

Article

Inverse Kinematics Data Adaptation to Non-Standard Modular Robotic Arm Consisting of Unique Rotational Modules

Štefan Ondočko¹, Jozef Svetlík^{1,*} , Michal Šašala¹, Zdenko Bobovský² , Tomáš Stejskal¹, Jozef Dobránsky³ , Peter Demeč¹ and Lukáš Hrivniak¹

¹ Department of Manufacturing Machinery and Robotics, Faculty of Mechanical Engineering, The Technical University of Košice, Letná 9, 04001 Košice, Slovakia; stefan.ondocko@tuke.sk (Š.O.); michal.sasala@tuke.sk (M.Š.); tomas.stejskal@tuke.sk (T.S.); peter.demec@tuke.sk (P.D.); lukas.hrivniak@tuke.sk (L.H.)

² Department of Robotics, Faculty of Mechanical Engineering, VSB—TU Ostrava, 17. listopadu 2172/15, 708 00 Ostrava-Poruba, Czech Republic; zdenko.bobovsky@vsb.cz

³ Department of Automotive and Manufacturing Technologies, Faculty of Manufacturing Technologies with a seat in Prešov, Technical University of Košice, Štúrova 31, 08001 Prešov, Slovakia; jozef.dobransky@tuke.sk

* Correspondence: jozef.svetlik@tuke.sk; Tel.: +421-55-602-2195

Abstract: The paper describes the original robotic arm designed by our team kinematic design consisting of universal rotational modules (URM). The philosophy of modularity plays quite an important role when it comes to this mechanism since the individual modules will be the building blocks of the entire robotic arm. This is a serial kinematic chain with six degrees of freedom of unlimited rotation. It was modeled in three different environments to obtain the necessary visualizations, data, measurements, structural changes measurements and structural changes. In the environment of the CoppeliaSim Edu, it was constructed mainly to obtain the joints coordinates matching the description of a certain spatial trajectory with an option to test the software potential in future inverse task calculations. In Matlab, the model was constructed to check the mathematical equations in the area of kinematics, the model's simulations of movements, and to test the numerical calculations of the inverse kinematics. Since the equipment at hand is subject to constant development, its model can also be found in SolidWorks. Thus, the model's existence in those three environments has enabled us to compare the data and check the models' structural designs. In Matlab and SolidWorks, we worked with the data imported on joints coordinates, necessitating overcoming certain problems related to calculations of the inverse kinematics. The objective was to compare the results, especially in terms of the position kinematics in Matlab and SolidWorks, provided the initial joint coordinate vector was the same.

Keywords: Matlab; CoppeliaSim Edu; V-Rep; SolidWorks; kinematics; inverse kinematics (IK); manufacturing technology; modular robots; coordinate transformation



Citation: Ondočko, Š.; Svetlík, J.; Šašala, M.; Bobovský, Z.; Stejskal, T.; Dobránsky, J.; Demeč, P.; Hrivniak, L. Inverse Kinematics Data Adaptation to Non-Standard Modular Robotic Arm Consisting of Unique Rotational Modules. *Appl. Sci.* **2021**, *11*, 1203. <https://doi.org/10.3390/app11031203>

Academic Editor: José Luis Guzmán Sánchez

Received: 6 January 2021

Accepted: 25 January 2021

Published: 28 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The paper addresses data processing (in particular those of joints coordinates, which are, in the case at hand, the angles of rotation) designated for a stationary robotic arm composed of the so-called URM modules, described in detail in [1–3]. We briefly mention their main attributes, which are unlimited rotation of the module around its own axis, availability of intrinsic power and modularity units. Thanks to the modularity and the advantages it represents [4–7], individual modules can be used for building various configurations and thus to adjust the arm to a function required of it, to create machines with different mobility capabilities, several degrees of freedom. In general, modular and reconfigurable robots offer great versatility, robustness and—thanks to their series production—low costs, which is mentioned by many authors addressing this issue [8–10]. Those modular, reconfigurable robots that consist of many modules (the number of their degrees of freedom is usually

much greater than 6) have the ability to reconfigure themselves into a large number of shapes. If this is the case, the robot can change its shape to meet the requirements of different tasks.

Modular robotic systems are systems composed of modules that can be disconnected and reconnected in various configurations, subject to maintaining the precision and rigidity parameters, which can thus create a new system enabling new or value-added functions [11]. The concept of the URM system has been conceived in accordance with the total productive maintenance principal implementation rules [12,13]. Individual rotation positions are the result of the inverse kinematics of the mentioned robot's arm. When it comes to kinematics of an inverse position, we look for such joint coordinate vectors of the open kinematic chain (assuming that the mechanism's size is known) as would suit the required position and the orientation of the end effector's coordinate system. Unlike the forward task, where the position vector is a function of the joint coordinate vector, the inverse task is substantially more complex because it necessitates solving a set of strongly nonlinear algebraic equations. These problems were first postulated by Paden B. [14] and Kahan W [15]. In most cases, these systems cannot be solved analytically. Thus, various types of iterative numerical methods are used, most often employing the Jacobian [16–21]. In calculating the inverse function, the inability to solve the task stems from the (configuration) workspace limitation, delimited by the configuration itself and the mechanism's physical properties. If we are located outside this workspace, there is, of course, no solution. Several solutions may exist since the end effector's defined position may be obtained in several manners. The situation can be even more varied in the case of the so-called redundant manipulators, where the joint coordinate vector's number of elements is greater than the degree of freedom in the given workspace or plane. This results in an infinite number of configurations through which the defined position and orientation can be reached. We do not deal directly with the inverse function in this paper. The respective joint coordinates are obtained from the model's CoppeliaSim Edu simulation (Coppelia Robotics AG, 8049 Zürich, Switzerland). The software calculating the inverse kinematics uses the pseudoinverse computational method, also called the Moore–Penrose method and the method of dampened least squares (DLS), also called the Levenberg–Marquardt method. More information about these and other methods can be found in [16,22–24]. Since other URM module configurations are contemplated for the future, or the redundant manipulator creation will be requested, it was mandatory to search for flexible solutions. To this end, the inverse kinematics is addressed in the CoppeliaSim Edu software. When the model is created in this environment, it enables the simulation of many types of movements and also the design of a working trajectory taking into account hurdles in the robotic arm's configuration space. Joint coordinate vector coordinates are obtained from the model created in the CoppeliaSim Edu. Thus, we have the individual joints coordinates available, which is necessary for their subsequent implementation into the models, this time in Matlab (MathWorks, 1 Apple Hill Drive, Natick, MA 01760, USA) and in SolidWorks (Dassault Systèmes, 10 rue Marcel Dassault, CS 40501, 78946 Vélizy-Villacoublay CEDEX, France). These data are subsequently compared to check the models' designs in the respective software environments. Should the designer's work show ideal precision, these results should be, especially for the position kinematics, identical. This should hold regardless of the fact that two different software environments are used.

2. Model Creation and Data Processing

Relations describing the position kinematics by means of position vectors p_i , where $i \in \{1; 7\}$ at individual open kinematic chain segments have also been derived in [25,26]. This is the case of the robotic arm's forward kinematics. The relations serve the purpose of calculating the position of a point that would enable the attachment of either an effector or another device. Thus, the position of a point $[p_{xi}, p_{yi}, p_{zi}]$ on the module will be defined by the function of the rotation positions φ_i , ϑ_i and the segment size of the structure at hand— $|r_i|$; $p_i = f(\varphi_i, \vartheta_i, r_i)$. The basis of the arm [1,2] is an autonomous, cylinder-shaped

module. Such modules have a single degree of freedom, namely the rotational one. The rotation happens around the module's main axis and is not limited in the interval under consideration ranging from 0 to 360 degrees. The rotation may occur continuously in either direction of rotation without limitation. The modules and joints are contemplated to be perfectly solid bodies. The individual modules' axes of rotation cross each other at the point where they are joined by passive joints, Figure 1. The distances of section points in space are quantified by the r_i vector (a kinematic chain segment), and this vector has its own Cartesian system of $S_i\{O_i, x_i, y_i, z_i\}$. The segment's rotation around the z_i axis is defined by the rotation matrix $R_{z_i}(\varphi_i)$ [3–5]. The segment's rotation around the y_i axis is defined by the rotation matrix $R_{y_i}(\vartheta_i)$.

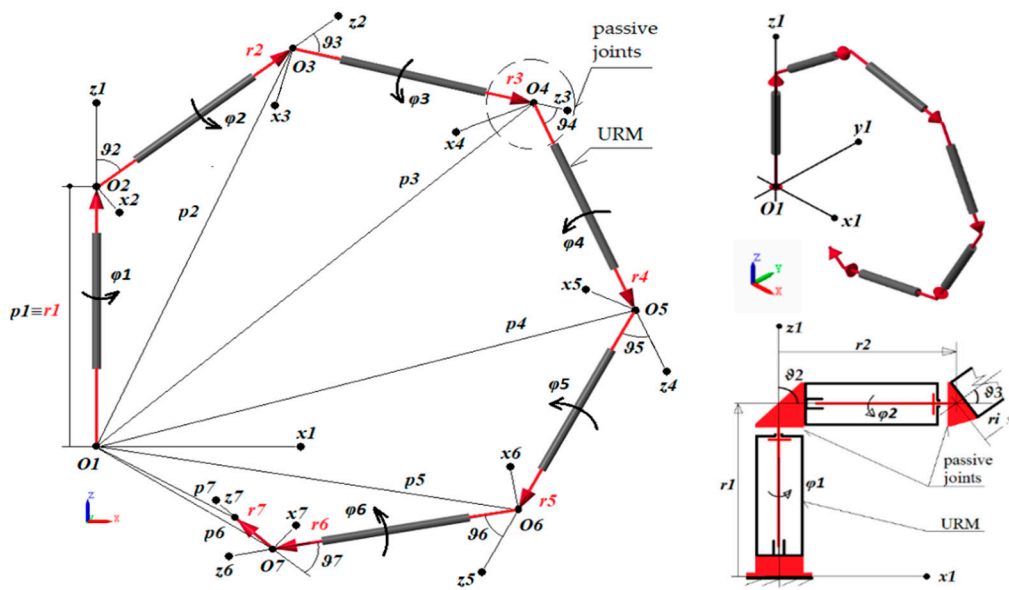


Figure 1. Robotic arm vector model composed of the universal rotational modules (URM) modules.

Thus, we can write the following for $R_{z_i}(\varphi_i)$:

$$R_{z_i}(\varphi_i) = \begin{bmatrix} \cos(\varphi_i) & -\sin(\varphi_i) & 0 \\ \sin(\varphi_i) & \cos(\varphi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Moreover, the following for $R_{y_i}(\vartheta_i)$:

$$R_{y_i}(\vartheta_i) = \begin{bmatrix} \cos(\vartheta_i) & 0 & \sin(\vartheta_i) \\ 0 & 1 & 0 \\ -\sin(\vartheta_i) & 0 & \cos(\vartheta_i) \end{bmatrix} \quad (2)$$

The r_1 vector is connected to the base perpendicularly to the plane with the x_1, y_1 axes. $p_1 \equiv r_1$, because the coordinate system is orthogonal. According to Figure 1, the following applies to the position vector p_1 :

$$p_1 = R_{z1}(\varphi_1) \times r_1 \quad (3)$$

According to Figure 1, the following applies to the position vectors p_2, p_3, p_4 , to p_i :

$$p_2 = p_1 + R_{z1}(\varphi_1) \times R_{y2}(\vartheta_2) \times R_{z2}(\varphi_2) \times r_2 \quad (4)$$

$$p_3 = p_2 + R_{z1}(\varphi_1) \times R_{y2}(\vartheta_2) \times R_{z2}(\varphi_2) \times R_{y3}(\vartheta_3) \times R_{z3}(\varphi_3) \times r_3 \quad (5)$$

$$p_4 = p_3 + R_{z1}(\varphi_1) \times R_{y2}(\vartheta_2) \times R_{z2}(\varphi_2) \times R_{y3}(\vartheta_3) \times R_{z3}(\varphi_3) \times R_{y4}(\vartheta_4) \times R_{z4}(\varphi_4) \times r_4 \quad (6)$$

A general position vector formula of this p_i structure will thus be:

$$p_{i+1} = R_{z1}(\varphi_1) \times \left[r_1 + \sum_{k=1}^i \left[\prod_{j=1}^k (R_{y(j+1)}(\vartheta_{j+1}) \times R_{z(j+1)}(\varphi_{j+1})) \right] \times r_{k+1} \right] \quad (7)$$

The resulting orientation of the r_i vector defined by the Euler's angle can be seen in [27] according to the $R_z(\gamma)R_y(\beta)R_x(\alpha)$ option will be determined from the relation below based on the final arm rotation:

$$R_{ZYX(i+1)} = R_{z1}(\varphi_1) \times \prod_{j=1}^i (R_{y(j+1)}(\vartheta_{j+1}) \times R_{z(j+1)}(\varphi_{j+1})) \quad (8)$$

Generally speaking, the final shape of the $R_{ZYX(i+1)}$ in terms of its i -segment will look as follows:

$$R_{ZYX(i+1)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad (9)$$

where a_{ij} for $i = 1, 2, 3, j = 1, 2, 3$ are the matrix elements. Considering the $R_z(\gamma)R_y(\beta)R_x(\alpha)$ option, the Euler's angles will then be as follows:

$$\alpha = \arctan\left(\frac{a_{32}}{a_{33}}\right) \quad (10)$$

$$\beta = \arcsin(-a_{31}) \quad (11)$$

or also:

$$\beta = \arctan\left(\frac{-a_{31}}{\sqrt{a_{32}^2 + a_{33}^2}}\right) \quad (12)$$

$$\gamma = \arctan\left(\frac{a_{21}}{a_{11}}\right) \quad (13)$$

Thus, for example, in the case of an arm with 3 active degrees of freedom, i.e., for the r_4 vector, the final rotation according to Equation (8) will be as follows:

$$R_{ZYX4} = R_{z1}(\varphi_1) \times R_{y2}(\vartheta_2) \times R_{z2}(\varphi_2) \times R_{y3}(\vartheta_3) \times R_{z3}(\varphi_3) \times R_{y4}(\vartheta_4) \quad (14)$$

Our case involves 6 degrees of freedom.

2.1. Modeling in CoppeliaSim Edu

An open kinematic chain model was created in CoppeliaSim Edu, in Figure 2, composed of modules featuring 6 degrees of freedom.

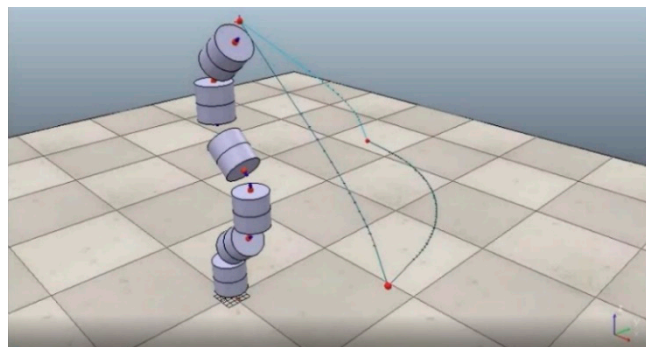


Figure 2. A model created in CoppeliaSim Edu.

The purpose of this model was especially calculating the inverse task from an arbitrary spatial trajectory the effector is to travel under 37 s. Joint coordinate vector data $\varphi = [\varphi_1, \varphi_2, \dots, \varphi_6]^T$ are obtained using a sampling period of T_{sp} each 0.005 s. The only (software-imposed) limitation is the degree of freedom itself, namely the possibility of rotation in the $<-\pi, +\pi>$ interval. This problem had to be subsequently addressed since it was causing a discontinuity in the simulation's operation as per the condition (18) and was thus disrupting the continuity of the trajectory on the models in the environments into which data were imported., i.e., Matlab, SolidWorks, but the same would have been the case in other environments, too.

2.2. Modeling in Matlab

The kinematic model in Figure 3 was created in Matlab using the Simscape/Multibody toolbox, the dimensions of which reflect the reality and are given in Table 1. The module diameter or volume need not be of interest when making a kinematic assessment, as it has no effect on kinematic parameters.

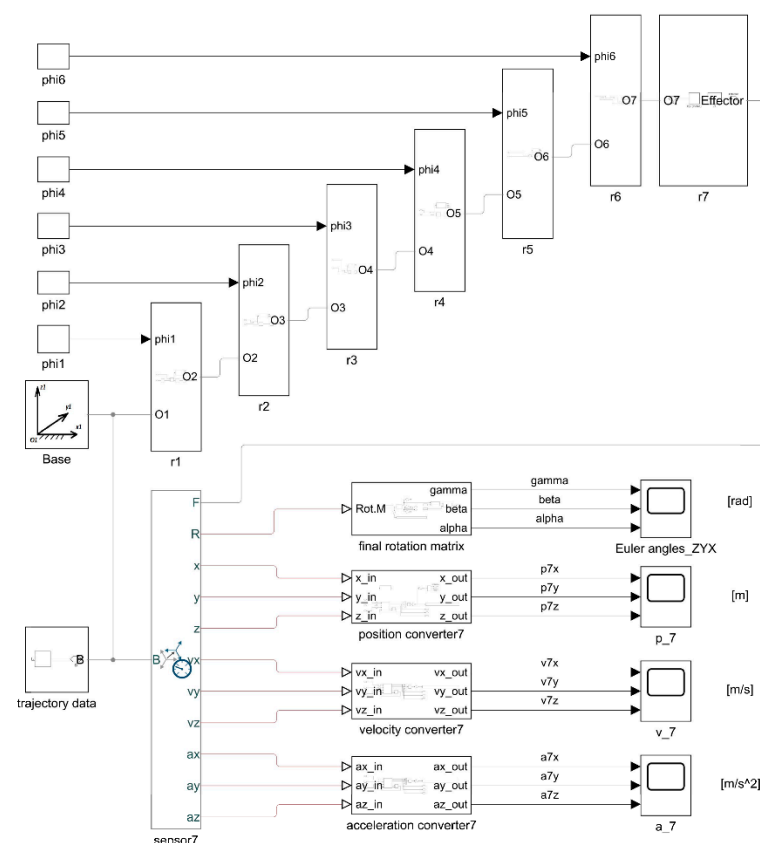


Figure 3. Model created in Matlab-Simulink using the Simscape/Multibody toolbox.

The model is composed of individual blocks that define its parts. In a nutshell, the model consists of a base identical to the “base” of the reference system and the individual blocks representing the r_i vector (a kinematic chain segment). As we can see in Figure 1, the “vector r_1 ” block then consists of the “URM” module and a passive joint part. The modules are connected in places where each joint represents its own Cartesian system of the module, starting in point O_i . In this experiment, the last part of the passive joint, vector r_7 , was considered to be an effector. A measuring device (sensor 7) is connected to the effector r_7 , checking physical quantities (position, velocity, acceleration). Such measuring devices can be mounted on other modules, too. The Euler’s angles’ values are also available in three consecutive rotations around the axes, often marked ZYX. The values apply to the effector and are the result of the final product of the individual spatial transformations

(8), as is usually the case with the open kinematic chain [27]. A model constructed in this way is then visualized in the Mechanics explorer window and appears in the form as seen in Figure 4.

Table 1. Dimensions table of the robotic arm of Figure 2 and also Figure 3, Figure 4, respectively.

Diameters	(mm)	Diameters	(mm)	Diameters	(mm)
r1	243.215	a1	73	b1	128
r2	212.430	a2	42.215	b2	128
r3	212.430	a3	42.215	b3	128
r4	212.430	a4	42.215	b4	128
r5	212.430	a5	42.215	b5	128
r6	212.430	a6	42.215	b6	128
r7	42.215 ¹				

¹ Effector (last part of the passive joint).

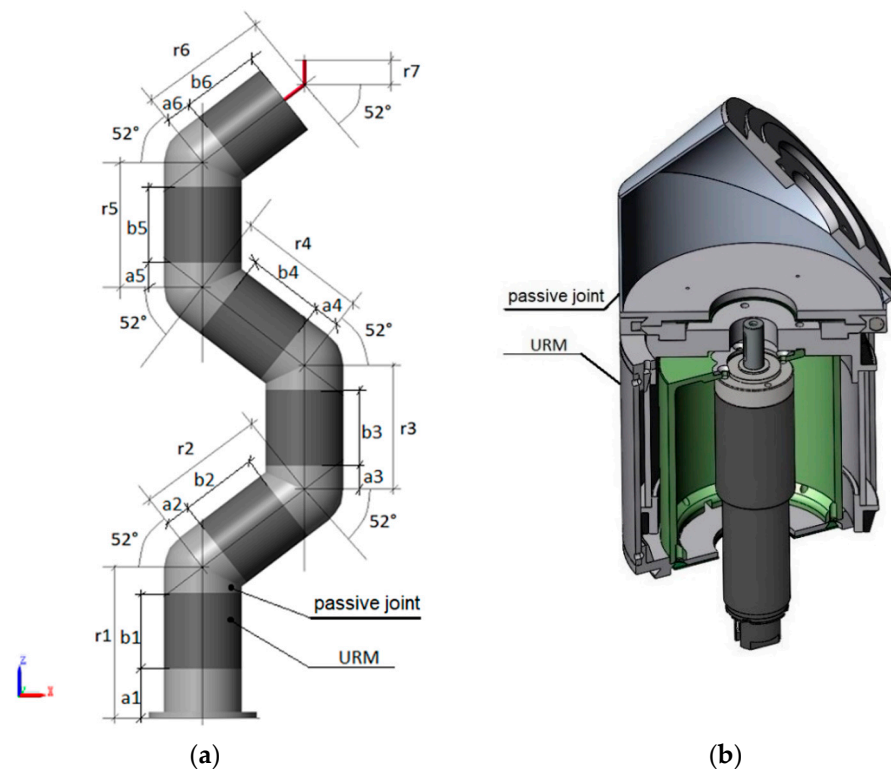


Figure 4. (a) Model visualization with hints of dimensions in Matlab-Simulink using the Simscape/Multibody toolbox. (b) Detail URM with a passive joint.

2.3. The Problem of a Sudden Change in Data Continuity and Its Solution in Matlab

The task was to use the computational core of CoppeliaSim Edu and apply it to the inverse kinematic task, to calculate the joint coordinates of $\varphi = [\varphi_1, \varphi_2, \dots, \varphi_6]^T$ for the entered trajectory. When the joint coordinates data were imported, in some cases, the above-mentioned problem with value repolarization emerged as a result of the software-imposed limitation on the joint rotation. In other words, in the case of a requirement arising from the calculation, the range of the degree of freedom upon exceeding the rotation values in the $\langle -\pi, +\pi \rangle$ interval must be switched to the value with the opposite sign to remain in the defined working interval. The values above the upper limit $+\pi$ will rise again, but this time starting with the lower value of $-\pi$. Moreover, the opposite values below the lower limit $-\pi$ will fall again, this time starting from the upper value of $+\pi$. This sudden change causes deformation or even computational instability of the p_7 trajectory of the effector, see

Figure 5. With time derivations of the position, the characteristics of instantaneous velocity and accelerations are deformed even more.

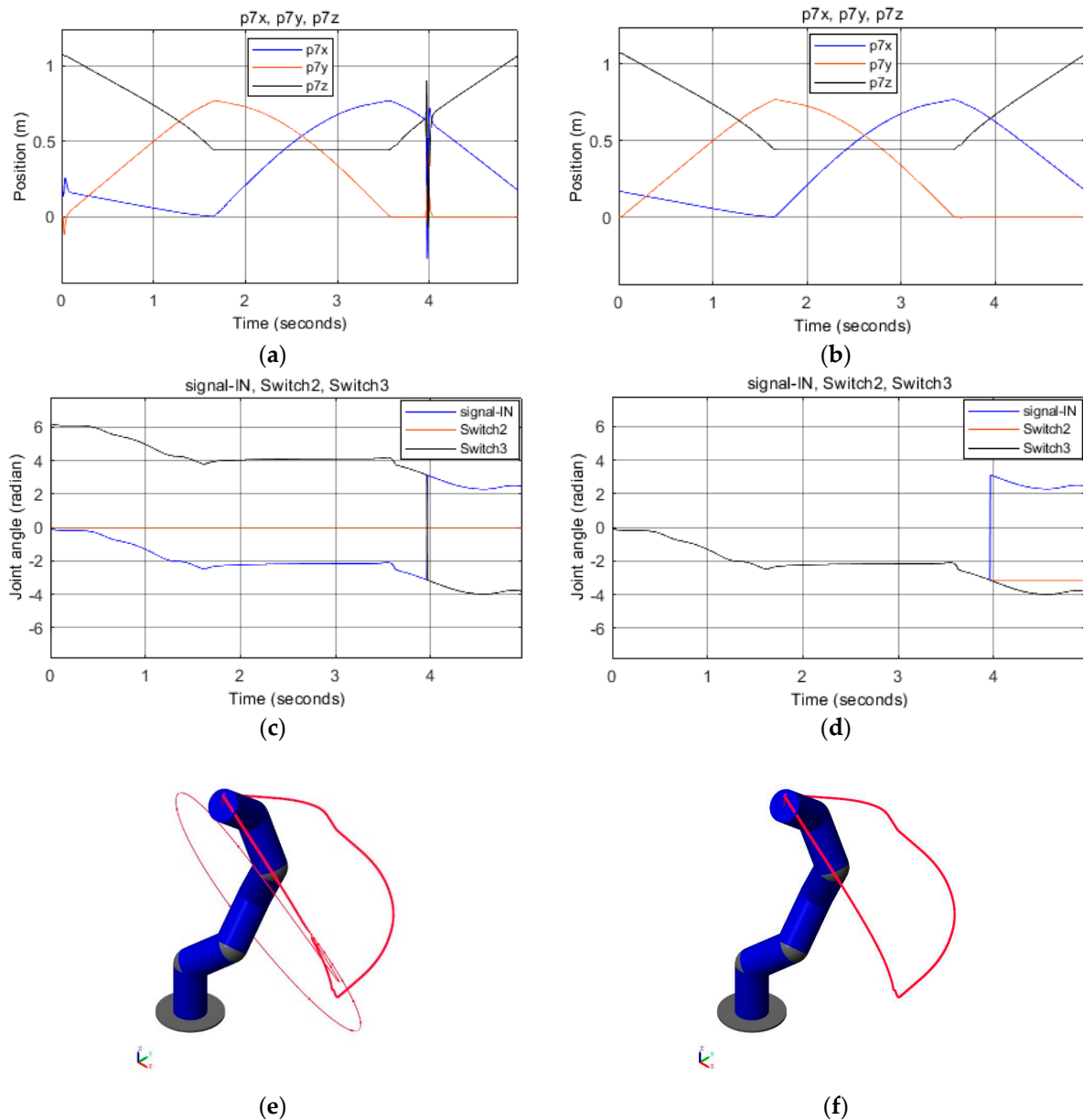


Figure 5. Illustration of the $p_7 = [p_{7x}, p_{7y}, p_{7z}]^T$ position vector (effector) depending on the joint coordinate vector $\varphi = [\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6]^T$ and its effect on the trajectory: (a) before repolarization; (b) after repolarization; the value of the joint coordinate vector depending on time $\varphi = [\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6]^T$; (c) before repolarization; (d) after repolarization; effect on the trajectory; (e) before repolarization; (f) after repolarization.

Since the precision of the trajectory traveled by any manipulator is critical in practical applications, it was necessary to somehow address this problem. Either through a labor-intensive rewriting of the imported data upon each change in the defined trajectory subject to the presence of those repolarizations in the joints or through a suitable solution. One such solution is the so-called repolarize, described below.

2.4. Composing a Repolarize

The search for a suitable and undemanding solution to this situation rested on the following conditions:

- To maintain the original trajectory, the initial values should not be skewed, i.e., if possible, they should not be averaged or approximated;
- There should be no phase shift of the initial values, as is the case with many filters.

For these reasons, we were trying to apply different approaches in the simulation, starting with a change in the initial data $\varphi = [\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6]^T$, which was very labor-intensive. The rate limiter has not met the expectations either [28] in switching the data flow at the moment of reaching the limits of the angles of rotation $\pm\pi$, because the characteristics of the p_7 trajectory were undulated anyway. Rate limiter is a marginalized derivation of an infinitely small difference to a difference of finite size. Or similar approaches based on a sequential omission of the undesired sample, such as in [29]. Or through averaging the values [30,31]. Neither of the approaches mentioned in this case has met our expectations. The best, if not ideal, results were finally achieved by the system working on this principle Figure 6 and explained in Figure 7. Joint coordinate data were measured in the sampling period T_{sp} every 0.005 s.

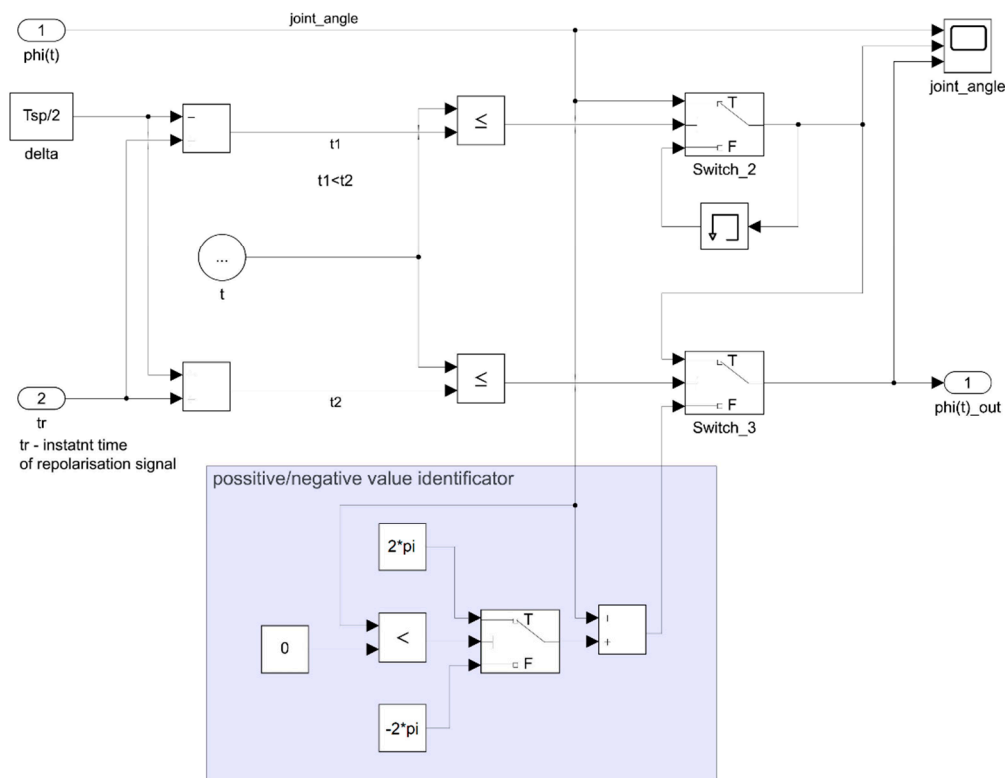


Figure 6. Constructed repolarizer.

When the data were uploaded, it was first necessary to establish the moment of repolarization t_r with the precision of $\delta = \pm 0.0025$ s when the variable $\varphi(t) = 0$, i.e., cuts through the time axis. Under the assumption of reaching the upper or the lower working interval limit $\langle -\pi, +\pi \rangle$ of the joint coordinate. The variable at the output from the repolarize is $\varphi_{out}(t)$. It was necessary to meet the following conditions:

$$t \leq t_1 | \varphi_{out}(t) = \varphi(t) \quad (15)$$

$$t_1 < t_2 \quad (16)$$

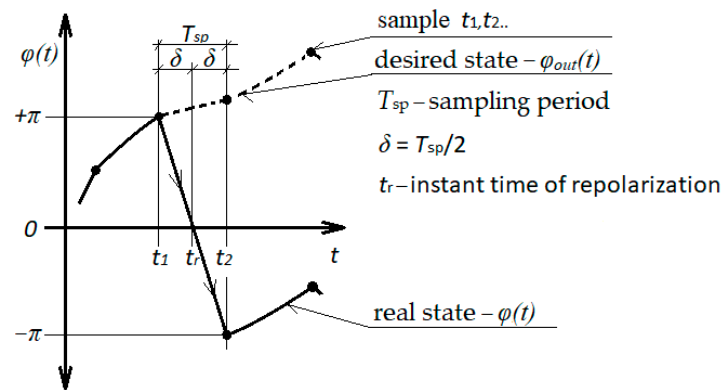


Figure 7. Plotted repolarization of rotation in the joint.

Conditions for identification of the sign and carrying out repolarization over time t_r :

$$t \geq t_2 \left| \varphi_{out}(t) = \begin{cases} 2\pi - \varphi(t), & \varphi(t) < 0 \\ \varphi(t) - 2\pi, & \varphi(t) > 0 \end{cases} \quad (17)$$

For the condition that can be considered discontinuation in the angle variable data $\varphi(t)$, the following applies:

$$\frac{|\varphi(t + T_{sp}) - \varphi(t)|}{T_{sp}} = \frac{2\pi}{T_{sp}} \quad (18)$$

The model was thus “excited” by the values of the joint coordinate vector $\varphi(t) = [\varphi_1(t), \varphi_2(t), \dots, \varphi_n(t)]^T$ obtained from the model created in CoppeliaSim Edu in the period of time T_{sp} . Under the condition of the prescribed maximum permissible deviation $[\Delta_{\max x}, \Delta_{\max y}, \Delta_{\max z}] = +/ - 0.0035$ m between the real trajectory and the one calculated on the basis of inverse kinematics. The data of joint coordinates $\varphi(t)$ correspond to the effectors of the trajectory traveled. The trajectory is made of a system of fixed points $A_s\{x_{As}, y_{As}, z_{As}\}$, where the sample $se < 1; 7383 >$ in Cartesian space Figure 8. Its real shape is shown in Figure 2.

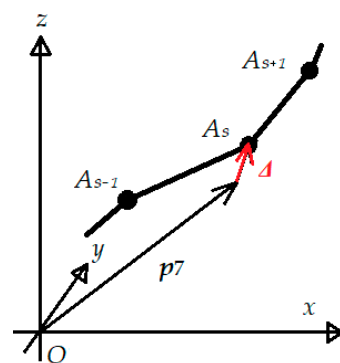


Figure 8. A vector difference Δ between the calculated position of the position vector of the p_7 effector and the position of the A_s points.

The mutual calibration of the models was done for the initial position of the robotic arm effector over time $t = 0$ s, $\varphi(0) = [0, \pi, 0, \pi, 0, \pi]^T$ radians see position in Figure 4, $p_7 = [0.167397, 0, 1.102644]^T$ meters. In order to check the models’ correctness, the vector difference $\Delta = [\Delta_x, \Delta_y, \Delta_z]^T$ between the position of the position vector effector $p_7 = [p_{7x}, p_{7y}, p_{7z}]^T$ and the position of the A_s points, composing the trajectory, shown in the graph of Figure 9 (Matlab), Figure 12 (SolidWorks) over the period of time T_{sp} . This

was done for each vector component. The following applies to Δ vector components, its norm $\|\Delta\|$ and the instantaneous time t of the sample in seconds:

$$\Delta_x = x_{As} - p_{7x}(\varphi) \quad (19)$$

$$\Delta_y = y_{As} - p_{7y}(\varphi) \quad (20)$$

$$\Delta_z = z_{As} - p_{7z}(\varphi) \quad (21)$$

$$t = 0.005(s - 1) | s \in \langle 1; 7383 \rangle \quad (22)$$

$$\|\Delta\| = \sqrt{\Delta_x^2 + \Delta_y^2 + \Delta_z^2} \quad (23)$$

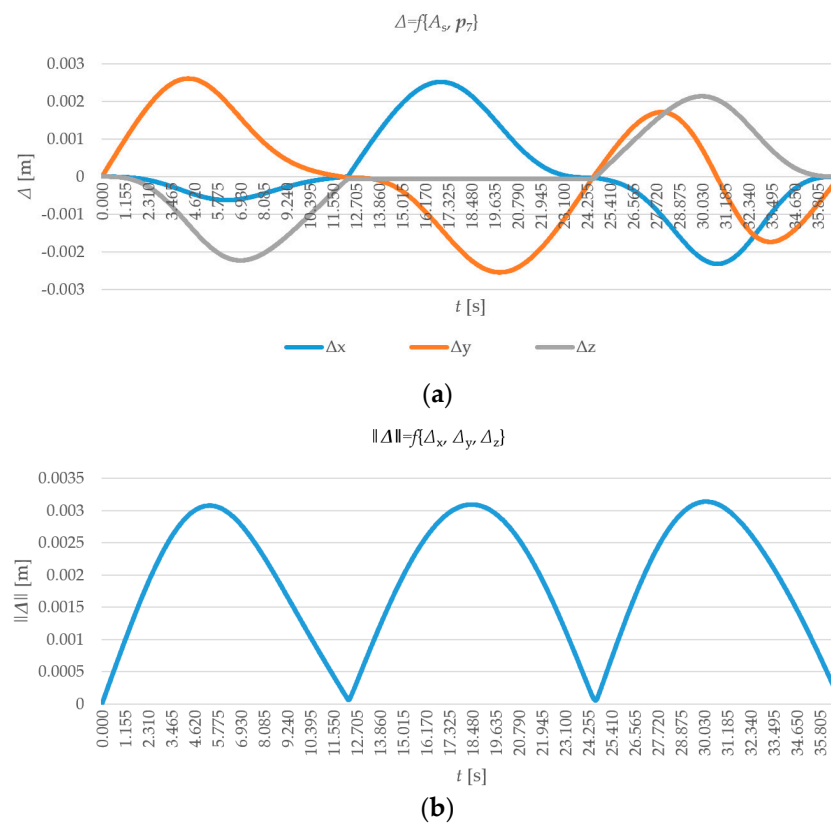


Figure 9. (a) The chart showing the difference Δ_x , Δ_y , Δ_z between the real trajectory (composed of the $A_s\{x_{As}, y_{As}, z_{As}\}$ points) and the one calculated (from the components of the position vector effector p_{7x}, p_{7y}, p_{7z}) in Matlab; (b) norm (size) of the vector $\|\Delta\|$.

As we can see, the CoppeliaSim Edu software complied with the tolerance range requirement ± 0.0035 m for each vector component, for the inverse calculation of the joint coordinate data $\varphi(t)$ with respect to the trajectory consisting of a finite set of A_s points. The characteristics of physical quantities from sensor 7 in the model see Figure 3 (position, velocity, acceleration) are shown in Figure 10. We note that this is a different data set (different trajectory) than the one used in the illustration of Figure 5.

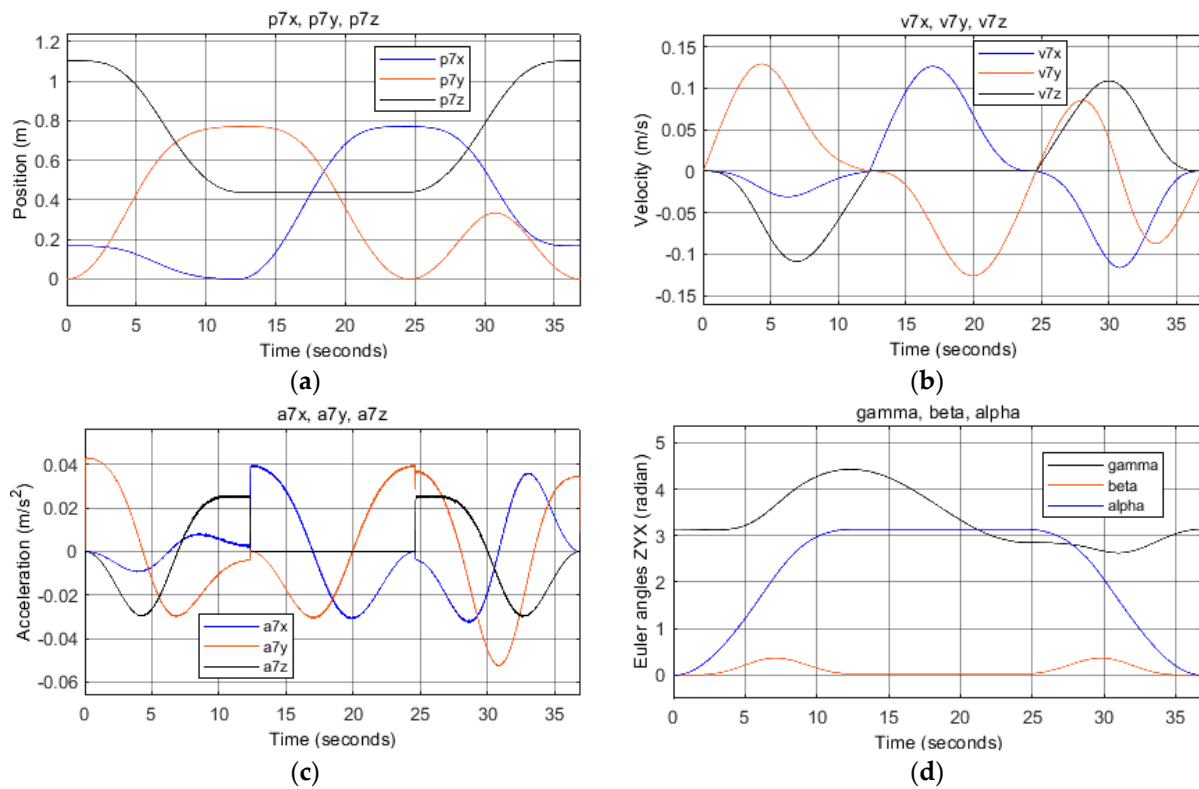


Figure 10. Effector values measured in Matlab: (a) position vector $p_7 = [p_{7x}, p_{7y}, p_{7z}]^T$ depending on the joint coordinate vector $\varphi(t)$; (b) velocity vector $v_7 = [v_{7x}, v_{7y}, v_{7z}]^T$ depending on the joint coordinate vector $\varphi'(t)$; (c) acceleration vector $a_7 = [a_{7x}, a_{7y}, a_{7z}]^T$ depending on the joint coordinate vector $\varphi''(t)$ and $(\varphi'(t))^2$; (d) Euler's angles (γ, β, α) depending on the joint coordinate vector $\varphi(t)$.

2.5. Modeling in SolidWorks

In order to carry out structural changes, various structural compositions of the robotic arm composed of the URM modules are created in SolidWorks are shown in Figure 11. This mechanically detailed model corresponding to the reality was in the same dimensional configuration Table 1, as was used for making a comparison of kinematic indices and also the $\Delta_x, \Delta_y, \Delta_z$ difference between the real and the calculated trajectory.

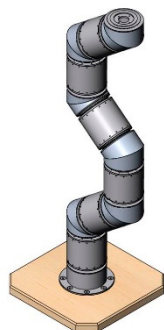


Figure 11. Model created in SolidWorks.

In this case, too, the graph of Figure 12 has confirmed that the requirement for a tolerance range of ± 0.0035 m for each vector component was adhered to by the inverse calculation of the joint coordinate data $\varphi(t)$ with respect to the trajectory composed of a finite set of the A_s points.

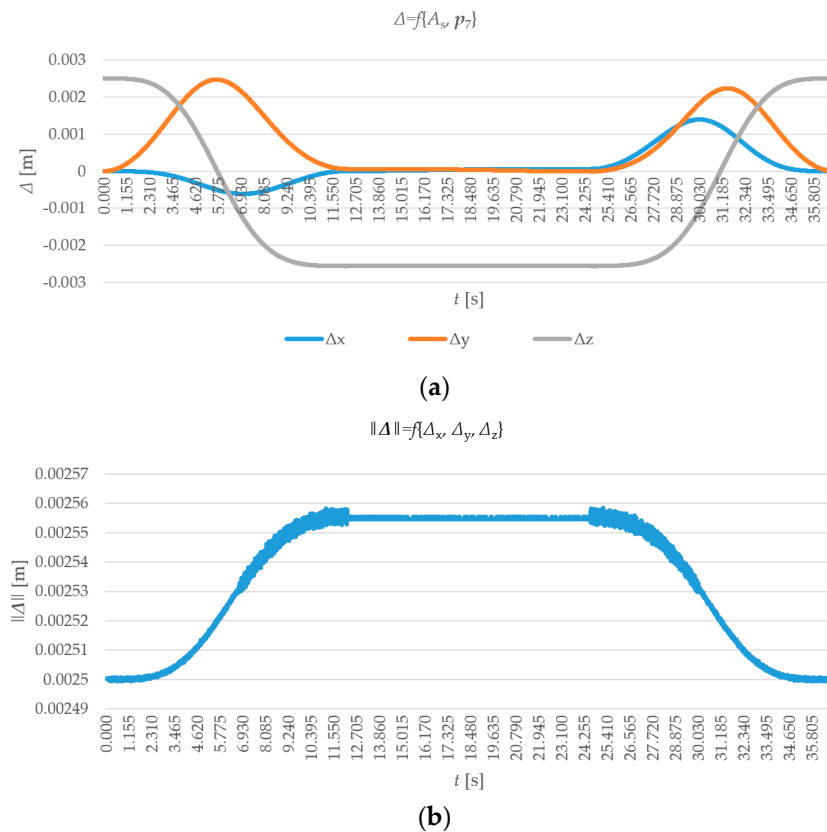


Figure 12. (a) A graph showing the Δ_x , Δ_y , Δ_z difference between the real trajectory (composed of $A_s\{x_{As}, y_{As}, z_{As}\}$ points) and the one calculated (from the components of the position vector effector p_{7x}, p_{7y}, p_{7z}) in SolidWorks; (b) norm (size) of the vector $\|\Delta\|$.

Here, however, certain changes in the shape of the course can be observed, which is logical, as we are dealing with two different software environments and the model structure itself as well. The norm, i.e., the size of the vector of deviation from the original trajectory $\|\Delta\|$ already has a permanent value of 0.0025 m here, and no significant minima can be seen in the three places where the trajectory “breaks” see Figure 2. The characteristics of physical quantities corresponding to the effector in the model see Figure 11 (position, velocity) are illustrated in Figure 13.

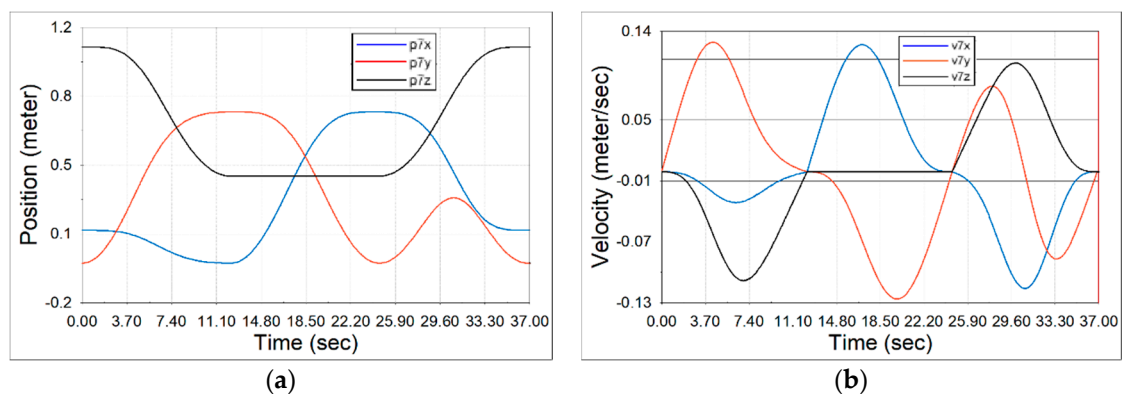


Figure 13. Effector values generated in SolidWorks: (a) position vector $p_7 = [p_{7x}, p_{7y}, p_{7z}]^T$ depending on the joints coordinate vector $\varphi(t)$; (b) velocity vector $v_7 = [v_{7x}, v_{7y}, v_{7z}]^T$ depending on the joints coordinate vector $\varphi'(t)$.

As we can see, the characteristics of position and velocity alike Figure 13a,b, corresponding to the effector in the model of Figure 11, are similar to the characteristics shown in Figure 10a,b.

3. Results

The robotic arm model was created in CoppeliaSim Edu in order to obtain the joints coordinates, corresponding to a description of the spatial trajectory consisting of the fixed points $A_s\{x_{As}, y_{As}, z_{As}\}$ with a permissible deviation of ± 0.0035 m from the original trajectory.

Results obtained in Matlab are shown in Figures 9 and 10. Figure 9 shows the difference $\Delta_x, \Delta_y, \Delta_z$ between the real trajectory (consisting of the $A_s\{x_{As}, y_{As}, z_{As}\}$ points) and the one calculated (from the components of the position vector effector p_{7x}, p_{7y}, p_{7z}). On the other hand, Figure 10 shows the values measured on the effector, namely:

- The position determined by the position vector $p_7 = [p_{7x}, p_{7y}, p_{7z}]^T$ depending on the joint coordinate vector $\varphi(t)$;
- The speed determined by the velocity vector $v_7 = [v_{7x}, v_{7y}, v_{7z}]^T$ depending on the first derivation (time-bound) of the joint coordinate vector $\varphi'(t)$;
- The acceleration determined by the acceleration vector $a_7 = [a_{7x}, a_{7y}, a_{7z}]^T$ depending on the second derivation (time-bound) of the joint coordinate vector $\varphi''(t)$ and $(\varphi'(t))^2$;
- The orientation determined by the Euler's angles according to the $[\gamma, \beta, \alpha]$ option, depending on the joint coordinate vector $\varphi(t)$.

Figure 5 shows the importance of using the repolarize. Otherwise, the joint coordinate data must be edited manually prior to their import, as was the case with SolidWorks.

The results obtained in SolidWorks are shown in Figures 12 and 13. Figure 12 shows the $\Delta_x, \Delta_y, \Delta_z$ difference between the real trajectory (consisting of the $A_s\{x_{As}, y_{As}, z_{As}\}$ points) and the one calculated (from the position vector effector components of p_{7x}, p_{7y}, p_{7z}). On the other hand, Figure 13 shows the values generated on the effector, namely:

- The position determined by the position vector $p_7 = [p_{7x}, p_{7y}, p_{7z}]^T$ depending on the joint coordinate vector $\varphi(t)$;
- The speed determined by the velocity vector $v_7 = [v_{7x}, v_{7y}, v_{7z}]^T$ depending on the first derivation (time-bound) joint coordinate vector $\varphi'(t)$.

4. Discussion

Based on the above results, we can conclude that the currently available software environments offer not only relatively powerful visualization tools but also those enabling various kinds of physical computations. The quality of results depends not only on the sophistication of the software and its preferable use but also on correct parameter specification, on the defined conditions and on the application of the software to its maximum potential. The aim of our work was especially to enable the comparison of data from the position kinematics in Matlab and in SolidWorks, subject to the same initial joint coordinate vector $\varphi(t)$. This has enabled us to check the designed models and to detect eventual structural discrepancies such as counter-rotation of the URM module or an undesirable shift of structural elements.

Matlab made it possible to choose the manner of processing the imported data, to align the sampling step with the CoppeliaSim imported data step and thus ensure that a particular input datum actually corresponded to the given output value over the same time. It was possible to additionally construct a system to accommodate some changes in the input data Figure 6 and thus to make the work involved in input data changes more effective. The difference between the real and the calculated trajectory was quantified using the Δ vector. Figure 9b shows that the vector size $\|\Delta\|$ reaches its maxima at the maximum effector velocities.

In SolidWorks, the difference between the real and the calculated trajectory Δ was already different in nature; see Figure 12. Here, the $\|\Delta\|$ vector size has a permanent value

and changes only slightly; no significant minima are observed here in the places where the trajectory “breaks”, as shown in Figure 2.

Author Contributions: Draft concept and formal analysis, J.S., M.Š., Š.O.; model design and data processing in CoppeliaSimEdu, Z.B.; model design and data processing in Matlab, Š.O.; model design and data processing in SolidWorks, M.Š. and L.H.; supervision, methodology and project administration, J.S., J.D., P.D., T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Slovak Research and Development Agency under Contract no. APVV-18-0413. This paper was prepared with the support of the grant project KEGA 025TUKE-4/2019.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

List of the most important quantities:

p_i	Position vector between the reference coordinate system $S_1\{O_1, x_1, y_1, z_1\}$ and the coordinate system $S_i\{O_i, x_i, y_i, z_i\}$
r_i	Vector quantifying a kinematic chain segment
φ_i	Angle of rotation around the z_i axis of the $S_i\{O_i, x_i, y_i, z_i\}$ system, joint coordinate
φ	Joint coordinate vector
θ_i	Angle of rotation around the y_i axis of the $S_i\{O_i, x_i, y_i, z_i\}$ system
R_{yi}	Rotation matrix for the transformation of the rotational movement around the y_i axis
R_{zi}	Rotation matrix for the transformation of the rotational movement around the z_i axis
$R_{ZYX(i+1)}$	Rotation matrix for calculating Euler’s angles α, β, γ
T	Sampling period
δ	A half period of the sampling period
t_r	Instant time of repolarization
t	Time
Δ	Vector difference between the p_7 position vector effector’s position and the position of the $A_s\{x_{As}, y_{As}, z_{As}\}$ points that make up the trajectory
v_i	Instantaneous velocity vector to the $\{O_i, x_i, y_i, z_i\}$ system
a_i	Instantaneous acceleration vector to the $\{O_i, x_i, y_i, z_i\}$ system

References

1. Svetlík, J.; Štofa, M.; Pituk, M. Prototype development of a unique serial kinematic structure of modular configuration. *MM Sci. J.* **2016**, 994–998. [\[CrossRef\]](#)
2. Svetlík, J. Contribution to Construction of Manufacturing Engineering on Flexible Architecture Basis. Habilitation Thesis, Technical University of Košice, Košice, Slovakia, 14 June 2012.
3. Štofa, M. Experimentálny vývoj rotačných modulov pre stavbu sériových kinematických štruktúr vo výrobní technike. Ph.D. Thesis, Technical University of Košice, Košice, Slovakia, 30 January 2020.
4. Gershenson, J.K.; Prasad, G.J. Modularity in product design for manufacturability. *Int. J. Agile Manuf.* **1997**, 3, 99–110.
5. Benderbal, H.H.; Dahane, M.; Benyoucef, L. Modularity assessment in reconfigurable manufacturing system (RMS) design: An Archived Multi-Objective Simulated Annealing-based approach. *Int. J. Adv. Manuf. Technol.* **2018**, 94, 729–749. [\[CrossRef\]](#)
6. Pérez, R.; Aca, J.; Valverde, A. A modularity framework for concurrent design of reconfigurable machine tools. In Proceedings of the International Conference on Cooperative Design, Visualization and Engineering, Heidelberg, Germany, 19–22 September 2004; Springer: Berlin, Germany, 2004.
7. Svetlík, J. Modularity of Production Systems. In *Machine Tools*; Šooš, L., Marek, J., Eds.; IntechOpen: London, UK, 2020; pp. 1–22.
8. Yim, M.; Duff, D.G.; Roufas, K.D. PolyBot: A modular reconfigurable robot. In Proceedings of the ICRA ’00, IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 1, pp. 514–520.
9. Yim, M.; Roufas, K.; Duff, D.; Zhang, Y.; Eldershaw, C.; Homans, S. Modular reconfigurable robots in space applications. *Auton. Robot.* **2003**, 14, 225–237. [\[CrossRef\]](#)
10. Accacia, G.; Bruzzone, L.; Razzoli, R. A modular robotic system for industrial applications. *Assem. Autom.* **2008**, 28, 151–162. [\[CrossRef\]](#)
11. Xu, W.; Han, L.; Wang, X.; Yuan, H. A wireless reconfigurable modular manipulator and its control system. *Mechatron.* **2021**, 73, 102470. [\[CrossRef\]](#)

12. Pacaiova, H.; Isiarikova, G. Base Principles and Practices for Implementation of Total Productive Maintenance in Automotive Industry. *Qual. Innov. Prosper.* **2019**, *23*, 45–59. [[CrossRef](#)]
13. Nikitin, Y.; Bozek, P.; Peterka, J. Logical–linguistic model of diagnostics of electric drives with sensors support. *Sensors* **2020**, *20*, 4429. [[CrossRef](#)] [[PubMed](#)]
14. Paden, B. Kinematics and Control Robot Manipulators. Ph.D. Thesis, University of California, Berkeley, CA, USA, 1986.
15. Kahan, W. *Lectures on Computational Aspects of Geometry*; Department of Electrical Engineering and Computer Sciences, University of California: Berkeley, CA, USA, 1983; Unpublished.
16. Buss, S.R. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE J. Robot. Autom.* **2004**, *16*, 1–19.
17. Duřeba, I.; Opalka, M. A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators. *Int. J. Appl. Math. Comput. Sci.* **2013**, *23*, 373–382. [[CrossRef](#)]
18. Meredith, M.; Maddock, S. *Real-Time Inverse Kinematics: The Return of the Jacobian*; Technical Report No. CS-04-06; Department of Computer Science, University of Sheffield: Sheffield, UK, 2004.
19. Šoch, M.; Lórencz, R. Solving inverse kinematics—A new approach to the extended Jacobian technique. *Acta Polytech.* **2005**, *45*, 21–26.
20. Virgala, I.; Gmitterko, A.; Surovec, R.; Vacková, M.; Prada, E.; Kenderová, M. Manipulator end-effector position control. *Procedia Eng.* **2012**, *48*, 684–692. [[CrossRef](#)]
21. Virgala, I.; Tomáš, L.; Miková, L. Snake robot locomotion patterns for straight and curved pipe. *J. Mech. Eng.* **2018**, *68*, 91–104. [[CrossRef](#)]
22. Nilsson, R. Inverse Kinematics. Master's Thesis, Luleå University of Technology, Luleå, Sweden, 2009.
23. Wampler, C.W. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Trans. Syst. Man, Cybern.* **1986**, *16*, 93–101. [[CrossRef](#)]
24. Lozhkin, A.; Bozek, P.; Maiorov, K. The method of high accuracy calculation of robot trajectory for the complex curves. *Manag. Syst. Prod. Eng.* **2020**, *28*, 247–252. [[CrossRef](#)]
25. Ondočko, Š.; Stejskal, T.; Svetlík, J.; Hrivniak, L.; Šašala, L. Direct kinematics of modular system. In *Automatizácia a riadenie v teórii a praxi 2020, Proceedings of the 14. ročník konferencie odborníkov z univerzít, vysokých škôl a praxe, Stará Lesná, Slovensko, 5–7 February 2020*; Šeminský, J., Mižáková, J., Šimšík, J., Balog, M., Eds.; Technická Univerzita v Košiciach: Košice, Slovakia, 2020.
26. Ondočko, Š.; Stejskal, T.; Svetlík, J.; Hrivniak, L.; Šašala, L.; Žilinský, A. Position forward kinematics of 6-DOF robotic arm. *Acta Mech. Slovaca.* (under review).
27. Murray, R.; Li, Z.; Shankar, S. *A Mathematical Introduction to Robotic Manipulation*, 1st ed.; CRC Press: Boca Raton, FL, USA, 2017; p. 519. ISBN 9781315136370.
28. Rate Limiter. Available online: https://www.mathworks.com/help/simulink/slref/ratelimiter.html?s_tid=srchtitle (accessed on 29 October 2020).
29. How to Remove Unwanted Short Time Signal Simulink. Available online: <https://stackoverflow.com/questions/52384093/how-to-remove-unwanted-short-time-signal-simulink/52409614> (accessed on 29 October 2020).
30. Filtering and Smoothing Data. Available online: <https://www.mathworks.com/help/curvefit/smoothing-data.html> (accessed on 29 October 2020).
31. Signal Smoothing. Available online: <https://www.mathworks.com/help/signal/ug/signal-smoothing.html> (accessed on 29 October 2020).