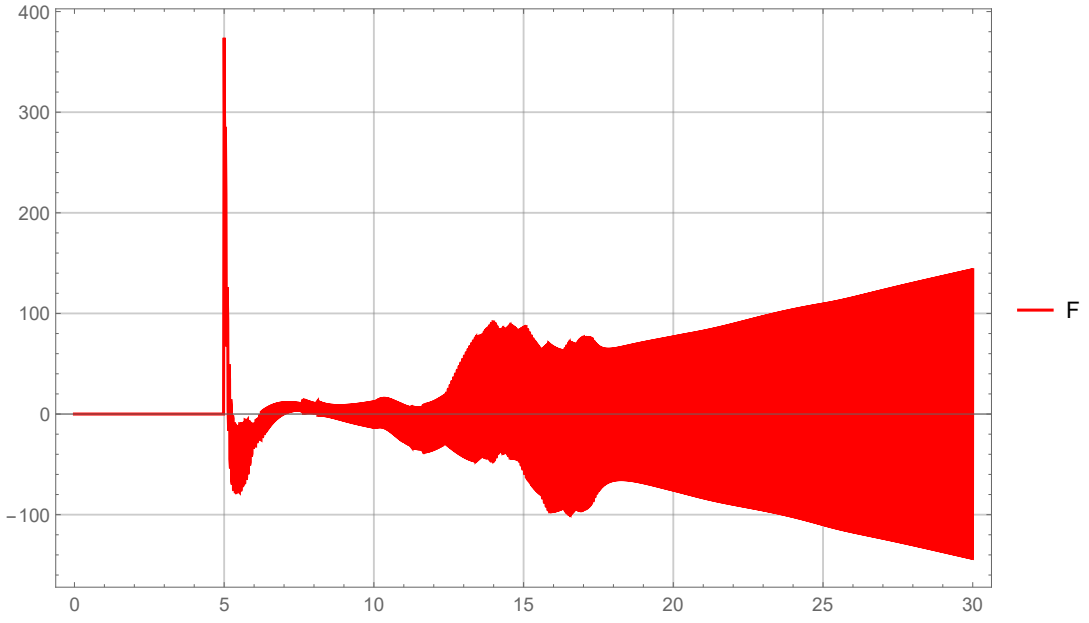


(a) *RecurDyn* results.

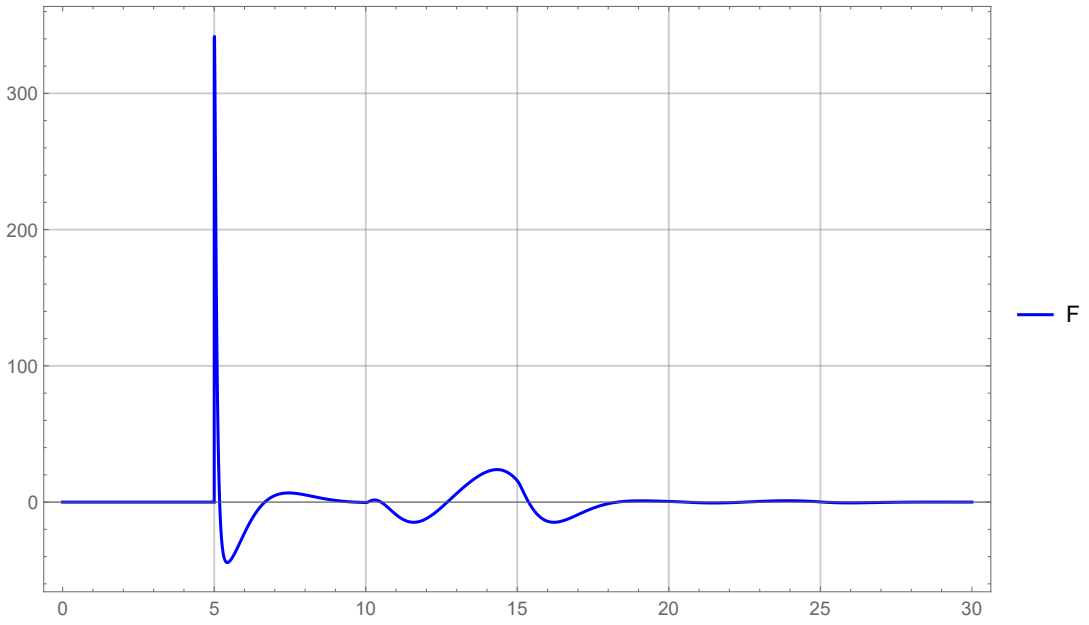


(b) *SystemModeler* results.

Figure 4.30: X_2 response for the closed-loop control with state feedback control designed using pole placement.



(a) *RecurDyn* results.

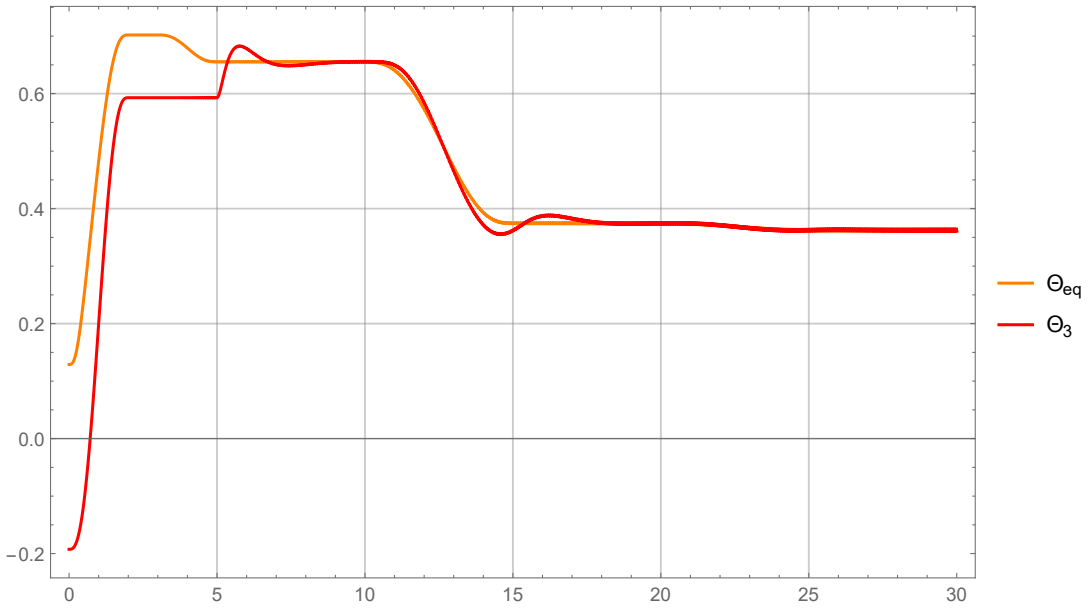


(b) *SystemModeler* results.

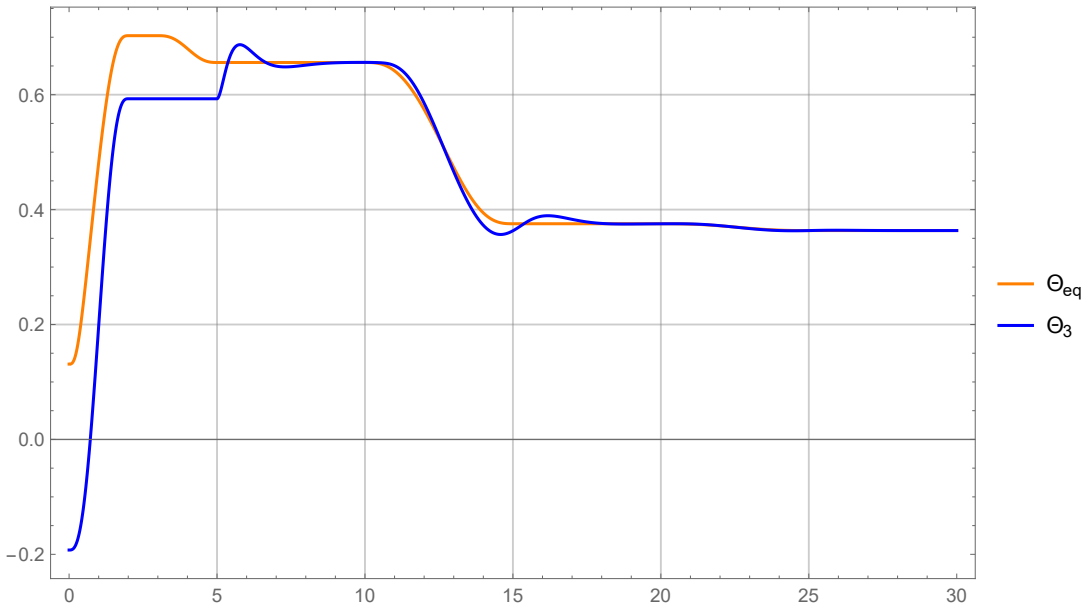
Figure 4.31: F action for the closed-loop control with state feedback control designed using pole placement.

4.5.2.5 State feedback control using a Linear-Quadratic Regulator

A state feedback control was designed using linear-quadratic regulator theory and the corresponding dynamic simulation was performed in *RecurDyn* and *SystemModeler*. The behavior of the relevant variables were extracted and plotted using *Mathematica*.

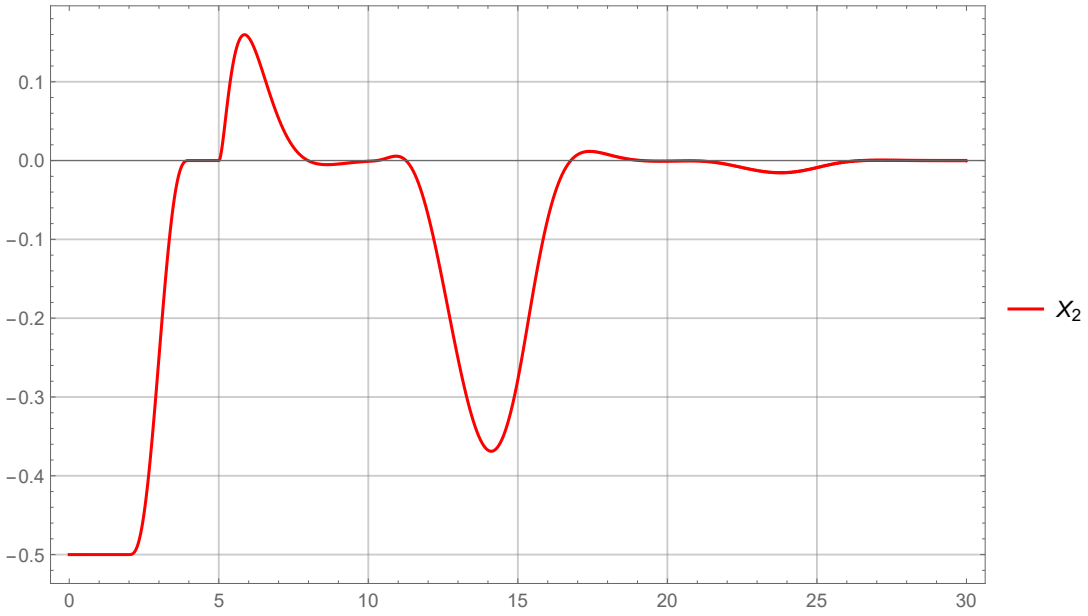


(a) *RecurDyn* results.

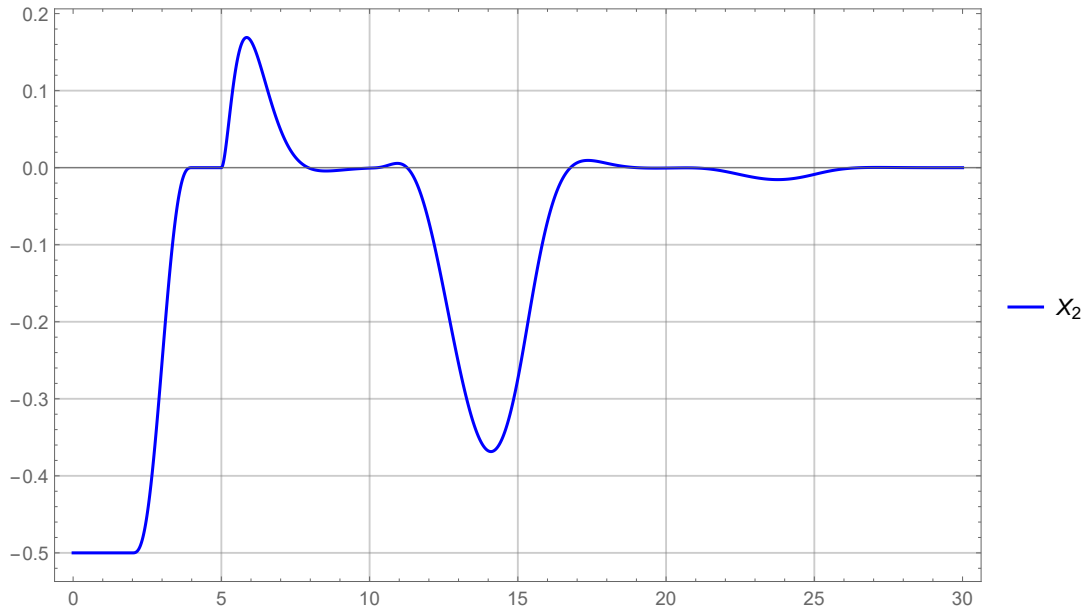


(b) *SystemModeler* results.

Figure 4.32: Θ_3 response for the closed-loop control with state feedback control designed using LQR.

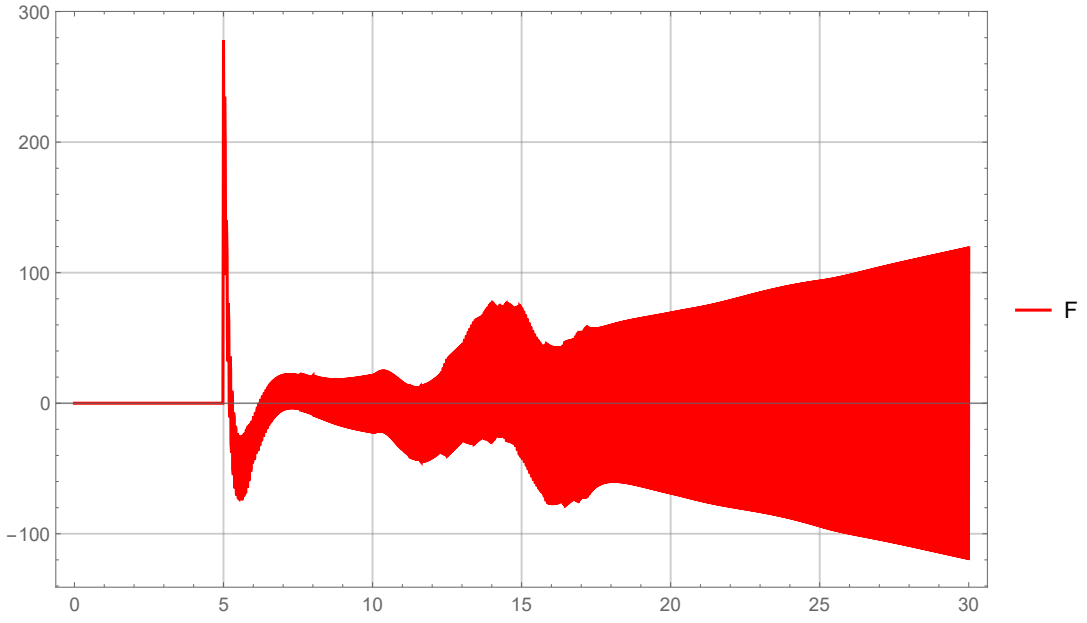


(a) *RecurDyn* results.

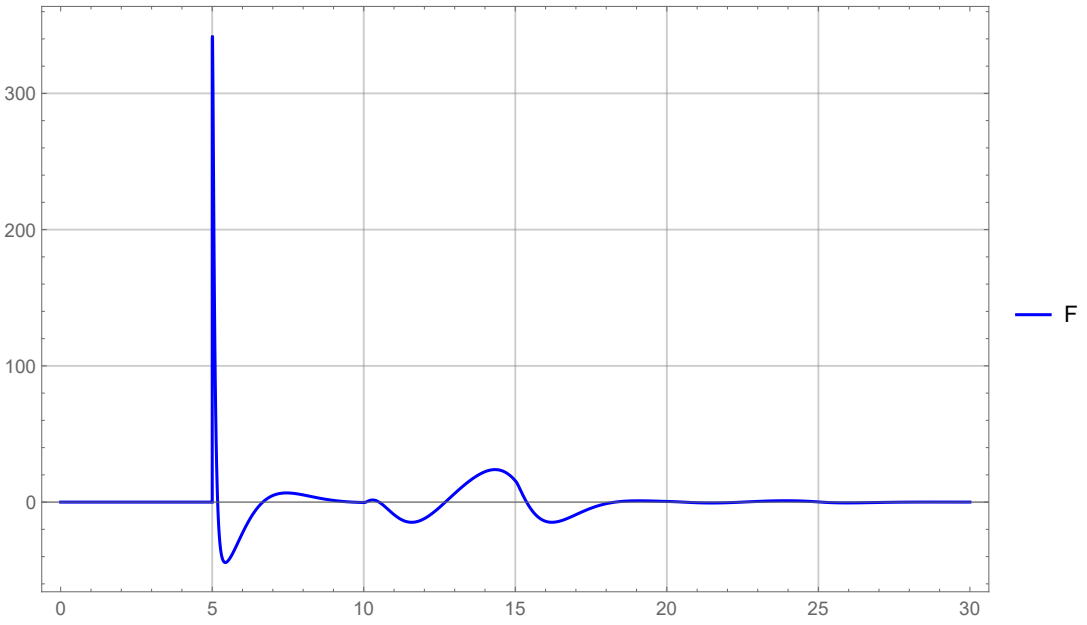


(b) *SystemModeler* results.

Figure 4.33: X_2 response for the closed-loop control with state feedback control designed using LQR.



(a) *RecurDyn* results.



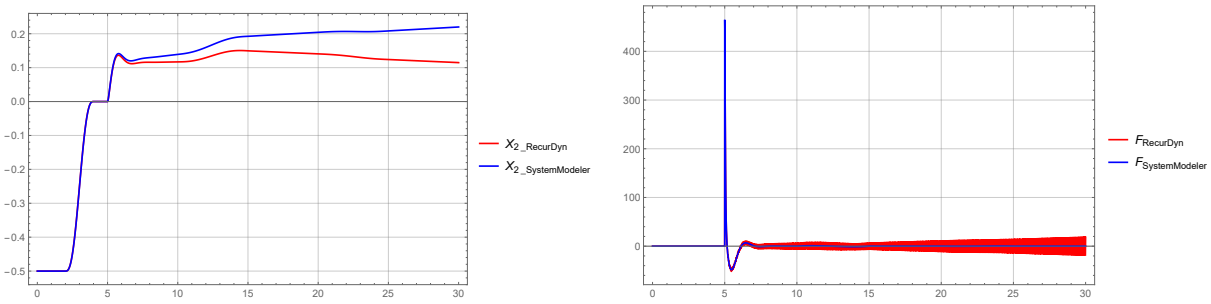
(b) *SystemModeler* results.

Figure 4.34: F action for the closed-loop control with state feedback control designed using LQR.

4.5.2.6 Discussion

The outputs of the simulations performed in *RecurDyn* and *SystemModeler* are very similar. In general terms, both models are equivalent because they yield the same results. However, there are some differences that must be noted.

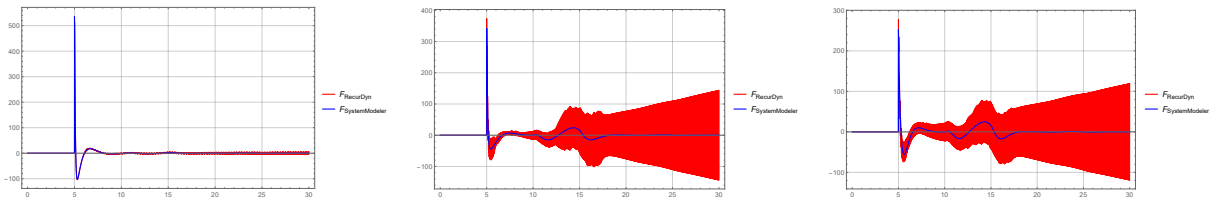
Even though the error tolerance of *RecurDyn* has been reduced significantly from the default value, the output of the variables have a considerable amount of noise compared to the results obtained in *SystemModeler*. This noise is not very noticeable in the value of the Θ_3 angle or the X_2 position, but it can be easily appreciated in the value of the control action F. In order to mitigate this effect, the error tolerance was reduced even further, but this was not successful. In many occasions the simulation failed during the first seconds of driving motions or during the actuator changes in the last part of the simulation. The best results were obtained with an error tolerance of 10^{-5}. The number of steps were also increased to try to improve the results, but it had no effect on noise reduction.



(a) Slider position X_2. **(b)** Control action F.

Figure 4.35: Comparison of slider position X_2 and control action F noise between *RecurDyn* and *Mathematica* for the single PID configuration.

In the configuration where only there is only a PID controller for the Θ_3 angle and the X_2 position is left uncontrolled, the effect of the noise in the simulation causes the result for the position of the slider in *RecurDyn* to have a different trend than that of *SystemModeler*.



(a) Double PID control scheme. **(b)** State feedback using pole position. **(c)** State feedback using LQR.

Figure 4.36: Comparison of control action F noise between *RecurDyn* and *Mathematica*.

As one can notice from the comparisons of the control actions, the result obtained from *RecurDyn* has a high frequency component that oscillates around the value obtained from the *SystemModeler* simulation. The expected final value of the translational force F to maintain the balance in the ideal conditions depicted in this model once the transient response is over should be 0, and this is what the simulation results in *SystemModeler* show.

The robustness of the different control schemes is something that can be observed from the results. The double PID controller and both of the state feedback controls are designed to have the same dominant dynamics (the dominant poles of the systems are practically the same), but their results are different. The results for the state variables of the system in the *SystemModeler* simulations of these controllers are going to be compared in the following figures.

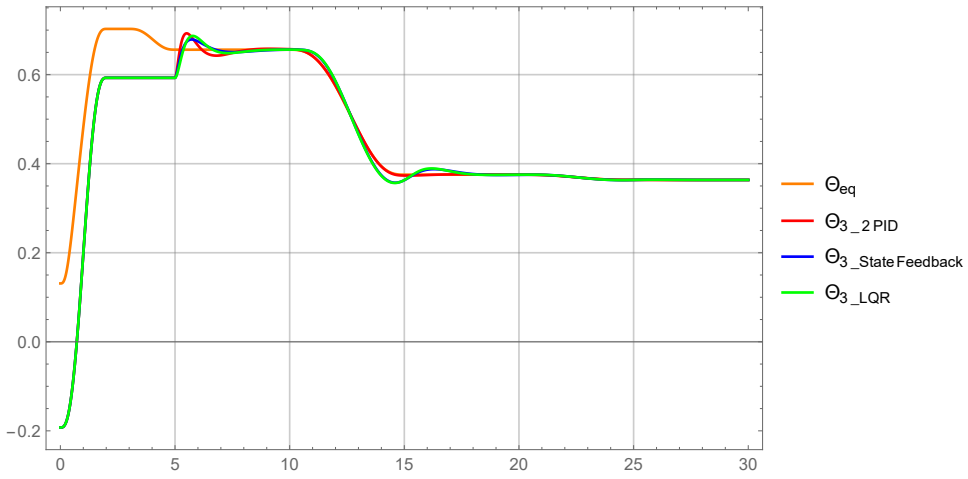


Figure 4.37: Comparison for the results of the Θ_3 angle with double PID, state feedback with pole placement and state feedback LQR controllers.

For the Θ_3 angle, the initial overshoot and settling time to reach the equilibrium angle are inside the specifications. The reference tracking for the angle is quite acceptable when changing the actuator lengths and therefore the inertial parameters of the system.

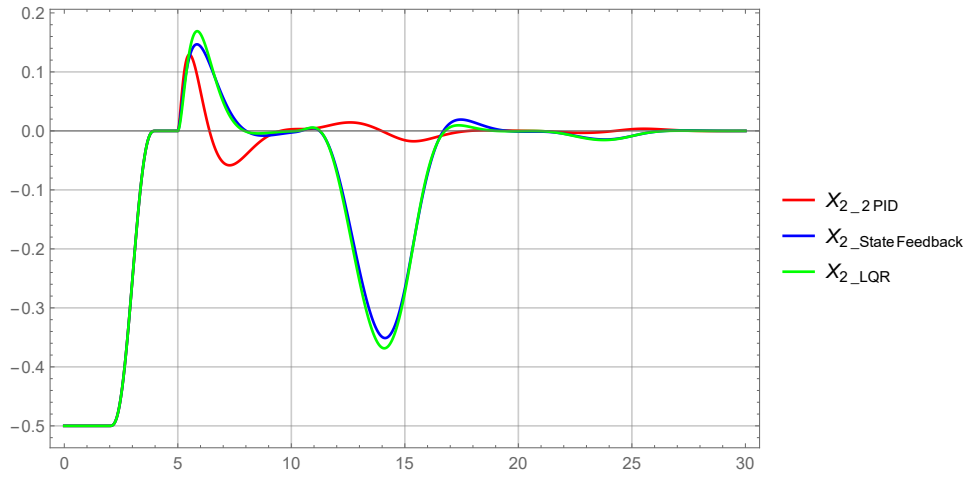


Figure 4.38: Comparison for the results of the X_2 position with double PID, state feedback with pole placement and state feedback LQR controllers.

The problem comes with the X_2 slider position, the initial overshoot is considerable but things get worse with the parameter change for the state feedback controllers. When the first actuator length is modified the overshoot is very considerable for the state feedback controllers, while its magnitude is much more reduced in the case of the double PID controller. A similar thing happens with the second actuator length change, but to a less extent.

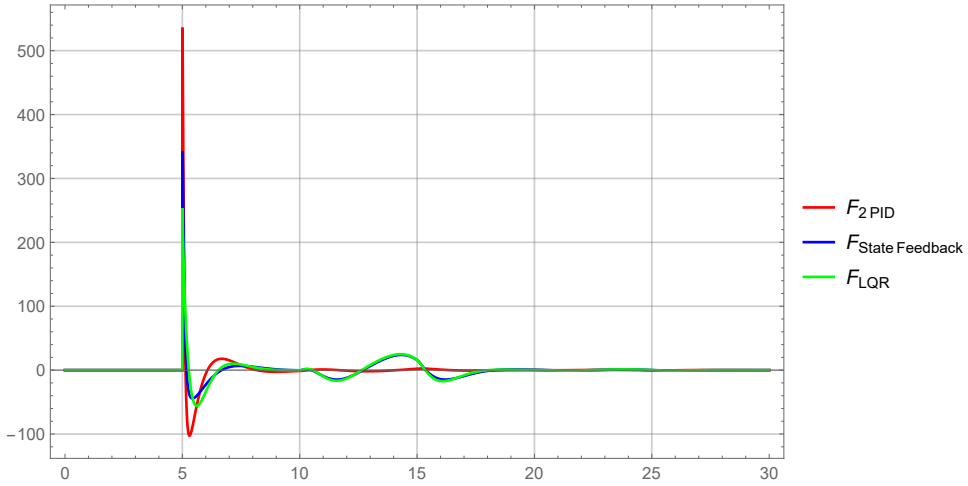


Figure 4.39: Comparison for the results of the F control action with double PID, state feedback with pole placement and state feedback LQR controllers.

As expected, the lower effort in control action F is obtained with the LQR controller, and the highest effort is obtained with the double PID controller.

The parameters of the model are m, M, g, ℓ, I_G and Θ_{eq}. While the mass and the gravity acceleration are invariants, with the change in actuator lengths the other three parameters have significantly changed their value:

$$\begin{aligned} La_1 = 0.282722 &\rightarrow 0.262722\, m \\ La_5 = 0.268034 &\rightarrow 0.278034\, m \\ \ell = 76.2733 \cdot 10^{-2} &\rightarrow 59.9480 \cdot 10^{-2}\, m \\ I_G = 12.6727 &\rightarrow 11.8332\, kg \cdot m^2 \\ \Theta_{eq} = 37.5884 &\rightarrow 20.8285^\circ \end{aligned} \tag{4.39}$$

These changes in parameters modify the dynamics of the system, which causes the controllers to have different behavior than expected. If the dynamic system is modified, the closed-loop results obtained with the controllers are different that those they were designed for. Although the impact they had has not been significant enough to make the system unstable, these modifications have changed the location of the poles of the system. The resulting poles of the closed-loop systems due to the parameter change are presented for each of the controller configurations.

For the closed-loop system with the single PID controller, the resulting pole modification is:

$$\begin{aligned} s_1 = -2 - 3.31196i &\rightarrow s_1 = -2.00186 - 3.29582i \\ s_2 = -2 + 3.31196i &\rightarrow s_2 = -2.00186 + 3.29582i \\ s_3 = -20 &\rightarrow s_3 = -20.8476 \end{aligned} \tag{4.40}$$

For the closed-loop system with the double PID controller, the resulting pole modification is:

$$\begin{aligned} s_1 = -1 - 1.04869i \rightarrow s_1 = -0.86203 - 1.07984i \\ s_2 = -1 + 1.04869i \rightarrow s_2 = -0.86203 + 1.07984i \\ s_3 = -2 \rightarrow s_3 = -1.44765 \\ s_4 = -4 \rightarrow s_4 = -5.88235 - 3.96815i \\ s_5 = -8 \rightarrow s_5 = -5.88235 + 3.96815i \end{aligned} \tag{4.41}$$

For the closed-loop system with state feedback obtained through pole placement, the resulting pole modification is:

$$\begin{aligned} s_1 = -1 - 1.04869i \rightarrow s_1 = -0.97791 - 1.09746i \\ s_2 = -1 + 1.04869i \rightarrow s_2 = -0.97791 + 1.09746i \\ s_3 = -4 \rightarrow s_3 = -3.05343 \\ s_4 = -8 \rightarrow s_4 = -10.54550 \end{aligned} \tag{4.42}$$

For the closed-loop system with state feedback obtained through LQR, the resulting pole modification is:

$$\begin{aligned} s_1 = -1.28626 - 1.13061i \rightarrow s_1 = -1.21121 - 1.26002i \\ s_2 = -1.28626 + 1.13061i \rightarrow s_2 = -1.21121 + 1.26002i \\ s_3 = -3.98836 - 0.247557 \rightarrow s_3 = -2.26877 \\ s_4 = -3.98836 + 0.247557 \rightarrow s_4 = -6.99710 \end{aligned} \tag{4.43}$$

Even though these changes in the dynamics of the system might not seem very considerable, this effect is very significant in state feedback controllers, while the double PID controller is able to follow the changing reference value very precisely. The double PID controller is able to track the reference angle change Θ_{eq} very precisely, with very little control effort causing a very small effect on the slider X_2 position. State feedback controllers have a different dynamic response to the change in equilibrium angle Θ_{eq} caused by the parameter change causing an overshoot in the Θ_3 angle response, while in the slider X_2 position this overshoot is amplified causing a very large oscillation.

Chapter 5

Discussion and conclusion

5.1 Discussion

5.1.1 Model simplification

The objective of this project was to obtain a simplified prototype of the *Handle* robot developed by *Boston Dynamics*. All the information in which the model is based on was obtained from the analysis of the original video available at the *Boston Dynamics YouTube* channel. The complexity of the real robot makes this an immeasurable task considering that very limited internal knowledge of its design and operation is known. The real 3-dimensional model was simplified to a 2-dimensional model with seven degrees of freedom, where most of the drivers are linear actuators and the wheels have been replaced for a slider.

From careful analysis of the video footage, a first kinematic diagram of the simplified model was obtained. It must be noted that the model was simplified further and a later diagram was made. With the mechanical configuration of the system identified, the first task was obtaining a virtual model using *SolidWorks*. The model consisted of eight different bodies: ground, wheel (slider), lower leg, upper leg, trunk, head, upper arm and lower arm. The linear actuators were not really considered as bodies, as they were modeled as a visual representation of a constraint. All the bodies were defined by setting the position of their relevant points, i.e. their center of mass and the points where they connected to other bodies or linear actuators, in terms of their geometrical dimensions. The tracer point was located at the end of the lower arm.

Using the *COSMOSMotion* add-in for *SolidWorks*, the definition of the joints and linear actuators was made in order to obtain a self-aligning model with no redundant constraints. The absence of redundant constraints in systems modeled in computer aided engineering softwares is a requisite in order to make sure that the simulations performed with the model will yield valid results.

5.1.2 Kinematic model

A complete mathematical model was developed in *Mathematica*, where the symbolic expressions for the position and orientation were obtained for the eight bodies that make up the model. The mechanical system definition was done using the *MechanicalSystems* package, which makes use of the constraint equations formulation for the description of mechanical systems. The bodies that make up the model were defined in terms of their geometric dimensions, inertial properties, and initial values of their generalized coordinates. The joints that connect the bodies together were defined and those that make up the degrees of freedom were expressed as functions of time. All the expressions for the generalized coordinates of the bodies were obtained in terms of the system's degrees of freedom by symbolically solving the constraint equations derived from the joint definitions. These expressions were derived twice to obtain the expressions for the generalized velocities and accelerations of the bodies in terms of the degrees of freedom and their derivatives. The center of mass expression of a group of bodies and their moment of inertia were also obtained in terms of the system's degrees of freedom.

The mathematical formulation of the variable-sided triangles defined by the linear actuator configuration was done using relative angle constraints instead of relative distance constraints, which resulted in a rather considerable reduction in expression complexity. The relative angle constraint formulation required of the cosine theorem to express the relative angle between bodies in terms of the actuator length that connects both of them together. This was a very interesting and unexpected result related to this type of basic mechanism that allowed a significant performance increase and complexity reduction over the prior approaches made in the definition of variable-sided triangles.

The kinematic solution to the mechanical system has a considerable relevance for the further developments. The position of every point in the model can be obtained for any moment in time for a defined output to the degrees of freedom. A kinematic simulation was performed in *COSMOSMotion* in which the position of the slider, the angle of the lower leg body, and the actuator lengths were driven from the starting point to a certain value. The motion was defined using STEP5 class C^2 functions and the results for the tracer point were compared to the results obtained from applying the same motion definition to the *Mathematica* kinematic model. The results for the tracer point motion of both softwares matched for position, velocity, and acceleration. The kinematic expressions of the tracer point coordinates, along with the relative angle formulation for the linear actuators allowed obtaining the area where the point can be located, what is known as the working range of the model.

The most important outcome of the kinematic model was the interactive visual representation that was developed in *Mathematica*. This graphical representation shows the resulting configuration of the model for any possible value of the model's degrees of freedom, with the possibility of overlapping the contour of the working range. The reduced complexity of the kinematic equations obtained allows for the model to update in real time while the degrees of freedom are changed using the provided slider widgets.

5.1.3 Dynamic model

The dynamic model of the robot was reduced to an inverted pendulum on cart model. All the upper bodies were the linear actuators are attached, bodies 3 through 8, were considered a single body with its inertial properties determined by the lengths of the actuators. This body, which is considered the pendulum, is attached to the slider body, which is the equivalent to the cart. The actuator lengths are driven degrees of freedom, while the cart position and the angle of the pendulum will be dynamically controlled using a translational force. The simplified model had two degrees of freedom, and the equations of motion for this model were obtained using three different ways that yielded the same results: using Lagrangian mechanics, applying Newton's Laws of motion and using constraint equations. An ideal model was considered, with no damping terms in neither the linear nor the angular position equations.

The resulting equations of motion were nonlinear and a linearization around an operating point was developed. The operating point selected for the model was the equilibrium angle of the pendulum, in which the center of mass of the pendulum is in the direct vertical from the revolute joint that joins the pendulum and the cart together. A dynamic control for this resulting model was designed, using a translational force applied to the cart as the control action to maintain the balance at equilibrium point. The model has two groups of parameters: the first group consisting on invariant terms, which were cart mass, pendulum mass, and gravitational acceleration; and the first group dependent on the actuator lengths, consisting in length to the center of mass, moment of inertia of the pendulum, and the equilibrium angle.

A open-loop dynamic simulation was performed in *Mathematica* and *RecurDyn*, yielding the same results in both of the softwares. This dynamic simulation consisted in a free motion simulation where the model was left to swing under the gravitational field and the tracer point trajectory was obtained and compared.

5.1.4 Control

Using the linearized mathematical model, two different closed-loop control schemes were implemented: a PID controller and a state feedback controller.

First, the heuristic Ziegler-Nichols closed-loop method was tried to obtain a PID controller for the angle position, but due to the dynamic characteristics of the model (with no damping whatsoever), it resulted to be unsuccessful. However, the period of oscillation of the closed-loop system when a proportional term is applied was compared to the expected mathematical result and it was close enough to consider the linearized model a valid approximation. Two different schemes of PID control loops were implemented: a single PID control for the pendulum angle and a double PID control for pendulum angle and cart position. The design method used was pole placement for both of them. For the state feedback controller, two different design methods were tried: pole placement and LQR. The dominating dynamics for the double PID controller and both of the state feedback controllers was selected to be the same.

These closed-loop control configurations were simulated on two multibody dynamics capable softwares, *RecurDyn* and *SystemModeler*, in order to perform a comparison between their results. The dynamic simulations consisted of two different stages. First, a kinematic driven motion to a certain point away from the equilibrium is performed and the dynamic control is enabled

to bring the model to equilibrium. The second phase consisted in changing some actuator lengths with the impact it has on the model, where the dynamics is modified and the closed-loop configuration changes. Both softwares yielded the same results overall, but there were some slight differences. The control action resulting from *RecurDyn* had high frequency noise as opposed to the *SystemModeler* results. The robustness of the PID controller versus the state feedback controller were also compared. The model change and equilibrium point modification resulting from the actuator lengths had a big impact in the state feedback controllers, resulting in poor reference tracking and large overshoots. On the other hand, PID controllers performed much better, due to resulting higher order closed-loop dynamics, resulting in controls that were more robust to model and reference modification than state feedback controllers.

5.2 Future work

5.2.1 Model complexity increase

The model has been significantly simplified in order to reduce the magnitude of the problem to tackle. Once the 2-dimensional kinematic model has been solved it could be possible to propose a 3-dimensional kinematic model. This model should have two arms and two legs that could move independent to each other. The *MechanicalSystems* package for *Mathematica* was used in its 2-dimensional formulation, but it could be used to solve the 3-dimensional problem using the *Modeler3D* option.

The complexity of the dynamic system could be increased in the current 2-dimensional formulation by including damping terms in the slider position and pendulum angle. Dynamic models with more degrees of freedom (including linear actuators as not constant values) could also be proposed. If the 3-dimensional model is developed, a 3-dimensional dynamic model should also be developed with the increased difficulty it supposes.

5.2.2 Control structures

The current control loops designed can be improved by considering other choices for the dominant dynamics of the closed loops and comparing their performance. A fine tuning procedure could be performed by trying different values for the selected poles and even mixing PID and state feedback controllers to improve the response of the slider position. Reference tracking improvement in the state feedback controllers could be achieved using by changing the dominant dynamics of the system and performing a slower reference change. Another option is to implement several controllers in terms of the parameters of the dynamic system and switch between them. New control structures could also be implemented, by using *state observers* or *Kalman filters*, to design a better control with a more robust response to changes of parameters during simulation.

The dynamics of driven motion is also something that must be addressed. When the robot is in motion with the translational motion driven by the slider (the wheels in the real model), the balance must be achieved by applying a torque at the angular motor of the lower leg.

An interesting control scheme to be designed is the *endpoint control*, where the tracer point is set to a fixed coordinate and one degree of freedom is driven and the others must keep the position while maintaining the balance of the complete model.

5.3 Conclusion

After the completion of a project of these dimensions, a moment must be devoted for the reflection on the conclusions that can be drawn from the work what has been done. It is therefore pertinent to make a review of the effort made, the knowledge obtained, and, ultimately, the new skills acquired.

Obtaining a virtual model of an existing robot might seem a simple task, but it is far more difficult than one can expect. Considering that the information available on the internal working of the robot is scarce, extracting a virtual model, even if it is a simplified one, is an iterative process. One realizes that not all the mechanical configurations are valid, and many problems arise when trying to define a mechanical system that is efficient and gets the work done while maintaining a reduced complexity. Several iterations were made when defining the 2-dimensional model, being necessary to simplify the mechanical structures in several occasions to increase performance and decrease model complexity. It is clear that this robot had a great deal of brainstorming and troubleshooting behind and it is the work of great designers and engineers.

With the design of the mechanical system and its bodies and joints, the next step is to perform a mathematical development. The mechanical configuration of the system must be expressed symbolically using mathematical equations for the kinematic constraints between the bodies. These equations can be solved in order to obtain what is known as the kinematic solution. Due to the symbolic nature of the kinematic solution expressions, they can be differentiated to obtain expressions for the velocity and acceleration of the bodies. The kinematic solution provides an unequivocal value for the generalized coordinates of the bodies and their derivatives at any given time in terms of the degrees of freedom of the model. There is a huge amount of mathematical elaboration in this process. The most challenging part was the inclusion of the actuator lengths as degrees of freedom of the model in the most simplified way possible, being able to reduce the complexity of the equations by using a relative angle constraint. The kinematic solution is of considerable importance and serves as the starting point of for the further developments of the model.

One step further lies the dynamic model of the system, that must also be obtained mathematically and using the results yielded by the kinematic development. The dynamic model is necessary to design the control loops that provide dynamic stability to the robot during its motion. In this case, the model was reduced to a well-known physical pendulum on cart model, a nonlinear dynamic system that had to be linearized around the upwards unstable equilibrium point, which is where the model should operate. There are many developments applicable to linear time-invariant systems, like the one obtained from the linearization, that provide tools to design different controller schemes. The closed-loop control provides the necessary dynamic stability to maintain the model at its unstable equilibrium point. Different closed-loop controls were implemented, comparing their performance and behavior when put though different dynamic scenarios. It was satisfying to be able to apply the theoretical developments taught at the several control courses I have attended to a practical case obtaining successful simulation results.

Different industry-leading computer-aided engineering software were used during the development of this project. Some of them, such as *SolidWorks* and *Mathematica*, were no strangers to me, but I have been able to increase my competence and operate them more

fluently. *RecurDyn* and *SystemModeler* were new tools I had to learn, and their performance was much better than I expected. Results from one software were exported and validated using another, providing a successful interaction between the different computer-aided engineering softwares.

The result of the execution of this project can be considered as a success, since the objectives have been achieved. A simplified mathematical model of an existing robot was extracted, obtaining the symbolic kinematic solution to the model and a dynamic model that was dynamically controlled using different closed-loop configurations. A wide variety of computer-aided engineering software were used in the process, acquiring new competences in the operation of computational tools that are widely used professionally in the field of mechanical engineering. One must admit that there are many improvements that could be done and more insight should be put on some of the developments, but I can affirm that I have learned a great deal during the development of this thesis. I conclude these lines hoping they are not the last I write about the topic of social robotics.

Part II

Budget

Chapter 1

Introduction

In this case, the budget will be written as if it were a custom project being done under request. The client wishes to create a virtual prototype of the *Handle* robot using Computer Aided Engineering (CAE) software. The senior engineer in charge is part of an engineering office and the only one designed for the task.

The development of the project consists mainly on electronic documents, containing no tangible assets. The cost of the corresponding training in the different softwares and methodologies will not be taken into account, even though a previous amount of time was needed in order to become familiar with some specific software and the theoretical background for some elaborations. Taking into account the above, the budget will be ordered by sections, which are listed here: labor, hardware, and software.

Finally, a summary of the budget will be presented, where all the previous sections are included and the total cost is calculated by applying the corresponding percentages of industrial benefit and taxes on the added value.

Chapter 2

Budget

2.1 Labor cost

In the labor force, only the hours invested by the senior engineer will be taken into account, who will be responsible for the mechanical design, mathematical development, controller design, and simulations of the virtual prototype of the robot.

According to the Spanish ministerial order *ESS/55/2018, de 26 de enero, por la que se desarrollan las normas legales de cotización a la Seguridad Social, desempleo, protección por cese de actividad, Fondo de Garantía Salarial y formación profesional para el ejercicio 2018* published in the Spanish Official State Gazette (*Boletín Oficial del Estado*) on January 29th, 2018, a rate of contribution of 28.3% to the Social Security in terms of common contingencies is established. A total of 46 work weeks per year will be considered, equivalent to the corresponding 230 working days of 8 hours, and 10 extra hours per month with an additional price of 20%.

Taking as reference the provisions of the *XVIII Convenio colectivo nacional de empresas de ingeniería y oficinas de estudios técnicos* published in the Spanish Official State Gazette on January 18th, 2017, the following values will be considered: €3 per day as food diet, and €2.28 per day for transport bonuses (12 kilometers on average per day with a fare of €0.19 per kilometer).

Description	**Annual cost (€)**	**Cost (€/h)**
Base salary (230 days)	40,000.00	21.74
Social security (28,3%)	11,320.00	6.15
Diets	690.00	0.38
Pluses	524.40	0.29
Extra hours (10 h/month)	3,000.00	1.63
Total	**€55,534.40**	**€30.18**

Table 2.1: Breakdown of the labor cost of the senior Industrial Engineer.

A detailed breakdown of the hours of work dedicated to each of the parts of which this project is composed is found below.

Description	**Quantity (hours)**	**Unit price (€/h)**	**Subtotal (€)**
Simplified 2-dimensional model in *SolidWorks*	30	30.18	905.45
Symbolic mathematical model in *Mathematica*	110	30.18	3319.99
Dynamic control in *RecurDyn*	50	30.18	1509.09
Modelica-based model in *SystemModeler*	70	30.18	2112.72
Drafting of the project documents	40	30.18	1207.27
Total labor cost	**300 hours**		**€9,054.52**

Table 2.2: Total labor budget.

Only effective hours of work in the development of the project will be considered in the labor cost, even though more hours were necessary to acquire the necessary competences in the operation of the softwares and to refresh the knowledge about certain aspects of the mathematical methods behind some elaborations.

2.2 Hardware cost

An amortization period for computer equipment of 3 years will be taken into account at a rate of 1840 hours per year (230 working days with days of 8 hours), calculating the cost of these per hour worked and using this value to calculate the corresponding part for the budget.

$$\text{Amortization cost (€/h)} = \frac{\text{Price of equipment (€)}}{\text{Amortization period (h)}}$$

Description	**Price (€)**	**Amortization cost (€/h)**	**Quantity (h)**	**Subtotal (€)**
Workstation HP Z2 Mini G3	1830.16	0.33	300	99.47
Total hardware cost				**€99.47**

Table 2.3: Total computer hardware budget.

2.3 Software cost

In this section, licenses for the use of all the software used for the development of the project will be considered.

An amortization period for software of 3 years will be taken into account at a rate of 1840 hours per year (230 working days with days of 8 hours), calculating the cost of these per hour worked and using this value to calculate the corresponding part for the budget.

$$\text{Amortization cost (€/h)} = \frac{\text{Price of license (€)}}{\text{Amortization period (h)}}$$

Description	**Licence price (€)**	**Amortization cost €/h)**	**Quantity (h)**	**Subtotal (€)**
SolidWorks + COSMOSMotion 2007	5300.00	0.96	30	28.80
Wolfram Mathematica 11	3445.00	0.62	110	68.65
RecurDyn V9R1	5000.00	0.91	50	45.29
Wolfram SystemModeler 5.1	4830.00	0.88	70	61.25
TeXstudio + MiKTeX	0.00	0.00	40	0.00
Windows 10 Pro	199.99	0.04	300	10.87
Total software cost				**€214.86**

Table 2.4: Total computer software budget.

Chapter 3

Summary

The final budget is detailed below, which includes all the previous points, adding a 6% industrial profit, the 13% for general expenses and the corresponding 21% VAT.

Description	**Quantity**	**Unit price (€)**	**Subtotal (€)**
Total labor cost	1.00	9,054.52	9,054.52
Total hardware cost	1.00	99.47	99.47
Total software cost	1.00	214.86	214.86
Budget of Material Execution			**€9,368.85**

Table 3.1: Budget of Material Execution.

Description	**Cost (€)**
Budget of Material Execution	9,368.85
General expenses (13%)	1,217.95
Industrial benefit (6%)	562.13
Budget for Contractual Execution	**€11,148.93**

Table 3.2: Budget of Contractual Execution.

Description	**Cost (€)**
Budget for Contractual Execution	11,148.93
VAT (21%)	2,341.28
Total Budget	**€13,490.21**

Table 3.3: Total Budget.

The total budget of the project amounts to THIRTEEN THOUSAND FOUR HUNDRED NINETY EURO AND TWENTY ONE CENTS.

Part III

Appendices

Chapter A

Self-aligning mechanisms

A.1 Basic notions

First, we must define a *kinematic pair*, or *joint*, as a linkage between elements which limits some relative motions while allowing others. The number of limited motions (constraints or linkage conditions) can be linear, along a coordinate axis, or angular, around a coordinate axis. This number, defined as *joint class* and designated by Roman numerals, represents the number of forces and torques that can be transmitted through the joint. The amount of allowed relative motions is referred to as *joint mobility*. The sum of the class of a kinematic pair and its mobility is equal to six.

The term *joint constraints* refers to the limited relative displacements, both linear and angular, along and around each one of the coordinate axes. A limited linear displacement in a joint induces a constraint force, while a limited angular displacement induces a constraint torque. The notion of constraints in kinematics are associated to the notions of constraint force or torque in dynamics. The joint must be correctly calculated, adequately dimensioning the bodies which make it up, in order to withstand the constraint forces and torques that will appear.

Figure A.1 extracted from Professor L. N. Reshetov's book shows a classification of kinematic pairs. They are classified in rows with Roman numerals according on their classes (number of constraints) while columns in Arabic numerals represent possible constructive solutions. The designation of each joint in the analysis of a mechanism will be done by the Roman numeral with a subindex according to the column number, this way it will be easy to locate in the table. The last column on the right determines the mobility of the kinematic pair, the number of linear and angular movements it allows between the elements connected by the joint. As seen in the table, the sum of a joint's class and its mobility is equal to six.

The *degrees of freedom* (DOF) of a body is the number of independent coordinates needed to uniquely specify the position of the body with respect to a given reference system. Similarly, the minimum number of coordinates needed to uniquely specify the positions of all the components in a rigid body system will be the degrees of freedom of said system.

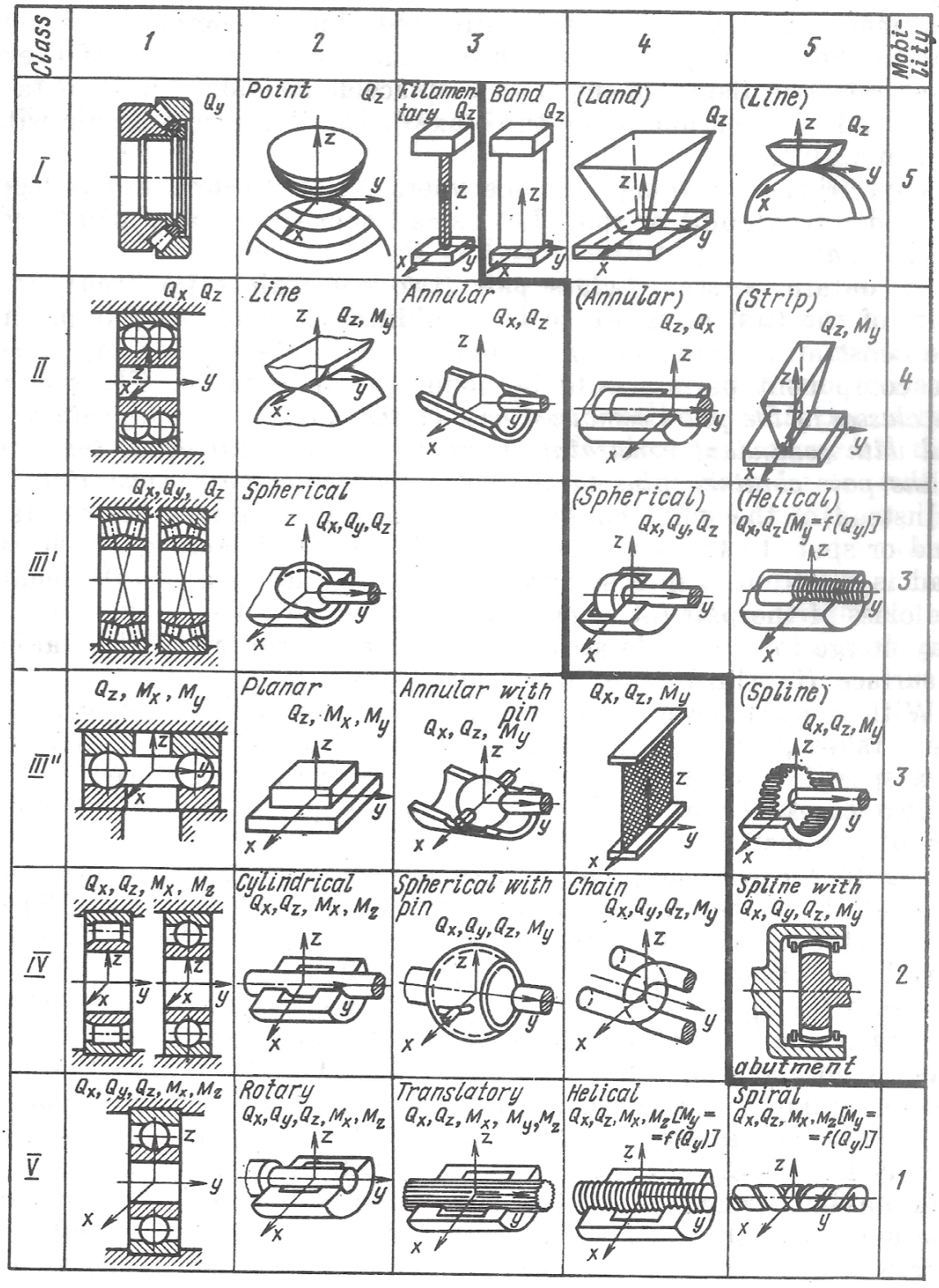


Figure A.1: Table of kinematic pairs from Professor L. N. Reshetov's book.

We will use the concept of degree of freedom in three different ways, applied to different elements, but closely related to one another. First, it will be the degrees of freedom of a body with respect to a reference system previously defined. Second, it will be the degrees of a kinematic pair. And third, it will also be the degrees of freedom of a mechanism. Because of this, and in order to make it more clear, when we address the degrees of freedom of a kinematic pair, we will use the term *connectivity*, f_i. This same term will apply to the relative degrees of freedom between two bodies. When referring to a mechanism, the degrees of freedom will be addressed as the *mobility* of said mechanism.

The formal definition of the terms is the following: (1) if a kinematic pair is defined between two bodies which are not connected to any other body, the connectivity of this joint is equal to the degrees of freedom of the bodies with respect to each other; (2) the mobility of a mechanism is the minimum number of coordinates necessary to specify the position of all the components of the mechanism with respect to a component of the mechanism chosen as the base (or fixed) body.

The mobility, or degrees of freedom (DOF) of a mechanism, is used to determined how many joint variables must be specified before locating the points of a mechanism as temporal functions. A mechanism must have a mobility equal or greater than one. Traditionally, almost all the mechanisms had only one degree of freedom; but, in modern design practice, it is usual to find mechanisms with two or more degrees of freedom. If the mobility is zero or negative, as determined by the equations of mobility further on, the assembly is a structure. If the mobility is equal to zero, the structure is designated as statically determined. If the mobility is negative, the structure is referred to a hyperstatic or statically indeterminate.

In order to calculate the mobility of a mechanism, we will first consider the case of a planar linkage, and the result will later be extended to the three dimensional world. Within a plane, a body which is free to move has three degrees of freedom. A planar mechanism, or planar linkage, is one where all of the bodies of the mechanism are restricted to lie on the same plane or in parallel planes. From this definition we can conclude that the vast majority of the mechanisms used in practice are planar, hence the importance of studying this type of mechanisms initially.

Lets consider a mechanism with N components or rigid bodies, and a number P of joints which connected them together. The number of components N must consider the number of moving bodies plus the fixed body or ground.

If the mechanism is contained in a plane, the total mobility of the mechanism will be:

$$M_{2D} = 3\,(N - P - 1) + \sum_{i=1}^{P} f_i \tag{A.1}$$

This equation is known as the 2-dimensional *mobility formula*. In the specialized literature, it is possible to find differently formulated versions of this formula, all of them equivalent to each other.

In space, the movement of each free body has six degrees of freedom instead of three. The mobility formula in this case is:

$$M_{3D} = 6\,(N - P - 1) + \sum_{i=1}^{P} f_i \tag{A.2}$$

This equation is a generalization of the previous one and it is referred to as the 3-dimensional *mobility formula*, or the *Grübler-Kutzbach criterion*.

It is this same criterion the one that *SolidWorks* add-on *COSMOSMotion* and multibody dynamics software *RecurDyn* uses internally to calculate the mobility or degrees of freedom of a mechanism. Multibody dynamic simulation software contains this mathematical tool, along with many others, in order to exhaustively analyze a mechanism and its mobility.

A.2 Redundant constraints and self-aligning mechanisms

The *theory of machines and mechanisms* is the science which studies the kinematic pairs (also known as linkages) and their importance in the development of mechanisms. This science must provide recommendations about the different existing joints and their applications.

One of the authors who have contributed most to this task is Professor Leonid N. Reshetov[1], with his book "*Self-aligning mechanisms*". Professor L. N. Reshetov concludes that in order to achieve a substantial improvement in the performance of the mechanism, it must be statically determined or, according to the author's nomenclature, it must be *self-aligning*. In this book, the theory behind self-aligning mechanisms is developed and applied to several examples with the final purpose to show designers and engineers how to get the mechanisms they use to be self-aligned.

To facilitate the assembly the mechanisms, it is convenient to choose a scheme such that the fact that the dimensions of its components differ from the theoretical does not suppose any problem. The most convenient way is to use statically determined mechanisms, mechanisms without redundant (or passive) constraints, which we will call *self-aligning mechanisms*. *Redundant constraints* are defined as constraints which can be removed without increasing the mobility of the mechanism, they occur when some joints are constraining the same degrees of freedom as other joints.

According to Professor L. N. Reshetov, the existence of redundant constraints in a mechanism is a harmful factor. The dimensions of the components of a mechanism can vary during the lifecycle of the machines as a result of many different reasons: subsidence of the foundations, wear after many cycles or hours of use, correction of the play in the kinematic pairs, elastic deformations, thermal dilation, as well as due to mistakes made during repair and assembly. A statically determined mechanism is not dependent on the variation of the dimensions of its elements. Therefore, the static determination of a mechanism not only solves the problem of reducing assembly expense, but also solves at the same time the problem of raising its reliability in service.

There some exceptional cases when the *Grübler-Kutzbach criterion* return an invalid value for the mobility of a mechanism, two of the most common ones will be discussed next.

The first case is when the system contains *redundant constraints*, in this case the mobility formula gives a negative value. The *apparent mobility* of a mechanism is the number of useful real degrees of freedom it contains, this is usually known by previous analysis of the mechanism. The difference between the apparent mobility, M_a, and the mobility formula, if the resulting value is positive, is the number of redundant constrains of the mechanism.

$$\text{Number of redundant constraints} = M_a - M_{3D} > 0 \tag{A.3}$$

[1] **Leonid Nikolayevich Reshetov** (1906-1998), was a distinguished professor at the Theory of Mechanisms and Machines Department of the Bauman Moscow State Technical University. He received the Honorary Title *Honored Inventor of RSFSR*, the highest scientific award available at the time at the Russian Federation. As a great engineer, scientist and inventor, he also took an active part in improvement of courses in the Theory of Mechanisms and Machines and development of Machinery Design.

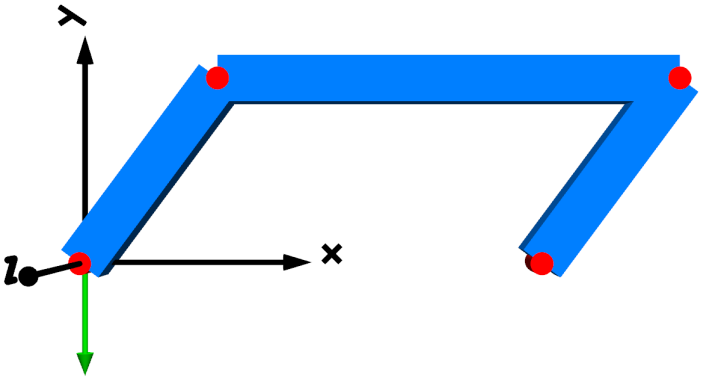


Figure A.2: A four-bar linkage depicted in *SystemModeler*.

An example of this is the four-bar linkage defined by three moving bodies and a fixed one joined together by four revolute (rotary) joints (class V joint only allowing relative rotation in one axis, connectivity = 1). This mechanism has one degree of freedom, therefore its apparent mobility is one. If the mobility formula is applied:

$$\begin{gathered} M_{3D} = 6\,(4-4-1) + (1+1+1+1) = -2 \\ \text{Number of redundant constraints} = 1-(-2) = 3 \end{gathered} \tag{A.4}$$

There are three redundant constraints in this mechanism, therefore some revolute joints must be replaced with joints with a higher connectivity. The sum of the replacement joints' connectivity must be three units higher. There are two solutions: replacing two revolute joints for an spherical (class III, connectivity = 3) and a cylindrical (class VI, connectivity = 2) joint, or replacing one revolute joint for a in line (class II, connectivity = 4) joint.

The second case is when there are *passive degrees of freedom* in the mechanism. This occurs when the joints have not been correctly selected and there are internal unwanted motions possible inside the mechanism. For example, when there is a body with two spherical joints at its extremities and the body can perform a rotation along its longitudinal axis.

$$\text{Passive degrees of freedom} = M_{3D} - M_a > 0 \tag{A.5}$$

Self-aligning a mechanism is an iterative process. Kinematic pairs have to be replaced for others with different connectivity, but some times not all possible combinations are valid. In very complex mechanisms, for example those that contain entangled closed loops or redundancy of mechanisms, some combinations can lead to the appearance of passive degrees of freedom. Some mechanisms have several solutions for the self-alignment that are valid and where the mobility formula returns the correct number of degrees of freedom.

A.3 Application to CAE software

Eliminating redundant constraints is a very important part in the dynamic analysis of motion in a multibody system. When the mathematical model of the system has several equations constraining the same degrees of freedom, these duplicate equations must be eliminated from the problem. Kinematic analysis of the system is independent of the number of redundant constraints, but the wrong choice of constraints can affect computational performance and convergence.

Rigid multibody systems with redundant constrains lack a unique solution for the calculation of reaction forces at joints. This kind of system is statically undetermined, with the number of equilibrium equations less than the number of unknown reaction forces. There are many applications for which the calculation of reaction forces is not necessary, for example when only the kinematic behavior of the mechanism is needed, but many times this is not the case. In complex models where the friction in joints has to be considered, or when an analysis of forces and torques is needed in order to correctly determine the size of the joints, the reaction forces must be obtained.

The constraints between pairs of bodies are defined by a set of algebraic constraint equations that are equivalent to the physical joint. There are different equations depending on the type of joint. In a general manner, the mathematical conditions on the relative motion between bodies imposed by the i-th joint is expressed by a constraint equation, which is a vector equation equivalent to a set of scalar equations:

$$\mathbf{\Phi_i(q)} = \mathbf{0} \tag{A.6}$$

Being $\mathbf{q} = (q_1, q_2, ..., q_n)$ the vector of the n generalized coordinates of the system.

The constraint equations for all the kinematic pairs can be unified in a vector equation. If the multibody system is described by n generalized coordinates, and the joints are expressed by m scalar equations, the vector equation is:

$$\mathbf{\Phi(q)} = \begin{pmatrix} \Phi_1(\mathbf{q}) \\ \Phi_2(\mathbf{q}) \\ \vdots \\ \Phi_m(\mathbf{q}) \end{pmatrix} = \begin{pmatrix} \Phi_1(q_1, q_2, ..., q_n) \\ \Phi_2(q_1, q_2, ..., q_n) \\ \vdots \\ \Phi_m(q_1, q_2, ..., q_n) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \mathbf{0} \tag{A.7}$$

The Jacobian matrix of constraint equations is a m by n matrix defined as:

$$\mathbf{\Phi_q(q)} = \begin{pmatrix} \frac{\partial \Phi_1}{\partial q_1} & \frac{\partial \Phi_1}{\partial q_2} & \cdots & \frac{\partial \Phi_1}{\partial q_n} \\ \frac{\partial \Phi_2}{\partial q_1} & \frac{\partial \Phi_2}{\partial q_2} & \cdots & \frac{\partial \Phi_2}{\partial q_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \Phi_m}{\partial q_1} & \frac{\partial \Phi_m}{\partial q_2} & \cdots & \frac{\partial \Phi_m}{\partial q_n} \end{pmatrix} = \begin{pmatrix} [\Phi_1]_q \\ [\Phi_2]_q \\ \vdots \\ [\Phi_m]_q \end{pmatrix} \tag{A.8}$$

Multibody dynamics software, such as *RecurDyn* or *COSMOSMotion*, analyzes the Jacobian matrix of constraint equations in search for redundant constraints and eliminates them from the set of equations. In the mathematical model, when the kinematic pairs repeat constraints imposed by other joints, equations from the vector of constraints appear to be linearly dependent from each other. In order to check if the constraint equations are independent, it is enough to check the rank of the Jacobian matrix. If the Jacobian matrix rank is equal to the number of equations, all the constraint equations are independent; but if it has a lower rank, there are dependent constraints equations in the model. These dependent equations define the redundant constraints that must be removed.

Elimination of redundant constraints is necessary to build a non-singular matrix for the linear equation set and to uniquely determine the Lagrange multipliers of the remaining constraints in order to apply iterative root-finding algorithms needed to obtain the numerical solution. Usually, there is not a unique solution for the identification of redundant constraints, this set of equations to be removed can be chosen in many ways. However, this is an internal procedure of the software and sometimes no information is given on which sets of equations were removed.

For example, lets consider a body with an arbitrary mass secured by two fixed joints at its ends. It is quite obvious, assuming that the body is rigid and has a uniform mass distribution, that the reaction force on each fixed joint is equal to half its mass. However, if this model is simulated in a multibody software environment the joints have different reaction forces: one joint absorbs all the weight of the body, while the other experiences no force. This is the typical result of a over-constrained model with redundant constraints. A body has six degrees of freedom in space, and one fixed joint is enough to restrict all of them. The software eliminates the six constraint equations of the second fixed joint which it considers as a redundant constraint, only taking into consideration one of them when doing the simulation. This results in one bearing all the weight while the other takes zero load. If only the kinematic behavior is important in this simulation there is no problem, but if the application requires to precisely measure the reaction forces in both joints the model must be self-aligned to get the correct results.

Despite the fact that kinematic behavior of the system is unaffected by existing redundant constraints, the problem of obtaining the constraint reactions has an infinite number of solutions. In this case, due to the sometimes arbitrary elimination of dependent constraint equations, not all the solutions are physically accurate or even possible. It is very important to correctly obtain a self-aligned mechanism in order to remove redundant constraints beforehand to avoid leaving the decision of equation elimination to the software, which could result in unexpected results. From the professional experience acquired working with simulation software and contacting support teams in the last years, avoiding redundant constraints is considered as good modeling practice.

A.3.1 Bushing forces

In many multibody dynamics software, such as *RecurDyn* or *MSC/ADAMS*, there is an alternative to the process of self-aligning: bushing forces.

Bushing forces are a way of avoiding the redundant constraint and the automatic equation elimination done by the software. It could be considered as a type of general constraint equation $\mathbf{\Phi_i(q) = 0}$ condition in which the displacement of the body can be restricted in its six degrees of freedom using a series of parameters.

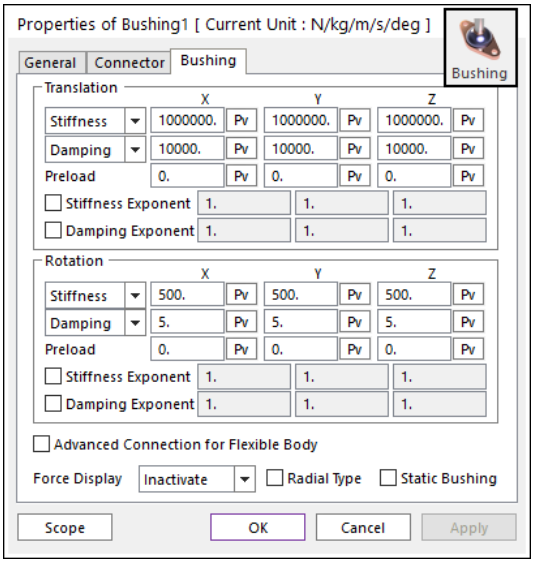


Figure A.3: Bushing force definition in *RecurDyn*.

Lets consider the classical example of a over-constrained rigid body resting on the ground secured by two fixed joints. According to classical mechanics, the reaction force on each fixed joint should the $R_y = \frac{mg}{2}$. If the model is simulated, due to the elimination of redundant constraint equations, all of the upwards reaction force is exerted by one joint, while the other is unloaded.

To obtain the correct reaction forces, joints must be replaced with busing forces. For these kinds of forces, a very high value of stiffness must be entered in the degree of freedom the joint constraints. For a fixed joint, the six degrees of freedom must be constrained (translation and rotation) with high stiffness values for all six of them. In a revolute joint the parameters restricting the rotational degrees of freedom must be zero, while the stiffness of the translational must be very high.

The counterpart of this is that the bushing force is not able of constraining a body as firmly as a real joint will, the body could experience some degree of displacement. The force exerted is proportional to the displacement, following Hooke's law $F = -k\,\Delta x$, therefore higher stiffness parameters reduce the amount of displacement that occur.

Even though bushing forces can replace self-aligning, it is advisable to go though the procedure of self-aligning which provides the designer a better understanding of the operation of the mechanism. Using bushing forces for large systems with many joints can be a tedious process and can lead to unexpected results due to the small displacements that are intrinsic to the bushing forces formulation and the large forces that can occur.

Chapter B

Variable-sided triangles

Triangles are used in a wide variety of applications. In static structures, the rigidity a triangle provides is well-known since the ancient times and it is still used in the present day as the main building block of trusses. Trusses support centenary churches and cathedrals, transmission towers, bridges, bicycle frames, buildings, timber roofs, and many more structures. The properties of triangles are perfectly defined using trigonometric formulas that provide relationships between their sides and angles.

However, triangles are much more than static figures. Extremely useful mechanisms arise when one of the sides of the triangle can vary in length. These *variable-sided triangles* are much more common than they seem. There are applications where the length of the variable side of the triangle is changed in a discrete way, for example the different positions of regulation in a deck chair; but in other mechanisms its length can be changed continuously producing very interesting machines such as car jacks and it is the basics behind some vehicle and motorcycle coil spring suspensions. The variable-sided triangle has one degree of freedom: once the movement of any part of the mechanism is known, the position of all the other parts is determined.

In the last half-century, many machines have been upgraded by the introduction of a hydraulic cylinder as the variable side on a triangle. A hydraulic cylinder allows the length of the actuator to be modified by pumping oil between the two different sides of the cylinder. In civil engineering there are plenty of examples of the applications of these variable-sided triangles run by hydraulic cylinders, for example the tipping mechanism of a dump truck or the arms of modern excavators and backhoes.

Variable-sided triangles are used extensively in the model of the *Handle* robot: all the linear actuators are examples of this kind of triangles. Using several trigonometric properties and formulas it will be possible to describe variable-sided triangles and provide a simple mathematical relationship between the two bodies they connect together.

B.1 Trigonometric notions

B.1.1 Tangent half-angle formulas

Using the *double-angle formulas* and the *Pythagorean trigonometric identity* it is possible to obtain formulas that relate the tangent of half of an angle to trigonometric functions of the whole angle.

$$\begin{aligned}
\sin\alpha &= 2\sin\frac{\alpha}{2}\cos\frac{\alpha}{2} = \frac{2\sin\frac{\alpha}{2}\cos\frac{\alpha}{2}}{\cos^2\frac{\alpha}{2}+\sin^2\frac{\alpha}{2}} = \frac{2\tan\frac{\alpha}{2}}{1+\tan^2\frac{\alpha}{2}} \\
\cos\alpha &= \cos^2\frac{\alpha}{2}-\sin^2\frac{\alpha}{2} = \frac{\cos^2\frac{\alpha}{2}-\sin^2\frac{\alpha}{2}}{\cos^2\frac{\alpha}{2}+\sin^2\frac{\alpha}{2}} = \frac{\frac{\cos^2\frac{\alpha}{2}}{\cos^2\frac{\alpha}{2}}-\frac{\sin^2\frac{\alpha}{2}}{\cos^2\frac{\alpha}{2}}}{\frac{\cos^2\frac{\alpha}{2}}{\cos^2\frac{\alpha}{2}}+\frac{\sin^2\frac{\alpha}{2}}{\cos^2\frac{\alpha}{2}}} = \frac{1-\tan^2\frac{\alpha}{2}}{1+\tan^2\frac{\alpha}{2}} \\
\tan\alpha &= \frac{\sin\alpha}{\cos\alpha} = \frac{\frac{2\tan\frac{\alpha}{2}}{1+\tan^2\frac{\alpha}{2}}}{\frac{1-\tan^2\frac{\alpha}{2}}{1+\tan^2\frac{\alpha}{2}}} = \frac{2\tan\frac{\alpha}{2}}{1-\tan^2\frac{\alpha}{2}}
\end{aligned} \tag{B.1}$$

These formulas, known as the *tangent half-angle formulas*, are useful to unify sines and cosines of an angle which appear in an equation under a single trigonometric unknown which is the tangent of half of the original angle. The equation complexity is reduced to a rational equation with a single final inverse trigonometric calculation.

For example, lets consider the equation:

$$a\sin\varphi+\frac{b}{2}\cos\varphi-\frac{b}{2}=0 \tag{B.2}$$

This equation has a variable φ which is contained in two different trigonometric functions. Through the tangent half-angle formulas, using the following substitution:

$$\begin{aligned}
u &= \tan\frac{\varphi}{2} \\
\sin\varphi &= \frac{2\tan\frac{\varphi}{2}}{1+\tan^2\frac{\varphi}{2}} = \frac{2u}{1+u^2} \\
\cos\varphi &= \frac{1-\tan^2\frac{\alpha}{2}}{1+\tan^2\frac{\alpha}{2}} = \frac{1-u^2}{1+u^2}
\end{aligned} \tag{B.3}$$

The trigonometric equation is transformed into a rational equation and solved using the inverse tangent function:

$$\begin{aligned}
\frac{2au-bu^2}{u^2+1} &= 0 \\
2au-bu^2 = u(2a-bu) &= 0 \\
\mathbf{1}.\ u=0 \rightarrow \tan\frac{\varphi}{2} &= 0 \rightarrow \varphi_1 = 0 \\
\mathbf{2}.\ 2a-bu=0 \rightarrow u = \tan\frac{\varphi}{2} &= \frac{2a}{b} \rightarrow \varphi_2 = 2\tan^{-1}\frac{2a}{b}
\end{aligned} \tag{B.4}$$

B.1.2 Cosine formula

In trigonometry, the *cosine formula* (also known as the *law of cosines*) is a generalization of the *Pythagorean theorem*, applying to any triangle instead of only to right angle triangles. This law relates the lengths of the sides of any triangle to the cosine of one of its angles.

$$c^2 = a^2 + b^2 - 2\,a\,b\,cos\gamma \tag{B.5}$$

It is easy to check that when applied to a right triangle with $\gamma = \frac{\pi}{2}$ *radians* the cosine formula is equivalent to the *Pythagorean theorem*.

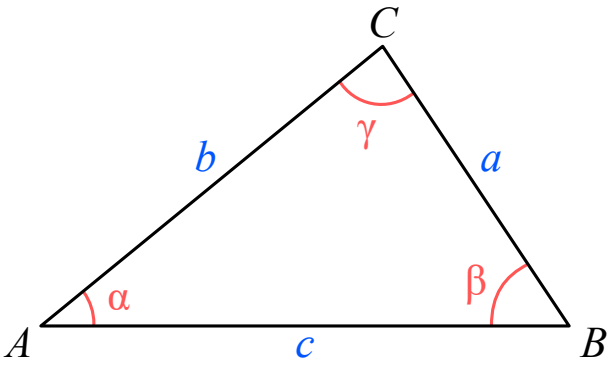


Figure B.1: A generic triangle ABC. The angles α (A), β (B), and γ (C) are respectively opposite to the sides a, b, and c.

The first appearance of this law is found in Euclid's masterpiece, *Elements*, a mathematical treatise consisting of 13 books published around 300 BC in Alexandria (currently Egypt). Its applications are widely used in triangulation, where three out of four variables (a, b, c, and γ) are known and the other is obtained using the formula. For example, in order to obtain the third side of a triangle knowing two sides and the angle between them, to obtain the angles of a triangle if its three sides are known, or to obtain the third side of a triangle if two sides and an angle opposite to one of them is known.

$$\begin{aligned} c &= \sqrt{a^2 + b^2 - 2\,a\,b\,cos\gamma} \\ \gamma &= \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2\,a\,b}\right) \\ a &= b\,cos\gamma \pm \sqrt{c^2 - b^2\,sin^2\gamma} \end{aligned} \tag{B.6}$$

B.2 Mathematical definition of variable-sided triangles

There are two ways to implement variable-sided triangles as constraints between pairs of bodies: as a *relative distance constraint* or as a *relative angle constraint*.

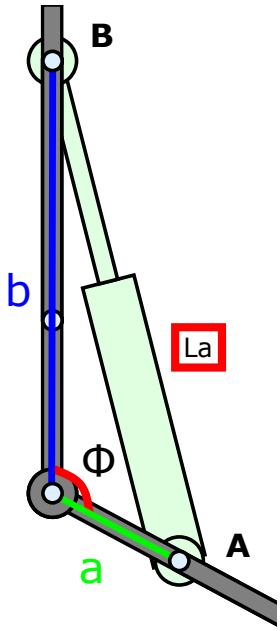


Figure B.2: Variable-sided triangle and relative angle constraint between bodies.

A *relative distance constraint* between point A on body i and point B on body j requires that the distance between these points is equal to a given constant $K > 0$. The constraint equation which describes this relationship between two bodies is:

$$\begin{gathered}(X_A - X_B)^2 + (Y_A - Y_B)^2 - K^2 = 0 \\ \Big((X_i + x'_{iA}\cos\Theta_i - y'_{iA}\sin\Theta_i) - (X_j + x'_{jB}\cos\Theta_j - y'_{jB}\sin\Theta_j)\Big)^2 + \\ + \Big((Y_i + x'_{iA}\cos\Theta_i - y'_{iA}\sin\Theta_i) - (Y_j + x'_{jB}\cos\Theta_j - y'_{jB}\sin\Theta_j)\Big)^2 - K^2 = 0\end{gathered} \tag{B.7}$$

Upper case letters represent absolute coordinates and the lower case represent local coordinates. The rotation angles of the bodies local coordinate axes with the absolute coordinate systems are represented by Θ.

A *relative angle constraint* requires that the difference between the rotation angles of bodies i and j is equal to a given constant ϕ. The constraint equation which describes this relationship between two bodies is:

$$\Theta_j - \Theta_i - \phi = 0 \tag{B.8}$$

This constraint can also be expressed by taking sines on both sides:

$$\sin(\Theta_j - \Theta_i - \phi) = 0 \tag{B.9}$$

This equation can be expanded into sums of multiplications of trigonometric functions (sines and cosines) of single angles.

The difference in complexity of these equations is obvious. Moreover, the Jacobian of relative angle constraints is much more less complex than that of relative distance constraints. The counterpart is that the K constant in the distance constraint is directly the length of the variable side of the triangle La (length of the actuator), but in the angle constraint the ϕ constant must be obtained using the cosine theorem.

For example, in Figure B.2 the constant for the relative angle between two bodies in terms of the length of the linear actuator is expressed as:

$$\phi = \cos^{-1}\left(\frac{a^2 + b^2 - La^2}{2ab}\right) \tag{B.10}$$

This formulation has another advantage, it is possible to obtain the maximum and minimum values of the length of the actuator from the expression of the relative angle. The arc cosine function $\cos^{-1}(x)$ is defined for a specific domain of x: $-1 \leq x \leq 1$. The maximum and minimum lengths can be obtained with the formula:

$$\left|\frac{a^2 + b^2 - La^2}{2ab}\right| = 1 \tag{B.11}$$

$$\begin{array}{ll} \dfrac{a^2 + b^2 - La^2}{2ab} = 1 & \dfrac{a^2 + b^2 - La^2}{2ab} = -1 \\ a^2 - 2ab + b^2 = La^2 & a^2 + 2ab + b^2 = La^2 \\ (a-b)^2 = La^2 & (a+b)^2 = La^2 \\ \pm(a-b) = La & \pm\,(a+b) = La \end{array}$$

There are four solutions, but only two of them are relevant. Considering that a, b and La are strictly positive values bigger than 0, the maximum value is $La_{max} = (a + b)$ while the negative $(-a - b)$ solution can be discarded. From the remaining two solutions the minimum value can be obtained, depending on which parameter is greater: if $a > b$, $La_{min} = (a - b)$; and if $b > a$, $La_{min} = (b - a)$. This result agrees with what can be easily deducted from the graphical analysis of the triangle.

These are the maximum and minimum mathematical values, but they must be compared not to surpass the minimum and maximum values established by the length of the rod and barrel of the cylinders that make up the actuator.

Chapter C

Interpolating step functions

In the real world, there is not such thing as a perfect step in the domain of time. When a system transitions from one state to another, it always happens in a finite time; even if this time interval is extremely small, it is never equal to 0. The same principle is valid in the realm of simulation.

In the mathematical world, a sudden change from one value to another is known as a step. The mathematical function which describes this phenomenon is the *Heaviside step function*, also known as the *unit step function*. The definition of this function is given by:

$$\mathbf{H}(\mathbf{x}) = \begin{cases} 0 & x \leq 0 \\ \frac{1}{2} & x = 0 \\ 1 & x \geq 0 \end{cases} \tag{C.1}$$

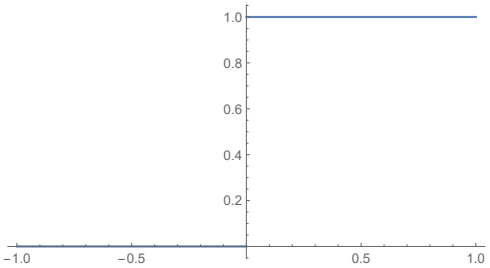


Figure C.1: Heaviside step function plotted using *Mathematica* for $-1 \leq x \leq 1$.

It is quite obvious that this function can cause difficulties in numerical integration due to the discontinuity at $x = 0$, therefore there are several analytic approximations available. These approximations rely on S-shaped curves, known as *sigmoid functions*, which produce smooth transitions from one value to another using trigonometric functions or polynomials.

RecurDyn can be considered as a descendant from *MSC/ADAMS*; and as it usually happens with offspring, they share many traits with their predecessors. Among many other things, the different ways of handling of step functions is something both softwares have in common.

C.1 STEP function

The first approximation is the well-known *STEP function*, which interpolates the Heaviside step function using a cubic polynomial. Its definition using a piecewise function is as follows:

$$\mathbf{STEP}(x, x_0, h_0, x_1, h_1) =$$
$$= \begin{cases} h_0 & x \leq x_0 \\ h_0 + (h_1 - h_0)\left(\dfrac{x - x_0}{x_1 - x_0}\right)^2 \left(3 - 2\,\dfrac{x - x_0}{x_1 - x_0}\right) & x_0 \leq x \leq x_1 \\ h_1 & x \geq x_1 \end{cases} \tag{C.2}$$

Being x the independent variable for the defined function, x_0 the starting point (x-value) at which the function begins, h_0 the initial value of the function, x_1 the ending point (x-value) at which the function ends, and h_1 the final value of the function.

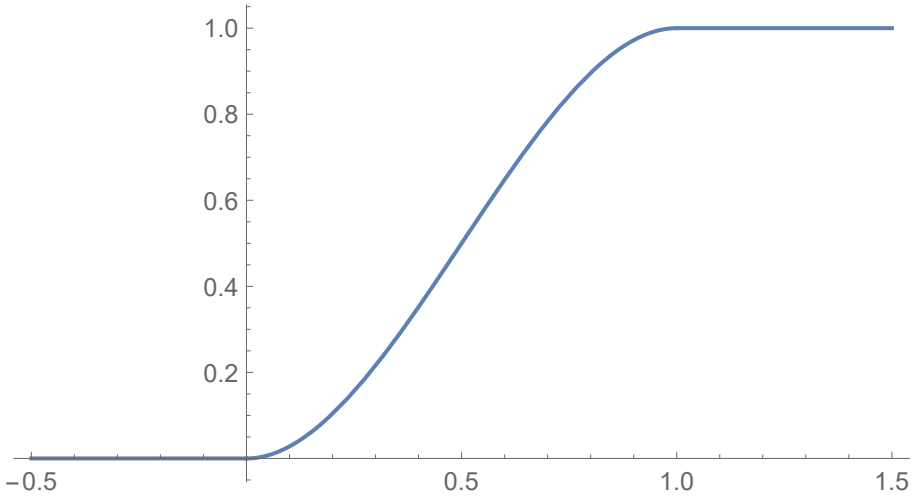


Figure C.2: STEP function plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$.

This function has a continuous first derivative, but with sharp edges at $x = x_0$ and $x = x_1$, causing the second derivative to be discontinuous at these points, making it a class C^1 function.

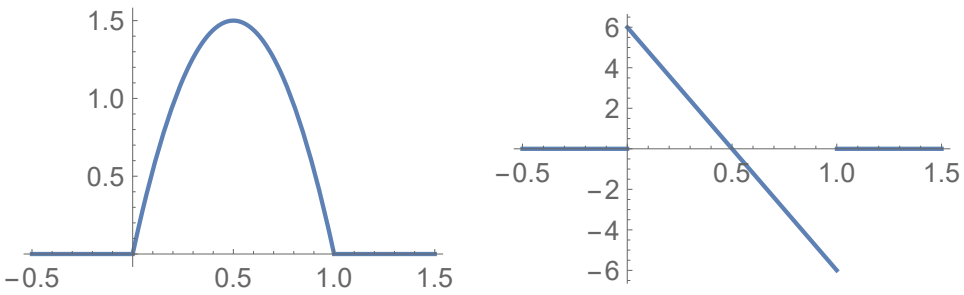


Figure C.3: STEP function derivatives plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$. First derivative on the left and second derivative on the right.

C.2 STEP5 function

Similar to the STEP function, the *STEP5 function* interpolates the Heaviside step function using a quintic polynomial. It is defined by:

$$\mathbf{STEP5}(x, x_0, h_0, x_1, h_1) =$$

$$= \begin{cases} h_0 & x \leq x_0 \\ h_0 + (h_1 - h_0)\left(\dfrac{x - x_0}{x_1 - x_0}\right)^3 \left(10 - 15\,\dfrac{x - x_0}{x_1 - x_0} + 6\left(\dfrac{x - x_0}{x_1 - x_0}\right)^2\right) & x_0 \leq x \leq x_1 \\ h_1 & x \geq x_1 \end{cases} \tag{C.3}$$

Being x the independent variable for the defined function, x_0 the starting point (x-value) at which the function begins, h_0 the initial value of the function, x_1 the ending point (x-value) at which the function ends, and h_1 the final value of the function.

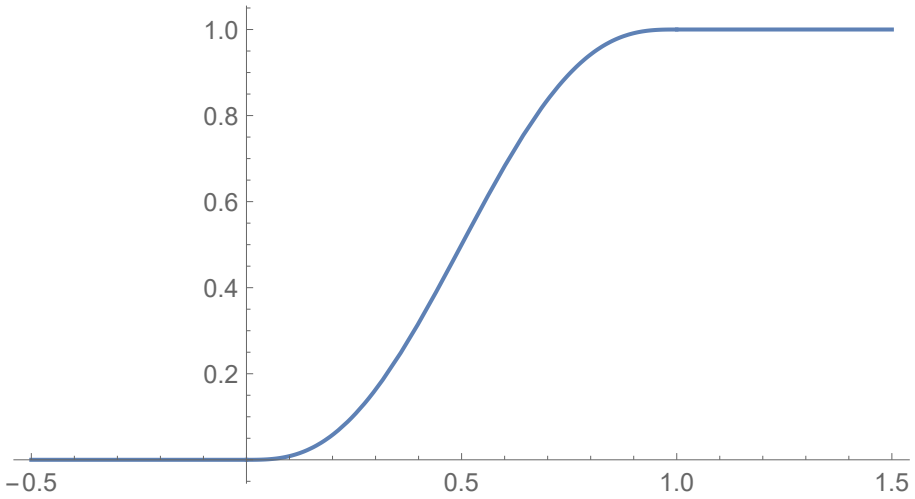


Figure C.4: STEP5 function plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$.

Because this function is a one degree higher polynomial function than STEP, it has a continuous first and second derivative, making it a class C^2 function. The second derivative has sharp edges at $x = x_0$ and $x = x_1$.

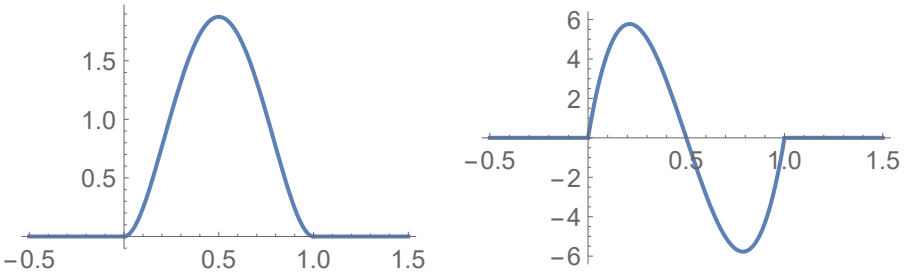


Figure C.5: STEP5 function derivatives plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$. First derivative on the left and second derivative on the right.

C.3 HAVSIN function

A third approximation is achieved using a partial *Haversine function*, this is an approximation of the Heaviside step function using a trigonometric function. It is defined by a piecewise function as follows:

$$\mathbf{HAVSIN}(x, x_0, h_0, x_1, h_1) = \\ = \begin{cases} h_0 & x \leq x_0 \\ \dfrac{(h_0 + h_1)}{2} + \dfrac{(h_1 - h_0)}{2} \, sin\left(\pi \dfrac{x - x_0}{x_1 - x_0} - \dfrac{\pi}{2}\right) & x_0 \leq x \leq x_1 \\ h_1 & x \geq x_1 \end{cases} \tag{C.4}$$

Being x the independent variable for the defined function, x_0 the starting point (x-value) at which the function begins, h_0 the initial value of the function, x_1 the ending point (x-value) at which the function ends, and h_1 the final value of the function.

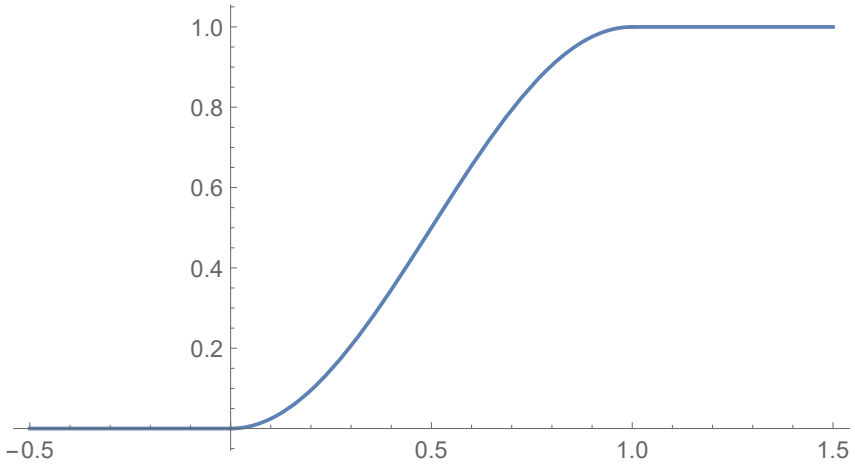


Figure C.6: HAVSIN function plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$.

This function has a continuous first derivative with sharp edges at $x = x_0$ and $x = x_1$ and a discontinuous second derivative, making it a class C^1 function.

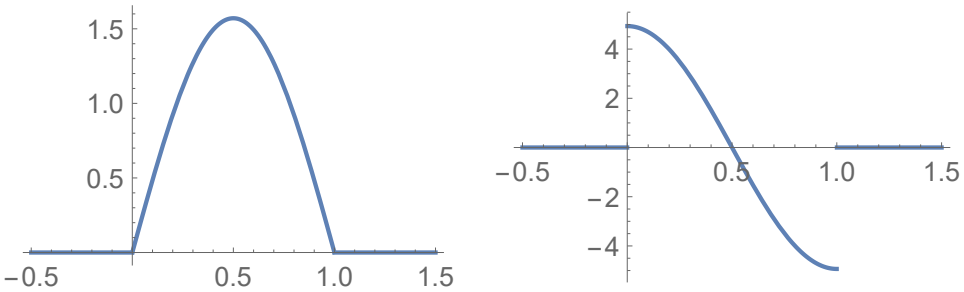


Figure C.7: HAVSIN function derivatives plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$. First derivative on the left and second derivative on the right.

C.4 LOGISTIC function

Another approximation of the Heaviside step function can be obtained using an exponential function, the *logistic function*. This function is defined as:

$$\textbf{LOGISTIC}(x, x_0, h_0, x_1, h_1, k) = \frac{h_1}{1 + e^{-k(x - x_m)}} - h_0 \tag{C.5}$$
$$x_m = (\frac{x_1 - x_0}{2})$$

Being x the independent variable for the defined function, x_m the x-value of the sigmoid's mid-point, x_0 the starting point (x-value) at which the function begins, h_0 the initial value of the function, x_1 the ending point (x-value) at which the function ends, and h_1 the final value of the function. The parameter k is a measure of the steepness of the curve (the bigger this value, the steeper the curve), and its typical value ranges between 5 and 20.

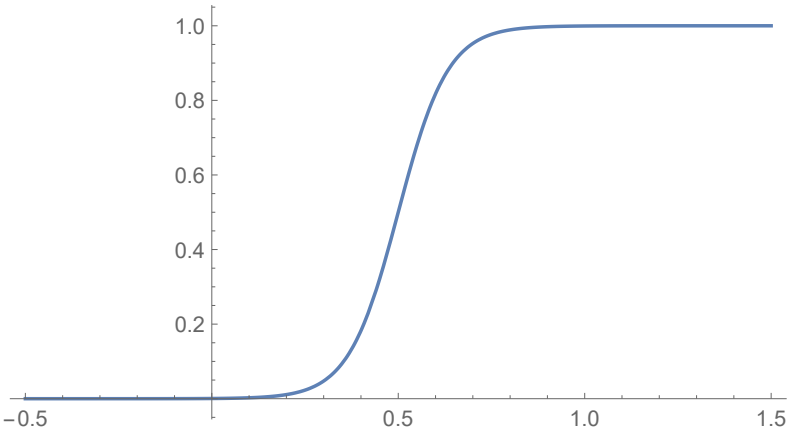


Figure C.8: LOGISTIC function plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$, $h_1 = 1$ and $k = 15$.

Due to its nature, being an exponential function without a piecewise definition, it is a smooth function. A smooth function, also known as a C^∞ function, is a infinitely derivable function that has continuous derivatives for all orders everywhere in its domain.

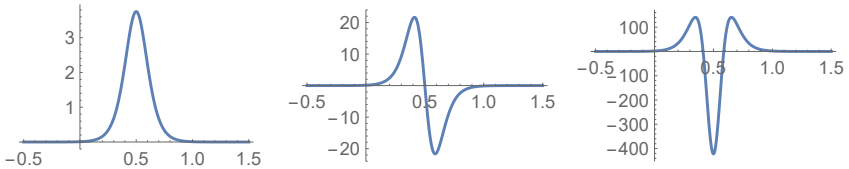


Figure C.9: LOGISTIC function derivatives plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$, $h_1 = 1$ and $k = 15$. First, second and third derivative from left to right.

The logistic function is not really an implemented function in RecurDyn or MSC/ADAMS, but an equivalent can be obtained using the hyperbolic tangent (TANH) function.

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x} = \frac{1}{2} + \frac{1}{2} tanh(\frac{x}{2}) \tag{C.6}$$

C.5 Function comparison

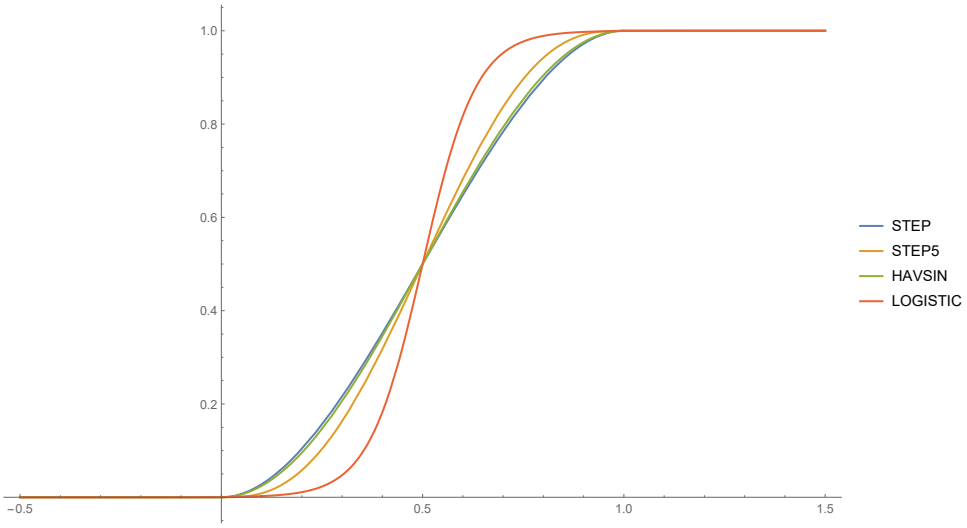


Figure C.10: Step functions plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$.

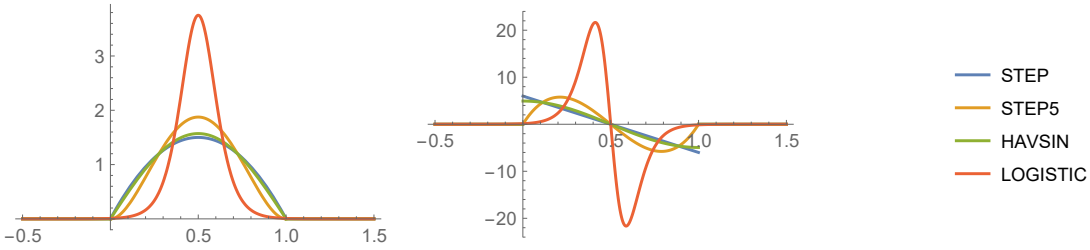


Figure C.11: First and second derivatives of step functions plotted using *Mathematica* for $-\frac{1}{2} \leq x \leq \frac{3}{2}$ with $x_0 = 0$, $x_1 = 1$, $h_0 = 0$ and $h_1 = 1$. First derivative on the left and second derivative on the right.

While all the analytical approximations of the step function are continuous, not all have a smooth derivative. Sharp edges in a first derivative at certain points cause discontinuities at these same points in the second derivative.

STEP and HAVSIN functions have continuous first derivatives with sharp edges at the transition points, causing discontinuous second order derivatives. The STEP5 function has a continuous first and second derivative, but sharp edges at the transition points will cause the third order derivative to be discontinuous. The LOGISTIC function is a special case, with continuous derivatives of any order, but it has stability problems because of the very steep slopes it contains, causing the derivatives to be ever-increasing at a considerable rate.

C.6 Conclusion

Sharp corners found at the transition points in the first derivatives of step function approximations have as a consequence a rapid change in value which can cause difficulties in the numerical integration algorithms of dynamic models. While there was a time in which the STEP function was predominant for describing transitions, the standard has been switched to the STEP5 function. STEP5 function has a continuous second order derivative, solving the issues of smoothness in STEP and HAVSIN functions, and because it is expressed as a polynomial, it is computationally less expensive than the LOGISTIC function.

Function	**Type**	**Smoothness**	**Computational cost**
STEP	Polynomial	C^1	Very low
STEP5	Polynomial	C^2	Low
HAVSIN	Trigonometric	C^1	Medium
LOGISTIC	Exponential	C^∞	High

Table C.1: Interpolating step function comparison.

According to Dr. Juhwan Choi, Chief Product Officer (CPO) at *FunctionBay*[1]: "*Until V8R5, the interpolation type for friction coefficients in contacts entities was defined by various types of functions such as HAVSIN, STEP and STEP5. Each contact entity used a different one for friction coefficient calculation, resulting in different values for contact force, as well as stability problems under some situations. In V9R1, FunctionBay decided to use consistent interpolation type (STEP5) for the friction coefficient evaluation*". It is quite interesting to highlight that this is considered by the *RecurDyn* development team as an upgrade to achieve *consistent* friction, confirming the fact that using a second order smooth function with continuous first and second derivative (class C^2 function) is the way to go.

[1]Extracted from Dr. Juhwan Choi speech at the *RecurDyn Technology Days 2017* event held in Munich, Germany on October 11th 2017.

Chapter D

Kinematics and dynamics of mechanical systems

D.1 Coordinate transformation

Any point on a given rigid body can be represented by a vector in a determined reference frame. In 2-dimensional space, this vector contains two coordinates and it can be represented in polar coordinates or over the Cartesian plane.

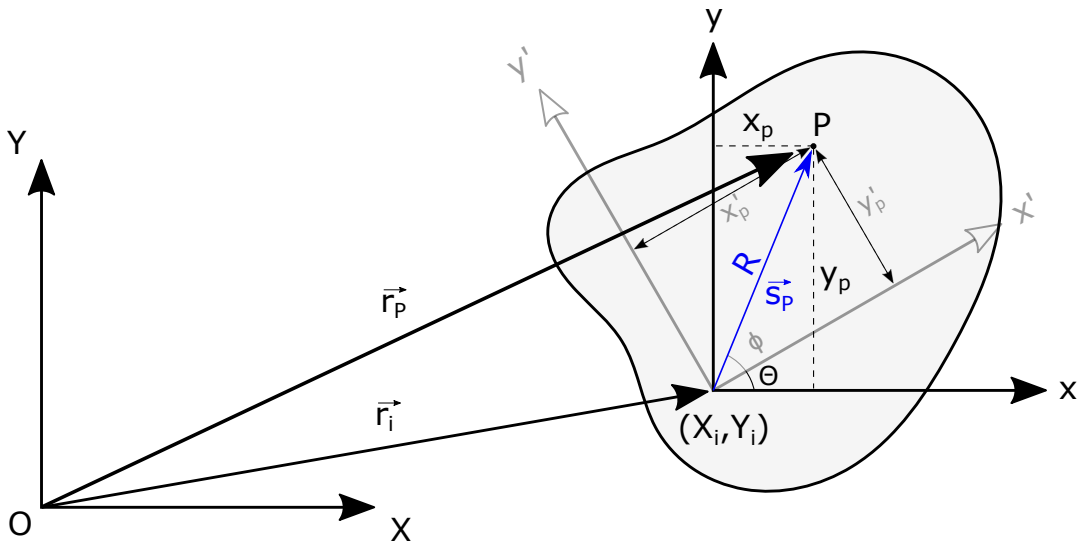


Figure D.1: Representation of a vector in different reference frames.

Considering a point P of a body i, it can be expressed in the local reference frame x'-y' in Cartesian or polar coordinates:

$$\mathbf{s_P'} = \{x_p', y_p'\}^T = \{R\cos\phi, R\sin\phi\}^T \tag{D.1}$$

If another local reference x-y is considered with the same origin as the first one, but with an angle of rotation Θ between the axes, the position vector of the point can be expressed in the new reference frame as:

$$\mathbf{s_P} = \{x_p^{'} \cos\Theta - y_p^{'} \sin\Theta, x_p^{'} \sin\Theta + y_p^{'} \cos\Theta\}^T \tag{D.2}$$

This transformation can be obtained using trigonometry or by using polar coordinates and considering trigonometric addition formulas:

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta \qquad \sin(\alpha+\beta) = \sin\alpha\cos\beta + \sin\beta\cos\alpha \tag{D.3}$$

$$\begin{gathered} \mathbf{s_P} = \{R\cos(\phi+\Theta), R\sin(\phi+\Theta)\}^T = \\ = \{R\cos(\phi)\cos(\Theta) - R\sin(\phi)\sin(\Theta), R\cos(\phi)\sin(\Theta) + R\sin(\phi)\cos(\Theta)\}^T = \\ = \{x_p^{'}\cos\Theta - y_p^{'}\sin\Theta, x_p^{'}\sin\Theta + y_p^{'}\cos\Theta\}^T \end{gathered} \tag{D.4}$$

The original $\{x_p^{'}, y_p^{'}\}$ coordinates, which are now the coefficients of the cosine and sine functions in the x-y coordinate system are the original coordinates of the body with angle of rotation $\Theta = 0$.

The vectors in the two local reference frames are related by the following matrix transformation:

$$\mathbf{s_P} = \mathbf{A}\,\mathbf{s_P^{'}} \tag{D.5}$$

Where $\mathbf{A}$ is the *planar rotation transformation matrix*:

$$\mathbf{A} = \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \tag{D.6}$$

The planar transformation matrix has full rank, therefore it has an inverse matrix and the transformation can be reversed. This matrix is orthogonal and it is easy to check that the inverse matrix is the transpose of the original matrix:

$$\mathbf{A^{-1}} = \mathbf{A^T} = \begin{pmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{pmatrix} \tag{D.7}$$

This means that the rotation transformation can be reversed:

$$\mathbf{s_P^{'}} = \mathbf{A^T}\,\mathbf{s_P} \tag{D.8}$$

Considering an absolute Cartesian reference frame X-Y, the vector that expresses the position of the point of the body in the absolute frame is:

$$\begin{gathered} \mathbf{r_P} = \mathbf{r_i} + \mathbf{s_P} = \mathbf{r_i} + \mathbf{A}\,\mathbf{s_P^{'}} = \\ \{X_i + x_p^{'}\cos\Theta - y_p^{'}\sin\Theta, Y_i + x_p^{'}\sin\Theta + y_p^{'}\cos\Theta\}^T \end{gathered} \tag{D.9}$$

D.2 Equations of motion

There are several ways of obtaining the equations of motion of a mechanical system, all of them being equivalent and providing the same result for the equations of motion.

In this analysis three different ways will be used:

- Newton's Laws
- Lagrangian mechanics
- Constraint equations

Each of them has its advantages and disadvantages. In this case, the three ways will be applied to a mechanical system with two degrees of freedom in order to obtain the differential equations that drive the motion of the system.

D.2.1 Newton's Laws

For a 2-dimensional mechanical system, Newton's laws of motion perform a balance of forces in the two coordinate directions and a balance of torques in the outward axis. These summations are equal to the inertial forces of the body, applied to its center of mass: mass times linear acceleration and moment of inertia times angular acceleration.

$$\begin{aligned} \sum F_x &= m\, a_x = m\, \frac{d^2x}{dt^2} \\ \sum F_y &= m\, a_y = m\, \frac{d^2y}{dt^2} \\ \sum M_z &= I_G\, \alpha = I_G\, \frac{d^2\Theta}{dt^2} \end{aligned} \tag{D.10}$$

It is important to state that the torque equation usually considers torques with respect to the center of mass, because the moment of inertia in this position is usually known. If the torques are taken with respect to another position, one must use the moment of inertia in that exact point transforming the moment of inertia from the center of mass using Steiner's parallel axis theorem:

$$I_P = I_G + m\, d^2 \tag{D.11}$$

Where d is the absolute distance between the center of mass and the considered point.

This method requires a deep analysis of the forces acting in the system, including reaction forces as well as external forces. The most laborious part is that free-body diagrams for each body must be done, identifying and obtaining all the reaction forces in each of the bodies.

D.2.2 Lagrangian mechanics

Instead of analyzing the forces involved in the system as Newton's laws do, Lagrangian mechanics uses the energy of the system to obtain the equations that rule the behavior of the system. The dynamics of the system in terms of energy is summarized by an energy function referred to as the Lagrangian. The Lagrangian is expressed in terms of kinetic (T) and potential (V) energies of the system in the following way:

$$L = T - V \tag{D.12}$$

The kinetic energy of the system must be expressed in terms of the linear and angular velocities of the different bodies. For a system with N bodies:

$$T = \frac{1}{2}\sum_{i=1}^{N} m_i v_i^2 + \frac{1}{2}\sum_{i=1}^{N} I_i \omega_i^2 \tag{D.13}$$

The potential energy of a mechanical system is expressed in terms of its distance from the origin of potential, in case of being under the action of the gravitational field:

$$V = \sum_{i=1}^{N} m_i y_i g \tag{D.14}$$

All the velocities and positions of the bodies in the equations are absolute, and they must be referred to the global coordinate system. The linear and angular velocities of the bodies must be taken at the same point in order to calculate the Lagrangian; they are usually taken at the center of mass, but they can be taken in any point if the moment of inertia is correctly transformed using Steiner's theorem. The origin of potential is also an arbitrary decision, but it is usually taken coinciding with an axis of the global coordinate system.

The degrees of freedom of the system are known as the *generalized coordinates*, and the external non-conservative applied forces are the *generalized forces*.

The equations of motion can be obtained by particularizing the Euler-Lagrange equation for the generalized coordinates of the system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = Q_j \tag{D.15}$$

Being L the Lagrangian of the system, q_j the generalized coordinate j, and Q_j the generalized force associated to the coordinate.

This method provides an equation of motion without having to dissect the whole system and the forces of reaction between the bodies, making it less time consuming and more methodical than Newton's laws of motion.

D.2.3 Constraint equations

The third way of obtaining the equations of motion of a mechanical system is derived from the *principle of virtual work* of a planar body, which states that the virtual work of the unbalanced force $m\ddot{\mathbf{r}} - \mathbf{F}$ and the virtual work of the unbalanced torque $J\ddot{\phi} - T$ is equal to 0.

$$\delta\mathbf{r}^T \left[\mathbf{F} - m\ddot{\mathbf{r}}\right] + \delta\phi \left[T - J\ddot{\phi}\right] = 0 \tag{D.16}$$

This equation is known as the *variational equation of motion of a rigid body* in the plane. The differential equations of motion can be obtained from the previous equation:

$$\begin{aligned} \mathbf{F} - m\frac{d^2\mathbf{r}}{dt^2} &= \mathbf{0} \\ T - J\frac{d^2\phi}{dt^2} &= 0 \end{aligned} \tag{D.17}$$

Where:

$$\mathbf{F} = \begin{pmatrix} F_x \\ F_y \end{pmatrix} \qquad \mathbf{r} = \begin{pmatrix} x \\ y \end{pmatrix}$$

The two terms of the variational equation of motion can be combined in a matrix equation in terms of the virtual displacement of the generalized coordinates, $\mathbf{q}$. The generalized force vector, $\mathbf{Q}$, will be defined as the combination of force and torque applied to the body. The mass matrix, $\mathbf{M} = diag(m, m, J)$, is a diagonal matrix which contains the mass and moment of inertia of the body.

$$\delta\mathbf{q}^T \left[\mathbf{Q} - \mathbf{M}\ddot{\mathbf{q}}\right] = 0 \tag{D.18}$$

Where:

$$\mathbf{q} = \begin{pmatrix} \mathbf{r} \\ \phi \end{pmatrix} \qquad \mathbf{Q} = \begin{pmatrix} \mathbf{F} \\ T \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{pmatrix}$$

The variational equations of motion for each body i in a planar multibody system with N bodies can be summed together to obtain the *system variational equation of motion*:

$$\sum_{i=1}^{N} \delta\mathbf{q_i}^T \left[\mathbf{Q_i} - \mathbf{M_i}\ddot{\mathbf{q}}_\mathbf{i}\right] = 0 \tag{D.19}$$

This summation can be written in a more compact form, similar to the equation of a single body, but with different elements and coefficients:

$$\delta\mathbf{q}^T \left[\mathbf{Q} - \mathbf{M}\ddot{\mathbf{q}}\right] = 0 \tag{D.20}$$

Where:

$$\mathbf{q} = \begin{pmatrix} \mathbf{q_1} \\ \mathbf{q_2} \\ \vdots \\ \mathbf{q_N} \end{pmatrix} \qquad \ddot{\mathbf{q}} = \begin{pmatrix} \ddot{\mathbf{q}}_\mathbf{1} \\ \ddot{\mathbf{q}}_\mathbf{2} \\ \vdots \\ \ddot{\mathbf{q}}_\mathbf{N} \end{pmatrix} \qquad \mathbf{Q} = \begin{pmatrix} \mathbf{Q_1} \\ \mathbf{Q_2} \\ \vdots \\ \mathbf{Q_N} \end{pmatrix}$$

$$\mathbf{M} = diag(\mathbf{M_1}, \mathbf{M_2}, \ldots, \mathbf{M_N})$$

The generalized forces vector, $\mathbf{Q}$, must include both constraint forces (reaction forces in the joints between the bodies) and external applied forces. This makes the equation quite difficult to apply in large multibody systems. This vector can be divided into $\mathbf{Q} = \mathbf{Q^A} + \mathbf{Q^C}$, where the first term denotes the generalized applied forces and the second one the constraint forces acting on the system.

A new expression can be written for a set of virtual displacements $\delta\mathbf{q}$ consistent with constraints, yielding the *constrained variational equations of motion*:

$$\delta\mathbf{q}^T \left[\mathbf{Q^A} - \mathbf{M}\ddot{\mathbf{q}}\right] = 0 \tag{D.21}$$

The kinematic couplings of joints and the independent driving constraints are described as constraint equations for the kinematic analysis, and their combined set of constraints can be written as:

$$\mathbf{\Phi}(\mathbf{q}, t) = \begin{pmatrix} \mathbf{\Phi^K}(\mathbf{q}) \\ \mathbf{\Phi^D}(\mathbf{q}, t) \end{pmatrix} = \mathbf{0} \tag{D.22}$$

The set of virtual displacements $\delta\mathbf{q}$ consistent with constraints must satisfy the equation:

$$\mathbf{\Phi_q}\delta\mathbf{q} = \mathbf{0} \tag{D.23}$$

With $\mathbf{\Phi_q}$ being the Jacobian of the constraint equations.

To complete this approach, the introduction of Lagrange multipliers will allow the reduction of the variational equation of motion to a mixed system of differential-algebraic equations.

The Lagrange multiplier theorem states that a Lagrange multiplier vector $\boldsymbol{\lambda}$ exists such that:

$$\left[\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi_q}^T\boldsymbol{\lambda} - \mathbf{Q^A}\right]^T \delta\mathbf{q} = 0 \tag{D.24}$$

Due to the nature of virtual displacements, the coefficient of $\delta\mathbf{q}$ must be $\mathbf{0}$. The equation obtained is the *Lagrange multiplier form of the constrained equations of motion* for the system:

$$\boxed{\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi_q}^T\boldsymbol{\lambda} = \mathbf{Q^A}} \tag{D.25}$$

If some of the bodies have a center of mass which is not found at their local origin of coordinates, the resulting equations must be modified. The mass matrix changes and a new vector of centrifugal forces is added to the equation. Let the vector $\mathbf{s_G} = \{x_G, y_G\}^T$ be the relative position of the center of mass in the local coordinate system of the body.

The moment of inertia must be displaced from the center of mass to the local origin of coordinates using Steiner's parallel axis theorem: $J_o = J_G + m\,d^2$, being $d = \sqrt{x_G^2 + y_G^2}$ the distance from the center of mass to the local origin.

The mass matrix, $\mathbf{M}$, which is usually a diagonal matrix which contains the mass and moment of inertia of the body, has terms outside of the diagonal if the center of mass of the body is not at its local origin of coordinates. The mass matrix of body i must be modified:

$$\mathbf{M_i} = \begin{pmatrix} m & 0 & m\frac{\partial x_G}{\partial \Theta} \\ 0 & m & m\frac{\partial y_G}{\partial \Theta} \\ m\frac{\partial x_G}{\partial \Theta} & m\frac{\partial y_G}{\partial \Theta} & J_G + m\left(x_G^2 + y_G^2\right) \end{pmatrix} \tag{D.26}$$

Considering the absolute vector of the center of mass $\mathbf{r_G} = \mathbf{r_i} + \mathbf{s_G}$, being $\mathbf{r_i} = \{x_i, y_i\}^T$ the vector from the origin of the absolute reference frame to the local reference frame of the body number i. The terms outside the diagonal are explained from the differentiation done in the first equation of motion.

$$\begin{gathered} \mathbf{F_i} = m\frac{d^2\mathbf{r_G}}{dt^2} \\ \frac{d^2\mathbf{r_G}}{dt^2} = \frac{d^2\mathbf{r_i}}{dt^2} + \frac{d^2\mathbf{s_G}}{dt^2} = \{\frac{d^2x_i}{dt^2} + \frac{d^2x_G}{dt^2}, \frac{d^2y_i}{dt^2} + \frac{d^2y_G}{dt^2}\}^T = \\ = \{\frac{d^2x_i}{dt^2} + \frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 + \frac{\partial x_G}{\partial \Theta}\ddot{\Theta}, \frac{d^2y_i}{dt^2} + \frac{\partial^2 y_G}{\partial \Theta^2}\dot{\Theta}^2 + \frac{\partial y_G}{\partial \Theta}\ddot{\Theta}\}^T \end{gathered} \tag{D.27}$$

The terms of the derivatives of the vector $\mathbf{r_i}$ correspond to the generalized coordinates of the body i and they are contained in the generalized accelerations vector $\ddot{\mathbf{q}}$, but the other two terms derived from the fact that the local position of the center of mass has an offset from this local reference frame must be considered.

The terms multiplying the second derivative of the generalized coordinate $\ddot{\Theta}$ in both x and y coordinates are the ones that appear in the mass matrix out of the diagonal. In this case the terms are:

$$\begin{aligned} \frac{\partial x_G}{\partial \Theta}\ddot{\Theta} &= (-x_G\sin(\phi) - y_G\cos(\phi))\ddot{\Theta} \\ \frac{\partial y_G}{\partial \Theta}\ddot{\Theta} &= (x_G\cos(\phi) - y_G\sin(\phi))\ddot{\Theta} \end{aligned} \tag{D.28}$$

The terms multiplying the first derivative of the generalized coordinate squared $\dot{\Theta}^2$ represent the centrifugal forces, these terms must be included in a new vector denominated as the *centrifugal forces vector* with the following structure:

$$\mathbf{C_i} = \begin{pmatrix} m\frac{\partial^2 x_G}{\partial \Theta^2}\dot{\Theta}^2 \\ m\frac{\partial^2 y_G}{\partial \Theta^2}\dot{\Theta}^2 \\ 0 \end{pmatrix} \tag{D.29}$$

The resulting equation, considering the modification of the mass matrix and the centrifugal term, is the **general Lagrange multiplier form of the constrained equations of motion** for the system:

$$\boxed{\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C} + \mathbf{\Phi_q}^T\boldsymbol{\lambda} = \mathbf{Q^A}} \tag{D.30}$$

Where:

$$\mathbf{M} = diag(\mathbf{M_1}, \mathbf{M_2}, \ldots, \mathbf{M_N})$$

$$\ddot{\mathbf{q}} = \begin{pmatrix} \ddot{\mathbf{q}}_\mathbf{1} \\ \ddot{\mathbf{q}}_\mathbf{2} \\ \vdots \\ \ddot{\mathbf{q}}_\mathbf{N} \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} \mathbf{C_1} \\ \mathbf{C_2} \\ \vdots \\ \mathbf{C_N} \end{pmatrix} \qquad \boldsymbol{\lambda} = \begin{pmatrix} \lambda_\mathbf{1} \\ \lambda_\mathbf{2} \\ \vdots \\ \lambda_\mathbf{N} \end{pmatrix} \qquad \mathbf{Q^A} = \begin{pmatrix} \mathbf{Q_1^A} \\ \mathbf{Q_2^A} \\ \vdots \\ \mathbf{Q_N^A} \end{pmatrix}$$

The resulting system of equations make up a *mixed system of differential-algebraic equations (DAE)*, since no derivatives of the Lagrange multipliers appear.

Now, there are two different ways to continue with the analysis. This system can be directly solved numerically to obtain the solution to all the generalized coordinates and the Lagrange multipliers, or it can be taken further to obtain the equations of motion.

The Lagrange multipliers designated for the driven equations of motion are equal to 0 because these are the unconstrained degrees of freedom of the system, which provides a further simplification on the system. When these unconstrained multipliers are considered to be zero, it is possible to recursively solve the system of the equation for each of all the multipliers until there are a number of equations left with no multipliers remaining to obtain. There equations, which will depend on the previously solved Lagrange multipliers, when substituted, make up the equations of motion of the system. The number of remaining equations must be the same as the number of unconstrained (not driven) degrees of freedom of the system.

This approach might seem more complicated than the previous ones, but it is more easily automatized and more adequate for mechanical systems that have many different bodies. Most computational tools of multibody dynamics that can provide the solution to the motion of any general mechanical system use this method and then solve the resulting differential equations using numerical methods.

D.3 Space state representation

A continuous dynamic system, linear or non-linear and time-invariant or time-variant, can be written in *general space state representation* in the form:

$$\frac{d}{dt}\mathbf{x}(t) = \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \tag{D.31}$$
$$\mathbf{y}(t) = \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t)$$

The first equation is the *state equation*, while the second one is the *output equation*. $\mathbf{x}(t)$ is the state vector formed by the system's dependent variables and their derivatives, $\mathbf{y}(t)$ is the output vector, $\mathbf{u}(t)$ is the input control or independent variable vector, $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix, $\mathbf{C}$ is the output matrix and $\mathbf{D}$ is the feed-through matrix. All the matrix considered are time-invariant, and $\mathbf{g}(\mathbf{x}, \mathbf{u}, t)$ contains all the non-linear and variable coefficient terms. In case of a linear time-invariant (LTI) system, the $\mathbf{g}(\mathbf{x}, \mathbf{u}, t)$ vector is $\mathbf{0}$.

The general representation shows the most general case in which the state equation is non-linear with respect to state $\mathbf{x}(t)$ and control input $\mathbf{u}(t)$ and the output equation depends on the control actions, but most of the times $\mathbf{D} = 0$ and it can be linearized in order to obtain $\mathbf{g}(\mathbf{x}, \mathbf{u}, t) = 0$.

If the system has a total of $\frac{n}{2}$ dependent variables with their $\frac{n}{2}$ derivatives, m control actions and p output variables:

$$\mathbf{x}(t) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \mathbf{u}(t) = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix} \qquad \mathbf{y}(t) = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{pmatrix} \tag{D.32}$$

In the state equation, $\mathbf{A}$ is a $n \times n$ matrix and $\mathbf{B}$ is a $n \times m$ matrix. In the output equation, $\mathbf{C}$ is a $p \times n$ matrix and $\mathbf{D}$ is a $p \times m$ matrix. The non-linear and variable terms are grouped in the vector $\mathbf{g}(\mathbf{x}, \mathbf{u}, t)$ that has $n \times 1$ dimension.

D.4 Space state linearization

In general, all systems are non-linear, but most of the analytical methods for system control can only be applied to linear time-invariant (LTI) systems. The goal is to obtain a linearized version of the non-linear and variable coefficient terms in order to apply the known methods of system analysis in a limited range around some reference state.

Considering the state equation of a non-linear system expressed in general space state form:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \mathbf{g}(\mathbf{x}, \mathbf{u}, t) \tag{D.33}$$

If the dependent state variables are written as deviations from an operating state, $\mathbf{x_o}$ and $\mathbf{u_o}$:

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x_o}(t) + \delta\mathbf{x}(t) \\ \mathbf{u}(t) &= \mathbf{u_o}(t) + \delta\mathbf{u}(t) \end{aligned} \tag{D.34}$$

The resulting expression is:

$$\frac{d}{dt}\mathbf{x_o}(t) + \frac{d}{dt}\delta\mathbf{x}(t) = \mathbf{A}\,\mathbf{x_o}(t) + \mathbf{A}\,\delta\mathbf{x}(t) + \mathbf{B}\,\mathbf{u_o}(t) + \mathbf{B}\,\delta\mathbf{u}(t) + \mathbf{g}(\mathbf{x_o} + \delta\mathbf{x}, \mathbf{u_o} + \delta\mathbf{u}, t) \tag{D.35}$$

The non-linear and variable coefficient terms can be written using first order Taylor series expansion around the operating point:

$$g_i(\mathbf{x}, \mathbf{u}, t) = g_i(\mathbf{x_o}, \mathbf{u_o}, t) + \sum_{j=1}^{N} \left.\frac{\partial g_i}{\partial x_j}\right|_{\mathbf{x_o},\mathbf{u_o}} \delta x_j + \sum_{k=1}^{M} \left.\frac{\partial g_i}{\partial u_k}\right|_{\mathbf{x_o},\mathbf{u_o}} \delta u_k + O((\delta x_j)^2) + O((\delta u_k)^2) \tag{D.36}$$

Disregarding higher order terms and using matrix notation:

$$\mathbf{g}(\mathbf{x}, \mathbf{u}, t) = \mathbf{g}(\mathbf{x_o}, \mathbf{u_o}, t) + \mathbf{J_x}(\mathbf{x_o}, \mathbf{u_o})\delta\mathbf{x}(t) + \mathbf{J_u}(\mathbf{x_o}, \mathbf{u_o})\delta\mathbf{u}(t) \tag{D.37}$$

The matrices $\mathbf{J_x}(\mathbf{x_o}, \mathbf{u_o})$ and $\mathbf{J_u}(\mathbf{x_o}, \mathbf{u_o})$ are referred to as Jacobian matrices and they are evaluated at the selected operating point. Their expressions are:

$$\mathbf{J_x}(\mathbf{x_o}, \mathbf{u_o}) = \left.\begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \frac{\partial g_n}{\partial x_2} & \cdots & \frac{\partial g_n}{\partial x_n} \end{pmatrix}\right|_{\mathbf{x_o},\mathbf{u_o}} \tag{D.38}$$

$$\mathbf{J_u}(\mathbf{x_o}, \mathbf{u_o}) = \left(\begin{array}{cccc} \frac{\partial g_1}{\partial u_1} & \frac{\partial g_1}{\partial u_2} & \cdots & \frac{\partial g_1}{\partial u_n} \\ \frac{\partial g_2}{\partial u_1} & \frac{\partial u_2}{\partial u_2} & \cdots & \frac{\partial g_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial u_1} & \frac{\partial g_n}{\partial u_2} & \cdots & \frac{\partial g_n}{\partial u_n} \end{array} \right) \Bigg|_{\mathbf{x_o}, \mathbf{u_o}} \tag{D.39}$$

Around the operating point, the following expression verifies:

$$\frac{d}{dt}\mathbf{x_o}(t) = \mathbf{A}\,\mathbf{x_o}(t) + \mathbf{B}\,\mathbf{u_o}(t) + \mathbf{g}(\mathbf{x_o}, \mathbf{u_o}, t) \tag{D.40}$$

If these results are substituted into *Equation D.35*, the result is:

$$\frac{d}{dt}\delta\mathbf{x}(t) = [\mathbf{A} + \mathbf{J_x}(\mathbf{x_o}, \mathbf{u_o})]\ \delta\mathbf{x}(t) + [\mathbf{B} + \mathbf{J_u}(\mathbf{x_o}, \mathbf{u_o})]\ \delta\mathbf{u}(t) \tag{D.41}$$

If the coefficient are unified into linear matrices $\mathbf{A}(\mathbf{x_o}, \mathbf{u_o})$ and $\mathbf{B}(\mathbf{x_o}, \mathbf{u_o})$ that depend on the operating point values $(\mathbf{x_o}, \mathbf{u_o})$, the *linearized space state representation* is obtained:

$$\boxed{\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}(\mathbf{x_o}, \mathbf{u_o})\,\mathbf{x}(t) + \mathbf{B}(\mathbf{x_o}, \mathbf{u_o})\,\mathbf{u}(t)} \tag{D.42}$$

Redefining the state and control variables as deviations from the operating point, which is equivalent to $\delta\mathbf{x}(t) \rightarrow \mathbf{x}(t)$ and $\delta\mathbf{u}(t) \rightarrow \mathbf{u}(t)$, and dropping the notation $(\mathbf{x_o}, \mathbf{u_o})$ from the linearized matrices, the resulting representation has the same form as a linear space state:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \tag{D.43}$$

It is important to consider that even though it is not explicit in the simplified representation, a linearized system is always linearized with respect an operating point and the state variables are deviations from the operating point values. This is an important consideration to have in mind when operating with the system, e.g. when implementing state feedback control.

The poles of the linearized system can be obtained in order to check the stability of the open-loop configuration. The poles are the eigenvalues of the linearized system matrix $\mathbf{A}(\mathbf{x_o}, \mathbf{u_o})$ which are the solution to the following matrix equation:

$$\det(s\,\mathbf{I} - \mathbf{A}(\mathbf{x_o}, \mathbf{u_o})) = 0 \tag{D.44}$$

The condition for stability that must verify is that the poles s_i have a negative real part, $Re(s_i) < 0$, this would mean that the system without any controller is stable. If the system has any poles with a positive real part, the system is unstable and it must be controlled using a closed-loop configuration.

Once the system is linearized, some further analysis can be done. State controllability and observability are two important properties of a dynamic system.

The *controllability* of a system determines if the external control input can transfer the internal state of the system from any initial state to another final state in a finite time interval. For linear time-invariant (LTI) systems with n states, the sufficient condition for the controllability of a system is that the controllability matrix, R, has full rank:

$$R = \begin{pmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{pmatrix} \qquad rank(R) = n \tag{D.45}$$

In cases where all the state variables of a system may not be directly measurable, it is necessary to estimate the values of the unknown internal state variables using only the available system outputs. The *observability* of the system addresses this issue. A system is observable if the current state can be determined from the system's outputs in finite time. This means that the behavior of the entire system can be determined from the system's outputs. For linear time-invariant (LTI) systems with n states, the sufficient condition for the observability of a system is that the observability matrix, O, has full rank:

$$O = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix} \qquad rank(O) = n \tag{D.46}$$

The controllability of a system is a requisite when designing control loops such as PID and state feedback controllers, and the observability of a system is required when state observer or state predictor controllers are to be implemented.

D.5 Space state to transfer function

It is possible to obtain the transfer functions of a dynamic system from the space state representation. The *transfer function* of a system is usually represented in the domain of frequency using Laplace transform and it relates a particular input variable to an output variable: expressed how the output behaves with changes in the input or control variables.

Considering the linear time-invariant space state representation:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t) \end{aligned} \tag{D.47}$$

Applying the Laplace transform (with all initial conditions set to zero):

$$\begin{aligned} s\mathbb{X}(s) &= \mathbf{A}\,\mathbb{X}(s) + \mathbf{B}\,\mathbb{U}(s) \\ \mathbb{Y}(s) &= \mathbf{C}\,\mathbb{X}(s) + \mathbf{D}\,\mathbb{U}(s) \end{aligned} \tag{D.48}$$

The transfer function is the quotient between $\mathbb{Y}(s)$ and $\mathbb{U}(s)$. First, it is necessary to obtain $\mathbb{X}(s)$ from the state equation:

$$\begin{aligned} s\,\mathbb{X}(s) - \mathbf{A}\,\mathbb{X}(s) &= \mathbf{B}\,\mathbb{U}(s) \\ (s\,\mathbf{I} - \mathbf{A})\,\mathbb{X}(s) &= \mathbf{B}\,\mathbb{U}(s) \\ \mathbb{X}(s) = (s\,\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B}\,\mathbb{U}(s) &= \mathbf{\Phi}(s)\,\mathbf{B}\,\mathbb{U}(s) \end{aligned} \tag{D.49}$$

$\mathbf{\Phi}(s) = (s\,\mathbf{I} - \mathbf{A})^{-1}$ is known as the *state transition matrix*, which describes how the system changes from the current state to the next.

Substituting in the output equation:

$$\mathbb{Y}(s) = \mathbf{C}\,\mathbf{\Phi}(s)\,\mathbf{B}\,\mathbb{U}(s) + \mathbf{D}\,\mathbb{U}(s) = (\mathbf{C}\,\mathbf{\Phi}(s)\,\mathbf{B} + \mathbf{D})\mathbb{U}(s) \tag{D.50}$$

The *transfer function matrix* is obtained:

$$\mathbb{G}(s) = \frac{\mathbb{Y}(s)}{\mathbb{U}(s)} = \mathbf{C}\,\mathbf{\Phi}(s)\,\mathbf{B} + \mathbf{D} = \mathbf{C}\,(s\,\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} + \mathbf{D} \tag{D.51}$$

Obtaining the transfer function from the space state representation is quite simple because the transfer function form of a system is unique. There are many state space representations of a given system, but all of these representations will result in the same transfer function.

Chapter E

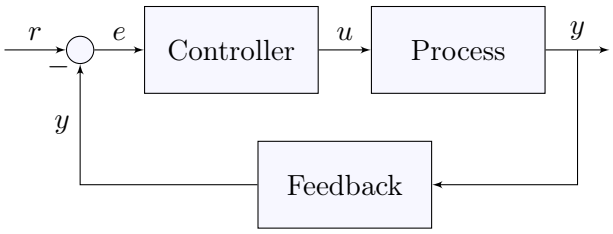
Controller design

E.1 Control theory

The operation of a process is based on the modification of an input variable, called *manipulated variable* or *control variable*, which applied to the input of the system results in the modification of its outputs.

A system with a controller alters this operation scheme, the controller is placed between the input and the process and receives a desired value that represents the output one wants to achieve for the system. The controller receives a feedback from the system, the output variable value is read through a sensor and it is compared to the desired value. The controller is responsible of setting the most appropriate value for the control variable considering the difference between the current operating point of the process and the desired one. The system, which at first was an *open-loop system*, becomes a *closed-loop system*.

(a) System without controller (open-loop)



(b) System with controller (closed-loop)

Figure E.1: Systems comparison.

Where:

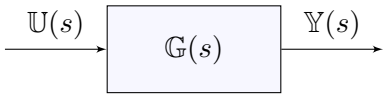
- u is the *manipulated variable* or *control action*.
- y is the output of the system or *process variable*.
- r is the desired or reference value, also known as *setpoint*.
- e is the *error* or difference between the desired value and the system output, $e = r - y$.

There are different types of controllers and different ways to implement them into the system. The fundamental parameters of the controllers can be calculated if the mathematical model of the process is available, or they can be manually or automatically tuned. Depending on the type of controller there are different ways to do so.

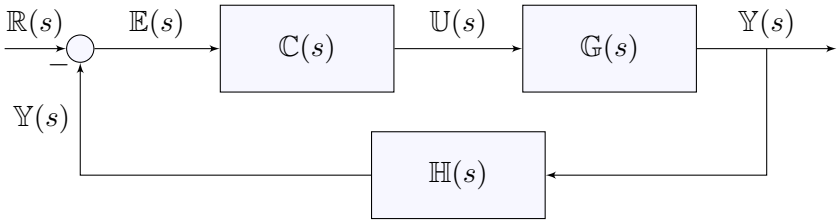
Systems can be classified into different categories depending on the number of inputs and outputs they have. The simplest system is one with only one input and one output, known as a single-input single-output (SISO) system. Systems with many inputs and outputs are known as multiple-input multiple-output (MIMO) systems, and there are also single-input multiple-output (SIMO) and multiple-input single-output (MISO) systems in between.

The analysis of a controlled system is usually done using the Laplace transform on the different variables that intervene in the system. Laplace transform is useful because it provides a way of analytically describing the behavior of the system in the frequency domain, providing relatively simple expressions for the integrals and derivatives of the system's variables.

Applying the Laplace transform to the open-loop and closed-loop SISO systems results in:



(a) Open-loop system



(b) Closed-loop system

Figure E.2: Laplace representation of open and closed-loop systems.

Where all the variables and processes are expressed in terms of their Laplace transforms: $\mathbb{R}(s)$ is the Laplace transform of the setpoint, $\mathbb{E}(s)$ is the Laplace transform of the error, $\mathbb{U}(s)$ is the Laplace transform of the control action, $\mathbb{Y}(s)$ is the Laplace transform of the setpoint, $\mathbb{G}(s)$ is the Laplace transform of the process, $\mathbb{C}(s)$ is the Laplace transform of the controller, and $\mathbb{H}(s)$ is the Laplace transform of the feedback.

Due to the linearity of the Laplace transform, from these representations the transfer functions of the systems can easily be obtained. Transfer functions provide a representation of how changes

in the input variable affect the output variable, they are expressed in Laplace transform as the quotient between output and input variables.

In the open-loop configuration it is trivial to obtain the *open-loop transfer function*:

$$\frac{\mathbb{Y}(s)}{\mathbb{U}(s)} = \mathbb{G}(s) \tag{E.1}$$

For the closed-loop configuration a *closed-loop transfer function* can also be obtained:

$$\begin{aligned} \mathbb{Y}(s) &= (\mathbb{R}(s) - \mathbb{Y}(s)\mathbb{H}(s))\,\mathbb{C}(s)\mathbb{G}(s) \\ \mathbb{Y}(s)(1 + \mathbb{C}(s)\mathbb{G}(s)\mathbb{H}(s)) &= \mathbb{R}(s)\mathbb{C}(s)\mathbb{G}(s) \\ \mathbb{Y}(s) &= \frac{\mathbb{R}(s)\mathbb{C}(s)\mathbb{G}(s)}{1 + \mathbb{C}(s)\mathbb{G}(s)\mathbb{H}(s)} \\ \frac{\mathbb{Y}(s)}{\mathbb{R}(s)} &= \frac{\mathbb{C}(s)\mathbb{G}(s)}{1 + \mathbb{C}(s)\mathbb{G}(s)\mathbb{H}(s)} \end{aligned} \tag{E.2}$$

In order to obtain the dynamic behavior of the system in terms of changes in its input, one must analyze the poles of its transfer function. The poles are the values of s that make the denominator of the transfer function equal to zero, these values are the ones that determine the dynamic response and the stability of the system. The equation that sets the denominator of the transfer function to zero is known as the *characteristic equation* of the system. The resulting polynomial expression in terms of s, which zeros are the poles of the system, is known as the *characteristic polynomial*. The order of the polynomial corresponds with the number of poles of the system. s is a complex variable, therefore it will have real and imaginary parts. Considering a linear time-invariant system, for it to be stable all of the poles of its transfer function must have negative real values (the real part of each pole must be less than zero), which means that the transfer function complex poles must reside in the open left half of the complex plane.

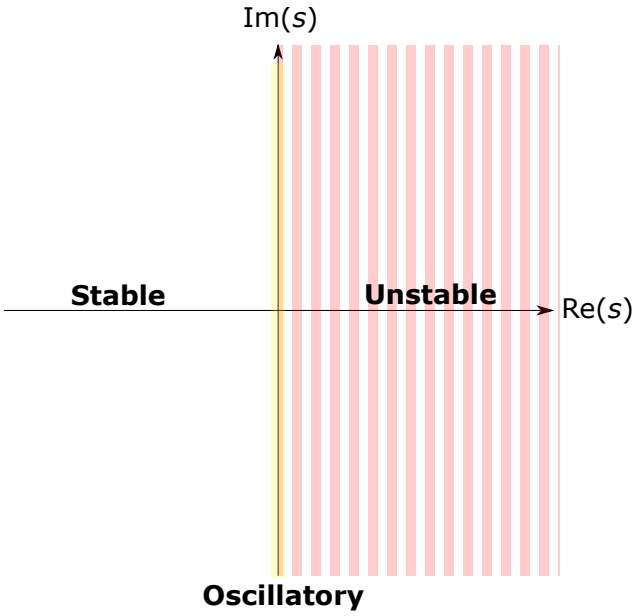


Figure E.3: Dynamic behavior of the system in terms of the positions of its pole in the complex plane.

For the open-loop transfer function, the characteristic equation is:

$$\frac{1}{\mathbb{G}(s)} = 0 \tag{E.3}$$

The characteristic equation of the closed-loop transfer function is:

$$1 + \mathbb{C}(s)\mathbb{G}(s)\mathbb{H}(s) = 0 \tag{E.4}$$

The closed-loop transfer function analysis can also be extended to multiple-input multiple-output systems with feedbacks, in which the characteristic equation can be obtained from the determinant of a matrix equation:

$$\det(\mathbf{I} + \mathbf{C}(s)\mathbf{G}(s)\mathbf{H}(s)) = 0 \tag{E.5}$$

In this case transfer functions have additional terms from the cross-influence of the other inputs of the system.

E.2 PID Controller

A proportional-integral-derivative controller, known as *PID* or *three-term controller*, is a closed-loop control mechanism that uses the desired value, or setpoint, and the actual value of a process variable received through feedback, to calculate the error and through automatic adjustment of the control variable tries to minimize it. The control action is adjusted according to three different parameters: the proportional action, which depends on the current error; the integral action, which takes into account the accumulation of previous errors; and the derivative action, which is a prediction of future errors according to the current error trend line.

Equation E.6 shows the equation that describes the standardized *PID* controller that transforms the error $e(t)$ into a control variable $u(t)$; where K_p is the *proportional gain*, T_i is the *integral time*, and T_d is the *derivative time*.

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de(t)}{dt} \right) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{E.6}$$

Where:

$$K_i = \frac{K_p}{T_i} \qquad K_d = K_p T_d \tag{E.7}$$

Applying the Laplace transform to this expression will yield the transfer function of the controller that will have the form indicated in *Equation E.8*. This expression is useful when the transfer function of the model is known and the expression for the transfer function of the whole closed-loop system wants to be obtained.

$$\mathbb{C}(s) = \frac{\mathbb{U}(s)}{\mathbb{E}(s)} = K_p + \frac{K_i}{s} + K_d s = K_p \left(\frac{T_d s^2 + s + \frac{1}{T_i}}{s} \right) = \frac{K_d s^2 + K_p s + K_i}{s} \tag{E.8}$$

E.2.1 Influence of the different terms

E.2.1.1 Proportional term

The proportional term $K_p\, e(t)$ depends only on the current value of the error between the setpoint and the process variable. Increasing the proportional gain will increase the speed of the control system response, as the reaction will be harder, but increasing the gain too much can cause oscillations in the process variable or even cause the system to become unstable. The increase of the proportional term reduces the steady-state error, but it causes a bigger overshoot. The *steady-state error* is the difference between the process variable and the setpoint once the controller action has stabilized (after a sufficient amount of time). An *overshoot* occurs when the output of the closed-loop system that is increasing or decreasing to get to the setpoint, surpasses the target value; the amount of desirable overshoot is a parameter to have in mind.

E.2.1.2 Integral term

The integral term $K_i \int_0^t e(\tau)\, d\tau$ sums the error over time, causing that even a small deviation will make the integral component increase. This means that unless the error is zero, the integral response will continually increase over time, causing the response of the system to have a zero steady-state error. It can make the system more slow (and even oscillatory) since when the error signal changes sign, it may take a while for the integrator to unwind (due to the sum of previous error). If the control action has a limitation and the integral action causes it to saturate, there is also a phenomenon of windup due to the accumulation of incorrigible error that must be compensated. Some controllers might need an *anti-windup* loop to compensate this phenomenon.

E.2.1.3 Derivative term

The derivative term $K_d \frac{de(t)}{dt}$ depends on the rate of change of the error, which has the opposite sign than the process variable. This term tries to anticipate error. If the process variable is increasing rapidly, the derivative component will cause the control action to decrease. Increasing the derivative time parameter will cause the controller to react more strongly to changes in the error and increasing the speed of response. It also tends to add damping to the system, decreasing overshoot. The problem with the derivative term is that it is very sensitive to small fluctuations or noise in the process variable feedback signal. If the signal is noisy or the control loop rate is too slow, the derivative term can cause the system to be unstable, this is the reason why most real world control systems have a small derivative term or none at all.

E.2.1.4 Considerations

Parameter	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**	**Stability**
$K_p \uparrow$	Decrease	Increase	Small change	Decrease	Decrease
$K_i \uparrow$	Decrease	Increase	Increase	Decrease	Decrease
$K_d \uparrow$	Small change	Decrease	Decrease	No change	Increase

Table E.1: Response of the closed-loop system when increasing the proportional, integral and derivative terms.

There are some considerations to make regarding the integral and derivative terms. The first is the implementation of a *anti-windup* mechanism in the accumulation of the integral error. This is useful when the control variable saturates and the error accumulates without the controller being able of correcting it. There are several ways of addressing this issue. The first way is by disabling integration when the output saturates. Another popular way is by using back-calculation, in which the difference between the control variable value and the saturated value is subtracted to the integral term with a gain of $1/T_{aw}$. The value of this gain determines how quickly the integral term is reset. There is also a problem with the derivative term when the setpoint is changing. The derivative term is the slope of the error value, which is the difference between the setpoint and the output of the system. If the setpoint changes drastically (switches from one value to another in a step) the slope of the error can be very large causing very large derivative action, this is known as *derivative kick*. The solution to this phenomenon is to take the negative derivative of the process variable, instead of the derivative of the error. This will make the derivative term independent of changes in the setpoint.

E.2.2 Tuning

Tuning is referred to as the process of obtaining the optimal proportional (K_p), integral (K_i) and derivative (K_d) gains that get an ideal response from the closed-loop system. There are different ways of tuning a PID controller: manually, heuristically or by pole placement using transfer function analysis. Manual and heuristic tuning are the most popular tuning techniques and they are valid when the transfer function of the process is unknown. The *transfer function* of a process is the mathematical expression, usually in Laplace domain, of how the process behaves to a change in the control action: $\frac{Y(s)}{U(s)}$.

E.2.2.1 Manual tuning

The most common way of tuning the controller gains is by manually using the *trial and error method*. First, the integral and derivative terms are set to zero and the proportional gain is increased until it has a fast enough response. Depending on the system, the response might be oscillatory. Second, the integral term is increased to stop the oscillations and to reduce the steady state error. This term must be tweaked to achieve a minimal steady-state error while having a reduced overshoot. Once the values of the proportional and integral term have been set, the derivative term is increased to reduce overshoot and increase stability, but makes the system more sensitive to noise.

There are several heuristic methods to tune a PID controller depending on the nature of the system. There are two main categories: open-loop and closed-loop methods. These methods receive the name of the engineers that described them using empirical results in the 1940s, 50s and 60s. Heuristic methods are not guaranteed to be optimal, most times they serve as a starting point that with a little tweaking can provide a desirable response.

Open-loop methods are suitable for stable self-regulating processes. The controller is tuned by analyzing the characteristics of *open-loop step response* (when a step is applied to the control action, u, working in open-loop). The process must exhibit a S-like curve response to the step action, the nature of the processes are usually first order processes with no integrators or processes that can be approximated to first order processes with a delay. The most popular open-loop methods are Ziegler-Nichols (open-loop variant), Cohen-Coon and CHR (Chien, Hrones and Reswick).

In closed-loop methods, the controller is tuned by analyzing the response of the closed-loop system. This methods are applicable to unstable open-loop systems that become stable in closed-loop configuration, and other higher order systems. The system must exhibit sustained oscillations when the proportional term is big enough in order for this method to be applied. The most popular is the Ziegler-Nichols closed-loop method, also known as the Ziegler-Nichols ultimate gain method. There are other methods, but most of them are based on the work done by Ziegler and Nichols in 1942 with different parameters or specifications.

Ziegler-Nichols ultimate gain method is based on a $\frac{1}{4}$ decay ratio design criterion for acceptable stability, this means that the ratio of the amplitudes of subsequent peaks of the system response to a step change in set-point or disturbance is approximately $\frac{1}{4}$. The resulting ratio after tuning the controller using this method might not be this exact value, but it should be close to it.

The Ziegler-Nichols closed-loop method is very systematic and easy to apply. First, the integral and derivative gains are set to zero, and the proportional gain is increased until the output begins to oscillate. This oscillations mean that the system is on the stability limit, and the proportional gain value that produces this is denoted as the *ultimate (or critical) gain*, K_u. If there is no value of K_p that makes the system oscillate this method is not applicable. The ultimate gain value must be found without any saturation occurring in the control signal, it must be the smallest K_p value that drives the control loop into sustained oscillations. Second, the period of the oscillations must be measured, this parameter is known as the *ultimate (or critical) period* T_u. With the ultimate gain and ultimate period parameters, there are empirical formulas to obtain the controller's parameters, depending on the type and desired characteristics.

Controller type	$\mathbf{K_p}$	$\mathbf{T_i}$	$\mathbf{K_i}$	$\mathbf{T_d}$	$\mathbf{K_d}$
P	$0.5K_u$	-	-	-	-
PI	$0.45K_u$	$T_u/1.2$	$0.54K_u/T_u$	-	-
PID	$0.6K_u$	$T_u/2$	$1.2K_u/T_u$	Tu/8	$0.075K_uT_u$
some overshoot	$0.33K_u$	$T_u/2$	$0.66K_u/T_u$	$T_u/3$	$0.11K_uT_u$
no overshoot	$0.2K_u$	$T_u/2$	$0.4K_u/T_u$	$T_u/3$	$K_uT_u/15$
Pessen Integral Rule	$0.7K_u$	$T_u/2.5$	$1.75K_u/T_u$	$3T_u/20$	$0.105K_uT_u$

Table E.2: Empirical formulas for the parameters of the different types of controllers obtained through the Ziegler-Nichols closed-loop method.

E.2.2.2 Pole placement

If the transfer function of the model is available, it is possible to apply more sophisticated methods of tuning such as the pole placement method. This method provides the parameters of the PID controller according to previously defined specifications.

The dynamic behavior of the closed-loop system is determined by the poles of the transfer function. The *pole placement method* consists in choosing the poles the closed-loop transfer function should have and calculating the controller parameters needed to obtain these poles.

The poles are chosen according to the dynamic specifications desired for the system, usually by setting the settling time T_{set} and the maximum percentage overshoot PO (in %). The term *overshoot* refers to the phenomenon when the output signal exceeds its reference value during the first control cycle, only to come back again and most times oscillate until it reaches its final state (if the resulting closed-loop system has an integral term, the final state coincides with the reference). The *settling time* is defined as the time elapsed until the output enters and remains in a specified error band around its final state, usually the error band taken consists of a 2% deviation from the final value.

The characteristic equation of a linear system is expressed as a polynomial equal to zero, where the polynomial is the characteristic polynomial. It is usually expressed as a monic polynomial, where all the coefficients have been divided by the leading one.

$$\lambda(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0 = 0 \tag{E.9}$$

The fundamental theorem of algebra states that every non-constant single-variable polynomial of degree n has n complex roots (real numbers are included in the field of complex numbers). The characteristic polynomial can therefore written as the product of its n poles:

$$\lambda(s) = \prod_{i=1}^{n}(s - p_i) = (s - p_n)(s - p_{n-1}) \cdots (s - p_1) = 0 \tag{E.10}$$

The objective of these method is to construct a polynomial of the same degree as the characteristic polynomial of the closed-loop system that contains the desired poles. These polynomials are then compared coefficient-wise in order to obtain the parameters of the controller that force the closed-loop system to have the desired dynamics.

Considering closed-loop functions with of second order or higher, the procedure is always the same. The first two poles are set using the desired overshoot and settling time the system must have. These will be the *dominant poles* that rule the dynamics of the system.

These two poles will form a pair of complex conjugate poles, which is equivalent to the expression of a second order system:

$$s^2 + 2\xi\omega_n s + \omega_n^2 = (s - (-\sigma + i\,\omega_d))(s - (-\sigma - i\,\omega_d)) \tag{E.11}$$

Where the real and imaginary part of the poles are expressed in terms of the damping ratio ξ and the natural frequency ω_n of the system: $\sigma = \xi\omega_n$ and $\omega_d = \omega_n\sqrt{1-\xi^2}$

From the definition of percentage overshoot PO, the damping ratio ξ can be obtained:

$$PO\,[\%] = 100\,\exp\left(\frac{-\pi\xi}{\sqrt{1-\xi^2}}\right) \qquad \rightarrow \qquad \xi = \frac{-ln\frac{PO}{100}}{\sqrt{\pi^2 + ln^2\frac{PO}{100}}} = \cos\varphi \tag{E.12}$$

From the approximation for the settling time for a 2% deviation, the natural frequency ω_n is obtained:

$$T_{set} = \frac{4}{\omega_n\,\xi} \qquad \rightarrow \qquad \omega_n = \frac{4}{T_{set}\,\xi} \tag{E.13}$$

The resulting poles are:

$$s_{1,2} = -\sigma \pm i\,\omega_d = -\xi\omega_n \pm \omega_n\sqrt{1-\xi^2} \tag{E.14}$$

Damping ratio is what makes the poles have an imaginary part, $\xi = \cos\varphi$, depending on the damping ratio the system can be: undamped ($\xi = 0$), poles with only imaginary part that provide an oscillatory response; underdamped ($\xi < 1$), complex poles that provide a decaying oscillatory response; or critically damped ($\xi = 1$) poles with only real part that provide no oscillatory response.

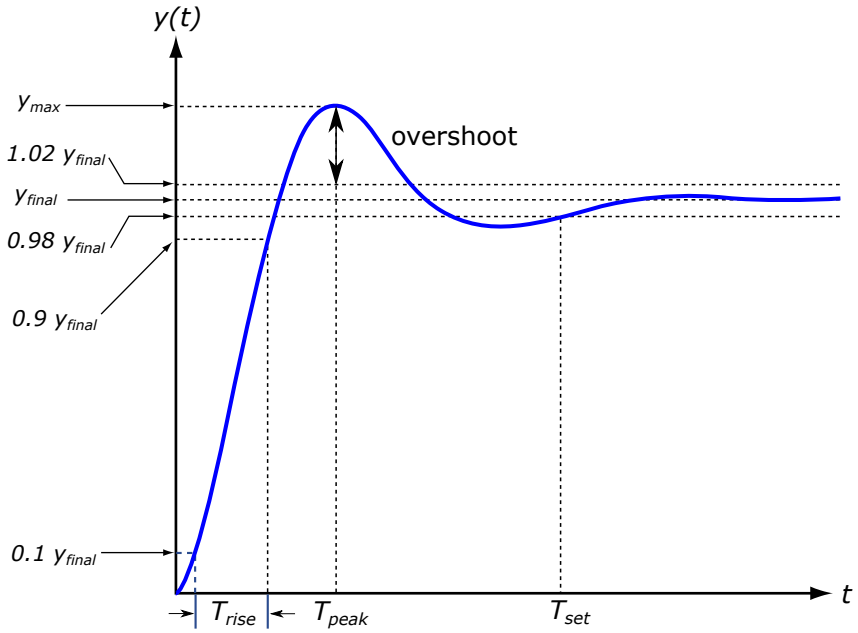


Figure E.4: Time response of an underdamped second order system.

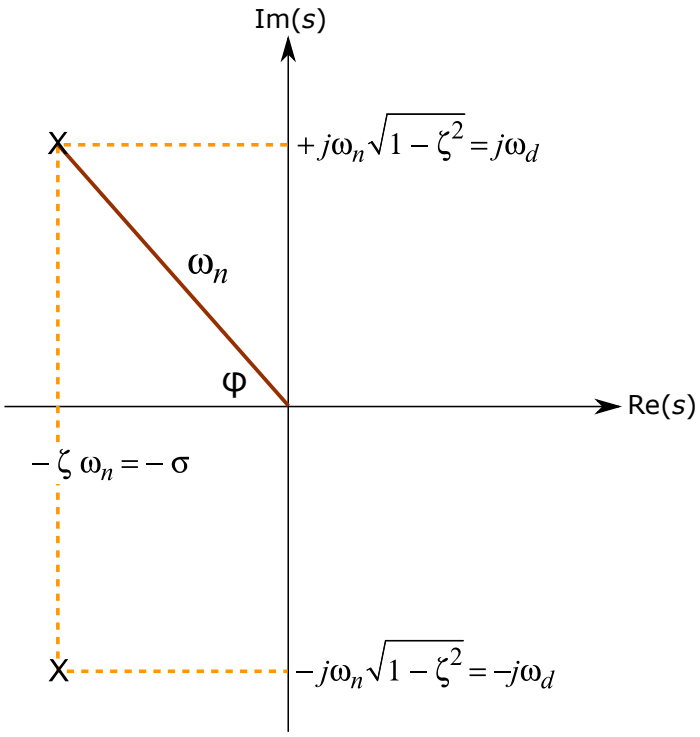


Figure E.5: Pole location for a second order system in terms of damping ratio ξ and natural frequency ω_n.

The rest of the poles until the crafted polynomial is of the same degree as the closed-loop system polynomial are chosen to be negative real numbers with a much more faster dynamics than the conjugate poles. The farther away the poles are from the imaginary axis, the faster their effect on the system is settled. The poles closest to the imaginary axis, the so called *slow* poles, are the dominant poles.

As a rule of thumb, the remaining poles are chosen to be at least twice as faster as the dominant poles. Considering the settling time for a first order system (a system with only one real pole, $p = -\sigma$) as approximately $T_{set} = 4/\sigma$, the remaining poles will be multiples of the real part of the complex conjugate poles $p_i = -k_i \cdot \sigma$, where $k_i \geq 2$. These poles will be k_i times faster than the original poles with $Re(s) = -\sigma$, but the counterpart is that the more this poles move to the left of the s-plane, the more control effort will it take for the system to stabilize.

The crafted polynomial must be then compared to the closed-loop polynomial to obtain the coefficients that cause this behavior con the system.

$$\begin{gathered} \lambda(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0 = 0 \\ \lambda'(s) = (s - (-\sigma + i\,\omega_d))(s - (-\sigma - i\,\omega_d))(s - (-k_1\sigma)) \cdots (s - (-k_{n-2}\sigma)) = 0 \\ \lambda(s) = \lambda'(s) = 0 \end{gathered} \tag{E.15}$$

The pole placement method is a really powerful method and allows the molding of the system to the desired specifications, the only consideration is that a very fast settling time for the dominant poles (i.e. a very high bandwidth ω_n) will cause a very aggressive control action. The poles must be located taking this into consideration.

Because all poles are being placed, the dominant second order behavior is pretty much guaranteed to be valid and all the desired specifications (settling time and overshoot) should be observed in the output of the controlled system.

$$\begin{gathered} T_{set} = \frac{4}{\omega_n\,\xi} \qquad PO\,[\%] = 100\,\exp\left(\frac{-\pi\xi}{\sqrt{1-\xi^2}}\right) \\ T_{rise} = \frac{1 + 1.1\xi + 1.4\xi^2}{\omega_n} \qquad T_{peak} = \frac{\pi}{\omega_n\sqrt{1-\xi^2}} \end{gathered} \tag{E.16}$$

E.2.3 Discrete approximation

In order to apply this type of controller in an electronic system, e.g. in a microcontroller or digital signal processor (DSP), it must be discretized. The control loop will not be applied in a continuous fashion in time, but in finite intervals instead; intervals determined by the type of electronic system used and its frequency of operation. The discretization is achieved by using the *finite difference method* to approximate the value of the derivative term and approximating the integral as a sum, being Δt or T_s the sampling time, which is the interval of time elapsed between each of the times in which the control will be applied to the movement.

$$\begin{aligned} \int_0^{t_k} e(\tau)\, d\tau &= \sum_{i=1}^{k} e(t_i)\Delta t \\ \frac{de(t_k)}{dt} &= \frac{e(t_k) - e(t_{k-1})}{\Delta t} \end{aligned} \tag{E.17}$$

The resulting equation is the formula that must be implemented in the electronic system.

$$u(t_k) = K_p \left(e(t_k) + \frac{T_s}{T_i} \sum_{i=1}^{k} e(t_i) + \frac{T_d}{T_s}(e(t_k) - e(t_{k-1})) \right) \tag{E.18}$$

It must be noted that the proportional term is independent from the sampling time T_s, but the integral and derivative terms are not. It is important to have a stable value for sampling time in order for a complete PID control to be implemented.

The problem with discrete controllers in electronic devices, where there is always some noise in the signal acquisition, e.g. the analog to digital converters have conversion errors. This causes the derivative term to be quite unstable in many practical implementations and that it might be necessary to set a threshold for the error in order to avoid control action when the deviation is small.

E.3 State feedback control

Considering a linear time-invariant system or a non-linear system that has been linearized and assuming that all the components of the state vector can be measured and are accessible, it is possible to design a controller that stabilizes the closed-loop system by using linear feedback from the states of the system using the K matrix. This is what is known as *linear full state feedback control*.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t) \end{aligned} \tag{E.19}$$

Where the control action is: $\mathbf{u}(t) = \mathbf{r}(t) - \mathbf{K}\,\mathbf{x}(t)$.

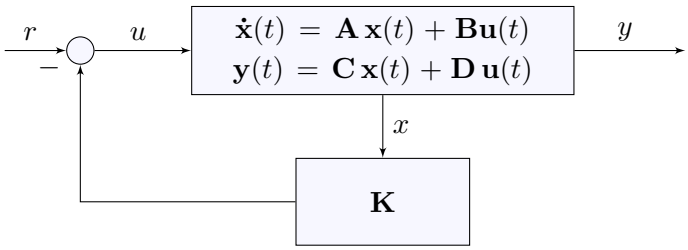


Figure E.6: Full state feedback control system.

Considering the dimensions of the vectors, $\mathbf{x}(t)$, $\mathbf{u}(t)$ and $\mathbf{K}$ must have compatible lengths:

$$\mathbf{x}(t) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \mathbf{u}(t) = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix} \qquad \mathbf{K} = \begin{pmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{2n} \\ \dots & \dots & \ddots & \vdots \\ k_{m1} & k_{m2} & \dots & k_{mn} \end{pmatrix} \tag{E.20}$$

The $\mathbf{K}$ is a $m \times n$ matrix with one proportional term per state and control variable.

For linearized systems, it is very important to consider that the state variables vector $\mathbf{x}(t)$ is not the absolute values of these states, but the deviation from the linearized state (usually the equilibrium point). In order for the feedback controller to achieve a zero final state error, the proportional matrix $\mathbf{K}$ must be really applied to the deviation state vector $\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_o(t)$.

Substituting the control action into the equations of the system yields:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= (\mathbf{A} - \mathbf{B}\,\mathbf{K})\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{r}(t) \\ \mathbf{y}(t) &= (\mathbf{C} - \mathbf{D}\,\mathbf{K})\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{r}(t) \end{aligned} \tag{E.21}$$

The poles of the resulting closed-loop system are the eigenvalues of the coefficient matrix that goes with the state vector, that matrix in this case is: $\mathbf{A} - \mathbf{B}\,\mathbf{K}$. This means that it is independent of reference value $\mathbf{r}(t)$ for each one of the states.

$$\lambda(s) = \det(s\mathbf{I} - (\mathbf{A} - \mathbf{B}\,\mathbf{K})) = 0 \tag{E.22}$$

The controllability of the system pair $(\mathbf{A}, \mathbf{B})$ must be considered. If the system is controllable, the eigenvalues of the characteristic polynomial $\lambda(s)$ can be placed arbitrarily such that the closed-loop poles are in the desired locations.

Each one of the k_i coefficients will multiply the value of each one of the x_i states, and the problem of how are these coefficients determined must be addressed. There are two ways of doing so: the first way is using the pole placement method and the second way is using a linear quadratic regulator.

E.3.1 Pole placement

In a similar way as it is done with PID controllers, the *pole placement method* consists in crafting a polynomial with the dynamic characteristics desired $\lambda'(s)$ for the system and equaling its terms to the resulting characteristic polynomial obtained from the system $\lambda(s)$ to obtain the coefficients $\mathbf{K}$ matrix.

The selection of the dynamics of the resulting closed-loop system $\lambda'(s)$ is done by a second order approach with the remaining poles being much faster that the second order ones, which are the dominant ones.

If the system is controllable, *Ackermann's formula* calculates the $\mathbf{K}$ matrix which places the poles of the closed-loop system in desired locations given by the chosen desired performances.

$$\mathbf{K} = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \mathbf{R}^{-1} \lambda'(\mathbf{A}) \tag{E.23}$$

The first vector is the last row of the $n \times n$ identity matrix. $\mathbf{R}$ is the controllability matrix, and it is clear that the system must be controllable because the inverse of the controllability matrix must be calculated. $\lambda'(\mathbf{A})$ is the desired characteristic polynomial of the closed-loop system considered as a matrix function and evaluated for the matrix $\mathbf{A}$.

E.3.2 Linear-Quadratic Regulator (LQR)

Considering a linear time-invariant system in which the dynamics are described by the group of linear differential equations:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\,\mathbf{x}(t) + \mathbf{D}\,\mathbf{u}(t) \end{aligned} \tag{E.24}$$

The objective is to place the poles of the closed-loop system so that the system minimizes the quadratic cost function:

$$J_{LQR} = \int_0^\infty \left(\mathbf{x}(t)^T\,\mathbf{Q}\,\mathbf{x}(t) + \mathbf{u}(t)^T\,\mathbf{R}\,\mathbf{u}(t) \right) dt \tag{E.25}$$

Where $\mathbf{x}(t)^T\,\mathbf{Q}\,\mathbf{x}(t)$ is the *state cost* with weight $\mathbf{Q}$ and $\mathbf{u}(t)^T\,\mathbf{R}\,\mathbf{u}(t)$ is the *control cost* with weight $\mathbf{R}$. No cross-coupling term $\mathbf{N}$ between state and control is considered. The conditions for the weighting matrices are $\mathbf{R} > 0$ and $\mathbf{Q} \geq 0$. This is the basic form of the infinite-horizon *linear-quadratic regulator* (LQR) problem.

The solution to this problem is given by the time-invariant linear state feedback control law:

$$\begin{aligned} \mathbf{u}(t) &= -\mathbf{K_{LQR}}\,\mathbf{x}(t) \\ \mathbf{K_{LQR}} &= \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \end{aligned} \tag{E.26}$$

The matrix $\mathbf{P}$ is the solution to the Riccati algebraic equation:

$$\mathbf{0} = \mathbf{P}\,\mathbf{A} + \mathbf{A}^T\mathbf{P} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} \tag{E.27}$$

In order to assign the values of $\mathbf{Q}$ and $\mathbf{R}$, the signals are normalized according to the maximum admissible deviations using *Bryson's rules*.

$$\begin{aligned} \mathbf{Q} &= \begin{pmatrix} q_1 & & \\ & \ddots & \\ & & q_n \end{pmatrix} \qquad \mathbf{R} = \rho \begin{pmatrix} r_1 & & \\ & \ddots & \\ & & r_m \end{pmatrix} \\ q_{ii} &= \frac{\alpha_i^2}{(x_i)_{max}} \qquad r_{jj} = \frac{\beta_j^2}{(u_j)_{max}} \end{aligned} \tag{E.28}$$

In this case the quadratic cost function becomes:

$$J_{LQR} = \int_0^\infty \left(\sum_{i=1}^{n} q_{ii} x_i(t)^2 + \rho \sum_{j=1}^{m} r_{jj} u_j(t)^2 \right) dt \tag{E.29}$$

The values of $(x_i)_{max}$ and $(r_i)_{max}$ represent the maximum acceptable value of the deviation from the equilibrium point and the maximum acceptable value for the control variable in equilibrium. This is equivalent to scaling the variables such that when they reach their maximum acceptable value the component of the diagonal is 1. $\sum_i \alpha_i^2 = 1$ and $\sum_j \beta_j^2 = 1$ are used to add relative internal weights between the state and control components. The ρ parameter provides a relative weighting between the control and state penalties to compare the relative importance between them. A bigger value will require less control action to minimize the cost function, while a small value will incentive a bigger control action disregarding the state variables.

Although Bryson's rules usually gives good results, it might require an additional trial-and-error iterative design procedure in order to obtain the desirable properties for the closed-loop system. The value of the ρ parameters, and possible the values of the $\mathbf{Q}$ and $\mathbf{R}$ matrices, must be tuned to obtain the ideal solution.

Comparing this method with the pole placement which uses a dominant second order approach where the closed-loop pole locations are placed with no regard to the amount of control effort required, the LQR solution selects the closed-loop poles finding a balance between state errors and control effort. This design method is focused on system performance rather than on the mechanics of the design process.

Once the space state representation of the system is obtained and the matrices and parameters set, the solution to the LQR problem must be obtained numerically. The *Mathematica* function that does this is **LQRegulatorGains**.

Chapter F

Considerations for the Modelica model

F.1 State variables

The *state variables* of a dynamic system are the set of variables within the equations of the system that can be used to describe the state of the system unmistakably. The behavior of the whole system can be determined for any moment of time for which the values of the state variables are known. The state variables depend on the nature of the system: in mechanical systems, state variables are usually position and velocities of the bodies that make the system; in electrical circuits, the state variables are the voltages in nodes and the currents going through the components; and in thermodynamic systems, state variables can include temperature, pressure and internal energy. State variables are those that are found differentiated at least once within the model equations, for example, position, velocity and acceleration. The differential equations involving state variables and their derivatives are the mathematical equations that describe the system and its behavior in the domain of time.

Sometimes different combinations of state variables exist. In this case, *SystemModeler* uses *dynamic state selection* to choose the appropriate state variables at runtime. A lists of possible time states to be selected is created, and when singularities occur during the simulation, the solver may change the state variables by selecting an alternative set of states from the lists. Dynamic state selection may cause simulation time penalties and it can be globally disabled and well as locally overridden. Under the *Tools→Options* menu, inside the *Global→Translation* screen, dynamic state selection can be globally disabled, enabled for all classes, or only of a specified set of classes. In this same window it is interesting to check the *Selected states* logging option, this will print the selections made regarding state variables in the *Build log*.

When using the *MultiBody* library, position and velocities (angular or linear) are the state variables of mechanical systems, and these magnitudes are usually relative from one body to another. Relative positions and velocities are defined in terms of the joints that connect bodies together. For example, a body that is connected to the ground is rotating at a speed of ten radians per second. This means that state variables are intrinsic to the definition of joints,

which provide the mathematical constrains that link bodies together. In the *Modelica Standard Library*, joint components contain special parameters that allow the custom selection of their relative positions and velocities as state variables, locally overriding dynamic state selection. This parameter is found under the *Advanced* tab in prismatic, revolute, cylindrical, universal and planar joints as the *stateSelect* parameter, and as the *enforceStates* parameter for spherical joints.

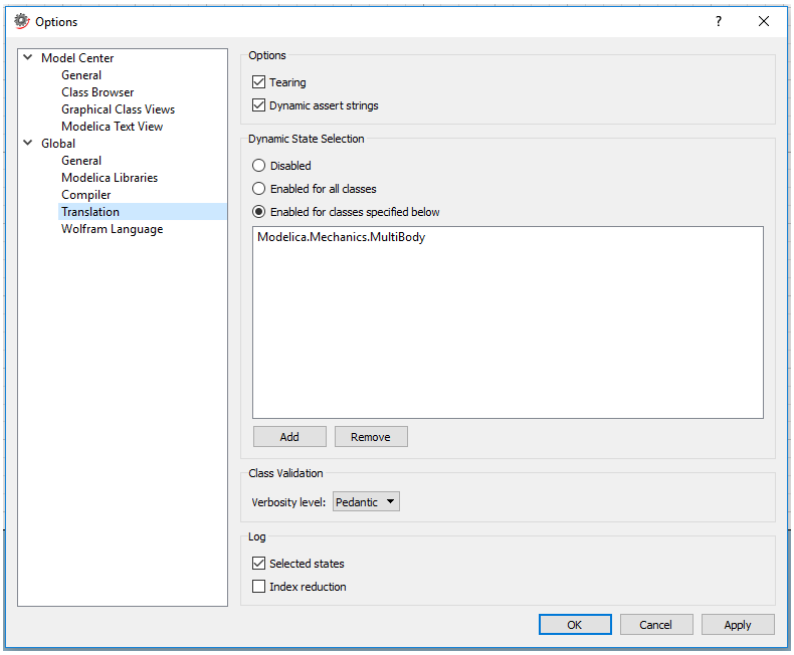


Figure F.1: SystemModeler window where state variables and *dynamic state selection* options can be tweaked.

There are five different possible values for the *stateSelect* attribute defined in the *StateSelect* enumeration: never, avoid, default, prefer and always. The *enforceStates* parameter is a boolean value, being equivalent to *StateSelect.default* when false and to *StateSelect.always* when true.

Value	Description
never	Do not use it as a state at all.
avoid	Use as state if it cannot be avoided (only if variable appears differentiated and no other potential state with attribute default, prefer, or always can be selected).
default	Use as state if appropriate, but only if variable appears differentiated.
prefer	Prefer it as state over those having the default value (also variables can be selected, which do not appear differentiated).
always	Do use it as a state.

Table F.1: StateSelect enumeration definition.

SystemModeler, and most *Modelica*-based platforms, use *dynamic state selection* to determine which state variables to choose in every time moment along with index reduction algorithms, which is known as *tearing*, in order to reduce high-index systems of differential-algebraic equations to lower index equations. The algorithm behind *dynamic state selection* is described in a paper

by Sven Erik Mattsson and Gustaf Söderlind from Lund University[1] and the *tearing* algorithm is based on a paper by Constantinos Pantelides from the Imperial College of London[2]. These algorithms are usually very efficient even for extremely large systems, but under some special circumstances they can fail or lead to unwanted results.

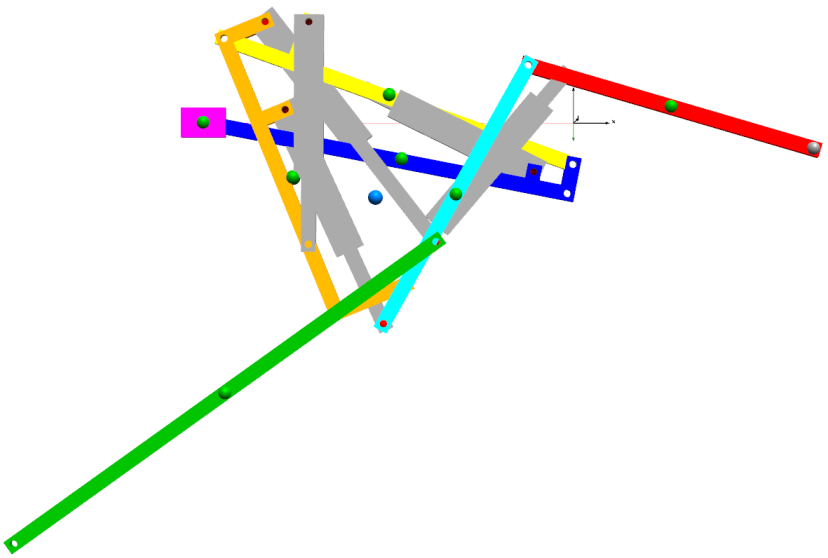


Figure F.2: Visual representation of a case of inadequate state selection for the model in *SystemModeler*. Choosing initial values and state variables incorrectly may lead to wrong initialization of the model causing most of the times simulation to fail.

In *Modelica*, variables that could be potentially chosen as state variables of the system can be given initial values. These variables have two parameters regarding this topic: *Initial Value* and *Fixed*. The *Initial Value* parameter, as its name indicates, defines the initial value of the variable, which by default is equal to zero or an array where all elements are zero. The *Fixed* parameter indicates the solver which value to use when solving the system of equations. If the parameter is set to false, which is the default setting for variables, the solver is free to use another initial value than the one specified if it is necessary to solve the equations. When set to true, the solver must use the initial value specified. When manually selecting state variables, it is very interesting to make use of this setting to achieve the correct solution to the initialization problem (initial position) of the model. For example, in some mechanical configurations there could be two different solutions for the position of a body which is connected to another one using a kinematic pair (the connected body could be on one side of the base body or on the other) and forcing the initial value on the correct variables will cause the model to start with the desired configuration.

Fixing the initial value for variables must be done with caution. Variables and their derivatives are found in nonlinear systems of equations, typically along with many other variables. Therefore it is not only one variable's initial value that matters, but several.

[1] **Mattsson, S. and Söderlind, G.**. "Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives." SIAM Journal on Scientific Computing 14, no. 3: pp. 677-692. (1993)

[2] **Pantelides, C.**. "The Consistent Initialization of Differential-Algebraic Equations." SIAM Journal on Scientific Computing 9, no. 2: pp. 213-231. (1988)

SystemModeler outputs notifications regarding state selection and the initialization problem in the model. These messages can be found in the build log of the Simulation Center. Selected state variables are listed after the message: *The following variables are selected as states.* If the initialization problem is not fully determined, which can lead to the solver choosing the undesired solution from the multiple possible solutions of the equations, the list to variables in conflict are listed under the message: *The initialization problem is underdetermined.* One must review the output of the build log to check if the state selection is correct and to correct underdetermined initialization problems.

The initialization problem of the model must be exactly determined regarding the relevant variables that have the choice of initial value specification. This is done by setting the *Fixed* parameter to true in the variables listed under the initialization message in the build log. Not all variables can have a fixed initial value, one must chose the most relevant or sensitive variables for the dynamic response of the model. If too much variables have fixed initial values or they contain incompatible combinations, the build log will include the notification: *The initialization problem is overdetermined.* Overdetermined initialization problems usually produce the simulation to fail before even starting because the solver is unable to find a suitable starting position. There is a limit on how many variables can have fixed initial values before the problem becomes overdetermined.

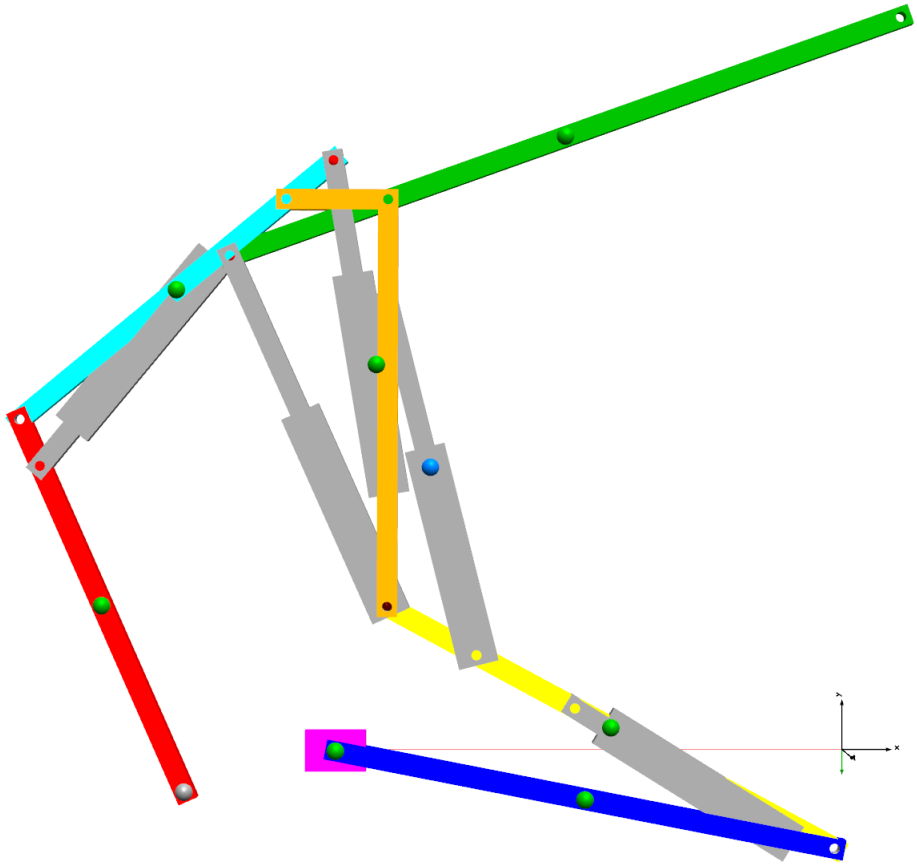


Figure F.3: Visual representation of the correctly initialized model in *SystemModeler*. The green spheres represent the center of mass of each of the bodies and the blue sphere the center of mass of the whole model.

F.2 Planar loops in the *MultiBody* library

Just as it happens in other CAE software, the presence of redundant constraints is a problem when defining mechanical systems in the *Modelica* language.

If a mechanical closed loop is defined in a classical way, without considering the theory of self-aligning mechanisms, it will have redundant constraints. A closed loop is a typical example of a mechanism where redundant constraints appear; three redundant constraints appear for planar loops where all the joints allow motion in the same plane, and five redundant constraints appear when the loop is defined in three dimensional space. This is an issue that has been addressed by the *Modelica* developers.

In order to resolve the problems derived from a planar closed loop formed by revolute joints, the *Mechanics.MultiBody.Joints.RevolutePlanarLoopConstraint* component was developed in the *Modelica Standard Library*. When a planar loop is present, e.g. a four-bar mechanism with four revolute joints with axes parallel to each other, there is no unique solution to the system if all the joints are standard revolute joints, causing the solving algorithms to fail. This specially patched component replaces the redundant constraints with the appropriate known variables, e.g. the force in the perpendicular direction of the loop (in the direction of the axis of rotation) is set to zero instead of being an unknown quantity as it happens in standard revolute joints.

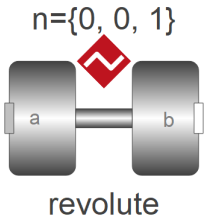


Figure F.4: *Modelica Mechanics.MultiBody.Joints.RevolutePlanarLoopConstraint* component.

The usage of these specially patched components available in *Modelica* is an alternative to self-aligning mechanisms, but it should be used with caution. For more complex mechanisms, there might not be possible to replace a revolute joint with this patched version of it, and it is more appropriate to take some time to consider how to remove redundant constraints using self-aligning mechanisms. This will make the designer have a greater insight of how the model works and be able to anticipate possible issues that could rise when choosing to build the mechanism in the physical world.

Bibliography

Beretta, Robert (1995). *Mechanical Systems Pack. User's Guide*. Ed. by Wolfram Research.

Bolt, Brian (1991). *Mathematics meets technology*. Ed. by Cambridge University Press.

Boston Dynamics (2017). *Handle*. URL: `https://www.bostondynamics.com/handle`.

FunctionBay Inc. (2017). *V9R1 RecurDyn Manual*.

Haug, Edward J. (1989). *Computer aided kinematics and dynamics of mechanical systems*. Ed. by Allyn and Bacon.

How, Jonathan and Frazzoli, Emilio (2010). *16.30 Feedback Control Systems. Fall 2010*. Ed. by Massachusetts Institute of Technology: MIT OpenCourseWare. URL: `https://ocw.mit.edu`.

McCormack, Anthony S. and Godfrey, Keith R. (1998). "Rule-Based Autotuning Based on Frequency Domain Identification". In: *IEEE Transactions on Control Systems Technology, Volume 6 Number 1, January 1998*.

Reshetov, Leonid Nikolayevich (1982). *Self-aligning mechanisms*. Ed. by Mir Publishers.

Shimizu, Nobuyuki (2015). *Recurdyn for Beginners. Innovation for Design and Analysis with Multibody Dynamics*. Ed. by FunctionBay Inc.

Tiller, Michael (2015). *Modelica by Example*.

Wolfram Research (2018). *Wolfram MathWorld*. URL: `https://mathworld.wolfram.com`.

Index