POLITECNICO DI TORINO

Master's Degree in Mechatronics Engineering

Master's Degree Thesis

Digital twin control system model development applied to 5-axis gantry stacker

Supervisors

Prof. Andrea TONOLI

Mr. Alessio CALLEGARI

Candidate

Paolo ARNAUDO

July 2023

Abstract

Flat glass represents one of the key materials employed in important market slices, like constructions and automotive, due to its recyclability and its intrinsic characteristics.

The production process plays an important role in the efficiency and the fulfillment of the market demand. Bottero S.p.A. is one of the leading companies in the glass technology and it is the principal partner of this research.

An upcoming problem that will be faced is the combination of shorter time-to-market required and delays in the supply chain which leads to limited design and installation phases. To overcome this challenge, the solution is the development of a digital twin for virtual commissioning.

The digital twin represents an emerging technology for a digital and intelligent design. This model is a virtual copy of the complete system capable of monitoring, simulating processes, predicting possible outcomes and optimizing the re-think of the automation unit at all the phases of the design.

The machinery under investigation is a 5-axis gantry stacker whose main purpose is the loading and unloading of glass sheets from the line conveyor to the storage racks. Mechanically, it is composed by a gantry portal on which a bridge can translate on wheels along the line direction. A carriage then slides perpendicularly on the bridge guides. A lifting piston governed by a chain mechanism then perform the movement in height. In the end, two motors define the rotation along the z axis and the tilting movement along the x axis.

From an electrical and automation point of view, the system is composed principally by two control units: a CPU (Siemens ET200SP) that performs the control algorithms on the IO coming from the operator (console and HMI), the processing line and the pneumatic commands for glass suction and blowing; the second component is a Motion Control Unit (Siemens SIMOTION D425-2) which governs the axis motion, synchronism and trajectory definition.

Once programmed the software codes, the interconnections between the subsystems are required and the SIMIT simulation software is the responsible for the different signals travelling in the machinery. With this software, it is possible to emulate the several instances of control units and investigate the response of the automation system to the several Input tests.

The final element of the digital twin is the definition of a proper multi-body model. From the CAD design, two simple MatLab models were developed investigating several aspects of the kinematics and dynamics of the machine. With the help of the Robotic Matlab Toolbox, it is possible to analyse the definition of the pose of the robot with an interactive GUI and, once defined the working environment, a

collision check can be performed with the several robot configurations considered during the trajectory definition.

With the SimScape model, instead, a trajectory planning analysis is developed considering the exploitation of the drive parameters, cycle time and dynamic response of the system to different solutions such as polynomial cubic and quintic function, harmonic trajectory and piecewise trajectories (trapezoidal and double-S).

This project represents the beginning of the virtual commissioning process that will be further developed with the interconnection with the native Siemens multi-body designer, capable of interacting with the IO and the automation. This project has the power to show the complete design process under a newer, brighter light.

Summary

This project has been developed with the contribution of the Technical Department of the flat glass branch of Bottero S.p.A., a leader company in the glass technology.

The core of this business unit is the design and installation of glass production lines whose final products varies from large sheets for construction purposes down to small displays for technological devices.

Float glass lines are the industry plants that transform the silica sand into glass sheet through a continuous stream of glass that floats over a tin bath due to the lower density. From there, this continuous rib of glass is moved with conveyors and cut. The handling and the storage of these glass lites is performed by anthropomorphic robots or portal gantries, depending on the weight to be transported. The final package is represented by a storage rack where the plates are piled in different packs with univocal production codes each one separated by cardboard spacer.

The other types of production line are categorised in off-line post treatment of the glass: these comprehend the coaterization and lamination of the glass. In this case the production sites are fed by the racks of pre-produced glass sheets. Aside from the processing unit of the glass, the production time of the system is highly influenced by the cycle time of the gantries that load the required sheets.

For the following reasons, a proper and optimized design of this machinery is the goal. Unfortunately, the combination of shorter time-to-market required and delays in the supply chain decreases the possibility of a proper project and installation phase. To overcome this challenge, the solution is the development of a digital twin for virtual commissioning.

With respect to classical simulation methods that have limited capabilities in evaluating system performance, the digital twin concept represents the breakthrough of limitations on the modeling and engineering analysis. A digital twin is an integrated multi-physics and multi-scale simulation of a system that can reproduce all aspects of the desired partner: starting from the mechanical to the software passing through the electrical emulation. Digital twin model technology is recognized as one of the technological milestones for achieving the goals of Smart Manufacturing and Industry 4.0. The most direct value of adopting digital twin design is to replace the costly pure-physical commissioning and test phases,

so-called Virtual Commissioning. However, the desired end goal does not include only the system validation but also the design innovation.

Starting from a predefined CAD design of this portal stacker, a simplified multi-body model has been developed. The structure of the gantry starts with the support structure on which the bridge can translate driven by two motors. Nested in the bridge beams a carriage performs another translation, perpendicular to the previous one. A chain mechanism performs the translation along the vertical direction of the cup frame. The cup frame is then governed by two different revolute joints, one for the rotation with the axis normal to the horizontal line and one tilting component.

With the help of `MatLab Robotic Toolbox`, a kinematic analysis as been performed, starting from the geometric robot parameters applied to the Denavit-Hartemberg convention up to the homogeneous transformation matrix that relates the pose of the end-effector (i.e. the cup frame) with the joint variables $\dot{\boldsymbol{q}}$. With this preliminary model it has been possible to define the robot configurations for several trajectories according to predefined position, but also by an interactive GUI. In addition, using the inverse kinematic solver and having defined the environment around the robot, it has been possible to check for collision between the robot configurations of a given trajectory and the surroundings.

A second mechanical model of the gantry has been developed using `MatLab Simscape Multibody` which offers a simulation environment for tri-dimensional mechanical systems using as building blocks bodies, joints, force constraints and so on. Given the proper inertia matrices at the bodies and defined the interconnections between rigid bodies and their relative joints, a trajectory response analysis has been investigated. In facts, each joint has been driven according to a predefined function that can be classified into macro-categories, such polynomials, piecewised trajectories and harmonic functions. In general, trajectories that exploit maximally the motor capabilities are faster in time, but cause abrupt vibrations to the mechanical system due to the infinite jerk applied to the joints. On the other hand smoother kinematic profiles will affect badly the cycle time. In addition, a spectral harmonic analysis of the trajectory is observed and defined as the input of a dynamic distributed parameters model (mass - spring - damper). The frequency response is analysed to better understand how the resonance peaks act on the mechanical system. From the Multi-body model it is possible to recover the kinematic profiles, but also the torques measured at the joints where it is possible to see the expected behaviours.

Once observed the mechanical point of view, it is important to better understand the electrical and software behaviour of the system. For sake of simplicity, the hardware configuration of the control network is composed by a CPU (`Siemens ET200SP`) and a motion control system (`Siemens SIMOTION D425-2`). The former is a control unit that performs logic and operations on IO and communicate with

other CUs and networks; the latter instead, control the drive commands, defining the positioning and ensuring the synchronism of motion between multiple axis with the creation of cams and gearings.

Concerning the PLC, the program code is developed as a combination of Ladder networks and Structured Code Language functions. The algorithm can be subdivided in the following macro-areas: access area and emergencies, communication with the process line and with the production manager, rack counter plate, position manager and checkpoints, input and sensor monitoring and pneumatic control for cups inclusion.

The program developed for the SIMOTION control unit govern the motion command to be imparted to the drives. This operative system works with tasks that can be recalled at event, or cyclically, or as interrupt, or in the background according to the level of priority. The algorithm principle requires that the drive commands (s.a. enable, positioning, camming...) must be done once and thus a case status code is developed according to the feedback state from the technological objects. The important aspect that has been developed regards the synchronisation of axis according to the cam. In facts, to complete a multiple-axis trajectory an axes acts as the master and its movement cause the slave axis in sync to move accordingly to an in-cycle designed cam.

The final element of this thesis is the development of the `Siemens SIMIT` project, which is a simulation software capable to create emulated instances for the control units in the system (respectively with `PLCSIM Advanced` and `SIMOSIM Advanced`). Once defined the coupling between the software and the emulated instances a communication between the hubs is created. Moreover graphs have been developed in order to govern the field inputs and to recreate black blocks that simulate the behaviour of external elements such as external encoders and the PROFIdrive telegram communication for the correct functioning of the closed-loop control positioning motors.

To summarize, the digital twin concept will be fulfilled once the multi-body model will be coupled with the `Siemens SIMIT` simulation environment. This final step will be performed in the near future by using the native Multi-body designer of Siemens, `Siemens NX Mechatronics Concept Design`, where encoders and other field bus inputs can be simulated according to a 3D design and a complete virtual commissioning can be performed.

This new solution for the design, validation and testing will lead to an immediate decrement of commissioning costs and, thanks to a new iterative project phase, to an improvement of the total quality of the system, opening the doors for the Industry 4.0 to the complete business unit.

Table of Contents

List of Figures

Chapter 1

Overview on flat glass industry

Flat glass, also known as sheet or plate glass, is most commonly used for manufacturing windows, doors, mirrors and solar panels. It is produced by melting a powder mixture, principally sand, into liquid and spreading it to the desired thickness. The molten liquid is then cooled to obtain the desired product. Flat glass is manufactured through controlled thermal and chemical reactions to ensure the required toughness in the structural formation. Nevertheless, flat glass is highly flexible and its features can be customised according to post treatments.

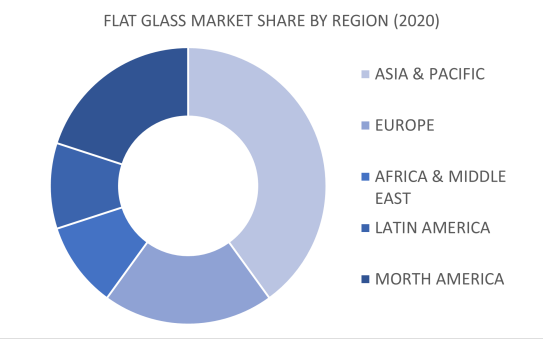


Figure 1.1: Market region share

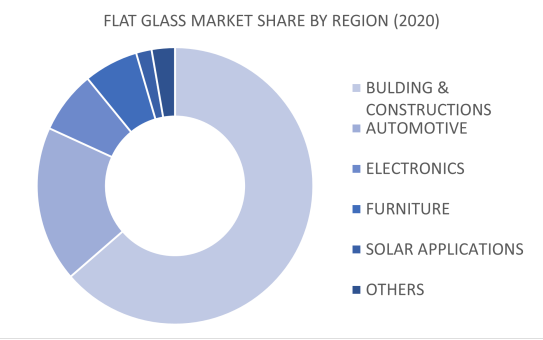


Figure 1.2: Market application share

Flat glass represents a substantial slice in the manufacturing market and it is interconnected with several sectors, such as automotive and constructions. In facts, in 2021, the global flat glass market size reached US$ 101.9 Billion and the estimations state that it will reach US$ 144.2 Billion by 2027, exhibiting a growth

rate of 5.87% during this period, keeping into account the fluctuations due to the COVID-19 global pandemic and its consequences on the different end use industries [1]. Analysing the customer sectors, the construction industry is the key factor driving the oscillations of the market.
In addition, the upcoming development of an energy-sustainable society, enlarge the request of flat glass as a key player in the construction of eco-friendly green buildings, aimed at minimizing carbon emissions into the environment. On the same theme, flat glass is widely used in photovoltaic modules, e-glass structures and solar panels. Thanks to its recyclablity features, insulated flat glass is gradually replacing the traditionally construction materials like bricks, stone and wood thus allowing a pollution reduction and enhancing the comfort and the light for the inhabitants. Another important field of application is the automotive. The automobile manufacturers are increasingly employing tempered glass due to its shatterproof properties that can prevent severe injuries and possible life threats in case of accidents.

1.1 Layout of a float glass production line

The majority of the sheet glass is manufactured through a float production line, which is characterised by a continuous stream of glass that is floating over a tin bath and so allowing the homogeneous cooling and the removal of undesired residual stresses.

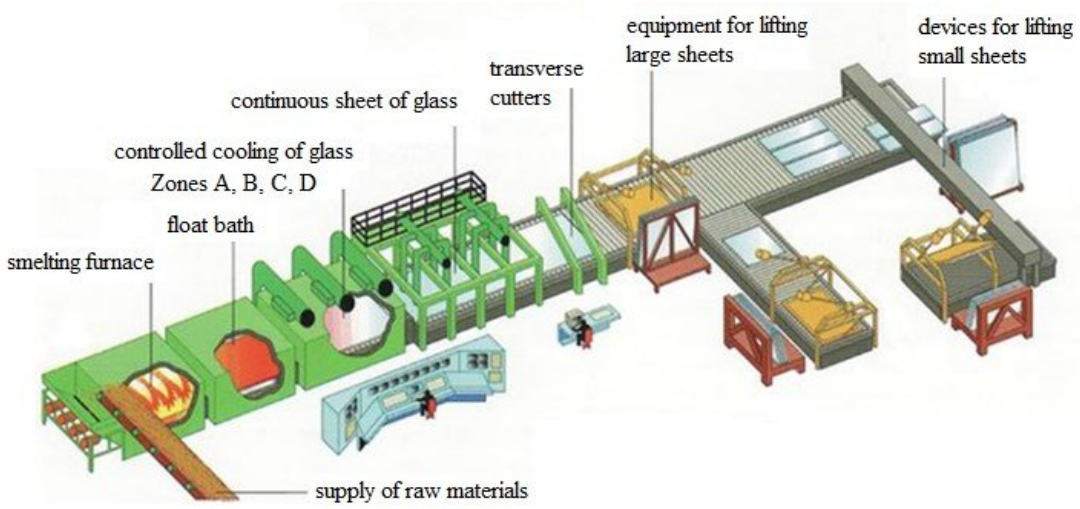


Figure 1.3: Layout of float production line

This process is the initial step for further treatments and for the manufacturing of more complex sheets, such as coated glass and laminated glass [2]. It is common sense to adopt a layout that consists of a main line, where the glass flow is cut, which feeds secondary lines that complete the manufacturing of the lite.

1.1.1 Glass formation and microstructure

A glass melting furnace is the starting point of the process and it is the stage where the raw materials are melted into glass. The principal aspect in this early process is the power required to modify the micro-structure of the batch of silica-sand. Depending on the intended use, there exist different designs of glass melting furnaces which use different power sources. Most of the glass furnaces are fossil-fueled (natural gas, heavy or light oil) or electrically powered, with the possibility of a combined energy to optimize the efficiency and the cost of the energy income.

Another fundamental aspect for the glass melting furnace is the refractory material building the chamber for the heating process. The melting tanks are copmosed of refractory materials. Typical elements are alumina (Al_2O_3), silica (SiO_2), magnesia (MgO), Zirconia (ZrO_2) and combined, it is possible to obtain the necessary refractory ceramic materials.

The glass raw materials are usually fed to the glass melting tank continuously or in discrete portions (batches) [3]. In addition to the basic components, such as the silica sand, the batch can also contains the cullet from recycled glass coming from the defective scraps discarded along the line process in order to lower the power consumption (approximately 2% energy savings for every 10% of cullet).

For economical reasons, the glass melting furnaces are operated continuously throughout the year for mass glass in order to save energy in the heat up process. Apart from small planned intermediate repairs, during which the furnace is taken out of operation, the furnace journey can last up to 16 years and more (depending on the product group, capacity and color change). The capacity can range from about one ton to over 2000 tons and the daily throughput can range from a few kilograms to over 1000 tons. The operating temperature inside the furnace, above the so-called glass bath is about 1550 °C. This temperature is determined by the composition of the batch and by the required amount of molten glass, as well as the design-related energy losses.

Continuously operated furnaces consist of two sections, the melting tank and the working tank. These are separated by a passage or a constriction. In the melting tank, the batch is melted and refined. The melt then passes through the passage into the working tank and from there into the feeder (fore-hearth). In float glass production, the glass is fed at special wide outlets as a glass ribbon over a so-called float bath of liquid tin and it is moved over a profiled roller [4].

If pure oxygen is used instead of air for the combustion of fossil fuel (preferably

gas), energy savings and, in the most favorable case, lower operating costs occur. The combustion temperature and thus the heat transfer are higher, while the gas volume to be heated is lower.

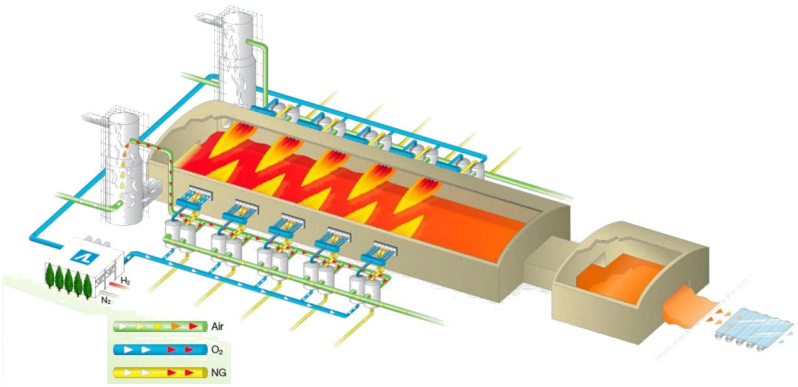


Figure 1.4: Cross-fired float furnace

In the heating of the combustion air, a regenerative or a recuperative system is adopted in order to increase the energy efficiency.

In a typical regenerator, the hot exhaust gases (1300 °C - 1400 °C) are fed discontinuously in the melting chambers through a latticework of refractory bricks that is heated during the process. After this storage of the thermal energy of the exhaust gas by the lattice, the direction of the gas flow is reversed and the fresh, cold air required for combustion now flows through the previously heated lattice work of the chamber. The combustion air is thereby preheated by the refractory materials up to approximately 1200 °C. This results in considerable energy savings. The process is repeated periodically at intervals of 20 to 30 minutes. The chambers are thus operated discontinuously. This process can lower the energy consumption up to 30%.

The other equipment used to improve the efficiency of the furnace is the recuperator. This subsystem operates continuously and consists of a metallic heat exchanger between the exhaust gas and fresh air. Because of the metallic exchanger surface (heat-resistant high-alloy steel tubes in combination with a metallic double shell), a recuperator can only be operated at lower exhaust gas temperatures and therefore work less effectively (40%). Thus, only relatively lower preheating temperatures (max. 800 °C) are achieved.

Float glass furnaces are plant specifically engineered for the production of soda lime glass. The requirements concerning glass quality are stricter and differ from

those related to hollow glass. In the adoption of this complete configuration, instead of the throat, a neck is present, where coolers are installed, ensuring the perfect temperature for the subsequent float process. The float process itself starts, when the glass leaves the furnace at an overflow and enters in the tin bath. The adequate furnace insulation and optimal flow profile of the combustion air, as well as efficient preheating of the combustion air, allow the operation of the furnace with minimum energy consumption.

In this embryonic stage, the micro-structure of the melted glass plays a crucial part. The glass(or vitreous solid) is a solid formed by rapid melt quenching. Principally, a glass is an unformed solid that exhibits all the typical mechanical parcels of a solid. The infinitesimal structure of a glass lacks the long- range periodicity observed in crystalline solids [5]. Due to chemical cling constraints, spectacles do retain a high degree of short-range order with respect to original infinitesimal polyhedra.

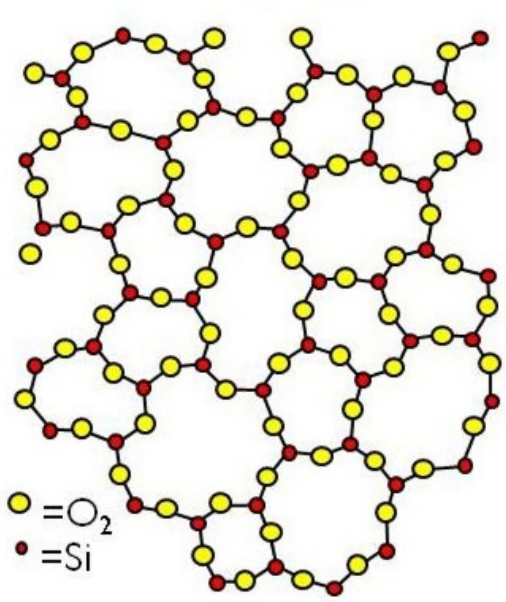


Figure 1.5: Micro-structure of the glass

From a microscopical point of view, glass can be identified as a liquid due to its lack of a first-order phase transition [6]. During this transformation, most of the thermodynamic variables, such as entropy and enthalpy, are not continuous through the glass transition range, hence the equilibrium theory of phase transformations does not hold for this material. The glass transition can not be classified as one of the classical equilibrium phase transformations in solids, but it concerns the decrement of viscosity with the lowering of the temperature. The typical

composition of the glass output from a float glass line is the soda-lime glass, where the amorphous structure of silica (SiO_2) stands for the 70% to 74% composition by weight. The remaining part comprehends several additives that modify also the physical characteristics of the material. Sodium carbonate (Na_2CO_3, "soda") is the principal additive and its contribution is related to the decrement of the glass-transition temperature. Unfortunately, this molecule is soluble in water and so other additives, such as lime (CaO, calcium oxide, generally obtained from limestone), magnesium oxide (MgO), and aluminium oxide (Al_2O_3) are commonly added to improve chemical durability and resistance. Soda–lime glasses represent over 75% of manufactured glass. The characteristics of Soda–lime–silicate glass is transparent, easily formed, but as drawbacks, it has a high thermal expansion and poor resistance to heat.

The second step of the production line is the glass annealing lehr oven, which is a long kiln with an end-to-end temperature gradient [7]. During this process, the newly made glass is annealed along the transportation thanks to the temperature gradient done with rollers or on a conveyor belt. This stress relieving process strengthen the glass reducing the internal stresses and as a consequence, lower the probability of crack formation.

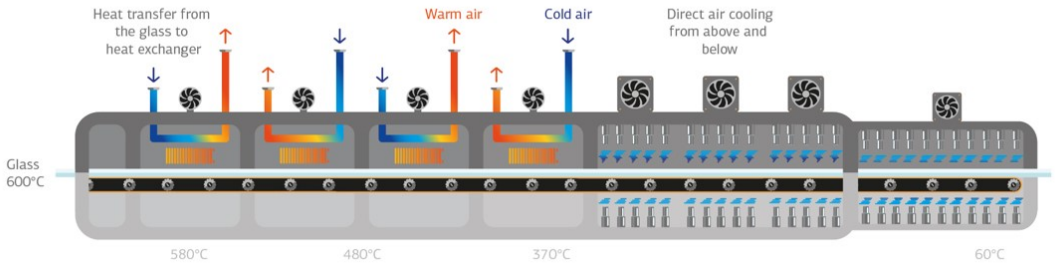


Figure 1.6: Glass annealing lehr oven

In facts, the rapid cooling of molten glass results in an uneven temperature distribution throughout the material, since the glass ribbon is subjected to different cooling rates on the surface and in internal body of the glass. This temperature difference represent the cause of mechanical stresses throughout the molten glass, which can be sufficient to cause the material to crack as it cools to ambient temperature.

In the process of annealing glass, the temperature is first equalised by holding the glass at the annealing point for a predefined period of time. The glass is then

slowly cooled at a rate that depends upon the maximum thickness of the glass. From a thermal analysis, the temperature at which the glass ribbon leaves the tin bath is circa 600°C, well above its annealing point. The annealing temperature range for float glass is defined from 540°C to 470°C and it is the working area of the central part of the lehr [8]. The basic working principle is to exploit the mass air to cool uniformly the glass to near room temperature without causing the glass to break. This stage requires a correct air recycle to be implemented and this is possible thanks to fans that blow air directly onto the top and bottom of the glass ribbon.

For the Float Annealing Lehr the transportation of the ribbon is performed with a drive mechanism that sets the rollers in motion which pull the glass ribbon from the tin bath into the lehr. An important aspect is speed of the ribbon, which is directly related to the velocity of the rollers. The affecting factor for this variable is the glass thickness that can be monitored in real time and from this evaluation the velocity is controlled to obtain the desired thickness. The float glass that exits from the annealing oven is at the correct temperature to be cut and subsequently stored.

1.1.2 Glass quality defect detection and cut definition

The definition of the cut is set in order to optimize the non-defective glass throughput based on two principal boundary conditions given by the production required by the user and the defects present on the glass ribbon. The former condition is extracted from a production worksheet, where the end-products available are defined according to the dimensions and the minimum quality accepted.

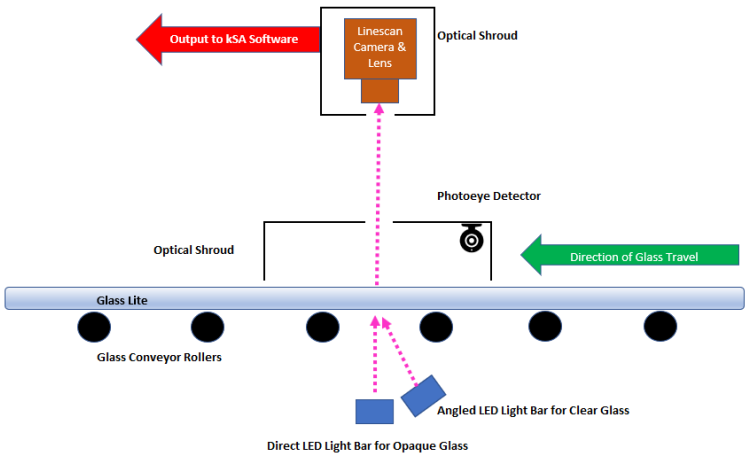


Figure 1.7: quality defect inspection portal

The latter demand comes from the online quality defect inspection portal which is located at the exit of the lehr oven. Typically, this detection method is performed with several digital monochrome cameras that analyse the light coming from a red LED light source that traverse the float glass ribbon. The inspection principle holds on the fact that good quality glass maintains a good optical homogeneity, resulting in the stability of refraction angle, light path and power of the red incident beam that pass through the glass. In case of abrupt changes in this refractive behaviour, a defect is acknowledged. This image processing requires high efficient algorithms based on the machine vision.

Figure 1.8: Real-time glass defect image

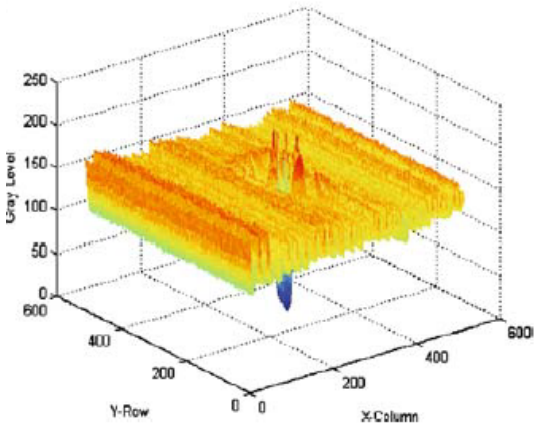


Figure 1.9: 3D image processing of the inspection

Typical defects detected are bubbles (gas inclusion inside the bulk glass), stones (partially melted particle of rock, clay or batch ingredient embedded in the glass), tin residues on the surface and the optical distortion [9]. This information on the defects (position, dimension and gravity) is then evaluated by the optimization cutting algorithm that defines the cutting lines on the ribbon and thus the plate dimensions to fulfill the production request and reduce the cullet, as shown in fig. 1.10.

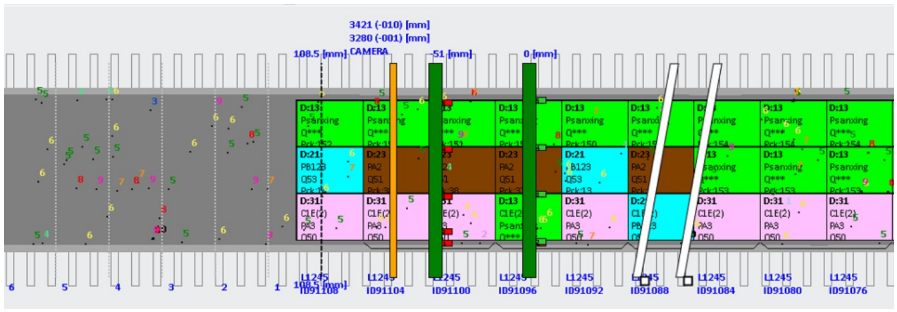


Figure 1.10: Glass cutting optimizer

The undesired portions of glass ribbon containing defects will be cut and further along the line this parts will be excluded from the processing thanks to tilting conveyors that sends the unwanted lites to scrap. These debris will be then recycled in the batch that will be melted in the starting oven.

1.1.3 Cutting and geometry check

Once defined the cutting scheme, the glass ribbon is moved by the conveyors through the cutting portal. This section of the line is composed by multiple bridges with one ore more cutting heads. The large majority of processing lines currently use mechanical roller cutting for sheet glass. The rollers have a wedge-shaped obtuse-angle section and are manufactured from hard alloys, based on tungsten carbide [10]. The cutting proceeds in two stages: first a scratch is made, thanks to the head tool, and, in a second stage, a bending force is applied across the cut line. From this, the sheets of glass are divided. A groove remains after the roller has passed along the surface of the glass forming a network of intersecting cracks in the surface layer. the path of this crack grid coincides with the action of the shear stress. Furthermore, plastic deformation of distant layers is exhibited and gives rise to additional stresses which cause the cracks to grow [11]. In general, after a scratch, three types of cracks can be identified: central normal, superficial and lateral as shown in fig. 1.12.

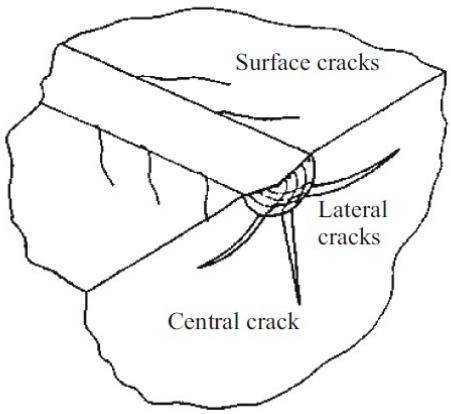


Figure 1.11: Crack formation scheme

As a result of the relaxation of the stresses, the crack starts to grow to equilibrium dimensions arising in the surface layers as a result of the wedging action of the roller and the residual stresses in the glass [12]. According to the intensity of the stresses, the crack increases with the load and decreases with the sharpening angle of the roller; the depth of the cracks is proportional to the intensity of the stresses. The forces exerted by the roller on the glass are directed perpendicular to the edges of the glass and act in shear in the adjoining layers. Considering a sharper roller, the forces act further away from the normal to the glass surface and therefore the region of the stresses results wider and in deeper layers of the sheet as shown in fig. 1.12. The sharpening angle of a roller is defined on the basis of the thickness of the glass to be cut.

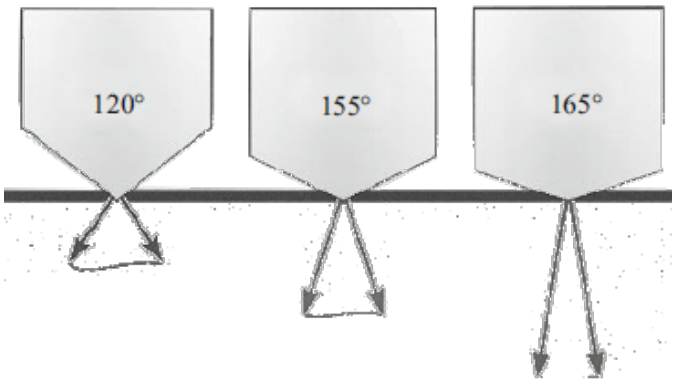


Figure 1.12: Different angles of the roller cut

In the case of acute-angle rollers, undesirable lateral cracks are free to grow. Hence the use of obtuse rollers is preferable, due to the fact that the stress action is directed primarily downwards. For thick glass, a larger force is required and this has a negative effect on the edge quality. In order to lower the breaking force, tensile stresses must be introduced along the line of the cut, and therefore an obtuse-angle roller must be adopted with an higher pressing force of the head cutter. Typically, to maintain an high quality cut, lubricating liquids are used in roller cutting. These liquids have several functions, from the increase of the roller service life to the creation of an hydraulic cushion that gives a more uniform force distribution during the cut. Mechanical rollers are the most suitable solution due to the following advantages:

- equipment is inexpensive;
- operating costs are low;
- cutting operation is fast, since the cutting speed can reach 120 m/min);
- no limits on the glass sheet dimension that can be cut

The drawbacks of this process can be summarised in the relative poor quality of the cut due to the non-planarity and high residual stresses of the sheet. The first cut is done in the transversal direction of the glass ribbon and is intended to split the continuous flow of glass that comes from the furnace in the desired sheets. This cut is performed by mechanical rollers whose trajectory has a certain angle with respect to the normal direction of the flow. Since the ribbon has a certain continuous velocity, in order to produce a precise perpendicular cut, the speed of the head cutter is calculated in relation to the glass flow speed and the misalignment angle.

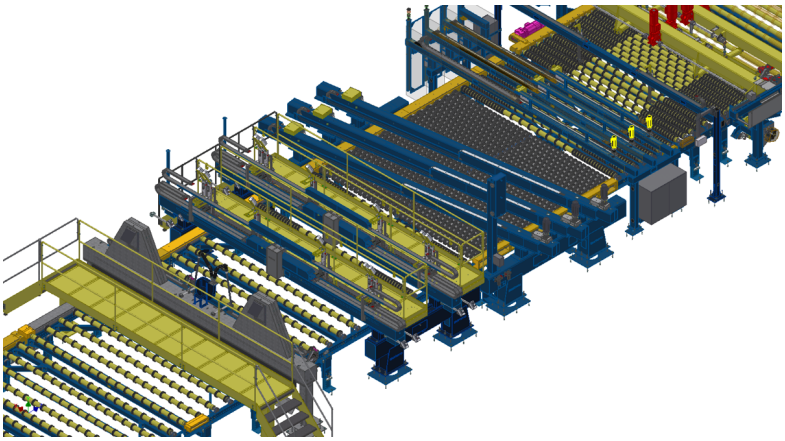


Figure 1.13: Float glass cutting mechanism

Multiple transversal head cutters can be adopted for closer transversal cuts in the case of smaller sheet dimensions. After this groove creation, a breakout bar is lifted up from under the glass ribbon and increase the tensile stress concluding the cut. To create a certain gap between the sheets, accelerating conveyors are adopted immediately after this stage. The next step is the longitudinal cut performing. Depending on the chosen production by the optimizer, at least two cuts are performed in the direction of the flow to remove the marks given by the dented gears that pulled the glass flow. In this sense, the glass ribbon drift must be taken into account in order to obtain the correct dimensions. An edge trimmer roller works in a similar way of the breakout bar, splitting the glass products from the marginal scrap. After this cutting process, a quality check is performed to evaluate the correctness of the glass sheet (perpendicularity and dimension tolerance) and to acknowledge the presence of a surplus of glass due to an incorrect break.

1.1.4 glass plate handling

The end products of the float glass line are the glass plates. Elements belonging to the same production quality are usually stored on racks in several packs, according to fig. 1.15.

Figure 1.14: Example of a glass rack

The storage of the glass lites can be done with the help of anthropomorphic robots in case of small dimensions or with a cartesian gantry stacker. The definition of the final destination is defined by the overall system and can be controlled and monitored by the SCADA, which is the high level control and supervision system of the production plant. This destination is typically located on the final end of a branch of the main line.

Figure 1.15: Anthropomorphic arm for glass plate handling

To redirect the glass plates to the correct path, the lite can be lifted up and rotated with pop-up conveyors and with crossed belts it can be forwarded to the secondary lines for the unloading. The unloading from the line can be performed in several manners. In facts, the lites can be positioned on the racks in portrait mode (where the largest lite dimension lies on the height of the rack) or, viceversa, in landscape mode. Another important differentiation is the picking method. In fact, the glass can be picked up from the top, in air mode or from the bottom where the suction cups lift the plate in tin mode. In the latter case, the tilting conveyors facilitate the maneuvers for the handling station. The focus of this analysis is keen on the behaviour and the simulation of a gantry stacker handling mechanism, which is the final actor in the processing of glass plate with larger dimensions. However, due to its flexibility, This machinery can be used also as the feeder (and thus, the starting point) of an automated line that works glass plates to transform it in more complex final products. Examples of this post-treating processes are the coating lines and the laminated lines, whose products are shown in fig. 1.16.

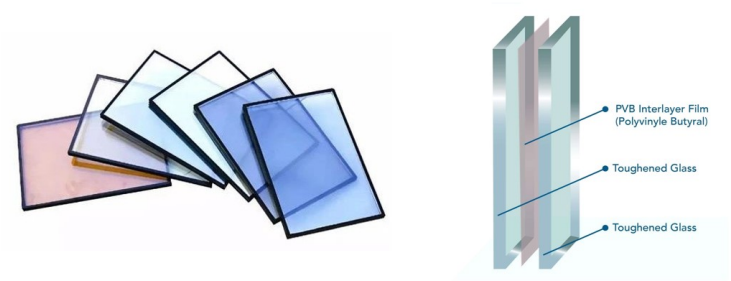


Figure 1.16: Examples of a coated and a laminated glass

Chapter 2

Digital twin concept

One of the fundamental architectural elements in the Industry 4.0 vision is the digital twin, based on the basic principle that each physical entity should have a reliable virtual representation in the digital world. Although this term is widely overused in a larger sense, this concept gives origin to widespread possibilities in the industrial scenario.

In the following chapter, a brief overview of the literature around the digital twin concept and the general principles of virtual commissioning and Industry 4.0 is presented.

2.1 Principles of Industry 4.0

At the beginning of the 21^{st} century, the fourth industrial revolution took place, reshaping the way individuals live and work. Industry 4.0 is a realistic answer to the seek for sustainability. However, this sustainability has not to be limited to the environmentalism, because it also considers the preservation of economical and social resources [13].

According to United Nations, sustainability refers to a movement for ensuring a better and more sustainable well-being for all the population, including the future generations, which aims to solve the everlasting global issues of injustice, inequality, peace, climate change, pollution, and environmental degradation.

In simple terms, in the industry 4.0 scenario, interconnected computers, smart materials, and IA driven machines communicate with one another, interact with the surroundings, and make decisions with large autonomy. Concerning the manufacturing, the digitization of the productive and business processes may offer numerous advantages such as the increment of manufacturing productivity and a limitation in the waste reduction [14].

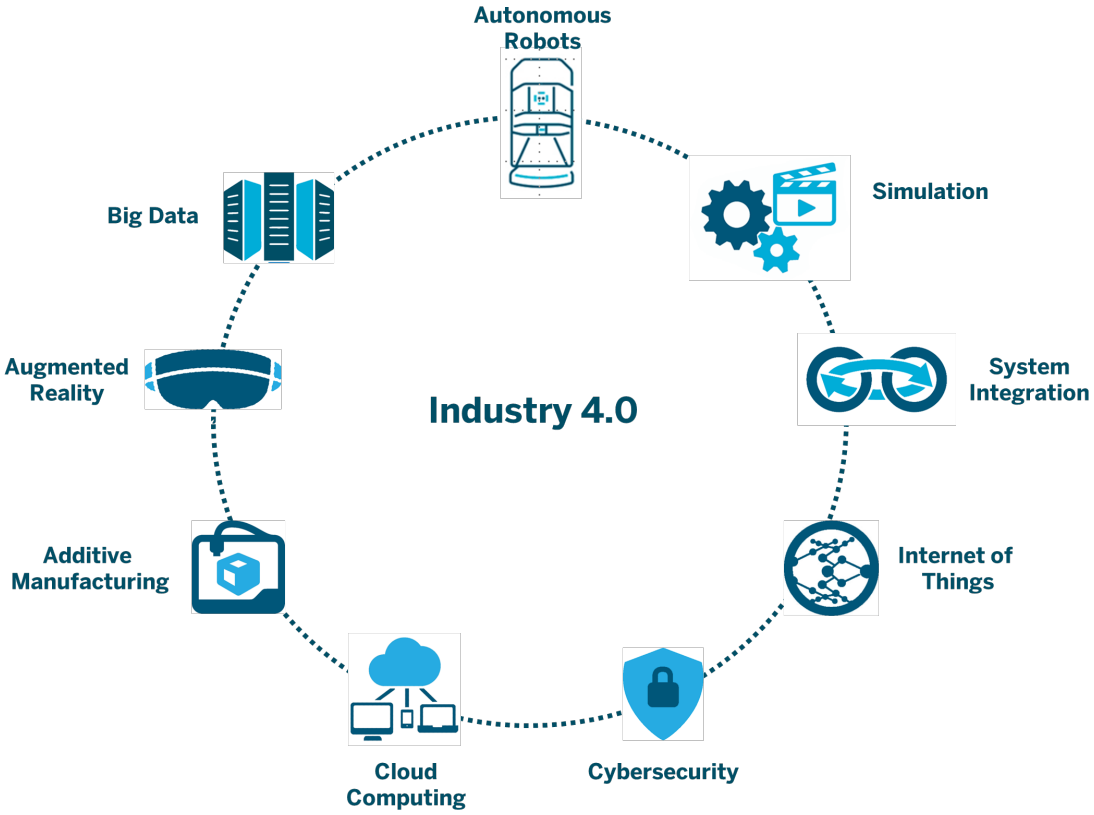


Figure 2.1: General overview of Industry 4.0

Originally, Industry 4.0 was a concept strictly related to the manufacturing industry. Nevertheless, this philosophy has evolved during the years and now involves the digital transformation of the whole industrial sphere and consumer markets.

According to the literature, in the following paragraphs are listed the basic notions and characteristics of the Industry 4.0 [15].

- **Business model novelty and innovation**: interoperability, decentralization, and real-time capability have drastically transformed how business units design and deliver goods and services.

- **Carbon/harmful gas emission reduction**: according to United Nations, industrial emissions are the cause of circa 40 % of greenhouse gas emissions. The upcoming manufacturing digitization creates several opportunities for lowering the carbon footprint by increasing efficiency and flexibility of production and reduce waste. In the extreme sense, new business models can be developed with the intent to shift the focus from mass production to the mass

customization and product individualization to decrease the contribution to the materialization of low-carbon future.

- **Corporate profitability improvement**: the corporate profitability is infected by the application of new technology trends such as IIoT, additive manufacturing, data analytics, and smart manufacturing that can lead to the following economical benefits: material flow optimization and inventory costs, better time-to-market of products, resources efficiency, superior product innovation and quality and improved production capacity and reliability.

- **Energy and resource sustainability**: Digital transformation supports environmental sustainability through sustainable energy and resource transformation. With the advent of Industry 4.0, the development of Smart energy grids facilitates the integration of power grids and renewable energy sources. Moreover, efficient production systems and advanced digital manufacturing technologies contribute to material efficiency and energy saving.

- **Environmental responsibility development**: Thanks to the development of reactive and proactive environmental-friendly practices it is estimated to have implications for socio-economic sustainability. An important aspect is represented by the productivity impact enabled by collaborative production management, supply chain-wide knowledge management capabilities and production flexibility.

- **Human resource development**: Smart manufacturing ensures process simplification and automation, thus resulting in enhanced decision-making processes that can improve HR efficiency [16]. With AI and data analytics tools, it is possible to predict meaningful career development patterns and learning programs based on the individual behaviour, experience and skills, personality, and learning patterns of each employee. With the use of Internet of People (IoP) working operators can interactively communicate with each other improving leadership and middle-management. Lastly, effective ways of industrial training can be performed with the help of augmented or virtual reality and simulation of the system.

- **Increased production efficiency and productivity**: According to [14], the Manufacturing digitization allows the implementation of an hybrid lean-agile manufacturing ecosystem. In addition, the automation interoperability affects positively the production efficiency and productivity by improving process control measures, facilitating real-time maintenance, monitoring machine performance in real-time [17]. In the end, the use of automation systems reduces human contribution leading towards lower human errors, risks, and safety countermeasures.

- **Manufacturing cost reduction**: The cost-saving advantages that Industry 4.0 offers are related to improved process control techniques, enhancement of manufacturing accuracy, precision and quality, real-time monitoring and accident prediction and prevention. In general, the production process towards an autonomous system leads to lower human resource costs and material/resource/energy efficiency.

- **Manufacturing agility and flexibility**: Through Industry 4.0 and smart manufacturing, manufacturers are enabled to adopt a more agile and flexible manufacturing system. With the help of industrial simulation, digital twins, and data analytics it becomes possible to micromanage processes and exploit real-time capability.

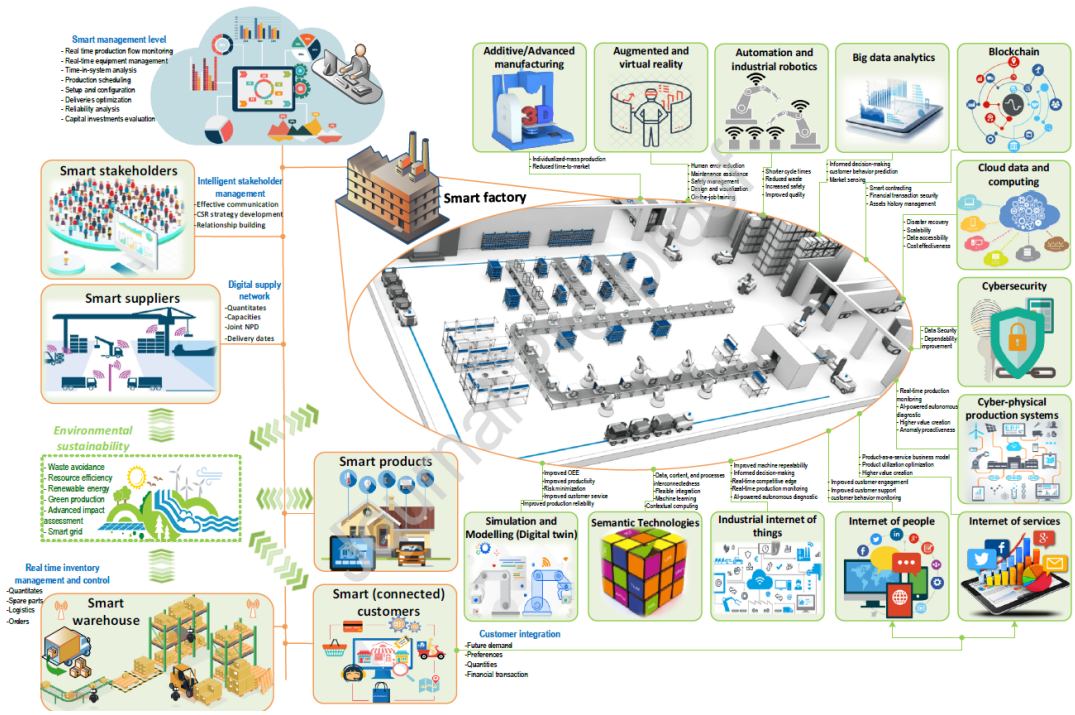


Figure 2.2: Architectural design of a smart industry

To achieve the Industry 4.0 expected goals, the new up-do-date technologies are exploited and coexist. In facts, the cooperation of Information, Digital, and Operation Technologies (IDOT), Automated Guided Vehicles (AGV), robots, Augmented and Virtual Reality (AVR), data analytics, cloud computing, Internet of Services (IoS) and Artificial Intelligence (AI) will lead to the desired industrial revolution. This solution has become feasible very recently, thanks to the maturity in terms

of integrability and interoperability between the different up-to-date technologies necessary for digitization.

As shown in 2.6, the industry 4.0 digital transformation relates to the complete digitization and integration of the final product, form the production to the end of its natural life-cycle. At the top of this architecture, the Digital Supply Network (DSN) manages an integrated supply network (both horizontal and vertical) including smart suppliers, connected customers, smart factories, production machinery that communicate and interact on a real-time basis in a worldwide scale [18].

This communication efficiency, transparency, surveillance, and control in the future smart factory leads to lower downtime, waste, defects, and risks across the whole production process. [15]. The virtualization principle of Industry 4.0 and the simulation of sensor data acquired from the physical world into digital twin models permit to evaluate the behaviour of the real copy and seek for predictions and optimized solutions of manufacturing operations [19].

Considering a real design department, the digital twin can be used to test and simulate the real future use of the physical products and ascertain the digital footprint throughout the complete lifecycle.

Lastly, the use of industrial robotics, automation and additive manufacturing push towards modularity characteristics to facilitate an agile, flexible and decentralized production environment tailored to the ever-changing customer requirements [20].

2.2 Digital twin concept

The progression of the technologies has improved the development of the design process in all its sides. Thanks to Computer-aided technologies (CAD, CAE, CAM, FEA,..) Big data, Internet of things (IoT), artificial intelligence, cloud computing, the industry conception has been revisited [21]. The coexistence of these technologies in the planning procedure creates a connection between the physical world and the digital world, which represents the future manufacturing idea, characterized by growing complexity and high demands from the final client. The process of this integration is at its first steps. The actual point in the technology S-curve for this scenario is set on the digital twin concept [22].

The digital twins concept is the redefinition of the modeling and simulation technology. With digital twins technology, automation designers are capable of overcoming the limitations on the engineering analysis capabilities of simulation. In general, digital models are divided in three principal categories according to the level of emulation:

- **digital replica**: considers principally the automatic projection of system constructions, predicting and optimizing scenarios.

- **digital shadow**: emphasizes mainly the mathematical modeling to describe the physical/chemical attributes of a given system.
- **digital twin**: an integrated multi-physics and multi-scale simulation of a system that can reproduce the mechanical, electrical, software, and other properties of the reality.

The focus of this project is the digital twin. This solution is typically developed in order to optimize the physical product/system based on the updated real-time data synchronized from smart sensors. Typical industry fields that adopted this method are the manufacturing, aviation, healthcare, and medicine.

In the Smart Manufacturing System Design, the simulation technology is not simply restricted to a standard tool for supporting designers to solve specific engineering problems, but it is the core concept for the improvement of the design phase to subsequent lifecycle activities.

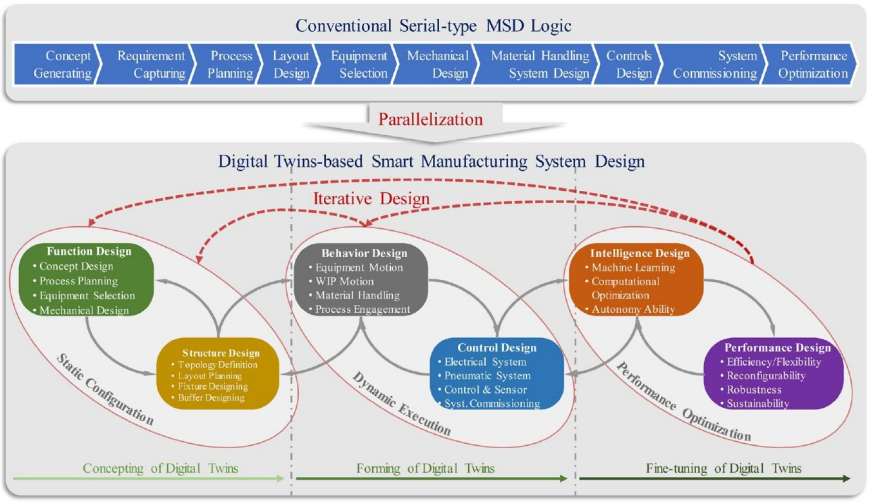


Figure 2.3: Concept map for Smart and Classical Manufacturing System Design

As shown in 2.3, the classical engineering and design methods lack of an effective iterative procedure of optimization made of small adjustments in the design process. Instead, digital twin models are macroscopic components that can be modified and tailored. Interesting outcomes form this new technology practice are complex design elements and data that can be analysed to avoid upcoming defects (i.e. production bottlenecks) [23].

The design process for the creation of a digital twins is composed by three phases:

1. **Conception stage of digital twin**: in this phase the static configuration of the system is been performed, defining the function design and the structural

design by a continuous interaction between the two spheres. The objective of this stage is not the pursuit of accuracy but a continuous trial-and-error design.

2. **Forming stage of digital twins**, where it is studied and developed the dynamic execution of the system. At this instance, the behavioural design and the control software design are developed with the continuous feedback from the several areas. In this step, key technical design parameters, like motor sizing, are determined by simulations. At this stage, it is a good practice to perform the semi-physical system commissioning required by industry-customer specifications and safety measures.

3. **Fine-tuning stage of digital twins** In this last step, the Performance optimization is searched by investigating the intelligence design and the performance design. In this interval, key factors are the flexibility, the robustness and the artificial intelligence teaching methods.

The digital twins is the appropriate settlement for the reduction of expensive trials and errors in the design process that can accurately spot the critical points that need to be studied, modeled and improved. In another words, this approach leads to a faster and cheaper achievement of the manufacturing validation by running fewer physical test, with the addition of higher design efficiency and quality. From a mere economical point of view, the most direct value of the digital twin implementation is the abolition of the costly pure-physical commissioning and test. Although, the higher returns are related to design innovation thanks to the intrinsic iterative process.

According to the previous development definition, it is possible to define the correct framework corresponding to the Digital Twin for Smart Manufacturing System Design, which is a *Function-Structure-Bevahiour-Control-Intelligence-Performance* (FSBCIP) as defined in fig. 2.4 [24].

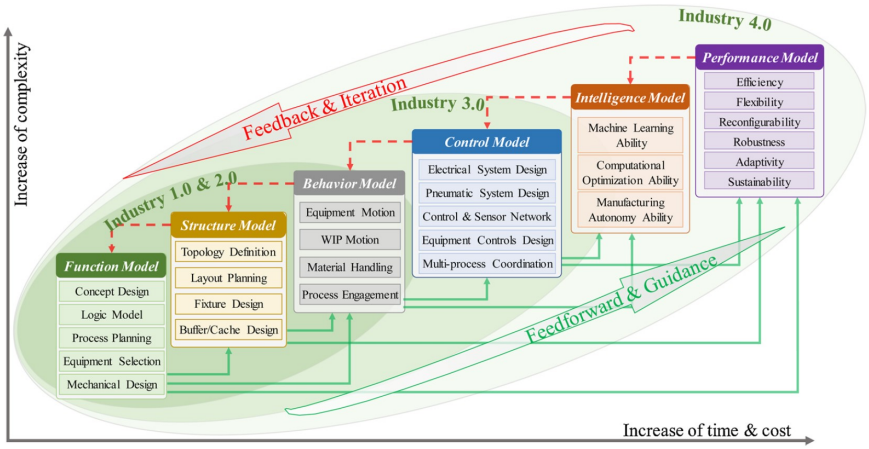


Figure 2.4: Framework of the smart manufacturing system design

In the following graphs the various stages of this process are briefly described:

1. **Function model**: This preliminary phase defines a structured description of the activities for manufacturing and underlines the relations between the manufacturer demand and the constitutive elements of product manufacturing, assembly, maintainability, and safety. This introductory model includes the general process planning, the equipment selection, and the mechanical design of parts.

2. **Structure model**: In the structural development phase, the assembly relations and connections among the mechanical structures before designed. The result of this interrelation is the basis for the processing and transportation processes of the goods, energy consumption, information networking, and motion behaviour. After this stage, the topology definition, layout planning, are determined.

3. **Behaviour model**: Here is described the mechanical motion transmission, transformation, and their mutual relations. This submodel comprehends the study of equipment motion, Work In Process (WIP) motion and material handling. From this phase, the principal elements of motion behaviour basically include forces, displacements, velocities, acceleration and jerk.

4. **Control model**. In this stage, the program codes that define the automation routines is developed to interacts correctly with the structure and the electrical sensor grid. In general, it comprehends electrical and pneumatic system designing, sensor networking and multi-processing coordination.

5. **Intelligence model**. In this progress stage, the Intelligent model is capable of the description, development, and validation and the learning ability. In

this moment, reasoning rules, machine learning algorithms and computational optimization algorithms are typically applied.

6. **Performance model**. In this final stage, the model includes the evaluation and optimization of the system performance in terms of efficiency, flexibility and robustness.

The input of the design process are typically defined by personalized requirements and parameters, while the output is represented by the virtual models, the motion scripts and trajectories, control schemes and intelligent algorithms.

2.3 Virtual commissioning

In this section, the final phase of this thesis, the **Virtual commissioning** is theorized. The principal goal of Virtual commissioning (VC) is to test manufacturing systems and associated control programs through simulation, conducted on emulated instances before the real systems are installed and realised. The expected benefits of this procedure consist in a reduction of debugging time and a limitation of corrections performed during real commissioning. To obtain this achievement, Virtual commissioning requires a detailed manufacturing system model available for the simulation of all the technical components [25].

Nowadays, the design of automation systems takes place in an industrial environment characterized by significant cost constraints, lack of material disposal and aggressive strategies for rapid time-to-market. For these reasons, the scheduled time-frame for engineering is progressively tightening whereas the demands on planning accuracy and planning quality are growing.

Manufacturing systems are a collective of various elements such as storage, robots, conveyors, handling and transportation systems, processing machinery tools, control and HMI systems, distinguished in standard parts and custom purpose-built components [26]. The development design of a production assembly comprehends several phases: facility design, mechanical engineering, electrical engineering and automation engineering (programming of robots, PLCs and HMI). These decision-making processes are often sequentially executed, but their consequences and their results affect all the other subsystems in development.

In the classical engineering point of view, after the completion of design, supply procurement and assembly, the real commissioning is completed. Conventionally, the starting of the test phase concerning the whole planned manufacturing system is not feasible before the fulfillment of the previous steps. As a consequence, a considerable number of design problems and faults is not detected until the first system start-up. This downstream redesign, due to test results, leads to expensive corrective measures with a short reaction time. Usually, these defect corrections,

since executed during the installation, commissioning and the early production phases are delayed incurring in increased costs for all the parties involved. According to [27], the commissioning time consumes up to 25% of the time available for plant engineering and construction and up to 15% is employed in solving bugs in the automation control software alone.

The solution to this problem presented in the Industry 4.0 vision is the Virtual Commissioning. During Virtual Commissioning, a digital duplicate of the manufacturing system is used to proceed with commissioning through simulation, without relying on the real components. The main goal of this procedure is the early detection and correction of design and programming errors.

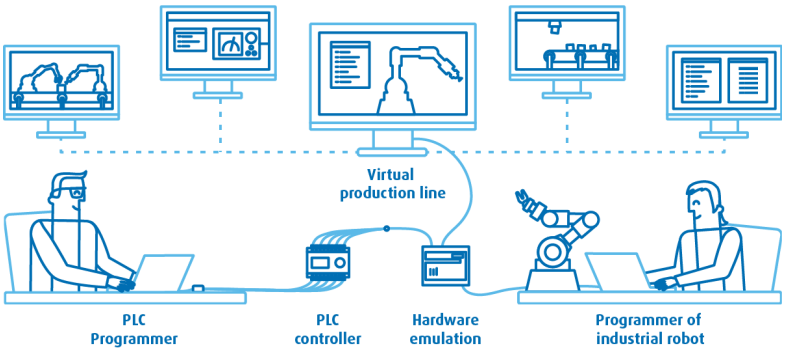


Figure 2.5: Virtual commissioning procedure

An experimental study performed in 2006 [27] shows the positive effects of Virtual Commissioning on the error rate and the debugging phase. In this experiment, two groups of engineers were compared on the design of an automation system development, one using virtual commissioning, while the other in a more classical way. The results showed a decrement of standard commissioning time by 75%, with enhanced software quality at the end of the project development. The fulcrum of this process is then the reproduction of an emulated model that can reproduce the real specimen behaviour, in a reasonable time amount.

The outcome of Virtual Commissioning process is the analysis and verification of all the design aspects of the project. The simplest behaviours that can be replicated and checked correspond to the geometry, kinematics and mechanical design, thanks to a 3D simulation of the assembly. With this opening phase simulation, the project designer is able to detect the mechanical responsibilities on geometrical planning errors.

The following step consists on the further verification of the control programs, the electrical network and the sensors efficiency. Emulating the series of I/O used, it is possible to detect deviations and bugs from the specified control software.

This verification of control can be arranged in four basic system configurations:

1. Real plant and real control system: the classical procedure for testing during real commissioning
2. Simulated plant and real control system: “Soft commissioning” often called “hardware in the loop” (HiL). The hardware controller is necessary in advance, while the mechanical elements are emulated. In simpler terms, the PLC is the real hardware controller. An improvement is the exploitation of an emulated PLC, a technique called "software in the loop (SiL)" simulation. Therefore, no hardware PLC is required.
3. Real plant and simulated control system (“Reality in the loop”)
4. Simulated plant and simulated control system

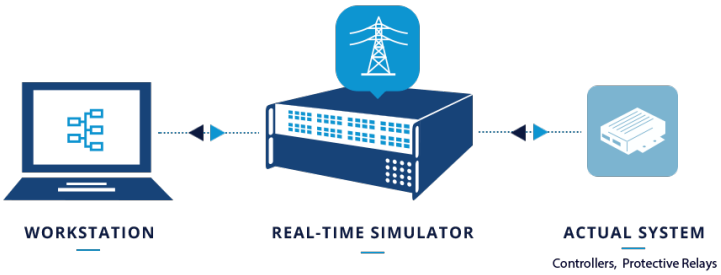


Figure 2.6: Hardware-in-the-Loop architecture

This last point represent the virtual commissioning and it is the combination of the 3D simulation and the control logic that leads to an integrated simulation environment that is capable of checking the feasibility of all complete functional chains from control programs through sensors, actuators and drives onto the mechanical movements.

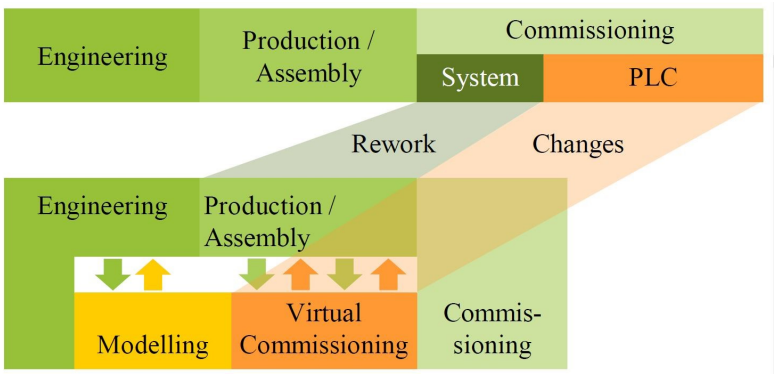


Figure 2.7: Comparison between Classical engineering and Virtual Commissioning

Fig. 2.8 shows the main differences between the classical engineering process and the design that exploits the virtual commissioning. The first step is the definition of the model of the production system. With respect to the classical sequence, the virtual commissioning allows the simultaneous development of the several areas, saving time and costs by avoiding the rework and changes with the optimization loops outlined by the arrows in figure.

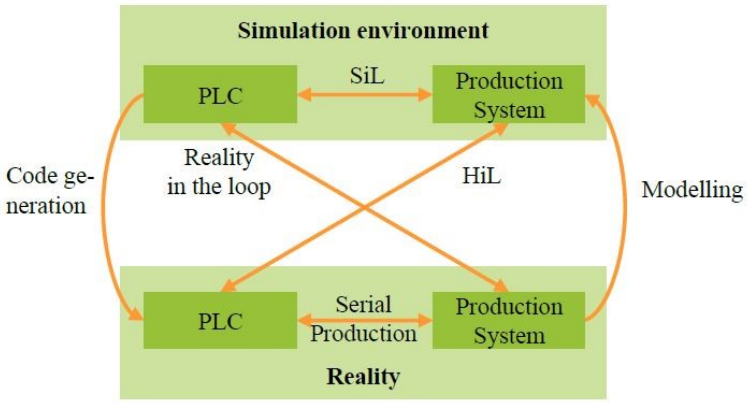


Figure 2.8: Architecture of the several concepts

The implementation of Virtual Commissioning in the actual engineering processes is hard for various reasons. The principal concerns are related to the high implementation costs at the beginning as well as the considerable complexity of the modelling of the reality. An important characteristics to be evaluated is the level of abstraction regarding the simulation model. In facts, increasing the level of accuracy of the simulation, the better the results at cost of exponentially computing requirements. Several conditions and requirements can be controlled, such as the collision check for the different configurations and bad designs. With Virtual Commissioning, the designers can verify the planned cycle time of the production system. Furthermore, critical and limit scenarios can be investigated without costly crashes and possible productive downtime [28] .

Chapter 3

Kinematics and Dynamics of the Gantry stacker

In order to better understand the behaviour of the gantry stacker, a kinematics and dynamics analysis is developed in this chapter. Starting from the mechanical design and specifics, the end-effector motion is developed with the help of MatLab software and its Robotic Toolbox add-in.

3.1 Direct Kinematics

The first step of this study is the development of a direct kinematic model of the robot. Typically, a manipulator is a structure composed by several rigid bodies, called links, which are interconnected through kinematic couples, also known as joints, constituting a kinematic chain. The joints are the fundamental elements that permits relative movement between links. There exists two principal types of joints: prismatic joints which permit relative translation and revolute joints which allow relative rotation. Each joint has a parent and a child, so forming the kinematic chain. The number of joints n in a tree structure represents also the degree of freedom of the manipulator. A robot configuration is a specified subset of values for the joint variables, resulting in a defined pose of the manipulator, which is the exact geometrical description of all the arms.

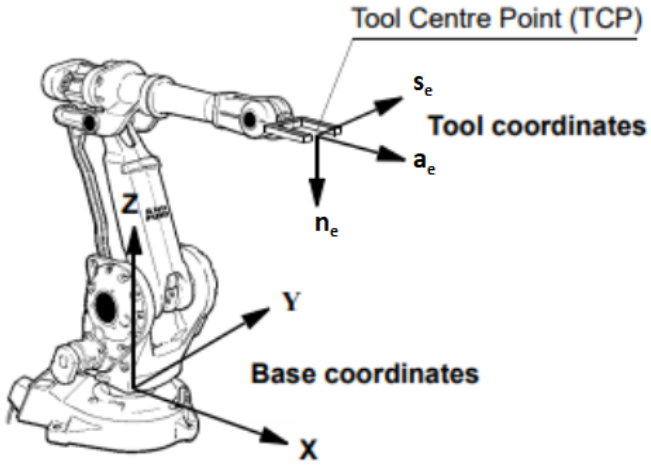


Figure 3.1: Robot reference frame

The first extreme of the kinematic chain is defined by the base, which defines also the principal reference frame O_b-$x_b y_b z_b$. The last extreme, instead, is the end-effector, which corresponds to the final tool of the manipulator such as a gripper or a drill. According to this final rigid body, the end-effector reference frame is defined with origin in the tool centre point (TCP) and the three unit vectors are found through a homogeneous transformation matrix that relates the end-effector reference frame O_e-$x_e y_e z_e$ with the base reference frame O_b-$x_b y_b z_b$ defined in the following way [29]:

$$\mathbf{T}_e^b(\mathbf{q}) = \begin{bmatrix} \vec{\mathbf{n}}_e^b(\mathbf{q}) & \vec{\mathbf{s}}_e^b(q) & \vec{\mathbf{a}}_e^b(q) & \vec{\mathbf{p}}_e^b(q) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where:

- $\mathbf{q}$ represents an $(n \times 1)$ vector containing the joint variables
- $\vec{\mathbf{p}}_e$ represents the position vector of the TCP with respect to the base reference frame
- $\vec{\mathbf{a}}_e$ represents the arbitrary unit vector chosen in the approach direction of the tool
- $\vec{\mathbf{s}}_e$ represents the arbitrary unit vector perpendicular to $\vec{\mathbf{a}}_e$ chosen in the sliding direction of the end-effector tool
- $\vec{\mathbf{n}}_e$ represents the normal vector and it is defined in order to make the triplet of versors orthonormal.

For complex structures, the geometric evaluation of this homogeneous matrix is not immediate, so a systematic procedure is preferable. It is possible, in facts, to use a recursive approach which considers the transformations between the several rigid bodies done by the joints.

Denavit-Hartemberg convention

Considering an open chain manipulator, $n+1$ links and n joints are defined. The interconnection is of the parent-child type, where each joint links the previous rigid body with the next link permitting a motion defined by the joint variable q_i. Therefore, reference frames of the links are related to the homogeneous matrices of the previous manipulator arms, leading to the equation 3.2 that defines the reference frame of the n-th link according to the initial reference frame.

$$\mathbf{T}_n^0(\mathbf{q}) = \mathbf{A}_1^0(q_1)\mathbf{A}_2^1(q_2)\dots\mathbf{A}_n^{n-1}(q_n) \tag{3.2}$$

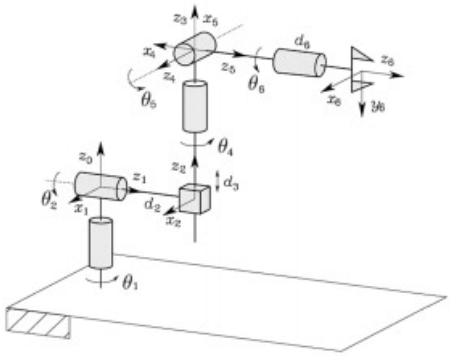
This recursive method requires a general and systematic procedure to determine position and orientation between two consecutive arms. Typically, the Denavit-Hartemberg convention is adopted. The DH convention defines the choice of the reference frame of each rigid body and the exploitation of the geometric parameters of arms, according to the joint variables $\mathbf{q}$. The reference frame of the i-th link is determined through the following rules

- selection of z_i axis lying on the axis of joint $i+1$. The reference frame n has not the z axis automatically defined since there is no other joint, but it is common sense, when the final joint is revolute, to align z_n with z_{n-1};

- reference frame origin O_i is found at the intersection between z_i axis and the normal direction between z_i and z_{i-1}. In the reference frame 0, only the z_0 axis is defined, then the origin O_0 can be chosen arbitrarily;

- x_i axis direction along the normal between z_{i-1} axis and z_i axis with positive orientation towards joint $i+1$. The convention is not univocal in the case of intersecting consecutive axes and so x_i axis direction is arbitrary. Same holds for x_0 since there is no previous joint;

- y_i axis direction is defined to complete an orthonormal basis triplet of vectors.



D-H parameters

Link	a_i	α_1	d_i	θ_i
1	0	$-\pi/2$	0	$\boldsymbol{\theta_1}$
2	0	$\pi/2$	d_2	$\boldsymbol{\theta_2}$
3	0	0	$\boldsymbol{d_3}$	0
4	0	$-\pi/2$	0	$\boldsymbol{\theta_4}$
5	0	$\pi/2$	0	$\boldsymbol{\theta_5}$
6	0	0	d_6	$\boldsymbol{\theta_6}$

Figure 3.2: Denavit-Hartemberg convention

When the reference frame definition is not arbitrary, the indetermination can be exploited to simplify the procedure. Once the orthonormal bases are chosen, the geometrical parameters of each arm is defined, according to the DH parameters [30].

- a_i as the distance along x_i between O_i and O_i'
- d_i as the coordinate on z_{i-1} of O_i'
- α_i as the angle between z_{i-1} axis and z_i axis around the x_i axis in counter-clockwise direction
- θ_i as the angle between x_{i-1} axis and x_i axis around the z_{i-1} axis in counter-clockwise direction

Two of these parameters, a_i and α_i, are constant and consider only the connection geometry between two consecutive joints. Depending on the nature on the joint, one of the other two parameter is related to the joint variable: if the i-th joint is prismatic, the variable is d_i, else the variable becomes θ_i in the presence of a revolute joint. According to DH convention, the homogeneous matrix that relates the reference frames of two consecutive links is obtained in the equation 3.3.

$$\mathbf{A}_i^{i-1}(q_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The complete transformation can be done through the reiteration of the process for all the joint in the rigid body tree. The final homogeneous matrix is evaluated through equation 3.2.

Direct Kinematics of the Gantry

The machinery under investigation is a Gantry-stacker adopted to the pick and place of glass sheets. The dimensions of these plates can range from 1 m up to 9 m of length and with a thickness of few centimeters with the possibility of handling up to four adjacent lites simultaneously. The total weight that is transported can rise up to over a ton. For these reasons, a simple anthropomorphic manipulator is not suitable for the task and a structure able to sustain and relieve the load is required. Hence, a gantry structure with several ribbons is a preferable solution. Being a gantry stacker, the kinematic chain can be distinguished in a Cartesian structure, composed of three prismatic joints and a semi-wrist part, consisting of two revolute joints.

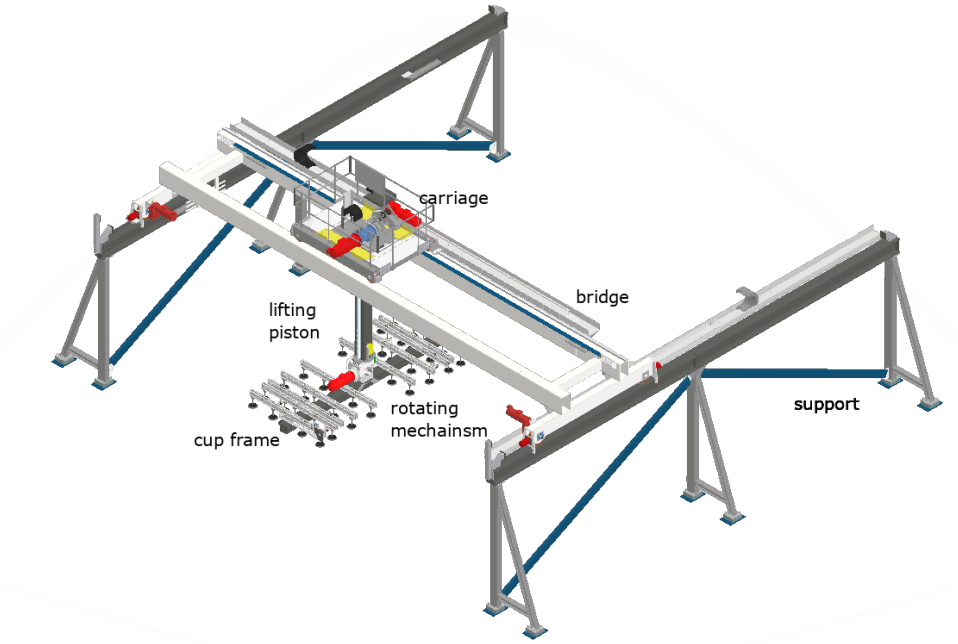


Figure 3.3: Gantry stacker

According to picture 5.17 it is possible to identify the following rigid bodies defining the kinematic chain:

- bridge supports, which act as guides for the translational movement along x axis. Their reference frame is define as base and it is the root link for our manipulator structure;

- bridge, a wheeled beam able to move over the support guides;

- carriage, a structure able to translate along y axis on the hollow guides internal to the bridge beam. This movement is performed through a catenary;
- lifting piston, mechanism able to perform z translation attached to the carriage;
- rotating structure, unit composed of a motor and revolving coupling that allows rotation along the z axis of the frame;
- cup frame, it is the end-effector of the robot. It is a structure containing 2D grid of vacuum cups capable of picking glass sheets thanks to negative pressure gradients. On this gripper tool are also present different sensors, such as sonars and proximity sensors.

Given this preliminary geometrical description and the mechanical drawing with the exact dimensions, the direct kinematic expression that links the 5 joint variable $\vec{\mathbf{q}}$ and the pose of the gantry can be computed, following the Denavit-hartemberg convention.

i-th link	a_i	d_i	α_i	θ_i
1 - Bridge	a_1	d_x	$-90°$	$+90°$
2 - Carriage	a_2	d_y	$-90°$	$+90°$
3 - Lifting piston	0	d_z	0	0
4 - Rotating structure	a_4	0	$+90°$	θ_z
5 - cup frame	a_5	0	$-90°$	θ_x

Once the definition of the reference frame for each robot link is developed, resulting in the Denavit-Hartemberg parameters expressed in the table and the homogeneous transformation matrices described in eq. 3.4.

$$\mathbf{T}_0^5 = \begin{bmatrix} \mathrm{s}_4 c_5 & \mathrm{c}_4 & \mathrm{s}_4 s_5 & \mathrm{d}_x \\ \mathrm{c}_4 c_5 & \text{-}\mathrm{s}_4 & \mathrm{c}_4 s_5 & \mathrm{d}_y \\ \text{-}\mathrm{s}_5 & 0 & \mathrm{c}_5 & \mathrm{d}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

Joint space and operative space

As shown before, the direct kinematic equation allows to express position and orientation of the end-effector reference frame as a function of the joint variables. During an automated operation, the assignment of position and orientation of the tool in function of the time is a key factor. This pose definition is simpler for the position, while is not immediate for the orientation since it is defined according to the three versors ($\boldsymbol{n}_e$, $\boldsymbol{s}_e$, $\boldsymbol{a}_e$) that must satisfy the orthonormality condition. In summary, it is possible to identify the tool position with a minimum number

of coordinates defined by the geometric of the system and the orientation can be described by a minimal representation of the rotation with respect to base reference frame. The final pose can be described with a $(m \times 1)$ vector with $m \leq 6$ which is defined according to eq. 3.5.

$$\boldsymbol{x}_e = \begin{bmatrix} \boldsymbol{p}_e \\ \boldsymbol{\phi}_e \end{bmatrix} \tag{3.5}$$

This description is defined in the operative space. In order to link this vector with the joint variable vector $\boldsymbol{q}$ described in the configuration space, it is fundamental to define the direct kinematic equation in the alternative form expressed in eq. 3.6.

$$\boldsymbol{x}_e = \boldsymbol{k}(\boldsymbol{q}) \tag{3.6}$$

This is a vectorial function in general non-linear able to evaluate the end-effector position in operative space starting from the joint configuration.

On the contrary, the inverse kinematic problem requires the definition of the joint variables given the end-effector pose. This problem plays a key role in the translation of the motion specifics, typically starting from a trajectory in the operative space and resulting in the corresponding joint motions that produce the desired movement. In general, the inverse kinematic problem is more complex since the non-linearity characteristic of the equations. Typically this problem can have multiple solutions due to the redundancy or, on the contrary do not have admissible solutions because the pose does not lie in the dexterity working region of the manipulator. The determination of closed-form solutions requires geometric and algebraic intuition to define the significant equations and couplings of the structure to reduce the number of joint variables related to an operative coordinate. For complex case, the search for analytic solutions can be solved with numerical algorithms that exploit the Jacobian matrices of the system.

Robot configuration and pose with `MatLab Robotic Toolbox`

In this subsection, a simple robot model is built and developed using the Robotic MatLab Toolbox [31]. Typical robot models designed with this application are the embryos of the digital twin concept, where simple feasibility checks can be done on the preliminary aspects of the automation mechanism. This robot representations simulate the kinematic and dynamic properties of manipulator robots and other rigid body systems. A kinematic chain is defined in MatLab as a `rigidBodyTree` object containing `rigidBodies` linked by `rigidbodyJoints` elements with their joint transformations and inertial properties [31]. The first step consists in the definition of a proper kinematic chain where, starting from the robot base, all the bodies are connected, up to the end-effector, thanks to the function `addBody`.

Each rigid body is attached by a joint, whose goal is the definition of the motion that the arm can perform with respect to its parent. Characteristics of

this `rigidbodyJoints` are the attachment point between the child and the parent kinetic links, but also the nature of the degree of motion defining three basic types of joint: fixed, revolute, and prismatic. In addition,Each joint is characterised by an axis of motion defined in its properties. The joint axis is represented by a 3D unit vector that, depending on the nature of the joint can define the axis of rotation (for revolute joints) or the axis of translation (for prismatic joints) [31].

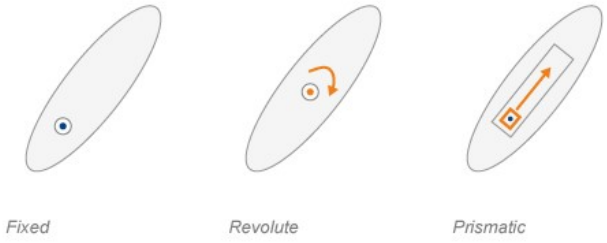


Figure 3.4: Definition of joints according to MatLab Robotic Toolbox

The `HomePosition` represents the starting position for each specific joint, which is a point within the position limits. The configuration composed by all the joints in their initial position can be recalled as `homeConfiguration`.

Joints also have properties that define the fixed transformation between parent and children body coordinate frames. In MatLab, these properties can be modified thanks to the `setFixedTransform` function. Depending on the definition of the kinetic chain and its transformation parameters, the joint transformation is defined according to the child or the parent body, either the `JointToParentTransform` or `ChildToJointTransform` property is set using this method. The image 3.5 shows the previous properties according to the kinematic chain [31].

Another important element of this multi-body model is represented by the collision object. This geometrical feature is the 3D space occupied by the body. According to the complexity of the modelling, it is possible to define collision with simple geometric figures (parallelepiped, sphere, cylinder), but there is also the possibility of adopting collision primitives from a convex collision mesh imported with an .STL or .DAE file. Even in this case, the specification of a rigid transform matrix is optimal to correctly pose the body geometry in the reference frame.

This MatLab script exploits the graphical visualization of the several elements of the robots and so it is possible to analyse and control the preliminary configurations that the manipulator should reach. These joint configurations $\boldsymbol{q}$ can be previously set with the code, but also visually redefined, thanks to an interactive GUI. With

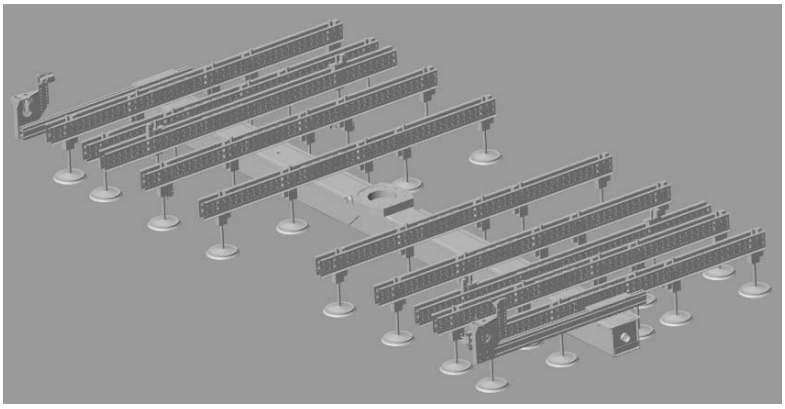


Figure 3.5: Cup frame representation in .STL file

this method, it is possible to transform the kinetic chain by moving the end-effector of the robot, but also by modifying the joint variable of each singular axes independently, by selecting the correct marker body to be translated or rotated as shown in fig. 3.6 and 3.7.

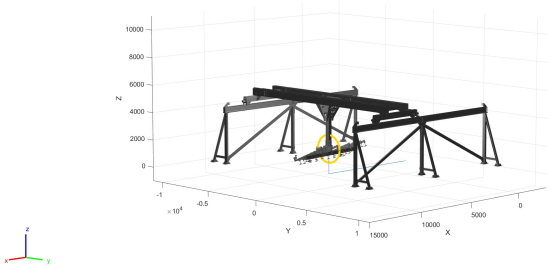


Figure 3.6: Single axis movement performed on the multi-body model

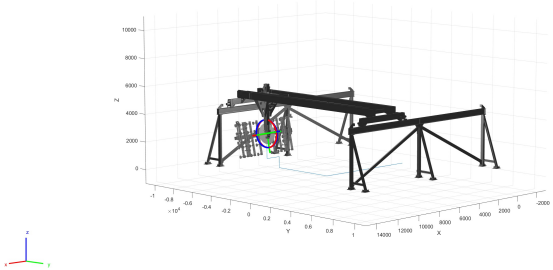


Figure 3.7: Movement of the end-effector performed on the multi-body model

With this initial multi-body, it is possible to confront also the principal waypoints of the trajectories performed by preliminary behavioural analysis and perform the collision check with the obstacle that occupy the workspace of the gantry stacker. Thanks to the `Inverse Kinematic Designer` app, it is possible to design an inverse kinematics solver for the robot model previously developed. In facts, by defining the parameters of the inverse kinematics solver, adding the constraints relative to the obstacles and the requested job task to be performed, the feasibility of the desired behaviour is controlled. In principles, the sequence performed in this environment are defined as following:

- Import of the rigid body tree model previously built.
- Adjust inverse kinematics solvers and import of the collision bodies of the other elements to be encountered.
- eventual definition of the constraints that affect the robot (locking a certain joint movement and boundary definition in the space for the end-effector).
- Definition of the joint configurations to be analysed and the relative waypoints in the trajectory.
- Export of the results and the review in the MatLab Workspace.

As a result, in this programming environment, it is possible to check which trajectories are feasible and which are not available due to collisions with the environment elements, but also against the links of the robot (self-collision), as shown in figure 3.8.

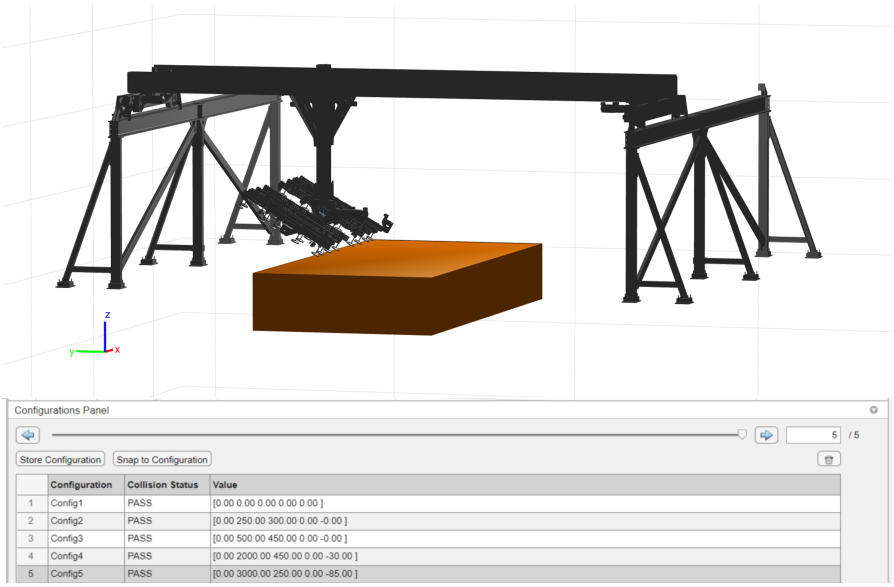


Figure 3.8: Collision check in the Inverse Kinematic Designer

To further develop the digital twin principles, once defined the model for the definition of the trajectory waypoints and their relative collision checks, it is fundamental to analyse the kinematics and dynamics of the system.

Jacobian matrix and differential kinematics

The objective of the differential kinematic is to determine the relationships between the joint velocities and the linear and angular velocity of the end-effector. This

principle is translated into eq. 3.7:

$$\begin{aligned} \dot{\boldsymbol{p}}_e &= \boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{\omega}}_e &= \boldsymbol{J}_O(\boldsymbol{q})\dot{\boldsymbol{q}} \end{aligned} \tag{3.7}$$

where $\dot{\boldsymbol{p}}_e$ is the linear velocity vector and $\dot{\boldsymbol{\omega}}_e$ is the angular velocity of the end-effector. This set of equation can be generalised into eq. 3.8.

$$\boldsymbol{v}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \boldsymbol{\omega}_e \end{bmatrix} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.8}$$

where $\boldsymbol{J}$ is the geometric Jacobian of dimensions $(6 \times n)$ of the manipulator. The starting point is the analysis of the velocity of a generic robot arm [29]. According to the Denavit-Hartemberg convention previously adopted, the link i connects the joints i to $i+1$. Being $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_i$ the position vectors for the origins of the reference frames for the given joints, the vectorial sum that relates the positions of two consecutive arms is described in eq. 3.9.

$$\boldsymbol{p}_i = \boldsymbol{p}_{i-1} + \boldsymbol{R}_{i-1}\boldsymbol{r}_{i-1,i}^{i-1} \tag{3.9}$$

According to the derivation of the rotation matrix, it is possible to evaluate the derivative $\dot{\boldsymbol{p}}_i$ which is described in eq. 3.10.

$$\dot{\boldsymbol{p}}_i = \dot{\boldsymbol{p}}_{i-1} + \boldsymbol{R}_{i-1}\dot{\boldsymbol{r}}_{i-1,i}^{i-1} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{r}_{i-1,i} = \dot{\boldsymbol{p}}_{i-1} + \boldsymbol{v}_{i-1,i} + \boldsymbol{\omega}_{i-1} \times \boldsymbol{r}_{i-1,i} \tag{3.10}$$

From this equation, it is possible to consider the linear velocity of arm i as a function of the linear and angular velocity of $i-1$ link. $v_{i-1,i}$ represents the velocity of the origin of the reference frame i with respect to the origin of the reference frame $i-1$. This general relationship for the speed of a robotic arm can be specified on the basis on the joint type that links the two rigid bodies. In the case of a prismatic joint, the reference frame orientation does not change, thus $\boldsymbol{\omega}_{i-1,i} = 0$. In addition, the linear velocity can e defined as eq. 3.11

$$\boldsymbol{v}_{i-1,i} = \dot{d}_i \boldsymbol{z}_{i-1} \tag{3.11}$$

where $\boldsymbol{z}_{i-1}$ is the unit vector of the axis of joint i. As a consequence, the velocity of the arm moved by the prismatic joint is defined by eq. 3.12.

$$\begin{aligned} \boldsymbol{\omega}_i &= \boldsymbol{\omega}_{i-1} \\ \dot{\boldsymbol{p}}_i &= \dot{\boldsymbol{p}}_{i-1} + \dot{d}_i \boldsymbol{z}_{i-1,i} + \boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i} \end{aligned} \tag{3.12}$$

On the other hand, the revolving joint links the orientations of the two reference frames with the joint variable, according to eq. 3.13.

$$\boldsymbol{\omega}_{i-1,i} = \dot{\theta}\boldsymbol{z}_{i-1,i} \tag{3.13}$$

The difference in linear velocity between frames derives from the induced motion of joint $i-1$ on the reference frame i. The expressions for the linear and angular velocity are defined according to eq. 3.14.

$$\begin{aligned} \boldsymbol{\omega}_i &= \boldsymbol{\omega}_{i-1} + \dot{\theta}_i \boldsymbol{z}_{i-1} \\ \dot{\boldsymbol{p}}_i &= \dot{\boldsymbol{p}}_{i-1} + \boldsymbol{\omega}_i \times \boldsymbol{r}_{i-1,i} \end{aligned} \tag{3.14}$$

Evaluation of the Jacobian

In order to evaluate the Jacobian is preferable to analyse separately the linear velocity from the angular velocity [29]. The linear velocity contribution can be stated as the time derivative of the pose $\boldsymbol{p}_e(\boldsymbol{q})$, defined as eq. **??**.

$$\dot{\boldsymbol{p}}_e = \sum_{i=1}^{n} \frac{\partial \boldsymbol{p}_e}{\partial q_i} \dot{q}_i = \sum_{i=1}^{n} \boldsymbol{j}_{P_i} \dot{q}_i \tag{3.15}$$

Each term represents the velocity contribution of the i joint to the velocity of the end effector, calculated when the other joints are locked. Similarly to the definition of the arm velocity, also the Jacobian contribution can be distinguished according to the type of joint. In the case of a prismatic joint, the variable q is represented by d and, consequently, the Jacobian contribution is described by eq. 3.16.

$$\boldsymbol{j}_{P_i} = \boldsymbol{z}_{i-1} \tag{3.16}$$

On another perspective, the joint variable for a revolute joint is defined by θ and its contribution on the linear velocity can be defined as eq. 3.17

$$\boldsymbol{j}_{P_i} = \boldsymbol{z}_{i-1} \times (\boldsymbol{p}_e - \boldsymbol{p}_{i-1}) \tag{3.17}$$

In addition, the angular velocity contribution is defined as eq. 3.18.

$$\boldsymbol{\omega}_n = \sum_{i=1}^{n} \boldsymbol{\omega_{i-1,i}} = \sum_{i=1}^{n} \boldsymbol{j}_{O_i} \dot{\boldsymbol{q}}_{\boldsymbol{i}} \tag{3.18}$$

It is fundamental to stress that in the case of a prismatic joint, its relative component of the jacobian matrix is null, due to pure translation, while considering a revolute joint, the contribution corresponds to the $\boldsymbol{z}_{i-1}$ axis. The final Jacobian matrix can be expressed as the separate contribution of $\boldsymbol{j}_{P_i}$ and $\boldsymbol{j}_{O_i}$ described in eq. 3.19.

$$\begin{aligned} \boldsymbol{j}_{P_i} &= \begin{cases} \boldsymbol{z}_{i-1} & \text{for a prismatic joint} \\ \boldsymbol{z}_{j-1} \times (\boldsymbol{p}_e - \boldsymbol{p}_{i-1}) & \text{for a revolute joint} \end{cases} \\ \boldsymbol{j}_{O_j} &= \begin{cases} \boldsymbol{0} & \text{for a prismatic joint} \\ \boldsymbol{z}_{i-1} & \text{for a revolute joint} \end{cases} \end{aligned} \tag{3.19}$$

In this way, it is possible to calculate the Jacobian directly with the kinematic relations because the vectors that appears in eq. 3.19 are functions of the joint variables $\boldsymbol{q}$.

To resume, the Jacobian matrix represent an effective instrument to evaluate the translational and rotational velocities of each point of the kinematic chain, with particular focus on the end effector. This matrix is the most important element in the kinematic analysis thanks to which it is possible to perform inverse kinematics algorithms, but is also part of the solution to better understand the dynamics of the manipulator.

3.2 Dynamics of the manipulator

The dynamic model of the manipulator is the instrument able to translate the joint actuations into the robotic structure motion. This model for the derivation of the motion law of the manipulator arm can be implemented through several methods. In this section the Lagrange's formulation is described.

3.2.1 Lagrange formulation

The Lagrangian of a mechanical system is defined as 3.20.

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{3.20}$$

where $\mathcal{T}$ is the kinetic energy and $\mathcal{U}$ is the potential energy of the system. This equation is dependent only on the generalised coordinates q_i that describe the position of all the mechanical elements of the system independently on the reference system. The Lagrange equations are then defined as in eq. 3.21:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \xi_i \tag{3.21}$$

where ξ_i is the generalised force associated to the generalised coordinate q_i. This equation can be rewritten in matrix form as in eq. 3.22

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}\right)^T = \boldsymbol{\xi} \tag{3.22}$$

The vector $\boldsymbol{\xi}$ comprehends all the non-conservative forces, considering the actuation torques generated at the joints, the friction forces between moving parts and also the forces derived from the interaction between the end-effector and the environment. As a first step, it is fundamental to evaluate the kinetic energy of the manipulator

given by the sum of the contributions of each arm in motion and the contributions of the actuators, according to eq. 3.23.

$$\mathcal{T} = \sum_{i=1}^{n} (\mathcal{T}_{l_i} + \mathcal{T}_{m_i}) \tag{3.23}$$

The kinetic energy contribution of each arm can be evaluated as eq 3.24:

$$\mathcal{T}_{l_i} = \frac{1}{2} \int_{V_{l_i}} \dot{\boldsymbol{p}}_{\boldsymbol{i}}^{*T} \dot{\boldsymbol{p}}_{\boldsymbol{i}}^{*} \rho dV \tag{3.24}$$

where the rigid body is subdivided in infinitesimal volume element dV with definite density ρ. $\dot{\boldsymbol{p}}_{\boldsymbol{i}}^{*}$ represents the linear velocity vector of dm particle. This velocity component can be expressed as the vector sum of the velocity of the centre of mass and a Coriolis term related to the rotation of the arm and the distance of the particle from the centre of rotation.

$$\dot{\boldsymbol{p}}_{\boldsymbol{i}}^{*} = \dot{\boldsymbol{p}}_{\boldsymbol{l_i}} + \boldsymbol{\omega_i} \times \boldsymbol{r_i} = \dot{\boldsymbol{p}}_{\boldsymbol{l_i}} + \boldsymbol{S}(\boldsymbol{w}_i)\boldsymbol{r_i} \tag{3.25}$$

The kinetic energy of the single arm can be re-expressed as 3.26,which is the decoupling of the translation and rotation contribution:

$$\mathcal{T}_{l_i} = \frac{1}{2} m_{l_i} \dot{\boldsymbol{p}}_{\boldsymbol{l_i}}^{T} \dot{\boldsymbol{p}}_{\boldsymbol{l_i}} + \frac{1}{2} \boldsymbol{\omega}_i^{T} \left(\int_{V_{l_i}} \boldsymbol{S}^{T}(\boldsymbol{r}_i) \boldsymbol{S}(\boldsymbol{r}_i) \rho dV \right) \boldsymbol{\omega}_i \tag{3.26}$$

The integral term can be substituted with the inertia tensor matrix relative to the centre of mass of the link, according to eq. 3.27.

$$\begin{aligned} \boldsymbol{I}_{l_i} &= \begin{bmatrix} \int (r_{iy}^2 + r_{iz}^2) \rho dV & -\int r_{ix} r_{iy} \rho dV & -\int r_{ix} r_{iz} \rho dV \\ * & \int (r_{ix}^2 + r_{iz}^2) \rho dV & -\int r_{iy} r_{iz} \rho dV \\ * & * & \int (r_{ix}^2 + r_{iy}^2) \rho dV \end{bmatrix} = \\ &= \begin{bmatrix} I_{l_i xx} & -I_{l_i xy} & -I_{l_i xz} \\ * & I_{l_i yy} & -I_{l_i yz} \\ * & * & I_{l_i zz} \end{bmatrix} \end{aligned} \tag{3.27}$$

This symmetrical inertia matrix is expressed according to the base reference frame O_b-$x_b y_b z_b$ and depends on the robot configuration. Considering the rotation matrix R_i that allows to express the angular velocity ω with a reference frame in accordance with i^{th} link, it is possible to define the inertia tensor with the same reference frame which results constant. This constant inertia tensor is defined $I_{l_i}^{i}$ and it is independent from the configuration. The total kinetic energy resulting from the translation and rotation contribution is then expressed according to eq. 3.28.

$$\mathcal{T}_{l_i} = \frac{1}{2} m_{l_i} \dot{\boldsymbol{p}}_{\boldsymbol{l_i}}^{T} \dot{\boldsymbol{p}}_{\boldsymbol{l_i}} + \frac{1}{2} \boldsymbol{\omega}_i^{T} \boldsymbol{R}_i \boldsymbol{I}_{l_i}^{i} \boldsymbol{R}_i^{T} \boldsymbol{\omega}_i \tag{3.28}$$

This equation must be then explicited according to the joint variables, in order to retrieve the kinetic energy from the generalised coordinates. Considering the previous adopted geometrical method for the Jacobian, it is possible to obtain the following relation for $\dot{\boldsymbol{p}}_{\boldsymbol{l_i}}$ and $\boldsymbol{\omega}_{\boldsymbol{l_i}}$.

$$\begin{aligned} \dot{\boldsymbol{p}}_{\boldsymbol{l_i}} &= \boldsymbol{j}_{\boldsymbol{P_1}}\dot{q}_1 + \cdots + \boldsymbol{j}_{\boldsymbol{P_i}}\dot{q}_i = \boldsymbol{J}_P^{(l_i)}\dot{\boldsymbol{q}} \\ \boldsymbol{\omega}_{\boldsymbol{i}} &= \boldsymbol{j}_{\boldsymbol{O_1}}\dot{q}_1 + \cdots + \boldsymbol{j}_{\boldsymbol{O_i}}\dot{q}_i = \boldsymbol{J}_O^{(l_i)}\dot{\boldsymbol{q}} \end{aligned} \tag{3.29}$$

With reference to eq. **??** the kinetic energy of the arm can be finally written as eq. 3.30.

$$\mathcal{T}_{l_i} = \frac{1}{2} m_{l_i} \dot{\boldsymbol{q}}^T \boldsymbol{J}_P^{(l_i)T} \boldsymbol{J}_P^{(l_i)} \dot{\boldsymbol{q}} + \frac{1}{2} \dot{\boldsymbol{q}}^T \boldsymbol{J}_O^{(l_i)T} \boldsymbol{R}_i \boldsymbol{I}_{l_i}^i \boldsymbol{R}_i^T \boldsymbol{J}_O^{(l_i)} \dot{\boldsymbol{q}} \tag{3.30}$$

The second contribution is given by the kinetic energy of the drive motor at the i^th joint and the procedure to evaluate it with the joint variables is similar. Considering electrical motors (able to actuate both rotation and prismatic joints through proper mechanisms), the effective contribution is given by the rotor, since the stator contribution is accounted in $\mathcal{T}_{l_i}$, according to eq. 3.31:

$$\mathcal{T}_{m_i} = \frac{1}{2} m_{m_i} \dot{\boldsymbol{p}}_{m_i}^T \dot{\boldsymbol{p}}_{m_i} + \frac{1}{2} \boldsymbol{\omega}_{m_i}^T \boldsymbol{I}_{m_i} \boldsymbol{\omega}_{m_i} \tag{3.31}$$

where m_{m_i} represents the rotor mass, $\dot{\boldsymbol{p}}_{m_i}$ represents the linear velocity of the centre of mass of the rotor, $\boldsymbol{I}_{m_i}$ represents the inertia tensor of the rotor relative to the centre of mass and $\boldsymbol{\omega}_{m_i}$ represents the angular velocity of the rotor. Indicating with θ_{m_i} the angular position of the rotor, in the rigid transmission hypothesis, the joint velocity $\dot{q}_i$ can be written as eq. 3.32:

$$\dot{q}_i = \frac{\dot{\theta}_{m_i}}{k_{ri}} \tag{3.32}$$

where k_{ri} is the mechanical transmission ratio. The total angular velocity of the rotor is the sum of the angular velocity of the previous adjacent arm and the angular contribution of the motor, as defined in eq. 3.33.

$$\boldsymbol{\omega}_{m_i} = \boldsymbol{\omega}_{i-1} + k_{ri}\dot{q}_i \boldsymbol{z}_{m_i} \tag{3.33}$$

even in this case, it is necessary to explicit the kinetic energy of the rotor as a function of the joint variables $\boldsymbol{q}$ and so it is necessary to take into account the Jacobian of the rotor as stated in eq. 3.34.

$$\begin{aligned} \dot{\boldsymbol{p}}_{\boldsymbol{m_i}} &= \boldsymbol{J}_P^{(m_i)}\dot{\boldsymbol{q}} \\ \boldsymbol{\omega}_{\boldsymbol{m_i}} &= \boldsymbol{J}_O^{(m_i)}\dot{\boldsymbol{q}} \end{aligned} \tag{3.34}$$

The two jacobian matrices related to m_i differs from the previous one related to the manipulator arm and their columns are defined according to eq. 3.35.

$$\begin{aligned}\boldsymbol{j}_{P_j}^{(m_i)} &= \begin{cases}\boldsymbol{z}_{j-1} & \text{for a prismatic joint}\\ \boldsymbol{z}_{j-1}\times(\boldsymbol{p}_{m_i}-\boldsymbol{p}_{j-1}) & \text{for a revolute joint}\end{cases}\\ \boldsymbol{j}_{O_j}^{(m_i)} &= \begin{cases}\boldsymbol{j}_{O_j}^{(l_i)} & \text{for a prismatic joint}\\ k_{ri}\boldsymbol{z}_{m_i} & \text{for a revolute joint}\end{cases}\end{aligned} \tag{3.35}$$

Finally, the kinetic energy of the i^{th} rotor can be written as eq. 3.36.

$$\mathcal{T}_{m_i} = \frac{1}{2}m_{m_i}\dot{\boldsymbol{q}}^T\boldsymbol{J}_P^{(m_i)T}\boldsymbol{J}_P^{(m_i)}\dot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{J}_O^{(m_i)T}\boldsymbol{R}_{m_i}\boldsymbol{I}_{m_i}^i\boldsymbol{R}_{m_i}^T\boldsymbol{J}_O^{(m_i)}\dot{\boldsymbol{q}} \tag{3.36}$$

In conclusion, the total kinetic energy can be defined as the total sum of the contributions of the links and the actuators, according to eq. 3.37.

$$\mathcal{T} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}b_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j = \frac{1}{2}\dot{\boldsymbol{q}}^T\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.37}$$

Where the matrix $\boldsymbol{B}(\boldsymbol{q})$ is the inertia matrix $(n\times n)$ that is configuration-dependent, symmetric and definitively-positive and can be evaluated as eq. 3.38.

$$\begin{aligned}\boldsymbol{B}(\boldsymbol{q}) = \sum_{i=1}^{n}\Big(&m_{l_i}\boldsymbol{J}_P^{(l_i)T}\boldsymbol{J}_P^{(l_i)} + \boldsymbol{J}_O^{(l_i)T}\boldsymbol{R}_i\boldsymbol{I}_{l_i}^i\boldsymbol{R}_i^T\boldsymbol{J}_O^{(l_i)}\\ &+ m_{m_i}\boldsymbol{J}_P^{(m_i)T}\boldsymbol{J}_P^{(m_i)} + \boldsymbol{J}_O^{(m_i)T}\boldsymbol{R}_{m_i}\boldsymbol{I}_{m_i}^i\boldsymbol{R}_{m_i}^T\boldsymbol{J}_O^{(m_i)}\Big)\end{aligned} \tag{3.38}$$

Recalling eq. 3.20, the second element to be investigated is $\mathcal{U}$, which is the total potential energy given by the sum of all the robot links and rotors as stated in eq. 3.39.

$$\mathcal{U} = \sum_{i=1}^{n}(\mathcal{U}_{l_i} + \mathcal{U}_{m_i}) \tag{3.39}$$

Under the assumption of the rigid body, and thus neglecting the elastic contribution, the potential neergy is given by the gravitational forces and so it is defined according to eq. 3.40:

$$\mathcal{U}_{l_i} = -\int_{V_{l_i}}\boldsymbol{g}_0^T\boldsymbol{p}_i^*\rho dV = -m_{l_i}\boldsymbol{g}_0^T\boldsymbol{p}_{l_i} \tag{3.40}$$

where $\boldsymbol{g}_0$ is the gravitational acceleration vector referred to the base reference frame and it considers the coordinate of the centre of mass of the i^{th} arm. Regarding the rotor contribution, it can be evaluated analogously as written in eq. 3.41.

$$\mathcal{U}_{m_i} = -m_{m_i}\boldsymbol{g}_0^T\boldsymbol{p}_{m_i} \tag{3.41}$$

Being the total potential energy only function of the vectors $\boldsymbol{p}_{l_i}$ and $\boldsymbol{p}_{m_i}$, it results that this component of the Lagrangian is dependent only on the joint variable $\boldsymbol{q}$ and not on the joint velocity $\dot{\boldsymbol{q}}$. From the definition of the total kinetic energy and potential energy, it is possible to derive according to eq. 3.21, where the partial derivative with respect to $\dot{\boldsymbol{q}}$ affects only the kinetic energy resulting in eq. 3.42.

$$\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.42}$$

and then deriving with respect to the time the first term of the Lagrangian equation can be developed into eq. 3.43.

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{B}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.43}$$

The second term, related to the partial derivative with respect to the joint variable, affects both the Lagrangian contributions, resulting in eq. 3.44.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} = \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{q}}\left(\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}\right)\right)^T - \left(\frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T \tag{3.44}$$

The final representation in matrix form of the Lagrangian formulation can be described according to eq. 3.45:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\xi} \tag{3.45}$$

where

$$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{B}}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{q}}\left(\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}\right)\right)^T + \left(\frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T \tag{3.46}$$

Other important observations can be done considering the single row of this matrix equation, since the total potential energy $\mathcal{U}$ does not depend on $\dot{\boldsymbol{q}}$. Hence:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) = \frac{d}{dt}\left(\frac{\partial \mathcal{T}}{\partial \dot{q}_i}\right) &= \sum_{j=1}^{n} b_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n} \frac{db_{ij}(\boldsymbol{q})}{dt}\dot{q}_j \\ &= \sum_{j=1}^{n} b_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{db_{jk}(\boldsymbol{q})}{dq_i}\dot{q}_j\dot{q}_k \end{aligned} \tag{3.47}$$

In a similar way, also the second term of the Lagrangian formulation can be evaluated observing the single row elements and it results in eq. 3.48.

$$\frac{\partial \mathcal{T}}{\partial q_i} = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial b_{jk}(\boldsymbol{q})}{\partial q_i}\dot{q}_j\dot{q}_k \tag{3.48}$$

By considering also the single row contribution for the Lagrangian term relative to the potential energy, according to eq. 3.49:

$$\frac{\partial \mathcal{U}}{\partial q_i} = -\sum_{j=1}^{n}(m_{l_j}\boldsymbol{g}_0^T\boldsymbol{j}_{P_i}^{(l_j)}(\boldsymbol{q}) + m_{m_j}\boldsymbol{g}_0^T\boldsymbol{j}_{P_i}^{(m_j)}(\boldsymbol{q})) = g_i(\boldsymbol{q}) \tag{3.49}$$

The final equation of motion defined for the i^{th} row can be rewritten as eq. 3.50:

$$\sum_{j=1}^{n} b_{ij}(\boldsymbol{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}(\boldsymbol{q})\dot{q}_k\dot{q}_j + g_i(\boldsymbol{q}) = \xi_i \tag{3.50}$$

where

$$h_{ijk} = \frac{\partial b_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial b_{jk}}{\partial q_i} \tag{3.51}$$

The equation 3.50 underlies several interesting physical interpretations. In facts, the terms multiplying the joint acceleration $\ddot{q}_j$ can be seen as the set of coefficient b_{ii} which represents the inertia moment related to the i^{th} in the current robot configuration where the other joints are fixed and the coefficients b_{ij} which take into account the effect of acceleration of the j^{th} joint on joint i.
Concerning the coefficients of the quadratic term of the joint velocity, it is possible to notice that the term $h_{ijj}\dot{q}_j^2$ represents the centrifugal effect induced by the velocity of joint j on joint i. A further observation leads to the conclusion that h_{iii} is null since the term $\partial b_{ii}/\partial q_i = 0$. On the other hand, the terms with mutual joint velocity $h_{ijk}\dot{q}_j\dot{q}_k$ accounts for the Coriolis effect induced to the i^{th} joint by the velocities at joints j and k.
Finally, for the terms only depending on the configuration, being described in $g_i(\boldsymbol{q})$ it represents the torque generated by the effect of gravity on the joint i given the actual configuration of the robot.
The non-conservative forces that acts on the joints of the manipulator are the resultant from the actuation torques $\boldsymbol{\tau}$ to which the viscous friction torques given by $\boldsymbol{F}_v\dot{\boldsymbol{q}}$ and the static friction torques $\boldsymbol{f}_s(\boldsymbol{q},\dot{\boldsymbol{q}})$. $\boldsymbol{F}_v$ represents an $(n \times n)$ matrix containing the viscous friction coefficients and analogously $\boldsymbol{F}_s$ for the static counterpart. $\boldsymbol{f}_s$ is then evaluated accounting the Coulomb friction model, where $\boldsymbol{f}_s = \boldsymbol{F}_s\ \boldsymbol{sgn}(\dot{\boldsymbol{q}})$ where $\boldsymbol{sgn}(\dot{\boldsymbol{q}})$ accounts for the sign of the velocity at the joints. An additional component to be evaluated is the contribution of the external environment given by the interaction with the end-effector and its contact forces. given $\boldsymbol{h}_e$ the vector containing the forces and moments executed by the terminal tool, the resulting torques on the joints are given by $\boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{h}_e$.
The final dynamic model in the joint space can be described as eq. 3.52:

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \dot{\boldsymbol{F}}_v\dot{\boldsymbol{q}} + \boldsymbol{F}_s\boldsymbol{sgn}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} - \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{h}_e \tag{3.52}$$

where $\boldsymbol{C}$ is a $(n \times n)$ matrix defined such that its elements satisfy the relation

$$\sum_{j=1}^{n} \sum_{k=1}^{n} h_{ijk} \dot{q}_k \dot{q}_j. \tag{3.53}$$

This dynamic model presents interesting properties that lead to the dynamic parameters identification and to the definition of some control algorithms.
The first characteristic is the anti-simmetricity of the matrix composed by $\dot{\boldsymbol{B}} - 2\boldsymbol{C}$ obtained by selecting the correct parameters of the non-univocal $\boldsymbol{C}$ matrix, which are the first type Christoffel symbols. A second property important to note is the linearity in the dynamic parameters. These observation will be better understood in the analysis of the torques measured at the joints in the multi-body model defined after the trajectory planning digression with the help of `MatLab SimScape Multi-Body Model`.

Chapter 4

Trajectory planning

Trajectory planning is one of the key aspects for the completion of tasks by the manipulator. The starting point is the definition of a series of parameters as input, such as a temporal sequence of interpolating points that build the desired trajectory. The control system is responsible for the fulfilment of the planned path. The correct trajectory planning is the key factor for the performance requirements, leading to the exploitation of the max velocities and accelerations taking into consideration the structural solicitations of the mechanical structure and its frequency response. As a starting point for this analysis, a point-to-point motion algorithm for a trajectory in the joint space will be analysed and then, a wider range of solutions will be explored in order to choose the most suitable one for the several tasks of the gantry stacker.

4.1 Trajectories in joint space

The specification of a given trajectory is typically defined with parameters in the operational space, such as initial and final configuration of the manipulator. In order to evaluate a trajectory in the joint space, it is necessary to extract the joint variables $\vec{\mathbf{q}}$ from position and orientation assigned to the given end-effector points with an inverse kinematic algorithm, as already seen in chapter 3. In the simplest case, the robot task is the movement from an initial configuration $\vec{\mathbf{q}}_i$ to the final configuration $\vec{\mathbf{q}}_i$ in a fixed time interval t_f. The goal of these algorithms is to generate a trajectory that can be a solution for an optimization problem for some quality factor index. A possible choice of this factor is the minimization of dissipated power for the motor. Given I, the moment of inertia of the robot arm relative to its axis of rotation, the torque τ applied by the motor is related to the variation of angular velocity $\dot{\omega}$ (where ω represents the joint variable derivative $\dot{q}$)

through eq. 4.1.

$$I\dot{\omega} = \tau \tag{4.1}$$

The suitable solution must satisfy the eq. 4.2 in order to go from a posture q_i to q_f in the predefined time and minimize the index quality given by the eq. 4.2.

$$\int_0^{t_f} \omega(t)dt = q_f - q_i \tag{4.2}$$

$$\min_{\forall\omega} \int_0^{t_f} \tau^2(t)dt \tag{4.3}$$

A simplification is required, because the moment of inertia of the axis is dependent on the complete configuration of the manipulator, being the distances of the link masses related to the joint variables $\vec{q}$ [29]. The resulting solution for this problem is of the form 4.4.

$$\omega(t) = at^2 + bt + c \tag{4.4}$$

4.1.1 Polynomial trajectories

The solution to this minimization problem is the cubic polynomial that defines the kinematic of the joint variable according to eqs. 4.5, 4.6 and 4.7.

$$q(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \tag{4.5}$$

$$\dot{q}(t) = 3a_3t^2 + 2a_2t + a_1 \tag{4.6}$$

$$\ddot{q}(t) = 6a_3t + 2a_2 \tag{4.7}$$

Since these equations take into account four parameters (a_0, a_1, a_2 and a_3), it is possible to impose four boundary conditions that are the values of the joint variable and its velocity at the initial and final configuration, according to eqs. 4.8, 4.9, 4.10 and 4.11.

$$q_i = a_0 \tag{4.8}$$

$$\dot{q}_i = a_1 \tag{4.9}$$

$$q_f = a_3t_f^3 + a_2t_f^2 + a_1t_f + a_0 \tag{4.10}$$

$$\dot{q}_f = 3a_3t_f^2 + 2a_2f_t + a_1 \tag{4.11}$$

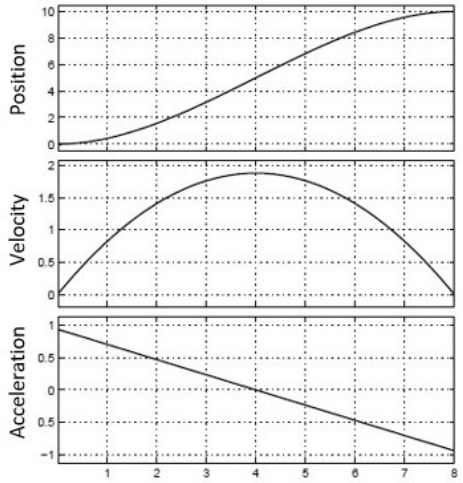


Figure 4.1: Cubic polynomial with null boundary velocities

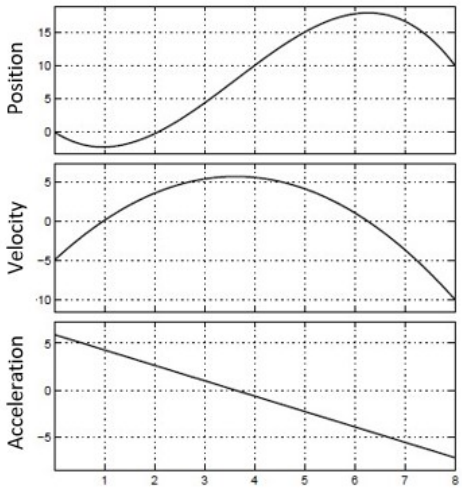


Figure 4.2: Cubic polynomial with boundary velocities not null

Another possible trajectory definition is the quintic polynomial . With respect to the former solution, this form contains two more coefficients, leading to the further assignment of the initial and final values of the joint acceleration $\ddot{q}$. The motion law for the generic joint expressed as quintic polynomial returns the kinematic variables according to eqs. 4.12, 4.13 and 4.14.

$$q(t) = a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{4.12}$$

$$\dot{q}(t) = 5a_5t^4 + 4a_4t^3 + 3a_3t^2 + 2a_2t + a_1 \tag{4.13}$$

$$\ddot{q}(t) = 20a_5t^3 + 12a_4t^2 + 6a_3t + 2a_2 \tag{4.14}$$

This model smooths the trajectory, defining the jerk for the motion. It is important to stress that this polynomial does not satisfy the minimization of the index quality stated in eq. 4.3. The boundary conditions to be imposed are defined in eq. 4.20.

$$q_i = a_0 \tag{4.15}$$

$$\dot{q}_i = a_1 \tag{4.16}$$

$$\ddot{q}_i = a_2 \tag{4.17}$$

$$q_f = a_5t^5 + a_4t^4 + a_3t_f^3 + a_2t_f^2 + a_1t_f + a_0 \tag{4.18}$$

$$\dot{q}_f = 5a_5t_f^4 + 4a_4t_f^3 + 3a_3t_f^2 + 2a_2f_t + a_1 \tag{4.19}$$

$$\ddot{q}_f = 20a_5t_f^3 + 12a_4t_f^2 + 6a_3t_f + 2a_2 \tag{4.20}$$

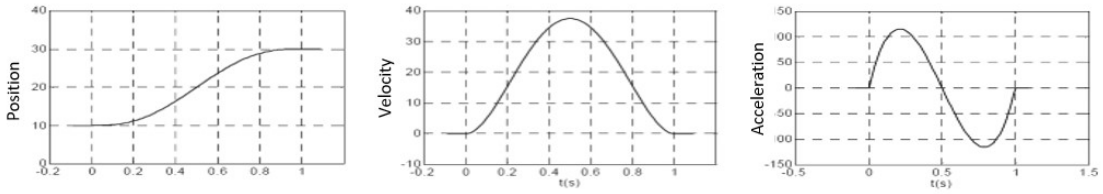


Figure 4.3: Quintic polynomial

In general, it is possible to adopt a *n-th* polynomial to describe the given trajectory.

$$q(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + \dots a_n(t - t_0)^n \tag{4.21}$$

The polynomial order n depends on the number of boundary conditions that need to be imposed, but also on the smoothness required for the motion. In addition to the initial and final conditions, it is possible to specify the kinematic values at general time instances t_j, by calculating the derivative of the *k-th* order, according to eq. 4.22 [32].

$$q^{(k)}(t_j) = k!a_k + (k+1)!a_{k+1}t_j + \dots + \frac{n!}{(n-k)!}a_n t_j^{n-k} \tag{4.22}$$

4.1.2 Trigonometric trajectories

In several manipulator tasks, the adoption of trigonometric functions is a viable solution with the purpose of planning trajectories with smoother acceleration or jerk profiles that can reduce resonance due to residual vibrations applied to resonant systems. Similarly to polynomial trajectories, the characteristic parameters can be evaluated on the basis of the desired bounds on velocity, acceleration, jerk, snap and so on. Furthermore, the equivalence between dynamic behaviour and trajectories expressed by analytic functions provides an immediate characterization of the motion from a spectral frequency analysis.
The first possibility is the harmonic function, defined according to eq. 4.23 where h represents the amplitude between q_f and q_i.

$$q(t) = \frac{h}{2}\left(1 - cos\frac{\pi(t - t_0)}{T}\right) + q_0 \tag{4.23}$$

This equation generalize the harmonic motion where the acceleration is directly proportional to the displacement, but with opposite sign. Being composed by trigonometric functions, the higher order derivatives are always continuous as

described in eq. 4.24 for the joint velocity, eq. 4.25 for the joint acceleration and eq. 4.26 for the joint jerk [32].

$$\dot{q}(t) = \frac{h}{2}\frac{\pi}{T} sin\left(\frac{\pi(t-t_0)}{T}\right) \tag{4.24}$$

$$\ddot{q}(t) = \frac{h}{2}\left(\frac{\pi}{T}\right)^2 cos\left(\frac{\pi(t-t_0)}{T}\right) \tag{4.25}$$

$$\dddot{q}(t) = -\frac{h}{2}\left(\frac{\pi}{T}\right)^3 sin\left(\frac{\pi(t-t_0)}{T}\right) \tag{4.26}$$

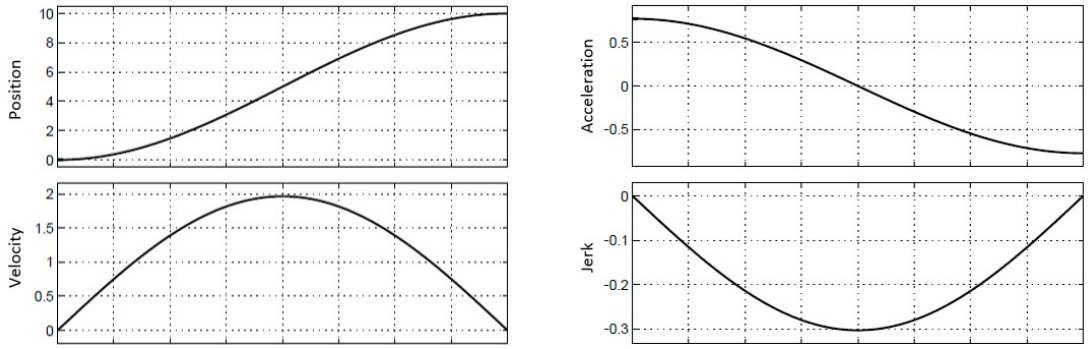


Figure 4.4: Harmonic trajectory and derivatives

The principal drawback of the harmonic function is the discontinuity in the acceleration in the initial and final configuration of the trajectory, thus resulting in a jerk that rises up to infinity. To overcome this problem and maintaining the trigonometric behaviour, the adoption of a cycloidal function 4.27 is a good compromise that ensure continuity in the acceleration.

$$\begin{cases} q(t) = h\left(\frac{t-t_0}{T} - \frac{1}{2\pi} sin\frac{2\pi(t-t_0)}{T}\right) + q_0 \\ \dot{q}(t) = \frac{h}{2}\frac{\pi}{T} sin\left(\frac{\pi(t-t_0)}{T}\right) \\ \ddot{q}(t) = \frac{h}{2}\left(\frac{\pi}{T}\right)^2 cos\left(\frac{\pi(t-t_0)}{T}\right) \\ \dddot{q}(t) = -\frac{h}{2}\left(\frac{\pi}{T}\right)^3 sin\left(\frac{\pi(t-t_0)}{T}\right) \end{cases} \tag{4.27}$$

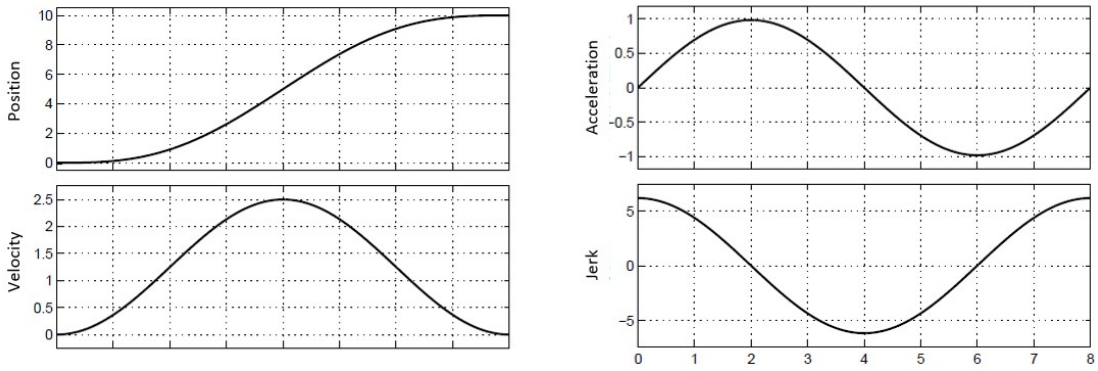


Figure 4.5: Cycloid trajectory and derivatives

4.1.3 Composite and piece-wise trajectories

An alternative approach for the calculation of the manipulator motion law is a mixed polynomial tailored for the industrial practice. these piece-wise functions can exploited defined accelerations and velocities according to the mechanical coupling. The most common solution belonging to this category is the velocity trapezoidal trajectory, which is characterised by 3 segments as referred in 4.28: at first the acceleration phase with positive constant a_2 in the time interval $t \in [0, T_a]$, followed by a constant cruise velocity in the time interval $t \in [T_a, T - T_a]$ that ends in a deceleration phase with negative constant $\ddot{q}$.

$$\begin{cases} q_a(t) = a_0 + a_1 t + a_2 t^2 & \text{for } 0 \leq t \leq T_a \\ q_b(t) = b_0 + b_1 t & \text{for } T_a \leq t \leq T_f - T_a \\ q_c(t) = c_0 + c_1 t + c_2 t^2 & \text{for } T_f - T_a \leq t \leq T_f \end{cases} \tag{4.28}$$

The trapezoidal function is also defined as 2-1-2 according to the order of the polynomial adopted in the various traits of the curve. As shown in eq. 4.28, 8 parameters have to be defined, also accounting for joint position and joint velocity continuous at the conjunctions of the traits. Other constraints are compulsory for the feasibility of the trapezoidal trajectory as requirements for the acceleration time (4.29) and the minimum acceleration to fulfil the displacement (4.30).

$$T_a \leq \frac{T}{2} \tag{4.29}$$

$$|\ddot{q}| \geq \frac{4|q_f - q_i|}{T^2} \tag{4.30}$$

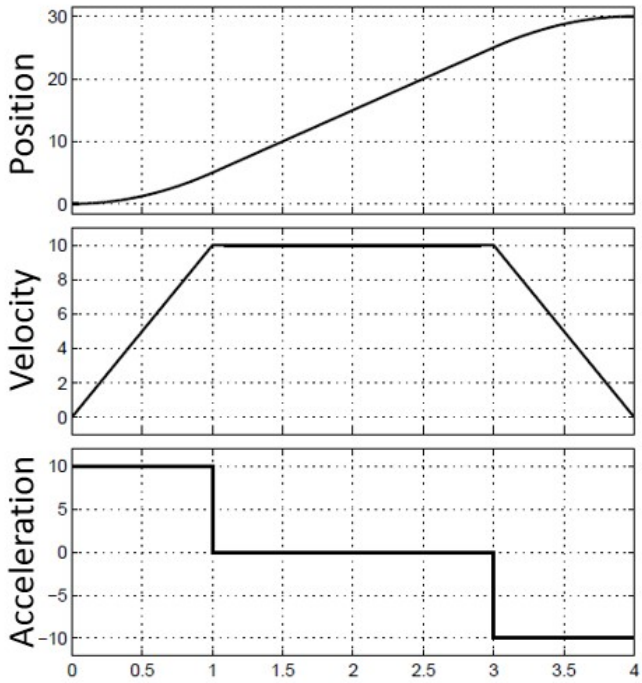


Figure 4.6: Trapezoidal trajectory

The configuration of these parameters can change according to the desired requirement. To exploit the maximum characteristic of the drive, it can be imposed maximum joint velocity and acceleration, which in the trapezoidal case result in the velocity in the constant trait and the acceleration values in the others, as defined in 4.31.

$$\begin{cases} \dot{q}_b(t) = v_{max} \\ \ddot{q}_a(t) = -\ddot{q}_c(t) = a_{max} \end{cases} \tag{4.31}$$

As a consequence, the acceleration time interval T_a and the total duration T are not predefined, but evaluated as 4.32.

$$\begin{cases} T_a = \frac{v_{max}}{a_{max}} \\ T = \frac{q_f - q_i}{v_{max}} + \frac{v_{max}}{a_{max}} \end{cases} \tag{4.32}$$

It is important to verify the condition of feasibility for the maximum velocity. In facts, for small joint displacements, a trajectory with a triangular velocity profile (so called bang-bang) is performed, formed by a trait of maximum acceleration and one of maximum deceleration. Another solution for the retrieval of the coefficients for the trapezoidal function is the imposition of the time characteristics T_a and T.

Since this piece-wise function presents discontinuity in the acceleration profile, this trajectory is not recommended for applications that require smoother motions.

Thus, piece-wise functions with polynomial junctions of order larger than 2 can be developed, imposing continuous acceleration and, furthermore, trajectories with continuous jerk. The 7-segments or double-S trajectory allows continuous second derivative and it is formed by three main phases: acceleration, constant velocity, deceleration. Differently from the trapezoidal velocity, the traits of acceleration/deceleration are divided in three sub-parts characterised by constant jerk (positive-null-negative and viceversa for deceleration phase).

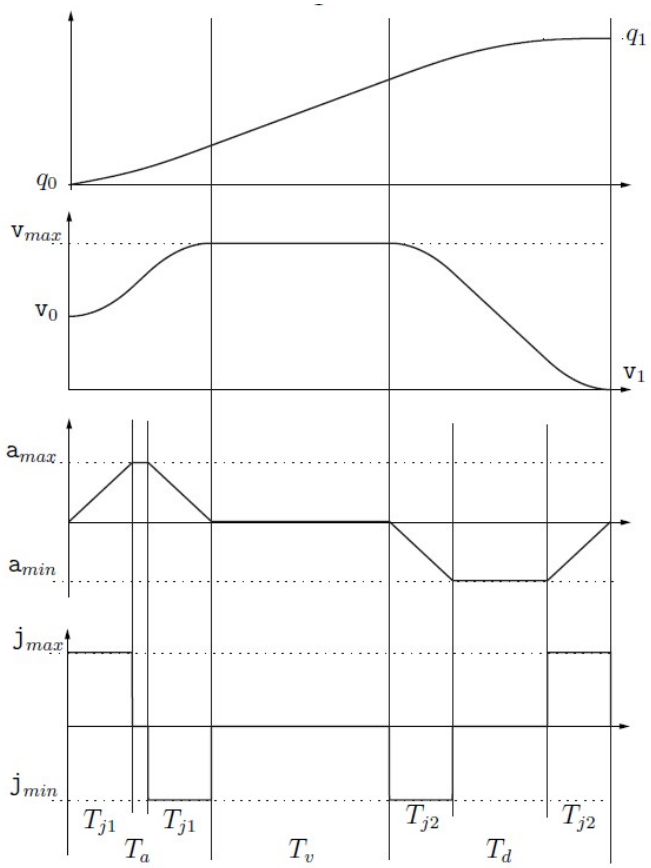


Figure 4.7: Double-S trajectory

4.1.4 Spline trajectories

The spline trajectories are multi-points functions composed by piece-wise polynomials (typically 3-th or 5-th order) with interpolation conditions through n points with continuity in the whole trajectory and its derivatives up to a certain order. It can be demonstrated that, given the conditions on continuity, the spline function

is the interpolating trajectory with minimum curvature.

$$\begin{aligned} s(t) &= \{g_k(t), t \in [t_k, t_{k+1}], k = 0, \dots n-1\}, \\ q_k(t) &= a_{k_0} + a_{k_1}(t-t_k) + a_{k_2}(t-t_k)^2 + a_{k_3}(t-t_k)^3 \end{aligned} \tag{4.33}$$

Considering n points, $n-1$ polynomials are defined. Starting from a cubic spline defined according to eq. 4.33, $4(n-1)$ parameters have to be defined according to the continuity constraints.

- $2(n-1)$ conditions on trajectory passing through extremities, since each cubic segment has to start and finish in two consecutive waypoints;
- $n-2$ conditions on continuity velocity at the boundary points of each interval;
- $n-2$ conditions on continuity acceleration in the predefined intermediate points.

In this case, The degree of freedom of the system of equations for the spline parameters can be evaluated according to equation 4.34.

$$4(n-1) - 2(n-1) - 2(n-2) = 2 \tag{4.34}$$

The final two parameters can be defined by imposing the initial and final velocity condition. Another possibility for a spline is the polynomial sequence with the definition of the desired joint velocity $\dot{q}_k$. As a result, the continuity condition on the acceleration falls and substituted by the equations imposing the velocity at the waypoints.

The computation algorithm for the evaluation of the coefficients a_{k_i} starts from the definition of the system of constraints, according to 4.35.

$$\begin{cases} q_k = q_k(t_k) = a_{k_0} \\ v_k = \dot{q}_k(t_k) = a_{k_1} \\ q_{k+1} = q_k(t_{k+1}) = a_{k_0} + a_{k_1}T_k + a_{k_2}T_k^2 a_{k_3}T_k^3 \\ v_{k+1} = \dot{q}_k(t_{k+1}) = a_{k_1} + 2a_{k_2}T_k + 3a_{k_3}T_k^2 \end{cases} \tag{4.35}$$

If the velocities at the junction points are not given, the evaluation of $v_1 \dots v_{n-1}$ can be done by imposing continuity in the acceleration, thus resulting in eq. 4.36.

$$\begin{aligned} &\ddot{q}_k(t_{k+1}) = \ddot{q}_{k+1}(t_{k+1}) \\ &2a_{k_2} + 6a_{k_3}T_k = 2a_{(k+1)_2} \end{aligned} \tag{4.36}$$

Substituting the expressions for the coefficients extracted from eq. 4.35 and multiplying by $\frac{T_k T_{k+1}}{2}$, the general equation for the k-th polynomial can be rewritten in a matrix form, according to the steps shown in 4.37.

$$T_{k+1}v_k + 2(T_{k+1}+T_k) + T_k v_{k+2} = \frac{3}{T_k T_{k+1}}[T_k^2(q_{k+2}-q_{k+1}) + T_{k+1}^2(q_{k+1}-q_k)] \quad (4.37)$$

This results in a matrix component comprehending the time interval elements **A** and the vector of velocities **v**. The matrix **C** is formed by constants depending only on the intermediate positions and the time interval of the segments.

$$\mathbf{A}' = \begin{bmatrix} T_1 & 2(T_0+T_1) & T_0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & T_2 & 2(T_1+T_2) & T_1 & & & & \vdots \\ \vdots & & & \ddots & & & & \\ & & & 0 & T_{n-2} & 2(T_{n-3}-T_{n-2}) & T_{n-3} & 0 \\ 0 & \cdots & & & 0 & T_{n-1} & 2(T_{n-2}+T_{n-1}) & T_{n-2} \end{bmatrix}$$

Given initial and final speed, it is possible to eliminate the first and last row, resulting in 4.38.

$$\mathbf{A}(\mathbf{T})\mathbf{v} = \mathbf{c}(\mathbf{T}, \mathbf{q}, v_0, v_n) \quad (4.38)$$

The new **A** matrix is diagonally-dominant and thus is always invertible, given that the time interval T_k are always positive. Furthermore, being a tri-diagonal matrix, efficient inversion algorithms can be exploited and thus, the vector v can be calculated as $\mathbf{v} = \mathbf{A}^{-1}\mathbf{c}$ and the spline coefficients are evaluated through eq. 4.35.

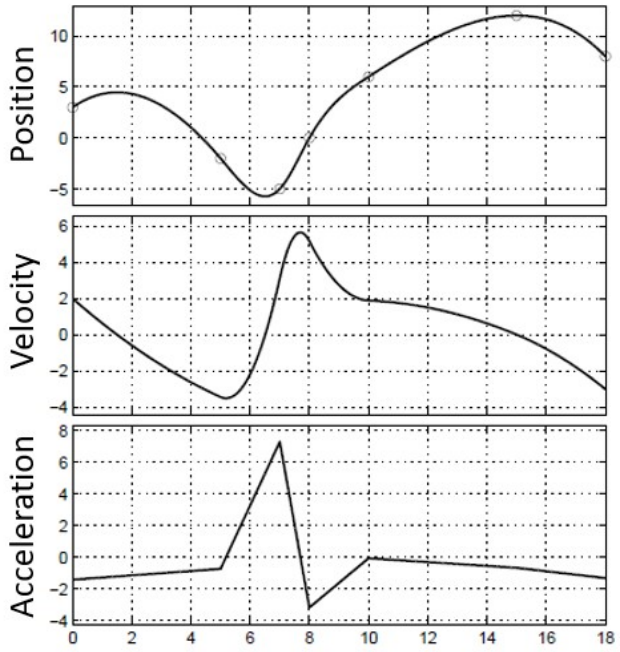


Figure 4.8: Spline interpolation

4.2 Analysis and comparison between trajectories

In order to satisfy certain constraints on maximum velocities or accelerations, it is convenient to consider trajectories expressed in the same boundaries and so the functions require to be normalised by applying scaling operation in the geometrical and temporal domain. Every trajectory is thus defined over a displacement $h = q_1 - q_0$ and a time length $T = t_1 - t_0$ and so it can be rewritten in normalised form as $q_N(\tau)$ where:

$$\tau = \frac{t - t_0}{t_1 - t_0} \tag{4.39}$$

$$q_N(\tau) = \frac{q(t) - q_0}{q_1 - q_0} \tag{4.40}$$

As a consequence, the temporal variable t is obtained scaling τ of a factor of $\lambda = T$. It follows that the derivatives defining the joint velocities and the other kinematic parameters are function of h and T according to eq. 4.41.

$$\begin{aligned} \dot{q}(t) &= \frac{h}{T}\dot{q_N}(\tau) \\ \ddot{q}(t) &= \frac{h}{T^2}\ddot{q_N}(\tau) \\ &\vdots \\ q^{(n)}(t) &= \frac{h}{T^n}{q_N}^{(n)}(\tau) \end{aligned} \tag{4.41}$$

The maximum values of velocity, acceleration and jerk are found in correspondence of the stationary points of the normalised derivatives. By modifying the duration of the trajectory T it is possible to exploit the kinematic saturation achieving the maximum velocity and the maximum acceleration. Considering the simplest trajectory, the cubic polynomial with a rest-rest trajectory (with initial and final joint velocity null), the peak of velocity is reached in the medium point of the trajectory with $\tau = 0.5$, while the max acceleration is performed at the initial time with $\tau = 0$. in eq. 4.42, the result for max acceleration and max speed are performed.

$$\begin{aligned} \dot{q}_N(\tau)_{max} = \dot{q}_N(0.5) = \frac{3}{2} &\implies \dot{q}_{max} = \frac{3h}{2T} \\ \ddot{q}_N(\tau)_{max} = \ddot{q}_N(0) = 6 &\implies \ddot{q}_{max} = \frac{6h}{T^2} \end{aligned} \tag{4.42}$$

Working in the same conditions, the quintic polynomial has a smoother profile and so the peak values of the derivatives are reached at different values of the time fraction, as stated in eq. 4.43.

$$\begin{aligned} \dot{q}_N(\tau)_{max} = \dot{q}_N(0.5) = \frac{15}{8} &\implies \dot{q}_{max} = \frac{15h}{8T} \\ \ddot{q}_N(\tau)_{max} = \ddot{q}_N\Big(\frac{3-\sqrt{3}}{6}\Big) = \frac{10\sqrt{3}}{3} &\implies \ddot{q}_{max} = \frac{10\sqrt{3}h}{3T^2} \\ \dddot{q}_N(\tau)_{max} = \dddot{q}_N(0) = 60 &\implies \dddot{q}_{max} = \frac{60h}{T} \end{aligned} \tag{4.43}$$

Considering instead the trigonometrical trajectories, the normalised cycloidal function is defined according to eq. 4.44.

$$q_N(\tau) = \tau - \frac{1}{2\pi} sin(2\pi\tau) \tag{4.44}$$

From which it is possible to derive the other kinematic functions, as stated in eq. 4.45

$$\begin{aligned} \dot{q}_N(\tau) &= 1 - cos(2\pi\tau) \\ \ddot{q}_N(\tau) &= 2\pi sin(2\pi\tau) \\ \dddot{q}_N(\tau) &= 4\pi^2 cos(2\pi\tau) \end{aligned} \tag{4.45}$$

The absolute maxima in these equations are found in the following time fractions τ, resulting in the peak values shown in eq. 4.46.

$$\begin{aligned} \dot{q}_{N_{max}} = \dot{q}_N(\tau = 0.5) = 2 &\implies \dot{q}_{max} = 2\frac{h}{T} \\ \ddot{q}_{N_{max}} = \ddot{q}_N(\tau = 0.25) = 2\pi &\implies \dot{q}_{max} = 2\pi\frac{h}{T} \\ \dddot{q}_{N_{max}} = \dddot{q}_N(\tau = 0) = 4\pi^2 &\implies \dot{q}_{max} = 4\pi^2\frac{h}{T} \end{aligned} \tag{4.46}$$

On the other hand, the harmonic motion profile is characterised by the normalised function described by eq. 4.47

$$q_N(\tau) = \frac{1}{2}(1 - cos(\pi\tau) \tag{4.47}$$

From which it is possible to derive the other kinematic functions, as stated in eq. 4.48

$$\begin{aligned} \dot{q}_N(\tau) &= \frac{\pi}{2} sin(\pi\tau) \\ \ddot{q}_N(\tau) &= \frac{\pi^2}{2} cos(\pi\tau) \\ \dddot{q}_N(\tau) &= -\frac{\pi^3}{2} sin(\pi\tau) \end{aligned} \tag{4.48}$$

The absolute maxima in these equations are found in the following time fractions τ, resulting in the peak values shown in eq. 4.49.

$$\begin{aligned} \dot{q}_{N_{max}} = \dot{q}_N(\tau = 0.5) = \frac{\pi}{2} &\implies \dot{q}_{max} = \frac{\pi h}{2T} \\ \ddot{q}_{N_{max}} = \ddot{q}_N(\tau = 0) = \frac{\pi^2}{2} &\implies \dot{q}_{max} = \frac{\pi^2 h}{2T^2} \\ \dddot{q}_{N_{max}} = \dddot{q}_N(\tau = 0.5) = \frac{\pi^3}{2} &\implies \dot{q}_{max} = \frac{\pi^3 h}{2T^3} \end{aligned} \tag{4.49}$$

This analysis, regarding the exploitation of the peak velocity and acceleration of the drive, returns that each profile has a minimum time to saturate the capacity of the motor. Depending on the parameters of the drive and the path extremes, it is possible to classify the faster trajectory, considering the highest T_{min} to complete the trajectory.

Trajectory	Formula	constraints	T_{min}
Cubic polynomial	$\dot{q}_{max} = \frac{3h}{2T}$	$T = \frac{3h}{2\dot{q}_{max}}$	$1.5\frac{h}{\dot{q}_{max}}$
	$\ddot{q}_{max} = \frac{6h}{T^2}$	$T = \sqrt{\frac{6h}{\ddot{q}_{max}}}$	$2.449\sqrt{\frac{h}{\ddot{q}_{max}}}$
Quintic polynomial	$\dot{q}_{max} = \frac{15h}{8T}$	$T = \frac{15h}{8\dot{q}_{max}}$	$1.875\frac{h}{\dot{q}_{max}}$
	$\ddot{q}_{max} = \frac{10\sqrt{3}h}{3T^2}$	$T = \sqrt{\frac{10\sqrt{3}h}{3\ddot{q}_{max}}}$	$2.403\sqrt{\frac{h}{\ddot{q}_{max}}}$
Harmonic function	$\dot{q}_{max} = \frac{\pi h}{2T}$	$T = \frac{\pi h}{2\dot{q}_{max}}$	$1.571\frac{h}{\dot{q}_{max}}$
	$\ddot{q}_{max} = \frac{\pi^2 h}{2T^2}$	$T = \sqrt{\frac{\pi^2 h}{2\ddot{q}_{max}}}$	$2.221\sqrt{\frac{h}{\ddot{q}_{max}}}$
Cycloidal function	$\dot{q}_{max} = \frac{2h}{T}$	$T = \frac{2h}{\dot{q}_{max}}$	$2\frac{h}{\dot{q}_{max}}$
	$\ddot{q}_{max} = \frac{2\pi h}{T^2}$	$T = \sqrt{\frac{2\pi h}{\ddot{q}_{max}}}$	$2.507\sqrt{\frac{h}{\ddot{q}_{max}}}$

4.2.1 Comparison on the actuation system exploitation

In the previous table, the trajectory are compared in terms of faster time given the same peak velocity and acceleration, but in practice there exist other parameters to take into account for the selection of a trajectory for a specified application, considering, for example, the actuation system. In facts, an electric motor has a typical torque-velocity behaviour, as shown in fig. 4.9, and the following parameters:

- peak torque (τ_p), which is the instantaneous maximum torque value capable to generate;
- effective or continuative torque (τ_c), which is the maximum torque value capable to generate indefinitely over time;

- nominal velocity (v_n), which is the maximum velocity rotation of the motor;
- maximum power (P_{max}), which identify the motor size;
- operative field, depending on the nature of the task (continuative or intermediate).

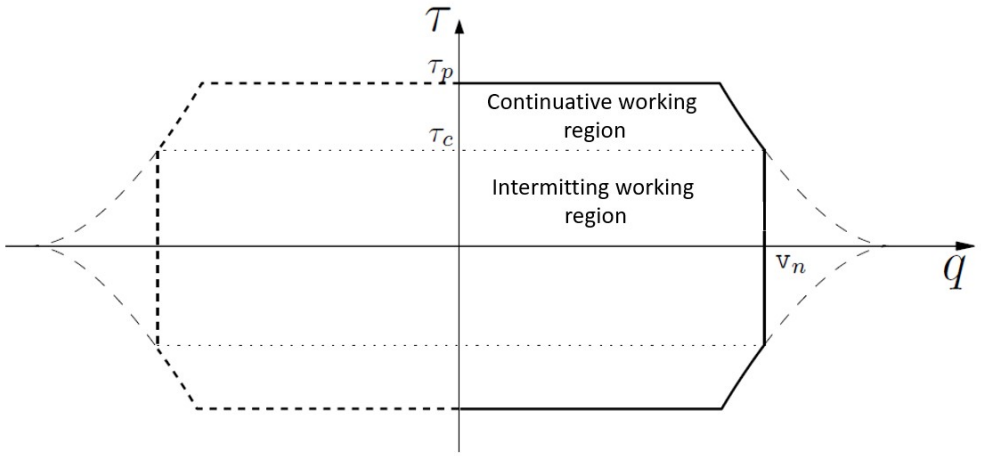


Figure 4.9: Operative field of the motor

It is fundamental to verify the compatibility of the motion profile characteristics with the actuator system limits. In addition to the simple condition on the maximum admissible velocity ($\dot{q}_{max} \leq v_n$), it is necessary to verify that the torque $\tau(t)$ requested to realize the tasks of the joint can be effectively supplied by the motor, according to eq. 4.50.

$$max\{\tau(t)\} = \tau_{max} \leq \tau_p \tag{4.50}$$

Considering a simple dynamic model, where only inertia and friction forces are considered, the torque expression results in eq. 4.51

$$\tau(t) = J_t\ddot{q}(t) + B_t\dot{q}(t) \tag{4.51}$$

where J_t represents the total moment of inertia, given by the contributes of the motor inertia J_m and the inertia of the load translated to the motor (scaled by the reduction ratio k_r^2). The same reasoning holds for the total damping coefficient $B_t = B_m + \frac{B_l}{k_r^2}$.

A motion profile is feasible if the mechanical task and so the curve described in the $(\dot{q}(t), \tau(t))$ is entirely included in the area below the torque-velocity characteristic of

the motor. In particular, when the load is predominantly inertial, the friction forces can be neglected and so $\tau(t) \approx J_t\ddot{q}(t)$ and so the acceleration profile represents a good estimate of the torque required for the execution of a given motion.

As an example for this feasibility analysis, two trajectories are analysed, according to fig. 4.10.

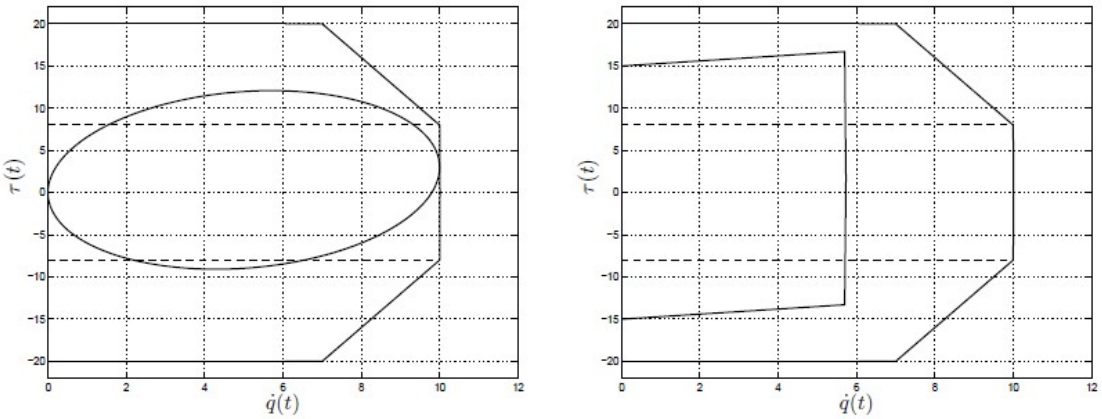


Figure 4.10: Torque velocity feasibility of motion profiles

The trajectory (a) represents a cycloidal point-to-point function and it is interesting to note the smooth behaviour in the curve, without abrupt non-derivative points. On the other hand, the motion profile (b) is the torque-speed requirement for a trapezoidal motion. Both trajectories are feasible because the curves lie below the motor characteristic. In the case of cyclic tasks, the thermal dissipation problem must be taken into account: in facts, the working point shall not stay in the intermittent working region. A possible feasibility impairment is the effective value for the torque along a period, that can be calculated with eq. 4.52 and must be lower than the continuative torque τ_c.

$$\tau_{eff} = \sqrt{\frac{1}{T}\int_0^T \tau^2(t)dt} \tag{4.52}$$

Reconsidering the definition of the torque, according to 4.51, the effective torque τ_{eff}^2 is given by 4.53.

$$\begin{aligned} \tau_{eff}^2 &= \frac{1}{T}\int_0^T \tau^2(t)dt \\ &= \frac{J_t^2}{T}\int_0^T \ddot{q}^2(t)dt + \frac{B_t^2}{T}\int_0^T \dot{q}^2(t)dt + 2\frac{J_tB_t}{T}\int_0^T \ddot{q}(t)\dot{q}(t)dt \\ &= J_t^2\ddot{q}_{eff}^2 + B_t^2\dot{q}_{eff}^2 \end{aligned} \tag{4.53}$$

This equation returns the effective values of joint velocity and joint acceleration. The knowledge of the peak and effective values of the velocity and acceleration profiles has a fundamental relevance on the sizing of the actuation system and, viceversa, on the feasibility of a given trajectory having fixed the drive. In order to compare trajectories it is a good practice to define the a-dimensional coefficients of velocity C_v and acceleration C_a independent from displacement h and period T, according to eq. 4.54.

$$\begin{aligned} C_v &= \frac{\dot{q}_{max}}{h/T} \implies \dot{q}_{max} = C_v \frac{h}{T} \\ C_a &= \frac{\ddot{q}_{max}}{h/T^2} \implies \ddot{q}_{max} = C_a \frac{h}{T^2} \end{aligned} \tag{4.54}$$

From the definition, the coefficients C_v and C_a can be seen as the maximum velocity and acceleration of the corresponding normalised trajectory $q_N(\tau)$. In the same way, it is possible to define the coefficient for the effective velocity $C_{v_{eff}}$ and effective acceleration $C_{a_{eff}}$, according to eq. 4.55.

$$\begin{aligned} C_{v_{eff}} &= \frac{\dot{q}_{eff}}{h/T} \implies \dot{q}_{eff} = C_{v_{eff}} \frac{h}{T} \\ C_{a_{eff}} &= \frac{\ddot{q}_{eff}}{h/T^2} \implies \ddot{q}_{eff} = C_{a_{eff}} \frac{h}{T^2} \end{aligned} \tag{4.55}$$

In the table are shown the values for the several coefficients depending on the chosen type of trajectory. It is

Trajectory	C_v	C_a	$C_{v_{eff}}$	$C_{a_{eff}}$
Polynomial 3-th grade	1.5	6	1.0954	3.4131
Polynomial 5-th grade	1.875	5.7733	1.1952	4.1402
Polynomial 7-th grade	2.1875	7.5017	1.2774	5.0452
Triangular (bang-bang)	2	4	1.1547	4
Trapezoidal	2	4.8881	1.2245	4.3163
Harmonic	1.5708	4.9348	1.1107	3.4544
Cycloidal	2	6.2832	1.2247	4.4428

Table 4.1: Comparison between trajectories

4.2.2 Comparison on the dynamic frequency response

The starting point of the dynamic analysis of a trajectory is the definition of the mechanical model of the system. During a motion, the structural elasticity and

dissipation behaviours of the system can create vibrational phenomenons. The precision, and thus the complexity of this modeling, that can be achieved depends on the time and costs limits. A possible solution is a FEM analysis of the system, where, starting from the elastic characteristic of the elements and retrieving the total response of the model.
A cheaper solution is the simplification of the mechanical parts that, from being intrinsically defined with distributed parameters (mass, elasticity and damping), are defined with elements with concentrated parameters defining mass elements without elasticity and viceversa. In addition, to simulate the energy dissipation due to the friction between moving parts, damping elements are added. The values of the mechanical parameters have to be determined and tuned in order to reproduce the behaviour of the various mechanical part from a kinematic and elastic point of view. The basic model is then composed by an actuator drive that put in motion a mass m. This system has a concentrated stiffness k and a dissipating element d, as reported in fig. 4.11 From this model, the dynamic of the system is described by

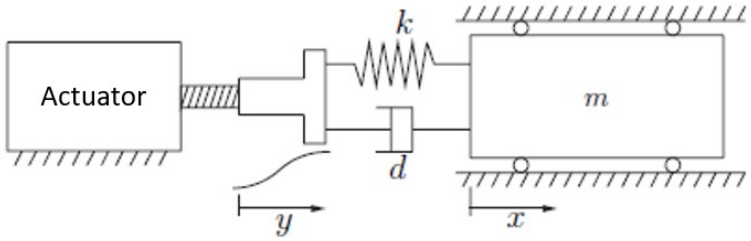


Figure 4.11: Single Mass-Spring-Damper model

the differential equations described in eq. 4.56:

$$\begin{aligned} m\ddot{x} + d\dot{x} + kx &= d\dot{y} + ky \\ m\ddot{z} + d\dot{z} + kz &= -m\ddot{y} \end{aligned} \tag{4.56}$$

that can be re-described with the canonical second order differential equation 4.57:

$$\ddot{z} + 2\delta\omega_n\dot{z} + w_n^2 z = -\ddot{y} \tag{4.57}$$

where

$$\begin{aligned} \omega_n &= \sqrt{\frac{k}{m}} \\ \delta &= \frac{d}{2m\omega_n} \end{aligned} \tag{4.58}$$

More complex models can be obtained by subdividing the single mass-spring-damper model into smaller ones taking into account n concentrated parameters elements, as shown in fig. 4.12, leading to a behaviour more similar to the distributed parameter system [32]. Supposing an ideal actuation system, where $y(t)$ corresponds to the

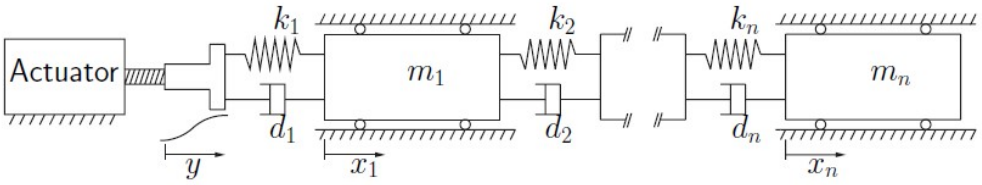


Figure 4.12: Multiple Mass-Spring-Damper model

joint variable $q(t)$, it is possible to evaluate the response of the system model to the several possible motion laws. Taking into consideration two opposite philosophies for the trajectory planning, the trapezoidal function with discontinuity in the acceleration and the smoother cycloidal function, the resulting response on the kinematic variables of the mass m is shown in fig. 4.13 and 4.14.

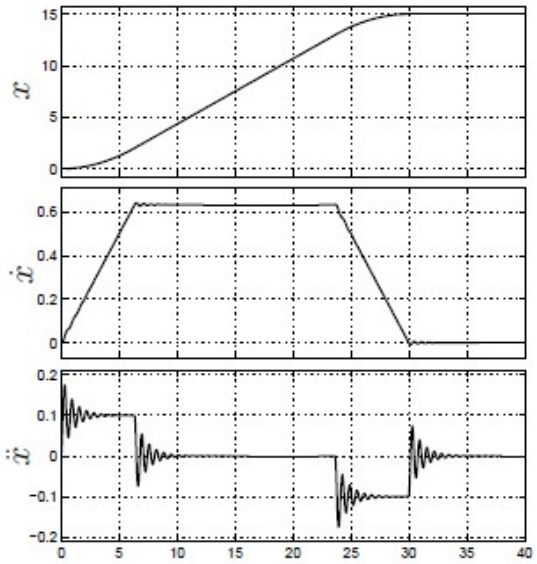


Figure 4.13: Trapezoidal function frequency response x

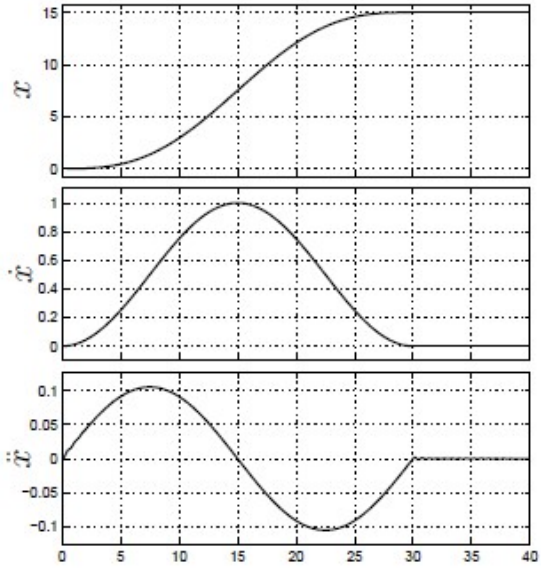


Figure 4.14: Harmonic function frequency response x

another observation can be done on the z profiles where it is considered the motion difference between the actuator drive and the mass element. With profiles

with discontinuity in the accelerations larger oscillations and are shown, according to fig. 4.15 and 4.16.

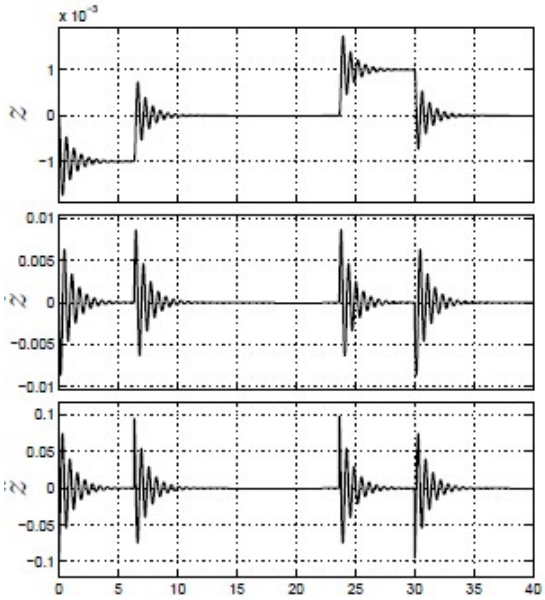


Figure 4.15: Trapezoidal function frequency response z

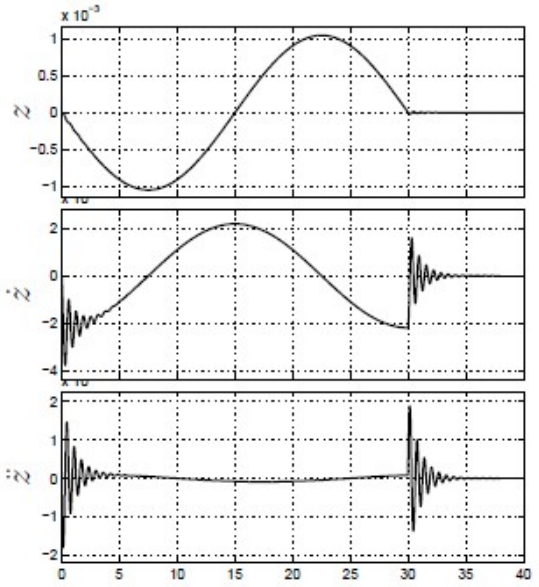


Figure 4.16: Harmonic function frequency response z

This phenomenon can be further analysed considering the harmonic content of the acceleration profile $\ddot{y}$ in relation eith the Bode diagram of the structure, focusing particularly on the natural frequency of the system ω_n. Recalling the Fourier's transform defined as 4.59, where the generic signal is defined as the sum of infinite sinusoidal waves with amplitude $V(\omega)$ and phase $\phi(\omega)$

$$x(t) = \int_0^{+\infty} V(\omega) cos[\omega t + \phi(\omega)] d\omega \tag{4.59}$$

it is possible to confront the amplitude at the several frequencies with the harmonic response value of the system, in order to verify the solicitation of the mechanical model. Important relevance relies on the avoidance of the resonance modes of the mechanical structure from the input trajectories. Considering again the single dof model stated in fig. 4.11 the transfer function in variable s that relates the acceleration $\ddot{y}$ and the difference $z(t)$ results in eq. 4.60.

$$G(s) = \frac{Z(s)}{A(s)} = \frac{-1}{s^2 + \frac{d}{m}s + \frac{k}{m}} \tag{4.60}$$

The corresponding Bode diagram for the amplitude of the system result in the function shown in fig. 4.17

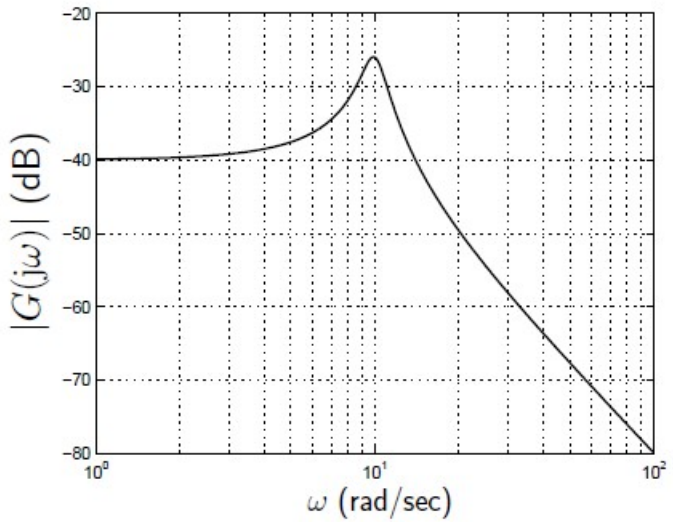


Figure 4.17: Bode diagram of the system transfer function

In order to avoid the vibratory phenomenon, the maximum pulsation of the trajectory requires to be significantly lower than the resonance frequency of the system, which corresponds to the peak frequency in the previous graph. The following step is the analysis of the frequency spectrum of the trajectory types. To express the relations in the same boundary conditions, the trajectories are normalised and the spectra are defined according to the a-dimensional variable Ω defined according to eq. 4.61.

$$\Omega = \frac{\omega}{\omega_0} \tag{4.61}$$

The spectra expressions can be determined in closed form or numerically and the following table contains the expressions for the principal trajectories.

Harmonic	$V'(\Omega) = \frac{h}{T}\frac{2\Omega}{1-4\Omega^2}\lvert\cos(\pi\Omega)\rvert$
Cycloidal	$V'(\Omega) = \frac{h}{T}\frac{2}{\pi(1-\Omega^2)}\lvert\sin(\pi\Omega)\rvert$
Polynomial degree 3	$V'(\Omega) = \frac{h}{T}\frac{6}{\pi^3\Omega^2}\sqrt{1+\pi^2\Omega^2}\lvert\sin(\pi\Omega)\rvert$
Polynomial degree 5	$V'(\Omega) = \frac{h}{T}\frac{30}{\pi^5\Omega^4}\sqrt{9+4\pi^2\Omega^2+\pi^4\Omega^4}\lvert\sin(\pi\Omega)\rvert$
Trapezoidal	$V'(\Omega) = \frac{h}{T}\frac{2}{(1-\alpha)\alpha\pi^2\Omega}\left\lvert\sin\left((1-\alpha)\pi\Omega\right)\sin\left(\alpha\pi\Omega\right)\right\rvert$
Double S	$V'(\Omega) = \frac{h}{T}\frac{2}{\alpha^2\beta(1-\alpha)(1-\beta)\pi^3\Omega^2}\left\lvert\sin\left((1-\alpha)\pi\Omega\right)\sin\left(\alpha(1-\beta)\pi\Omega\right)\sin\left(\alpha\beta\pi\Omega\right)\right\rvert$

Figure 4.18: Spectrum expression for different trajectories

Considering the piecewise functions, the parameter α and β represent respectively the fraction of time of the acceleration phase (for both trapezoidal and Double S trajectory) and the time ratio during the acceleration of constant jerk in the double S trajectory. In the figure 4.20, is shown the spectrum of the trapezoidal trajectory as a function of the α parameter. Decreasing α, the acceleration becomes more and more impulsive and so the harmonic components forming the trajectory signal will have undesired higher frequencies.

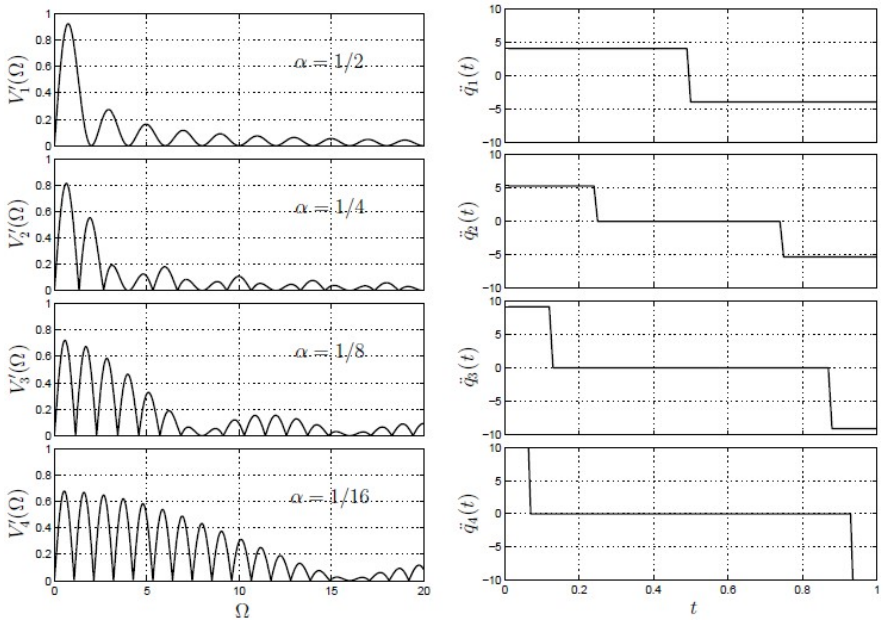


Figure 4.19: Trapezoidal function frequency response

On second analysis, the harmonic content of the acceleration profile of the Double S trajectory with $\alpha = 0.25$ and β variable. In this case, decreasing β means increasing the jerk value, thus resulting in a similar trajectory to the trapezoidal function, that represents the limit case with $\beta = 0$.

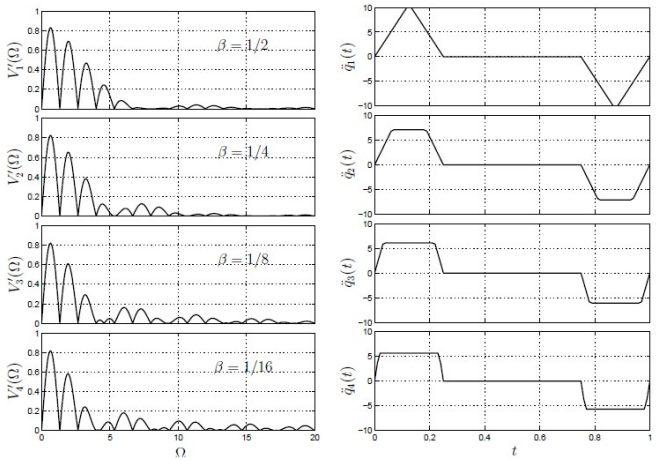


Figure 4.20: Double S frequency response

As last analysis, the spectrum content of the acceleration profile of the polynomial and sinusoidal trajectories are exposed in fig. 4.21.

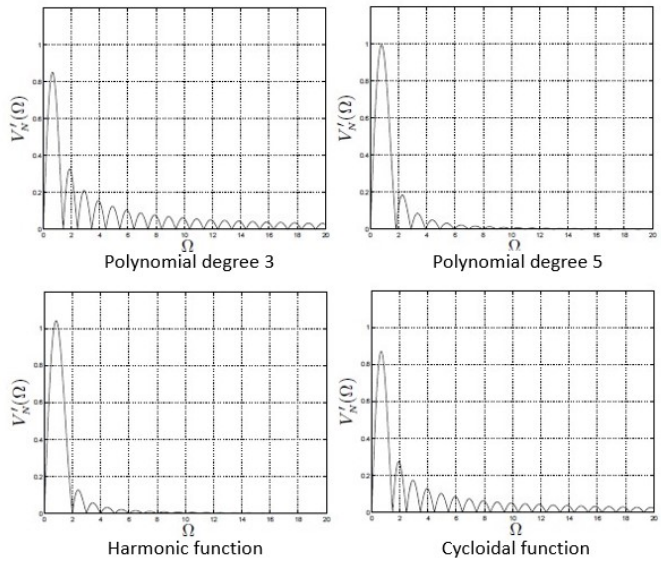


Figure 4.21: Frequency response for different trajectories

4.2.3 Trajectory analysis with `MatLab SimScape Multi-body`

In this section, a new model of this gantry stacker is developed eith the help of the `MatLab SimScape Multi-body`. In order to build this virtual partner, it is important to understand how this multi-body elements are designed and work. In the `SimuLink` environment, the first elements to be defined to start the composition of the kinematic chain is defined by the following parameters, as shown in fig. 4.22 [33].

- **Solver configuration**: in this block it is possible to define how the software elaborate the inputs of the system, for example by defining the equation formulation type as time or frequency analysis.
- **World Frame Reference**: Provides access to the world or ground frame, a unique motionless, orthogonal, right-handed coordinate frame predefined in any mechanical model. World frame is the ground of all frame networks in a mechanical model.
- **Mechanism configuration**: In this block element it is possible to set mechanical and simulation parameters that apply to an entire machine, the target machine to which the block is connected. In this area it can be specified uniform gravity for the entire mechanism.
- **Base rigid body**: represents the starting rigid body that refers to the global reference frame.

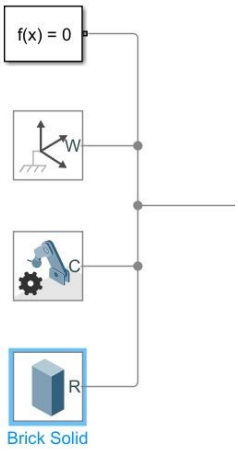


Figure 4.22: Definition of environment of the model

From this point on, the definition of the successive robotic arms and its corresponding joints. In facts, the modular element corresponding to the rigid link can be

defined as a MatLab subsystem defined by two rigid transforms related to a solid rigid body as shown in fig. 4.23.

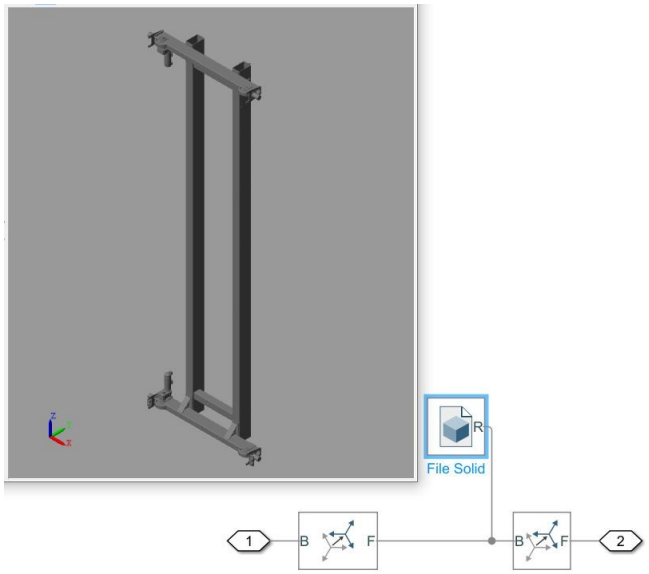


Figure 4.23: Definition of the single rigid body arm

The first block element defines a fixed 3D rigid transformation between two frames. The characteristics of this homogeneous transformation can be specified in accordance to the desired translational and rotational components. The second block represents a solid component whose geometry, material and visual properties can be retrieved from a CAD file designed with CATIA, NX, and other softwares. The File Solid block obtains the inertia from the geometry and density, from the geometry and mass, or from an inertia tensor previously specified. The final component required is the definition of the joint, as the interconnection mechanism between two rigid bodies. According to the nature of the desired joint, it is possible to define several parameter, that are briefly described in the following list [33].

- **State Targets**: definition of position and velocity targets
- **Internal mechanics**: in this section it is possible to specify the spring stiffness and the damping coefficients properties of the joint. In addition, it is possible to specify the limits of the primitives.
- **Actuation**: In this part the actuation modality is defined according to the force and motion. These two modes can be provided by the input or automatically computed starting from the system perturbation.

- **Sensing**: In the sensing area, the parameters that are ticked are displayed in a physical output port that can be further observed with a scope.

Once built the complete robotic structure, each joint can be controlled with a **q** function that can be defined directly in the SimuLink model with the help of the trajectory block functions or through functions created in the MatLab script environment. The final multi-body SimuLink project is shown in fig. 4.24

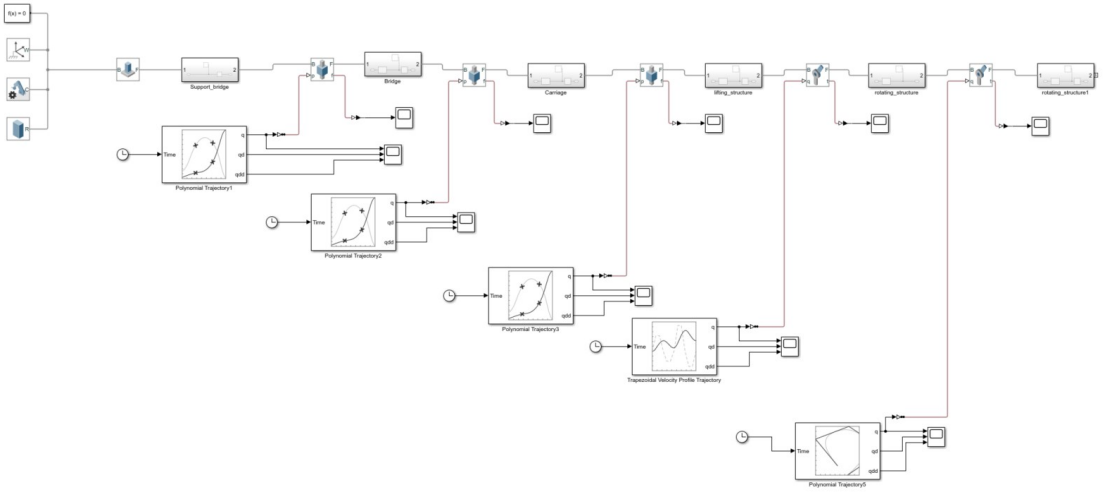


Figure 4.24: SimScape Multi-body model of the gantry

The trajectories that can be analysed in the previously described way, are polynomial and trapezoidal functions.

The Polynomial Trajectory block generates trajectories through waypoints at the given time points using cubic, quintic, or B-spline polynomials. The block outputs developed are position, velocity, and acceleration profiles that defines the trajectory based on the Time input. Positions of waypoints of the trajectory at given time points are specified through an $n \times p$ matrix, where n is the dimension of the trajectory (typically three dimensional) and p represents the number of waypoints. Additional parameters are the boundary constraints required to completely define the trajectories. In facts, for cubic polynomials Velocity boundary conditions for waypoints must be specified as an $n \times p$ matrix, where each row corresponds to the velocity at each of the p waypoints. In addition for quintic polynomials, acceleration boundary conditions for waypoints must be specified.

On the contrary, trapezoidal velocity profile trajectory block creates trajectories through multiple waypoints using trapezoidal velocity profiles. Parameters that can be defined to correctly design the proper trajectory are:

- **Peak velocity** of the profile segment. This peak velocity is the highest velocity that is achieved during the trapezoidal velocity profile. Depending on

its definition, a scalar value is applied to all the segments of the trajectory and between all waypoints. If using an n element vector is each segment of the trajectory is limited by the relative value in the vector.

- **Acceleration** of the velocity profile. This acceleration defines the constant acceleration from zero velocity to the previously defined Peak Velocity value.

- **Time duration** of trajectory segment, defines the time interval required to complete the given segment of the trapezoidal trajectory.

- **Duration of acceleration phase** of velocity profile. It defines the t_a constraint for the acceleration and deceleration phases in the several trapezoidal segments.

In order to communicate between the q output and the SimScape multi-body model and viceversa, PS-Simulnk and Simulink-PS converters convert the Physical signals from and to the Simulink output signals.

Thanks to this model, it is possible to define the trajectories through waypoints at each joint simultaneously, with the help of vectors (fixed or defined by the script). As an output, position, velocity, acceleration profiles are shown and analysable as shown in fig. 4.25 and 4.26 as previously theorized in this chapter.

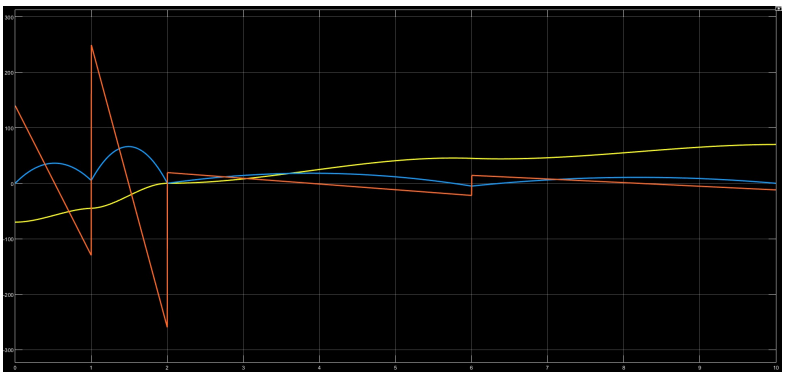


Figure 4.25: Polynomial profiles at the joint 1 (bridge movement)

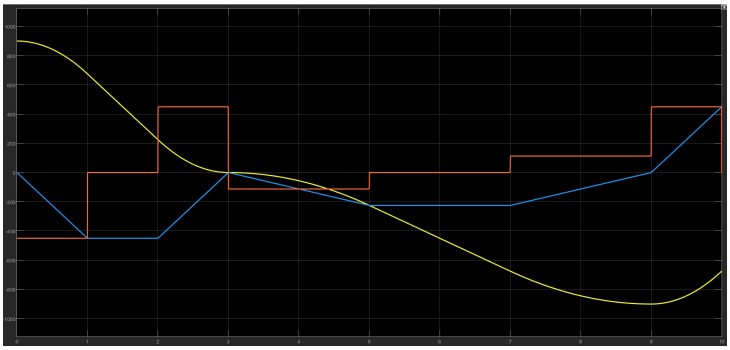


Figure 4.26: Trapezoidal profiles at the joint 5 (cup frame tilting)

In addition, the torque performed at the several joint can be shown at the output port of the block and their behaviour is in accordance to the theory explained in the last part of the fourth chapter. Torque graphs are depicted in fig. 4.27 and 4.28.

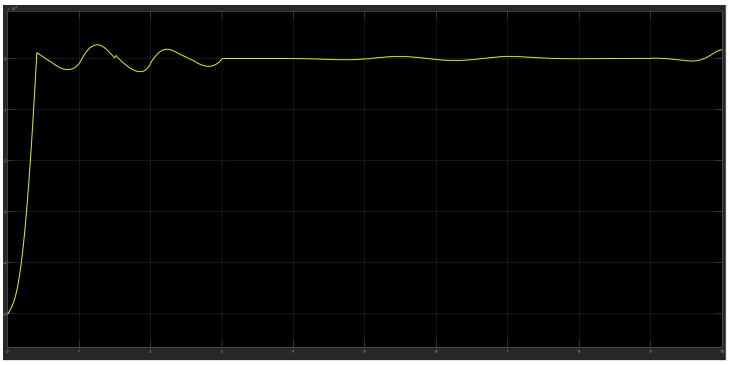


Figure 4.27: torque profiles at the joint 1 (bridge movement)

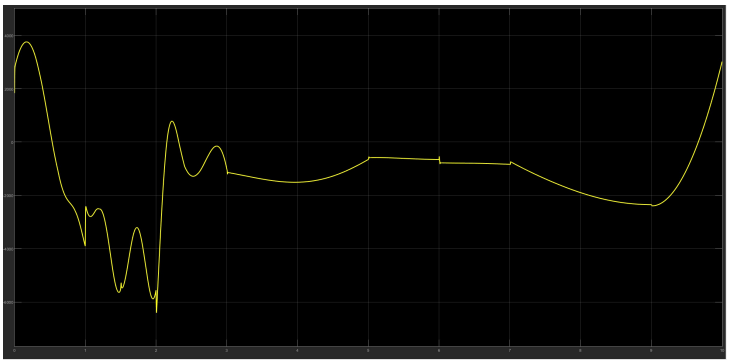


Figure 4.28: SimScape Multi-body model of the gantry

Another interesting outcome is a 3D representation of the simulated trajectory that is shown in the MatLab environment as shown in fig. 4.29.

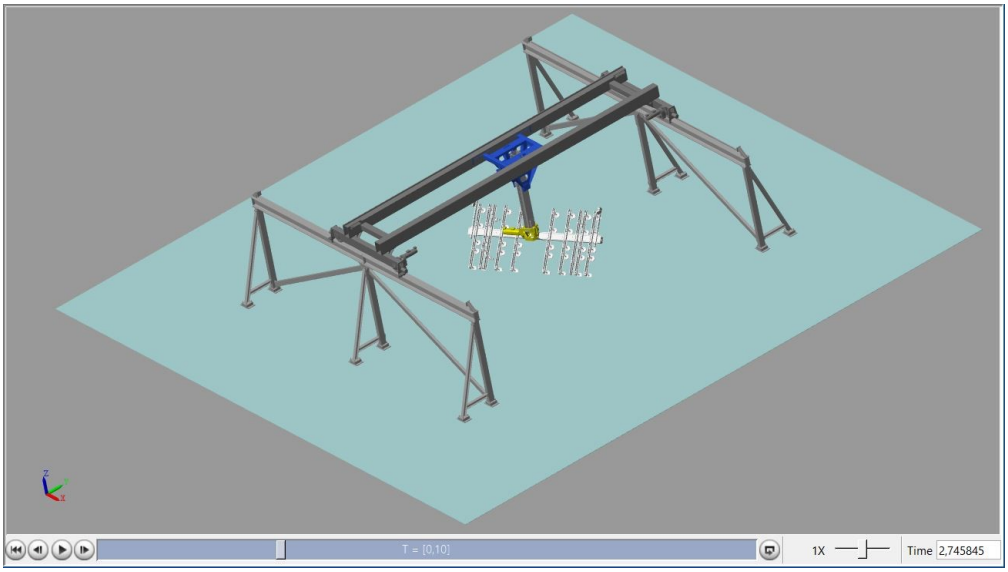


Figure 4.29: SimScape Multi-body model 3D visualization

Possible adjustments to this multi-body model are the experimental tuning of the stiffness and damping coefficients of the several joints, but this additional experimentation is not the focus of this thesis.

Chapter 5

Control architecture of the system

After having analysed the mechanical and trajectory behaviour of the gantry stacker, the following step described in this chapter is the development of the hardware and software architecture for the control algorithms that govern this machinery.

5.1 Overview of the hardware adopted

The general architecture of this machinery is composed by a central CPU (model: Siemens ET200SP) which elaborates the inputs coming from the several sensors, the communication with the other PLC networked in the industry plant and the Human Machine Interface adopted (i.e. WeinTek HMI or Siemens T900 Comfort).

This CPU interacts with the SIMOTION D 425-2 Device which is a driver Control Unit with trivial combinatory logic functionalities and motion control algorithms. This choice is adopted to exploit its extremely high performance characteristics to satisfy fast and precise cycle times. This Control Unit has the possibility to govern several Dirve-CLiQ communication interfaces which allows a simpler transmission of data from and encoders and motors.

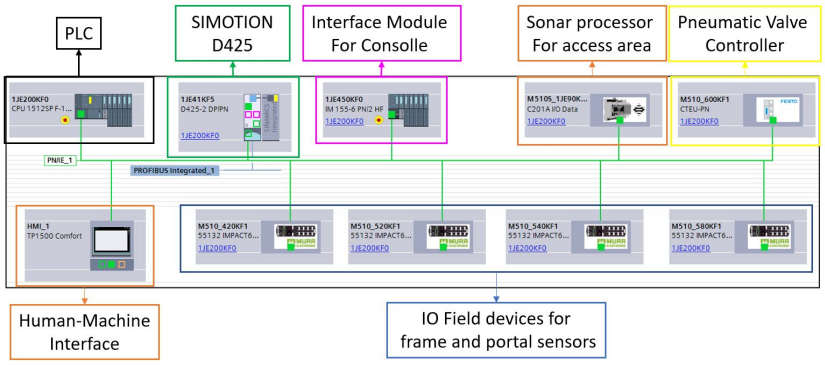


Figure 5.1: Hardware configuration of the gantry stacker

In addition, several hardware components are adopted to define the IOs corresponding to different areas of the gantry. Here are listed the principal elements:

- **Interface module**: this device allows to have additional IO cards in a remote position with respect to the PLC. This is typically used to manage the inputs coming from the console governed by the human operator and its relative output lights for the information management.
- **Pneumatic valve controller**: This device is the interface between the control logic signals and the pneumatic commands. With this valve controller it is possible to decide the vacuum and blow intervals, but also the inclusion/exclusion of the cup groups and the working behaviour of the grippers.
- **IO Field devices**: this devices are typically used to obtain digital and analog IOs from remote zones of the gantry. As an example, Murr Elektronik sockets are put on the bridge and carriage to obtain the signals from the overstrokes and mechanical cams that are in the upper physical positions.
- **Radar micro-processor for access area**: This device is adopted to govern the accesses controlled by several radars for safety countermeasures.

This basic Hardware is the basis adopted in this type of automation system, with additional components inserted according to particular specific design. In general, the disposal of this components for the testing of complete automation system is available only on field during the installation phase on-site. For this reason, the commissioning phase typically starts from here.

5.2 Siemens TIA Portal - ET200SP CPU

In the following section, an overview of the developed software is performed and its relative Human-Machine-Interface management screens are shown.

The working principle of a PLC software program is realized through sofware blocks. There exist four types of blocks [34]:

- **Organization Blocks** (OB) define the overall structure of the user program and are divided into subcategories that allow to decide how the tasks are recalled, cyclically or at a particular event (i.e. in case of Hardware failure). This type of block also includes the Main block, which is principal program in which all the subtasks are recalled cyclically in a maximum cycle time.
- **Function Blocks** (FB) that are code blocks that store their values permanently in their instance data blocks. This functions allows to have a dedicated memory for a particular program code that remains available after the block execution.
- **Function calls** (FC) are code blocks or subroutines without dedicated memory, where the variables are defined outside the function recall, but the memory space occupied is not linked to the FC.
- **Data Blocks** (DB) are used to store program data and thus contain variable data that is used by the user program. Global data blocks store data that can be used by all other blocks. In this category fall also the instance Data Blocks created by the FB.

In addition, variables can not only be saved within Data Blocks but also within tag tables. In this case it is necessary to assign them an address inside the CPU. These variables are divided into three categories: Inputs, Outputs and Merker memories.

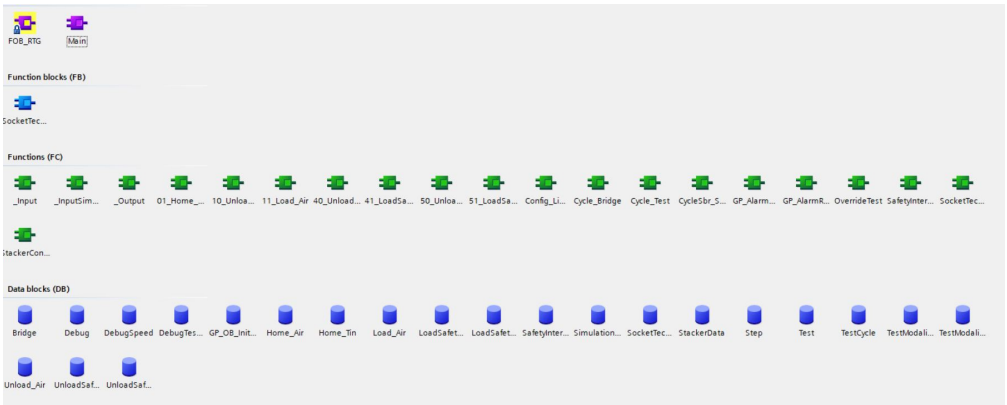


Figure 5.2: Example of program blocks

Concerning inputs and outputs, the address in the PLC is associated to the channel physically wired to the given signal, while for the Merkers a series of addresses available. The programming languages adopted are SCL (Structured Code Language) and the ladder code.

Input and Output

Given the program code, its results depend on the several digital inputs coming from the physical signals, but also the HMI control. In the main block, a good practice is the wiring of the input and output signals in auxiliary variables inside the interested DB. To do so, an input function call is required. In addition, to retrieve the information and commands from the HMI panel a similar solution is developed. Similarly, physical outputs and variables read from the HMI are written by symmetric FC (called "Outputs" and "HMI_Read") [35].

Access area control

The following subroutine recalled concerns the access area management. In facts, to maintain the proper level of safety in accordance to the industry compliance, Safety Inputs require an additional control on the consistency of the signal channel. Access doors and photocells signals are typically F-DI (safety inputs). However, in order to perform maintenance or changeovers, it is required to cut these barriers without setting the system in emergency. To do so, there are access area selectors with safety key in order to consent the transit. This accesses are observed and summarised in a relative HMI screen as shown in fig. 5.3.

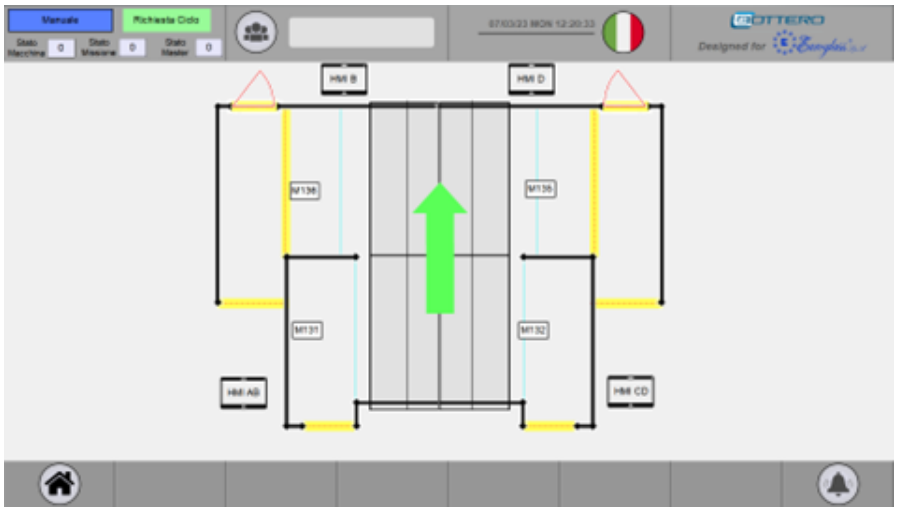


Figure 5.3: HMI panel: Access area

Rack counter plate

A fundamental part of the gantry stacker software is the production. In these functions, the number of lites and plates are defined and the stacker works in order to complete correctly the several packs present in a rack. Pack composition can be

defined by external production manager or from local HMI panel and it is possible to perform, several forcing commands like the decrement of a plate (in case of broken glass sheet) or the forced end rack. A general screen for rack counter control is depicted in fig. 5.4.

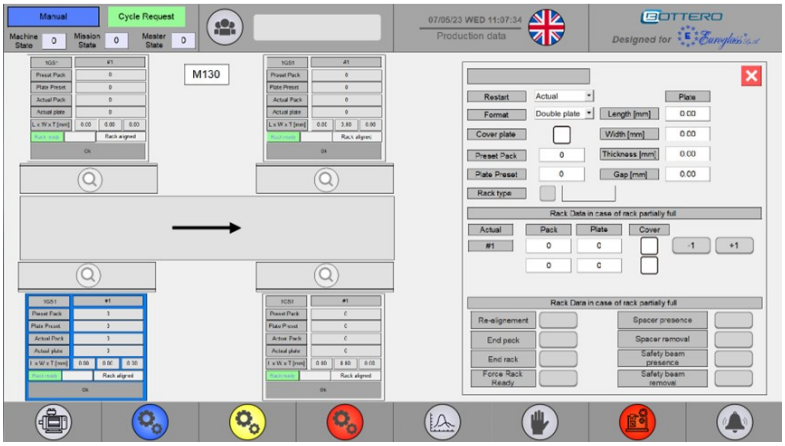


Figure 5.4: HMI panel: Rack counter plate

Rack manager point

In addition, in order to have faster time cycles, the gantry stacker is capable of predicting the position of the next load/unload on the rack depending on the geometrical configuration of the rack and the material thickness deposited or taken. Typical parameters are defined in an operator HMI screen as shown in fig. 5.5.

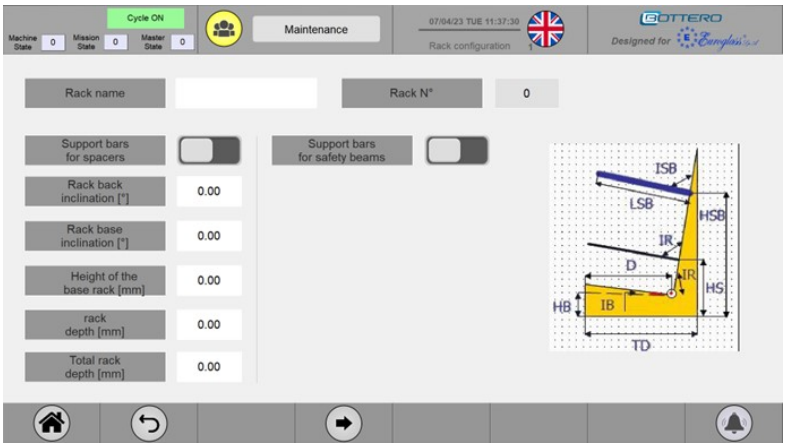


Figure 5.5: HMI panel: geometrical definition of the rack

Motion mission control

The motion of this machinery is defined in different cycles consisting on several phases. The first one is the homing cycle, which is a fundamental trajectory capable of returning the system in a well-known and predefined condition. This cycle is defined through fixed points which are out of encumbrance with respect to conveyors, racks and other obstacles present. Each phase is completed once fulfilled a given condition and is then reset and the cycle movement proceed to the following one. Other cycles consist of Load Air, Unload Air, Cycle from rack, Cycle from conveyor and so on.

Pneumatic control

Another important program area is the pneumatic control definition. Physically, the cups are regulated by air pressure and can be inserted or excluded according to a manual selection or an automatic distinction depending on the dimensions of the glass plates handled. Example of this control is shown in fig. 5.6.

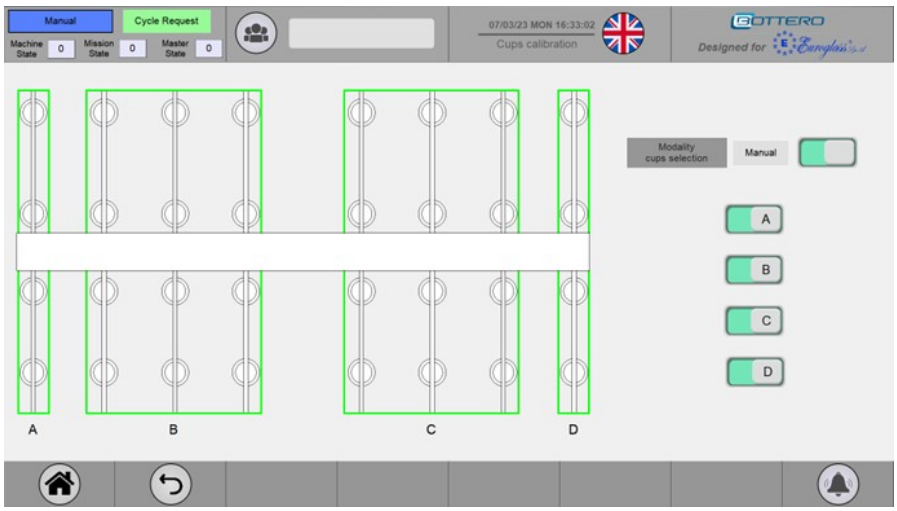


Figure 5.6: HMI panel: cups selection

Communication

The last important aspect is the communication of this network with the other control units and managers. Typically, the communication between different CPUs is developed through the I-Device protocol, which is a particular function of Siemens CPU where a master controller defines the memory areas which are shared between the control units. In particular, each transmission is done by allocating a given memory reserve for receiving and another for transmitting. In order to have the

correct communication, the standard adopted is the definition of a proper UDT (User Defined Datatype) which is a particular structure composed by multiple variables (independent on the type of the single ones). This generated structure is used to map the common I-Device memory from the PLC that send the information, but also from the PLC that receive the data. Another possible communication protocol is the standard TCP/IP communication, where two control units communicate on the IP network. This communication is possible with the functions `TSEND_C` and `TRECV_C` that transmit and receive data on a given port at a given IP address.

The development of the code is only part of the software definition, since a debugging and testing part is fundamental to avoid bugs and to consider all the possible cases. A preliminary stage can be performed by simulating brief portions of the software and by emulating inputs and outputs and seeing the behaviour of the program code. This is the so-called "Soft commissioning".

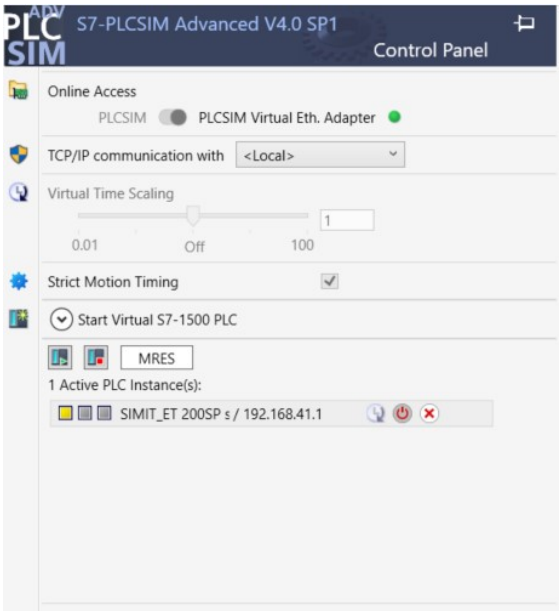


Figure 5.7: Example of a PLCSIM Advanced instance

This debugging phase can be performed with a physical CPU connected to the ethernet network or with an emulated instance with the software `PLCSIM Advanced` [36].

This program is the simulation software able to emulate the behaviour of PLCs from the families SIMATIC 1500 and ET 200SP series in order to test the correct functioning of the control algorithm. Instead of a hardware connection with the PLC network, the software provides two main communication interfaces: Local (PLCSIM) and with a Virtual Ethernet Adapter [37].

5.3 SIMOTION Control Unit

`Simotion Scout` is the development ambient for the drive commands and the definition of the synchronized movement of the axes. The execution system of the Simotion control system is based on the multi-tasking system principle [38].

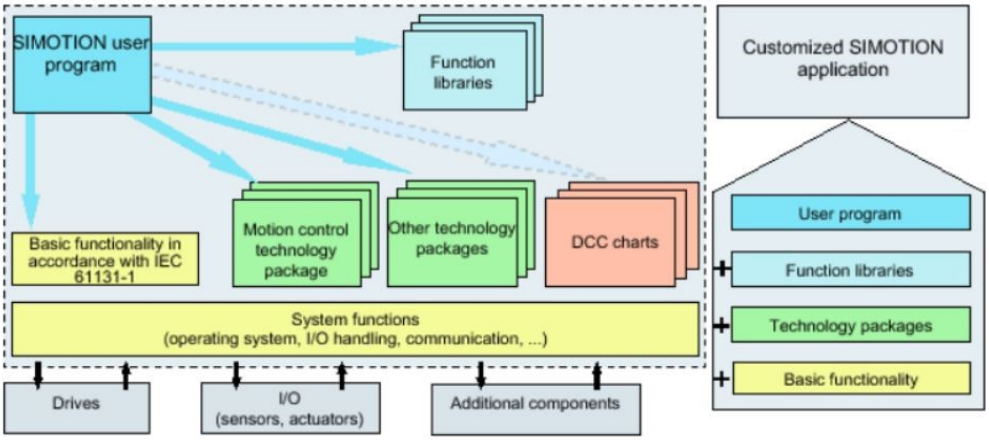


Figure 5.8: General architecture of the SIMOTION Control Unit

The general control algorithm is defined by tasks which are recalled in three different ways:

- **Background tasks**: Cyclically processed blocks of the program
- **time-triggered**: synchronous compilation of the function at a fixed time interval
- **interrupt-triggered**: tasks called by the fulfillment of a certain event with its priority

The algorithm principle requires that the drive commands must be recalled once and thus a case status code is developed according to the feedback state from the technological objects.

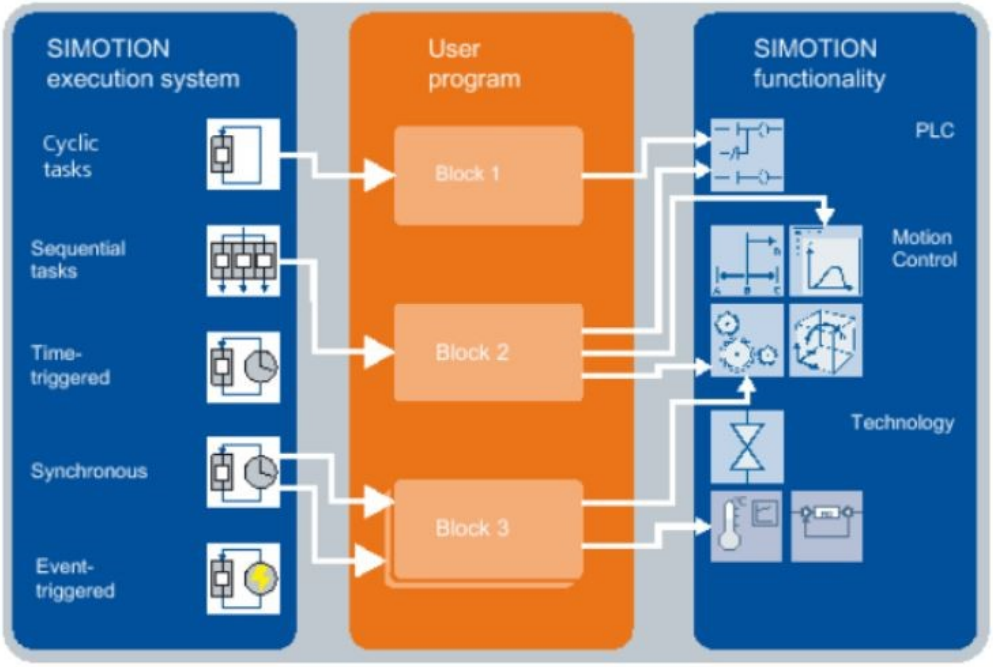


Figure 5.9: Simotion execution system principle

Technology Objects (TO)

Technological objects are the black boxes that provide the functionality for motion and technical control. A technology object (TO) represents a technological functionality in the SIMOTION user software. As shown in fig. 5.10 the technological object is the interactive link between sensors, actuators (Hardware components) and the command and parametrization of the software counterpart. Moreover, each technology object provides a set of alarms and value statuses in order to better understand the behaviour and the reaction of the motion system.

Technology Objects can be activated or deactivated from the program and so it defines an intrinsic adaptation to the several machine configurations (i.e. axis not required, synchronised motion not necessary). Typical TOs are:

- **TO axis** for drive and encoder
- **TO external encoder** for one encoder only
- **TO Following Object** for the synchronous operation between two axes
- **TO cam** for the definition and representation of complex programmable functions (adopted for trajectories)

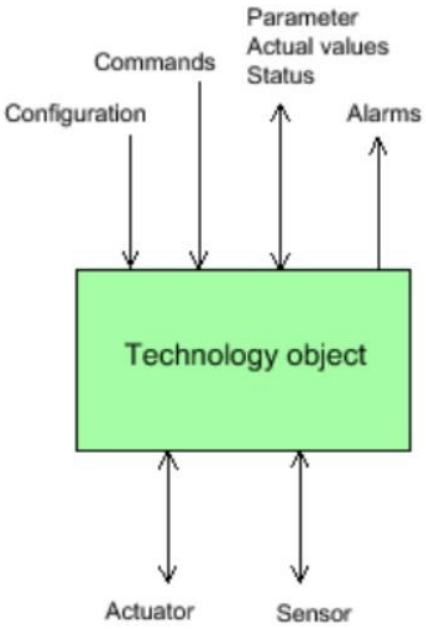


Figure 5.10: General scheme of a Technology Object

Programming model

The commands are connected to respective tasks which are assigned to the execution levels of the motion control system. The execution time of a general motion command on the technology object depends on the nature of the action to be performed. In general, the commands issued from the user program, the user program tasks, can be categorized as follows:

- Commands which are executed immediately in the context/sequence of the user program tasks These are handled like a function within the user program tasks. These commands are synchronous since the user program is continued only upon the return of the function result.
- Commands which are queued in a command buffer and which overwrite each other due to higher priority
- Commands which are entered in a command buffer and which are rejected if the command buffer is occupied

In a further analysis of the execution task system and its levels, it is possible to distinguish between sequential tasks and cyclic tasks.

After the recalling, sequential tasks (i.e. Motion Tasks) are executed only once and then are terminated. In this way, the functions and the commands that are

coded in this section are singularly executed. IN addition, at each new instantiation, all the local variables are initialized. Since sequential tasks do not considering any time monitoring, their execution time can be extended as desired by the program. Sequential tasks are subject only to the control of the application developed, in facts, the user program can start, stop, pause and resume tasks. The general principle in this type of tasks is the consecution of single commands that are usually performed when the previous task has been completed. The step enabling condition that must be implemented to obtain this behaviour is `WHEN_COMMAND_DONE` in the next Command parameter of the successive command. This execution model is typically called synchronous execution.

On the other hand, cyclic tasks (such as the Background Task) are periodically recalled immediately after their completion or after a certain time interval. In this case, the static variables of the program belong to a retentive memory. Cyclic tasks are subjected to time monitoring and a pre-defined error response is performed in case of the exceeding of the maximum execution time. As a result, the code contained in cyclic tasks must be efficient and quick. This characteristics are not typical of motion commands (i. e. positioning of an axis), where the completion is achieved after several clock cycles. With respect to sequential tasks, cyclical TO commands have the parameter `IMMEDIATELY` in the step enabling condition for the next Command parameter. This procedure is called instead asynchronous execution.

The SIMOTION execution system comprises System task and user-defined tasks. Each task is subjected to a level of priority and associated execution level. System tasks are regularly executed by the system and comprehends principally communication and motion control and are performed at different levels, from the isochronous bus base cycle up to faster IPO cycles, for the interpolation calculation of setpoints. On the opposite, User applications can be defined as event-controlled execution tasks (so called interrupt tasks) or startup/stop tasks events. Although, the basic user applications belongs to the freely running execution level, where a Round Robin priority process execute background and the motion tasks. A typical time cycle is shown in fig. 5.11.

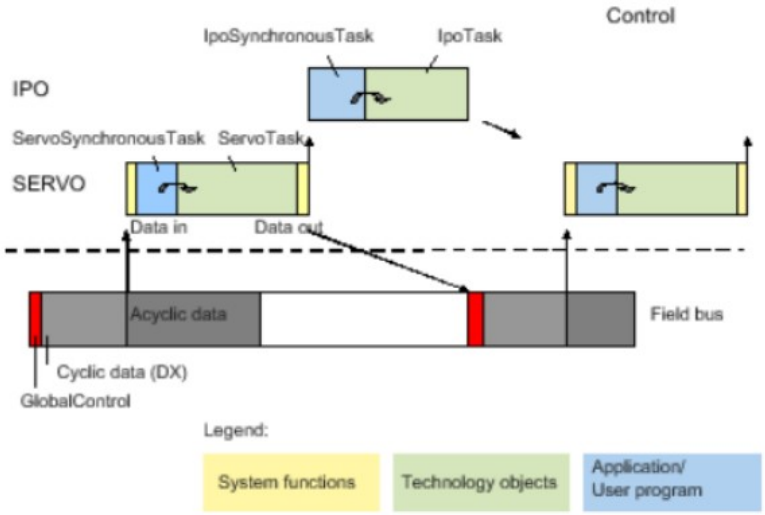


Figure 5.11: Definition of the time usage in the execution system

The SIMOTION D Control Unit is the device under investigation for the motion control. SIMOTION D is a drive-based version of SIMOTION based on the SINAMICS S120 drive family. The SIMOTION D425-2 is particularly designed to fulfill the needs of applications with many coordinated axes with high precision at high speeds. Typical applications include:

- Compact multiple-axis machines
- High-performance applications with short machine cycles
- Distributed drive concepts
- Applications with multiple axes

This control unit is general connected to other hardware components that are governed by this central brain. In facts, SIMOTION D represents the Control Unit consisting of the SIMOTION runtime system (operative system that execute the applications) and the SINAMICS drive control (interaction and governance of the drives and the axes). Attached to the CU is then connected the SINAMICS Integrated drive with various SINAMICS S120 drive modules to perform open-loop and closed-loop control of the axis movement. Other additional components that can be connected are the Sensor Modules (encoder systems) that are easily connected with the DRIVE-CLiQ interface as shown in fig. 5.12.

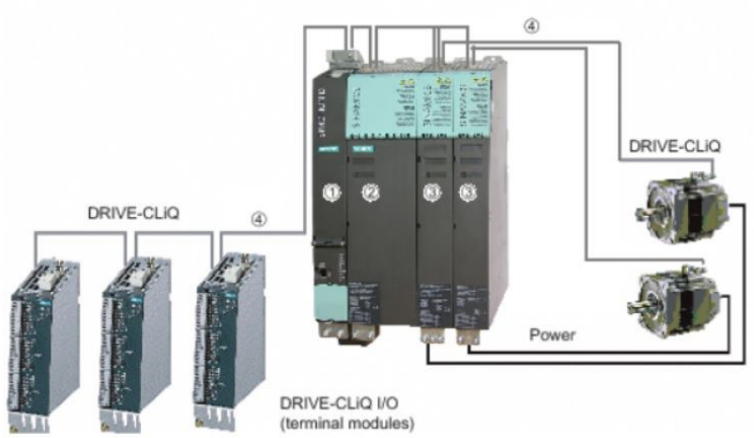


Figure 5.12: Example of SIMOTION D425-2 Architecture

Software development in SIMOTION SCOUT

In the following paragraphs, the developed code for the motion control algorithm is analysed. For sake of simplicity, the principal codes developed are available in the Appendix A.

For the complete development of the software, several elements need to be defined. To start, the technological objects that characterise the project must be created and parametrized. Initially, it is important to configure all the possible axis to be controlled with the Simotion Control Unit. Each axis can be defined as a speed control, a positioning axis or a possible synchronous operation axis. This distinction accounts the possible methods of axis movement and its intrinsic feedback control. Additional parameters to be set regard the type of axis (electrical, hydraulic or virtual) and its driver assignment with, if necessary, its relative encoder for the closed-loop positioning control. For the simulation purposes, all these axis are then simulated, since without the real Hardware it is not possible to emulate the relative drive component.

Another fundamental technological object is the cam. This structure can be used to define a transmission function and apply it to other technology objects. In general, a cam describes the dependency of an output variable on an input variable, as $y = f(x)$. Typical input variables are the actual position of a master axis, a virtual master value source, or the time, that control output variables like the set position of a following axis, the setpoint profile, or the velocity profile.The cam is defined as a stand-alone technology object, but its working principle is satisfied when interconnected with other technology objects.

After defining these TOs and their relative connections, the following step is the definition of the tasks and the development of the programs. The strategy adopted

in this code is the background management of the motion missions received by the controller. As an additional background task, the control on the abilitation of the several axis is performed.

The simplest motion function that is developed is the possibility of a manual jog. In this case, the operator (for maintenance and service purposes) requires to perform a movement on the singular axis. From a communication point of view, the Simotion receives the axes affected and their relative commands of positive or negative jogs (typically controlled by physical push buttons on the console). Once the system is set in its manual condition, it investigates the existence of a manual command and then executes a positioning towards the motion limits, that can be software-defined or, in case of maintenance access level, consistent with the hardware-controlled endstops. This positioning motion continues until the button is pushed. Once released, the motion is immediately paused and, after a certain predefined idle time, the motion command is stopped. The results and the analysis of this commands and motion variables can be obtained with the powerful trace system of the simotion scout, where it is possible to observe the queue of the motion commands, the states of the technology objects and the kinematic variables of the several axis. This possibility is detailed in fig. 5.13, where a consecution of the manual jog is performed.

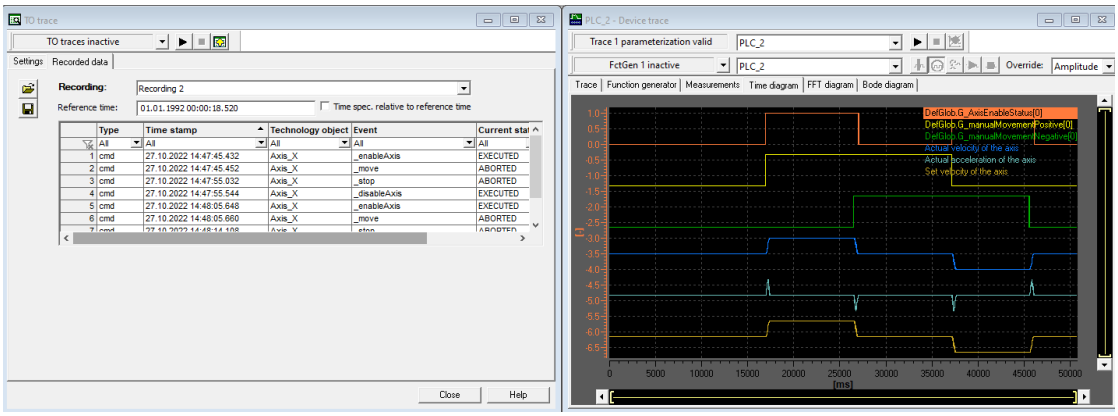


Figure 5.13: Trace and TO trace of a manual jog simulation

The movement of a single axis can be also performed by defining its velocity-profile or position-time profile. To do so, the Technology Object Cam plays a fundamental role. In facts, thanks to the definition of a function profile, it is possible to command an axes with a `_runPositionLockedVelocityProfile` or a `_runTimeLockedPositionProfile` that is synchronised with a cam. The definition of a cam requires a specific function program, where, starting from the waypoints and their relative geometric characteristics draws a graph profile. The parameters required for each point can be summarized as the following:

- **Master point position** (X): it represents the setpoint of the master variable for which an image in the slave domain
- **Slave point position** (Y): represents the value of the defined profile at the X master position. In general $Y = f(X)$
- **Geometrical velocity** (m): it represents the first derivative coefficient of the profile at its relative waypoint. Considering a cam drawn for a position-time profile it represents the instantaneous velocity detected at the given point.
- **Geometrical acceleration** (k): it represents instead the second order derivative of the function at the given point.
- **Profile**: this parameter represents the type of interpolation that the function exploit for the connection between the several segments. The possible profile selection comprehends the polynomial functions (straight, cubic and quintic polynomial), the harmonic functions (cicloidal and acicloidal).

Simotion Scout offers the possibility of the definition of a cam through the native Siemens library of technology commands or through a cam edit tool. With the former, it is possible to redesign on-line profiles with the modification of the input points, while with the latter, a drawing graph environment helps to better understand the behaviour of the function profile. In addition, it is possible to evaluate and analyse the cam calculated in the online project as shown in fig. 5.14.

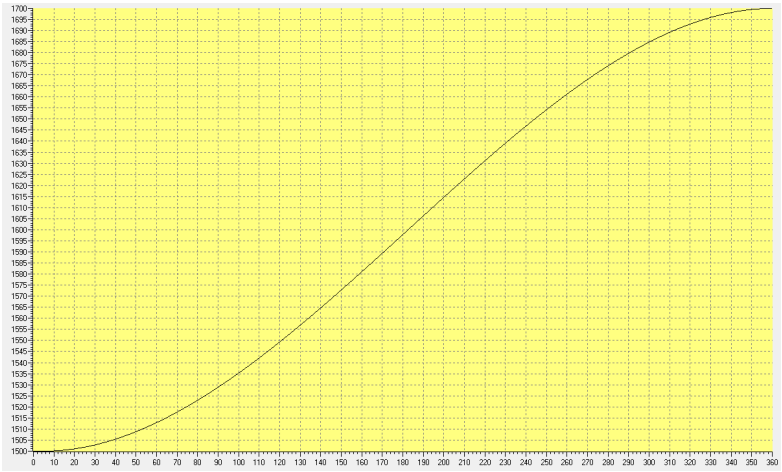


Figure 5.14: Example of a Simotion Cam

A complete different approach regards the synchronised movement of multiple axis. In this concept, the cam is the interconnection object between two axis. In a simpler case, a slave axis needs to perform a motion in accordance with the position

movement of a master axis, so the slave axis requires to be in camming with the master. The motion of the master axis causes a synchronised motion of the slave one following the cam profile. For complex applications, a synchronised motion of multiple axis is performed with the use of a virtual master axis that leads the motion of the following synchronised objects. It is important to stress the correct tuning of the several cams which are drawn separately in order to perform the desired movement.

From this point, the possible implementations vary according to the needs of the general trajectory planning and completion. A possible example is the start-stop-restart of a synchronised motion of multiple axis. In this case the positioning command on the virtual axis is stopped and resumed by external commands and an example of this behaviour is shown in fig. 5.15.

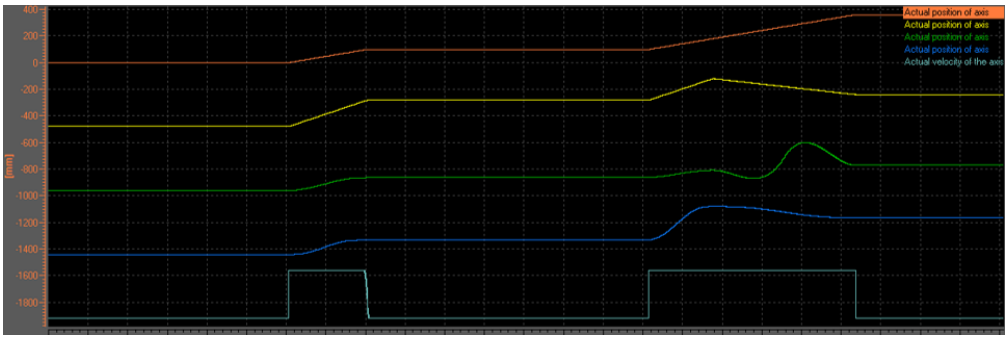


Figure 5.15: Start-Stop on a synchronised cam

Another possible development is the application of two consecutive cams. This solution allows to design different trajectory segments that can be travelled in sequence without the loss of time. This solution is briefly depicted in fig. 5.16 where the light blue line corresponding to the synchronisation status of the cam interconnection is immediately reattached, while the position profile is changed according to the two designed profiles (first one is a straight profile, while the second one is a cubic polynomial function).

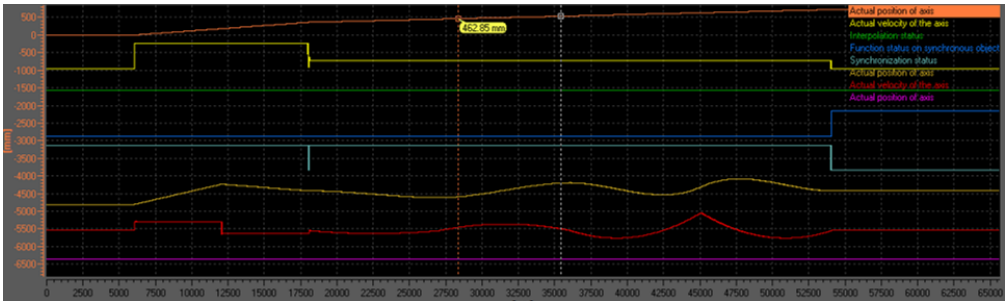


Figure 5.16: Travelling of consecutive cams

Other motion strategies are represented by the manual jog on the cam, with the possibility of running in forward and backward the gantry in order to help the operator in the maintenance and control of critical motions. Proper simulations and debugging of this motion control program is performed with the SIMOSIM software, capable of emulating the instances of the Simotion control units.

5.4 SIMIT

In the previous sections of this chapter, the single environments and codes were analysed. However, an important aspect of the whole software architecture system that was hidden regards the interconnection between the two control systems and the communication. In addition, concerning the simulation of several elements external to the control units (i.e. input sensors, output actuators, encoders), the only alternative was the soft-commissioning of specific algorithms of the program [39].

Instead, studies of cycle operations, feasibility and reliability of the automated system require an analysis on the final design as a complete automation system. Thanks to the simulation software `Siemens SIMIT`, it is possible to connect to a single simulation platform all the virtual components investigated to complete comprehensive tests of automation projects.

This simulation platform represents the milestone for the virtual commissioning, where the designed automation system interacts with the simulated environment and it can be subjected to operator sequence of manual tasks for a training and time-study analysis.

The principal benefits coming from a `Siemens SIMIT` solution are the following:

- Improved engineering and higher automation quality;
- Detection and correction of automation errors in upstream phases;
- Faster commissioning and start-ups with reduced risks;

- Increased plant availability and safety throughout the entire life cycle
- Operator training with actual commands prior to plant start-up
- Risk-free testing of optimization and alternative solutions
- Study and development of expansion projects of actual designs

`Siemens SIMIT` software is able to link all necessary couplings for communication between the simulation and automation environment. In facts, it permits communication with real hardware controllers through the use of a SIMIT unit, thus developing a HiL (Hardware-in-the Loop) system.

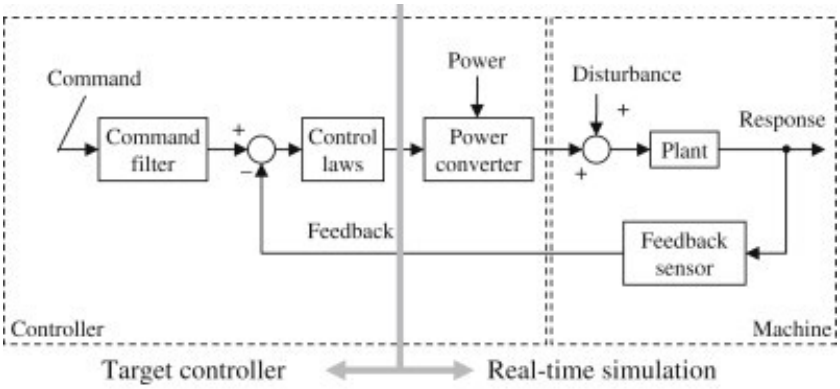


Figure 5.17: Hardware in the loop scheme

The Hardware-in-the-Loop testing is a procedure where real signals coming from a controller are connected to the test system that simulates reality, eluding the controller into computing logic and executing tasks in the assembled virtual environment. Test and design iteration take place as though the real-world system interacts with the controller and so it is possible to run through thousands of possible scenarios to properly exercise the logic of the controller without the cost, time and risks associated with actual physical tests.

For this analysis to be coherent with reality, the quality of the simulation software is of utmost importance. Simulation software must be paired with hardware that not only accounts for system specifications such as connector type and I/O but also allows for fault insertion and the ability to test real-world scenarios.

On the other hand, there is the possibility of a Software-in-the-loop analysis (SiL) where each component is simulated, comprehending the controllers via emulators such as `SIMATIC S7-PLCSIM Advanced` and `SIMOSIM Advanced`.

The Software-in-the-Loop represents the integration of compiled production source code into a mathematical model simulation, in a virtual environment for

the development and testing of detailed control strategies for large and complex systems.

With SiL, all the components can be emulated on the computer and the whole system can be iteratively tested and, according to its response, modify the source code, by directly connecting software to a digital plant model substituting for costlier systems, prototypes or test benches. SiL makes it possible to test software prior to the initialization of the hardware prototyping phase, significantly accelerating the development cycle.

SiL enables the earliest detection of system-level defects or bugs, significantly reducing the costs of later stage troubleshooting, when the number and complexity of component interactions is greater. SiL provides an excellent complement to traditional Hardware-in-the-Loop (HiL) simulation, while helping to accelerate time-to-market and ensuring the more efficient software development.

Software-in-the Loop is the technical solution adopted for the virtual commissioning of this gantry stacker.

Development of the SIMIT project

The `SIMIT` application is a field environment simulation software that can be used for different applications:

- Complete plant simulation, comprehending the emulation of signals, devices and plant response;
- Input and output simulator of test signals for an automation controller;
- Testing and commissioning of the automation software

These steps can be progressively performed, starting from the testing of signals and simulation models up to the complete virtualization of the overall system in order to obtain the dynamic responses of the plant. In `Siemens SIMIT` there exist basically three macro-elements, that can be listed as following:

- **Couplings** are the interfaces to the automation system and is required for signal exchange.
- **Charts** represent the control logic of the inputs and the external sensors and actuator of the system not simulated by the control units.
- **Visualization Objects** are the tools to obtain an overview of the signals from the plant with simple graphical objects or control elements

The general structure of the simulation is defined in charts with components, controls and connections and is linked to the coupling with connectors. The charts

contain the simulation model. In these graphical representations, the components, controls and connections compose the behaviour of the surroundings . As previously stated, charts consist of the following elements:

- Components for logical and arithmetic functions and for drives, sensors, connections and communication.
- Controls for specifying values in a current simulation or displaying values derived from the control logic.
- Connections are defined with signals and are represented as wiring connections or as connectors. Signals are the interconnection systems between components, controls and coupling of the automation system.

The first step to be done for the development of a simulation platform for the testing of the project is the definition of the couplings. Couplings are used by `SIMIT` to communicate with the automation system. A coupling is an interface between the simulation environment and a coupling partner (i.e. an automation system or an emulated virtual controller). In general, the coupling fulfills the following tasks:

- Signal exchange with the coupling partner (The input/output signals (I/O signals) of the coupling partner are exchanged with SIMIT over the coupling)
- Coordination of signal exchange between SIMIT and the coupling partner

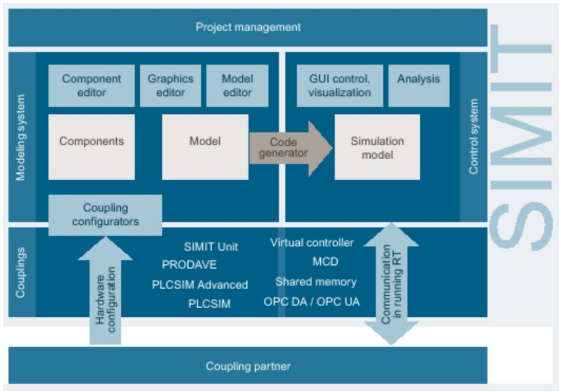


Figure 5.18: Simit working architecture

Each simulation system consists of two regions, which can briefly summarized in the modeling system definition (where to configure the interfaces to the coupling partners) and the control system running simulation. In this context, the coupling provides the connection between the simulation model run by the control system and the coupling partner variables.

In the developed project, the coupling adopted is related to the PLCSIM Advanced coupling, which, thanks to the HWCN Export, it is possible to create the emulation of two instances related to the TIA E200SP CPU (with a PLCSIM Advanced Instance) and to the SIMOTION D425-2 (with a SIMOSIM Advanced Instance). From this import, all the Inputs and Outputs present in the two control units are the initial signals that can be used in the charts paragraphs. As an example, it is possible to govern the several inputs that affect the complete automation. With a simple chart, it is possible, for example, the testing and validation of the safety area accesses.

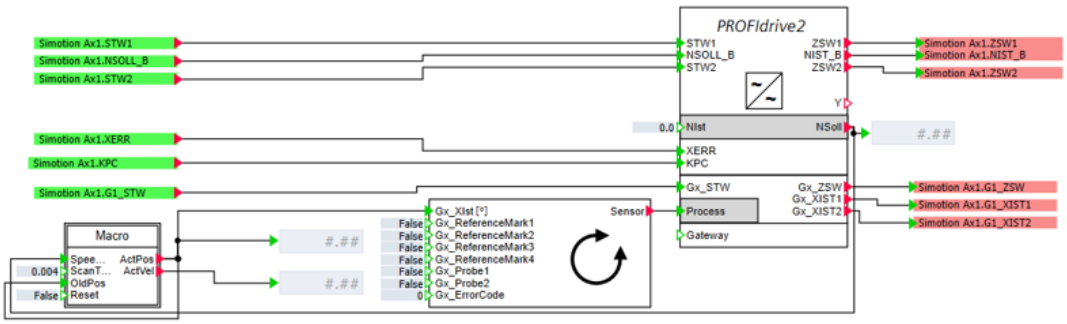


Figure 5.19: Drive chart emulation

The focal point of this SIMIT project consists in the emulation of the drives, in order to obtain the correct motion control, as shown in fig. 5.19. In principles, motion drives communicate and act through predefined telegrams composed of bits and words with precise meanings and parameters. To emulate this, it is necessary to choose the correct drive accordingly to their control (i.e. Closed loop control). The analysed system is principally composed by drive that are described by the `PROFIdrive2`, which is a popular Siemens standard. From a chart point of view, it is important to understand the presence of an emulated encoder (here called sensor) that is fed by a macro that integrates the output velocity of the drive, thus retrieving the position of the axis. A general description of the PZD words exchanged between the Simotion Control Unit and the SIMIT environment is well described in the graph 5.20.

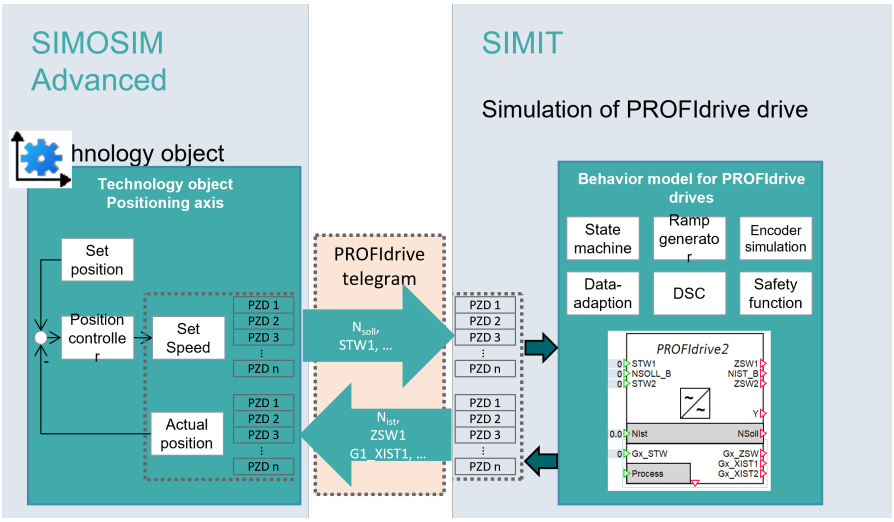


Figure 5.20: communication between SIMIT and Simotion control unit

With this basis developed, it is now possible to start the virtual commissioning with Software-in-the-Loop of the automation system of the gantry stacker. For a better understanding of the motion trajectories and mission cycles, the SIMIT software has been also coupled with `Siemens Tecnomatix Plant Simulation`, which is a process simulator for the workflow of multiple machines, as shown in fig. 5.21. This software is not designed for the digital twin concept of a single automation system, but it is used to investigate the behaviour of a complete manufacturing line.

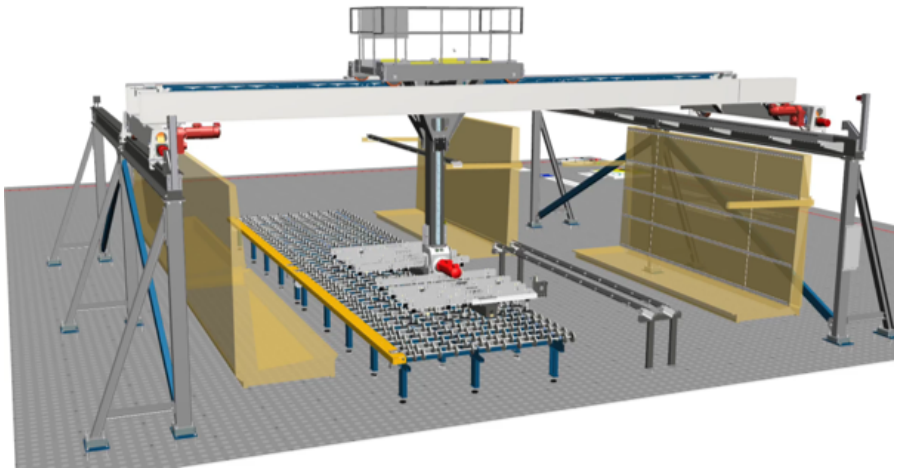


Figure 5.21: Virtual commissioning with Siemens Tecnomatix

Chapter 6

Conclusions

The aim of thesis was the creation and the development of the instruments that, combined together, form the digital twin concept the gantry stacker for glass plate handling.

In facts, starting from the basic CAD geometries of a previously designed machine, a simple kinematic model was developed in order to investigate the general behaviour and check the trajectory feasibility for several production layouts. Once accepted these waypoints, a theoretical and experimental analysis was developed for the advantages and drawbacks concerning the trajectory planning strategies. To obtain these results, a `SimScape Multi-body model` was created and the different motion profiles were run at the several joints of the kinematic chain. In general, the chosen trajectory profile represents the trade-off of different observed parameters, such the dynamic resonance, the required time and the correct exploitation of the actuators power. Depending on the nature of the mission to be performed, a particular solution was adopted.

Moving forward, this project steered to the automation sphere. A basic complete automation software was completed, starting from the PLC control program that governs the logic of all the aspects (aside from the motion) of the gantry stacker. In addition, an HMI interface was developed in order to ensure the operator interaction with the automation system.

Instead, the motion control system was programmed for a different hardware component, the SIMOTION control unit. In this code, the possibility of synchronised movements between different axis was investigated and several commands were accounted for and developed.

From a project point of view, the classical design would have required a real commissioning, where the complete machinery was installed in a test factory building with the disposal of the hardware components of the network and a sufficient supply power. This test phase is mandatory before the installation phase in order to the debug the coded software and to highlight the critical points of the

overall design. Apart from the expensive costs required for the shipment, mounting and renting costs of this solution, this tests are performed in late stages of the design process, where the possible changes are costly and time-critical.

The virtual commissioning is the solution adopted in our project that does not rely on the physical components and represents one of the pillars of Industry 4.0. The missing element for the complete emulation of the automation software was the definition of the simulation environment and the interconnection between the control units. To fill this gap, a SIMIT project was developed and the debug of the software was performed. In addition, a coupling with a simulation plant software permits a graphical visualization of the motion that the controlled stacker accomplishes.

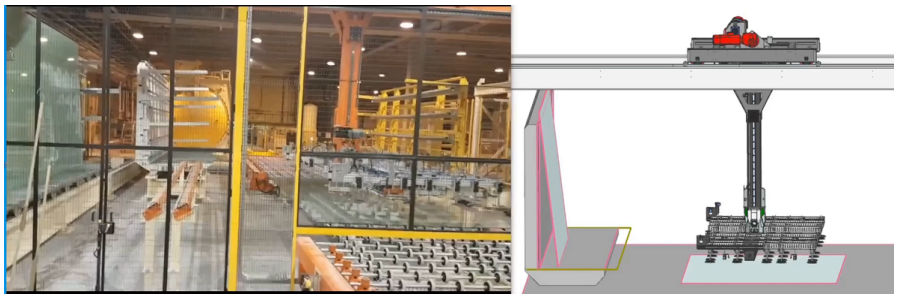


Figure 6.1: Comparison between reality and digital twin (NX MCD)

This collective automation subsystem and the multi-body model represent the two halves of the digital twin concept. In future works, the goal is to relate these two environments, allowing the interactions between the mechanical parts in motion and the several sensors and actuators, to better investigate the complete automation system, from a mechanical, electrical and software point of view.

From this project, Bottero S.p.A., with the adoption of the Siemens native multi-body model (`Siemens NX MCD`), will create the digital twin of a widely used machine, aiming to a revolutionary engineering process that is the starting point of the Industry 4.0 philosophy.

Appendix A

SIMOTION codes

```
1  INTERFACE
2      USEPACKAGE cam;
3      USES DefGlob;
4
5      PROGRAM Enable_Axis;
6      PROGRAM Mode_Selection;
7
8  END_INTERFACE
9
10 IMPLEMENTATION
11     PROGRAM Enable_Axis
12         VAR_TEMP
13             i : INT;
14         END_VAR
15         (*FOr each axis, check the enable status and act consequently
       *)
16         FOR i := 0 TO NR_AXIS-1 BY 1 DO
17             IF (G_EnableAxis[i] XOR G_AxisEnableStatus[i]) THEN
18                 (*If the command requires the axisEnabled, proceed*)
19                 IF (G_EnableAxis[i] AND AxisStruct[i].control =
       INACTIVE) THEN
20                     ErrorAxisEnable[i] := _enableAxis(
21                         axis := AxisStruct[i]);
22                 END_IF;
23                 (*If the command turn off the enabling, check that
       all the axis movements are stopped*)
24                 IF (NOT G_EnableAxis[i] AND AxisStruct[i].control =
       ACTIVE) THEN
25                     IF (AxisStruct[i].motionStateData.motionState <>
       STANDSTILL) THEN
26                         ErrorAxisStop[i] := _stop(
```

```
27                                  axis := AxisStruct[i]
28                                  ,stopMode := STOP_AND_ABORT
29                                  ,stopSpecification := ALL_AXIS_MOTION(*
30                                  ,stopId :=
31                                  ,positiveAccelType :=
32                                  ,positiveAccel :=
33                                  ,negativeAccelType :=
34                                  ,negativeAccel :=
35                                  ,positiveAccelStartJerkType :=
36                                  ,positiveAccelStartJerk :=
37                                  ,positiveAccelEndJerkType :=
38                                  ,positiveAccelEndJerk :=
39                                  ,negativeAccelStartJerkType :=
40                                  ,negativeAccelStartJerk :=
41                                  ,negativeAccelEndJerkType :=
42                                  ,negativeAccelEndJerk :=
43                                  ,velocityProfile :=
44                                  ,mergeMode :=
45                                  ,nextCommand :=
46                                  ,commandId :=
47                                  ,movingMode :=
48                                  ,abortAcceleration := *)
49                              );
50                          ELSE
51                              ErrorAxisEnable[i] := _disableAxis(
52                                  axis := AxisStruct[i]);
53                          END_IF;
54                      END_IF;
55                  END_IF;
56                  (*Memory set for axis status*)
57                  G_AxisEnableStatus[i] := G_EnableAxis[i];
58          END_FOR;
59      END_PROGRAM
60
61      (*Here is selected the working mode of the motion control machine
        *)
62      PROGRAM Mode_Selection
63          VAR_TEMP
64              i:INT;
65          END_VAR
66
67          (*Emergency activated*)
68          IF (G_emergencyStop) THEN
69              G_manualMovement := FALSE;
70              G_CamMovement := FALSE;
71          END_IF;
72
73          (*Manual mode activation*)
74          IF (G_manualMovement) THEN
```

```
75              _startTask(MotionTask_1);
76          ELSE
77              (*Activation of the cam of the mission*)
78              IF (G_CamMovement) THEN
79                  (*Check if there is a new mission present*)
80                  IF (G_newMission) THEN
81                      _startTask(MotionTask_2);
82                  END_IF;
83
84                  (*check for automatic or automatic jog*)
85                  IF (G_missionInCam) THEN
86                      //automatic
87                      IF (G_auto) THEN
88                          _startTask(MotionTask_3);
89                      END_IF;
90                      //manual jog ni te sync cam
91                      IF (G_automaticJog) THEN
92                          _startTask(MotionTask_4);
93                      END_IF;
94                      //cam velocity profile (velociity-space)
95                      IF (G_automaticVelocityPosition) THEN
96                          _startTask(MotionTask_5);
97                      END_IF;
98                      //cam space profile (space-time)
99                      IF (G_automaticPositionTime) THEN
100                         _startTask(MotionTask_6);
101                     END_IF;
102                 END_IF;
103
104                 (*addition of a new cam attached to the actual cam*)
105                 IF (G_attachNewCam) THEN
106                     _startTask(MotionTask_7);
107                 END_IF;
108
109             ELSE
110                 FOR i := 0 TO NR_AXIS-1 DO
111                     G_EnableAxis[i] := FALSE;
112                 END_FOR;
113             END_IF;
114         END_IF;
115
116     END_PROGRAM
117 END_IMPLEMENTATION
```

```
1 INTERFACE
2     USELIB Bottero_01, Cam_01, UDT_gantry;
3     USES DefGlob;
```

```
4
5       PROGRAM automatic;
6       PROGRAM automaticJog;
7       PROGRAM automaticVelocityPosition;
8       PROGRAM automaticPositionTime;
9   END_INTERFACE
10
11  IMPLEMENTATION
12      PROGRAM automatic
13          //se non sono ancora in movimento ed ho un segnale di
        partenza, attivo il posizionamento del master fino a fine camma
14          IF (g_automaticMovement = FALSE AND G_automaticStart) THEN
15              ErrorAutomaticPos := _pos(
16                  axis := Virtual_Master
17                  ,direction := POSITIVE
18                  ,positioningMode := ABSOLUTE
19                  ,position := G_rMasterDistance
20                  ,velocityType := DIRECT
21                  ,velocity := G_rMasterVelocity
22                  (*,positiveAccelType :=
23                  ,positiveAccel :=
24                  ,negativeAccelType :=
25                  ,negativeAccel :=
26                  ,positiveAccelStartJerkType :=
27                  ,positiveAccelStartJerk :=
28                  ,positiveAccelEndJerkType :=
29                  ,positiveAccelEndJerk :=
30                  ,negativeAccelStartJerkType :=
31                  ,negativeAccelStartJerk :=
32                  ,negativeAccelEndJerkType :=
33                  ,negativeAccelEndJerk :=
34                  ,velocityProfile :=
35                  ,blendingMode :=
36                  ,mergeMode :=
37                  ,nextCommand :=
38                  ,commandId :=
39                  ,abortAcceleration :=*)
40              );
41              //variabile ausiliaria sul movimento del master
42              G_automaticMovement := TRUE;
43          END_IF;
44          //se è in movimento, ma ho lo stop del ciclo, eseguo il _stop
        (senza abort per poter continuare il movimento)
45          IF (G_automaticMovement AND G_automaticStop) THEN
46              ErrorAutomaticStop := _stop(
47                  axis := Virtual_Master
48                  ,stopMode := STOP_WITHOUT_ABORT
49                  ,stopSpecification := ALL_AXIS_MOTION
50                    //   ,stopId :=
```

```
51                  ,positiveAccelType := DIRECT
52                  ,positiveAccel := 100
53                  ,negativeAccelType := DIRECT
54                  ,negativeAccel := 100
55                  ,positiveAccelStartJerkType := DIRECT
56                  ,positiveAccelStartJerk :=100
57                  ,positiveAccelEndJerkType := DIRECT
58                  ,positiveAccelEndJerk := 100
59                  ,negativeAccelStartJerkType := DIRECT
60                  ,negativeAccelStartJerk := 100
61                  ,negativeAccelEndJerkType := DIRECT
62                  ,negativeAccelEndJerk := 100
63                  ,velocityProfile := SMOOTH
64                  ,mergeMode := IMMEDIATELY
65                  ,nextCommand := IMMEDIATELY
66                    //   ,commandId :=
67                  ,movingMode := CURRENT_MODE
68                  ,abortAcceleration := NO
69              );
70              //variabile ausiliaria sul movimento del master
71              G_automaticMovement := FALSE;
72          END_IF;
73          //se lo stop si è resettato, riattivo il posizionamento del
        master fino a fine camma
74          IF (G_automaticStop = FALSE AND G_automaticMovement = FALSE
        AND G_automaticContinue) THEN
75              ErrorAutomaticContinue := _continue(
76                  axis := Virtual_Master
77                  ,continueSpecification := ALL_AXIS_MOTION
78                  (*,continueId :=
79                  ,nextCommand :=
80                  ,commandId :=*)
81              );
82              //variabile ausiliaria sul movimento del master
83              G_automaticMovement := TRUE;
84          END_IF;
85      END_PROGRAM
86
87
88      PROGRAM automaticJog
89          VAR
90              i : INT;
91              AutomaticJogDirection : EnumDirection;
92              CammingDirection : EnumCammingDirection;
93              AutomaticjogPosition : REAL;
94              AutomaticjogStart : REAL;
95          END_VAR
96          //se è presente un comando di jog positivo/negativo
97          IF (G_automaticJogPositive XOR G_automaticJogNegative) THEN
```

```
98              //descrivo i due posizionamenti e ativazioni della camma
    in base alla direzione
99              IF (G_automaticJogPositive) THEN
100                 AutomaticJogDirection := POSITIVE;
101                 CammingDirection := POSITIVE;
102                 AutomaticjogPosition := 360.0;
103                 AutomaticjogStart := 0.0;
104             ELSE
105                 AutomaticJogDirection := NEGATIVE;
106                 CammingDirection := NEGATIVE;
107                 AutomaticjogPosition := 0.0;
108                 AutomaticjogStart := 360.0;
109             END_IF;
110             //abilitazione stessa camma per muovermi da 0 in su e 360
     in giù sul master
111             FOR i := 0 TO NR_AXIS-1 DO
112                 //se la camma non è più attiva, ma la differenza di
    posizione e target è diversa da zero, riattivo la stessa camma
113                 IF (G_EnableAxis[i] = TRUE AND SyncStruct[i].state =
    INACTIVE AND ABS(Virtual_Master.positioningState.actualPosition-
    AutomaticjogPosition)> POS_TOLERANCE) THEN
114                     ErrorEnableCam[i] := _enableCamming(
115                         followingObject := SyncStruct[i]
116                         ,direction := POSITIVE
117                         ,masterMode := ABSOLUTE
118                         ,slaveMode := ABSOLUTE
119                         ,cammingMode := NOCYCLIC
120                         ,cam := CamStruct[i]
121                         ,synchronizingMode := IMMEDIATELY
122                         ,syncPositionReference:=
    BE_SYNCHRONOUS_AT_POSITION
123                         ,syncProfileReference :=
    RELATE_SYNC_PROFILE_TO_TIME
124                         ,syncLengthType := DIRECT
125                         ,syncLength := 1.0
126                         ,camStartPositionMasterType := DIRECT
127                         ,camStartPositionMaster := AutomaticjogStart
    (*
128                         ,syncPositionMasterType :=
129                         ,syncPositionMaster :=
130                         ,syncPositionSlaveType :=
131                         ,syncPositionSlave :=
132                         ,velocityType :=
133                         ,velocity :=
134                         ,positiveAccelType :=
135                         ,positiveAccel :=
136                         ,negativeAccelType :=
137                         ,negativeAccel :=
138                         ,positiveAccelStartJerkType :=
```

```
139                         ,positiveAccelStartJerk :=
140                         ,positiveAccelEndJerkType :=
141                         ,positiveAccelEndJerk :=
142                         ,negativeAccelStartJerkType :=
143                         ,negativeAccelStartJerk :=
144                         ,negativeAccelEndJerkType :=
145                         ,negativeAccelEndJerk :=
146                         ,velocityProfile :=
147                         ,mergeMode :=
148                         ,nextCommand :=
149                         ,commandId :=
150                         ,synchronizingDirection :=*)
151                     );
152                 END_IF;
153             END_FOR;
154             //se il master è fermo e la distanza tra posizione e
    target è dversa da zero, attivo il posizionamento del master
155             IF (virtual_master.motionStateData.motionState =
    STANDSTILL AND ABS(Virtual_Master.positioningState.actualPosition-
    AutomaticjogPosition)> POS_TOLERANCE) THEN
156                 ErrorAutomaticJogPos := _pos(
157                     axis := Virtual_Master
158                     ,direction := AutomaticJogDirection
159                     ,positioningMode := ABSOLUTE
160                     ,position := AutomaticjogPosition
161                     ,velocityType := DIRECT
162                     ,velocity := 10
163                         (*,positiveAccelType :=
164                         ,positiveAccel :=
165                         ,negativeAccelType :=
166                         ,negativeAccel :=
167                         ,positiveAccelStartJerkType :=
168                         ,positiveAccelStartJerk :=
169                         ,positiveAccelEndJerkType :=
170                         ,positiveAccelEndJerk :=
171                         ,negativeAccelStartJerkType :=
172                         ,negativeAccelStartJerk :=
173                         ,negativeAccelEndJerkType :=
174                         ,negativeAccelEndJerk :=
175                         ,velocityProfile :=
176                         ,blendingMode :=
177                         ,mergeMode :=
178                         ,nextCommand :=
179                         ,commandId :=
180                         ,abortAcceleration :=*)
181                 );
182             END_IF;
183         ELSE
```

```
184             //se non ho alcun comando di jog, stoppo e aortisco il
    comando di posizionamento
185             IF (virtual_master.motionStateData.motionState <>
    STANDSTILL AND virtual_master.motionStateData.motionState <>
    DECELERATING) THEN
186                 ErrorAutomaticJogStop := _stop(
187                     axis := Virtual_Master
188                     ,stopMode := STOP_AND_ABORT
189                     ,stopSpecification := ALL_AXIS_MOTION
190                         //  ,stopId :=
191                     ,positiveAccelType := DIRECT
192                     ,positiveAccel := 100
193                     ,negativeAccelType := DIRECT
194                     ,negativeAccel := 100
195                     ,positiveAccelStartJerkType := DIRECT
196                     ,positiveAccelStartJerk :=100
197                     ,positiveAccelEndJerkType := DIRECT
198                     ,positiveAccelEndJerk := 100
199                     ,negativeAccelStartJerkType := DIRECT
200                     ,negativeAccelStartJerk := 100
201                     ,negativeAccelEndJerkType := DIRECT
202                     ,negativeAccelEndJerk := 100
203                     ,velocityProfile := SMOOTH
204                     ,mergeMode := IMMEDIATELY
205                     ,nextCommand := IMMEDIATELY
206                         //  ,commandId :=
207                     ,movingMode := CURRENT_MODE
208                     ,abortAcceleration := NO
209                 );
210             END_IF;
211         END_IF;
212     END_PROGRAM
213
214
215     PROGRAM automaticVelocityPosition
216         //calcolo one-shot della camma velocità-spazio
217         IF (xCamvelocityCalcDone = FALSE) THEN
218             CamProfileVelocity := Calc_Cam_Generic(Cam_Points :=
    CamPointVelocity);
219             ifb_velocityCam(
220                 ProfileCam := CamProfileVelocity
221                 ,Cam_Id := Cam_velocity_master
222                 ,CalcOk := xVelocityCamCorrectness
223             (* ,Result =>*)
224             );
225             xCamvelocityCalcDone :=TRUE;
226         END_IF;
227         //eseguo un lancio one-shot del movimento in camma di velocit
    à
```

```
228         IF (G_velocityStart AND G_velocityStop = FALSE AND
    Virtual_Master.velocityPositionProfileCommand.state = INACTIVE)
    THEN
229             ErrorVelocityProfile := _runPositionLockedVelocityProfile
    (
230                 axis := Virtual_Master
231                 ,profile := Cam_velocity_master(*
232                         ,positiveAccelType :=
233                         ,positiveAccel :=
234                         ,negativeAccelType :=
235                         ,negativeAccel :=
236                         ,positiveAccelStartJerkType :=
237                         ,positiveAccelStartJerk :=
238                         ,positiveAccelEndJerkType :=
239                         ,positiveAccelEndJerk :=
240                         ,negativeAccelStartJerkType :=
241                         ,negativeAccelStartJerk :=
242                         ,negativeAccelEndJerkType :=
243                         ,negativeAccelEndJerk :=
244                         ,velocityProfile :=
245                         ,mergeMode :=
246                         ,nextCommand :=
247                         ,commandId :=
248                         ,movingMode := *)
249             );
250         END_IF;
251
252         //se viene lanciato uno stop
253         IF (G_velocityStop AND Virtual_Master.
    velocityPositionProfileCommand.state <> INACTIVE) THEN
254             ErrorVelocityStop := _stop(
255                 axis := Virtual_Master
256                 ,stopMode := STOP_AND_ABORT
257                 ,stopSpecification :=ALL_AXIS_MOTION (*
258                 ,stopId :=
259                 ,positiveAccelType :=
260                 ,positiveAccel :=
261                 ,negativeAccelType :=
262                 ,negativeAccel :=
263                 ,positiveAccelStartJerkType :=
264                 ,positiveAccelStartJerk :=
265                 ,positiveAccelEndJerkType :=
266                 ,positiveAccelEndJerk :=
267                 ,negativeAccelStartJerkType :=
268                 ,negativeAccelStartJerk :=
269                 ,negativeAccelEndJerkType :=
270                 ,negativeAccelEndJerk :=
271                 ,velocityProfile :=
272                 ,mergeMode :=
```

```
273                 ,nextCommand :=
274                 ,commandId :=
275                 ,movingMode :=
276                 ,abortAcceleration := *)
277             );
278         END_IF;
279     END_PROGRAM
280
281     PROGRAM automaticPositionTime
282         //calcolo one-shot della camma spazio-tempo
283         IF (xPosTimeCamCorrectness = FALSE) THEN
284             CamProfilePosTime:= Calc_Cam_Generic(Cam_Points :=
    CamPointPosTime);
285             ifb_PosTimeCam(
286                 ProfileCam := CamProfilePosTime
287                 ,Cam_Id := X_Cam_PosTime
288                 ,CalcOk := xPosTimeCamCorrectness
289             (* ,Result =>*)
290             );
291             xCamPosTimeCalcDone :=TRUE;
292         END_IF;
293
294         //eseguo un lancio one-shot del movimento in camma di
    posizione
295         IF (G_PosTimeStart AND G_PosTimeStop = FALSE AND
    Virtual_Master.positionTimeProfileCommand.state = INACTIVE) THEN
296             ErrorVelocityProfile := _runTimeLockedPositionProfile(
297                 axis := Virtual_Master
298                 ,profile := X_Cam_PosTime
299                 ,startTime := 0.0
300                 ,profileDataMode := RELATIVE(*
301                 ,velocityType :=
302                 ,velocity :=
303                 ,positiveAccelType :=
304                 ,positiveAccel :=
305                 ,negativeAccelType :=
306                 ,negativeAccel :=
307                 ,positiveAccelStartJerkType :=
308                 ,positiveAccelStartJerk :=
309                 ,positiveAccelEndJerkType :=
310                 ,positiveAccelEndJerk :=
311                 ,negativeAccelStartJerkType :=
312                 ,negativeAccelStartJerk :=
313                 ,negativeAccelEndJerkType :=
314                 ,negativeAccelEndJerk :=
315                 ,velocityProfile :=
316                 ,mergeMode :=
317                 ,nextCommand :=
318                 ,commandId := *)
```

```
319             );
320         END_IF;
321         //se viene lanciato uno stop
322         IF (G_PosTimeStop AND Virtual_Master.
    positionTimeProfileCommand.state <> INACTIVE) THEN
323             ErrorVelocityStop := _stop(
324                 axis := Virtual_Master
325                 ,stopMode := STOP_AND_ABORT
326                 ,stopSpecification :=ALL_AXIS_MOTION (*
327                 ,stopId :=
328                 ,positiveAccelType :=
329                 ,positiveAccel :=
330                 ,negativeAccelType :=
331                 ,negativeAccel :=
332                 ,positiveAccelStartJerkType :=
333                 ,positiveAccelStartJerk :=
334                 ,positiveAccelEndJerkType :=
335                 ,positiveAccelEndJerk :=
336                 ,negativeAccelStartJerkType :=
337                 ,negativeAccelStartJerk :=
338                 ,negativeAccelEndJerkType :=
339                 ,negativeAccelEndJerk :=
340                 ,velocityProfile :=
341                 ,mergeMode :=
342                 ,nextCommand :=
343                 ,commandId :=
344                 ,movingMode :=
345                 ,abortAcceleration := *)
346             );
347         END_IF;
348
349     END_PROGRAM
350 END_IMPLEMENTATION
```

Bibliography

[1] Nora Wintour. *The glass industry: Recent trends and changes in working conditions and employment relations*. Aug. 2021. DOI: 10.1163/2210-7975\{_}hrd-4022-2015081. URL: https://doi.org/10.1163/2210-7975_hrd-4022-2015081 (cit. on p. 2).

[2] Mikell P. Groover. *Fundamentals of modern manufacturing : materials, processes, and systems*. Jan. 1996. URL: http://ci.nii.ac.jp/ncid/BA55024566 (cit. on p. 3).

[3] Reinhard Conradt. «Prospects and physical limits of processes and technologies in glass melting». In: *Journal of Asian Ceramic Societies* 7.4 (Sept. 2019), pp. 377–396. DOI: 10.1080/21870764.2019.1656360. URL: https://doi.org/10.1080/21870764.2019.1656360 (cit. on p. 3).

[4] «Review Lecture: The float glass process». In: *Proceedings of the Royal Society of London* 314.1516 (Dec. 1969), pp. 1–25. DOI: 10.1098/rspa.1969.0212. URL: https://doi.org/10.1098/rspa.1969.0212 (cit. on p. 3).

[5] Charles Angell. «Glass . Nature, Structure, and Properties. Horst Scholze. Springer-Verlag, New York, 1991. xiv, 454 pp.» In: *Science* (May 1992). DOI: 10.1126/science.256.5057.682.b. URL: https://doi.org/10.1126/science.256.5057.682.b (cit. on p. 5).

[6] W. H. Zachariasen. «THE ATOMIC ARRANGEMENT IN GLASS». In: *Journal of the American Chemical Society* 54.10 (Oct. 1932), pp. 3841–3851. DOI: 10.1021/ja01349a006. URL: https://doi.org/10.1021/ja01349a006 (cit. on p. 5).

[7] Granger K. Chui. «Heat Transfer and Temperature Control in an Annealing Lehr for Float Glass». In: *Journal of the American Ceramic Society* (Nov. 1977). DOI: 10.1111/j.1151-2916.1977.tb14086.x. URL: https://doi.org/10.1111/j.1151-2916.1977.tb14086.x (cit. on p. 6).

[8] *Float Glass Annealing Lehr | Stewart Engineers*. URL: https://stewartengineers.com/en/innovations/float-glass-annealing-lehr/ (cit. on p. 7).

[9] Francesco Adamo, Filippo Attivissimo, Attilio Di Nisio, and Mario Savino. «A low-cost inspection system for online defects assessment in satin glass». In: *Measurement* 42.9 (Nov. 2009), pp. 1304–1311. DOI: 10.1016/j.measurement.2009.05.006. URL: https://doi.org/10.1016/j.measurement.2009.05.006 (cit. on p. 8).

[10] V.A. Litvinov, I. A. Maistrenko, E.A. Tarasov, and F. B. Grinberg. «Cutting glass with a hard-alloy roller». In: *Glass and Ceramics* 29.12 (Dec. 1972), pp. 793–795. DOI: 10.1007/bf00674317. URL: https://doi.org/10.1007/bf00674317 (cit. on p. 9).

[11] M.I. Smirnov, Yu. A. Spiridonov, and A.R. Karapetyan. «Modern sheet-glass cutting technologies». In: *Glass and Ceramics* (May 2011). DOI: 10.1007/s10717-011-9310-3. URL: https://doi.org/10.1007/s10717-011-9310-3 (cit. on p. 9).

[12] P. V. Popov. «CALCULATION OF CONTACT PRESSURE IN CUTTING SHEET GLASS WITH A HARD-ALLOY METAL ROLLER». In: *Glass and Ceramics* (July 2001). DOI: UDC.666.1.053.2.001.24 (cit. on p. 10).

[13] Héctor Cañas, Josefa Mula, Manuel Díaz-Madroñero, and Alexandre Dolgui. «Implementing Industry 4.0 principles». In: *Computers Industrial Engineering* 158 (Aug. 2021), p. 107379. DOI: 10.1016/j.cie.2021.107379. URL: https://doi.org/10.1016/j.cie.2021.107379 (cit. on p. 14).

[14] Matteo Rossini, Federica Costa, Guilherme Luz Tortorella, and Alberto Portioli Staudacher. «The interrelation between Industry 4.0 and lean production: an empirical study on European manufacturers». In: *The International Journal of Advanced Manufacturing Technology* 102.9-12 (Mar. 2019), pp. 3963–3976. DOI: 10.1007/s00170-019-03441-7. URL: https://doi.org/10.1007/s00170-019-03441-7 (cit. on pp. 14, 16).

[15] Morteza Ghobakhloo. «Industry 4.0, digitization, and opportunities for sustainability». In: *Journal of Cleaner Production* 252 (Apr. 2020), p. 119869. DOI: 10.1016/j.jclepro.2019.119869. URL: https://doi.org/10.1016/j.jclepro.2019.119869 (cit. on pp. 15, 18).

[16] Brijesh Sivathanu and Rajasshrie Pillai. «Smart HR 4.0 – how industry 4.0 is disrupting HR». In: *Human Resource Management International Digest* 26.4 (May 2018), pp. 7–11. DOI: 10.1108/hrmid-04-2018-0059. URL: https://doi.org/10.1108/hrmid-04-2018-0059 (cit. on p. 16).

[17] Keliang Zhou, Taigang Liu, and Lifeng Zhou. «Industry 4.0: Towards future industrial opportunities and challenges». In: (2015), pp. 2147–2152 (cit. on p. 16).

[18] Lorenzo Ardito, Antonio Messeni Petruzzelli, Umberto Panniello, and Achille Claudio Garavelli. «Towards Industry 4.0». In: *Business Process Management Journal* 25.2 (July 2018), pp. 323–346. DOI: 10.1108/bpmj-04-2017-0088. URL: https://doi.org/10.1108/bpmj-04-2017-0088 (cit. on p. 18).

[19] Qinglin Qi and Fei Tao. «Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison». In: *IEEE Access* 6 (Jan. 2018), pp. 3585–3593. DOI: 10.1109/access.2018.2793265. URL: https://doi.org/10.1109/access.2018.2793265 (cit. on p. 18).

[20] Jorge Posada et al. «Visual Computing as a Key Enabling Technology for Industrie 4.0 and Industrial Internet». In: *IEEE Computer Graphics and Applications* 35.2 (Mar. 2015), pp. 26–40. DOI: 10.1109/mcg.2015.45. URL: https://doi.org/10.1109/mcg.2015.45 (cit. on p. 18).

[21] Martin Sjarov, Tobias Lechler, Jonathan D. Fuchs, Matthias Brossog, Andreas Selmaier, Florian Faltus, Toni Donhauser, and Jörg Franke. «The Digital Twin Concept in Industry – A Review and Systematization». In: Sept. 2020. DOI: 10.1109/etfa46521.2020.9212089. URL: https://doi.org/10.1109/etfa46521.2020.9212089 (cit. on p. 18).

[22] Michael W. Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*. Aug. 2016, pp. 85–113. DOI: 10.1007/978-3-319-38756-7\{_}4. URL: https://doi.org/10.1007/978-3-319-38756-7_4 (cit. on p. 18).

[23] Soumya Singh, Max Weeber, and Kai Peter Birke. «Advancing digital twin implementation: a toolbox for modelling and simulation». In: *Procedia CIRP* 99 (Jan. 2021), pp. 567–572. DOI: 10.1016/j.procir.2021.03.078. URL: https://doi.org/10.1016/j.procir.2021.03.078 (cit. on p. 19).

[24] Pingyu Jiang, Dewen Wang, Weiming Shen, Quan-Ke Pan, Qiang Liu, and Xin Chen. «Digital twins-based smart manufacturing system design in Industry 4.0: A review». In: *Journal of Manufacturing Systems* 60 (July 2021), pp. 119–137. DOI: 10.1016/j.jmsy.2021.05.011. URL: https://doi.org/10.1016/j.jmsy.2021.05.011 (cit. on p. 20).

[25] Tobias Lechler, Eva K. Fischer, Maximilian Metzner, Andreas Mayr, and Jörg Franke. «Virtual Commissioning – Scientific review and exploratory use cases in advanced production systems». In: *Procedia CIRP* 81 (Jan. 2019), pp. 1125–1130. DOI: 10.1016/j.procir.2019.03.278. URL: https://doi.org/10.1016/j.procir.2019.03.278 (cit. on p. 22).

[26] Sophie Prat, Jeremy Cavron, Djamal Kesraoui, Philippe Rauffet, Pascal Berruet, and Alain Bignon. «An Automated Generation Approach of Simulation Models for Checking Control/Monitoring System». In: *IFAC-PapersOnLine* 50.1 (July 2017), pp. 6202–6207. DOI: 10.1016/j.ifacol.2017.08.1014. URL: https://doi.org/10.1016/j.ifacol.2017.08.1014 (cit. on p. 22).

[27] Zheng Liu, Nico Suchold, and Christian Diedrich. *Virtual Commissioning of Automated Systems*. July 2012. DOI: 10.5772/45730. URL: https://doi.org/10.5772/45730 (cit. on p. 23).

[28] Mathias Oppelt and Leon Urbas. «Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering». In: Oct. 2014. DOI: 10.1109/iecon.2014.7048867. URL: https://doi.org/10.1109/iecon.2014.7048867 (cit. on p. 25).

[29] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics*. Springer Science Business Media, Aug. 2010 (cit. on pp. 27, 36, 37, 46).

[30] L. Keith Barker. *Modified Denavit-Hartenberg Parameters for Better Location of Joint Axis Systems in Robot Arms*. Jan. 1986 (cit. on p. 29).

[31] *Robotics System Toolbox Documentation - MathWorks Italia*. URL: https://it.mathworks.com/help/robotics/index.html?s_tid=CRUX_lftnav (cit. on pp. 32, 33).

[32] Luigi Biagiotti and Claudio Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. Springer Science Business Media, Oct. 2008 (cit. on pp. 48, 49, 62).

[33] *Simscape Documentation - MathWorks Italia*. URL: https://it.mathworks.com/help/simscape/ (cit. on pp. 67, 68).

[34] *Totally Integrated Automation – Product Guide*. 2011 (cit. on p. 75).

[35] P. Papcun, Erik Kajati, and Jiri Koziorek. «Human Machine Interface in Concept of Industry 4.0». In: Aug. 2018. DOI: 10.1109/disa.2018.8490603. URL: https://doi.org/10.1109/disa.2018.8490603 (cit. on p. 76).

[36] Enrique García Viñuela, Dániel Darvas, and Gyula Sallai. «Testing Solutions for Siemens PLCs Programs Based on PLCSIM Advanced». In: *Conf. on Acc. and Large Exp. Physics Control Systems* (Aug. 2020), pp. 1107–1110. DOI: 10.18429/jacow-icalepcs2019-wepha018. URL: https://jacow.org/icalepcs2019/papers/wepha018.pdf (cit. on p. 79).

[37] Daynier Rolando Delgado Sobrino, Roman Ružarovský, Radovan Holubek, and Karol Velíšek. «Into the early steps of Virtual Commissioning in Tecnomatix Plant Simulation using S7-PLCSIM Advanced and STEP 7 TIA Portal». In: *MATEC web of conferences* 299 (Jan. 2019), p. 02005. DOI: 10.1051/matecconf/201929902005. URL: https://doi.org/10.1051/matecconf/201929902005 (cit. on p. 79).

[38] Michael Braun and Wolfgang Horn. *Object-Oriented Programming with SIMOTION*. John Wiley Sons, June 2017 (cit. on p. 80).

[39] Mathias Oppelt, Oliver Drumm, Benjamin Lutz, and A G Gerrit Wolf Siemens. «Approach for integrated simulation based on plant engineering data». In: *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*. 2013, pp. 1–4. DOI: 10.1109/ETFA.2013.6648156 (cit. on p. 89).

Acknowledgements

First, I would like to express my deep gratitude to Professor Andrea Tonoli, my thesis advisor, for his patience and invaluable support during the development of this project.

I would also like to thank all the people in the technical department of flat glass business unit of Bottero S.p.A. for all the insights and suggestions. In particular, I am so grateful to Alessio Callegari and Alberto Cigliutti for the chance given to prove myself with this challenge, to Luca Dardanello, for the technical guidance for this project and to Pierfranco Bergese for his experience and support.

Thank you to all my colleagues, with whom I shared these fantastic years, creating a wonderful environment even in a pandemic scenario. To my old and new friends that helped me in all the stages of this path with laughs and magical memories that I will preserve for the rest of my life.

A special thanks goes to my granddad, that didn't had the chance to see me graduate, but always believed in me and taught me principles and morals of inestimable value. Then, I must express my profound gratitude to family, that always supported and encouraged me to pursue my goals.

This accomplishment would not have been possible without you. Finally, thank you Denise, for the love that you give me every day and for giving me the spark of hope needed to accomplish this journey that leads to our future together.

Paolo Arnaudo