

POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di laurea magistrale in Mechatronic Engineering

Master Thesis

MATLAB Simscape Multibody model based simulation for mechatronic systems

Comau AGILE 1500 Lifter design validation



Academic Supervisor

Prof. Massimo Violante

Comau Supervisors

Ing. Simone Belardinelli

Dott. Andrea Ascheri

Ing. Luca Lachello

Candidate

Andrea Giacotto

MathWorks Supervisors

Aldo Caraceto

Giovanna Galliano

March 2019

*Dedicated to
my family, my girlfriend, and my friends,
whose love and understanding always help me to achieve my goals.*

Acknowledgement

I would like to thank my academic supervisor, Professor Massimo Violante, who gave me the possibility to work with two big companies, Comau and MathWorks, for this interesting project, and supported me during the development of this thesis.

My sincere thanks goes to my managers in Comau, first of all Simone Belardinelli, but also Andrea Ascheri and Luca Lachello. It has been a pleasure to start my first company experience with them. They always showed support, patience and willingness and thanks to them I have learned important skills for the professional life.

A special recognition goes to Aldo Caraceto, my manager in MathWorks, who spent a lot of time and effort in the technical support. His advice, both from technical and personal point of view, has always been useful.

I would like to thank also the other colleagues in Comau, Saverio, Onofrio, Andrea, Aldo, Gianni, Maurizio, who helped me also to integrate in the office.

I want to thank my girlfriend, Andrea, whose love and cheerfulness always help me during all my experiences.

Of course, a big thank you also for all my friends, that are always close to me also with their advice.

Last but not least, I want to really thank my parents and my sisters, who support me everyday with their love, patience and advice, and give me the possibility to chase my dreams.

Contents

Acknowledgement	ii
Introduction	1
1 Comau Lifter 500	3
2 MATLAB: Simscape and Simscape Multibody	6
2.1 Simscape	6
2.1.1 Physical network approach	6
2.1.2 Variables	7
2.1.3 Connector ports and connection lines	7
2.1.4 Library structure overview	7
2.1.5 Building a model	8
2.1.6 Representation of the physical system	9
2.1.7 Simscape simulation phases	9
2.1.8 Setting up solvers	11
2.1.9 Code generation from Simscape models	12
2.2 Simscape Multibody	12
2.2.1 Simscape Multibody's features	13
2.2.2 How Simscape Multibody software works	15
2.2.3 Simulation errors	16
2.2.4 Simscape Multibody's block libraries	18
2.2.5 Essential steps to build a model	18
2.2.6 Configure, improve performances and run a model	20
2.2.7 Code generation	21
2.2.8 From Solidworks CAD to Simscape Multibody	22
3 Model of Lifter 500	23
3.1 Start Simscape Multibody	23
3.2 Commonly used blocks	25

3.2.1	Solid Block	25
3.2.2	Rigid Transform Block	25
3.2.3	Revolute Joint Block	26
3.2.4	Prismatic Joint Block	27
3.2.5	Lead Screw Joint Block	28
3.2.6	Simulink Blocks	29
3.3	Libraries	29
3.4	Creation of the model	30
3.4.1	Base system of the lifter	31
3.4.2	Axes system	47
3.4.3	Upper system	55
3.4.4	Complete model with Rack and Pinion system	67
3.5	Model import from SolidWorks	72
4	Simulation and Validation	74
4.1	Friction values	74
4.2	Solver Profiler	75
4.3	Force and indirect torque sensing	76
4.4	Motor torque	81
4.5	Model with load position control	85
4.6	Model with changing parameters	86
4.6.1	Stop simulation	90
4.7	Results with more accurate inputs	91
4.8	Validation of the model	95
5	Conclusions	97
5.1	Simplified model of the lifter	97
5.2	Cost-Benefit Analysis	102
5.3	Closing remarks and next steps	112
	Bibliography and sitography	113

List of Figures

1.1	Complete system: Agile 1500 and Lifter 500	3
1.2	Agile 1500	4
1.3	Double scissor lift jack screw	4
1.4	Brushless motor and covering	4
1.5	Conveyor unit	5
3.1	New Project	23
3.2	Library	24
3.3	Solver Configuration Block	24
3.4	World Frame Block	24
3.5	Mechanism Configuration Block	25
3.6	Solid Block	25
3.7	Rigid Transform Block	25
3.8	Revolute Joint Block	26
3.9	Prismatic Joint Block	27
3.10	Lead Screw Joint Block	28
3.11	Simulink S-PS and PS-S Converter Block	29
3.12	Scope Block	29
3.13	From-To Workspace Blocks	29
3.14	Library Box	30
3.15	Lifter base	31
3.16	Lifter base model	32
3.17	Lifter base subsystem	32
3.18	Base block fixed side	33
3.19	Base block fixed side model	33
3.20	Link segment with one hole	34
3.21	Base block fixed side model subsystems	35
3.22	Base block fixed side subsystem	35
3.23	Base block input side	36

3.24	Base block input side model	36
3.25	Base block input side subsystem	37
3.26	Base track	38
3.27	Base track profile on XY plane	38
3.28	Base track model	39
3.29	Mobile base block	40
3.30	Mobile base block model	40
3.31	Adaptor Profile on XY plane	41
3.32	Mobile base block subsystem	41
3.33	Screw system	42
3.34	Screw model	43
3.35	Screw subsystem	43
3.36	Base linchpin	44
3.37	Base linchpin model	44
3.38	Base linchpin subsystem	44
3.39	Assembly of base system	45
3.40	Assembly of base system model	46
3.41	Single axis	47
3.42	Single axis model	47
3.43	Assembly of two axes with a linchpin	48
3.44	Assembly of two axes with a linchpin model	48
3.45	Assembly of two axes with a linchpin model subsystem	49
3.46	Central bearing	50
3.47	model of central bearing	50
3.48	subsystem of model of central bearing	51
3.49	Axes system model	52
3.50	Assembly of axes system with base system	53
3.51	Axes system without closed chain	54
3.52	Assembly of axes system with base system model	54
3.53	Upper block fixed side	55
3.54	Upper block fixed side model	55
3.55	Upper block fixed side model subsystem	56
3.56	Upper plate	57
3.57	Upper plate XY plane general extrusion	57
3.58	Upper plate model	58
3.59	Upper plate model subsystem	58
3.60	Lateral plate	59

3.61	Lateral plate model	59
3.62	Lateral plate model subsystem	60
3.63	Upper system	61
3.64	Upper system shaft side model	62
3.65	Upper system rigid connection sequence	62
3.66	Upper system model	63
3.67	Assembly of subsystems	64
3.68	Position control for Prismatic Joint	65
3.69	Lifter model	66
3.70	Rack	67
3.71	Pinion	67
3.72	Rack and Pinion Constraint	67
3.73	Rack and Pinion model	68
3.74	Rack and Pinions of the lifters	69
3.75	Complete model	70
3.76	Complete model starting position	71
3.77	Complete model end of stroke	71
3.78	Model imported from SolidWorks	72
4.1	Step size	75
4.2	Velocity of mobile block	76
4.3	Velocity of the screw	77
4.4	Model for load sensing	77
4.5	Load position on Z axis	78
4.6	Load position on Z axis	79
4.7	Torque sensing model	79
4.8	Torque for single lifter	80
4.9	Torque at motor pinion	80
4.10	Model for data acquisition	81
4.11	Motor shaft position	82
4.12	Motor shaft velocity	82
4.13	Motor shaft acceleration	82
4.14	Model with motion input	83
4.15	Motor shaft torque	84
4.16	Load control model	85
4.17	Load position and velocity	86
4.18	Parameters settings	86
4.19	Matlab script	87

4.20	Torque with changing <i>Start_Position</i>	88
4.21	MATLAB script for Load Position and Torque relationship	89
4.22	Load Position and Torque relationship	89
4.23	Stop simulation model	90
4.24	Motor Torque with input filtering of 100 Hz	91
4.25	Velocity and acceleration with input filtering of 100 Hz	92
4.26	Velocity and acceleration new input profiles	92
4.27	Inputs with Integrator Blocks	93
4.28	Velocity and acceleration with new input filtering (100 Hz)	93
4.29	Motor Torque with new input filtering (100 Hz)	94
4.30	Different acceleration profile	94
4.31	Velocity and Motor Torque with different acceleration profile (100 Hz) . . .	95
4.32	Quadrature current	96
5.1	Simplified lifter	98
5.2	Simplified model of the lifter	99
5.3	Velocity of mobile element	100
5.4	Force on mobile element	100
5.5	Torque at the motor with the complete system	101
5.6	Analysis of the current development process	106
5.7	Analysis of the new possible development process	107
5.8	Torque behaviour obtained with Excel	108
5.9	Torque behaviour obtained with Simscape Multibody	109
5.10	Different torque behaviour obtained with Simscape Multibody	109
5.11	Final cost-benefit analysis	111

MSc thesis in Mechatronic Engineering

Andrea Giacotto

March 2019

Introduction

In the industrial world, one of the objectives of these days is to always adopt new technologies, that let the companies to produce new products spending the least possible amount of time and money.

This is the aim of the project that includes Comau and Mathworks, which I followed during my thesis work.

Comau's objective is to adopt a new tool, provided by MathWorks, able to cover the following tasks: design and modelling, simulation and pre-validation. The first expected result is to cut down the costs of the modelling and testing phases. More generally, the goals of this project are the following ones:

- To help the operator in the correct sizing of the mechanical and electrical components of each product.
- To give information on the reliability of the overall product with dynamic simulations, reducing test times on real prototypes.
- To avoid errors during development process.
- To reduce costs due to errors.

Comau would like to verify how much promising can be the introduction of these tools into the design and development process. To do that, the first simulation used to test the new tool is done about an already finished product, the Lifter 500. In this way, the tests results obtained by the model can be compared with the data obtained by the real system, to verify that the model is able to correctly represent it.

The main steps of the project will be to create the model of the Lifter 500 on MATLAB Simscape Multibody, to simulate and to validate the model, and finally to perform a cost-benefit analysis.

In the first chapter I will show the complete system that I have created on Simscape Multibody: in particular I will explain the main characteristics of the Lifter 500 with the main features of this type of robot with his AGV system for motion.

The second chapter will be focused on MATLAB's tools: Simscape and in particular Simscape Multibody. I will explain the basic operating principle of these tools and the main features of them. The first part of the chapter will be focused generally on Simscape, while the second part of the chapter will be related to Simscape Multibody, the tool used during this project.

The third chapter will be related to the creation of the complete model of the Lifter 500 on Simscape Multibody. First thing will be explained how to start a new project on this tool and the basic working principle of the main blocks and library elements used in Simscape Multibody. Then I will explain how I have created the model in three sections: base system of the lifter, axes system, upper system and finally I will show the complete system assembled. The last part of the chapter will be focused on the model import from SolidWorks, and I will show how the model appears when it is imported directly from SolidWorks.

The fourth chapter will be focused on all the different methods to simulate and validate the model obtained previously. In the first part I will explain how to integrate in the model the friction values and how to verify if the simulation runs correctly using the Solver Profiler. Then I will explain how I have obtained the desired values (for example the torque at the motor) from the model. After this part I will show two slightly different models in which some values are represented by parameters that are able to change during the simulations thanks to a MATLAB script: this topic is really important to show another feature of Simscape Multibody that could be really helpful during the development process of a new product. Finally, I have showed more accurate results (at 100 Hz) that can be obtained giving as input to the model for its motion directly the velocity or acceleration profile compared to the position profile that was given as input during the previous simulations (at 10 Hz).

The final chapter will be divided in two parts: during the first one I will show an easier model created in a really short time but that is able to give as output the same results of the more accurate model, to explain how Simscape Multibody can be used to obtain results of realistic dynamic simulations in a really short time. Then, the second part of this final chapter will be focused on the cost-benefit analysis with the conclusions.

Chapter 1

Comau Lifter 500

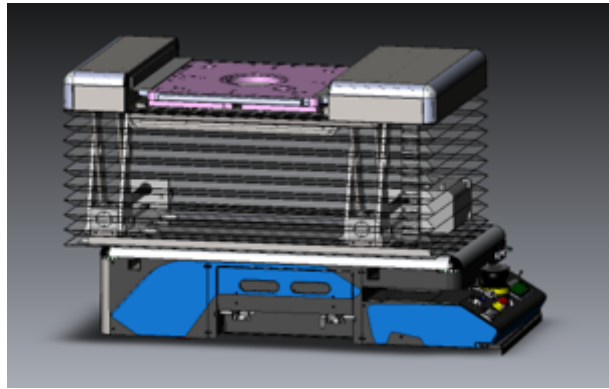


Figure 1.1: Complete system: Agile 1500 and Lifter 500

The complete system developed by Comau is composed by two parts: Agile 1500 and Lifter 500. Then, over the lifter is possible to add a particular end-effector as the transfer unit used by Comau to move pallets.

The Agile 1500 is an AGV (Automated Guided Vehicle) developed by Comau. In this application is used to move the lifter system from station to station to transfer pallets following a specified path.

The Lifter 500, that is the system that I will study during this project, is composed by a double scissor lift jack screw moved by a brushless motor, that is connected through a belt system. The rotation imposed by the motor to the screws creates the vertical motion of the two subsystems. On top of these components it is installed a transfer unit able to move pallets. The single elements that compose the complete system can be seen into the following figures.

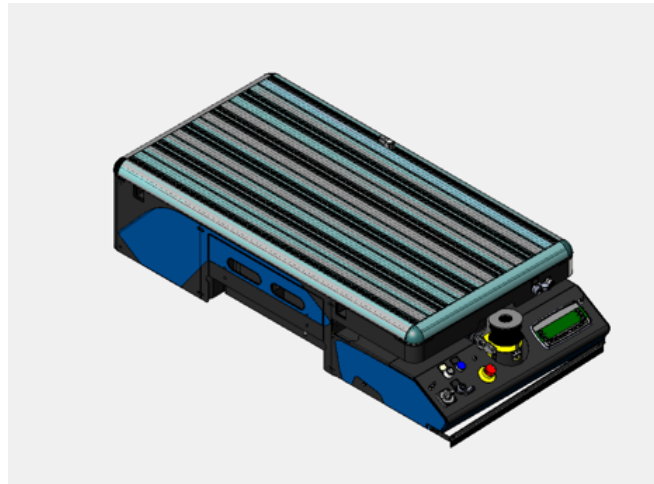


Figure 1.2: Agile 1500

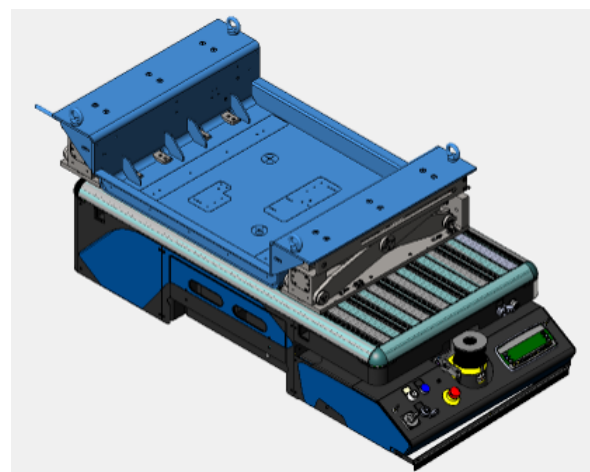
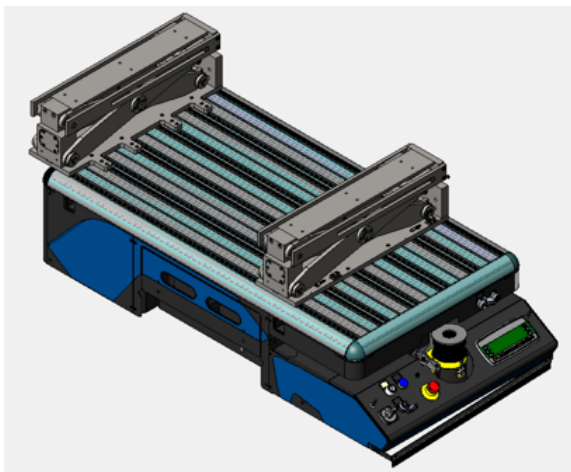


Figure 1.3: Double scissor lift jack screw

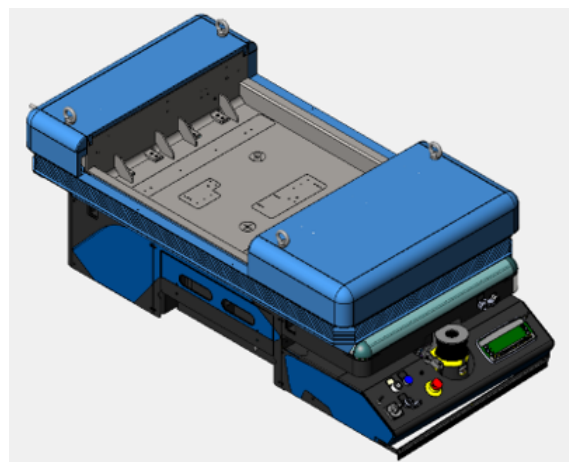
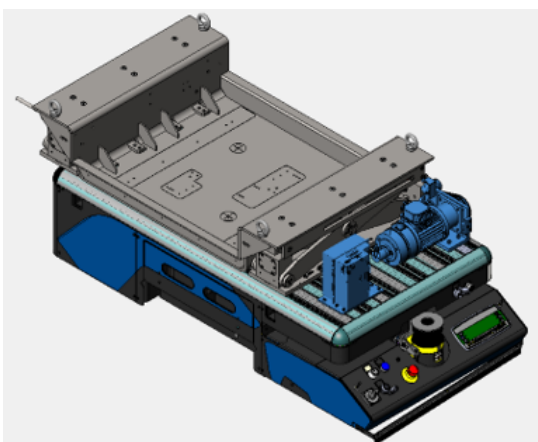


Figure 1.4: Brushless motor and covering

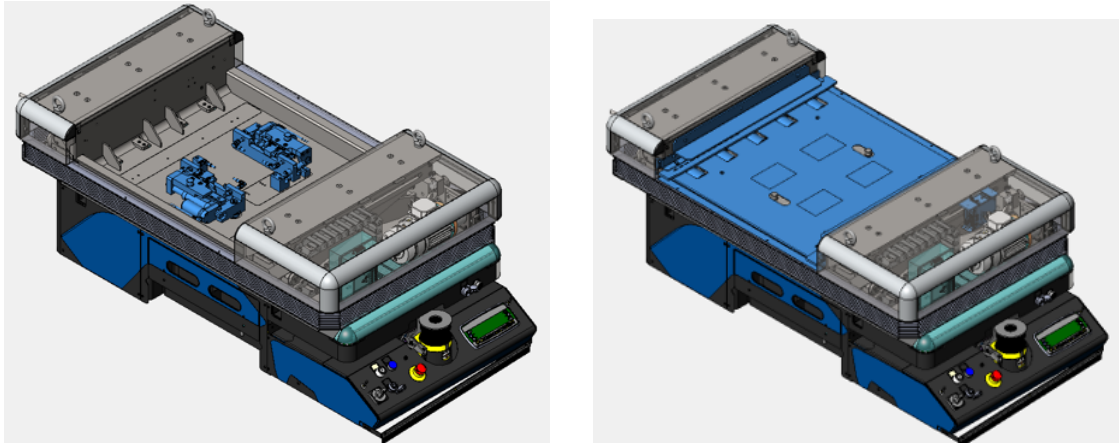


Figure 1.5: Conveyor unit

The lifter 500 has a total lifting capacity of 600 kg: in particular, the payload can have a maximum value of 500 kg (including the transfer unit). The lift stroke is about 400 mm and the total motion can be accomplished in 6 seconds.

The transfer unit that is present on the lifter system has a weight of 100 kg and can handle a maximum payload of 400 kg. The conveyor speed is around 12 m/min and is actuated by a brushless motor.

Here there are some of the distinctive features of the lifter with the conveyor unit:

- High payload.
- Fast lifting speed.
- Compact design.
- Three side accessibility: at the front side is present the sensor of the AGV.
- Possible height adjustments for ergonomic working conditions and anthropometric adaptation.
- Safe and collaborative AGV application.
- Modular design.
- Plug and play.
- Suitable for hybrid configurations: manual stations, as stop&go or continuous moving, and/or automatic stations, with exchange within automatic stations or pallet exchange with traditional conveyor systems.

Chapter 2

MATLAB: Simscape and Simscape Multibody

2.1 Simscape

2.1.1 Physical network approach

Simscape software is a set of block libraries and special simulation features for modelling physical systems in the Simulink environment. It employs the Physical Network approach, which differs from the standard Simulink modeling approach and is particularly suitable to simulating systems that consist of real physical components.

In Simulink each block represents a basic mathematical operation, and connecting Simulink blocks together is possible to obtain a diagram that is equivalent to the mathematical model of the system under design. Using the Simscape technology you will be able to create a network representation of the system under design based on the Physical Network approach: so, each system is represented by functional elements that interact with each other by exchanging energy through their ports.

These ports are nondirectional: they represent physical connections between elements, so connecting blocks together is analogous to connecting real components. In other words, Simscape diagrams mimic the physical system layout. If physical components can be connected in a real situation, their models can be connected too, without the need to specify flow directions and information flow. The Physical Network approach automatically resolves all the issues with variables, directionality, and so on.

Each element that is represented into the system can have a number of connections that is determined by the number of energy flows it exchanges with other elements. An energy flows is characterized by its variables, one Through and one Across: The Through ones are the variables that are measured with a gauge connected in series to an element, while

the Across ones are the variables that are measured with a gauge connected in parallel to an element.

The complete model can be more complex or less complex: for example, it could account friction, fluid compressibility, inertia of the moving parts, and so on. Changing the complexity of the model by including these elements, however, the basic configuration would remain the same, meaning that the Physical Network approach let you obtain models with different levels of complexity without introducing any changes to the schematic.

2.1.2 Variables

Each variable is characterized by its magnitude and sign, which can be positive or negative depending on the result of measurement orientation. Basic elements normally are characterized by two ports with one pair of variables: Through and Across variables. This approach to the direction of variables can have three benefits:

- It is possible to easily and consistently determine whether an element is active or passive.
- Simplifies the model description.
- Let you apply the oriented graph theory to network analysis and design.

2.1.3 Connector ports and connection lines

Each block in Simscape may have two type of ports:

- Physical conserving ports: nondirectional ports (for example, hydraulic or mechanical) that represent physical connections and relate physical variables based on the Physical Network approach.
- Physical signal ports: unidirectional ports transferring signals that use an internal Simscape engine for computations.

2.1.4 Library structure overview

Simscape block library contains two libraries that belongs to the Simscape product:

- Foundation library, that contains basic hydraulic, mechanical, electrical, magnetic, thermal, thermal liquid, two-phase fluid, gas, moist air, and physical signal blocks, organized into sub-libraries according to technical discipline and function performed; using these elements you can create more complex components that span different physical domains.

- Utilities library, that contains essential environment blocks for creating Physical Networks models.

Among the Utilities libraries for example there are two important blocks:

- Solver Configuration block, which contains parameters relevant to numerical algorithms for Simscape simulations.
- Simulink-PS Converter block and PS-Simulink Converter block to connect Simscape and Simulink blocks, respectively Simulink out-ports to Physical Signal in-ports, and Physical Signal out-ports to Simulink in-ports.

All blocks can be combined into the Simscape diagram to model physical systems, using also the basic Simulink blocks such as sources or scopes.

Simscape block libraries contain a comprehensive selection of blocks that represent engineering components such as valves, resistors, springs, and so on. These prebuilt blocks, however, may not be sufficient to address your particular engineering needs. When is needed to extend the existing block libraries, it is possible to use the Simscape language to define customized components, or even new physical domains, as textual files. Then is possible to convert the textual components into libraries of additional Simscape blocks that can be used into model diagrams.

2.1.5 Building a model

To build a physical model with Simscape software you must follow some rules:

- Build your physical model by using a combination of blocks from the Simscape Foundation and Utilities libraries that interacts with each other.
- Each Simscape diagram must contain a Solver Configuration block.
- In hydraulic systems the working fluid used in the circuit defines the global parameters.
- If there are gas elements into the model you have to select the correct air mixture properties.
- To connect Simulink blocks to your physical network diagram, use the converter blocks.
- Use the incremental modelling approach. Start with a simple model, run and troubleshoot it, then add the desired special effects.

- Each domain requires at least one reference block.
- Each circuit requires at least one reference block.
- Choose the correct configuration of the physical model to avoid numerical difficulties or a slowing down of the simulation.

2.1.6 Representation of the physical system

The mathematical representation of a physical system contains ordinary differential equations (ODEs), algebraic equations, or both:

- ODEs govern the rates of change of systems variables and contain some or all the time derivatives of the system variables.
- Algebraic equations specify functional constraints among system variables, without time derivatives of system variables.

So, without algebraic constraints the system is differential (ODEs), while without ODEs the system is algebraic. With ODEs and algebraic constraints, the system is mixed differential-algebraic (DAEs).

2.1.7 Simscape simulation phases

Model validation

The first step is the validation of the model configuration performed by the solver:

- All Simscape blocks in a diagram must be connected into one or more physical networks.
- Each physical network requires one Solver Configuration block.
- If there are hydraulic elements into the model is mandatory to have a Custom Hydraulic Fluid block to define the fluid properties that acts as global parameters for all the block connected to the hydraulic circuit.
- If the model contains gas elements the default properties are for dry air, so, if you need, you can change gas properties for all the block connected to the circuit attaching a Gas Properties block to the circuit.

Network construction

After validating the model, the Simscape solver starts to construct the physical network based on two important principles:

- Two directly connected Conserving ports have the same values for all their Across variables.
- Any Through variable transferred along the Physical connection line is divided among the multiple component connected to the branches.

Equation construction

Based on network configuration and parameters the Simscape solver constructs the system of equations for the model.

These equations contain two types of system variables:

- Dynamic: time derivatives of these variables appear in equations, to add dynamics to the system.
- Algebraic: time derivatives of these variables do not appear in equations.

The solver also is able to perform the analysis of the model to eliminate variables that are not needed to solve the system of equations.

Initial and transient conditions

Only at the beginning of the simulation the Simscape solver computes the initial conditions, by finding values for all the system variables that exactly satisfy all the model equations. You can also specify the initial values of some variables and, depending on the results of the solve, some of these targets may or may not be satisfied. The initial values specified can have high-priority or low priority, and the solver tries to satisfy the higher-priority targets first: the solver tries to find a solution where all the high-priority variable targets are met exactly, and the low-priority targets are approximated as closely as possible. If the solver cannot find a solution that exactly satisfies all the high-priority targets, it issues a warning, and it can enter in a second stage where high-priority is relaxed to low.

After computing the initial conditions, or after a subsequent event, the Simscape solver performs transient initialization. This process fixes all dynamic variables and solves for algebraic variables and derivatives of dynamic variables: the goal of transient initialization is to provide a consistent set of initial conditions for the next phase, transient solve. The last phase is the transient solve of the system of equations: continuous differential

equations are integrated in time to compute all the variables as a function of time. The solver continues to perform the simulation until it encounters an event: in this case, the solver returns to the phase of transient initialization, and then back to the transient solve. This cycle continues until the end of the simulation.

2.1.8 Setting up solvers

The default solver is `VariableStepAuto`, but it is possible to select a different solver. You can choose one from a suite of both variable-step and fixed-step solvers, and also among explicit and implicit.

For physical models, Mathworks recommends implicit solvers, such as `ode14x`, `ode23t` and `ode15s`.

Variable-step and fixed-step solvers

Variable-step solvers are the usual choice for design, prototyping, and exploratory simulation, and to precisely locate events during simulation. Normally they are not useful for real-time simulation and they can be costly if there are many events. A solver of this type automatically adjusts its step size as it moves forward in time to adapt to how well it controls solution error. Adjusting the solver tolerance, you can control the accuracy and speed of the variable-step solution.

Fixed-step solvers are recommended if you want to make performance comparisons across platforms and operating systems, to generate code version of your model, and to bound or fix simulation cost. Generally, it is used for real-time simulations. With a solver of this type you can specify the time step size to control the accuracy and speed of the simulation, but the solver will not adapt to improve accuracy or to locate events. In some situations, these limitations can lead to significant inaccuracies.

Explicit and implicit solvers

The choice between an explicit or implicit solver is influenced by the degree of stiffness and the presence of algebraic constraints in the model. Basically, explicit and implicit solvers use different numerical methods to simulate a system.

If the system is a non-stiff ODE system, it is better to use an explicit solver, that requires less computational effort with respect to implicit solvers.

If the system is stiff, it is better to use an implicit solver, that is more accurate and often essential to obtain a solution. Implicit solvers require per-step iterations within the simulated time steps.

Optimal solver choices and performances

For a typical Simscape model, Mathworks recommends the Simulink variable-step solvers ode15s and ode23t. The ode15s solver is more stable, but tends to damp out oscillations, while the ode23t solver captures oscillations better but is less stable.

Generally, a larger time step or tolerance results in faster simulation, but also less accurate and less stable simulation. If a system undergoes sudden or rapid changes, larger tolerance or step size can cause major errors. So, if the simulation is not accurate or looks unphysical, exhibits discontinuities in state values or reaches the minimum step size allowed without converging, a solution can be tightening the tolerance or the step size.

2.1.9 Code generation from Simscape models

The Simulink Coder software is able to generate C or C++ code from Physical Networks models and enhance simulation speed and portability.

Code generation has many purposes:

- Compiled code versions of Simulink and Simscape models run faster than the original block diagram models.
- Converting part or all of the model to code, it is possible to deploy the standalone executable program on virtually any platform, independently of Matlab.

2.2 Simscape Multibody

Simscape Multibody software is based on the Simscape software, the platform product for the Simulink Physical Modeling family, encompassing the modeling and design of systems according to basic physical principles. Simscape software runs within the Simulink environment and interfaces seamlessly with the rest of Simulink and with Matlab. Unlike other Simulink blocks, which represent mathematical operations or operate on signals, Simscape blocks represent physical components and structure of a machine or relationships directly.

Basically, Simscape Multibody software is a block diagram modeling environment for the engineering design and simulation of rigid multibody machines and their motions, using the standard Newtonian dynamics of forces and torques. With Simscape Multibody software, it is possible to model and simulate mechanical systems with a suite of tools to specify bodies and their mass properties, their possible motions, kinematic constraints, and coordinate systems, and to initiate and measure body motions. A mechanical system is represented by a connected block diagram, like other Simulink models, also with the

possibility to incorporate hierarchical subsystems. The visualization tools of Simscape Multibody software display and animate 3-D machine geometries, before and during simulation.

So, Simscape Multibody software is a set of block libraries and mechanical modeling and simulation tool for use with Simulink. It is also possible to connect Simscape Multibody blocks to normal Simulink blocks through Sensor and Actuator blocks.

2.2.1 Simscape Multibody's features

Bodies, coordinate systems, joints and constraints

Body block make possible to model bodies specifying their masses, inertia tensors and attached Body coordinate systems (CSs). The bodies can be connected with joints that represent the possible motions of bodies relative to one another, the system's degrees of freedom (DoFs). Many joints impose one or more restrictions, called assembly restrictions, on the placement of the bodies that they join: the conjoined bodies must satisfy these restrictions at the beginning of the simulation and thereafter within assembly tolerances that you specify.

Kinematic constraints can be imposed on the allowed relative motions of the system's bodies and are able to restrict the DoFs or drive the DoFs as explicit functions of time.

There are many ways to specify CSs, constraints or drivers and force or torques, in fact it is possible to:

- Attach a Body CSs to different points of body blocks to specify local axes and origins for actuating and sensing.
- Take Joint blocks from the Simscape Multibody library or extend the existing Joint library by constructing different custom Joints.
- Use other Simulink tools as well as Matlab expression.

Each Simscape Multibody model contain a single inertial reference frame called World, but then is possible to add other local CSs. Each CSs can be grounded, so attached to ground blocks, or can be fixed on and move rigidly with the bodies.

After the creation of different bodies and coordinate systems the next step is the specification of kinematic relations between bodies, constraining the motion of the system by connecting Constraint blocks.

Sensors, actuators, friction, and force elements

Sensors and Actuators are the blocks used to interface between normal Simulink blocks and Simscape Multibody blocks. Force Elements represent internal forces that require no

external input.

Actuator blocks specify the motions of Bodies or Joints, so can be used to:

- Apply a time-varying force or torque to a body or joint.
- Specify the position, velocity, and acceleration of a joint or driver as a function of time.
- Specify the initial position and velocity of a joint primitive.
- Specify the mass and/or inertia tensor of a body as a function of time.

Sensor blocks can detect motions, forces and torques, and constraint reactions forces and torques of Bodies and Joints: they can provide a Simulink signal that can be reused for other purposes and displayed using a Simulink Scope block.

Force Elements model internal forces between bodies or acting on joints between bodies and are independent of external signals.

Simulating and analysing mechanical motion

Simscape Multibody provides four modes for analysing the mechanical systems you simulate: Forward Dynamics, Trimming, Inverse Dynamics and Kinematics. Another feature is the conversion of any mechanical model, in any mode, to a portable, generated code version, that can provide a higher simulation speed and model portability.

To simulate the model some important parameters and connections must be specified:

- Masses and inertia tensors of all bodies must be known.
- All forces and torques acting on each body at each instant of time must be known.
- All kinematic constraints must be specified in a correct way.
- Initial conditions must be specified and consistent with all constraints.
- Each element must be connected to at least one ground.
- The elements of the model interfaced to Simscape mechanical circuits must satisfy both Simscape Multibody and Simscape modelling rules.

If inverse dynamics is used, to obtain the input forces/torques, the motions must be specified. Depending on the topology of the system, there can be two Simscape Multibody modes for efficiently analysing its inverse dynamics: Inverse Dynamics mode, that works with open topology systems (model diagrams without closed loops), or Kinematics mode, that analyses the motion of closed-loop models.

The Forward Dynamics mode is used to obtain the resulting motions integrating applied forces/torques, thanks to the Simulink suite of ordinary differential equation (ODE) solvers to solve Newton's equations. The ODE solvers project the motion of the DoFs onto the mathematical manifold of the kinematic constraints and yield the forces/torques of constraint acting within the system.

The Trimming mode is able to search for steady or equilibrium states in mechanical motion using the Simulink trimming features and setting them as starting point for linearization analysis.

Visualizing and animating models

During modelling it is possible to display the bodies and their Body coordinate systems. To create them there are some standard geometries that can be modified, or it is possible to create new ones. Besides displaying model's bodies either while the model is being built or as a completed model, it is also possible to keep the visualization window open while a model is running in the Simulink model window.

2.2.2 How Simscape Multibody software works

The machine simulation sequence has four major phases and the first two occur before machine motion actually starts.

Model validation

The first steps are the check of the data entries from the dialogs and of the local connections among neighbouring blocks. After that, the software validates the Body coordinate systems, the joints, the constraints, the driver geometries, and the model topology. All Body positions and orientations defined by Body dialog entries constitute the home configuration.

Machine initialization

The simulation next checks the assembly tolerances of joints that have been manually assembled. Then, each closed loop is cut once, and an implicit constraint replaces each cut Joint, Constraint, or Driver block. So, the simulation checks all constraints and drivers for mutual consistency and eliminates redundant constraints. It also checks whether a small perturbation to the initial state changes the number of constraints, because such a singularity might lead to a violation of the constraints during the machine motion. Now any Joint Initial Condition Actuators impose initial positions and velocities, changing body geometries from their dialog box configurations as necessary and transforming the

machines from their home configuration to their initial configuration. For each disassembled joint the simulation finds an assembly solution, and initializes it in position and velocity, defining the assembled configuration. During this phase all assembly tolerances are checked again. A joint primitive that is actuated by a Joint Stiction Actuator can be in one of the three stiction modes: locked, waiting, or unlocked. The simulation finds a mutually consistent set of stiction modes for all sticky joints.

Force analysis and motion integration

In Forward Dynamics or Trimming analysis mode the simulation begins the solution of the machine motion by applying and integrating external forces and torques. It maintains assembly, constraint, and solver tolerances and checks driver and constraint consistency. It also detects the so-called joint axes singularities: within each Joint block, distinct joint primitive axes can align and destroy one or more independent DoFs.

In Inverse Dynamics and Kinematic modes, during this phase, the simulation applies motion constraints, drivers, and actuators to find the machine motion and derive forces and torques. Also in this case, the simulation checks tolerances and consistency and detect possible singularities.

Stiction mode integration

The simulation, if stiction is present, checks at each time step whether the sticky joints transition from one stiction mode to another. After that, it checks for mutual consistency of locked and unlocked sticky joint primitives across the whole model.

2.2.3 Simulation errors

The simulation can stop before completion with one or more error messages. There are different type of errors, that generally are included in one of the following groups.

Data validation errors

In Simscape Multibody every numerical entry must be a real numerical expression or Matlab equivalent.

Ground and body geometry errors

Every machine must have at least one Ground block. So it is important to directly or indirectly define the Body coordinate systems of a machine relative to a Ground. It is not possible to enter cyclic Body CS definitions, that must separately satisfy these criteria in the Position and Orientation tabs of the Body dialog. Each Body to be displayed in

visualization must be connected to at least one Joint that is connected to the rest of the machine or to Ground.

Joint geometry errors

Each Joint block has particular rules that must be satisfied to avoid conflict with assembly requirements and restrictions. All joints must satisfy assembly tolerances all times. When a massless connector is used to hold the distance between two Body CS origins during motion, the initial distance must be nonzero. Certain Joint blocks place restrictions on their primitive axes: so, it is important to follow these rules during the creation of the joint between two reference frames.

Motion inconsistency and singularity errors

Problems related to motion inconsistency can arise from misapplication of constraints, drivers, and actuators, from conflicting stiction requirements, and incorrect simulation dimensionality. Often these types of problems can be caused also by singularities.

Each body that is moving and that is activated by forces or torques must have nonzero mass or a nonzero inertial moment. Within a single Joint block, two distinct axes or two distinct revolute axes should never align during the simulation. Each machine must have degrees of freedom to move: Constraint, Driver, and motion-actuating Actuator blocks that are added into a machine can reduce the number of degrees of freedom, so, to have at least one independent degree of freedom, one or more of these blocks should be removed. Some constraints can be restricted what another constraint is already restricting: in these cases, is important to identify and remove the redundant constraints. Another problem can occur during the simulation: some machine motions might not be able to maintain assembly tolerances at a particular simulation step while simultaneously satisfying the constraints, so one or more joints may become disassembled, leading to errors in the model. For these types of errors there are many ways to solve the problem: it is possible to decrease the Simulink solver tolerances or the step size, to switch to a more robust Simulink solver, to decrease the constraint solver tolerances, to increase the redundant constraint tolerance, to switch to the machine precision constraint solver, or to increase the assembly tolerances.

Another problem can occur when on a joint there are more than one actuator, but this is not possible because a joint can simulate the movement created by only one actuator.

Finally, there can be a conflict in the simulation when there are two or more stiction-actuated joints: to solve that problem it is possible to change the Joint Stiction Actuator locking thresholds or to remove one or more stiction actuators.

2.2.4 Simscape Multibody's block libraries

The Simscape Multibody's block library is organized in separate sub-libraries, each with a different type of blocks. The following are some of the most important ones:

- **Body Elements:** in this library there are blocks related to the creation of different bodies, in particular the Solid Block, able to create different type of shapes with their dimensions, inertia and mass. It is also possible to choose some graphic properties and to create other frames on the body in addition to the Reference Frame that is located at the center of mass.
- **Joints:** this library provides blocks to represent the relative motions between bodies as degrees of freedom (DoFs). Here are present some assembled joints that are able to represent a specific relative motion like rotation, translation, roto-translation and spherical, telescopic or universal movement. Each joint has specific characteristics and different modes to couple the axes used for the relative motion between the reference frames of the bodies.
- **Constraints:** this library contains blocks used to specify prior restrictions on DoFs between bodies. These restrictions can be time-independent or time-dependent, and so controlled by Simulink signals.
- **Force Elements:** this library provides blocks for creating forces or torques between bodies.
- **Utilities:** this library contains miscellaneous blocks useful in building models.
- **Frames and Transforms:** this library contains blocks related to reference frames, like the World Frame, and to rigid transforms between reference frames.
- **Belt and Cables, and Gears and Couplings:** these two libraries contain different blocks used for specific purposes, to obtain models of more particular and complete systems.

2.2.5 Essential steps to build a model

The essential steps to create a new model are always the same, regardless of its complexity:

1. Selection of Ground, Body and Joint blocks.

It is possible to drag and drop the Body and the Joint blocks needed to create the new model. Each model must have at least one Solver Configuration block, a World Frame block and a Mechanism Configuration block, to have respectively a solver

for the model, an immobile ground point at rest in absolute (inertial) space and a block able to set mechanical and simulation parameters for the entire machine. The Body block will represent the rigid bodies and the Joint blocks will represent relative motions between Body blocks.

2. Positioning and connection of blocks.

The blocks taken from the libraries can be included into the model. An essential thing is to follow a “valid tree” block diagram composed in this order: Mechanism Configuration, Ground, Solver, Joint, Body, Joint, Body, ...

A body can have more than one joint attached, to create more complex sequences, but Joint blocks can be attached only to two frames (Bodies).

3. Configuration of Body blocks.

Clicking on the Body blocks it is possible to open the dialog boxes to set all the parameters. The first thing is the specification of the dimensions, for which is possible to use the default shapes or strings to create more complex shapes. Then can be specified the mass properties (masses and moments of inertia), and finally the position and orientation of the bodies with respect to the ground or to another coordinate system.

4. Configuration of Joint blocks.

In each Joint block is possible to set parameters related to translations axes, rotations axes, or both.

5. Selection, connection, and configuration of Constraint and Driver blocks.

From the library can be added to the model blocks to create constraints or drivers, to restrict or drive the relative motion between two bodies.

6. Selection, connection, and configuration of Actuator and Sensor blocks.

These blocks are used to impart and sense motion. A reconfiguration of the Body, Joint, and Constraint/Driver blocks is needed to accept Sensor and Actuator connections. Then is possible to specify control signals or what type of output is measured. Actuator and Sensor blocks connect Simscape Multibody blocks to normal Simulink blocks. It is not possible to connect Simscape Multibody blocks to regular Simulink blocks otherwise. Actuator blocks take input signals from normal Simulink blocks (for example, from the Simulink Sources library) to actuate motion. Sensor block output ports generate Simulink signals that you can feed to normal Simulink blocks.

7. Encapsulation into subsystems.

Each system that is created into a model can be encapsulated in a subsystem, to

obtain a clearer view of the entire model when it is more complex. Furthermore, a subsystem can be reused connecting it into another larger model. In this way is possible to create a library of subsystems that can be reused more times in different models.

2.2.6 Configure, improve performances and run a model

When the model is complete Simscape Multibody is able to run the model in different ways: Forward Dynamics, Inverse Dynamics, Trimming, ...

Another feature of Simscape Multibody is able to show the animation of the model: so, it is possible to visualize the complete model before, during and after the simulation.

To obtain better performances for the simulation there are some expedients:

- To simplify the degrees of freedom: more degrees of freedom means slower simulation, so, if it is possible, some of them shall be removed. Also removing stiction actuators the performance can be improved, because stiction requires more computational capacity. Then, if the machine that is simulated can move only in two dimensions is better to simulate that in two dimensions, to reduce the complexity of the model.
- To adjust constraints tolerances: more constraints means more computational effort. Removing or relaxing some of them generally speeds up the simulation.
- To remove motion singularities: singularities in a system's equations of motion can dramatically slow down a standard Simulink solver. So, it is necessary to avoid singular configurations also in starting positions.
- To change the Simulink solver and tolerances: to obtain a better performance it is possible to change the Simulink solver or to adjust relative and absolute tolerances. For example, the default solver requires too much time to solve systems that are stiff. Generally, for most mechanical systems, variable time-step solvers are preferable, because fixed time-step solvers often fail to resolve certain motion details.
- To adjust the time step in the real-time simulation: A real-time simulation using code generated and compiled from your model must keep up with the actual mechanical motion. To this end, it is important to ensure that the solver time step is greater than the computation time needed by the compiled model. To meet this condition there are two possibilities: to increase the time step or to decrease the computation time.

2.2.7 Code generation

As said before, it is possible to use Simscape Multibody software with Simulink Coder to generate stand-alone C code from a mechanical model and enhance the simulation speed and portability. With Simulink, Simulink Coder, and Simulink Real-Time software it is possible to link existing code to a model and generate a code version of a model.

In general, there are not many differences between the code generation of Simulink, Simscape, and Simscape Multibody. The major difference between Simscape and Simscape Multibody code generation is that a few Simscape Multibody blocks do support a limited set of tunable parameters.

When Simscape Multibody software generates the code from a model it creates a set of code source and header files: in particular there are the *modelname.c* and *modelname_data.c*, where are present all the model's run-time parameters. Then, Simscape Multibody generates also two files that contain data structures and function prototypes for the Simscape Multibody blocks alone. The *modelname.c* file contains all the run-time parameters used in the compiled simulation, while the *modelname_data.c* and the two additional files created by Simscape Multibody are used to locate and change the run-time parameters.

To change these parameters, it is possible to modify their values in the block parameters data structure implemented in *modelname_data.c*. Here all the block parameters are grouped together into a single vector. The two files *rt_mechanism_data.h* and *rt_mechanism_data.c* allow to modify Simscape Multibody block parameters in generated code. The special header file contains a data structure, *MachineParameters_modelname_uniqueid*, for each machine into the model: here are present different fields for each block run-time parameter.

So, to modify mechanical run-time parameters:

1. Use the function *rt_vector_to_machine_parameters_modelname_uniqueid* to create an instance of the machine parameters data structure from the vectorized parameters.
2. Now it is possible to modify the values into the data structure.
3. It is possible to use *rt_machine_parameters_to_vector_modelname_uniqueid* to reconstruct the vectorized parameters from the data structure instance.
4. The last step is to recompile the generated code.

2.2.8 From Solidworks CAD to Simscape Multibody

The Simscape Multibody Link enables to integrate a Solidworks CAD assembly with electrical, hydraulic, and control systems, and simulate the entire system within the Simulink environment. A plug-in lets export Solidworks CAD assemblies to an XML file and corresponding geometry files. Then this XML file can be imported into Matlab that is able to create bodies using the geometry files and convert CAD mate definitions to joints.

So, MATLAB is able to automatically assemble multibody simulation models from Solidworks CAD assemblies. Also all the parameters like mass, inertia and location of the center of gravity are automatically transferred to the simulation model, like topology and constraints. The rigid subassemblies of Solidworks are converted into a single rigid body in Simscape Multibody.

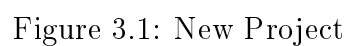
This feature of MATLAB can lead to some benefits:

- Fewer iterations on mechanical design because requirements are accurate and consistent. The main advantage of this process is the possibility of constructing complex machine simulation models with the use of geometrical models created in other CAD programs without complex mathematical derivations. So, the user does not need to possess programming skills.
- Fewer mechanical prototypes because it is possible to find errors earlier.
- Reduced system cost because components are not oversized.
- Less system downtime because integrated system is tested using virtual commissioning on a digital twin.

Model of Lifter 500

There are two ways to start a new project with Simscape Multibody:

- These functions are able to open a Simscape Multibody model template with some commonly used blocks and the Simscape Multibody's Library:



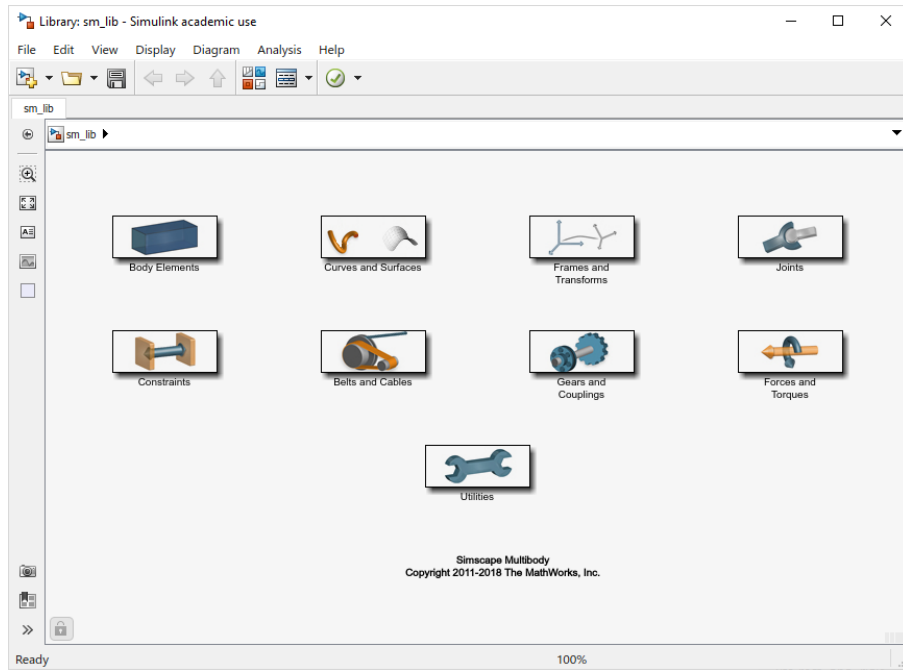


Figure 3.2: Library

Among these blocks, three of them are required to create all kinds of models:

- Solver Configuration block: it is able to specify the solver parameters needed by the model to begin the simulation.



Figure 3.3: Solver Configuration Block

- World Frame block: it represents the global reference frame in a model. It is an inertial frame at absolute rest, and ,rigidly connecting a frame to this block, the frame becomes inertial. So, directly or indirectly, all frames are defined with respect to the World Frame.



Figure 3.4: World Frame Block

- Mechanism Configuration block: it is used to provide mechanical and simulation parameters to a mechanism, for example gravity value. This block is optional, but if omitted the model will have the gravity acceleration vector set to zero.



Figure 3.5: Mechanism Configuration Block

3.2 Commonly used blocks

During the model's creation I have used different blocks with their parameters, so in this section I will describe them in more detail before starting the description of the creation of the model.

3.2.1 Solid Block



Figure 3.6: Solid Block

This block adds to the attached frame a solid element with geometry, inertia and visual properties, connected to the model through the Reference Frame R.

With the Geometry Properties is possible to define a particular Shape: there are some more simple shapes, like Cylinder, Brick or Sphere, with their parameters, or a useful option named General Extrusion, where is possible to specify the set of points on the xy plane that compose the cross-section of the shape that must be created. Then, setting also the length of the object it is possible to obtain a 3-D solid.

The Inertia Properties make possible to define density or mass of the solid. This is an important parameter because the gravity force that acts on the solid will depend on that. Finally, there are also the Graphic Parameters, where is possible to define for example colour and opacity of the solid.

3.2.2 Rigid Transform Block



Figure 3.7: Rigid Transform Block

This block is used to apply a time-invariant transformation between two frames. In particular, the Follower port frame is rotated and/or translated with respect to the Base

port frame. During the simulation, the two frames connected with this block will remain fixed with respect to each other, moving only as a single unit. So, to create more complex rigid bodies is possible to combine Rigid Transforms and Solid Blocks.

- **Rotation:** there are different methods to represent a rotation. During the creation of the model I have used principally the Standard Axis method, specifying the axis and the angle, or the Rotation Matrix method, useful when there was more than one rotation around different axes.
- **Translation:** also for translation there are different possibilities. In particular I have used the Standard Axis method, specifying the axis and the distance for a 1-D translation, or the Cartesian method, useful to represent 3-D translations with a single vector.

3.2.3 Revolute Joint Block

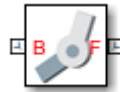


Figure 3.8: Revolute Joint Block

This block represents a joint with one rotational degree of freedom. The base and follower frame origins remain coincident during simulation. The Z axes of the two frames connected are coincident and the follower frame can rotate with respect to the base frame around Z axis.

With this block is possible to define also some parameters:

- **State Targets:** specification of initial position or initial velocity when the simulation is started. For each value is possible to define the priority. With a high priority the system must satisfy the target precisely, while with a low priority the target can be satisfied approximately.
- **Internal Mechanics:** here there are three important parameters to characterize the motion of the joint. The Equilibrium Position specifies the rotation angle at which the spring torque is zero. The Spring Stiffness represents the torque required to rotate the joint by a unit angle. Finally, the Damping Coefficient represents the torque required to maintain a constant angular velocity between the two frames.
- **Actuation:** it is possible to provide by input a torque to the joint computing automatically the motion or provide by input the motion that must be obtained computing automatically the torque required. Generally, when the option of providing

by input a value is selected, a new port will appear on the block, and it is possible to connect to this port something able to provide a certain value (constant or variable in time).

- Sensing: in the sensing menu is possible to select the variables to sense in the revolute joint primitive. If a variable is selected a new physical port will appear to output the measured quantity as a function of time. Here each quantity is measured for the follower with respect to the base frame.
- Composite Force/Torque Sensing: in this section is possible to select the composite forces and torques to sense. Unlike the normal Sensing, here the forces and torques measured are not at individual joint primitives, but at the whole joint. It's also possible to select the direction and the resolution frame for the measurement.

3.2.4 Prismatic Joint Block

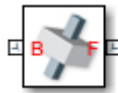


Figure 3.9: Prismatic Joint Block

This block represents a joint with one translational degree of freedom. In particular the follower frame is able to translate with respect to the base frame along the common Z axis. As in the other blocks that create a specific movement, there are some parameters that can be specified:

- State Targets: specification of initial position or initial velocity when the simulation is started. For each value is possible to define the priority. With a high priority the system must satisfy the target precisely, while with a low priority the target can be satisfied approximately.
- Internal Mechanics: here there are three important parameters to characterize the motion of the joint. The Equilibrium Position specify the distance between base and follower at which the spring force is zero. The Spring Stiffness represents the force required to displace the joint by a unit distance. Finally, the Damping Coefficient represents the force required to maintain a velocity between the two frames.
- Actuation: it is possible to provide by input a force to the joint computing automatically the motion or provide by input the motion that must be obtained computing automatically the force required. Generally, when the option of providing by input

a value is selected, a new port will appear on the block, and it is possible to connect to this port something able to provide a certain value (constant or variable in time).

- Sensing: in the sensing menu is possible to select the variables to sense in the prismatic joint primitive. If a variable is selected a new physical port will appear to output the measured quantity as a function of time. Here each quantity is measured for the follower with respect to the base frame.
- Composite Force/Torque Sensing: in this section is possible to select the composite forces and torques to sense. Unlike the normal Sensing, here the forces and torques measured are not at individual joint primitives, but at the whole joint. It's also possible to select the direction and the resolution frame for the measurement.

3.2.5 Lead Screw Joint Block

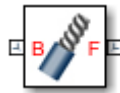


Figure 3.10: Lead Screw Joint Block

With respect to the previous blocks, the Lead Screw Joint is a little bit different. With this block is possible to specify one rotational degree of freedom coupled with one translational degree of freedom. This type of coupling ensures that whenever the joint frames rotate relative to each other, they also translate by a commensurate amount and vice-versa. To specify this relationship the joint lead is used: it represents the translation distance associated with a unit rotation angle. As in the other blocks, there are some parameters to specify:

- Lead Screw Primitive: here is possible to define the lead parameter and also the direction of the movement. In particular, with the Right-Hand setting a positive rotation leads to a positive translation, while with the Left-Hand setting a positive rotation leads to a negative translation.
- State Targets: in this section is possible to specify the position and the velocity of the joint at the start of the simulation. As in the other blocks there are two priority levels, high or low. The difference with respect to the other blocks is that position and velocity can be specified for the rotational or translational movement.
- Sensing: position, velocity and acceleration can be measured with respect to translation and rotation as a function of time. Each quantity is measured for the follower frame with respect to the base frame.

- Composite Force/Torque Sensing: as in the other blocks, here is possible to select the composite forces and torques to sense at the whole joint, not only at individual joint primitives.

3.2.6 Simulink Blocks

In Simscape Multibody is possible to use also Simulink's blocks, in particular to work with inputs and outputs of the Simscape blocks. Some of them are listed here:

- Simulink-PS and PS-Simulink Converter: these blocks are used to convert a Simulink signal into a Physical signal and vice-versa. In the Units page is possible to select the unit assigned to the Physical signal. In the Input Handling page is possible to select the input derivatives, that can be automatically calculated or explicitly provided through additional signal ports.



Figure 3.11: Simulink S-PS and PS-S Converter Block

- Scope: this is the block used to plot the output signal.



Figure 3.12: Scope Block

- To/From Workspace: these blocks are used to export/import data to/from workspace. So data obtained can be analysed into a simple scope in Simscape or also saved and then plotted in Matlab, or reused as input in another simulation using the From Workspace Block.



Figure 3.13: From-To Workspace Blocks

3.3 Libraries

During the project of the Lifter 500, to create more complex shapes, I have used also the Simscape Multibody Parts Library. To open it you have to run the MATLAB Code

startup_sm_parts: a new window will be opened with all the elements of the library listed. Each element can be created in a new project using the Solid Block: selecting General Extrusion is possible to define the cross section of the new solid, and to create a particular shape the code of each element can be used specifying the different values.

As an example, the first element of the library, Box extrusion, is showed below:

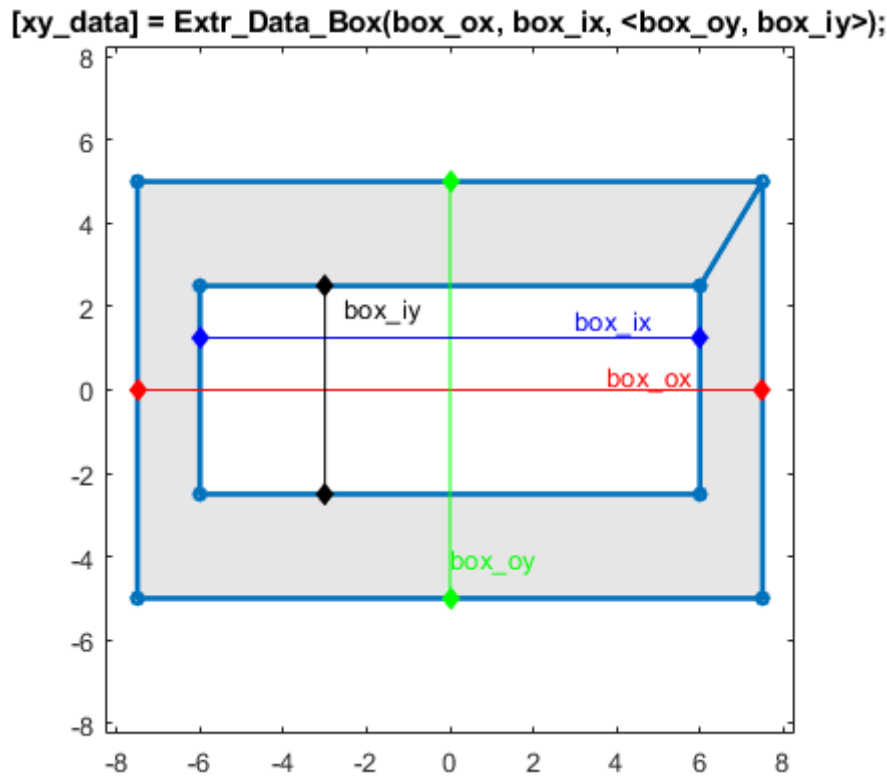


Figure 3.14: Library Box

Into the library there are also example models for each shape, so another possibility is to copy and paste the element from the example entering the values that are needed.

3.4 Creation of the model

During the creation of the model, to obtain a well-organized file, I have created each single element of the system in a different new project. Then, after the creation of more parts, I have joined all single parts in a single project to create the complete system gradually. In this way, for the creation of a complex system, is easier to find problems related to single elements, and is possible to verify that each joint is working properly. Moreover, if each part of the system is created separately, is easier to eventually create libraries for future models.

In the next sections I will explain how I have created each part of the model and how I have assembled it.

3.4.1 Base system of the lifter

Base of the lifter

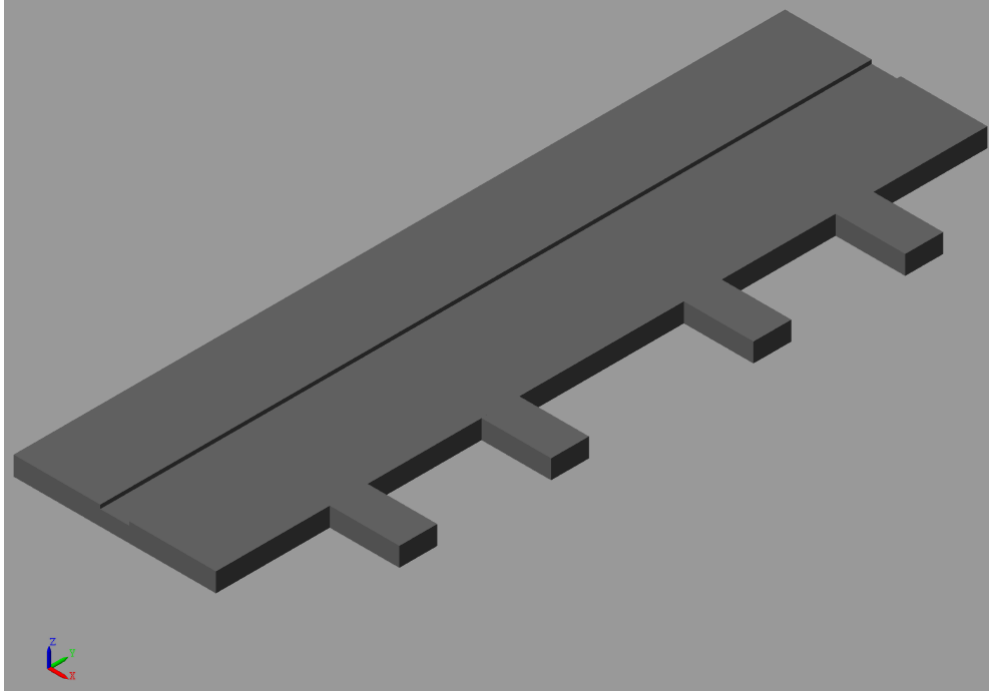


Figure 3.15: Lifter base

The base of the lifter has a particular shape, due to the fact that there is a space dedicated to the track. So, to create this shape I have used a general extrusion into a Solid block: in this way I have selected the coordinates on the XY plane to create the correct cross-section and then I have selected the correct length of the solid.

After that, on the side of the first solid I have added four simple elements that are part of the lifter base and used to fix the lifter to the AGV. During this step the complex part is to use the Rigid Transform block to position correctly the four solids. In this case I have to perform two consecutive rotations with respect to the base frame to correctly orient the solids, and then I have to position them in different places using a translation. To create two consecutive rotations with a single Rigid Transform block it is possible to use a rotation matrix, calculated using MATLAB. In this case for example I have a first rotation around X axis of -90 degrees and a second rotation around Z axis of -90 degrees.

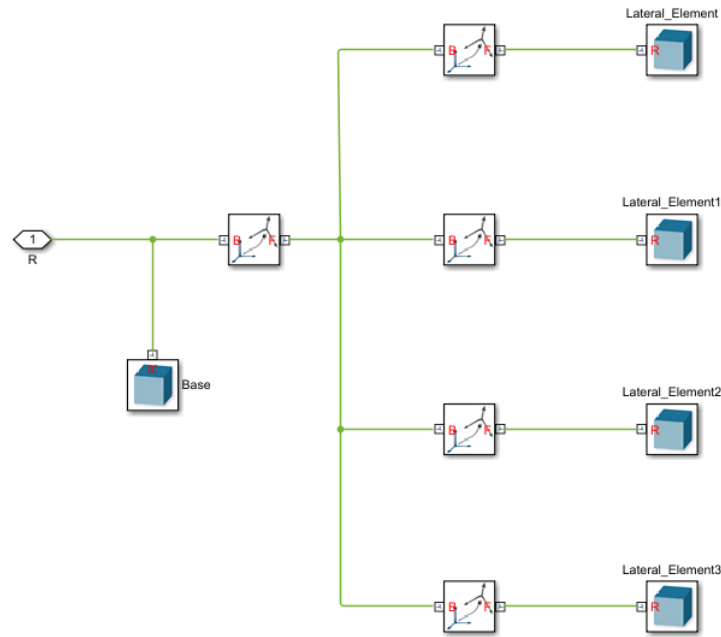


Figure 3.16: Lifter base model

Finally, to create a simpler visualization I have selected all blocks to create a Subsystem, connected to the ground through a rigid transform to orient correctly the system. In this single element and also in all the following ones, the connection to the ground in the model is necessary to run the simulation to visualize the created system in 3D space.

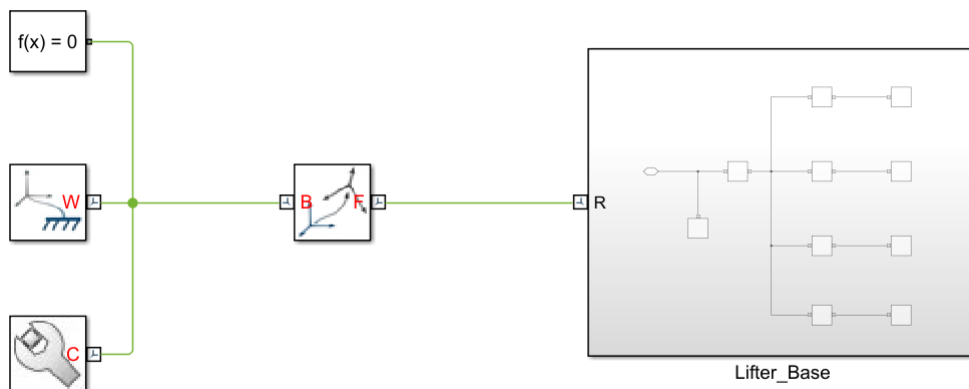


Figure 3.17: Lifter base subsystem

Base block fixed side

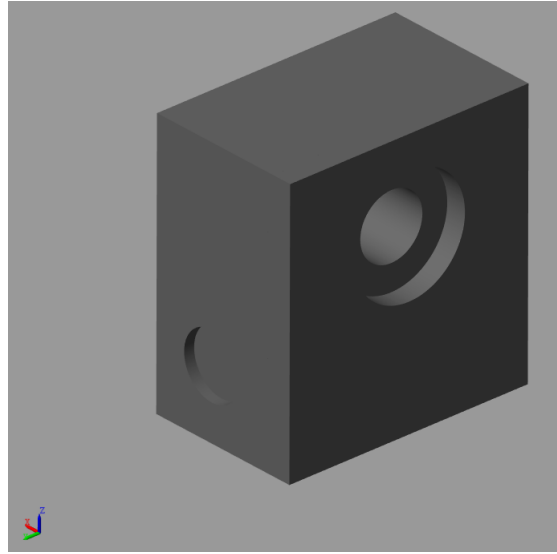


Figure 3.18: Base block fixed side

This block has a particular shape, so to create it in Simscape Multibody is necessary to join different elements with Rigid Transforms. To do that I have used one of the shapes described into the library: the simple link with one hole. Joining two of these elements with a rigid transform that represents a 180 degrees rotation, is possible to create a brick with one hole. If the hole has different values of diameter into the brick, as in this case, it is possible to join more subsystems like this. I have repeated this procedure also to create the holes on both sides of the block, as shown in the figure:

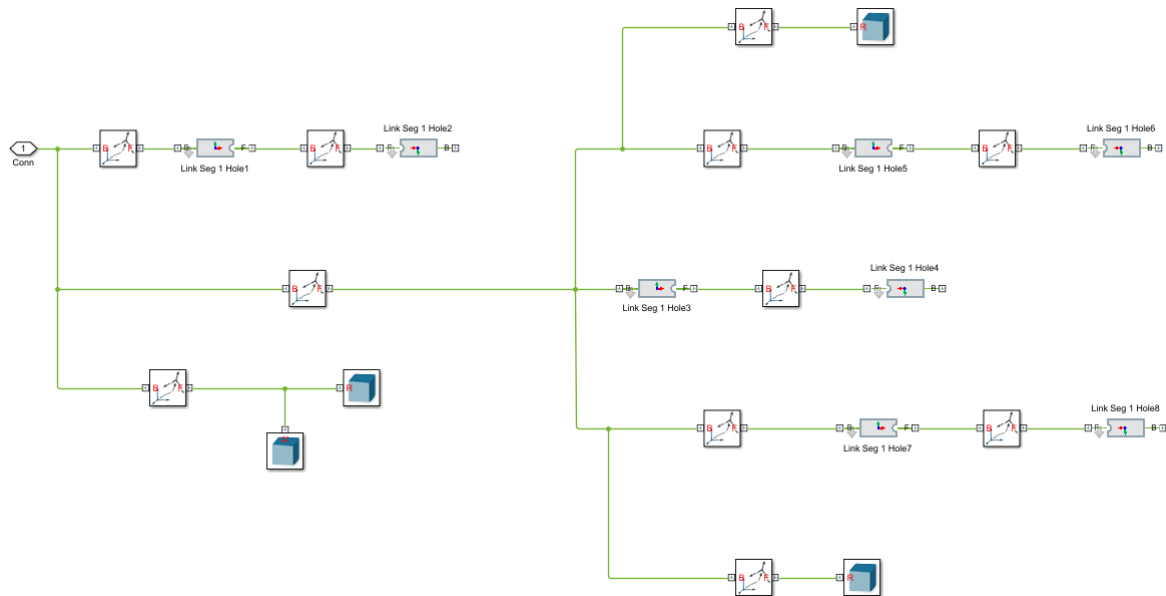


Figure 3.19: Base block fixed side model

To use the link element from the library it is possible to copy the element with its mask from an example model. Double clicking on the element the mask will open and is possible to insert all the needed values. The following figure shows the parameters that can be modified to create the link with one hole.

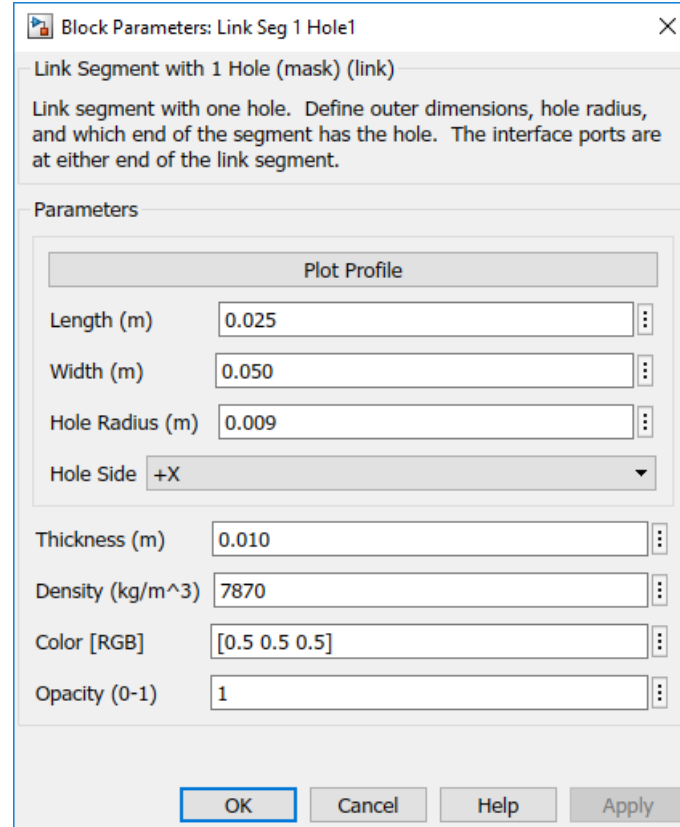


Figure 3.20: Link segment with one hole

Then, after the creation of a subsystem from this part of the model, I have added the rear flange of the block, and also a spacer that represents a bearing for the screw. To create this element, I have used another script of the library: the Ring. Also in this case is necessary to create two equal parts of the ring that later are coupled with a 180 degrees rotation using a Rigid Transform block.

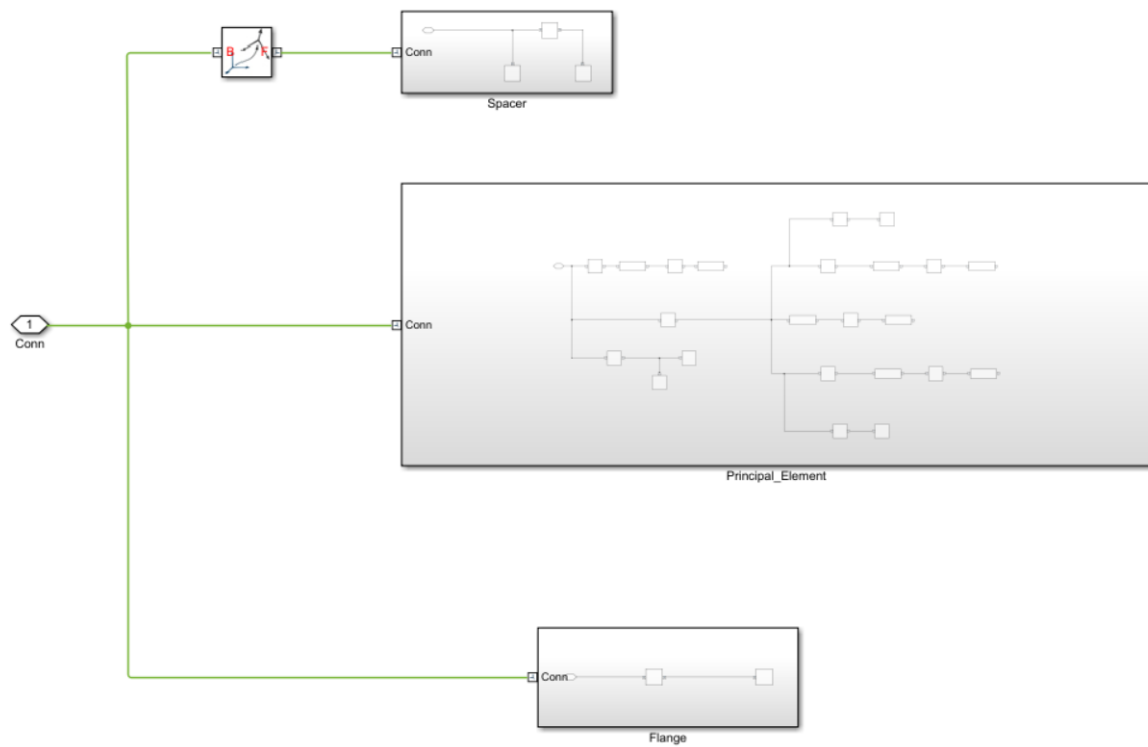


Figure 3.21: Base block fixed side model subsystems

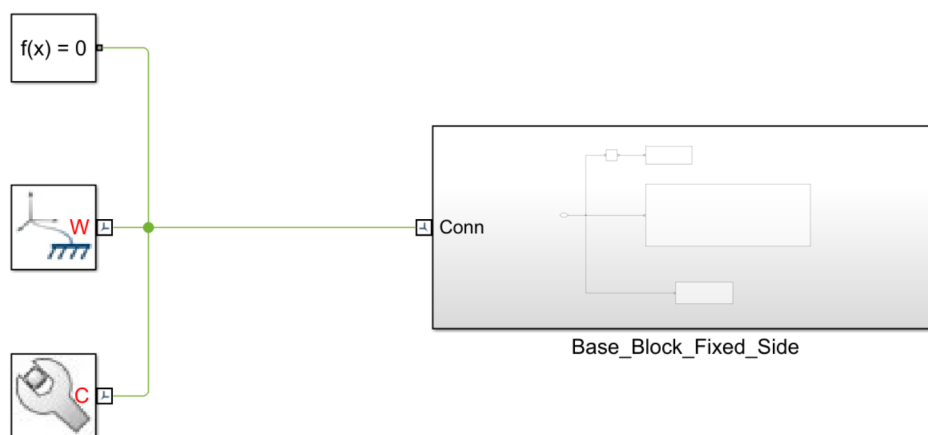


Figure 3.22: Base block fixed side subsystem

Base block input side

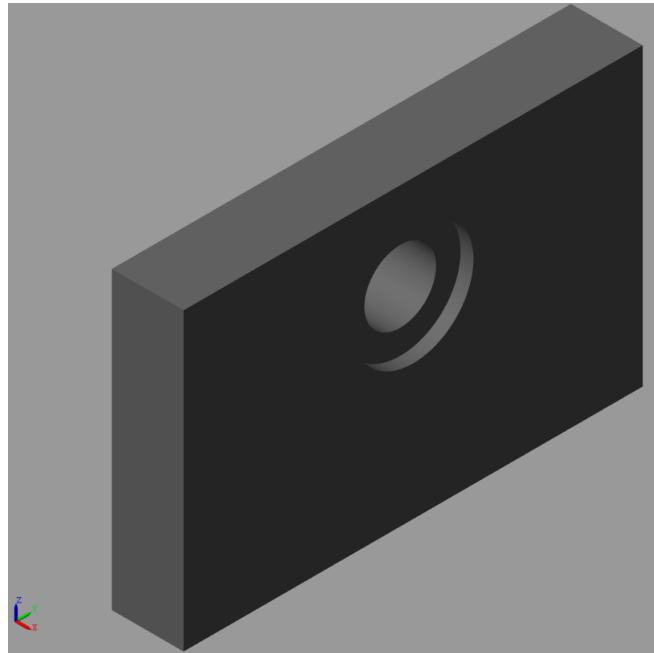


Figure 3.23: Base block input side

The creation of this block is similar to the previous one. To create the principal element of the block I have used again the link with one hole from the library of Simscape Multibody, joining the single elements with Rigid Transforms to create a single block. After that, I have added two rings that represent bearings used to couple this base block with the screw.

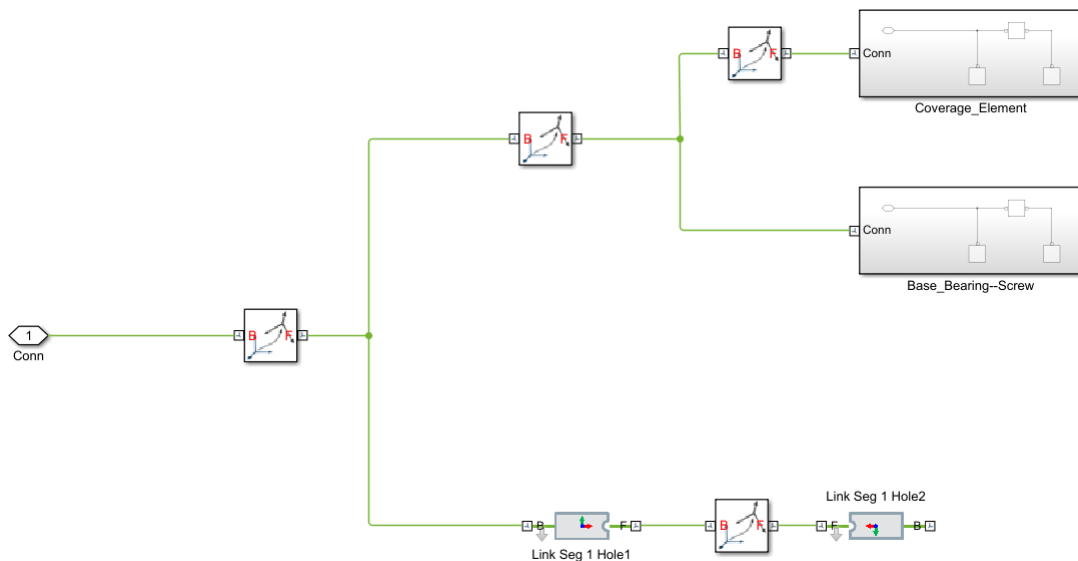


Figure 3.24: Base block input side model

As I have done also in previous cases, selecting all these blocks I have created a single subsystem connected to the ground.

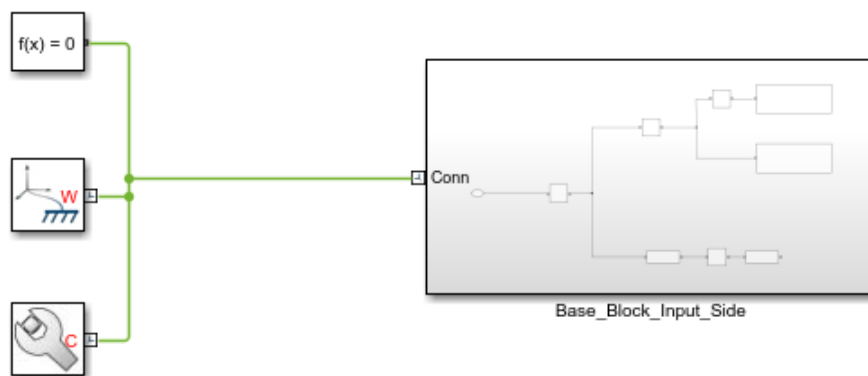


Figure 3.25: Base block input side subsystem

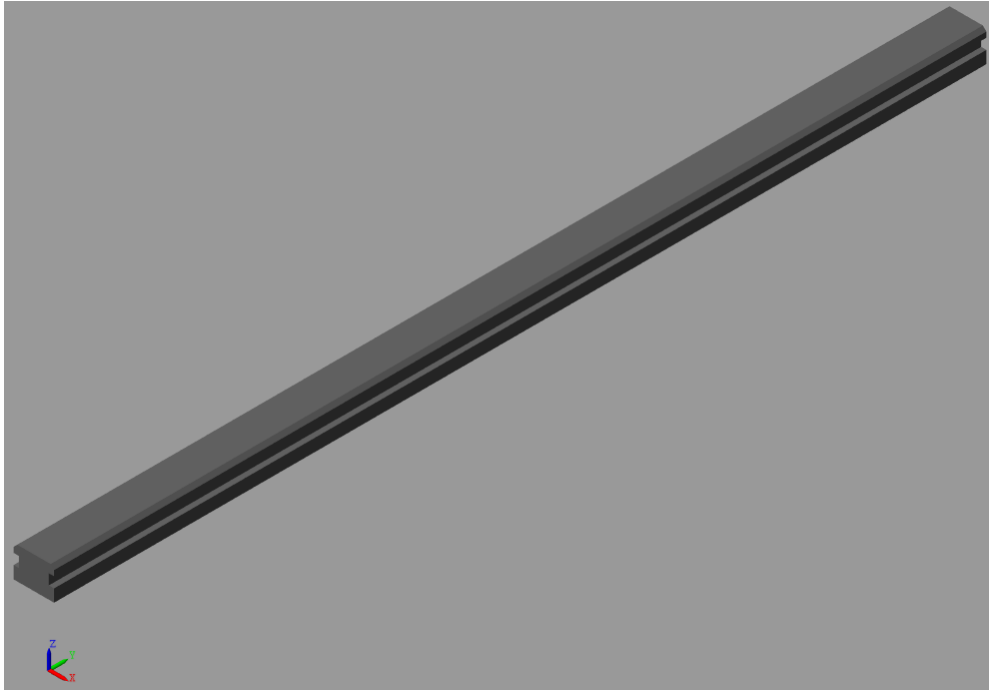
Base track

Figure 3.26: Base track

The base track is the element rigidly connected to the base of the lifter on which there will be the translation movement of the mobile base. To create this block is possible to use a single Solid block using the general extrusion: in this way is possible to create the shape on the XY plane and then to select the length of the solid.

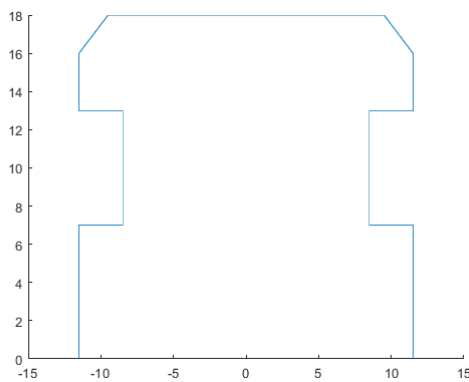


Figure 3.27: Base track profile on XY plane

Then is possible to create the connection to the ground for the 3D visualization.

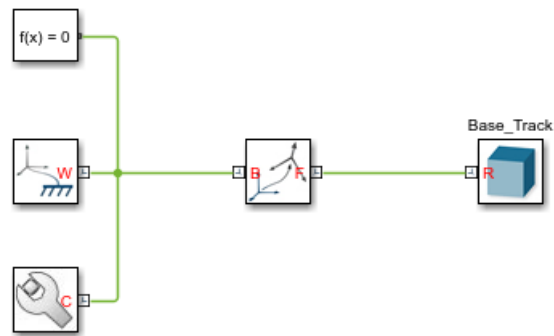


Figure 3.28: Base track model

Mobile base block

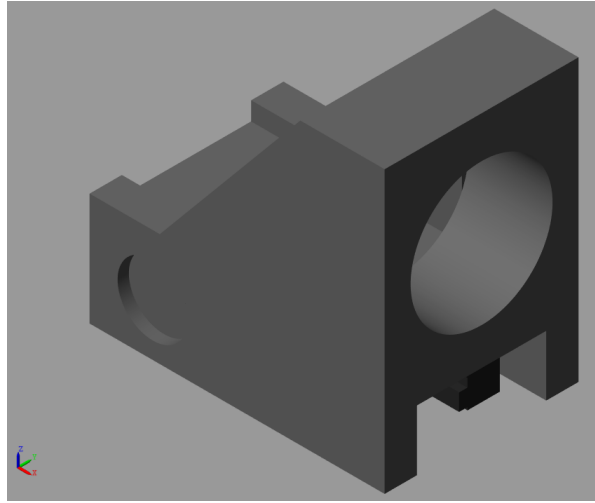


Figure 3.29: Mobile base block

This subsystem is the mobile element that is able to translate on the track creating the motion of the axes that are connected to it. The force for this motion is given through the rotation of the screw that is connected to this block. To create this block, I have used again the link with one hole that is present into the Simcape Multibody's library. The central part is composed by a system created with this method and a simple brick connected to it. The two lateral parts are created using the same element of the library to create the lateral hole, adding some simple bricks with Rigid Transforms to create the correct shape of the block.

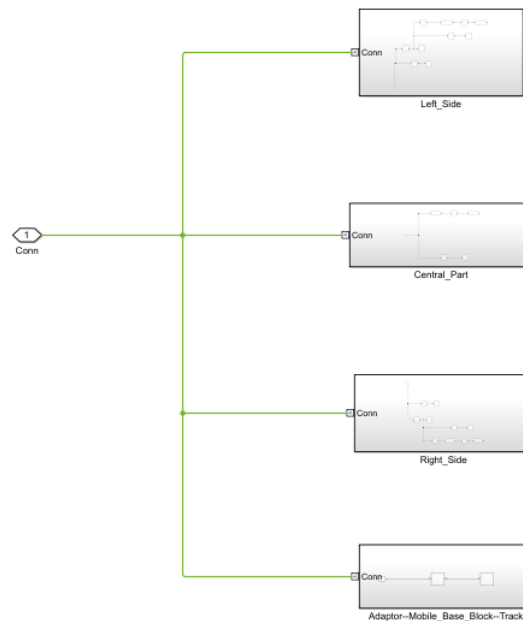


Figure 3.30: Mobile base block model

The last part is the adaptor to the track, that is connected to the central part of the block rigidly. It is done with the same method of the track, using the general extrusion in which is possible to define the points of the shape on the XY plane selecting the length of the block.

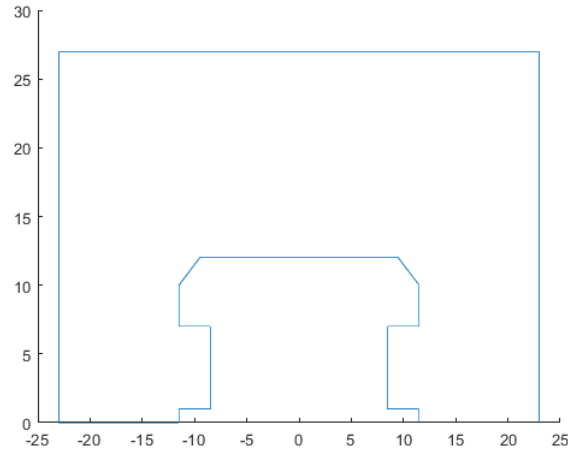


Figure 3.31: Adaptor Profile on XY plane

As in the previous cases, the last step is the creation of the subsystem selecting all the blocks and connecting it to the ground for the final visualization:

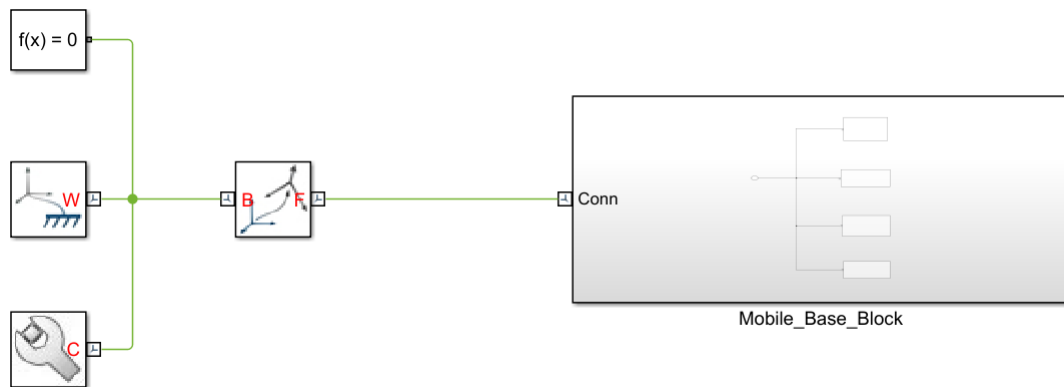


Figure 3.32: Mobile base block subsystem

Screw system

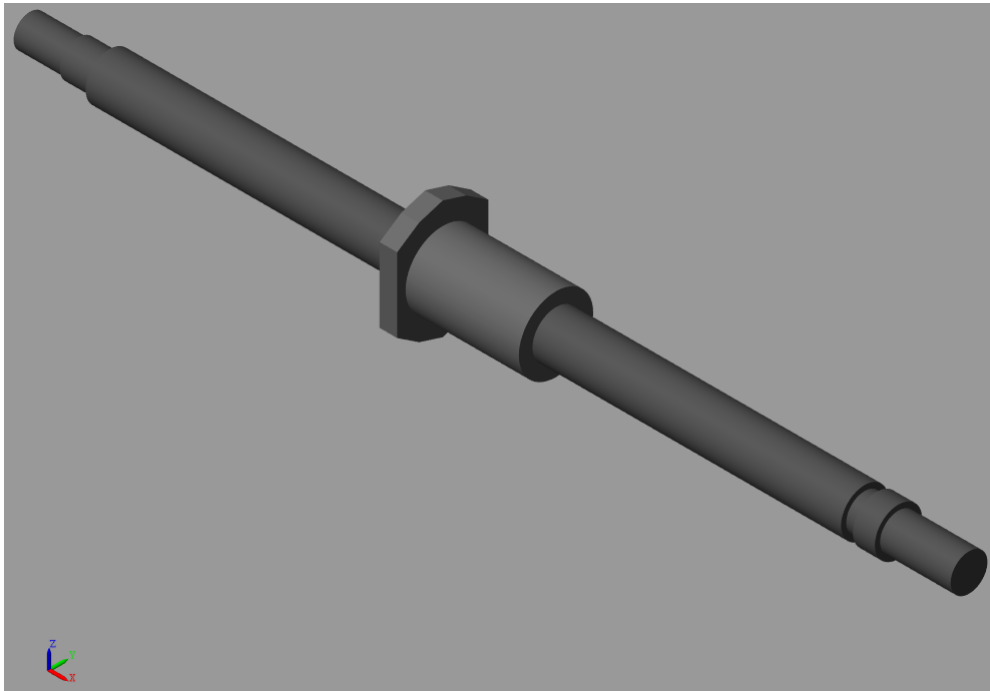


Figure 3.33: Screw system

The screw system is connected to three points: the extremities to the base fixed blocks and the central part to the mobile base block.

In Simscape Multibody it is not necessary to create a real screw, because using a simple cylinder and a Lead Screw Joint it is possible to simulate the same behaviour. So, the screw is composed by cylinders with different lengths and diameters joined with Rigid Transform blocks.

The mobile block is composed by a ring, created again using the library blocks, and a particular shape created joining rigidly a subsystem composed by a link with one hole and two curved solids, created using a general extrusion.

After the creation of the blocks it is possible to connect them with the Lead Screw Joint: the screw is the base for the movement, while the mobile block is the follower that will move with respect to the screw. For the Lifter 500 the screw pitch is 10 mm, so in the joint settings I have selected a lead of 10 mm/rev, with a right-hand direction.

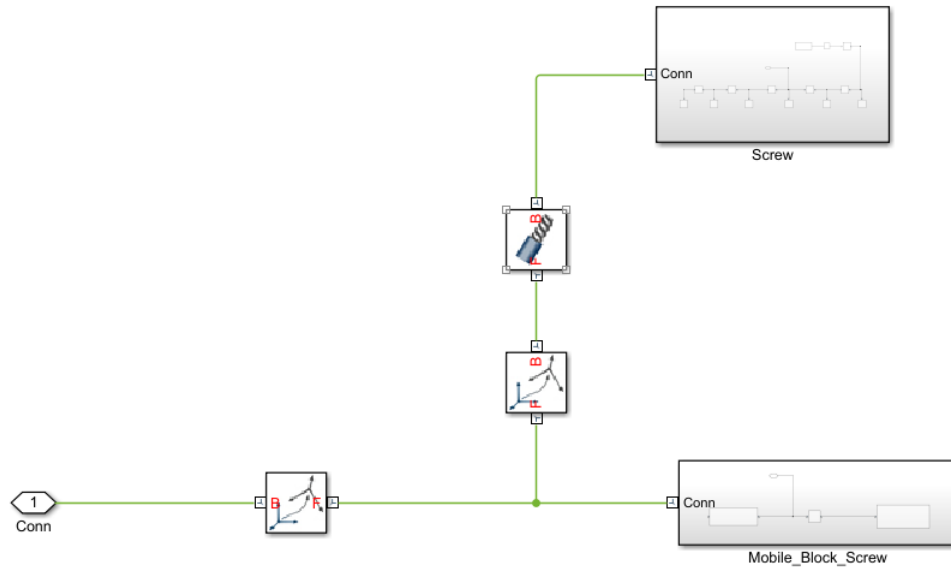


Figure 3.34: Screw model

Finally I have created a subsystem from these blocks connected to the ground to visualize it:

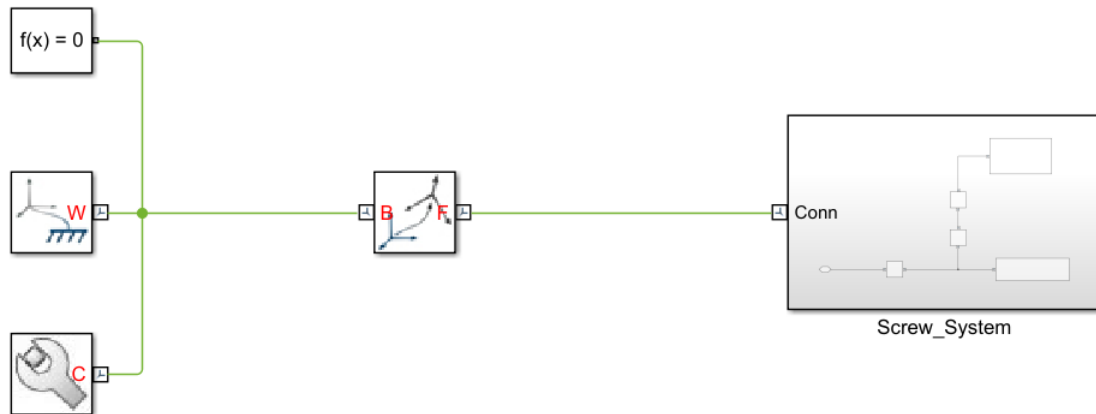


Figure 3.35: Screw subsystem

Base linchpin

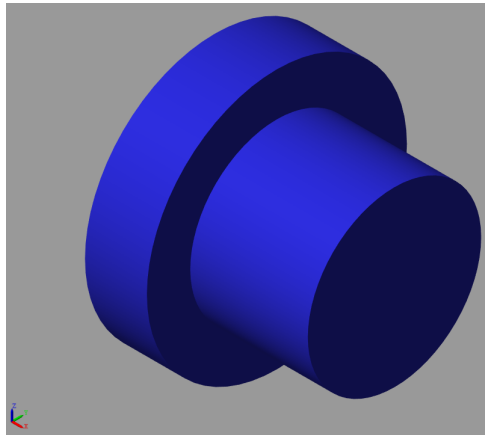


Figure 3.36: Base linchpin

The linchpin is connected to the base blocks and is a simple but very helpful block. In this case I have created it joining two cylinders with a Rigid Transform block. This element is then connected to each side of the base blocks and its Z axis is used as base for the Revolute Joint that allows the motion of the axes of the lifter.

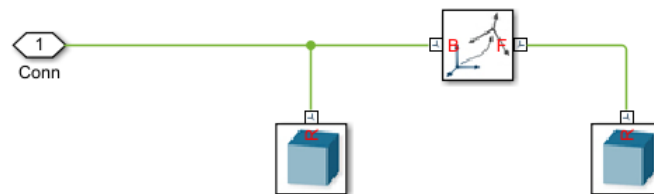


Figure 3.37: Base linchpin model

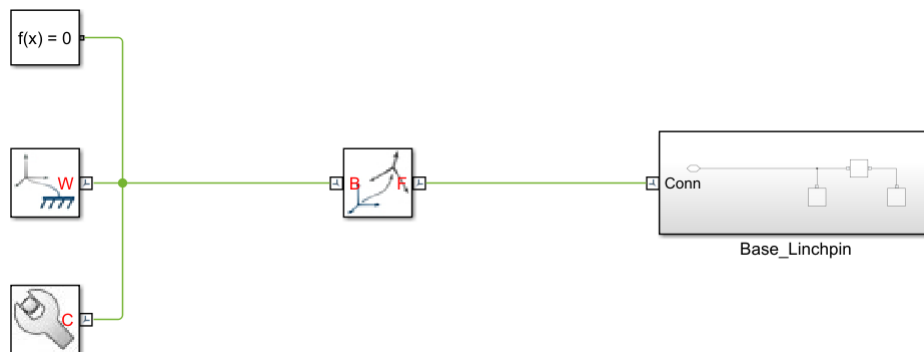


Figure 3.38: Base linchpin subsystem

Assembly of base system

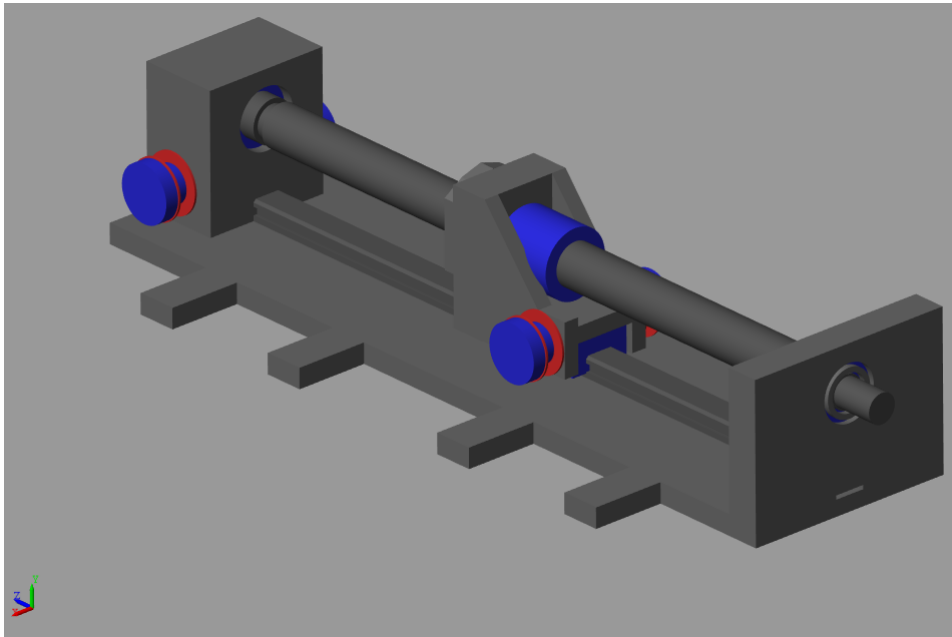


Figure 3.39: Assembly of base system

The whole base system is composed by all the previous blocks. In particular the track, the base block at the input side and the base block at the fixed side are connected to the base of the lifter with Rigid Transform blocks to correctly positioning them.

The mobile block is connected to the track with a Prismatic Joint to allow the translation. Then I have joined the mobile element of the screw system, that is connected to the screw with the Lead Screw Joint, to the mobile block. In this way I have obtained the correct movement of the mobile block that depends on the rotation of the screw. One problem that can occur during this phase is related to the correct creation of the Lead Screw Joint: to avoid a wrong behaviour it is better to join first the mobile element of the screw system to the mobile block, creating then the joint with the screw. In this way the mobile element of the screw system and the mobile base block are rigidly connected like a single block, and it is easier to create the complete system.

To fix also the position of the screw I have connected its extremities to the two blocks, that are fixed to the base of the lifter, through two Revolute Joints, to allow the rotation of the screw with respect to the base blocks.

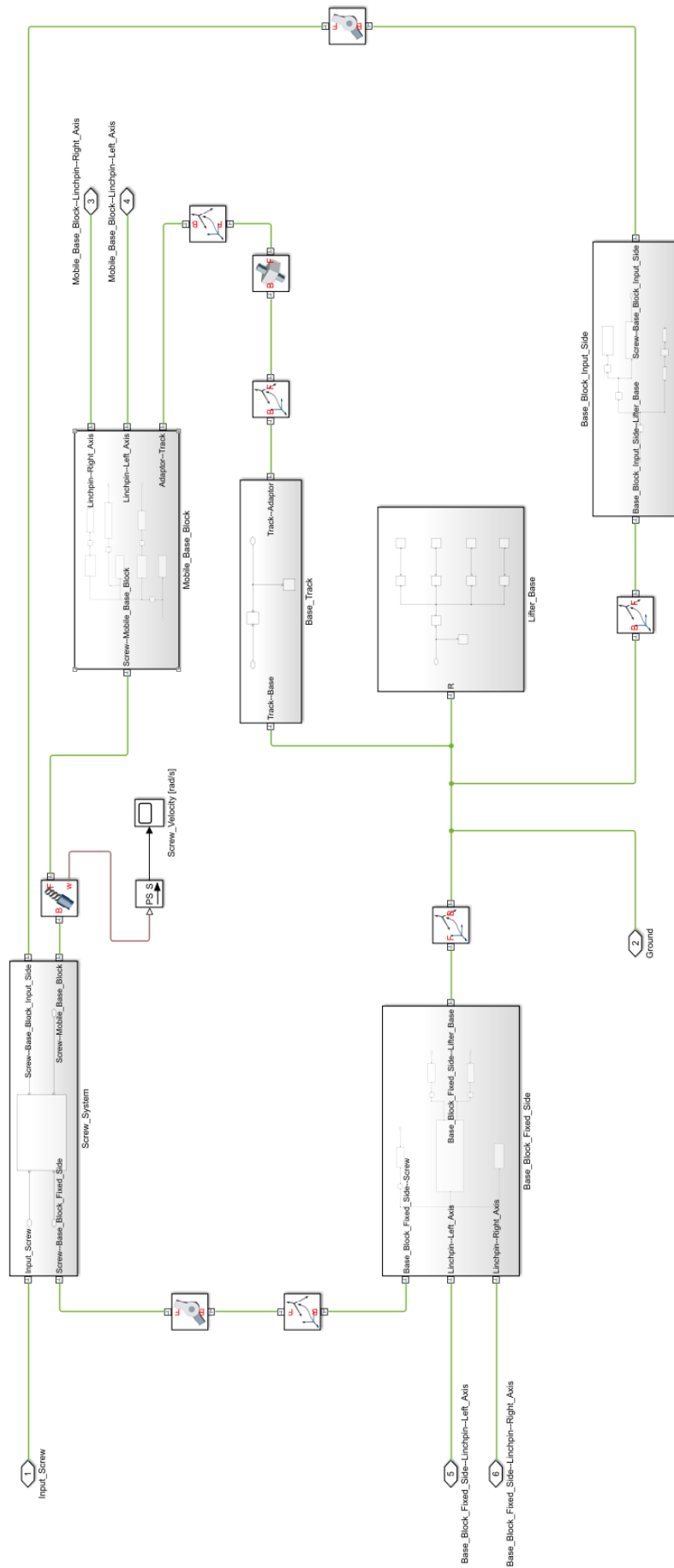


Figure 3.40: Assembly of base system model

3.4.2 Axes system

Single axis

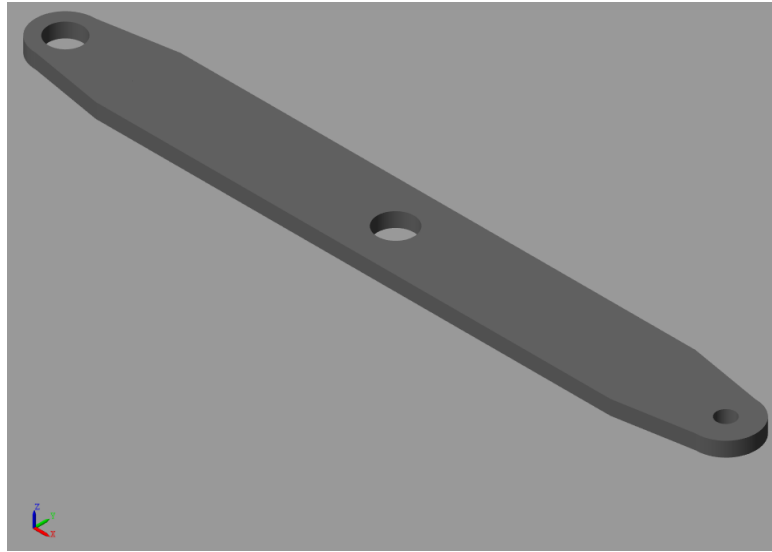


Figure 3.41: Single axis

Each single axis in been created adding with Rigid Transforms more links with one hole: in this way is possible to create an axis with holes of different diameters and at different distances. Then, using again Rigid Transforms to create a rigid system, I have added some solids to create a good shape for the axis.

In the next figure is possible to see the central part of the axis composed by different links with holes, and also other more little solids used to create the correct shape of the axis.

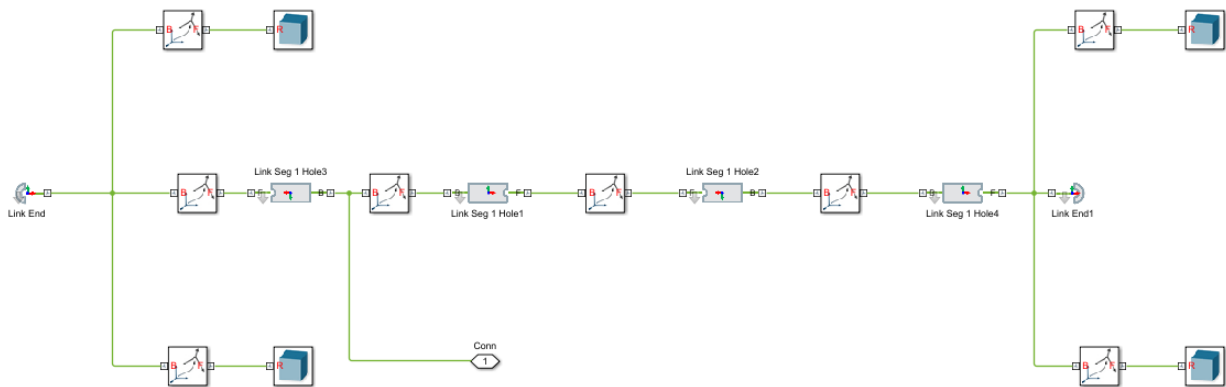


Figure 3.42: Single axis model

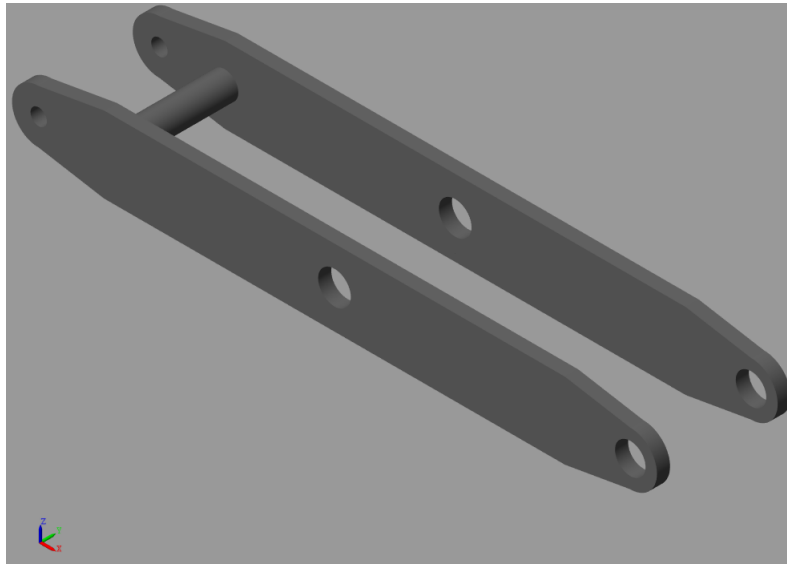
Assembly of two axes with a linchpin

Figure 3.43: Assembly of two axes with a linchpin

The next step to create the axes system is the creation of the linchpin used to connect two axes. The linchpin is a simple cylinder, that is connected to the axes through two Rigid Transforms. To do that, a connection port is been created to connect the linchpin at the correct point of the axis.

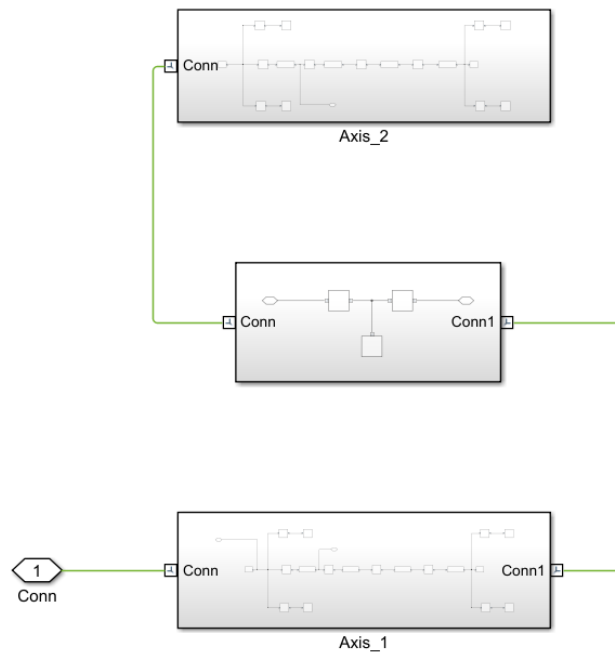


Figure 3.44: Assembly of two axes with a linchpin model

Then, as in the previous cases, a subsystem is been created to have a more clear visualization of the model.

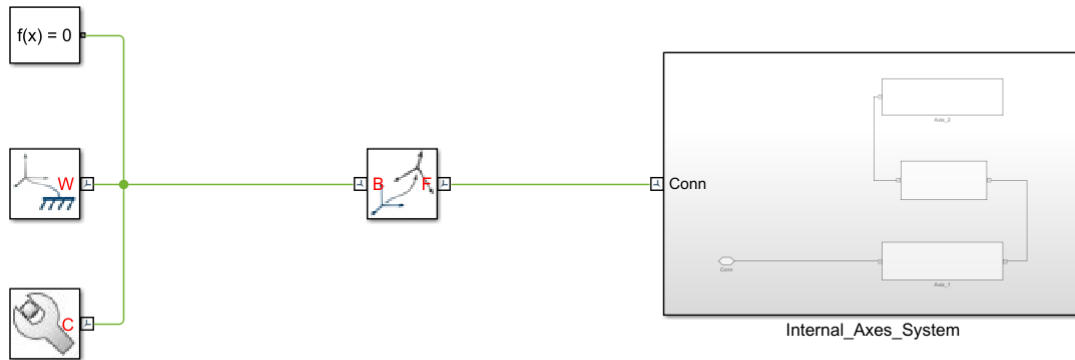


Figure 3.45: Assembly of two axes with a linchpin model subsystem

I have repeated this procedure to create also the other couple of axes.

Central bearings

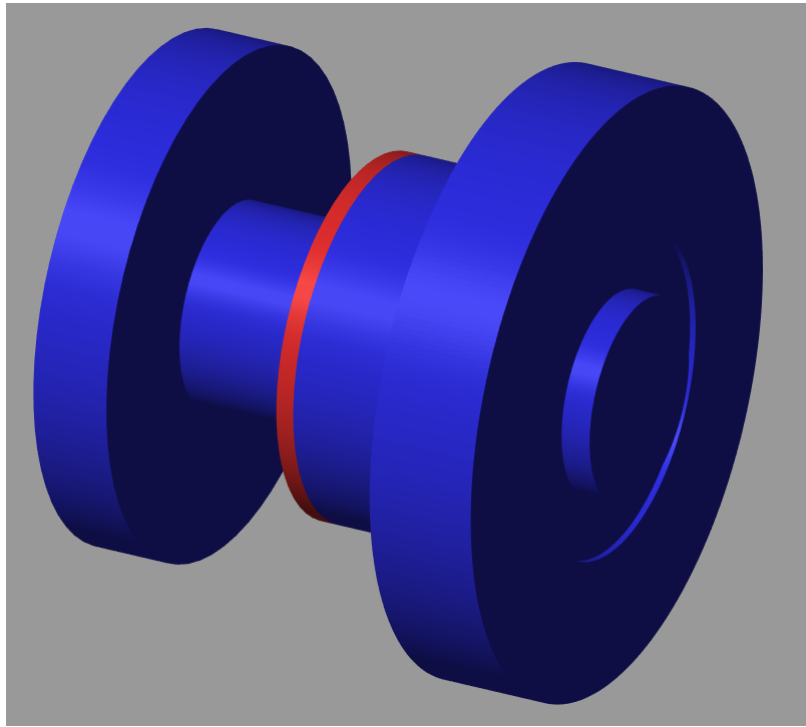


Figure 3.46: Central bearing

The central bearing represents the central connection between the two subsystems of axes. The two axes in the model have different diameters of the central holes, so to create the bearing I have added with different Rigid Transforms more cylinders of different dimensions. The red part is been added later and represents an adaptor to avoid to leave an empty space.

The model of the complete central bearing is represented in the following figure.

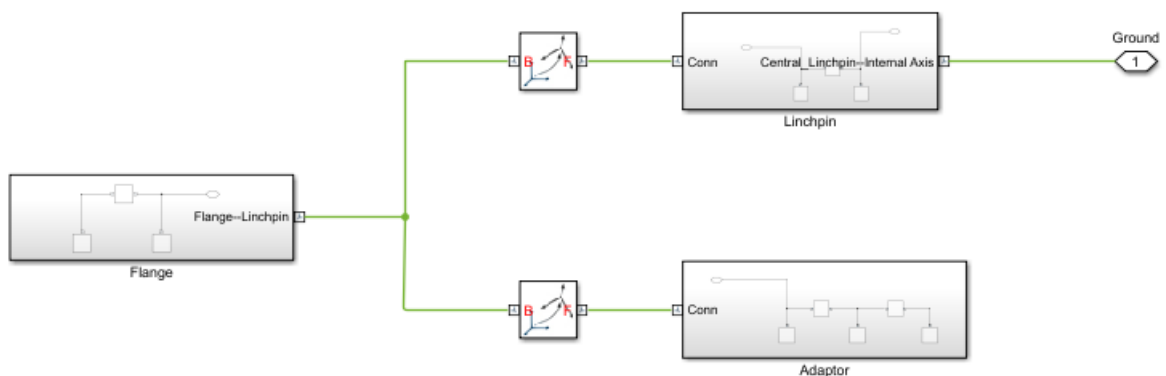


Figure 3.47: model of central bearing

After the creation of the subsystem the model appears like in the following figure.

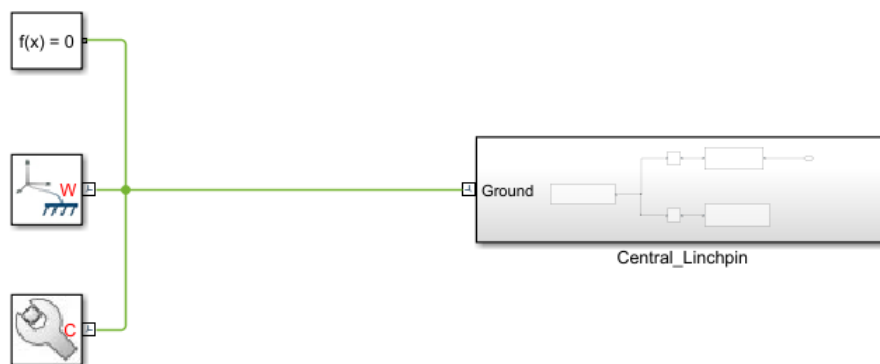


Figure 3.48: subsystem of model of central bearing

Assembly of all four axes with central bearings

Now is possible to create the axes system: to do that the two central bearings are been used as base for the revolute joints used to connect the whole system, while the follower for these joints is represented by the Z axis of the central hole of each axis system.

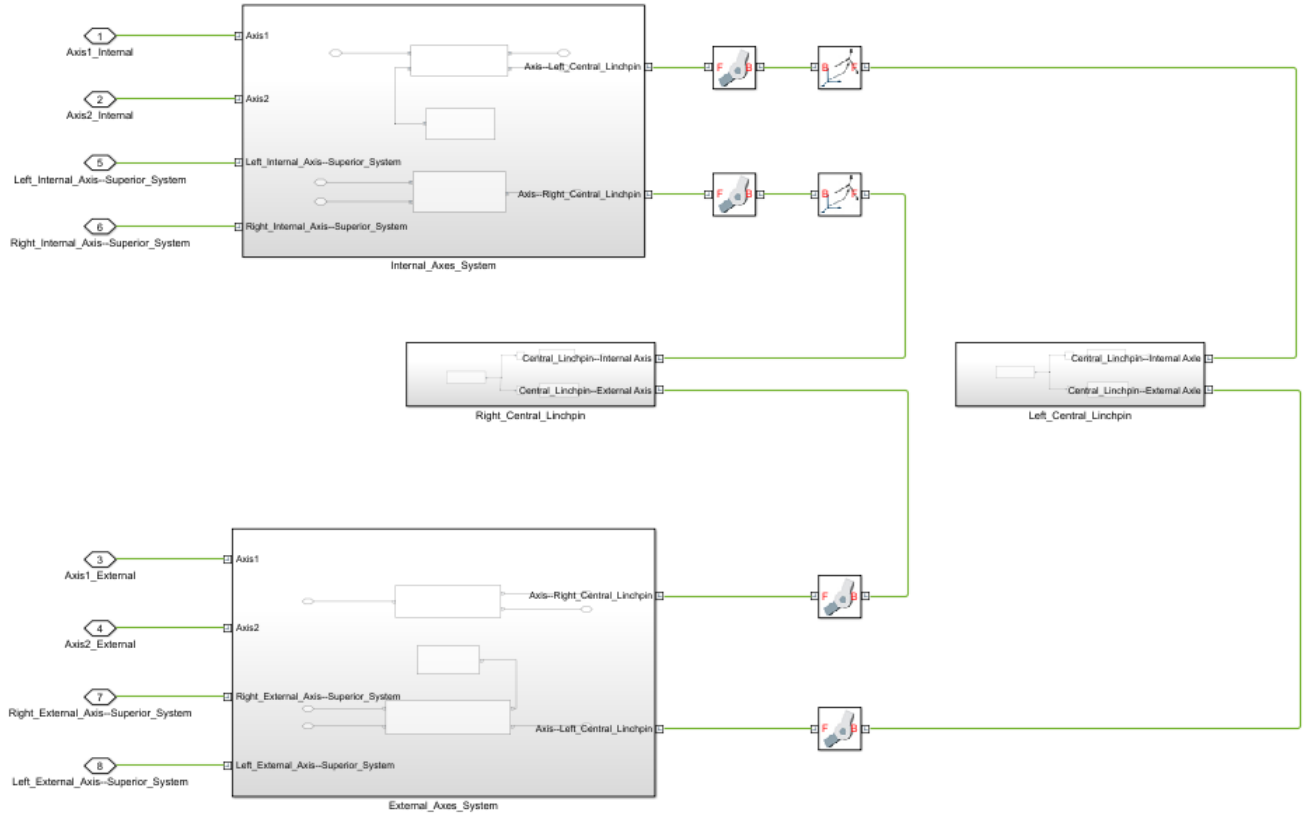


Figure 3.49: Axes system model

In particular, each axis is connected to the correct position on the central bearing: to do that I have added to the connection link a Rigid Transform to create a translation on the Z axis in addition to the Revolute Joint.

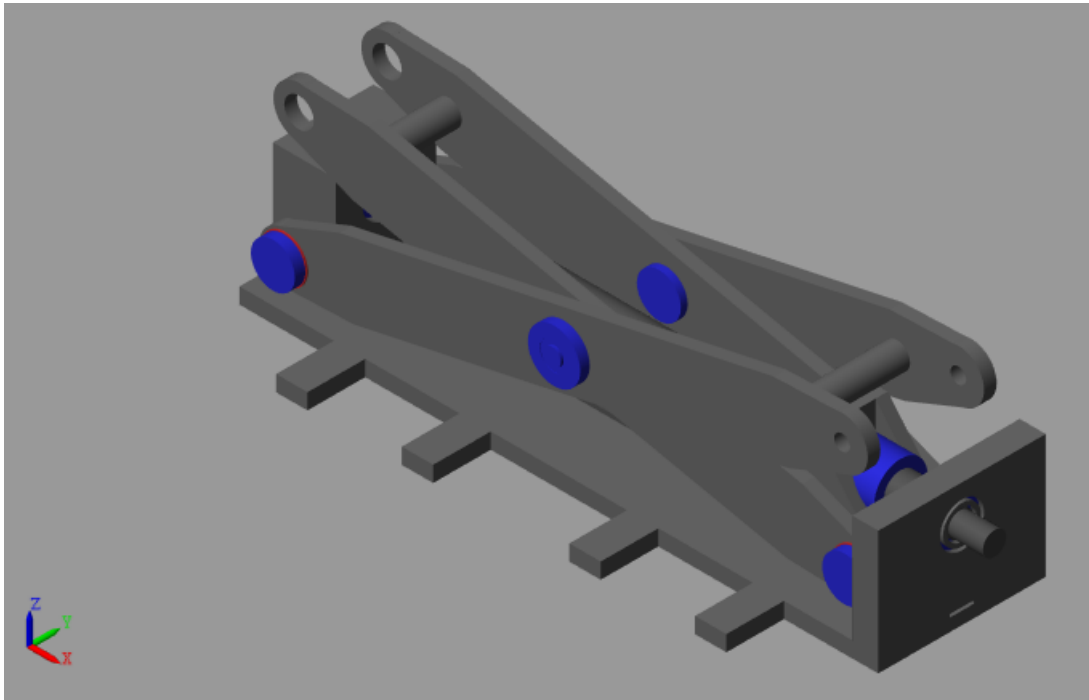
Assembly of axes system with base system

Figure 3.50: Assembly of axes system with base system

The connection between the axes system and the base systems is been created through Revolute Joints that are able to create the movement of the axes with respect to the base. In addition, Rigid Transforms are been used to correctly positioning the axes with respect to the bearings.

During this phase a problem occurred: Simscape Multibody gave an error related to the creation of a closed chain. This is a common problem during the creation of more complex systems, but there is the possibility to solve it. Each axis is connected to the base system and then will be connected also to the upper system. In addition, each axis is connected to the other axis through a linchpin. To solve the problem of the closed chain I have disconnected one of the two axes by the linchpin: in this way the axis remains correctly connected to the whole system without errors. In fact, into this system, when the axes are correctly connected, there is not a relative motion between the two axes of each subsystem, so this means that is not necessary to connect them also using a linchpin. Finally, the subsystems of the two couples of axes appear like in the next figure.

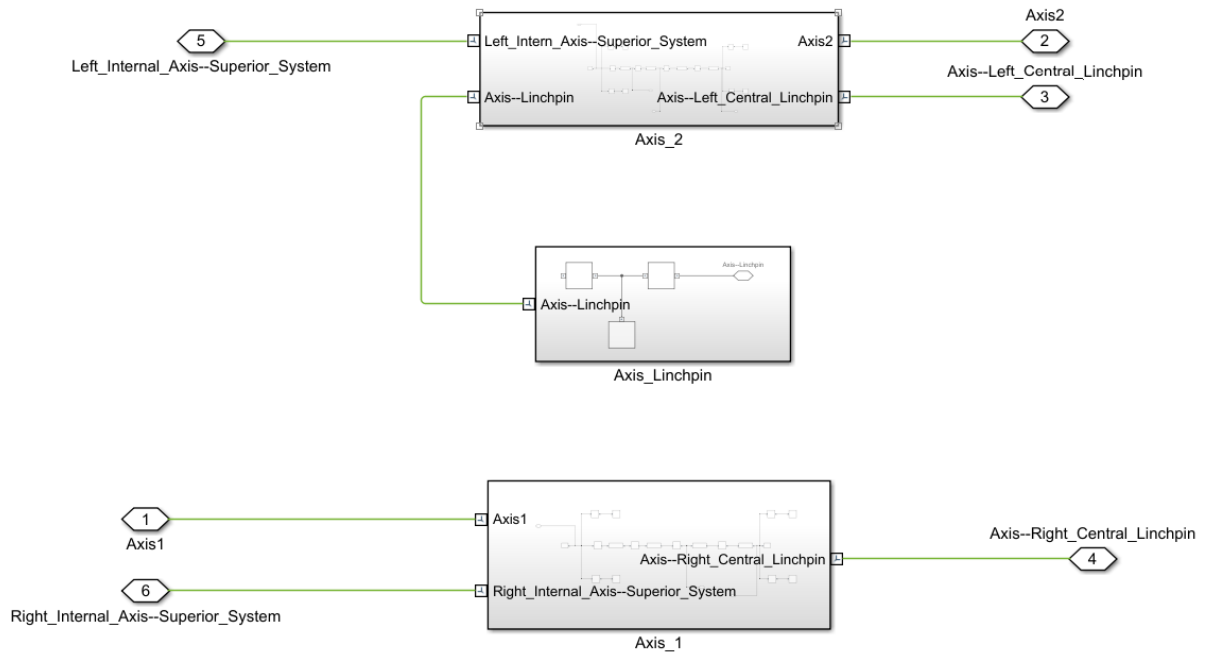


Figure 3.51: Axes system without closed chain

After the connection between the two systems and the solution of the problem mentioned before, the model appears like in the next figure.



Figure 3.52: Assembly of axes system with base system model

3.4.3 Upper system

Upper block fixed side

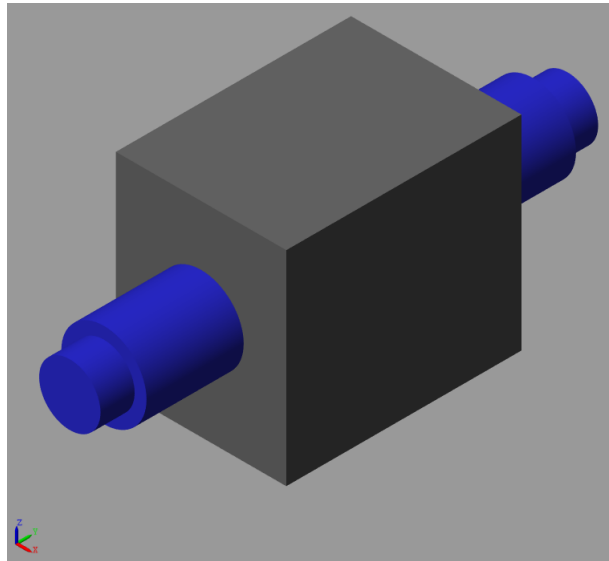


Figure 3.53: Upper block fixed side

The first element of the upper system that I have done is the block positioned at the fixed side. It is composed by a simple central block, that at the right and left side has two cylindrical elements that work as a linchpin for the axes. The elements at the right and left side are fixed with respect to the central block, so they are connected through Rigid Transforms.

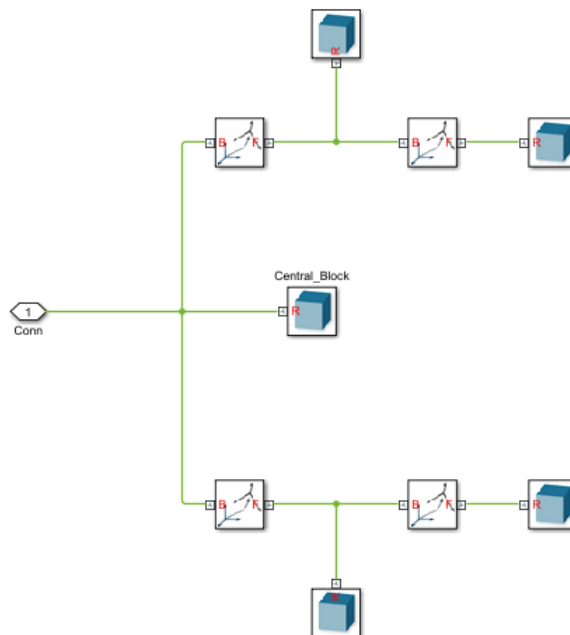


Figure 3.54: Upper block fixed side model

After the creation, the model is included into a subsystem for the same reasons of the previous cases.

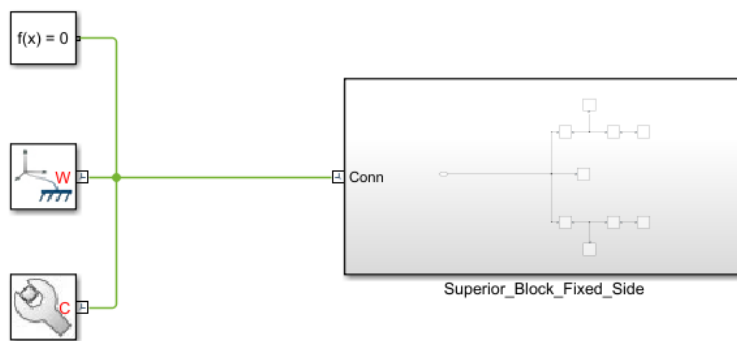


Figure 3.55: Upper block fixed side model subsystem

Upper plate

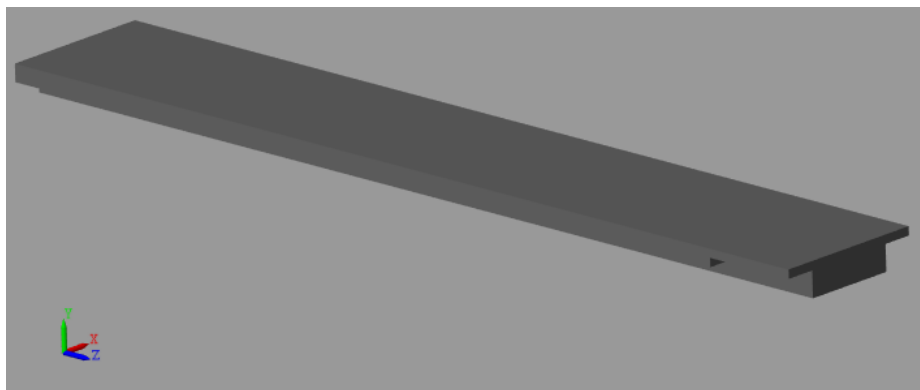


Figure 3.56: Upper plate

The upper plate is composed by two solids created through a general extrusion and then rigidly connected through a Rigid Transform. The following pictures show the shape created with a series of points on the XY plane and the model of the complete upper plate.

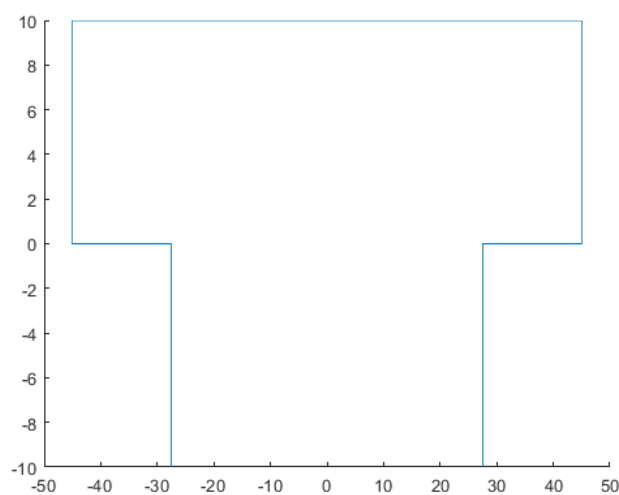


Figure 3.57: Upper plate XY plane general extrusion

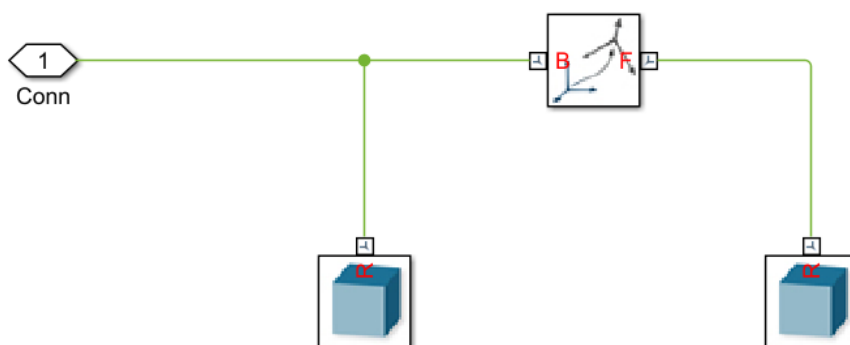


Figure 3.58: Upper plate model

Then the blocks are included into a subsystem.

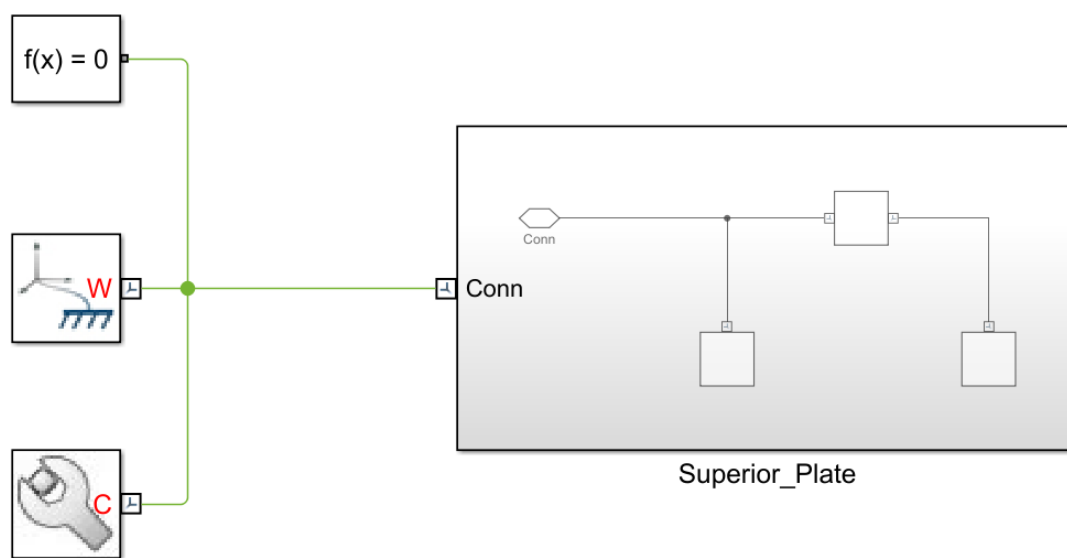


Figure 3.59: Upper plate model subsystem

Lateral plate

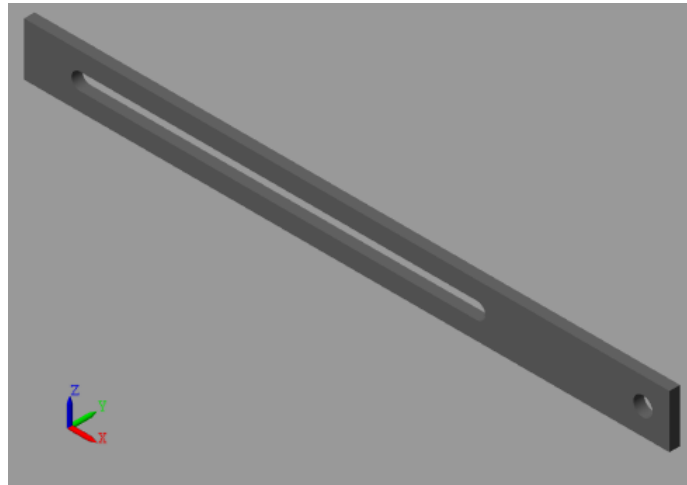


Figure 3.60: Lateral plate

To create this part of the upper system I have used again the element of the library called link with one hole. In particular I have used two of them, but in the second, due to the fact that the hole represents a rail for the mobile part of the lifter, I have added two solids between the two parts of the link.

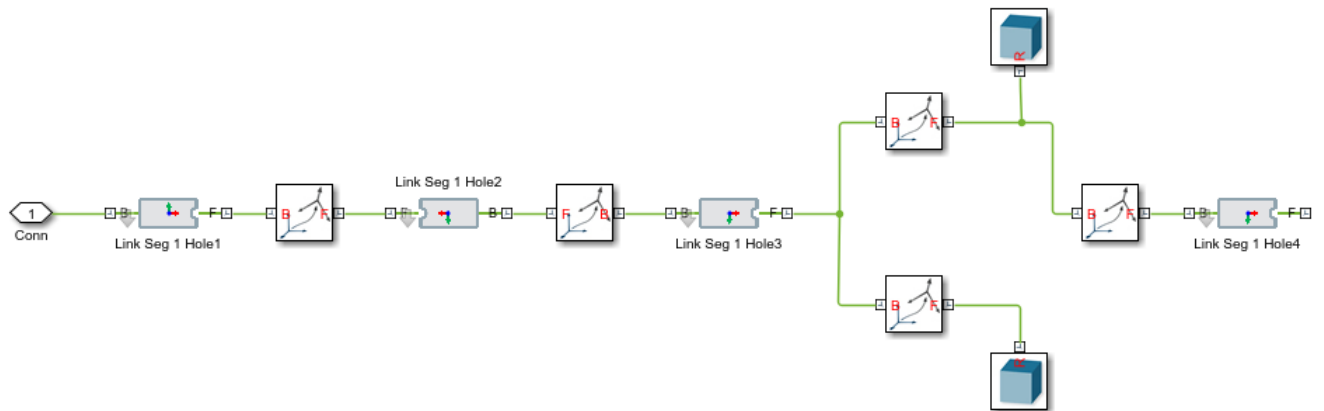


Figure 3.61: Lateral plate model

Then I have created a subsystem that is been used two times into the complete model of the upper part.

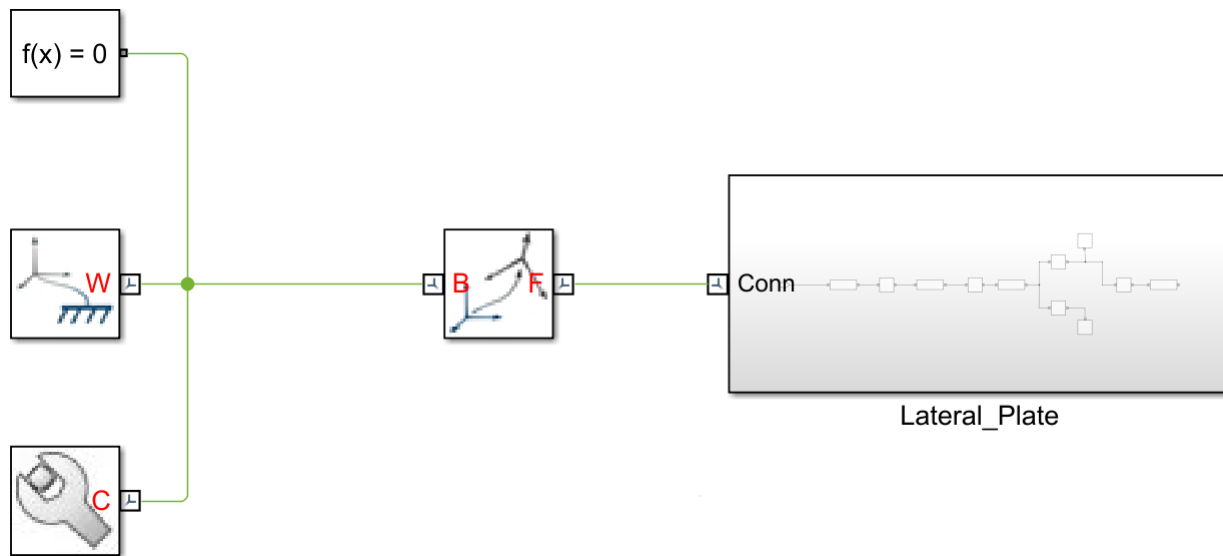


Figure 3.62: Lateral plate model subsystem

Assembly of upper system

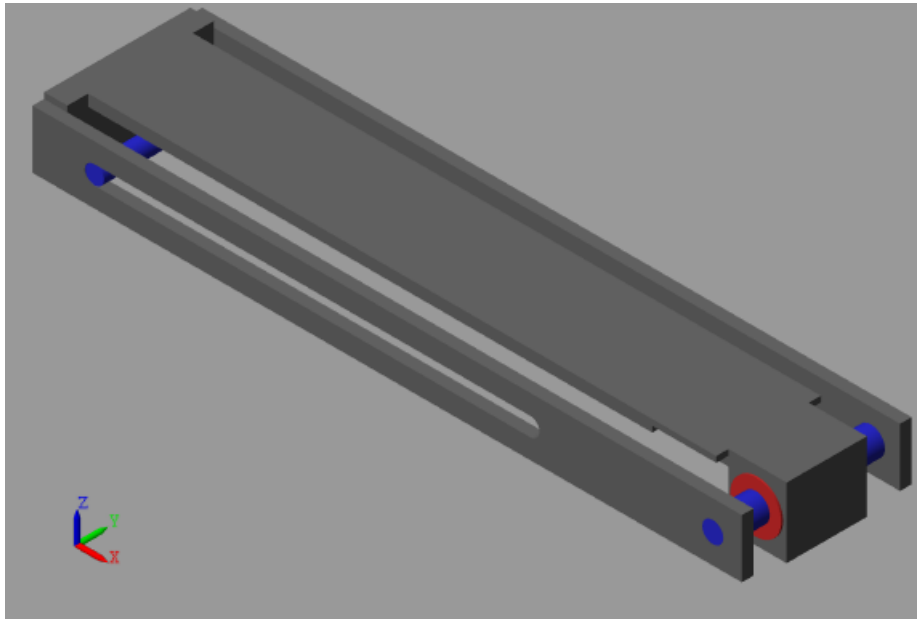


Figure 3.63: Upper system

The assembly of the upper system is composed by the previous elements and also a simple block positioned at the opposite side with respect to the block with linchpins, and a simple cylinder, that represents a shaft and it is important to create the relative movement between the axes and the upper system. This relative movement can be divided in two parts:

- Translation with respect to the upper system.
- Rotation with respect to the axes system.

The following figure shows the left part of the upper system, and in particular is possible to see on the left side the blocks used to create the translational movement between the shaft and the lateral plate.

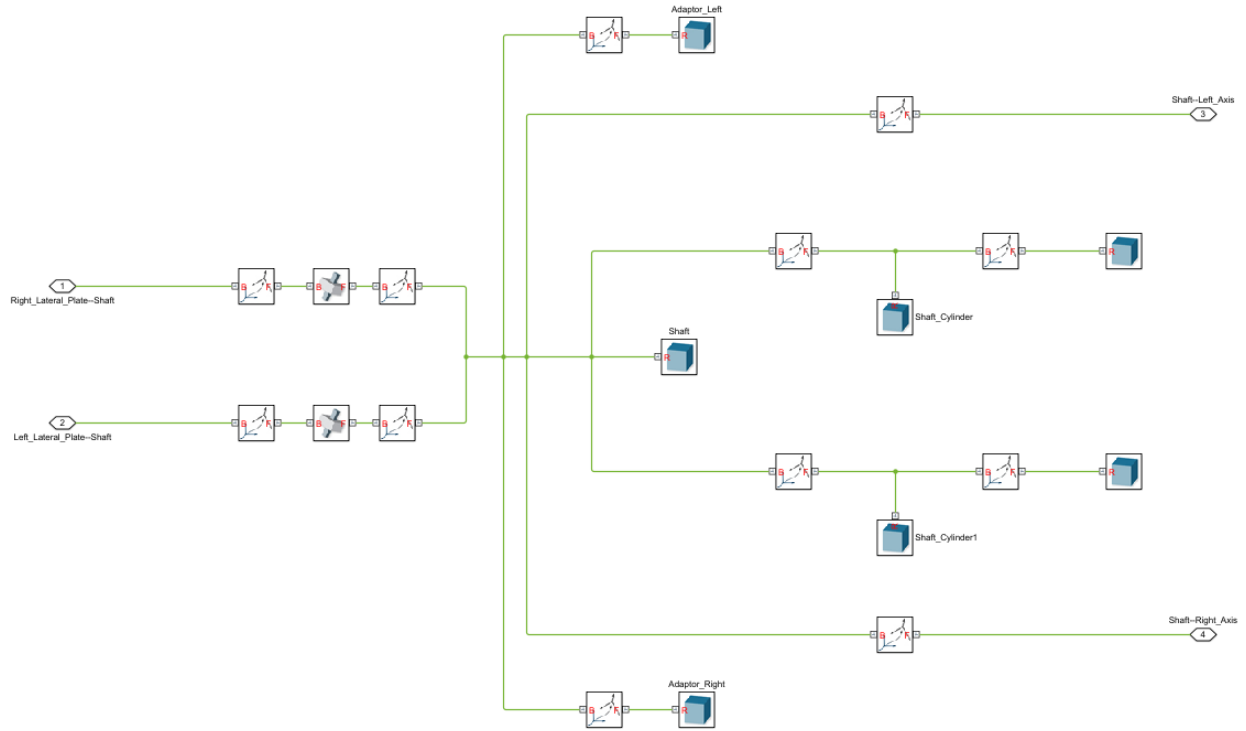


Figure 3.64: Upper system shaft side model

The creation of the upper system leads to a problem: also in this case there can be the creation of a closed chain, that creates an error for Simscape Multibody. To solve this problem I have not connected all the elements with Rigid Transforms, in particular the lateral plates: in fact, connecting only one side of the lateral plate it will remain fixed, because the Rigid Transform doesn't leave degrees of freedom. So, to create correctly the upper system I have rigidly connected the single blocks following the red arrows represented into the next figure, where it is possible to see the top side of the upper system. The last rigid connection, where the red cross is present, is not done to avoid the closed chain.

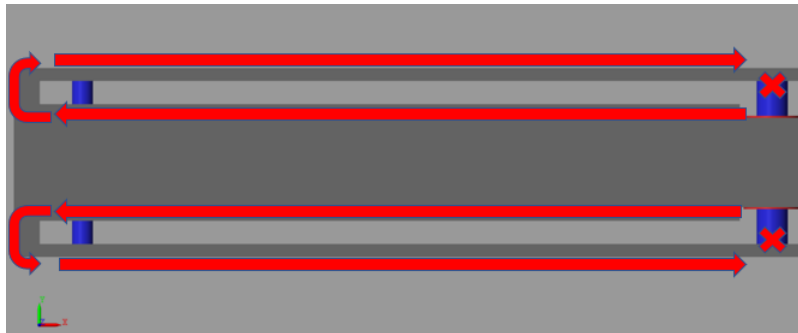


Figure 3.65: Upper system rigid connection sequence

Finally it is been possible to create the whole upper system as shown into the next figure.

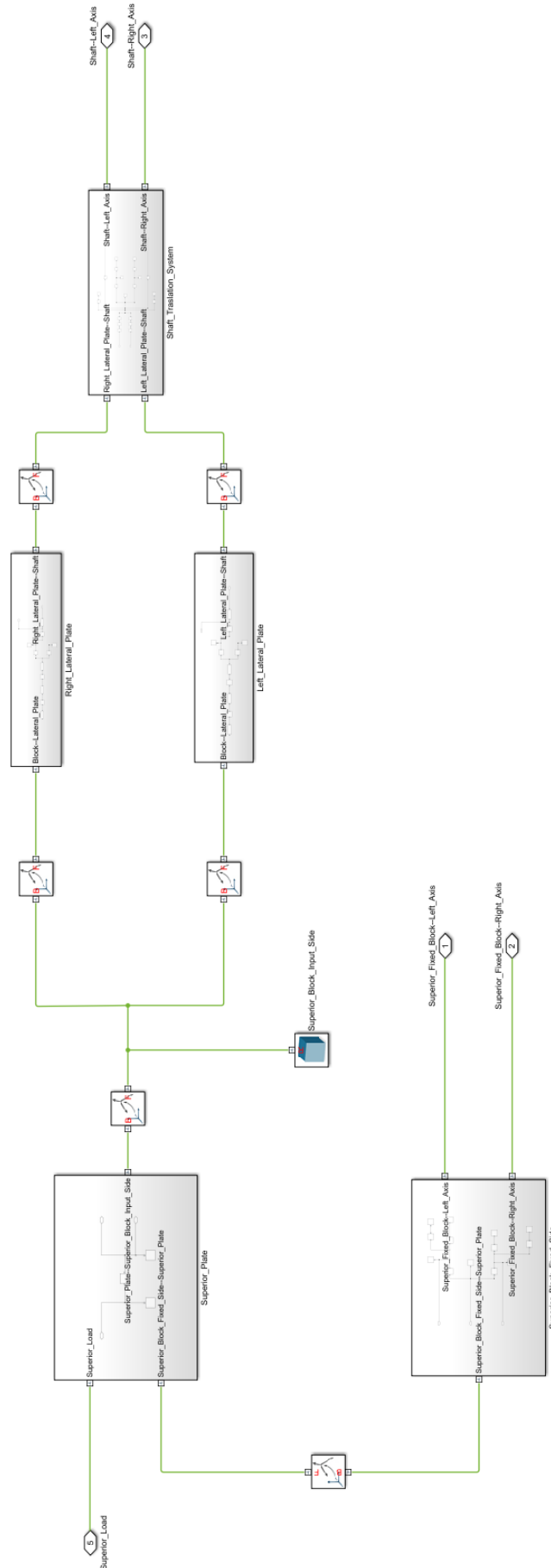


Figure 3.66: Upper system model

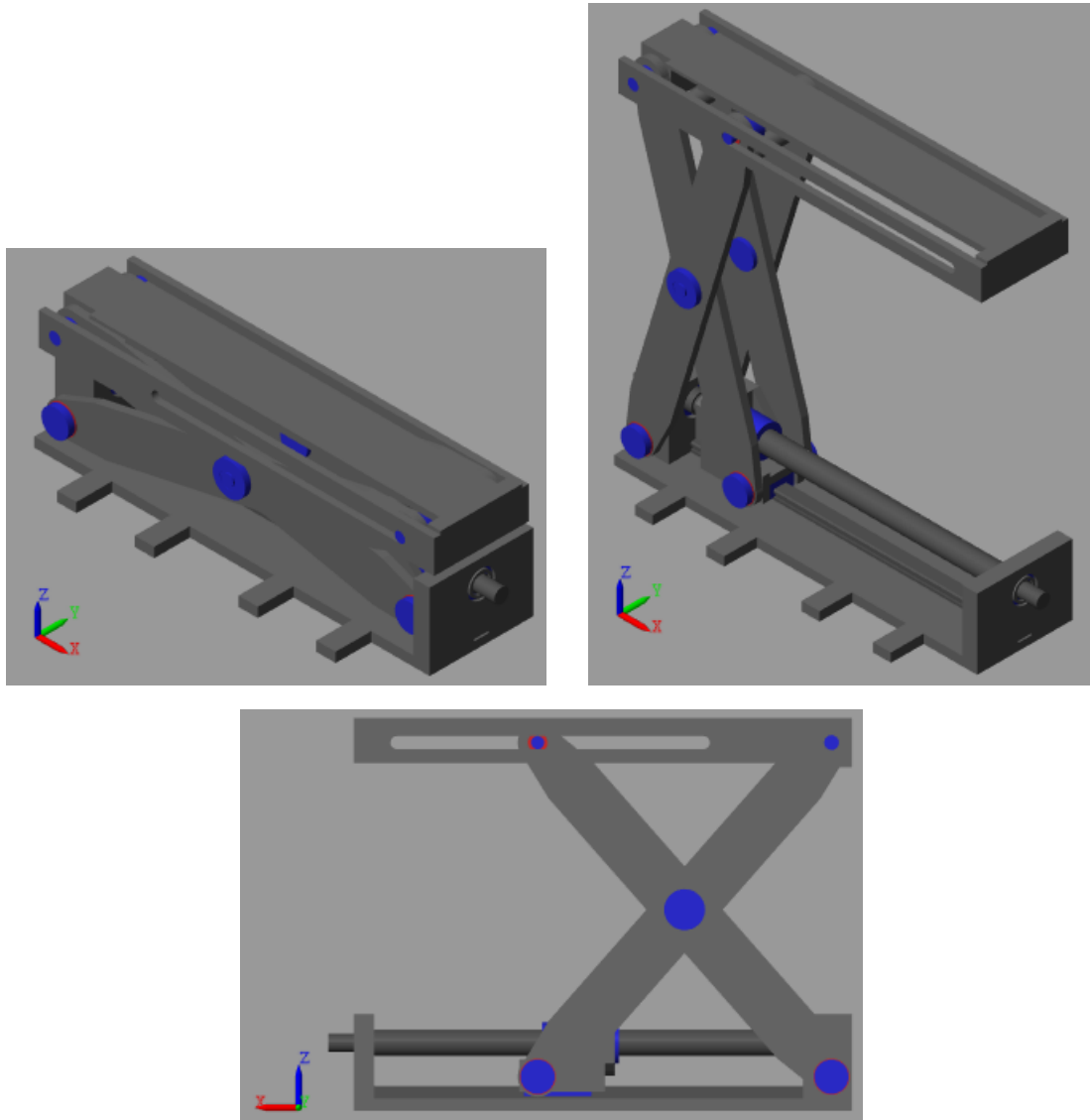
Assembly of upper system with axes system

Figure 3.67: Assembly of subsystems

To create the complete system of a lifter the last step is to add the upper system to the previous parts. As I have done during the connection between the axes system and the base system, now a revolute connection between axes and linchpins of the upper part is needed. From one side I have connected one axes subsystem to the fixed block's linchpins using Revolute Joints with Rigid Transforms to correctly positioning each axis with respect to its linchpin. From the other side I have done the same thing connecting the other axes subsystem to the shaft that is able to translate through the rail of the upper system.

A problem that occurred during this phase is related to the correct position of the upper system: without any control on the Revolute Joints it was positioned in a reversed way. To avoid this situation, I have added a position target to two Revolute Joints to correctly positioning the system: I have set the target with a low priority because I don't know the angle exactly.

Then, to create the correct starting position also for the mobile element of the screw, I have modified the position of the reference frame used for the connection between the track and the mobile element, and, after that, I have set the target position of the Prismatic Joint at zero, this time with an high priority, because I want this starting position complied with.

To see the movement of the mobile element on the screw and to control that the system is correct, I have imposed a set of positions in time to the Prismatic Joint of the mobile element on the track: to do that I have imposed into the Prismatic Joint Actuation that the Motion is "Provided by Input" and the Force is "Automatically Computed". Then I have created a simple signal to move the mobile element from 0 to 380 mm on the track in 6 seconds and the return back again in 6 seconds:

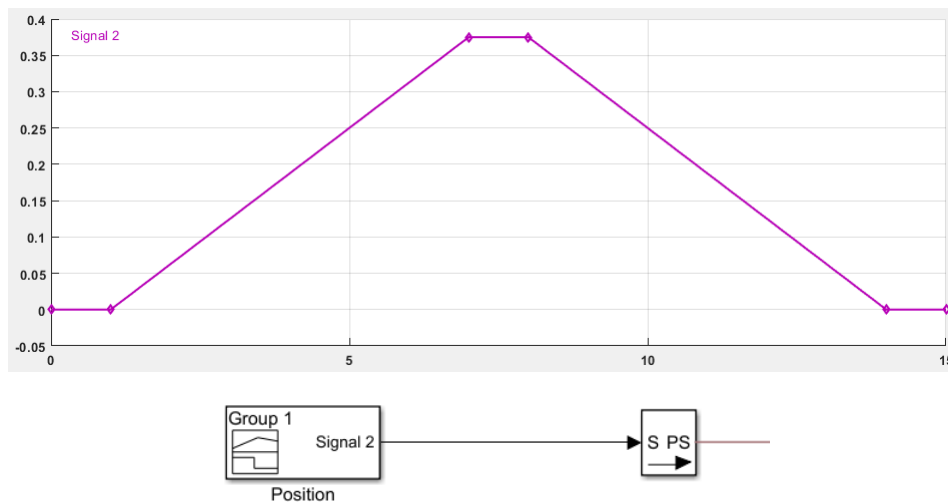


Figure 3.68: Position control for Prismatic Joint

To connect this type of signal that represents the position in time a Simulink-PS Converter is needed: this block using this information is able to automatically compute also velocity and acceleration. Into the parameters section of the block is possible to select "Filter input, derivatives calculated" in the "Filtering and derivatives" menu, and "Second-order filtering" into the "Input filtering order" menu. Now running the model is possible to see the movement of the lifter.

The complete system of the lifter can be seen in the following figure, and it is divided into three subsystems: base, axes, and upper part.

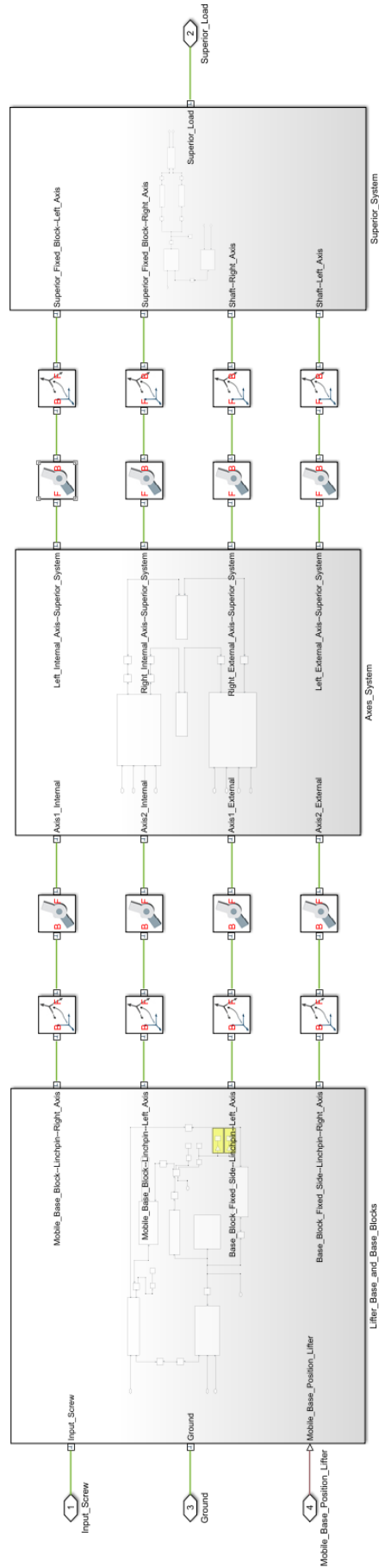


Figure 3.69: Lifter model

3.4.4 Complete model with Rack and Pinion system

To create the whole system of the lifter it is possible to duplicate the lifter and then it is necessary to link the two lifters, using the Rack and Pinion model, that is also connected to the motor. When all elements will be connected, the pinion of the motor will provide the correct torque to move the rack and so the other two pinions connected to it, to create the target motion of the system.

The Rack and Pinion system is composed by three principal elements:

- Rack: this element is able to translate with respect to the ground following the motion of the Pinion. All parameters like number of teeth and dimensions of the rack can be set into the mask.



Figure 3.70: Rack

- Pinion: this element is able to rotate with a motion linked to the Rack. Into the mask is possible to set all the options related to dimensions of the block.

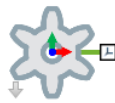


Figure 3.71: Pinion

- Rack and Pinion Constraint: this block is able to create the connection between Rack and Pinion. The only parameter that is needed is the radius of the Pinion to set the distance between the two blocks. The Pinion represents the Base of the block while the Rack represents the Follower of the block.



Figure 3.72: Rack and Pinion Constraint

So, a simple Rack and Pinion system can be represented like in the following figure.

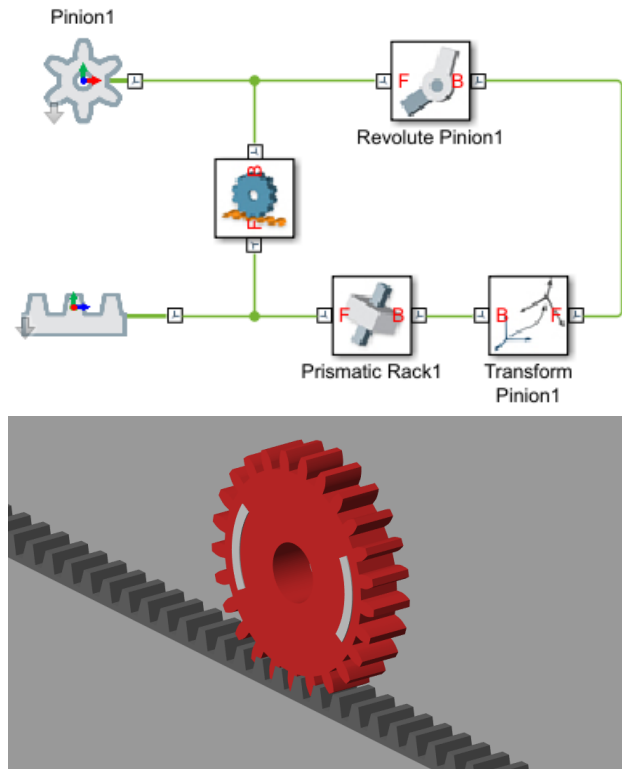


Figure 3.73: Rack and Pinion model

In the complete system of the lifters each pinion is been connected to a screw, while an additional pinion is been connected to the shaft of the motor, a simple block where is present the Revolute Joint that will be used to sense the motion parameters or give the motion inputs.

During this phase of the project I have had some problems related to the positioning of the Rack and Pinion system: in fact, the rack and also each pinion must be positioned into the correct position and with the correct orientation with respect to the ground, otherwise it is impossible to obtain a correct behaviour. So, into the next figure it is possible to see the two pinions connected to the same rack, with all the Rigid Transforms needed to correctly positioning each element of the system with respect to the ground.

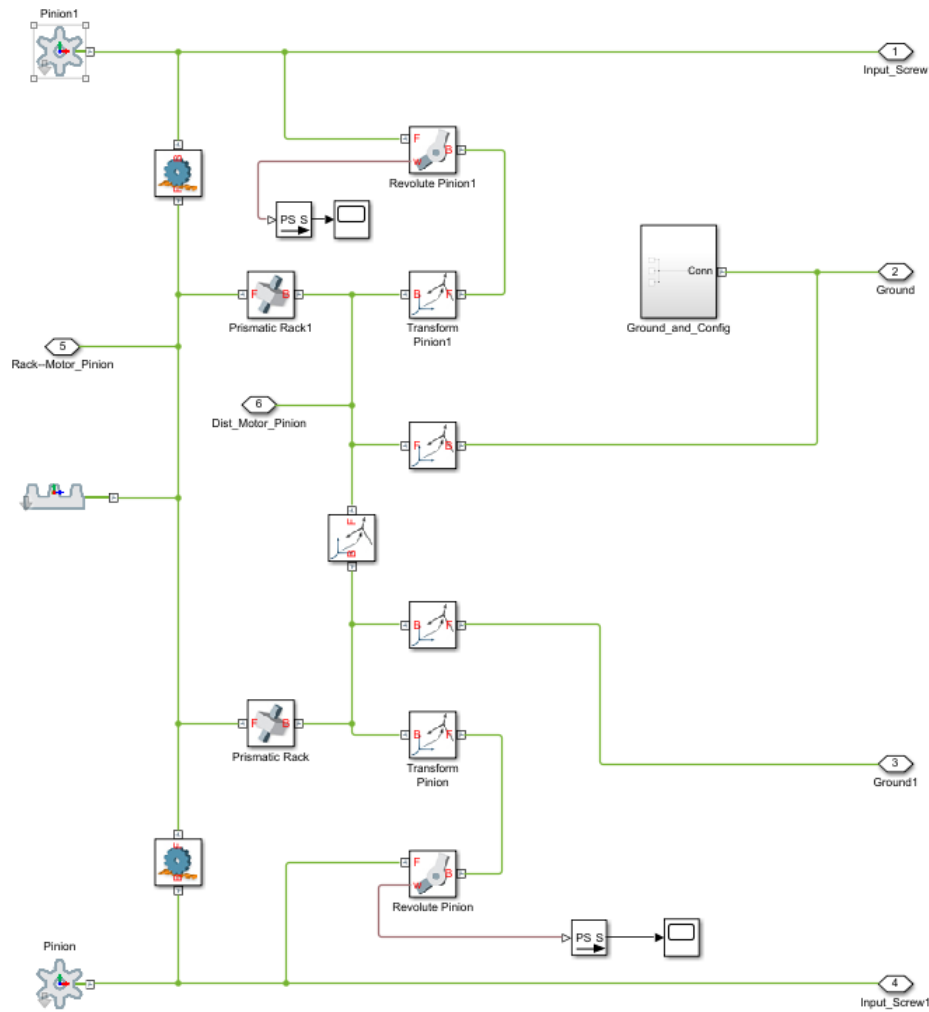


Figure 3.74: Rack and Pinions of the lifters

After the creation also of the motor pinion, connected to the same rack of the two lifters, I have added to the system a simple block with a mass of 600 kg to represent the maximum load that the complete lifter can lift.

So, now the project is complete and in the next two figures it is possible to see the complete Simcape Multibody's model and the 3D representation that appears compiling the model, at the starting position and at the higher point reachable by the lifter.

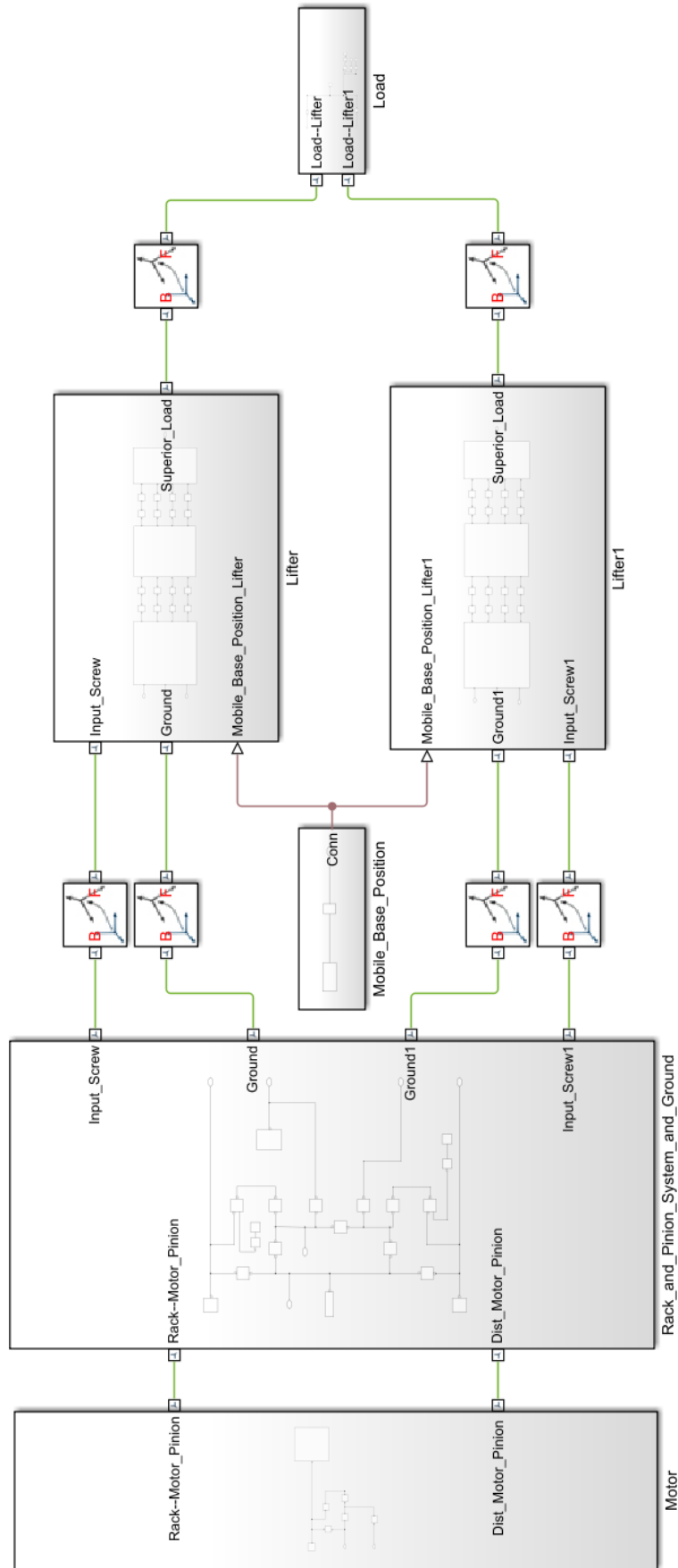


Figure 3.75: Complete model

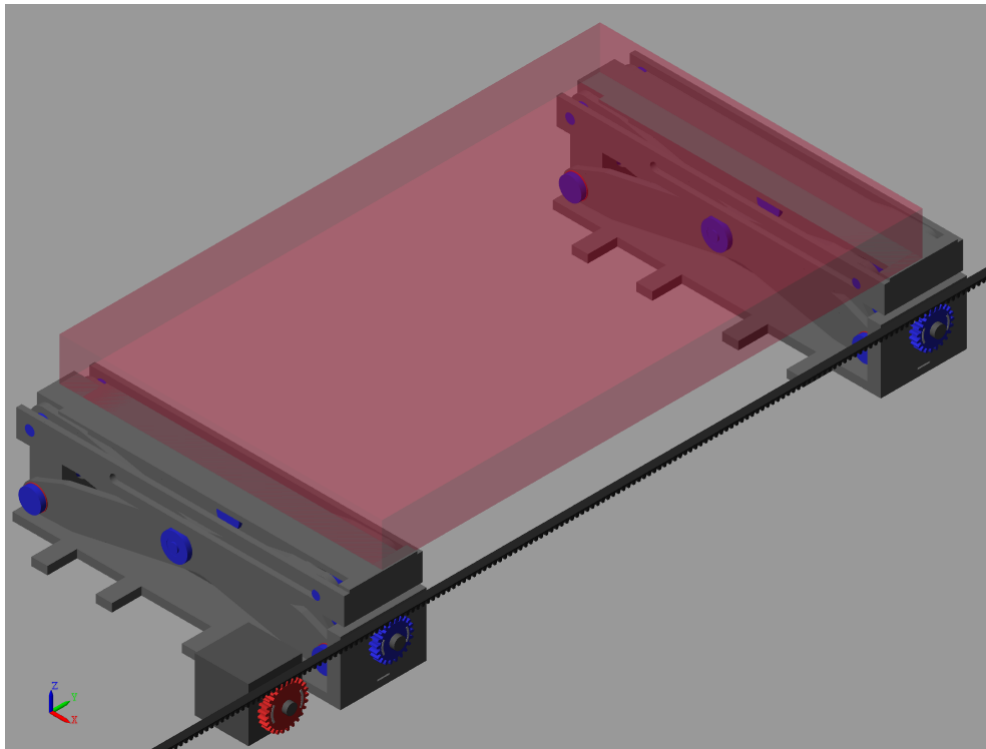


Figure 3.76: Complete model starting position

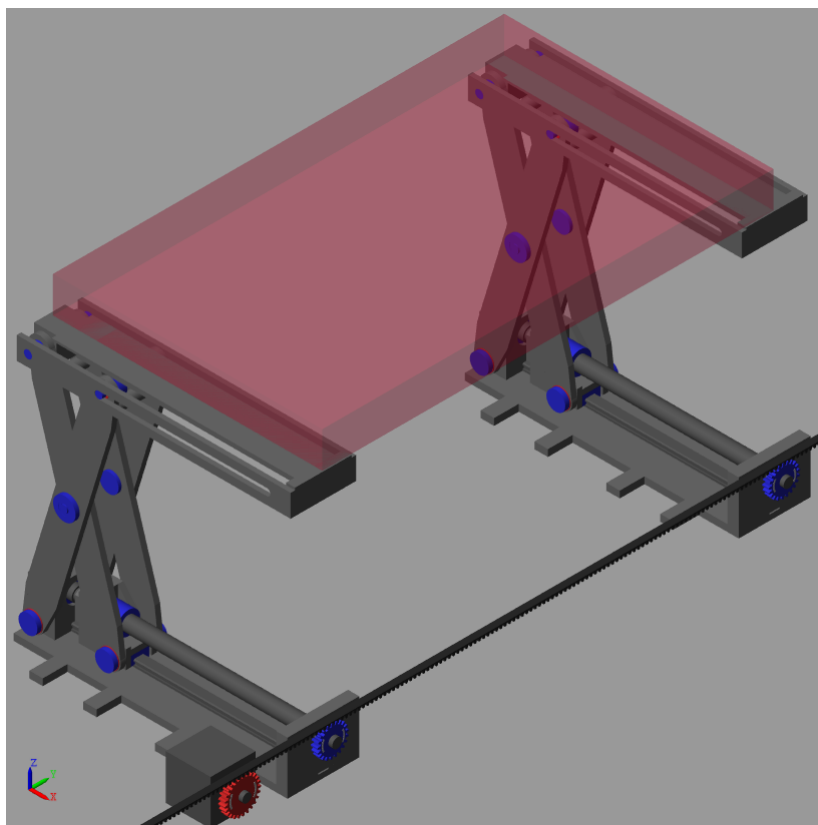


Figure 3.77: Complete model end of stroke

3.5 Model import from SolidWorks

One of the features of Simscape Multibody makes possible to import a CAD model: then the model can be modified and for example a motion input can be added to see the behaviour of it. The CAD software, for example SolidWorks, must be able to export the model in a specific format that can be read by MATLAB (.xml).

The first step is to install the Simscape Multibody link, that is a necessary plug-in for the export-import phases. Then in SolidWorks will be possible to select a new add-on to enable the CAD export.

The export procedure will produce a set of files: the .STEP ones represent the single elements of the system, while the .xml file is the one that will be opened in MATLAB to generated the Simscape Multibody model. With the function `smimport` in the command line is possible to select this .xml file and then MATLAB will be able to generate the model taking also all information from the .STEP files of each component.

To test this feature of MATLAB I have opened the model of a single lifter and I have exported it from SolidWorks. Then, after the import operation, running the model I have obtained the following model, that is perfectly equal to the CAD model, but with the possibility to study also its dynamic.

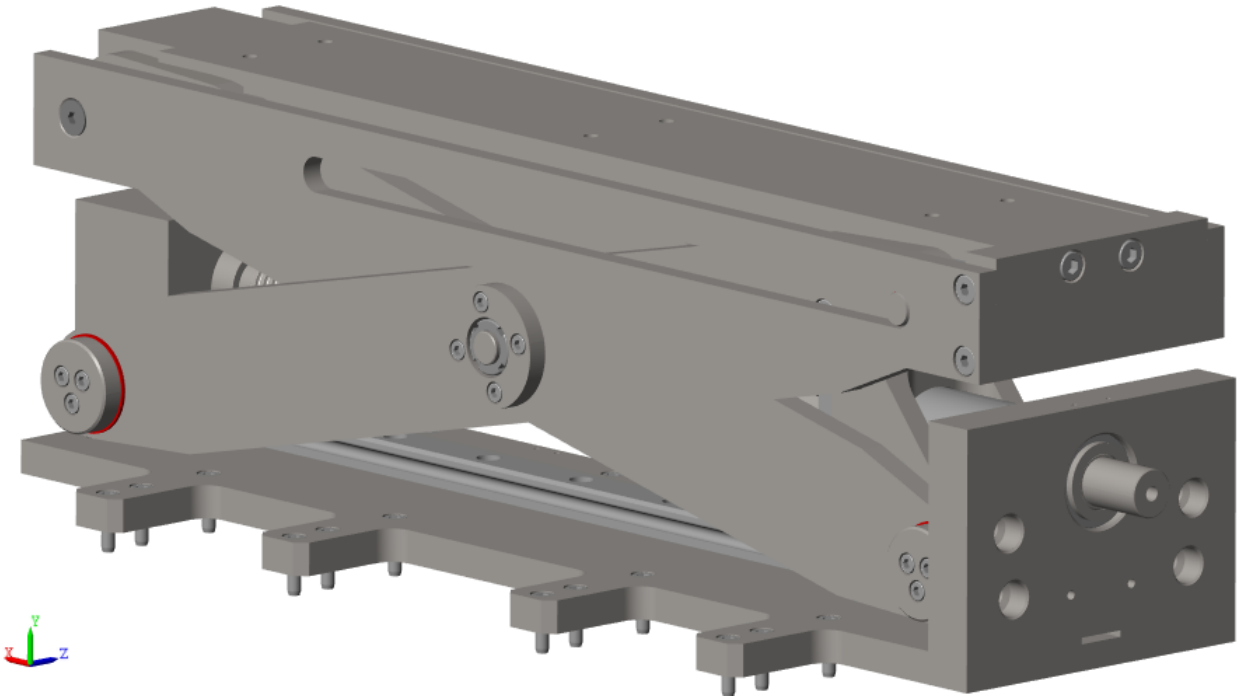


Figure 3.78: Model imported from SolidWorks

This feature of MATLAB could be used also in other cases: for example, if it is necessary to model one single part of a system that has a particular shape in Simscape Multibody, it can be easily created in SolidWorks and then imported into the model.

Chapter 4

Simulation and Validation

4.1 Friction values

The last step to complete the model for the simulation is to insert the values of friction, otherwise the model will not work properly.

Into the real model all joints are composed by ball bearings, so there are really low values of friction. Simscape Multibody's blocks provide a set of parameters related to the internal mechanics that can be set for each joint:

- **Equilibrium Position:** this value represents the rotation angle (or distance) at which the spring torque (or force) is zero.
- **Spring Stiffness:** this is the torque (or force) required to obtain a motion of a unit angle (or distance).
- **Damping Coefficient:** this is the torque (or force) required to maintain a constant joint primitive angular (or linear) velocity between base and follower frames.

Due to the fact that all frictions are really low into the model, I have decided to specify a low value of damping only to obtain a correct behaviour during the simulation. In particular, I have set a value of damping about 10^{-5} Nm/(deg/s) for all Revolute Joints that represent ball bearings in the real model. Then I have set a little bit bigger value of damping (10^{-4} N/(m/s)) for the Prismatic Joint that represents the motion of the mobile element, because this block has a relative motion with respect to the track but also with respect to the screw, and so it can be expected to have a little bit more friction. Now the model is complete for the simulation: as in the real case, the frictions do not have a remarkable effect on the motion, so the first and only element to consider is the effect of the load on the structure.

4.2 Solver Profiler

When the model is complete, to see if the solver works properly and if there are factors affecting the simulation, it is possible to use an instrument of Matlab: Solver Profiler.

The Solver Profiler presents graphical and statistical information about the simulation, solver settings, events, and errors. Then is possible to use this data to identify locations in the model that caused simulation bottlenecks. There are multiple factors that can limit the simulation speed: zero-crossing events, solver exception events, solver reset events, Jacobian computation events.

In particular, into this model now, as described in the last part of the previous chapter, I'm controlling the motion of the mobile block with a signal that can be divided in five phases:

1. The block remains at the start position for 1 second.
2. It goes from 0 to 380 mm in 6 seconds.
3. It stays at 380 mm for 1 second.
4. It goes from 380 to 0 mm in 6 seconds.
5. It stays at 0 mm for 1 second.

Running the simulation with this tool I have obtained the following graph that represents the trend of the step size.

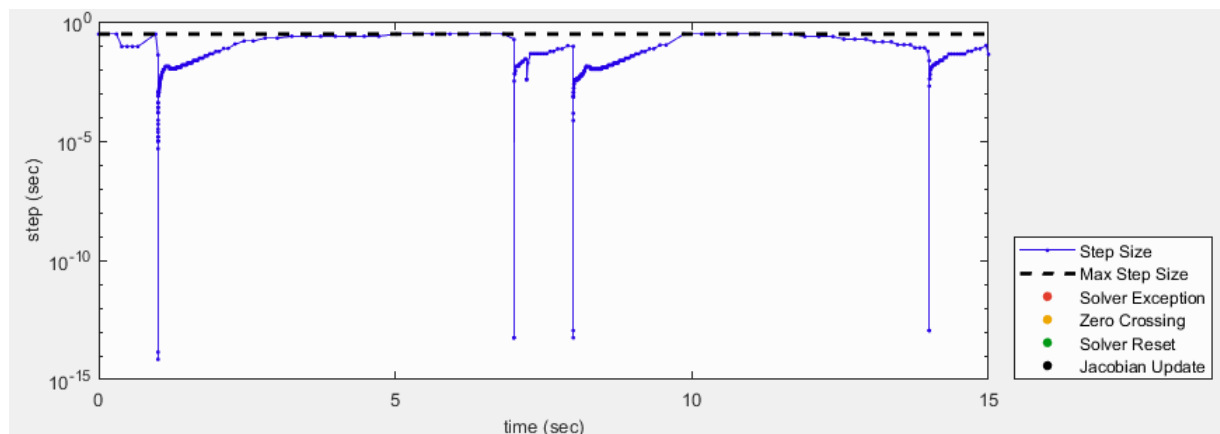


Figure 4.1: Step size

It is possible to see that the step size is rather constant during the constant velocity phases, while there is a decrease of it in correspondence of changes of type of motion. This is a normal behaviour for the step size, so this means that the simulation runs correctly.

4.3 Force and indirect torque sensing

The first goal after the creation of the model was to obtain the torque trend during the simulation. In this case, thanks to the input signal that I have created for the mobile block (Figure 3.68), I have the possibility to see the curves trend during the ascent and on descent of the lifter. The problem during this phase was that into the Lead Screw Joint is not possible to sense the torque. So, I have calculated it indirectly.

The first element that I have visualized was the velocity at the Prismatic Joint, to verify the motion.

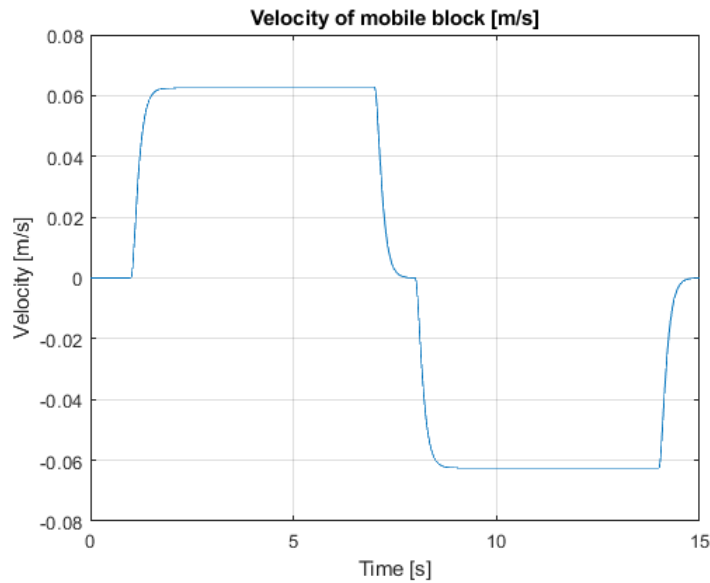


Figure 4.2: Velocity of mobile block

During the constant velocity phase, the velocity value is around 63 mm/s. This is the correct value because the motion is about 380 mm in 6 seconds.

A second element that I have checked was the rotational velocity of the screw measured at the Lead Screw Joint.

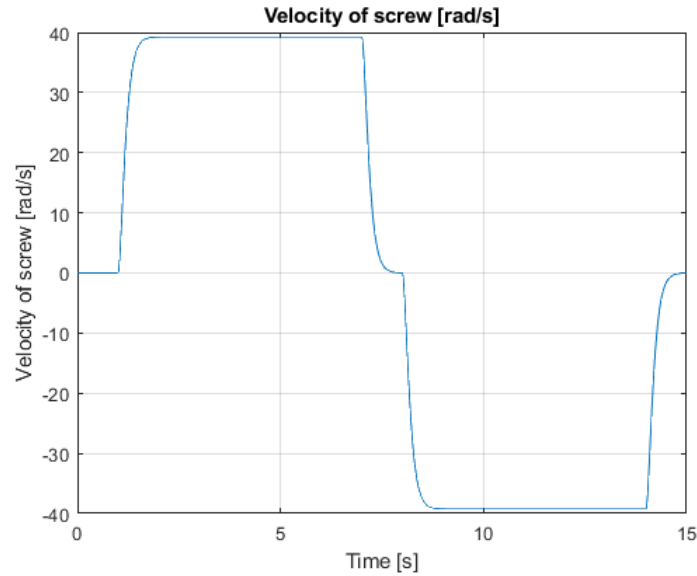


Figure 4.3: Velocity of the screw

Also this measurement was good because with a lead of the screw of 10 mm the correct velocity must be around 40 rad/s.

Another graph that can be obtained is the vertical position of the load, that starts from 0 and arrives to 400 mm on the Z axis. To sense this motion, I have had to use another method, because there is not in this case a simple joint from which is possible to sense the position. The Simscape Multibody's library provides a block called Transform Sensor, that can be connected from the base side to the ground and from the follower side to the reference frame to control. Then into the block is possible to select which values I need to measure. The structure in this case will be like in the following figure.

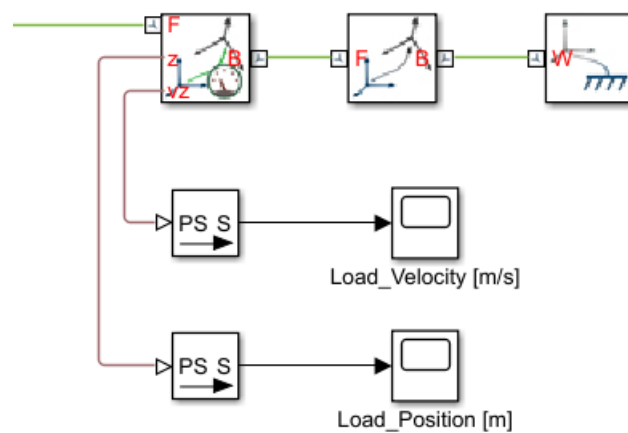


Figure 4.4: Model for load sensing

So, the load position during the simulation will be like in the following figure, and it is as expected: due to the axes system of the lifter the change of the vertical position is not constant, because the initial velocity is higher than the final one.

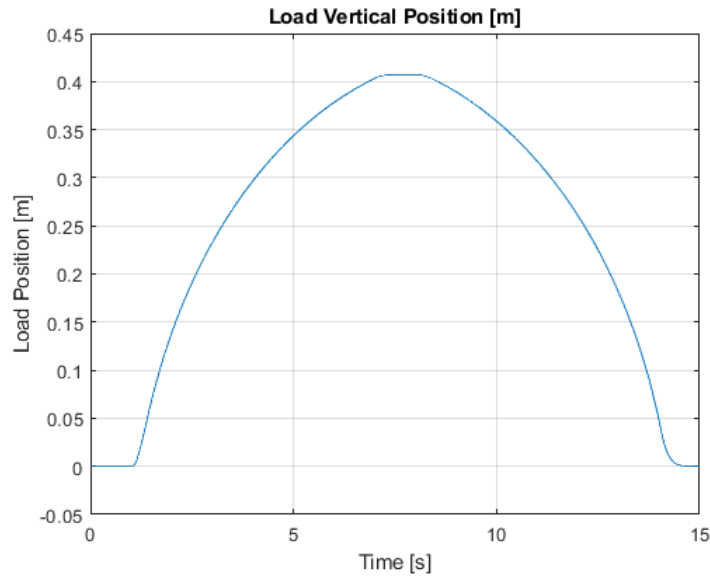
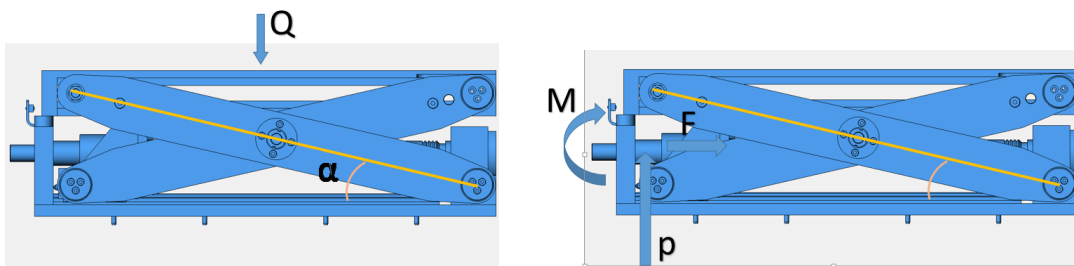


Figure 4.5: Load position on Z axis

Then I have checked another really important parameter for the model: the horizontal force on the mobile element on the screw. The maximum load for the system is about 6000 N, but the horizontal force is changing during the motion of the system due to the position of the axes. In particular, the horizontal force can be evaluated using the following formula: $F = Q / \tan(\alpha)$, where α is the angle between the axes and the ground.



At the starting position with this formula it is possible to obtain a value around 14000-15000 N for the single lifter. Sensing the force into the model during simulation it is possible to obtain the graph with the value of it during the motion of the system.

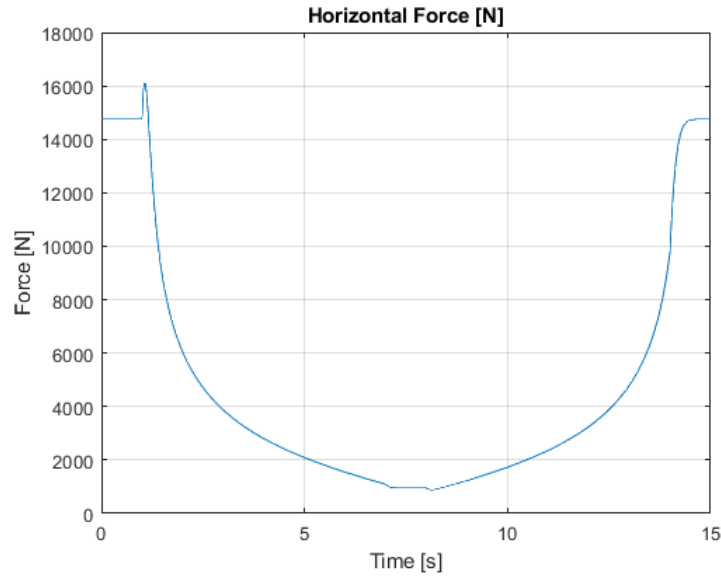


Figure 4.6: Load position on Z axis

Due to the problem related to the torque measurement that I have mentioned before, I have calculated indirectly the torque from these values of the horizontal force. To do that I have used a Gain block to represent the formula that links the force and the torque that the system must give to the screw to allow the movement of the lifter. The formula that links force and torque in this case is the following one:

$$T = \frac{F \cdot p}{2000 \cdot \pi \cdot \eta}$$

Where p is the lead of the screw and η is the efficiency of the lifter, in this case set to 1.

Into the model this method appears like in the following figure.

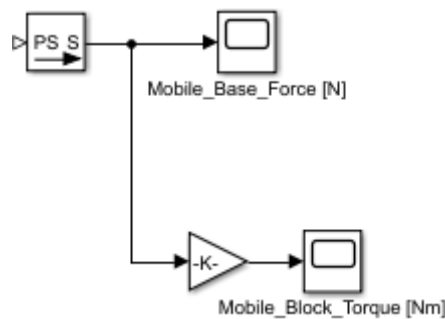


Figure 4.7: Torque sensing model

The torque obtained for the single lifter is represented in the following graph, and the values are as expected, because using the previous formula, for example, for the starting position it is possible to obtain a torque value around 24000 Nm.

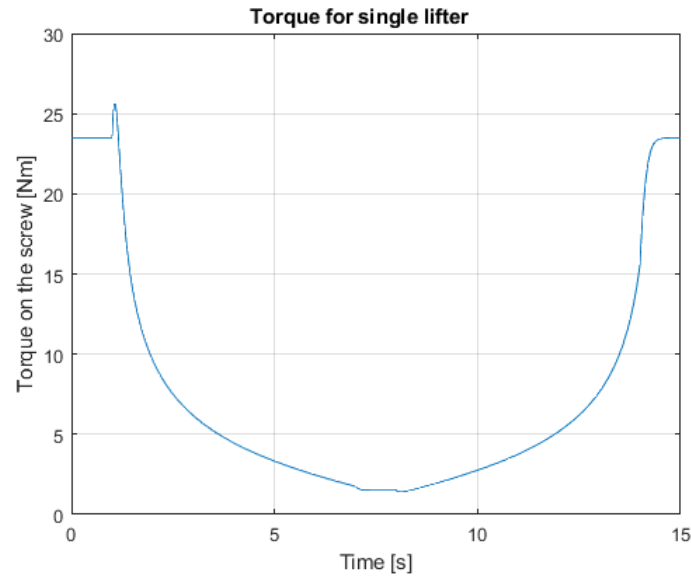


Figure 4.8: Torque for single lifter

With the same data I have also calculated the torque required at the motor pinion for two lifters, including also an efficiency value of 0,9. The result is represented in the following graph.

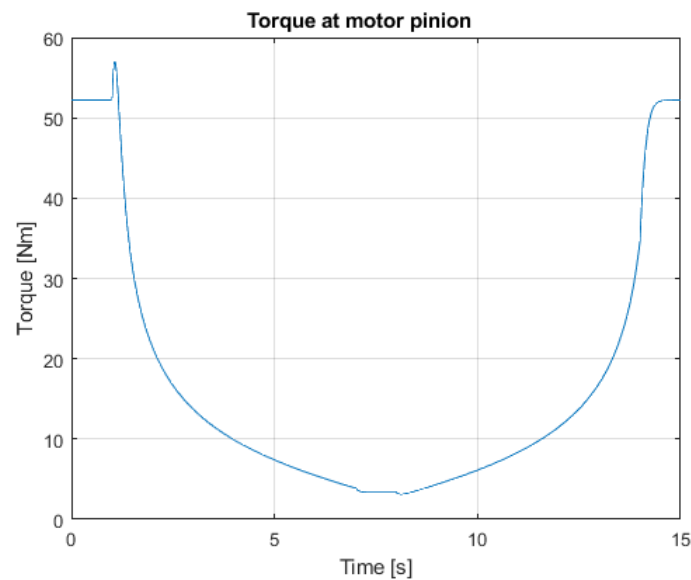


Figure 4.9: Torque at motor pinion

4.4 Motor torque

During the simulation phase another problem occurred: imposing a motion profile to the mobile block on the screw the real torque in the model can be seen only at the screw. In fact, when the simulation is running, the pinion of the motor doesn't sense the torque related to the load, but only the torque required to move the motor pinion. On the other hand, if I apply a torque directly to the motor pinion, without imposing a motion to the mobile blocks, from the motor I can feel all the force caused by the load. To solve this problem and to see if the torque at the motor pinion was correct, I have used a particular method with two simulations:

1. During the first simulation I have imposed the motion to the mobile block on the screw, and I have measured position, velocity and acceleration of the motor pinion. I have not only displayed these values, but I have also collected them using the To Workspace block of Simulink. With this block is possible to store timeseries into the workspace.

Into the model I have connected the different blocks as in the following figure.

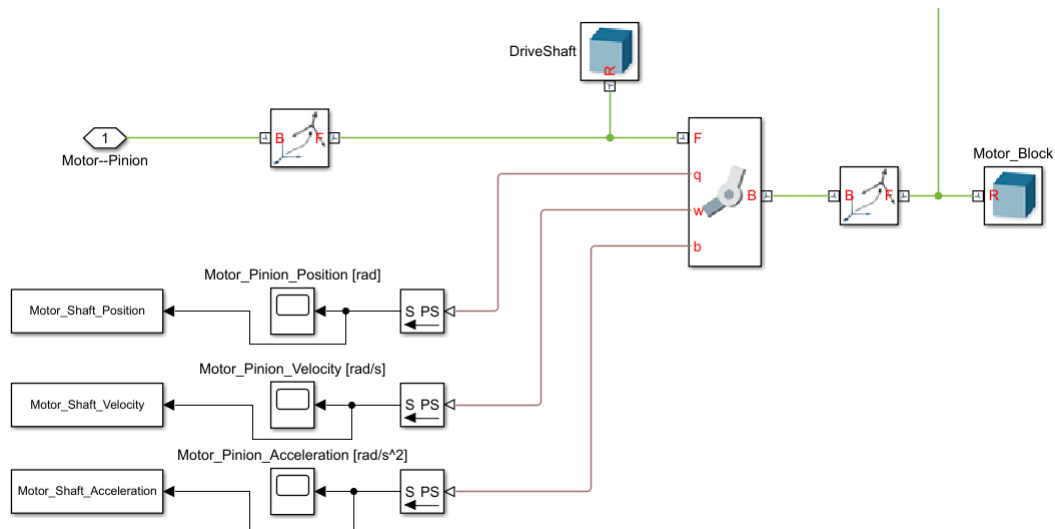


Figure 4.10: Model for data acquisition

The values of position, velocity and acceleration measured at the motor pinion are represented in the following graphs.

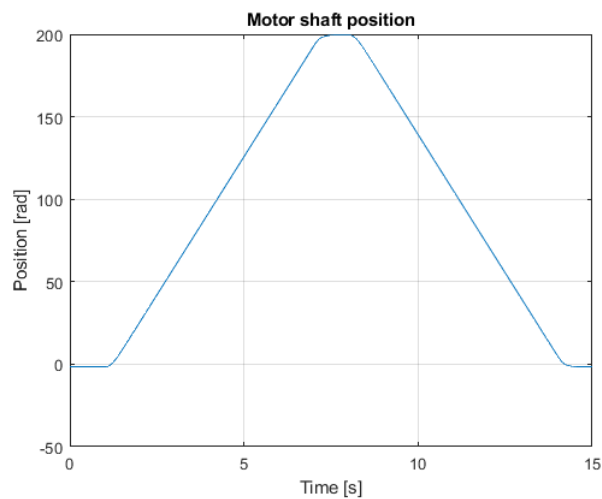


Figure 4.11: Motor shaft position

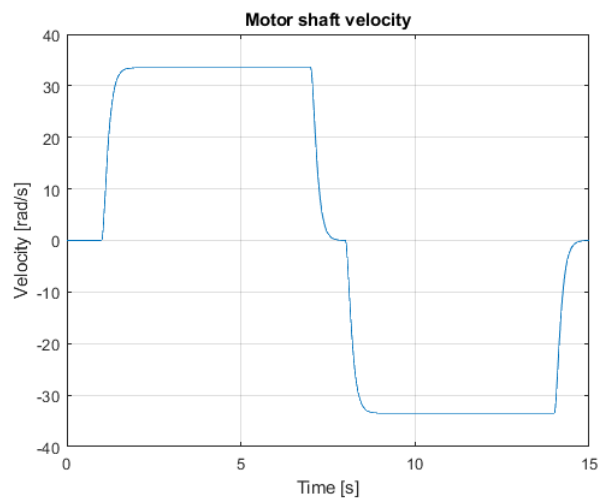


Figure 4.12: Motor shaft velocity

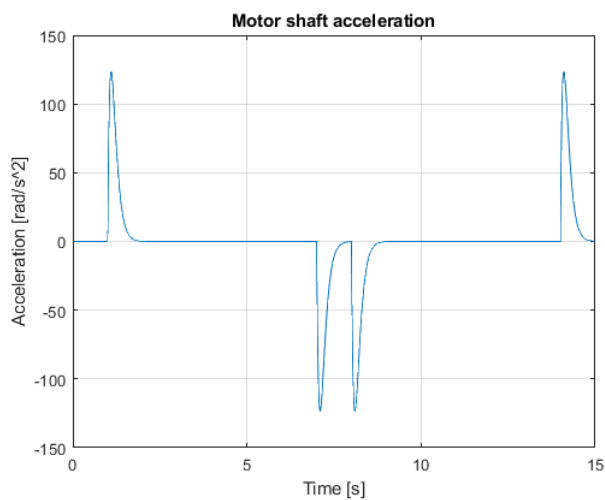


Figure 4.13: Motor shaft acceleration

2. During the second simulation I have changed the input for the system's motion. The mobile block is free, while the Revolute Joint of the motor pinion has as input the values of position, velocity and acceleration previously calculated and taken from the workspace using three From Workspace blocks. Into the Simulink-PS Converter I had to select different options to specify that I'm giving to the block all the signals, and it is not necessary to calculate derivatives: I have selected "Provide signals" in the "Filtering and derivatives" menu, and "Input and first two derivatives" in the "Provide signals" menu.

The new structure of this part of the model will be the following one.

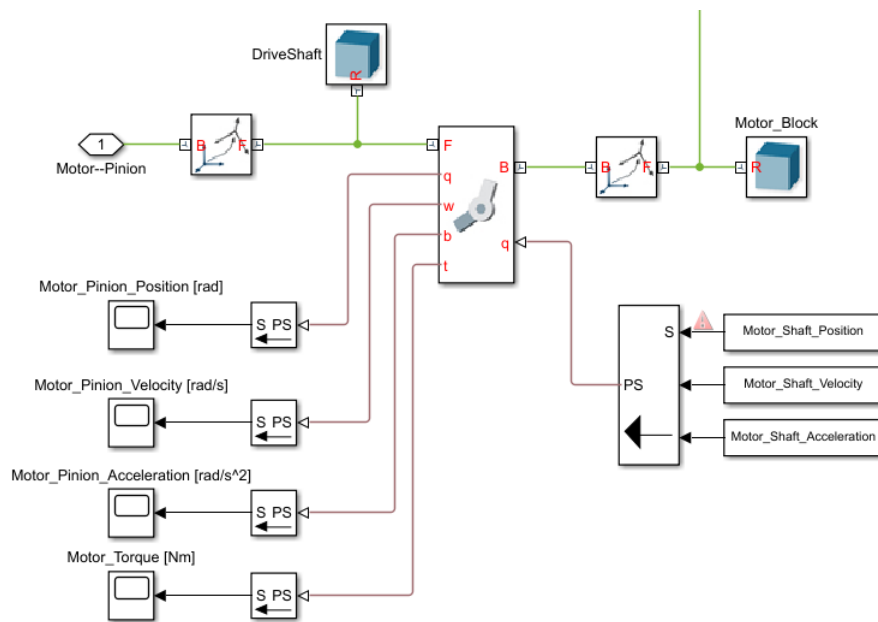


Figure 4.14: Model with motion input

Now the motor pinion can sense the correct torque because the whole model is connected to the motor without any motion input. In fact, the torque measured at the motor now will be as in the following graph.

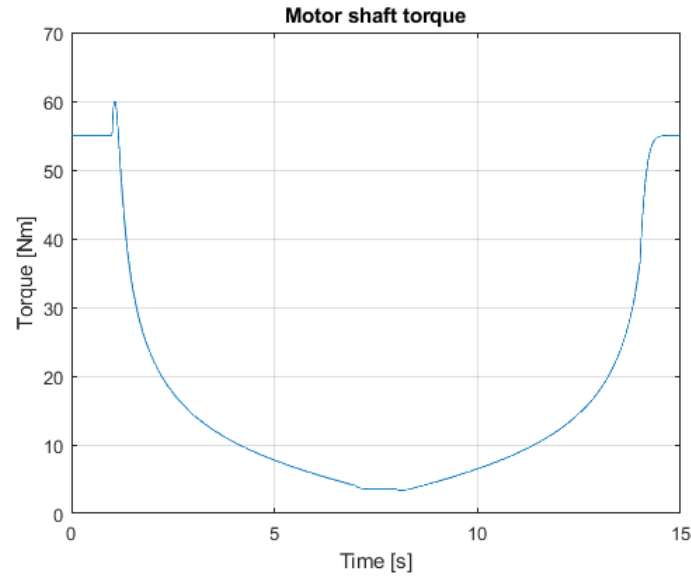


Figure 4.15: Motor shaft torque

It is possible to notice that the torque measured in this way is a little bit higher with respect to the torque measured at the mobile block on the screw using the indirect method with the formula in the Gain block (with efficiency supposed to be 0,9). The torque measured in this way is the correct torque necessary to move the complete system from the motor side.

4.5 Model with load position control

The version of the model seen until now has a control input on the position of the mobile block on the screw. Another possibility, to suit also Comau's needs related to the model, is to control the position of the load in the Z axis directly.

Usually, to control the end-effector into a model with Simscape Multibody, a 6-DOF Joint block is used. With this block the joint has three translational and three rotational degrees of freedom. For each axis is possible to specify state targets, frictions, actuation and sensing. So, with this block, the end-effector can be controlled on a specific translation or rotation but remains free for all other types of motion.

In this case I have created a connection between the ground and the load, and then I have specified the motion on the Z axis as in the previous cases, using a Signal Builder and a Simulink-PS Converter. As in the previous cases I have set a constant motion from 1 second to 7 second to go from 0 to 400 mm, and a motion from 8 second to 14 second to go from 400 to 0 mm. So, a complete motion is performed again in 6 seconds to reach the highest position and 6 seconds to return to the lowest position.

The model of these blocks is represented in the following figures, also included then into a subsystem.

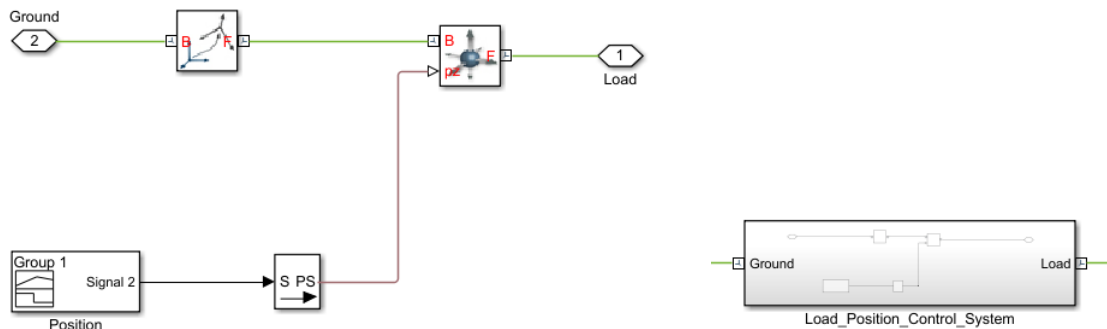


Figure 4.16: Load control model

In the following graphs is possible to see the trends of the position of the load and its velocity. Unlike the previous cases with a constant velocity on the mobile block on the screw, now the load has a constant velocity, so its position now changes in a constant way on the Z axis, while the mobile block on the screw will have a different behaviour, that basically will not be constant again.

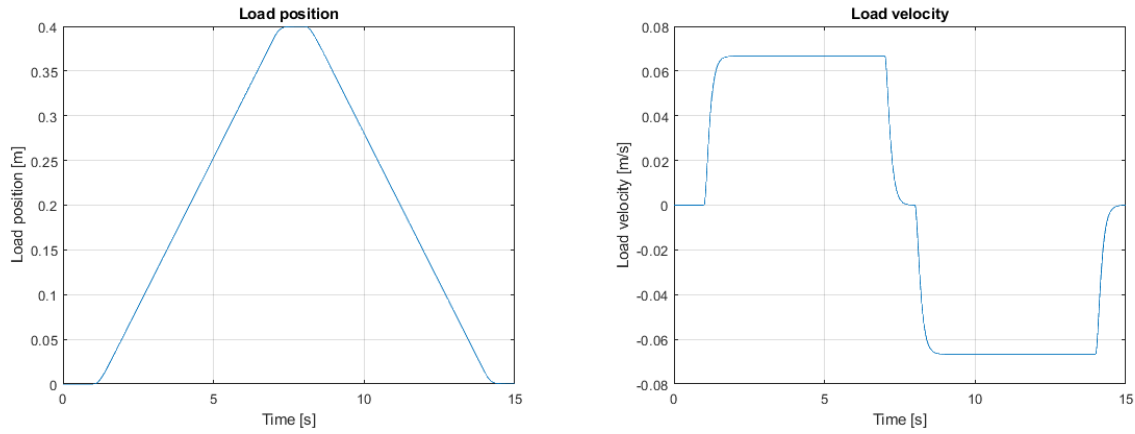


Figure 4.17: Load position and velocity

However, the controller of the motor uses the position of the load on the Z axis only as a reference signal and controls the torque that moves the mobile element on the screw to obtain the correct position of the load.

4.6 Model with changing parameters

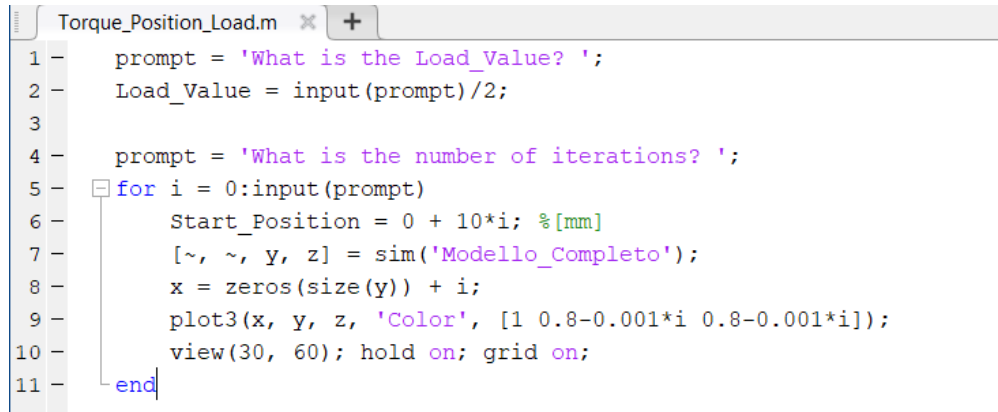
Thanks to one of the features of Simscape Multibody is possible to analyse motion at various parameter values. To do that, it is necessary to perform two principal steps: to name the value that has to change during simulations, and to write a script where all the characteristics of the simulation will be specified.

To create a simulation with changing parameters, first thing it is necessary to decide which parameters will be considered. In the model of the lifter the two most important values are the mass of the load and the start position of the mobile block on the screw, because they affect the value of the torque of the system. So, I have named the mass of the load as *Load_Value* and the position of the mobile block as *Start_Position*. The first parameter can be easily set into the Solid block that represents the load, while the second value requires another method: I have included into the model near the Prismatic Joint of the mobile block on the track a Rigid Transform, where I have set a translation along the Z axis about a value equal to *Start_Position*.

Inertia			Translation		
Type	Calculate from Geometry		Method	Standard Axis	
Based on	Mass		Axis	+Z	
Mass	Load_Value	kg	Offset	Start_Position	mm

Figure 4.18: Parameters settings

The next step is to create the MATLAB script that will control the simulation. In this case I have created a script able to run the model a specified number of times changing a parameter when the other is set. For example, if the *Load_Value* is set and is necessary to simulate more times the model with the *Start_Position* value that is increasing, the script will be the following one.



```

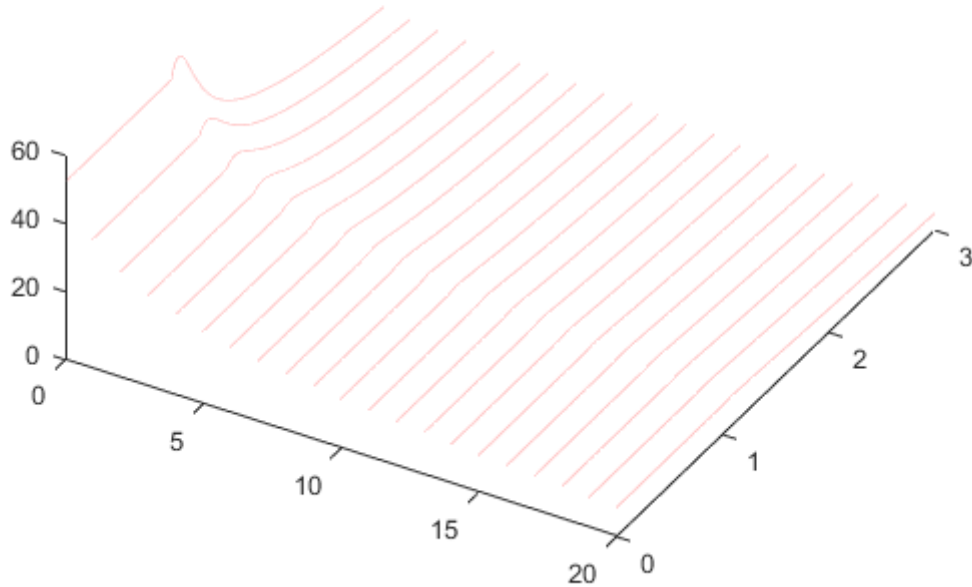
1 - prompt = 'What is the Load_Value? ';
2 - Load_Value = input(prompt)/2;
3
4 - prompt = 'What is the number of iterations? ';
5 - for i = 0:input(prompt)
6 -     Start_Position = 0 + 10*i; %[mm]
7 -     [~, ~, y, z] = sim('Modello_Completo');
8 -     x = zeros(size(y)) + i;
9 -     plot3(x, y, z, 'Color', [1 0.8-0.001*i 0.8-0.001*i]);
10 -    view(30, 60); hold on; grid on;
11 - end

```

Figure 4.19: Matlab script

In this case I have set the *Load_Value* at 300 kg in the command prompt, and then I have set 20 iterations with the *Start_Position* changing as specified ($0 + 10i$). So, this first part of the script is used to specify the number of iterations and the values of the parameters. It is possible also to select a fixed value for the *Start_Position* with a changing value of *Load_Value* inverting the position of the two variables.

The second part of the script is used to acquire values from the model and to plot them in a 3D graph. The function at line 7 is able to acquire the values that arrive from the model with output ports, saving them in two variables: *y* and *z*. So, in this case the Outport block number 1 will provide values for *y* while the Outport block number 2 will provide values for *z*. The *x* value represents the number of the iteration in the 3D graph. Finally, at line 9 is possible to see the function used to create the 3D graph: in this case the Y axis represents the time of the simulation, acquired thanks to a Clock block connected to the Outport block 1, while the Z axis represents the value of the torque on the screw. Running the script and with a simulation time of only 3 seconds, the results in this case will be plotted as in the following figure.

Figure 4.20: Torque with changing *Start_Position*

From this example it is possible to notice that with the same load, but increasing the starting position of the mobile block, the necessary torque to start the motion of the lifter always decrease. So, a result of this type can be used for example to set the maximum starting position available for the operations of the lifter to let the system work properly with the lowest possible motor torque.

Another test that I have done it's been about the relationship between the torque given at the screw and the position of the load on Z axis. With the results of this relationship is possible to select for example the positions that the load can reach with the torque that the motor is able to provide. To do that, I have created another script on MATLAB, used to give the parameters of the starting position on the screw, and the value of the load. Into the second part of the scrip I have used a function that is able to run the model and then I have plotted the values on a 2D graph. Then, with the values chosen from this graph, it is possible to see the complete behaviour of the torque during the simulation using the model of the complete system.

In the following two figures it is possible to see the script that I have created and the final relationship between torque and position, in this case starting from the initial position and with a load of 700 kg on the complete system of the lifter. The values are given through the command prompt and are necessary to start the simulation running the following part of the code.

```
Torque_PositionZ.m  X  +
1 - prompt = 'What is Start_Position value? ';
2 - Start_Position = input(prompt);           %[mm]
3
4 - prompt = 'What is the Load_Value? ';
5 - Load_Value = input(prompt)/2;           %[kg] (CaricoTot/2)
6
7 - sim('Modello_Completo');
8 - plot(Load_Position_Z, Torque);
9 - title('Load Position - Torque');
10 - ylabel('Torque [Nm]');
11 - xlabel('Load Position on Z axis');
12 - grid on;
```

Figure 4.21: MATLAB script for Load Position and Torque relationship

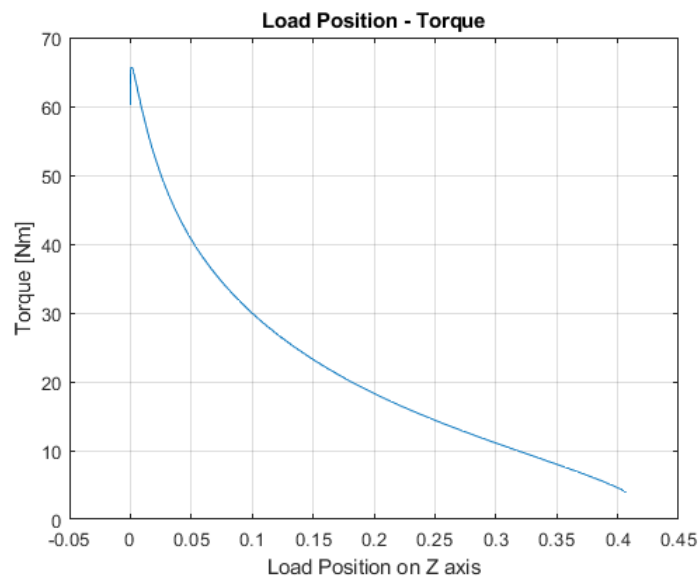


Figure 4.22: Load Position and Torque relationship

4.6.1 Stop simulation

During the simulations with changing parameters a problem occurred: increasing the starting position, without any constraint, the mobile element didn't stop at the end of the track. This condition is a problem because generally a strange behaviour appears in the simulation output data.

To avoid this behaviour I have added a system able to stop the simulation when a particular condition is true: I have compared with a Relational Operator block the value of the translation on the screw with a constant value that is the threshold value where the system reaches the higher position. The Relational Operator block, when a particular condition is true, creates an output able to activate the Stop block, that stop the simulation. After that, the program will show the final output or it will start again another simulation if there are more iterations.

This procedure can be repeated also to stop the simulation when the model reaches the lowest position.

In the next figure it is possible to see the system able to stop the simulation at a certain threshold.

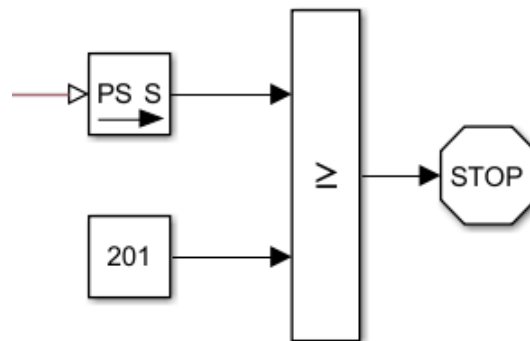


Figure 4.23: Stop simulation model

4.7 Results with more accurate inputs

Until now I have used as input for the lifter's motion the graph of the position of the mobile block on the screw that can be seen in Figure 3.68. To obtain good results I have set the input filtering at 0,1 (10 Hz): in this way the output velocities and accelerations were more linear, without strange peaks, and this is more similar to the real behaviour of a system.

To obtain results closer to the real case, first thing I have set the input filtering to 0,01 (100 Hz), and using the same position input to move the system I have verified that this type of input cannot be similar to a real input. In fact, the output torque that I have calculated with a simulation with these parameters is represented in the following figure.

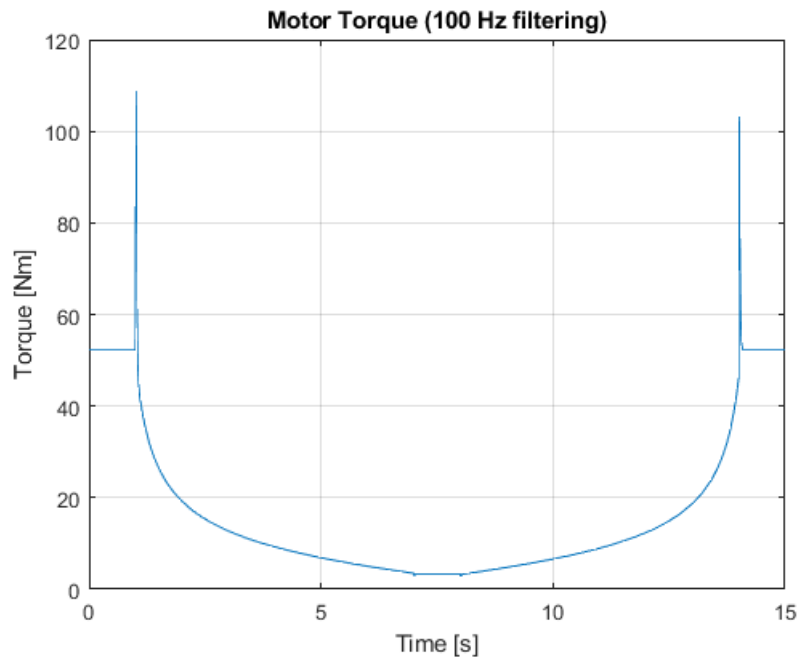


Figure 4.24: Motor Torque with input filtering of 100 Hz

As it is possible to see when the position of the mobile element starts to change, there is a very high acceleration that requires a high value of torque: this is not possible in a real case. The following graphs represent the velocity and acceleration profiles with the position input of Figure 3.68 calculated at 100 Hz.

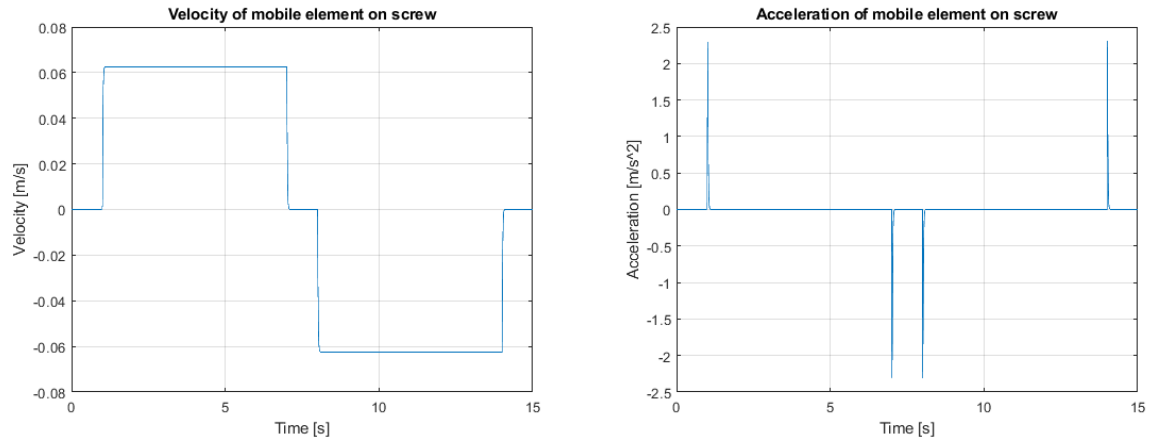


Figure 4.25: Velocity and acceleration with input filtering of 100 Hz

To avoid this situation and to give to the system an input more similar to a real one, I have generated two signals that are able to replicate the same motion of the mobile element, but giving to the system a different velocity profile or a different acceleration profile. The two graphs with velocity and acceleration values are represented in the following two figures.

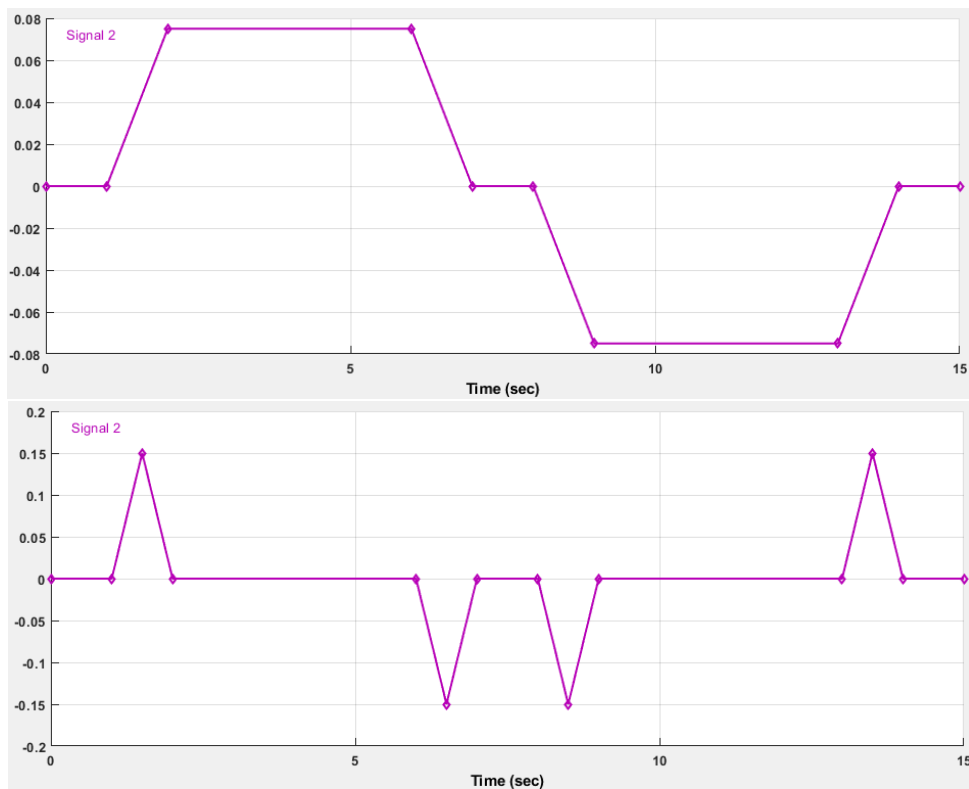


Figure 4.26: Velocity and acceleration new input profiles

Now the profiles of velocity and acceleration are not ideal yet, but it is possible to see the transient between the minimum and maximum values of velocity and acceleration during the motion. So, the velocity and the acceleration will not be represented by rectangles or peaks, but for example respectively by a trapezoid and a triangle, with a less marked behaviour.

To do that, the problem was that the Prismatic Joint that controls the motion of the mobile block wants as input a position value to reach at a specific time. So, to avoid this problem, I have integrated one time the velocity profile and two times the acceleration profile with the Integrator Block of Simulink, as represented in the following figure.

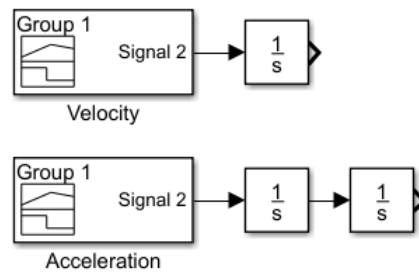


Figure 4.27: Inputs with Integrator Blocks

The values obtained for velocity, acceleration and motor torque in the system also with an input filtering of 100 Hz now certainly are more realistic, and they are represented in the following figures.

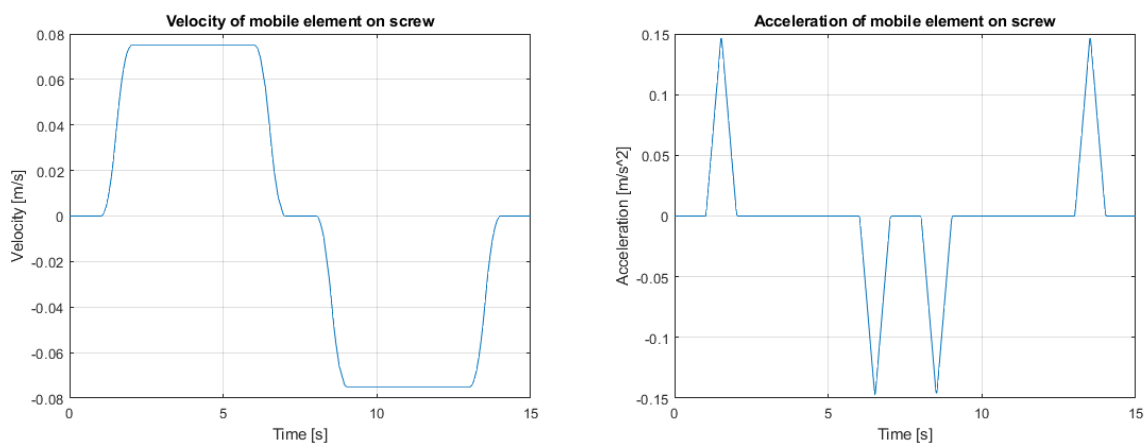


Figure 4.28: Velocity and acceleration with new input filtering (100 Hz)

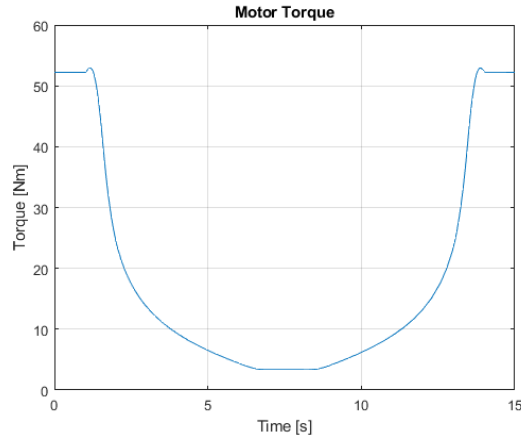


Figure 4.29: Motor Torque with new input filtering (100 Hz)

To create different profiles it is possible to modify the input graphs: for example, to obtain less marked accelerations it is possible to modify again the acceleration profile. In this case, to obtain a torque that is again more linear with respect to the previous one, it could be a good choice to have a minor acceleration when the lifter is in the lowest positions. To do that, I have created a slightly different acceleration profile as represented in the following figure.

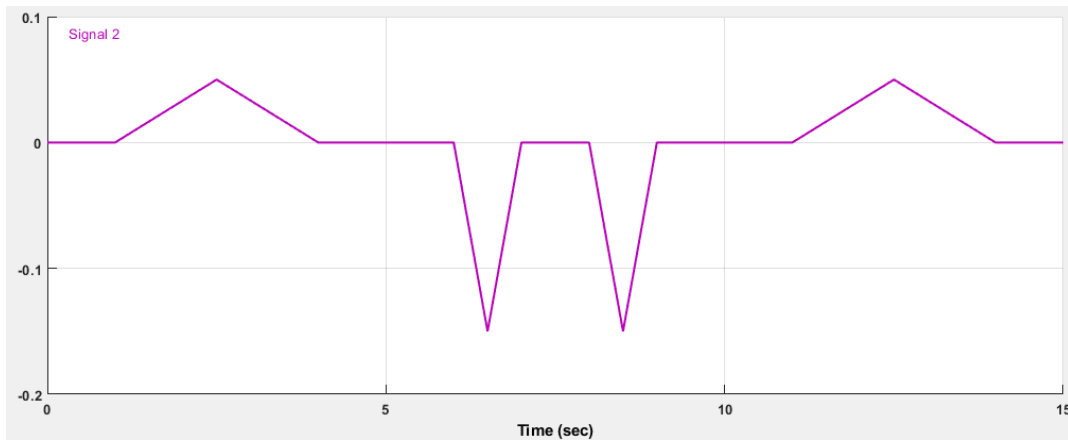


Figure 4.30: Different acceleration profile

Running the model with this input the velocity and the torque of the motor that it is possible to obtain are again more similar to a real case, with no peaks and a linear behaviour, as shown in the following figures.

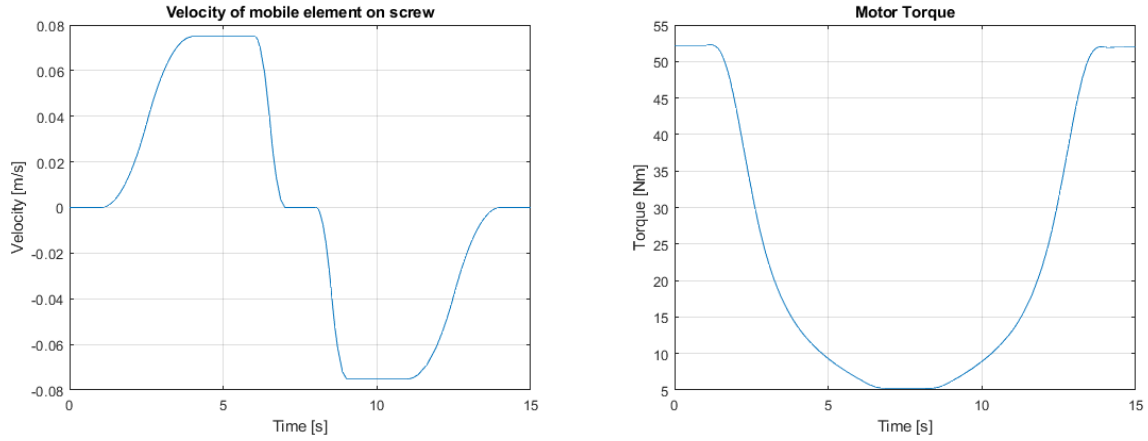


Figure 4.31: Velocity and Motor Torque with different acceleration profile (100 Hz)

With this configuration for the motion the initial peak of the Motor Torque is only about 0,07 Nm.

4.8 Validation of the model

The final step to validate the model was to compare the results of Simscape Multibody with the real behaviour of the system. To do that different data were available: the most important one, connected to the torque value at the motor pinion, was the quadrature current.

During the fatigue test the Lifter 500 had a total load of 400 Kg: the peak value of the quadrature current corresponds with a torque value of 33-34 Nm, as it is possible to see in the next figure. Running the model on Simscape Multibody with this load value it is possible to obtain the same torque value, and also the behaviour during the motion is the same.

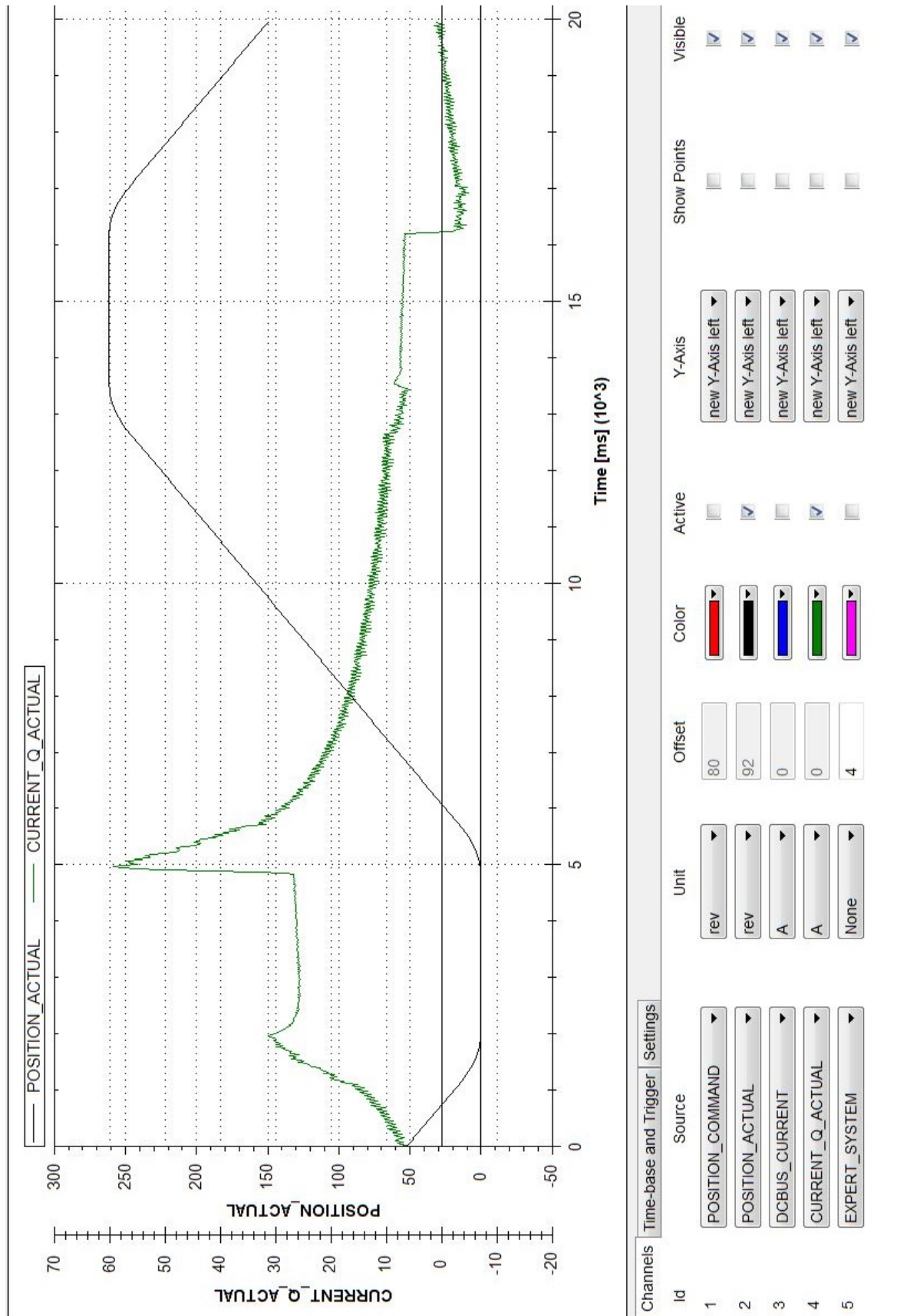


Figure 4.32: Quadrature current

Chapter 5

Conclusions

5.1 Simplified model of the lifter

Simscape Multibody is able to represent a real system also using a really simple model from the graphic point of view. During the first phase of this project I have created a complex model that represents the real lifter both from the mechanical point of view and from the graphic point of view. The creation of this model requires more time, that in this case is about a month.

The real potentiality of this tool of MATLAB is to provide results close to the real case with a model that from the graphic point of view is really simple, but similar to the real system from the working point of view. With this method is possible to create a model in a really short time using simple blocks and shapes. So, to demonstrate this potentiality of Simscape Multibody, I have created a simple model with the same working principles of the lifter showed in the previous chapters, but that is able to provide the same results of the simulation.

The model from the graphic point of view appears like in the following figure.

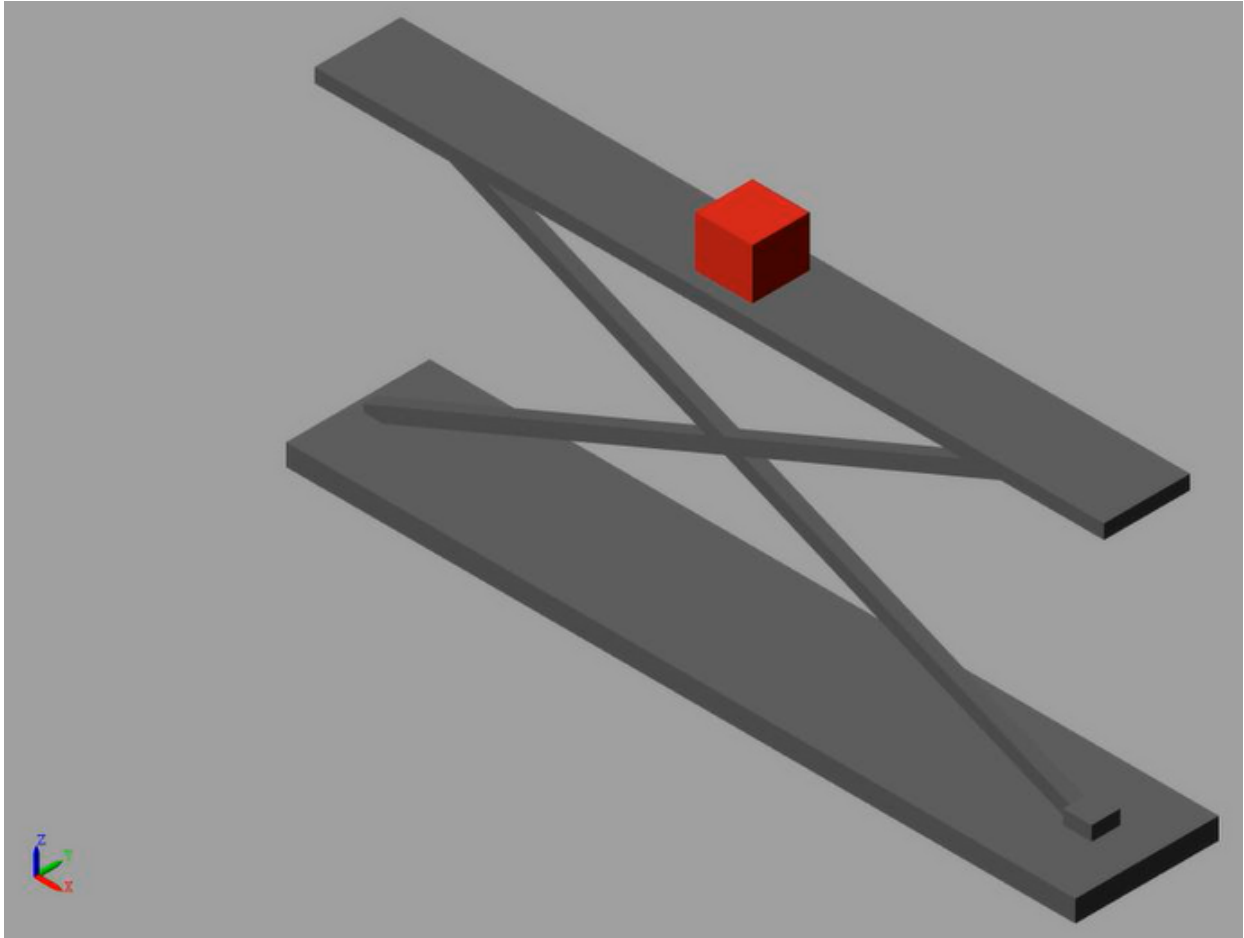


Figure 5.1: Simplified lifter

As it is possible to see, from the graphic point of view the model is really simple: basically, it is composed only by two plates, the base and the upper one, and by the two axes that allow the motion of the system. Then, on the top of the system, I have added a simple block that represents the load on the lifter. At one side of each axis I have added a little solid (a cube) to simplify the creation of the joints where there are two types of motion: rotation with respect to the axis and translation with respect to the plate.

The model on Simscape Multibody is represented in the following figure.

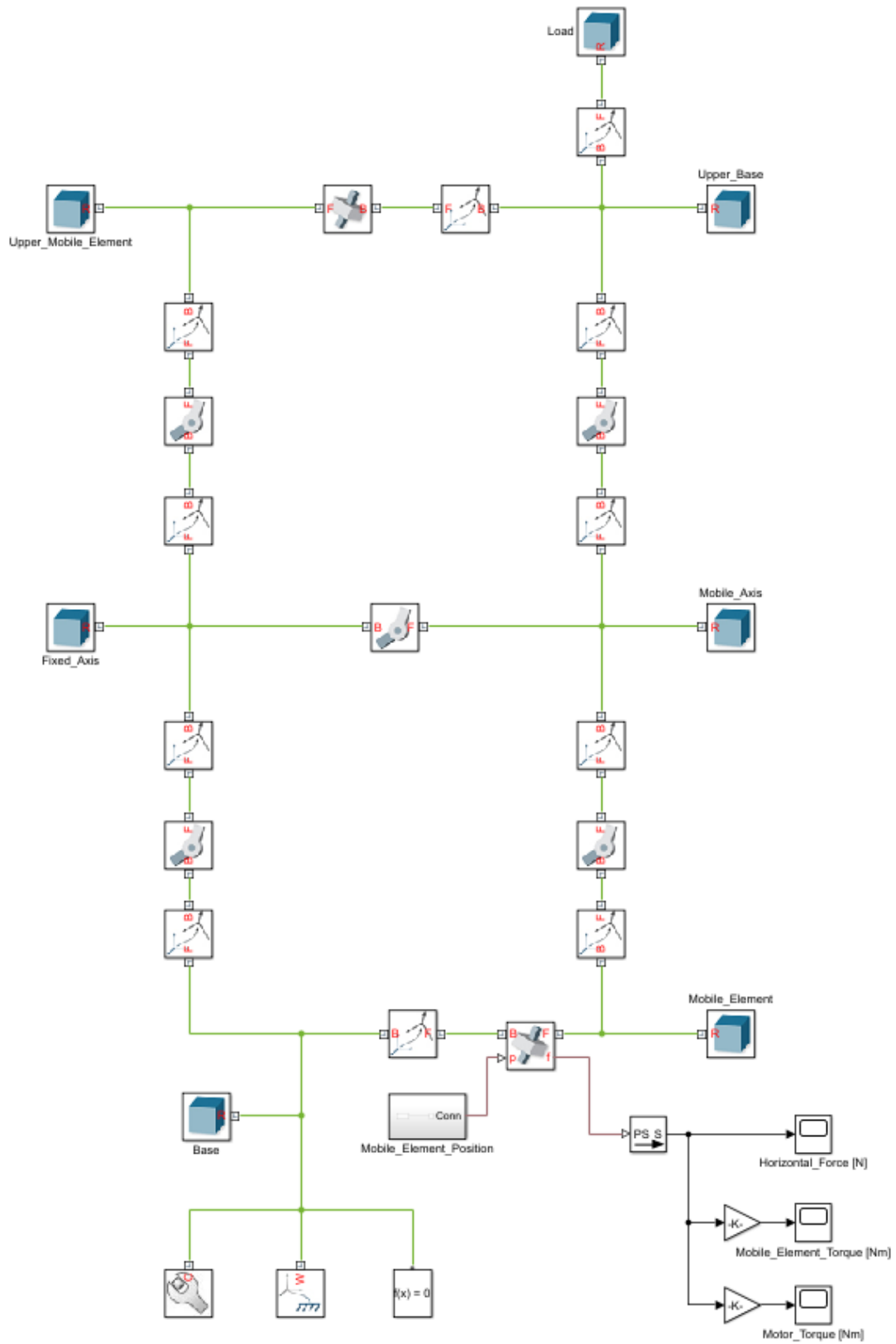


Figure 5.2: Simplified model of the lifter

To see how much quickly a simple but correctly working model can be created, I have set up only the mass value of each solid, without setting up other values like for example junction's frictions. Therefore, I've been able to create this model using only two hours, but the simulation results are the same of the more complex model, as represented into the following graphs.

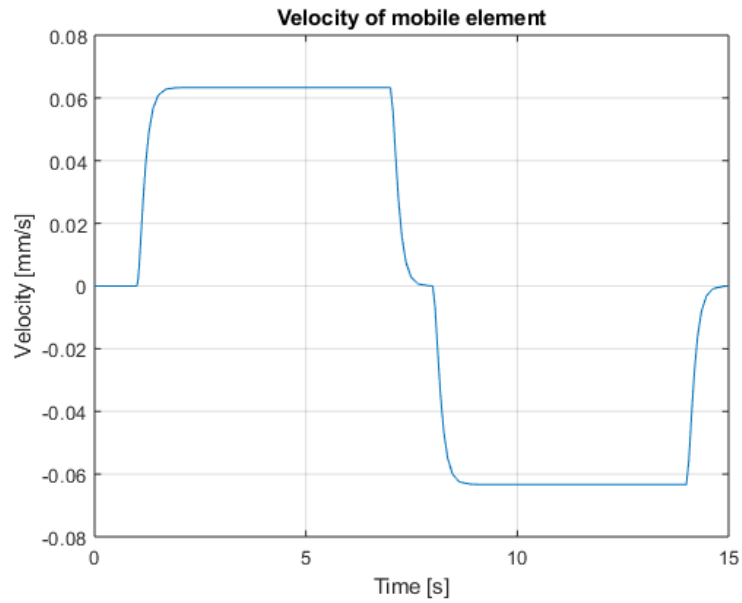


Figure 5.3: Velocity of mobile element

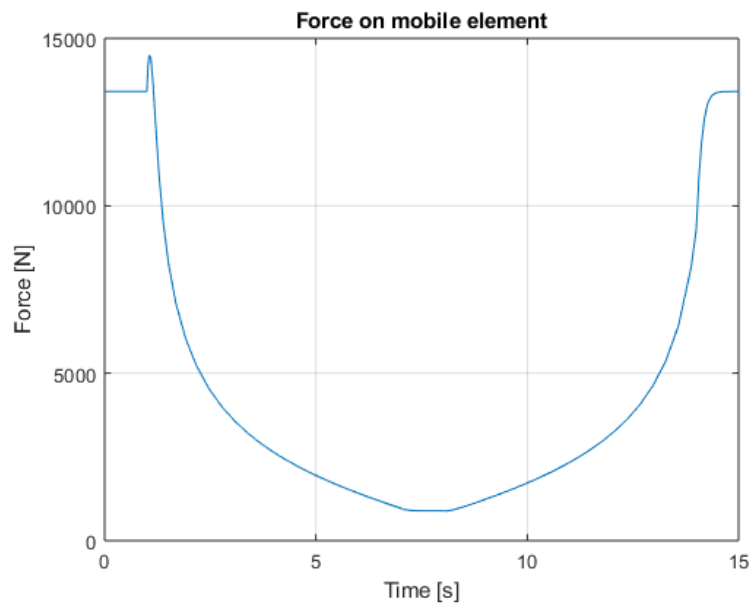


Figure 5.4: Force on mobile element

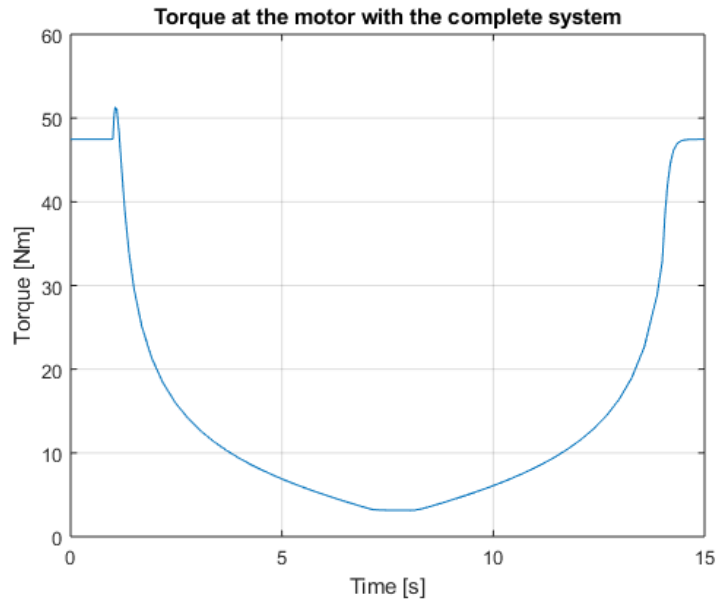


Figure 5.5: Torque at the motor with the complete system

So, after this test it is possible to say that Simscape Multibody can be really useful during the development phase of a project, since it allows to obtain detailed results of dynamic simulations of a model quickly, allowing also to study different alternatives in a short time just changing some parameters into the model.

5.2 Cost-Benefit Analysis

The first step during the cost-benefit analysis it's been to identify all the different phases of the development process of a new product and all the people involved in each one. This study has been done about the current development process and about a possible new development process that includes Simscape Multibody.

The current development process is composed by different phases connected each other:

1. Product requirements/specifications.

This is the first step for the creation of a new product. During this phase the idea of a new product is created, to satisfy specific requirements/specifications given by the market, or by a specific company need. The project manager starts to organize the development process.

2. Concept 3D.

During this phase a first solution is created to satisfy the new product needs. Usually the concept is created as a simple CAD file, by a mechanical designer.

3. Mechanical design.

Now the creation of the new product starts to be more detailed: the mechanical designer creates a complete assembly of the product.

4. Static calculations and dimensioning.

This phase is connected to the previous one. The mechanical engineer is responsible to verify if the first concept is feasible, with static calculations, and works with the mechanical designer to solve different problems related to the structure of the new product. Currently, the only available tool to perform this study is Excel.

5. Hardware design.

When the main structure of the product it's been created, the hardware designer starts to work with the Mechanical designer, to insert into the product all the hardware elements needed.

6. Control design.

Now also the software designer starts to work with the people involved in the previous phases. During this phase the control system of the new product is designed.

7. Purchase.

During this phase the project manager must collect all the single elements required to assemble the new product (BOM) starting to buy components by suppliers.

8. Manufacturing.

During this phase the mechanical and electrical systems are assembled.

9. Commissioning.

After the assembly work the whole system is integrated with the control system and the new product is able to work.

10. Test.

This is a very important phase: after the static tests done during the development phase, now the first dynamic tests are performed. All the components are tested: mechanical, hardware and software. If problems are observed, it is necessary to return to the product development phases, to find the real reason of the problem and to solve it. Then the complete product will be tested again.

11. Final tests and release to market.

If the product it's been redesigned, a final test will be performed to verify again that all components work properly. After this phase, in which also final little changes could be done, the product is ready to be released on market.

The problems of the current development process are basically two:

- During the design phases there is not the possibility to create a dynamic model of the new product, and only a static study is performed. So, the real dynamic behaviour of the product is verified only during tests one the complete assembled product.
- The second problem is related to the first one. When a problem occurs during dynamic tests, it is necessary to return to the development phases to redesign the product. This process will take a large amount of time, because it is necessary to modify different elements and to verify if they can be assembled yet in the complete system, and it is also necessary to buy again new components by suppliers and to wait for their arrival time.

The development process that can be obtained by introducing a new tool as Simscape Multibody can be different:

1. Product requirements/specifications.

This is the first step for the creation of a new product. During this phase the idea of a new product is created, to satisfy specific requirements/specifications given by the market, or by a specific company need. The project manager starts to organize the development process.

2. Concept 3D and mechanical design with Simscape Multibody.

This is the most different phase with respect to the previous process. Simscape Multibody, cooperating also with CAD tools, can be used to create a first simple concept and then a more detailed model of the new product. From this model it is possible to obtain all static and dynamic simulation data, testing the behaviour of the created system in different situations. For example it is possible to look at the behaviour of the system when a specific parameter is varying, allowing the designer to choose the best value: this can be a really helpful feature during mechanical design phases.

3. Hardware design.

When the main structure of the product it's been created, the hardware designer starts to work with the Mechanical designer, to insert into the product all the hardware elements needed.

4. Control design.

Now also the software designer starts to work with the people involved in the previous phases. During this phase the control system of the new product is designed.

5. Purchase.

During this phase the project manager must collect all the single elements required to assemble the new product (BOM) starting to buy components by suppliers.

6. Manufacturing.

During this phase the mechanical and electrical systems are assembled.

7. Commissioning.

After the assembly work the whole system is integrated with the control system and the new product is able to work.

8. Test.

All components of the system are tested, to verify the behaviour already seen using Simscape Multibody during the development phase. During this phase, with this type of development process, will be more difficult to find errors, avoiding a possible complex redesign phase. So, it is possible to go directly to the next steps without wasting time.

9. Final adjustments and release to market.

During this phase some final adjustments could be done and then the product will be released to market.

In the following two tables I have represented all the different phases of the current development process and of the possible new development process.

For each phase I have added different information:

- Name of the phase.
- People involved during each phase.
- Input and output information during the development process.
- Software tool used during each phase.
- Indication about training need.
- Hours estimate.
- Costs estimate.

Phase	People Involved	Input/Output	SW Tool Needed	Training	Hours	Cost
Product Requirements and Specifications	Project Manager	In: product requirements Out: product specifications		No	40	\$1000
Concept 2D/3D	Mechanical Designer	Out: project idea to satisfy requirements/specifications	SolidWorks/ CATIA/AutoCAD	No	100	\$1000
Preliminary static calculations	Mechanical Designer	Out: first feasibility study	Excel		50	\$500
Mechanical Design	Mechanical Designer	Out: mechanical complete assembly of the product	SolidWorks/CATIA	No	1000	\$10000
Static Calculations and Dimensioning	Mechanical Engineer	Out: accurate study on requirements of the mechanical assembly from the static point of view	Excel	No	40	\$1000
HW Design	HW Designer	Out: design of the HW required by the product	ePLAN	No	100	\$10000
Control Design	SW Designer	Out: design of the control system of the product	Different tool for each application	No	100	\$10000
Purchase	Project Manager	In: list of elements that compose the complete system (BOM) Out: single parts of the system ready to be assembled		No		
Manufacturing	Mechanical/Electrical Technician	Out: mechanical system assembled		No	100	\$10000
Commissioning	SW Designer	Out: whole system assembled and ready to work	Different tool for each application	No	40	\$1000
Test	SW Designer Mechanical Designer HW Designer Technician	Out: accurate dynamic tests on each part of the system (if errors are found it is necessary to go back to the related design phase)			10000	\$10000000
Redesign (correct design based on test feedback)		In: new informations acquired by tests Out: new design of the product (after all development phases)			10000	\$10000000
Project Management	Project Manager	Out: coordination of the project development	Excel	No	100	\$10000
Release to Market	Sales/Proposal/ Marketing	Out: complete system able to work properly			100	\$1000
					Total hours	\$100000
					1775	\$100000

Figure 5.6: Analysis of the current development process

Phase	People Involved	Input/Output	SW Tool Needed	Training	Hours	Cost
Product Requirements and Specifications	Project Manager	<u>In:</u> product requirements <u>Out:</u> product specifications		No	10	1000
Concept 3D Mechanical Design with Simscape Multibody (Static and Dynamic Tests)	Mechatronic Engineer	<u>Out:</u> - project idea to satisfy requirements/specifications from both static and dynamic point of view - mechanical complete assembly and accurate study of the product from both static and dynamic point of view	Simscape Multibody (SolidWorks/CATIA)	Yes	100	10000
HW Design	HW Designer	<u>Out:</u> design of the HW required by the product	ePLAN	No	100	10000
Control Design	SW Designer	<u>Out:</u> design of the control system of the product	Different tool for each application	No	100	10000
Purchase	Project Manager	<u>In:</u> list of elements that compose the complete system <u>Out:</u> single parts of the system ready to be assembled		No		
Manufacturing	Mechanical/Electrical Technician	<u>Out:</u> mechanical system assembled		No	100	10000
Commissioning	SW Designer	<u>Out:</u> whole system assembled and ready to work	Different tool for each application	No	100	10000
Test	SW Designer Mechatronic Engineer HW Designer Technician	<u>Out:</u> accurate dynamic tests on each part of the system (if errors are found it is necessary to go back to the related design phase)			10000	100000000
Redesign (correct design based on test feedback)		<u>In:</u> new informations acquired by tests <u>Out:</u> new design of the product (after all development phases)			10000	100000000
Project Management	Project Manager	<u>Out:</u> coordination of the project development	Excel	No	100	10000
Release to Market	Sales/Proposal/Marketing	<u>Out:</u> complete system able to work properly			100	10000
					Total hours	Total cost
					10000	100000000

Figure 5.7: Analysis of the new possible development process

As it is possible to see, Simscape Multibody can help to reduce the amount of time needed to develop a new product up to 20 %, allowing to study with more detail, but using less time, the new system.

Another important element is the possibility to study the dynamic behaviour of the system during development phases. This possibility will affect also the test and redesign phases. In particular, the number of tests will remain the same, to check if all single elements of the system work properly: the number of tests is represented by Y variable. Then I have inserted other two variables, X and Z: they represent the number of malfunctions found during dynamic tests during the current development process and during the new one, respectively. Since the current process doesn't let to study the dynamic behaviour of the system during development phases, the value of X is greater than Z. This will affect the total amount of hours needed to create the new product, increasing the total cost of the development process.

A realistic example can be the choice of the motor during the development process of the Lifter 500. Looking at the data obtained by the static dimensioning with Excel, the maximum torque needed by the system was 48 Nm, as it is possible to see in the next figure.

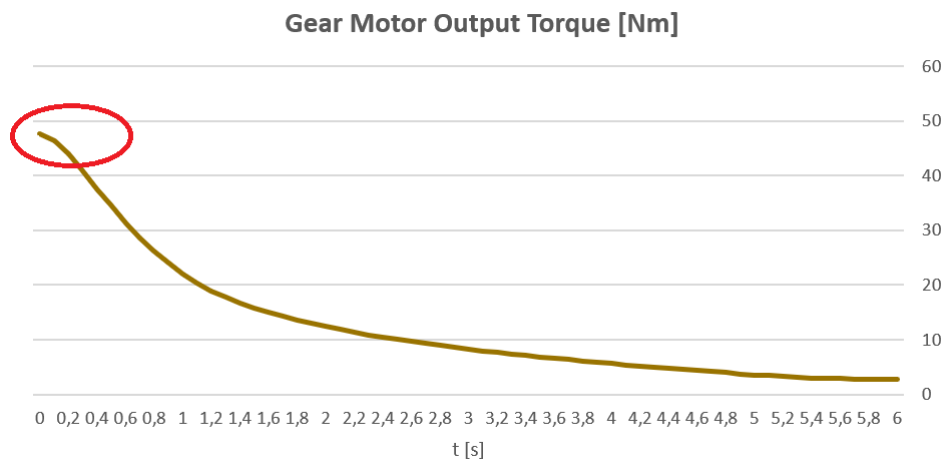


Figure 5.8: Torque behaviour obtained with Excel

During the first test, Comau has verified that the motor able to provide this maximum torque was not able to move the system. So, after new calculations and tests, Comau has changed the motor of the Lifter 500, with new costs and new time wasted due to the arrival time of the new motor and to the time needed to assemble and test again the whole system.

With Simscape Multibody this situation could have been avoided in two ways:

1. With the dynamic simulation it could have been possible to see that when the system starts its motion there is a peak of torque (more than 48 Nm) due to the starting acceleration. This behaviour can be seen in the next figure, obtained with a Simscape Multibody simulation.

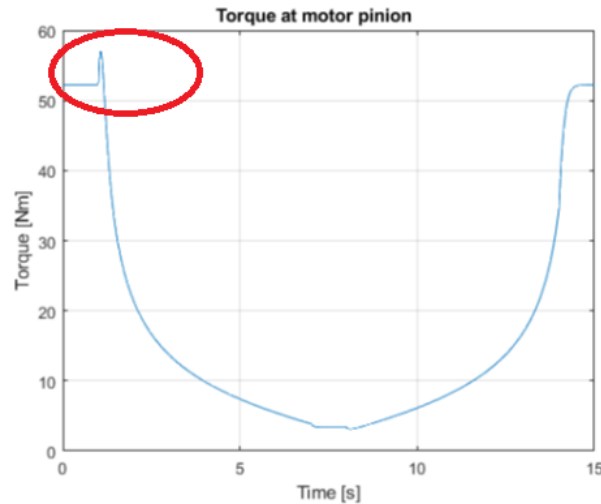


Figure 5.9: Torque behaviour obtained with Simscape Multibody

2. With a more detailed simulation it could have been possible to see that with a lower acceleration the torque peak also is lower, and so a motor that is able to provide a lower torque to the system could be suitable, and it is not necessary to change the motor. This variation in the dynamic behaviour of the system can be seen in the next figure, obtained with Simscape Multibody changing the acceleration input.

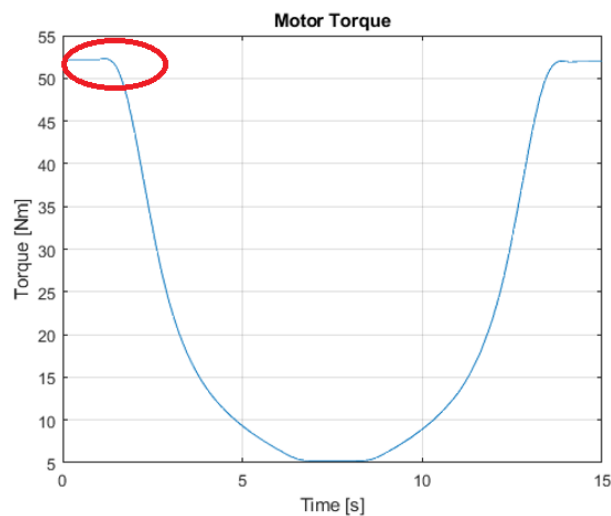


Figure 5.10: Different torque behaviour obtained with Simscape Multibody

For what concern the investment that is necessary for this project I have considered three principal elements:

- The cost of the MATLAB licence
- The cost of training related to the new tool
- The cost of a specialized resource able to work with this tool

These costs are represented in the following table, in wich it is possible to see also the total benefit on the development process.

The first case, base line, represents a comparison between the current process and the new process with MATLAB without considering errors and redesign phases, and the total benefit can be up to 20 %.

The second case represents the Lifter 500 process: here the comparison is between the current process with one error and so one redesign phase (due to the error of the motor) and the new process during which this error and also the related redesign phase can be avoided. In this case the benefit can be up to 25 %.

Finally, I have created two other hypotheses with a different number of errors:

- The hypothesis number one is a comparison between the current process with two errors and so two redesign phases with the new process with MATLAB where one of these two errors is avoided. In this scenario the total benefit can be up to 30 %.
- The hypothesis number two represents another scenario where the current process with three errors is compared with the new process with only one error. In this case the total benefit can be higher again, up to 40 %.

These values are related to a single project, so considering the possibility to start more processes in parallel, the net investment project benefit can be multiplied by the number of processes, obtaining a higher total value of benefit.

Then, every year these percentages can grow thanks to the increase of the abilities pf the operator that will use this tool.

One of the most important things is that after the cost-benefit analysis it is possible to see that the total investment for Comau can be paid back already during the second year following the adoption of the new tool.

	Licence	€ 1.000,00	
	Investment for training	€ 1.000,00	
	Specialized resource	€ 1.000,00	
	Total investment	€ 3.000,00	
Base Line	Current product development cost	€ 10.000,00	Only design costs included, without considering errors and redesign phases
	Product development cost with MATLAB	€ 25.000,00	
	Benefit on development process (%)	30%	
	Benefit on development process (€)	€ 7.500,00	
	Net investment project benefit	€ 4.500,00	
Lifter 500 case	Current product development cost	€ 100.000,00	Full project costs included, considering also errors and redesign phases: 1 error in the current development process with respect to 0 errors in the new one
	Product development cost with MATLAB	€ 10.000,00	
	Benefit on development process (%)	75%	
	Benefit on development process (€)	€ 75.000,00	
	Net investment project benefit	€ 15.000,00	
Hypothesis 1	Current product development cost	€ 110.000,00	Full project costs included, considering also errors and redesign phases: 2 errors in the current development process with respect to 1 error in the new one
	Product development cost with MATLAB	€ 10.000,00	
	Benefit on development process (%)	30%	
	Benefit on development process (€)	€ 33.000,00	
	Net investment project benefit	€ 23.000,00	
Hypothesis 2	Current product development cost	€ 110.000,00	Full project costs included, considering also errors and redesign phases: 3 errors in the current development process with respect to 1 error in the new one
	Product development cost with MATLAB	€ 10.000,00	
	Benefit on development process (%)	30%	
	Benefit on development process (€)	€ 33.000,00	
	Net investment project benefit	€ 23.000,00	

Figure 5.11: Final cost-benefit analysis

5.3 Closing remarks and next steps

So, in conclusion, Simscape Multibody is able to improve the current development process introducing:

- Possibility to study in detail different concepts of new products in a short time.
- Realistic and detailed dynamic simulations.
- Time to market reduction.
- Overall project cost reduction.

Also considering that the investments can be paid back already during the second year following the introduction of the new tool in the development process, it is possible to say that the implementation of Simscape Multibody can be to Comau's advantage.

The next steps for this project will be related to the improvement of the model adding:

- Electrical drive model.
- Electric motor model, that can be obtained by the company that produces it.
- Control system model.

In this way, the model will reproduce exactly the complete real mechatronic system.

Bibliography and sitography

- Karnopp, Margolis, Rosemberg, *System dynamics: modeling, simulation, and control of mechatronic systems*, Wiley, 2012
- Campbell, Brown, *Benefit-Cost Analysis, financial and economic appraisal using spreadsheets*, Cambridge University Press, 2003
- MATLAB Documentation
- *www.mathworks.com*, Community File Exchange