applied system innovation

MDPI

Article

Modular Robotics Configurator: A MATLAB Model-Based Development Approach

Ernest Andrei Grosz and Marian Borzan *

Department of Manufacturing Engineering, Faculty of Machine Building, Technical University of Cluj-Napoca, B-dul Muncii, No. 103-105, 400641 Cluj-Napoca, Romania; ernest.grosz@campus.utcluj.ro
* Correspondence: marian.borzan@tcm.utcluj.ro

Abstract: In recent years, modularity has become increasingly present in the field of robotics and mechatronics. The need to easily calculate and generate modular robots in an easier and faster way and for a wide range of robotics projects has increased. This paper aims to present a new modular software development method for an automatic configurator of serial robots through the model-based development technique using the MATLAB Simulink R2024b software. Using this functionality, the user has the possibility to configure, generate, and analyze serial robots, starting from a kinematic scheme that complies with the development requirements of the project. Having a high degree of flexibility and avoiding multiple instances of debugging, caused by the need to use different environments, as well as connectivity problems between them or a lack of support in development, the user has the possibility to configure and reconfigure serial robots, easily adapting to the high degree of dynamism in the development of current projects. Because of the modularity on which the software development is based, the user has the possibility to optimize and improve the quality and performance of the system, along with easy adaptation to the speed of technological advancement.

Keywords: modularity; serial robots; robotics software; model-based development; interchangeable configuration

Academic Editor: Igor Korobiichuk

Received: 4 December 2024
Revised: 20 January 2025
Accepted: 27 January 2025
Published: 3 February 2025

Citation: Grosz, E.A.; Borzan, M. Modular Robotics Configurator: A MATLAB Model-Based Development Approach. *Appl. Syst. Innov.* **2025**, *8*, 21. https://doi.org/10.3390/asi8010021

1. Introduction

Robotics is a field that focuses on the development and study of robots, encompassing the modeling, design, and manufacturing of robotic systems. Industrial robots are extensively employed in the automotive and electronics industries, as well as in other fields [1]. In today's manufacturing industry, vision-guided robotic picking systems play a vital role in automating various tasks. However, achieving rapid changeover and adaptability for diverse workpiece types has remained a significant challenge [2]. Modular robotic systems are systems that are composed of modules that can be disconnected and reconnected in different arrangements to form a new system, enabling new functionalities [3]. The term "modular robotics" refers to a family of robotic systems composed of interconnected smaller units called "modules", joined together by docking interfaces [4]. Modularity offers significant functional and economic advantages over more traditional fixed-structure robots. The ability to reconfigure the morphology by rearranging their parts enables modular robots to adapt to changes in the environment [4]. The robot's links are connected in serial by rotary or translatory motion joints, which form a kinematic chain. The kinematic chain end is called the end-effector or gripper, which is modeled on the human hand [5]. Modularity is a means of decomposing a large and complex system or product into small, simple,

independent, and manageable modular units that can be easily composed, decomposed, and replaced, and it can be frequently found in both nature and industrial systems [6].

Software process simulation modeling is increasingly being used to address a variety of issues, from the strategic management of software development to supporting process improvements [7]. Not only do modeling and simulation help to provide a better understanding of how real-world systems function, but they also enable us to predict system behavior before a system is built and analyze systems accurately under varying operating conditions [8]. MATLAB is a powerful environment for linear algebra and graphical presentation that is available on a wide range of computer platforms [9]. Nowadays, new product development makes strong use of IT tools, such as virtual simulation tools. The main motivation for the introduction of virtual simulation tools in new product development is to speed up development and reduce costs. Virtual simulation tools can achieve much more than simply speeding up product development and reducing the cost [10–13]. At the same time, however, they also trigger profound changes in problem solving [10]. One of the most important motivations for the introduction of virtual simulation tools is that the cost and time requirements of physical experiments are not always compatible with the strict time and cost constraints of the new product development process [11]. The ability to predict the maximal performance of an industrial robot in executing non-deterministic tasks can improve process productivity through time-based planning and scheduling strategies. These strategies require the configuration and comparison of a large number of tasks in real time for decision making; therefore, an efficient task execution time estimation method is required [12].

The automotive industry was the first to promote the development of large-scale applications of model-based systems technology and, as a result, produced some of the most advanced prototypes and products [13]. Model-based systems provide a good basis for problem solving because they are based on the separation of the problem solving algorithm from the model and the compositionality of this model, which addresses the variant problem. Once a library of suitable component models is established, only a structural description of this device (e.g., obtained from design data) is required to automatically generate a system model and, based on it, a dedicated problem solving system for this one device [13].

Organizations today are facing rising labor costs and a shortage of workers and are therefore investing in robotics. Robots can perform tasks that most humans cannot, such as working in difficult conditions and being able to generate data with the utmost precision [14]. In the past, robots were predominantly used in manufacturing to perform labor-intensive tasks that were hazardous for humans. As robots' capabilities rapidly increased, they were used for routine tasks that required speed, agility, and dexterity, such as spot welding, stacking, drilling, cutting, and so on. Today, autonomous cars increasingly incorporate sensors and machine learning, which increase the types of tasks that can be undertaken. Therefore, robots can be used for a growing number of activities within firms, including transporting goods, assessing quality, testing products, and so on. One of the main reasons that firms adopt robots is their performance in terms of productivity and therefore production. Usually, robots can work continuously and perform functions 24 h a day [15].

In recent years, an increasing emphasis on the modularity of robots has emerged, generating them with the help of command-and-control software. There is a wide range of software that allows the generation and analysis of robots and the possibility of manipulating their individual components.

Computer modeling, simulation, and implementation tools have been widely used to support and develop nonlinear control, robotics, and MATLAB/Simulink courses. MAT-

LAB, with its toolboxes, such as Simulink, is one of the most widely accepted software packages and is used by researchers to enhance teaching regarding the transient and steady-state characteristics of control and robotics [16].

In Industry 4.0, robots are regarded as one of the key components. In recent years, collaborative robots (cobots) have risen in relevance and have been included in the industry to perform tasks alongside humans [17].

The digital era necessitates robotic automation, which utilizes intelligent automation technologies to streamline processes and execute tasks according to predetermined programs, thus automating the process. The adoption of robotic process automation fosters significant increases in productivity and efficiency across various sectors [18].

From the point of view of robot producers, such unprecedented diffusion has necessitated the development of innovative safety features, aimed at making the use of cobots increasingly secure and accessible to the wider public [19].

In the era of rapid development and prototyping, limitations like multiple software programs used in the calculation and generation of robots, multiple debugging activities, and different software with different language programs make development activities difficult. The need to know the connections between such tools can lead to higher effort and the higher consumption of resources and time to solve errors, as compared to the necessary investments for development and creation. The increased amount of knowledge required, together with the large number of tools, makes development activities more challenging for the user. Starting with these, this paper aims to develop a user-oriented application that will help a user to configure serial robots in an environment, being guided step by step throughout its use.

Considering the computing power and the vast engineering functionalities of the MATLAB Simulink software, MATLAB is the software that the authors propose to use as the source program in the development of this functionality.

In this paper, the main goal is to develop a new modular and compact Simulink function, using the model-based development technique, where the user can configure serial robots starting from the number of kinematic couples, the specific type of movement, the geometric parameters, and the generalized coordinates of each component. After calculating the kinematic couplings and determining the configuration of the serial robot, the end-effector will be attached. This block will calculate the displacements, velocity, and accelerations for the end-effector and for each kinematic coupling separately in the three-dimensional space. These aspects will be represented in specific graphs.

The application will be developed on several layers, each of them having the possibility to be individually managed, giving the application modularity from the beginning.

2. Materials and Methods

The motivation for the development of this application comes from the need to realize a project in a compact and well-structured environment. Moreover, it comes from the need to be able to realize a prototype in an easier way, without requiring a large range of software programs or additional ones to be able to configure and design a complete project.

At the same time, there is the need for an application that is easy to use, in which it is not necessary to invest additional time in its operation or in its connection with other software. The focus has to be oriented towards the realization of the calculations needed for the specific robot, where there are no administrative blocking points caused by the used tools. Finally, there exists the need to bring all of the necessary information into one place, with the possibility of realizing a large project with a focused set of knowledge.

The main goal of this project is to create a function that gives the user the possibility to generate any configuration for a serial robot, with the opportunity to include as

many degrees of mobility as needed, having only fifth-class kinematic couples (rotation or translation). Such a function would ultimately enable one to configure, calculate, analyze, and reconfigure as many combinations of robots as possible in the shortest possible time, making it more efficient not only from the point of view of time but also from the point of view of resources.

With advancements in technology, time becomes increasingly compressed. For this reason, users require all components to be accessible and intuitive. Starting with this, this project aims to create a MATLAB Simulink function that requires minimal investment of time in understanding its operation. At the same time, we seek to offer the highest level of capability and performance with minimal knowledge about its implementation. We adopt a user-oriented usability perspective, providing one of the easiest ways to approach building, calculation, and robot configuration. At the same time, this application aims to create a graphical representation of the displacement, velocity, and acceleration of the robot's kinematic couples and also for its end-effector, in accordance with the design and implementation requirements of a wide range of projects.

2.1. General Information

In the paper "Design and Realisation of the Simulation Model of the Stewart Platform using the MATLAB-Simulink and the Simscape Multibody Library" [20], written by P. Noskievič and D. Walica, MATLAB's Simscape Multibody Link is used to create and manipulate a Stewart robot. CAD software is used to generate a 3D model of the Stewart robot. Before being able to implement the robot in Simscape Multibody Link, the generated files from CAD have to be adapted to MATLAB using an add-on, which will transform the information into a format that can be interpreted by the Simscape library. This process prolongs the time required in building the robot, requiring a large amount of time for the projection of the model. After creating the CAD model and transferring the information to the Simscape library, it is necessary to create a Simulink model to be able to control and manipulate the robot. This model needs to be created with the help of the library, which requires particular knowledge. After completing the CAD and creating the Simulink model based on the Simscape Multibody Link library, it is possible to visualize the robot using the Mechanics Explorer interface.

Following this development path, some weak spots have been identified in the use of this workflow. The first is the need for the prototyping of a CAD model and the need of knowledge of the library for Simscape Multibody Link to be able to create the Simulink models from scratch. At the same time, the possibility of connectivity and communication errors between the various programs used is observed. The use of multiple software programs simultaneously to reach the final point of generating the desired robot is not a user-oriented approach. The use of multiple software packages brings the need to have knowledge in their use and to invest time in possible debugging activities, as well as the need for multiple licenses, in addition to the actual time required for the development, generation, calculation, and analysis of the robots. All of these factors lead to the sub-optimal use of resources and time.

In the paper "Implementation of a Two Wheel Self-Balanced Robot using MATLAB Simscape Multibody" [21], written by S. Mohapatra, R. Srivastava, and R. Khera, the same development path is presented using the MATLAB Simscape Multibody Link library. The first necessary step is to design a CAD model. After this, the CAD is converted using an additional tool into a data type that is read by the MATLAB Simscape Multibody Link library; then, the MATLAB Simulink model will be developed.

The need for knowledge has been identified in the design and prototyping of a robot model in order to create a CAD model. This process is time-consuming. Moreover, there

is a need for several tools in creating the CAD model. Furthermore, add-ons to convert CAD files into files that are readable by the MATLAB software are necessary. As a final step, knowledge of the use of the MATLAB Simulink software from scratch is required, as well as an understanding of the use of the MATLAB Simscape Multibody Link library.

For comparison purposes, Table 1 presents a description of a regular robot's configuration and simulation using the standard method and the proposed method.

Table 1. Comparison of the standard method and proposed method.

Comparison Area	Standard Method	Proposed Method
Development of a 3D model of the robot	A CAD model must be created.	A robot schematic must be created.
Connectivity and programs used	The end-user must use multiple software packages. The end-user must have knowledge about the connections of the used tools to be able to convert the information from the 3D modeling tool to MATLAB. The end-user must import the information into MATLAB and verify its correctness and also conduct troubleshooting if needed.	The end-user uses only the proposed MATLAB function.
Modeling and use of the method	After the import is finished, the end-user must compile the model and create a second Simulink model for simulation. The end-user must configure the parameters of each block, establish a connection, and verify the correctness of the model.	The end-user enters the necessary information regarding the robot design into the MATLAB Simulink functionality block. The end-user will be guided by the function with the help of assistive questions.
Results and interpretation	After running the program, the user will have access to the Mechanics Explorer platform to view the robot.	After running the function block, the calculated matrices are presented together with the graphs for the displacement, velocity, and acceleration.

In order to use the application, it is necessary to carry out an analysis of the desired movement, along with the schematization of the robot's movements, attaching a coordinate system $[T_0]...[T_n]$, $[n \in \mathbb{N} \mid n \geq 0]$ to each element. To realize this, the robotic schematic reference shown in Figure 1 is proposed.

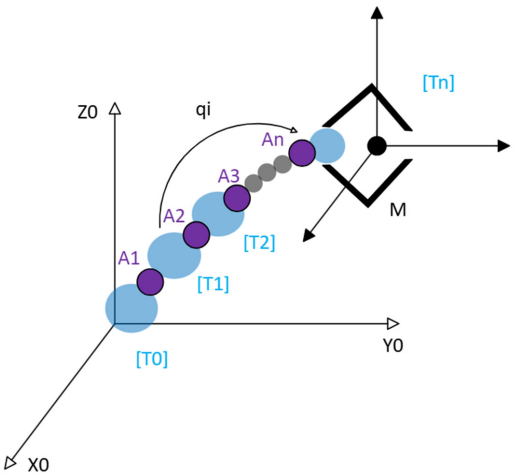


Figure 1. Attachment of coordinate systems and robot elements.

The relative motion between two consecutive elements can be described by a 4×4 homogeneous matrix. This matrix is called a transfer/transformation matrix and has the form

$$A_{i,i+1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where a_{14}, a_{24}, a_{34} represent the coordinates of the origin with the higher index in rank with the coordinate system with the lower index.

The remaining elements of the matrix $A_{i,i+1}$ are the cosines of the angles between the axes of the system, $[T_i]$ and $[T_{i,i+1}]$, and are given by the relations

$$a_{11} = \cos(\mathrm{Ox_i}, \mathrm{Ox_{i+1}}); \mathrm{a_{12}} = \cos\left(\mathrm{Ox_i}, \mathrm{Oy_{i+1}}\right); \mathrm{a_{13}} = \cos(\mathrm{Ox_i}, \mathrm{Oz_{i+1}}) \quad (2)$$

$$a_{21} = \cos(\mathrm{Oy_i}, \mathrm{Ox_{i+1}}); \mathrm{a_{22}} = \cos\left(\mathrm{Oy_i}, \mathrm{Oy_{i+1}}\right); \mathrm{a_{23}} = \cos(\mathrm{Oy_i}, \mathrm{Oz_{i+1}}) \quad (3)$$

$$a_{31} = \cos(\mathrm{Oz_i}, \mathrm{Ox_{i+1}}); \mathrm{a_{32}} = \cos\left(\mathrm{Oz_i}, \mathrm{Oy_{i+1}}\right); \mathrm{a_{33}} = \cos(\mathrm{Oz_i}, \mathrm{Oz_{i+1}}) \quad (4)$$

$$A_i = \begin{bmatrix} \mathrm{R_i} & \mathrm{P_i} \\ 0 & 1 \end{bmatrix} \quad (5)$$

$$R_i = [3 \times 3] - \text{Orientation sub-matrix}$$

$$P_i = [3 \times 1] - \text{Position sub-matrix}$$

- The geometric parameters are given for the first kinematic couple that realizes the movement between the systems $[T_0]$ and $[T_n]$.
- For the following kinematic couplings, the geometric parameters are required in relation to the kinematic coupling attached to the preceding element.
- The geometric parameters of the end-effector will be determined in relation to the coordinate system $[T_n]$ attached to element n.

The matrix P_n is generated, which is the matrix of the coordinates of point M in relation to the system $[T_n]$, along with the end-effector (EF), which represents the characteristic point of the robot.

$$P_n = \begin{bmatrix} \mathrm{P_x} \\ \mathrm{P_y} \\ \mathrm{P_z} \\ 1 \end{bmatrix} \quad (6)$$

To establish the parking position, the generalized coox1 wzqsa rdinates q_i for each kinematic couple will have a value of 0; later, it is possible to change their values through the user interface.

For the simulation of the robot, homogeneous transfer matrices will be generated, which are calculated with the following formula:

$$A_{0,n} = \prod_{i=0}^{n-1} A_{i,i+1} \quad (7)$$

The position of the characteristic point attached to the final effector in relation to the system $[T_0]$, expressed with the help of the transfer matrices, will be calculated using the relation

$$P_{0,n} = \prod_{i=0}^{n-1} A_{i,i+1} \cdot P_n \quad (8)$$

The Denavit–Hartenberg homogeneous operator method assumes that the elements of homogeneous transformation matrices of size 4×4 can be expressed using four independent parameters, called Denavit–Hartenberg parameters. These are defined as follows:

- angle θ_i—the angle between the axes x_{i-1} and x_i;
- distance a_i—the distance between the origins O_{i-1} and O_i measured on the x_i axis;
- distance d_i—the distance between the origins O_{i-1} and O_i measured on the z_{i-1} axis;
- angle α_i—the angle between the z_{i-1} and z_i axes.

The transformation of the coordinates between the elements i and $i+1$ results in a sequence of four elementary movements, namely

- a rotation with the angle θ_i performed around the z_i axis;
- a translation with the distance d_i along the z_i axis;
- a translation with the distance a_i along the x_{i+1} axis;
- a rotation with the angle α_i around the x_{i+1} axis.

The homogeneous transformation matrix of size 4×4 characterizes the relative movement between the elements i and $i+1$. It is obtained after obtaining the product of the homogeneous matrices that characterize the four movements:

$$A_{i,i+1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$A_{i,i+1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cdot\cos(\alpha_i) & \sin(\theta_i)\cdot\sin(\alpha_i) & a_i\cdot\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cdot\cos(\alpha_i) & -\cos(\theta_i)\cdot\sin(\alpha_i) & a_i\cdot\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

2.2. Application Overview

In the following paragraphs, the process of creating the MATLAB Simulink function for the structure of a serial robot will be explained.

Starting with the mathematical models, MATLAB Simulink requires the following system and sub-system blocks to be generated:

1. Rotation around the X-axis sub-system model;
2. Rotation around the Y-axis sub-system model;
3. Rotation around the Z-axis sub-system model;
4. Translation along the X-axis sub-system model;
5. Translation along the Y-axis sub-system model;
6. Translation along the Z-axis sub-system model;
7. Null sub-system model (matrix unit);
8. Movement module function sub-system model;
9. Robotic configuration sub-system model;
10. End-effector sub-system model;
11. The sub-system of the input parameters' manipulation;
12. Displacement, velocity, and acceleration sub-system model;
13. Robot generation functionality model.

The robotic generation functionality structure is shown in Figure 2.

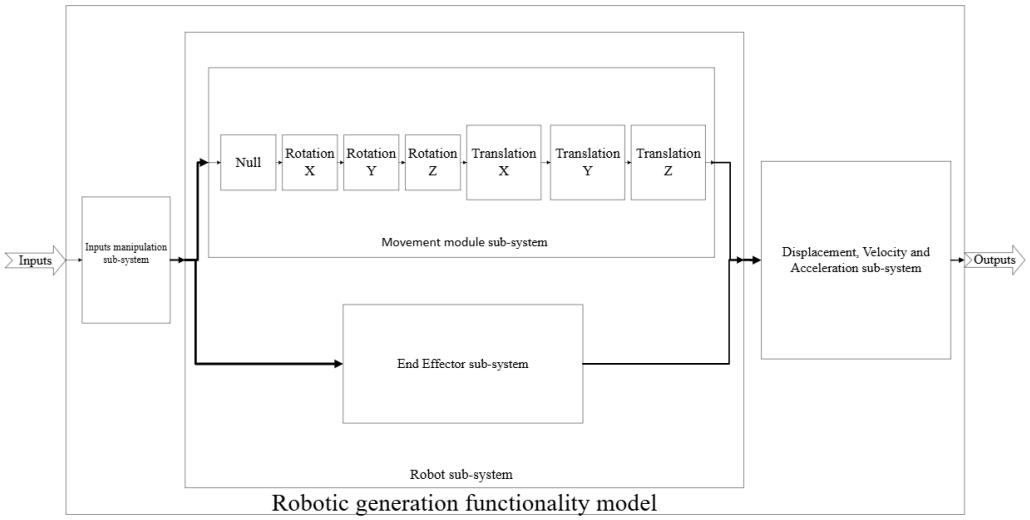


Figure 2. Robotic generation functionality structure.

The following steps represent the working method of the functionality.

Step 1: The end-user will introduce the inputs through the input port (geometric parameters, generalized coordinates, number of modules, type of movement of the modules).
Step 2: The inputs will be processed in the input manipulation sub-system.
Step 3: Based on the inputs, the type of movement will be configured and calculated in the movement module sub-system through the null, rotation X, rotation Y, rotation Z, translation X, translation Y, and translation Z sub-systems.
Step 4: Based on the inputs, the robot modules will be configured and calculated together with the end-effector inside the robot sub-system.
Step 5: The displacement, velocity, and acceleration will be calculated and represented as graphs in the displacement, velocity, and acceleration sub-system.
Step 6: The calculated information and graphs will be sent outside, in the robotic generation functionality model, through the output port.

2.3. Development of the MATLAB Simulink Serial Robot Functionality

Simulink is a block diagram environment used to design systems and sub-systems with several domain models, in which the user can create a simulation before moving forward to hardware and deployment without writing code.

In the following paragraphs, the development of the MATLAB Simulink functionality block will be presented.

As a first step, the seven Simulink models (three translations, three rotations, and the null block) used in the motion type model shown in Figure 2 will be created. The next step is to create the motion type block and incorporate it, together with the final effector block, into the robot block. This will be followed by the development of the displacement, velocity, and acceleration calculation block. Finally, all of these will be incorporated into the modular robot generation and calculation function.

2.3.1. Development of the Movement Sub-System Models

In the following, the rotational and translational motion models will be created, along with the unitary matrix model, to further create the entire motion block.

For the translational couple along the x axis, the homogeneous transfer matrix will be used and converted to Simulink:

$$A_{i,i+1}^{T,x} = \begin{bmatrix} 1 & 0 & 0 & q_i + a_x \\ 0 & 1 & 0 & a_y \\ 0 & 0 & 1 & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

The Simulink model in which $A_{i,i+1}^{T,x}$ is adapted to the block diagram is presented in Figure 3.

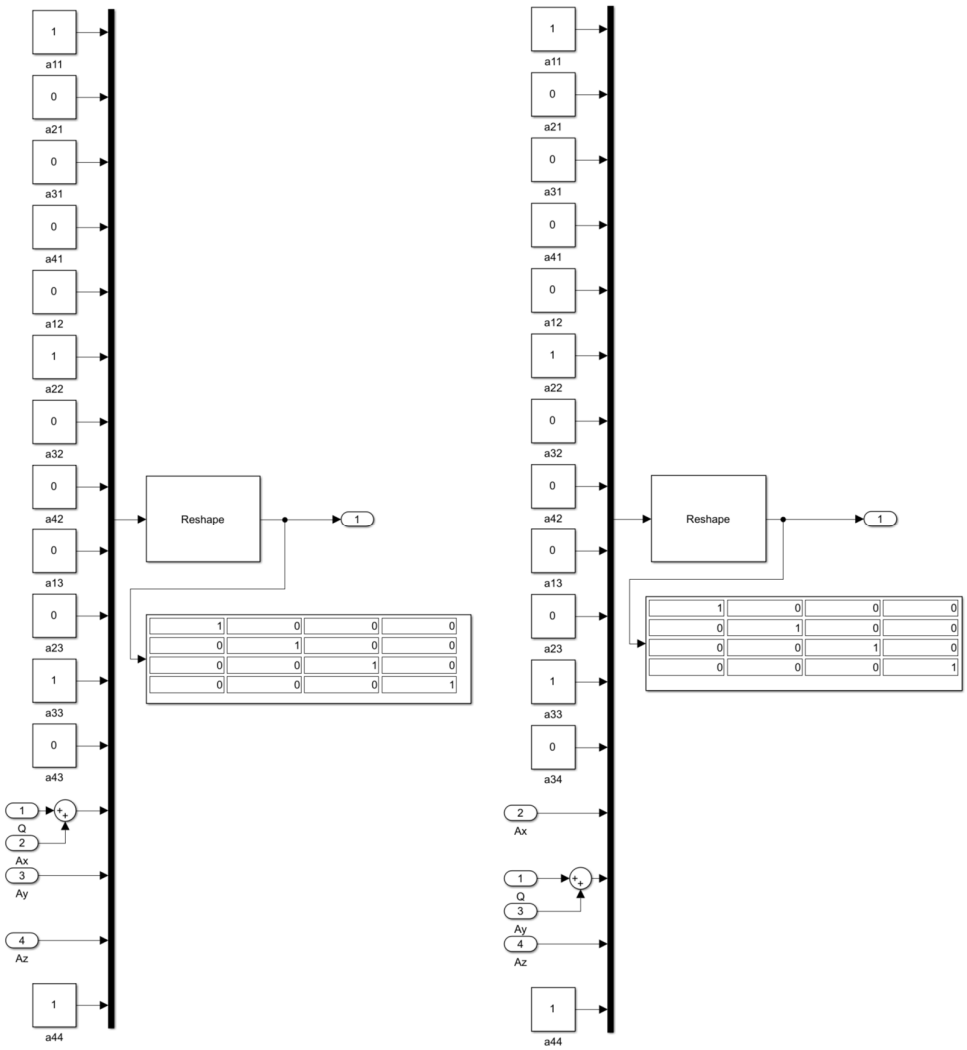


Figure 3. Translation along the (**a**) X- and (**b**) Y-axis sub-system models.

For the translational couple along the y axis, the homogeneous transfer matrix will be used and converted to Simulink:

$$A_{i,i+1}^{T,y} = \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & 1 & 0 & q_i + a_y \\ 0 & 0 & 1 & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

The Simulink model in which $A_{i,i+1}^{T,y}$ is adapted to the block diagram is presented in Figure 3.

For the translational couple along the z axis, the homogeneous transfer matrix will be used and converted to Simulink:

$$A_{i,i+1}^{T,z} = \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & 1 & 0 & a_y \\ 0 & 0 & 1 & q_i + a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

The Simulink model in which $A_{i,i+1}^{T,z}$ is adapted to the block diagram is presented in Figure 4.

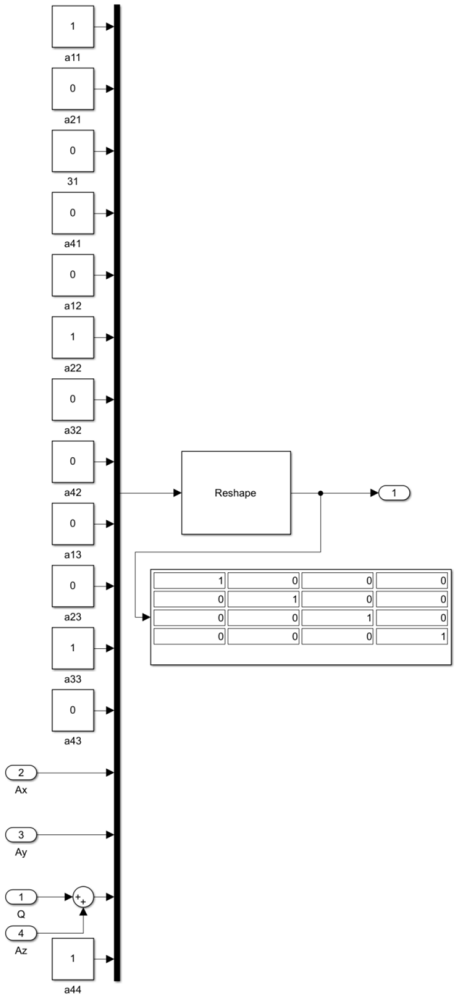


Figure 4. Translation along the Z-axis sub-system model.

For the rotation couple around the x axis, the homogeneous transfer matrix will be used and converted to Simulink:

$$A_{i,i+1}^{R,x} = \begin{bmatrix} 1 & 0 & 0 & a_x \\ 0 & \cos(q_i) & -\sin(q_i) & a_y \\ 0 & sin\ (q_i) & \cos(q_i) & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

The Simulink model in which $A_{i,i+1}^{R,x}$ is adapted to the block diagram is presented in Figure 5.

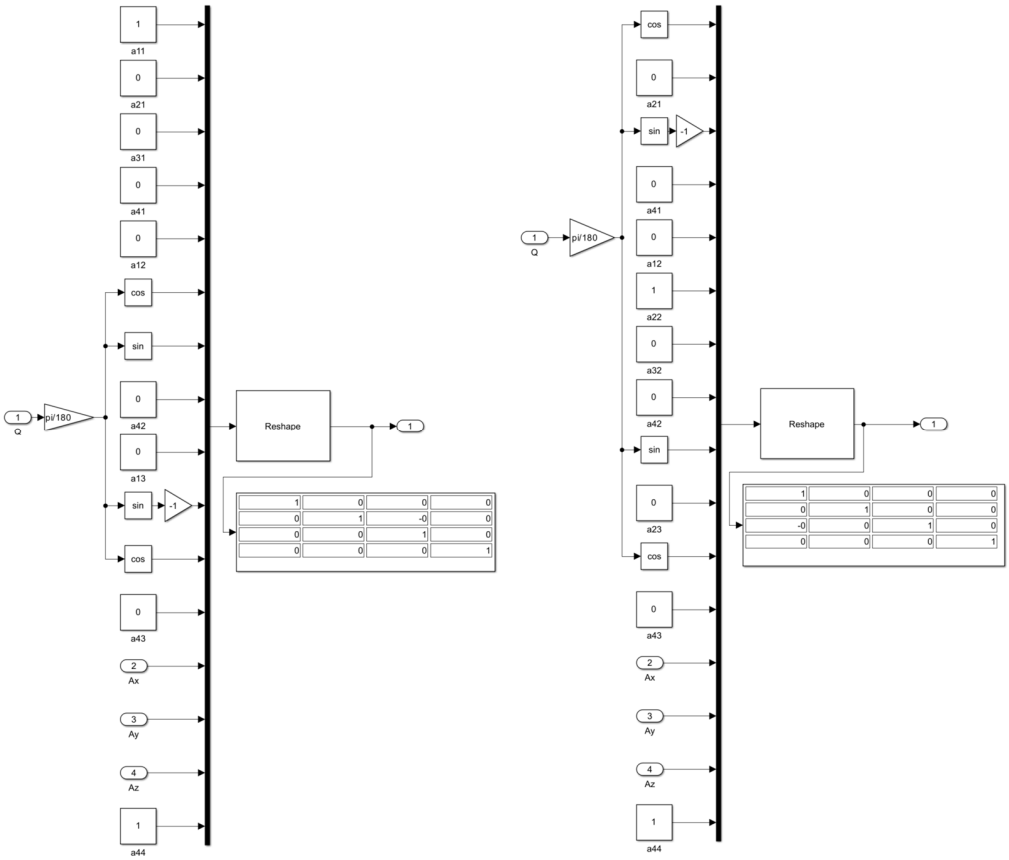


Figure 5. Rotation around the (**a**) X- and (**b**) Y-axis sub-system models.

For the rotation couple around the y axis, the homogeneous transfer matrix will be used and converted to Simulink:

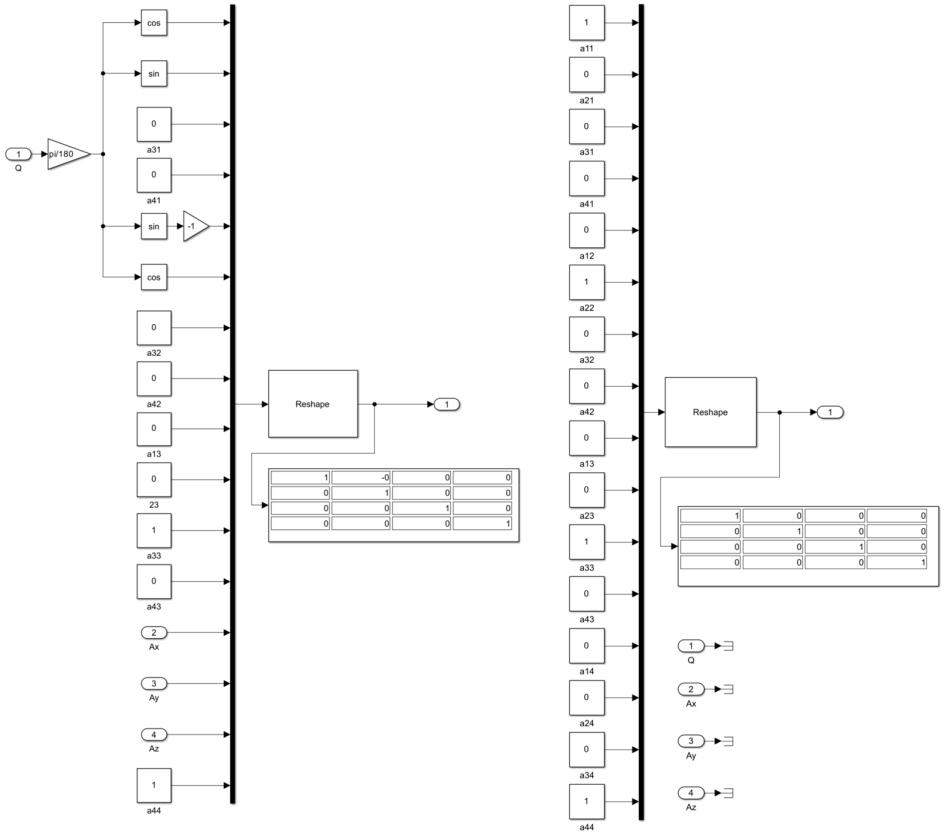
$$A_{i,i+1}^{R,y} = \begin{bmatrix} \cos(q_i) & 0 & \sin(q_i) & a_x \\ 0 & 1 & 0 & a_y \\ -\sin(q_i) & 0 & \cos(q_i) & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

The Simulink model in which $A_{i,i+1}^{R,y}$ is adapted to the block diagram is presented in Figure 5.

For the rotational couple around the z axis, the homogeneous transfer matrix will be used and converted to Simulink:

$$A_{i,i+1}^{R,z} = \begin{bmatrix} \cos(q_i) & -sin\,(q_i) & 0 & a_x \\ \sin(q_i) & \cos(q_i) & 0 & a_y \\ 0 & 0 & 1 & a_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

The Simulink model in which $A_{i,i+1}^{R,z}$ is adapted to the block diagram is presented in Figure 6.



(**a**) (**b**)

Figure 6. (**a**) Rotation around the Z-axis and (**b**) null (matrix unit) sub-system models.

For the null sub-system, the unit matrix will be used and converted to Simulink. The Simulink model is presented in Figure 6.

To create the sub-systems, the following elements are used:

- Constant to set the fixed values of the 4×4 matrix elements;
- Input (q_i, a_x, a_y, a_z) and output ports;
- Terminator;
- Mux;
- Reshape to shape the input to a 4×4 matrix;
- Display to visualize the matrix.

2.3.2. Development of Movement Module Function

The blocks created in the previous subsection will be used to create the modular movement type selection block. This is presented in Figure 2 in the movement model sub-system.

To call the blocks already created, the MATLAB Simulink model block will be used, as well as a system constant called "Movement_Type". This will be generated with values from 0 to 6, as shown in Figure 7 in the "General Info about Type_of_Movement" box. Using the system's constant, the type of movement will be selected through a switch block. Once the movement is selected, all of the other unused matrices will result in the unity matrix. Therefore, in the end, all matrices will be multiplied, but the result of the calculation will depend only on the model chosen by the constant. The MATLAB Simulink model of the movement module is presented in Figure 7.

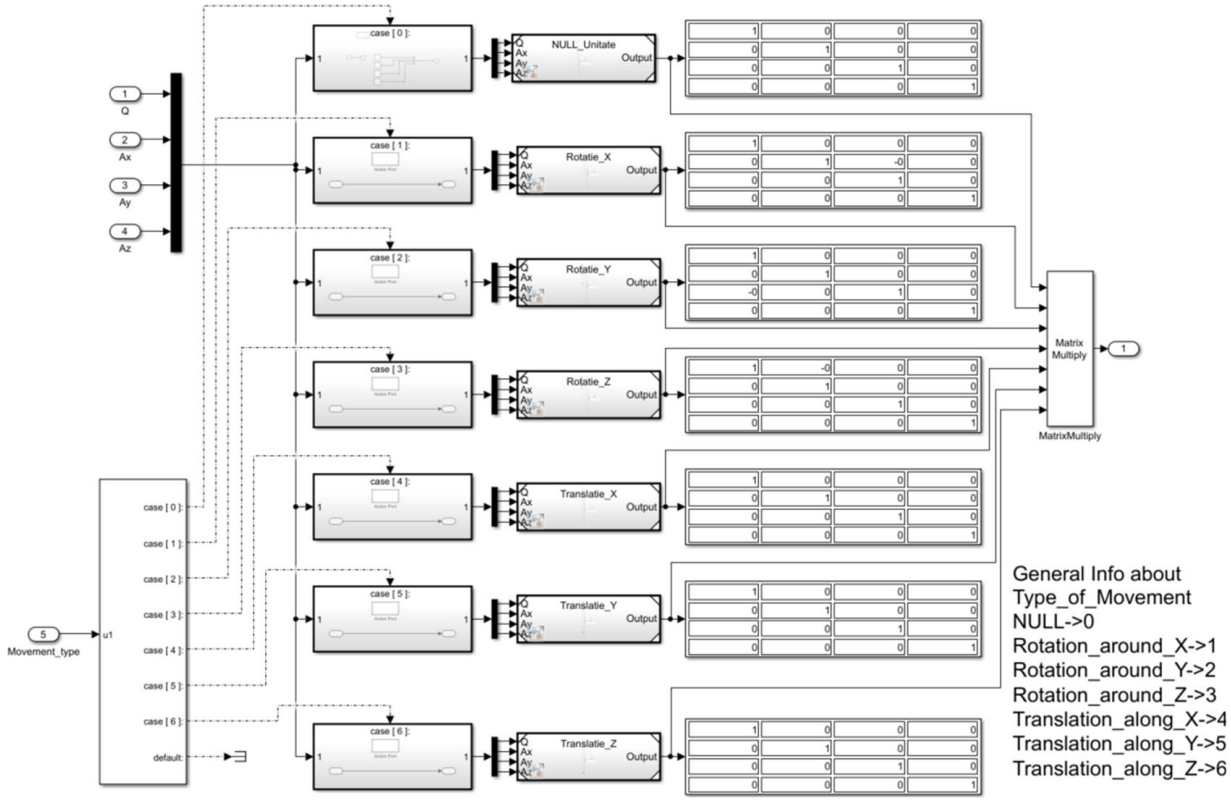


Figure 7. Movement type modular block.

To create the sub-system, the following elements are used:

- Input (q_i, a_x, a_y, a_z, $Movement\_type$) and output ports;
- Switch case block;
- Switch case action block;
- Model block;
- Display;
- Mux;
- Demux;
- Matrix multiply block.

2.3.3. Robotic Configuration Sub-System Model

In the following, the robot sub-system block presented in Figure 2 will be created. In its implementation, all previously developed sub-systems will be called upon.

A new constant will be used for the active number of modules, through which the inserted kinematic couplings will be unlocked. Similarly to the previous point, the Simulink model block will be used to call the module of the modular movement type created. At this point, the number of inputs for the robot block will increase, and they will be created for each individual kinematic couple. For this experiment, three kinematic couples will be used, as shown in Figure 8. In this system, the geometric parameters of the final effector will be entered as inputs, where a 4×4 matrix will be created, with the geometric parameters corresponding to the three-dimensional positioning.

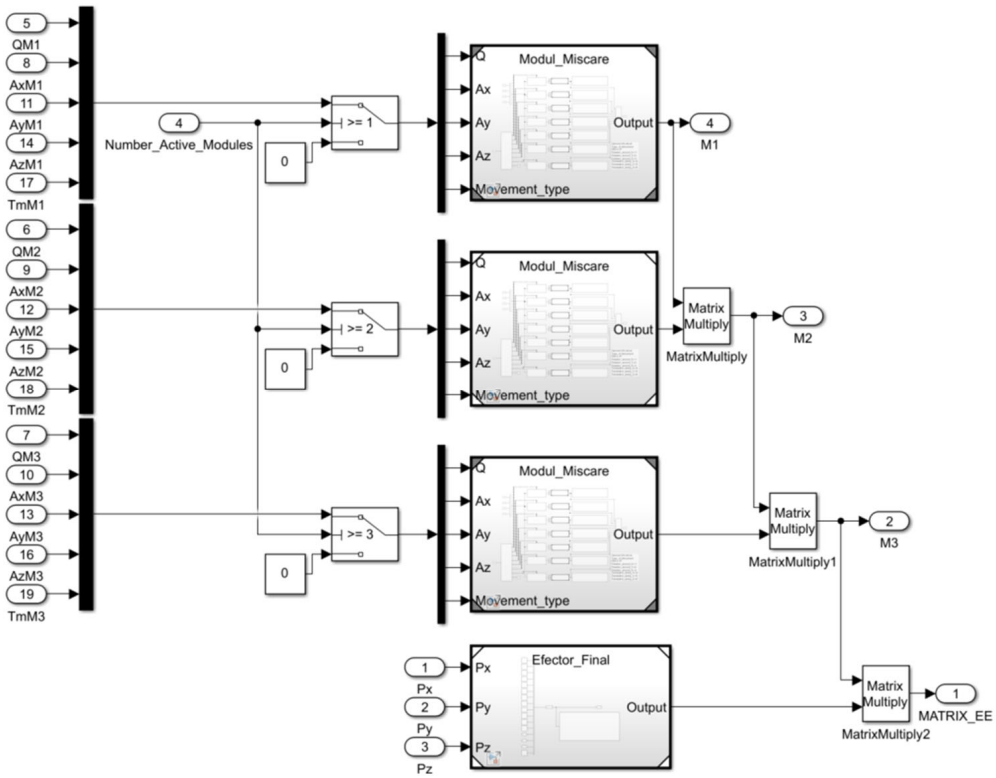


Figure 8. Modular robot block.

To create the sub-system, the following elements are used:

- Input (q_{Mi}, a_{xMi}, a_{yMi}, a_{zMi}, $Movement_{type}$, $Number_{Active_{Modulse}}$, P_x, P_y, P_z) and output ports;
- Switch block;
- Constant block;
- Model block;
- Mux;
- Demux;
- Matrix multiply block.

2.3.4. Displacement, Velocity, and Acceleration Sub-System Model

For the development of the next computational block, the output from the robot block will be used. To create the displacement, velocity, and acceleration graphs, the resulting final matrix of the end-effector, as well as each individual kinematic couple, will be analyzed and decomposed in order to perform the necessary calculations for the representation of each individual graph. This calculation module is shown in Figure 9.

To create the sub-system, the following elements are used:

- Input and output ports;
- Submatrix block;
- Derivative block;
- Mux;
- Display;
- Scope.

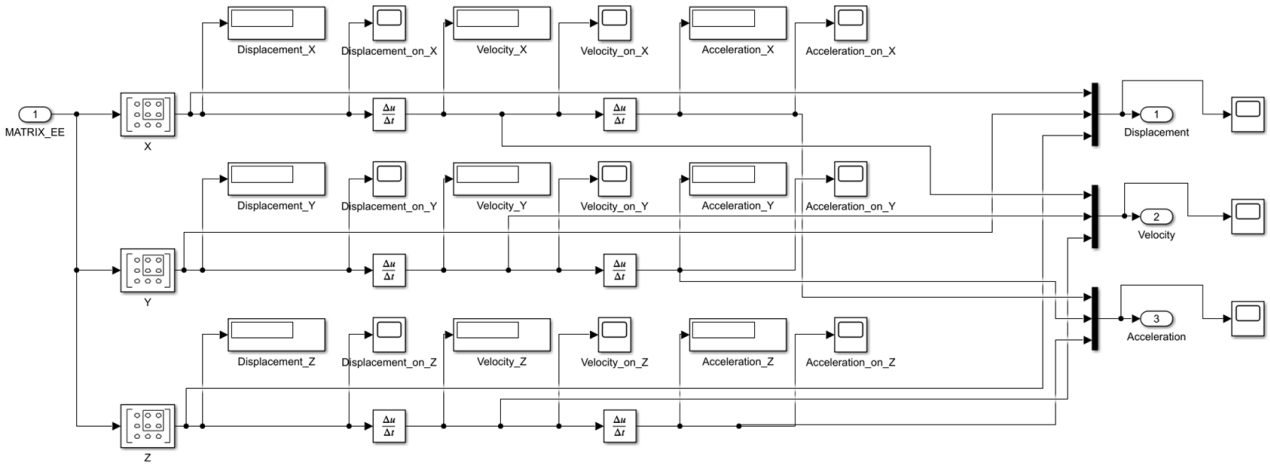


Figure 9. Displacement, velocity, and acceleration calculation model block.

2.3.5. Robot Generation Functionality Model

All modules presented in the previous sub-sections are included in the final implementation of the generating function of modular robots, which is presented in Figure 10. Here, the rotational and translational motion modules are incorporated, along with the motion type modules. All of them are present in the robot sub-system.

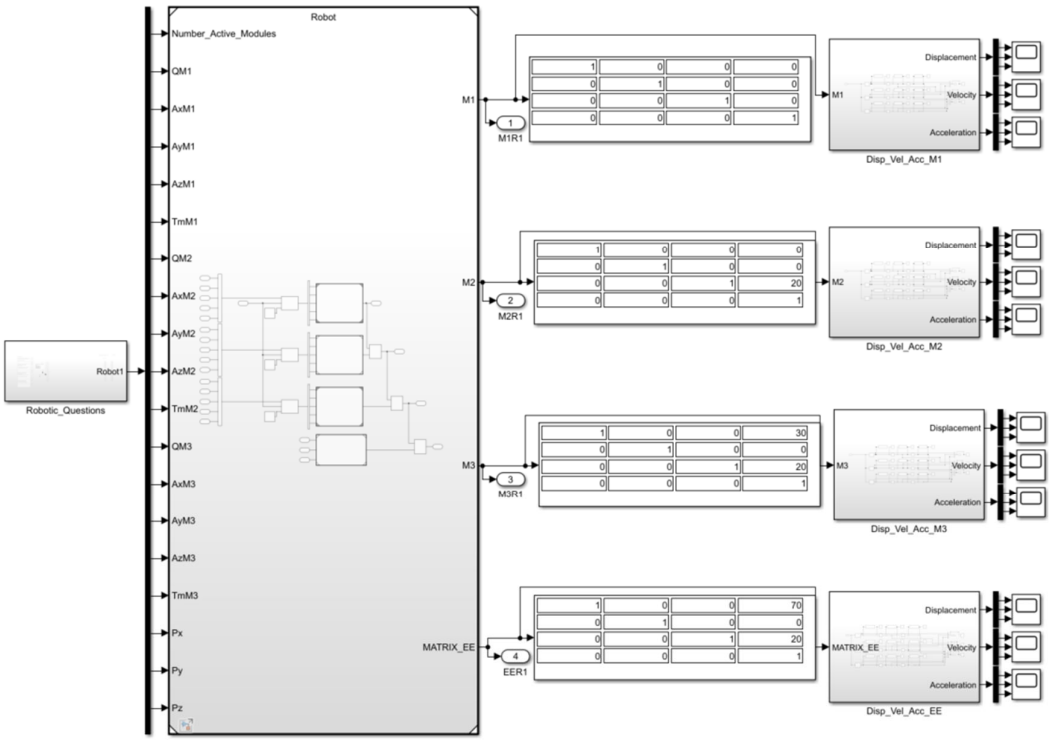


Figure 10. Generating function of modular robots.

Inside this block, all of the presented models in the previous sub-sections, and all of the performed calculations, will be included. Using this module, the construction of modular serial robots will be enhanced in terms of resource optimization and time, as the information is collected in one step. In the end, the development progress, along with the simplified information, will ease the system's understanding for new users, and even students will have the possibility to improve their knowledge in the calculation and development of serial robots.

3. Results

In order to validate the calculations of the function, the example of the $R_z R_y R_y$ robot, as shown in Figure 11, is proposed.

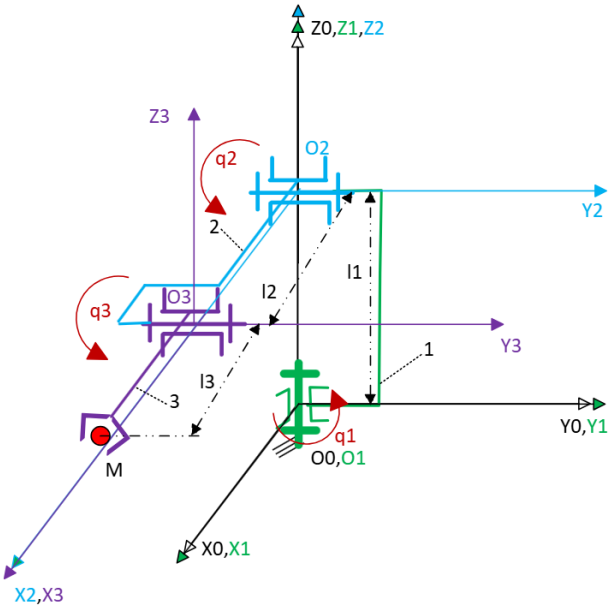


Figure 11. Schematic of the $R_z R_y R_y$ robot that will be analyzed.

The defining matrices of the robot will be written as follows:

$$A_1 = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$A_2 = \begin{bmatrix} \cos(q_2) & 0 & \sin(q_2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(q_2) & 0 & \cos(q_2) & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$A_3 = \begin{bmatrix} \cos(q_3) & 0 & \sin(q_3) & l_2 \\ 0 & 1 & 0 & 0 \\ -\sin(q_3) & 0 & \cos(q_3) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

$$P_M^3 = \begin{bmatrix} l_3 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{20}$$

where

$$l_1 = 20 \text{ [m]} \tag{21}$$

$$l_2 = 30 \text{ [m]} \tag{22}$$

$$l_3 = 40 \text{ [m]} \tag{23}$$

In order to complete the input information that will be used to validate the proposed serial robot, the questionnaire presented in Figure 12 will be completed. Based on the questionnaire answers, the robot drawn in Figure 11 will be built.

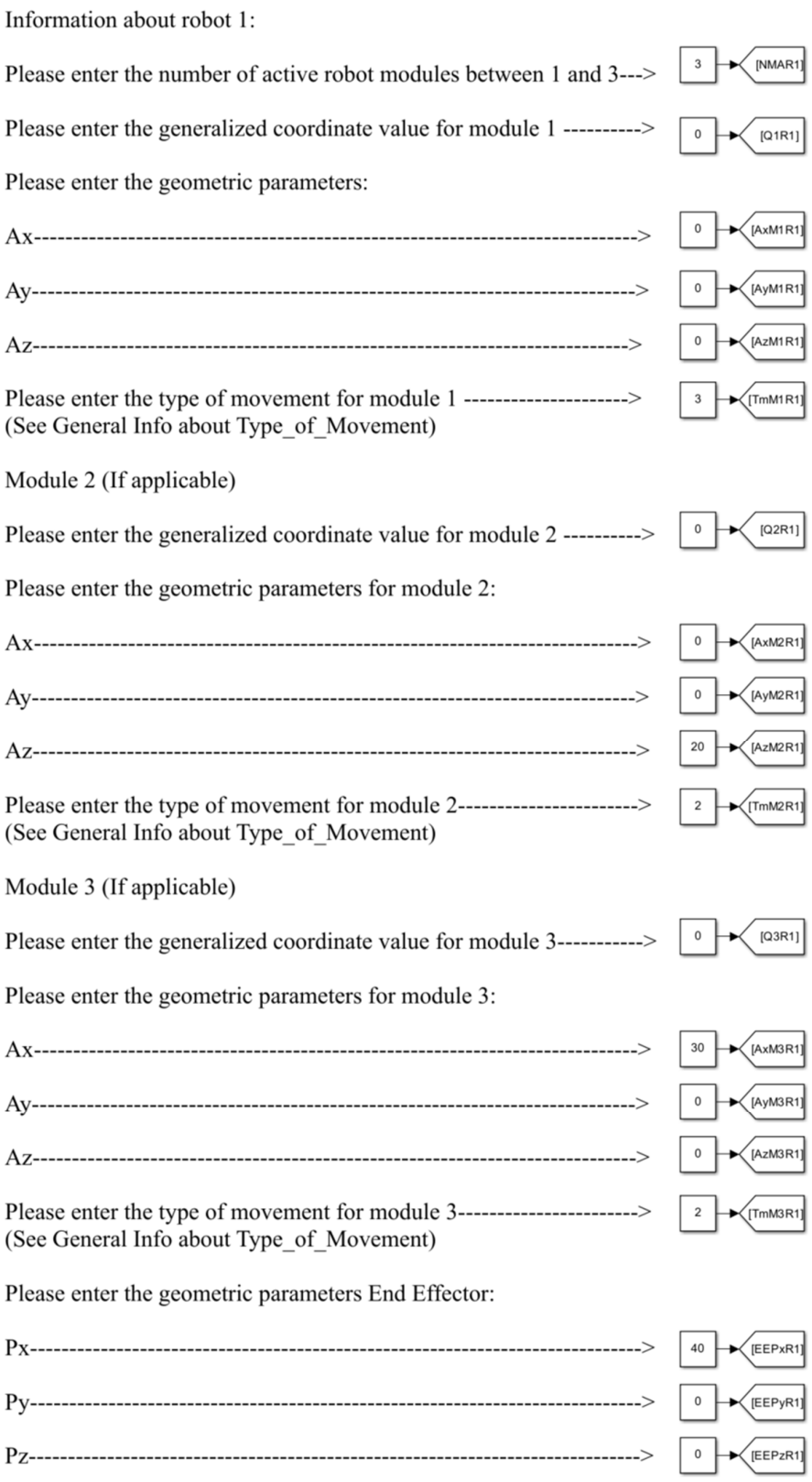


Figure 12. Robotic questions—questionnaire.

After running the created MATLAB Simulink model, the calculation results for the matrices of the M_1, M_2, M_3 couples and of the end-effector (the matrix Matrix_EE) will be obtained, as illustrated in Figure 13.

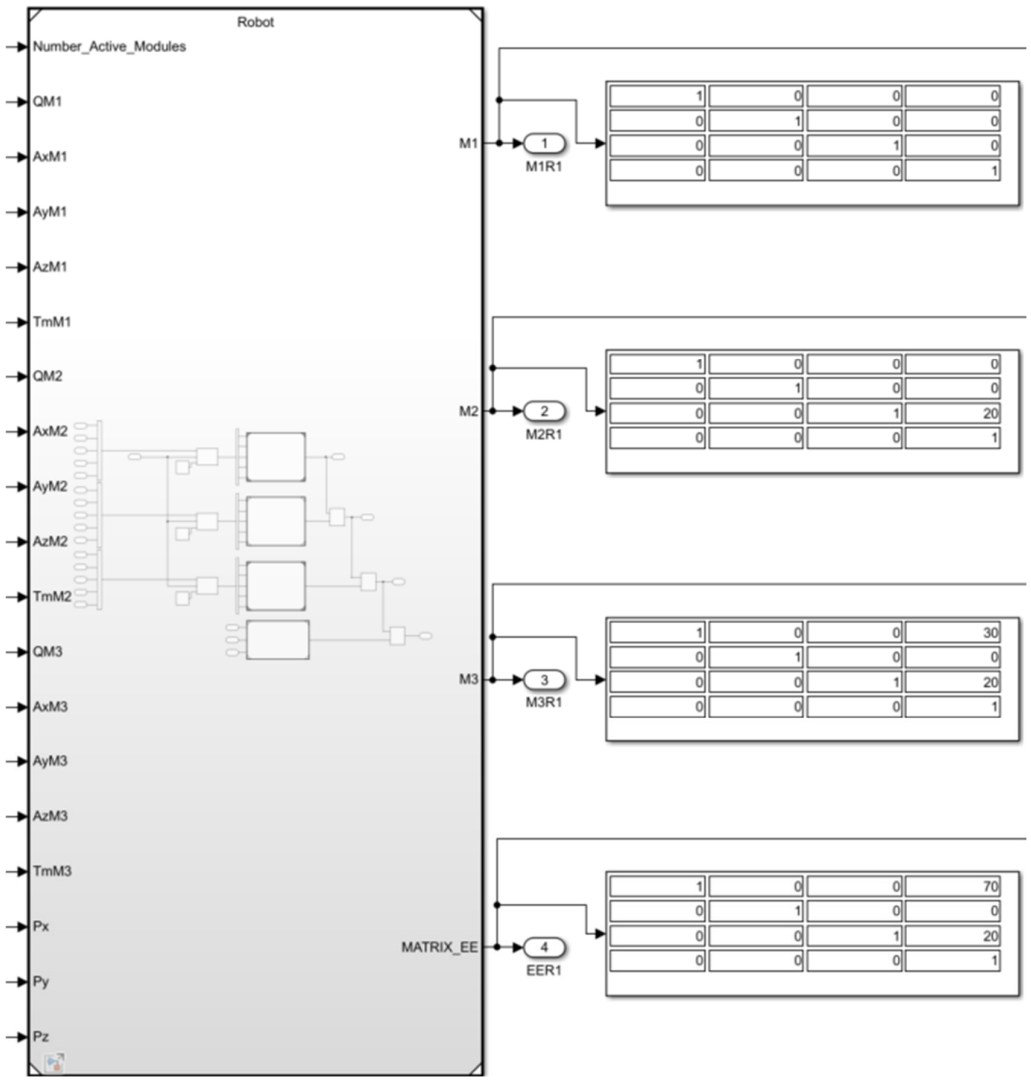


Figure 13. The result of the generating function of modular robots in the parking position.

In order to obtain a visual representation of the $R_zR_yR_y$ robot, Figure 14 was generated. In this figure, the plot of the $R_zR_yR_y$ robot is presented. The kinematic couples M_1, M_2, and M_3 are represented by a green sphere. The robot's end-effector is represented by a red sphere. Next to each kinematic couple and next to the end-effector are the 3D positional values, which are also presented in Figure 13 in the matrices M_1R_1, M_2R_1, M_3R_1, and EER_1.

In order to validate the calculations performed, together with the implementation of the functionality in MATLAB, additional validation was performed using the Mathcad Prime 7.0.0.0 software. Equations (7), (8), and (17)–(23) were introduced into the Mathcad software. The geometric parameters l_1, l_2, and l_3 were introduced, together with the generalized coordinates q_1, q_2, and q_3, with values of zero for generalized coordinates, to calculate the parking position. The calculation of the A_1, A_2, A_3, and P_M^3 matrices of the robot was performed. Finally, the kinematic couple matrices M_1R_1, M_2R_1, M_3R_1 were calculated together with the final effector position matrix EER_1, which are presented in Figure 13. The mathematical validation through the Mathcad software is presented in Figure 15.

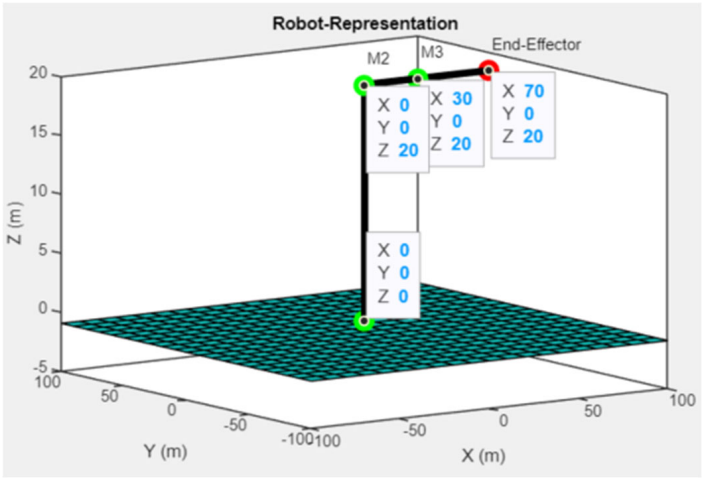


Figure 14. The plot of the robot $R_z R_y R_y$.

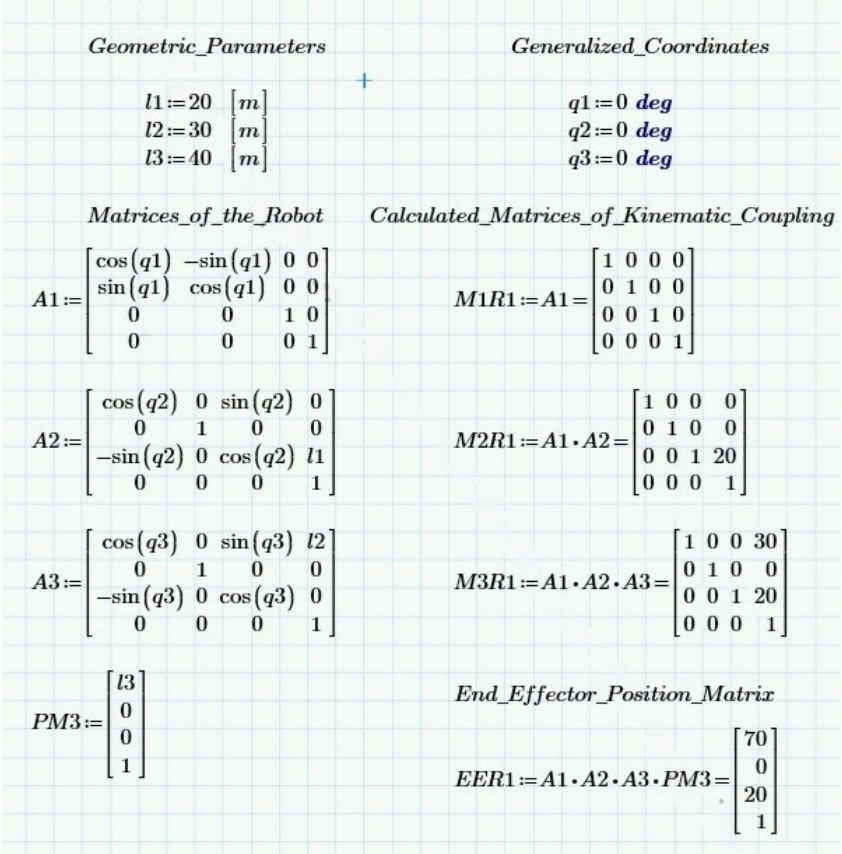


Figure 15. Mathematical validation through Mathcad software.

The authors also propose a demonstration of the representation of the displacement, velocity, and acceleration graphs; for this, a repeating signal, as shown in Figure 16, will be used. It will be attached to the kinematic coupling R_y and a repetitive movement will be implemented for the generalized coordinate q_3 of the robot. This will be linked to $[Q_3 R_1]$ from the robotic questions in Figure 12.

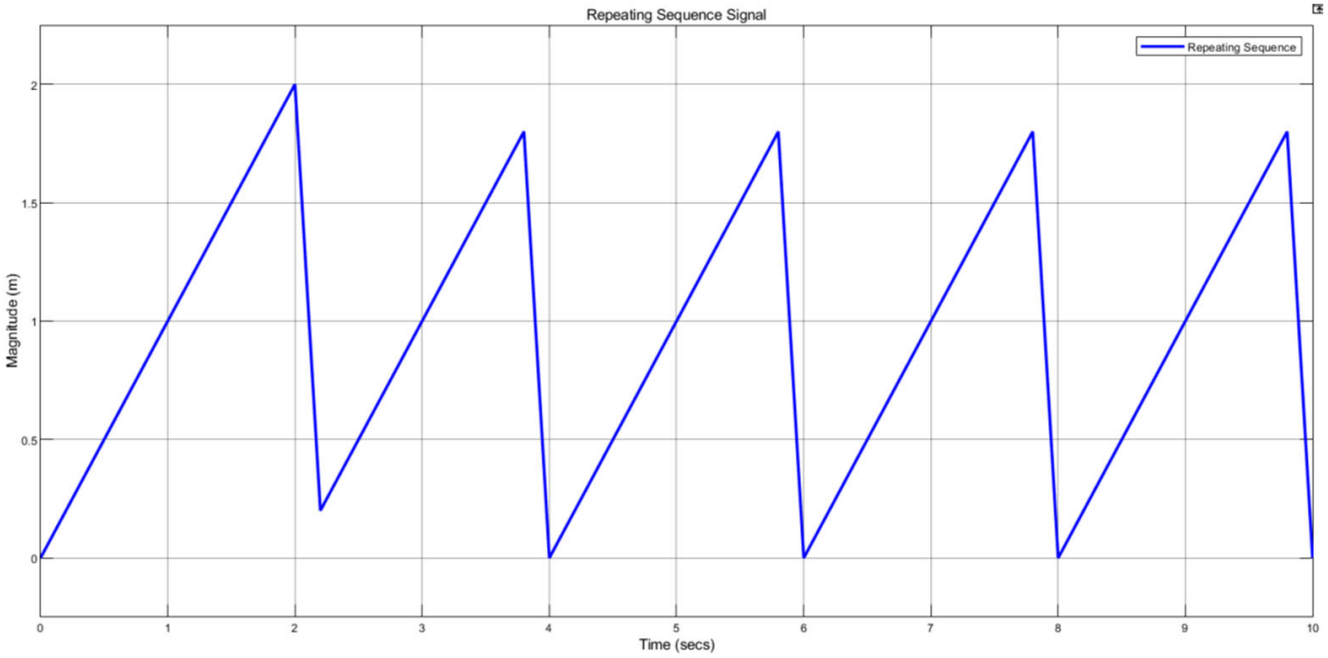


Figure 16. Repeating signal attached to Q_3, simulation graph.

Considering the repetitive signal 0 [m]–2 [m], which is attached to q_3, shown in Figure 16, a displacement on the Z-axis will be observed. Moreover, an associated oscillation on the Z-axis of the end-effector with a displacement of ± 2 [m] is visible.

In Figure 17, the displacement on the X-, Y- and Z-axes of the end-effector is represented, along with the variation statistics on the Z-axis.

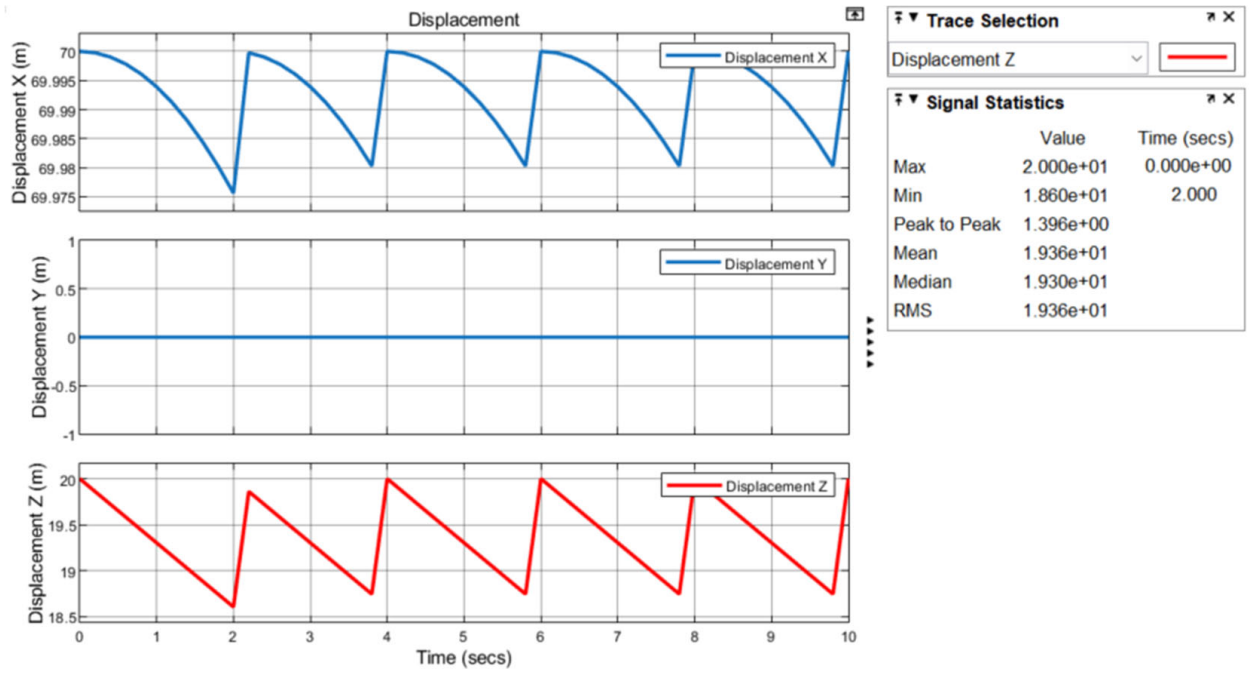


Figure 17. Displacement graph on the X-, Y-, and Z-axes of the end-effector.

In Figure 18, the velocity is represented on the X-, Y-, and Z-axes of the end-effector, along with the variation statistics on the Z-axis.

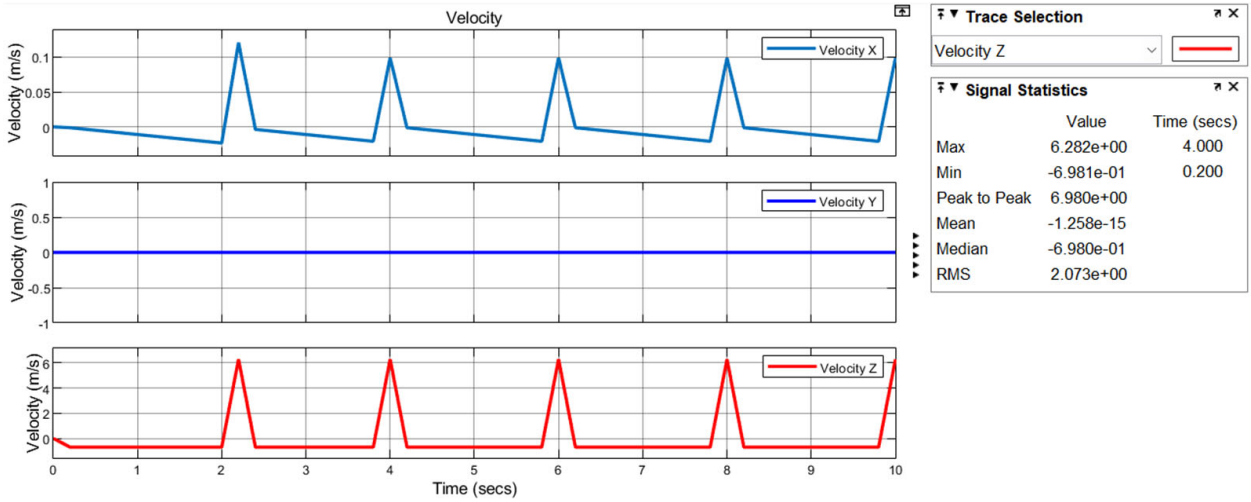


Figure 18. Velocity graph on the X-, Y-, and Z-axes of the end-effector.

In Figure 19, the acceleration is represented on the X-, Y-, and Z-axes of the end-effector, along with the variation statistics on the Z-axis.

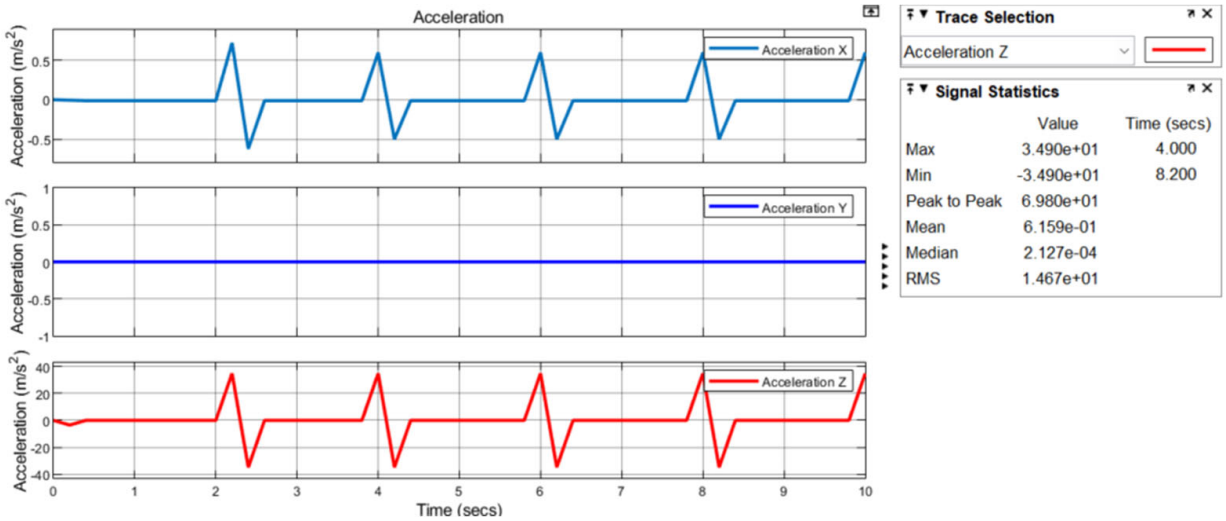


Figure 19. Acceleration graph on the X-, Y-, and Z-axes of the end-effector.

In summary, the ease of completing the input values in the application and its user-oriented implementation can be observed. The robot schematization will be created in order to provide an overview of the desired robot and the geometric parameters needed to complete the guidance questionnaire. In a small amount of time, it is possible to obtain the results of the matrix calculations for the serial robot. After performing the parking position calculation, a repetitive motion equation is introduced to demonstrate the calculation of the displacements, velocity, and accelerations of the end-effector. The displacement graphs on X, Y, and Z, together with the velocity and accelerations on the same axes for the end-effector will be accessed instantly for analysis, as shown in Figures 15–17. In these analysis windows, the statistics of each signal can be followed. The blocks can be opened and analyzed for each kinematic coupling in particular. Returning to the questionnaire, by modifying the geometric parameters, the equations of motion, the type of movement of the couplings, or even their number, the user has the possibility to reconfigure the robot and analyze different optimal scenarios for the projects in which it will be used. By using this

function, the user will no longer need to use multiple software packages for the calculation and analysis of serial robots.

4. Conclusions

Considering the above, using the MATLAB programming environment, the authors of this paper propose the development of a functionality model block developed in MATLAB Simulink, oriented towards the end-user/client, using the model-based development method. This function can be used and incorporated into a real module to streamline and simplify serial robot calculations and optimize the use of resources and time. In order to realize a complex project, the possibility of generating one or more robots whose morphology and implementation are modular can be realized through this function.

In this paper, the aim was to create a function using the MATLAB Simulink software in which the user would have the possibility to configure serial robots containing fifth-class kinematic couplings, rotation, and translation in their structures. The mathematical equations of rotation and translation were modeled on all three axes, X, Y, and Z. A modular movement block and a modular operational block were created for serial robots, which can be accessed with the help of guiding questions. Calculation and representation blocks for the displacement, velocity, and acceleration were created for the final effector and for all individual kinematic couplings.

Using this function, users will no longer need to use additional software or other libraries to calculate robotic matrices, with all of the information being inserted inside the function block. By introducing the inputs, namely the generalized coordinates, the geometric parameters, and the number of kinematic couples and their types, the function can calculate the direct kinematics of the serial robot and create displacement, velocity, and acceleration graphs for the end-effector and for each kinematic couple.

Future Opportunities

As future aspirations, for the generating function for serial robots, a user-oriented interface will be developed. The application will have a user-oriented interface, which will guide the user step by step to introduce the necessary information in the block, in order to achieve the goal of serial robot generation. After obtaining a configuration that complies with the basic requirements of the applied project, a working space will be configured. Both the robot and the working environment will be configured according to the needs of each.

In the next stage, after the generation of the robot is completed, users will have working programs for the functionalities of the robots, graphs to use for the trajectory calculations, and the possibility to generate and simulate them inside a graphic interface. By using this function, a result will be achieved rapidly in terms of the time invested. Users have the advantage of being able to focus on the project, rather than on the administrative aspects regarding the connectivity of the tools and knowledge of the libraries. They will be able to benefit from an application that can easily guide them towards the realization of their projects. Additionally, they will have the possibility to implement the entire workspace in a single environment with high-precision calibration. Design errors will be easily identifiable, with one of the aims being to avoid them starting from the initial steps of the project.

Author Contributions: Conceptualization, E.A.G.; methodology, E.A.G.; software, E.A.G.; formal analysis, E.A.G. and M.B.; resources, E.A.G.; writing—original draft preparation, E.A.G.; writing—review and editing, E.A.G. and M.B.; visualization, E.A.G.; supervision, M.B.; project administration, M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jia, J.; Sun, X. Structural Optimization Design of a Six-Degrees-of-Freedom Serial Robot with Integrated Topology and Dimensional Parameters. *Sensors* **2023**, *23*, 7183. [CrossRef] [PubMed]
2. Tsai, C.H.; Hernandez, E.E.; You, X.W.; Lin, H.Y.; Chang, J.Y. RoboTwin Metaverse Platform for Robotic Random Bin Picking. *Appl. Sci.* **2023**, *13*, 8779. [CrossRef]
3. Yim, M.; Duff, D.G.; Roufas, K.D. PolyBot: A modular reconfigurable robot. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; IEEE: Piscataway, NJ, USA; Volume 1, pp. 514–520.
4. Moubarak, P.; Ben-Tzvi, P. Modular and reconfigurable mobile robotics. *Robot. Auton. Syst.* **2012**, *60*, 1648–1663. [CrossRef]
5. Hazem, Z.B.; Ince, R.; Dilibal, S. Joint Control Implementation of 4-DOF Robotic Arm Using Robot Operating System. In Proceedings of the 2022 International Conference on Theoretical and Applied Computer Science and Engineering (ICTASCE), Ankara, Turkey, 29 September–1 October 2022; IEEE: Piscataway, NJ, USA; pp. 72–77.
6. Zhang, C.; Zhu, P.; Lin, Y.; Jiao, Z.; Zou, J. Modular soft robotics: Modular units, connection mechanisms, and applications. *Adv. Intell. Syst.* **2020**, *2*, 1900166. [CrossRef]
7. Kellner, M.I.; Madachy, R.J.; Raffo, D.M. Software process simulation modeling: Why? what? how? *J. Syst. Softw.* **1999**, *46*, 91–105. [CrossRef]
8. Chaturvedi, D.K. *Modeling and Simulation of Systems Using MATLAB and Simulink*; CRC Press: Boca Raton, FL, USA, 2017.
9. Corke, P.I. A computer tool for simulation and analysis: The Robotics Toolbox for MATLAB. In Proceedings of the 1995 National Conference of the Australian Robot Association, Melbourne, Australia, 5–7 July 1995; pp. 319–330.
10. Baba, Y.; Nobeoka, K. Towards knowledge-based product development: The 3-D CAD model of knowledge creation. *Res. Policy* **1998**, *26*, 643–659. [CrossRef]
11. Becker, M.C.; Salvatore, P.; Zirpoli, F. The impact of virtual simulation tools on problem-solving and new product development organization. *Res. Policy* **2005**, *34*, 1305–1321. [CrossRef]
12. Righettini, P.; Strada, R.; Cortinovis, F. Neural network mapping of industrial robots' task times for real-time process optimization. *Robotics* **2023**, *12*, 143. [CrossRef]
13. Struss, P.; Price, C. Model-Based Systems in the Automotive Industry. *AIMag* **2003**, *24*, 17.
14. Qureshi, M.O.; Syed, R.S. The impact of robotics on employment and motivation of employees in the service sector, with special reference to health care. *Saf. Health Work.* **2014**, *5*, 198–202. [CrossRef] [PubMed]
15. De Backer, K.; DeStefano, T.; Menon, C.; Suh, J.R. *Industrial Robotics and the Global Organisation of Production*; OECD Science, Technology and Industry Working Papers, No. 2018/03; OECD Publishing: Paris, France, 2018. [CrossRef]
16. Piltan, F.; Yarmahmoudi, M.H.; Shamsodini, M.; Mazlomian, E.; Hosainpour, A. PUMA-560 robot manipulator position computed torque control methods using Matlab/Simulink and their integration into graduate nonlinear control and Matlab courses. *Int. J. Robot. Autom.* **2012**, *3*, 167–191.
17. Toledano-García, A.A.; Pérez-Cabrera, H.R.; Ortega-Cabrera, D.; Navarro-Durán, D.; Pérez-Hernández, E.M. Trajectory Generator System for a UR5 Collaborative Robot in 2D and 3D Surfaces. *Machines* **2023**, *11*, 916. [CrossRef]
18. Karupusamy, S.; Maruthachalam, S.; Veerasamy, B. Kinematic Modeling and Performance Analysis of a 5-DoF Robot for Welding Applications. *Machines* **2024**, *12*, 378. [CrossRef]
19. Carbonari, L.; Palpacelli, M.C.; Callegari, M. Inverse kinematics of a class of 6R collaborative robots with non-spherical wrist. *Robotics* **2023**, *12*, 36. [CrossRef]
20. Noskievič, P.; Walica, D. Design and realisation of the simulation model of the Stewart platform using the MATLAB-Simulink and the Simscape Multibody Library. In Proceedings of the 2020 21th International Carpathian Control Conference (ICCC), High Tatras, Slovakia, 27–29 October 2020; IEEE: Piscataway, NJ, USA; pp. 1–5.
21. Mohapatra, S.; Srivastava, R.; Khera, R. Implementation of a two wheel self-balanced robot using MATLAB Simscape Multibody. In Proceedings of the 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), Gangtok, India, 25–28 February 2019; IEEE: Piscataway, NJ, USA; pp. 1–3.