COMPARISON SELECTED NUMERICAL METHODS FOR THE CALCULATION INVERSE KINEMATICS OF NON-STANDARD MODULAR ROBOTIC ARM CONSISTING OF UNIQUE ROTATIONAL MODULES

STEFAN ONDOCKO, JOZEF SVETLIK, TOMAS STEJSKAL, MICHAL SASALA, LUKAS HRIVNIAK

Department of Manufacturing Machinery and Robotics, Faculty of Mechanical Engineering, Technical University of Kosice, Kosice, Slovakia

DOI 10.17973/MMSJ.2021_6_2021042

e-mail : stefan.ondocko@tuke.sk

The paper compares the most commonly used numerical methods of solving a set of nonlinear equations, especially in terms of computational speed. The methods are applied to a set of nonlinear equations that describe the forward kinematics of a non-standard robotic arm. This arm is an open-loop kinematics chain, composed of special rotary modules. A non-standard feature of the modules is the unlimited rotation around their own axis. This robotic arm consists of six such modules and, thus, has six degrees of freedom. Computations of this nonlinear set of equations are also called inverse kinematics. All computations were performed in Matlab. The same initial conditions, the computation input parameters, and the same structure of the program was used with each method. By applying the below mentioned known methods to the same set, we sought to choose a suitable computation method for the given type of mechanism.

KEYWORDS

Matlab, inverse kinematics, Jacobian, robotics, unlimited range rotary module, numerical algebra, nonlinear equations

1 INTRODUCTION

The paper compares several numerical methods often applied to solving a set of nonlinear equations used in the kinematics of mechanisms and robots. In this case, they have been applied to solving a set of equations describing the so-called forward kinematics. However, we're already looking for joint coordinates vector depending on the known location of effector. This is the concept of the so-called inverse kinematics problem for the open-loop kinematics chain (eventually serial kinematics chain). Our computations refer to a stationary robotic arm composed of the so-called Unlimited Rotation Module (URM) modules, described in detail in [Svetlik 2013 and 2016, Stofa 2019] and in terms of position kinematics in [Svetlik2 2013, Ondocko 2020]. The solution method is also applicable to other assemblies. The URM module's main attributes are its unlimited rotation around its own axis, availability of integrated energy source as well as embedded control unit and the undeniable advantage of its modularity. Thanks to the modularity, individual modules can be assembled into various types of interesting configurations. Where it comes to inverse kinematics of the position, such joint coordinates are searched for the given open-loop kinematics chain as to suit the position and orientation required of the coordinate system of the end effector. Of course, provided that we know the dimensions of the mechanism at hand [Murray 1994, Grepl 2007, Siciliano 2007]. The inverse task is much more complicated than the direct one, where the position vector is a function of the joint coordinates, because in most cases it is necessary to solve the set of strongly nonlinear algebraic equations. Hence the use of computer support in the environments enabling just that [Coleman 1999]. One of the first pioneers in the formulation of this problem was, for example, [Paden 1986]. In most cases, these sets cannot be solved analytically. Therefore, different kinds of iterative numerical methods are used [Peiper 1968], most often using the Jacobian [Otto 2005, Zhao 2007]. In addition to other methods of verifying the collision states between the robot and its environment, the method by [Hrubos 2016] is also one that can be useful. When computing the inverse function, we often come across unsolvability of the task for the reasons bound to the limitation of the configuration space itself, (in addition to other). Everything depends on the configuration and the physical properties of the mechanism. For example, when the position of the effector is defined outside the given configuration space, no solution exists. There can also be several solutions in the configuration space, as the defined position of the end effector can be achieved in several ways. These, of course, grow in number especially with the growth in the mechanism's degrees of freedom. Then the vector of joint coordinates has a number of elements greater than the degree of freedom (DOF) in a given space or plane.

2 COMPUTATION METHOD OF JOINT COORDINATES FOR A GIVEN TRAJECTORY

The program's algorithm will be the same to ensure objective comparison of the individual numerical iteration methods. The method of data processing will be explained and evident from the logic of the flow chart shown in Fig. 1-part 1, 2. Thus, for each method of numerical computation, the structure of the program itself will be the same. The program was written in Matlab.

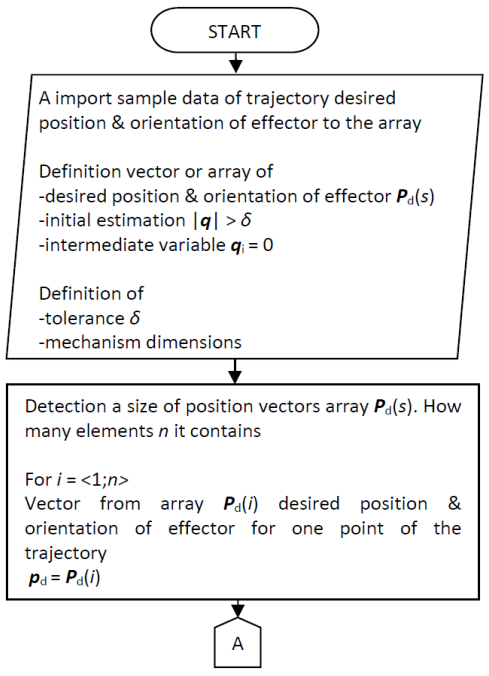

Figure 1 - part 1. Logic flow chart of the program used for computing the inverse kinematics

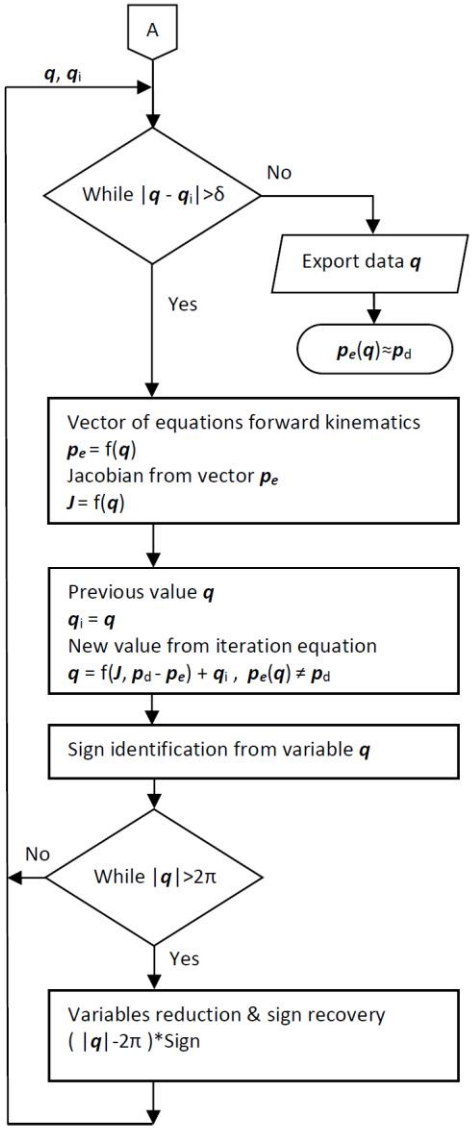

Figure 1 - part 2. Logic flow chart used for computing the inverse kinematics, continued

At the beginning of the process, the input data of the desired position's trajectory are uploaded in the form of points $A_s\{x_{As}, y_{As}, z_{As}\}$, in Cartesian space. The same applies to the data of the desired effector orientation, expressed by the definition of Euler angles α, β, γ. To access this data, it is necessary to store it in a predefined array, in the order in which it was taken from the given trajectory. In this case, there were six two-dimensional vector arrays of data describing the desired position and orientation of the effector in space. Each of the six coordinates depends on the sequence number of the sample s. We can also imagine the sample number to be the time at which the sample was taken from the trajectory. Finally, we arrive at the definition of the vector array of the desired position and orientation, which we designate as $\boldsymbol{P}_d(s)$. An important value in the numerical calculation is the vector of initial estimation of joint coordinates $\boldsymbol{q}$. The following applies to this value

$$|\boldsymbol{q}| > \delta \tag{1}$$

Where δ defines the minimum admissible error in computing the joint coordinates, the so-called tolerance. Further we need a joint coordinates vector of intermediate variable $\boldsymbol{q}_i$ to store the value of the vector from the preceding iteration. The following will hold for this variable

$$\boldsymbol{q_i} = 0 \tag{2}$$

Furthermore, the mechanism's dimensions, such as link lengths or rotation of passive joints [Ondocko 2020], are defined. Another task is to determine the number of elements (samples) n from the array of vectors of the effector's desired position and orientation $\boldsymbol{P}_d(s)$ of the imported data. The vector of the effector's desired position & orientation for a specific point A_s on the trajectory will be that of $\boldsymbol{p}_d$. The iteration process begins with a conditional cycle based on the difference in the values of the initial estimation vectors $\boldsymbol{q}$ and the intermediate variable $\boldsymbol{q}_i$. That is, as long as the following condition is met

$$|\boldsymbol{q} - \boldsymbol{q_i}| > \delta \tag{3}$$

then the joint coordinates are inserted into the set of equations describing the robotic arm's direct kinematics. Thus, in general, the effector vector of position & orientation $\boldsymbol{p}_e$ will consist of six components of p_{e1}, p_{e2}, p_{e3}, p_{e4}, p_{e5}, p_{e6}, which represent the analytical form of the equations. Three of them determine the effector's position in Cartesian space

$$p_{e1} = p_{ex}(\boldsymbol{q}) \tag{4}$$
$$p_{e2} = p_{ey}(\boldsymbol{q}) \tag{5}$$
$$p_{e3} = p_{ez}(\boldsymbol{q}) \tag{6}$$

and three of them its orientation via Euler angles, calculated from the final rotation of the open-loop kinematics chain.

$$p_{e4} = \alpha_e(\boldsymbol{q}) \tag{7}$$
$$p_{e5} = \beta_e(\boldsymbol{q}) \tag{8}$$
$$p_{e6} = \gamma_e(\boldsymbol{q}) \tag{9}$$

Thus, we arrive at the set of nonlinear equations, which, subjected to derivation operations according to time, can be written in the following matrix form

$$\frac{d}{dt}\begin{bmatrix} p_{e1} \\ p_{e2} \\ p_{e3} \\ p_{e4} \\ p_{e5} \\ p_{e6} \end{bmatrix} = \boldsymbol{J}\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} \tag{10}$$

where $\boldsymbol{J}$ is the Jacobian matrix (the so-called Jacobian), which represents the change in the vector function $\boldsymbol{p}_e$ according to the vector $\boldsymbol{q}$. This was computed symbolically from the $\boldsymbol{p}_e$ using Matlab to save computation time during the program run. This brings us to the iteration equation itself, which is obtained from equation (10) by converting the total differential to the iteration difference

$$\begin{bmatrix} p_{d1}-p_{e1} \\ p_{d2}-p_{e2} \\ p_{d3}-p_{e3} \\ p_{d4}-p_{e4} \\ p_{d5}-p_{e5} \\ p_{d6}-p_{e6} \end{bmatrix} = \boldsymbol{J}\begin{bmatrix} q_1-q_{i1} \\ q_2-q_{i2} \\ q_3-q_{i3} \\ q_4-q_{i4} \\ q_5-q_{i5} \\ q_6-q_{i6} \end{bmatrix} \tag{11}$$

Equation (11) is the basis for other known iterative methods of inverse kinematics computation, to be presented and tested on the given mechanism in Chapter 3. An interesting overview of these and other methods can be found, for example, in [Buss 2004, Aristidou 2009, Grepl 2007].

The program further identifies the sign for the $\boldsymbol{q}$ variable from the following relation

$$\frac{\boldsymbol{q}}{|\boldsymbol{q}|} = sign \tag{12}$$

And the next decision subject to the condition below

$$|\boldsymbol{q}| > 2\pi \tag{13}$$

on the need to reduce the joint variable vector value $\boldsymbol{q}$. The need for reduction is based on the value of the initial estimate of $\boldsymbol{q}$ and also on the property of the singular states. A large number of iterations takes place then and the joint coordinate thus acquires very large values. The value preserving the information of the URM module's real angle with unlimited rotation can be reduced by a decrement of 2π. This lasts until condition (13) ceases to apply and where it comes to the value of $\boldsymbol{q}$, we turn the direction of the modules' rotation according to the sign as follows.

$$\boldsymbol{q} = |\boldsymbol{q}| sign \tag{14}$$

New values from the previous iteration process re-enter the process. Based on condition (3), they either re-enter the next iteration or are written to the array of joint variables as a result. The difference between $\boldsymbol{p}_d$, $\boldsymbol{p}_e(\boldsymbol{q})$ values is described via relation

$$\boldsymbol{\Delta} = \boldsymbol{p}_d - \boldsymbol{p}_e(\boldsymbol{q}) \tag{15}$$

Is the vector difference $\boldsymbol{\Delta}$ between the computed position of the position vector effector $\boldsymbol{p}_e(\boldsymbol{q})$ and the desired position determined by the $\boldsymbol{p}_d$ vector of the A_s points on the trajectory. See the next Fig. 2.

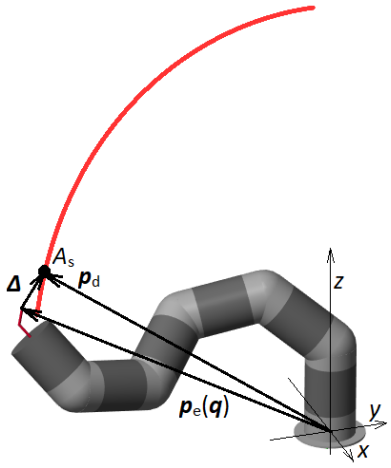

Figure 2. The vector difference $\boldsymbol{\Delta}$ between the computed position of the vector effector $\boldsymbol{p}_e(\boldsymbol{q})$ and the desired position determined by the $\boldsymbol{p}_d$ vector of the A_s points forming the trajectory. Modeled in Matlab.

2.1 Inverse Jacobian

One of the basic iterative methods for solving a set of nonlinear equations is the method using the Jacobian matrix inversion. We obtain it from the relation (11) by inversion of the Jacobian matrix. Hence the name. The method was used for this type of mechanism also when the calculation results of the set of nonlinear equations were compared with the standard Matlab function "fsolve" (designed by MathWorks) as described in [Ondocko 2021].

$$\begin{bmatrix} q_1 - q_{i1} \\ q_2 - q_{i2} \\ q_3 - q_{i3} \\ q_4 - q_{i4} \\ q_5 - q_{i5} \\ q_6 - q_{i6} \end{bmatrix} = \boldsymbol{J}^{-1} \begin{bmatrix} p_{d1} - p_{e1} \\ p_{d2} - p_{e2} \\ p_{d3} - p_{e3} \\ p_{d4} - p_{e4} \\ p_{d5} - p_{e5} \\ p_{d6} - p_{e6} \end{bmatrix} \tag{16}$$

By expressing the dependent variable of the $\boldsymbol{q}$ vector, we can express the relation (16) in the vector form as follows

$$\boldsymbol{q} = \boldsymbol{J}^{-1}(\boldsymbol{p}_d - \boldsymbol{p}_e) + \boldsymbol{q}_i \tag{17}$$

The condition for using this method is that we work with a set of equations, the number of which is identical to the number of the unknowns. Thus, the $\boldsymbol{J}$ is square matrix. In other words, the mechanism has exactly 6 ° of freedom in space. The results of this method on our mechanism are shown in Fig. 3, 4.

2.2 Pseudo-inverse method

The second, very similar iterative method for solving a set of nonlinear equations is the pseudoinversion method. Known as the Moore-Penrose pseudoinversion, it is also suitable for sets of equations when the number of the unknowns is greater than the number of the equations. Such situation occurs in the case of the so-called redundant manipulators that have an excessive number of degrees of freedom. The iterative relationship of the dependent $\boldsymbol{q}$ vector variable in the vector form will be as follows

$$\boldsymbol{q} = \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{J}\,\boldsymbol{J}^{\mathrm{T}})^{-1}(\boldsymbol{p}_d - \boldsymbol{p}_e) + \boldsymbol{q}_i \tag{18}$$

This method is even more often used in robotics than the method of Jacobian inversion precisely because of the versatility of application even to redundant manipulators. The results of pseudoinverse method on our mechanism are shown in Fig. 5, 6.

2.3 Damped Least Squares (DLS) method

Another method applied to solving the set of equations for the inverse kinematics of our arm was the method called damped least squares (DLS). Or, especially in solving sets of nonlinear equations, known as Levenberg-Marquardt method. The iterative relationship used for the dependent variable $\boldsymbol{q}$ in the vector form is

$$\boldsymbol{q} = \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{J}\,\boldsymbol{J}^{\mathrm{T}} + \lambda^2\boldsymbol{I})^{-1}(\boldsymbol{p}_d - \boldsymbol{p}_e) + \boldsymbol{q}_i \tag{19}$$

It is clear from the relationship that if the so-called damping coefficient λ is very small, almost close to zero, it is essentially a pseudoinversion. Increasing the damping constant leads to computation inaccuracies but speeds up the iteration process. The damping constant can, to some extent, influence the computation behavior around singularities. The method is also applicable to redundant manipulators and the results of DLS method on our mechanism are shown in Fig. 7, 8.

2.4 Transpose method

Another method tested to solve the set of equations for the inverse kinematics of our arm was the method the essence of which is to replace the inversion of Jacobian $\boldsymbol{J}$ by transposition, an example [Hock 2018]. Thus, for the iterative relationship used for the dependent variable $\boldsymbol{q}$, the following holds

$$\boldsymbol{q} = \alpha\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{p}_d - \boldsymbol{p}_e) + \boldsymbol{q}_i \tag{20}$$

Where the optimal determination of the α coefficient is

$$\alpha = \frac{(\boldsymbol{p}_d - \boldsymbol{p}_e)^{\mathrm{T}}\boldsymbol{J}\,\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{p}_d - \boldsymbol{p}_e)}{\|\boldsymbol{J}\,\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{p}_d - \boldsymbol{p}_e)\|^2} \tag{21}$$

The results of this method on our mechanism are shown in Fig. 9, 10. Compared to the above-mentioned methods, this method was relatively slow. More detailed comparison of all approaches via simulation results is shown in Tab. 1 in the following chapter.

3 EXPERIMENTAL COMPARISON OF INDIVIDUAL METHODS

As said before, the individual iteration methods were applied to the same program algorithm and their results are shown in Tab. 1. Number of samples from the original trajectory n=2464 A_s points; i5-7200 processor, 2.5GHz, 16GB RAM, Disc SSD PG SX8200 Pro, Matlab 2020b, x64.

Table 1. Data measured under individual methods. Processor i5-7200, 2,5GHz, 16GB RAM, Disk SSD PG SX8200 Pro, Matlab 2020b, x64. Number of samples from the original trajectory n=2464

Method	Calculation duration T_{CPU} ± 10% [second]	Max.absolute value of vector $\|\|\Delta\|\|$ [meter]	Vector Q accuracy δ [radian]
Inverse Jacobian	28.9	0.0033	10^{-5}
Pseudoinverse	38.2	0.0033	10^{-5}
DLS (λ=0.02744)	47.1	0.0429	10^{-5}
Transpose	1137.7	0.0530	10^{-5}

Joint coordinate values $\boldsymbol{q}=[q_1, q_2, q_3, q_4, q_5, q_6]^T$ are plotted in graphs of the corresponding computation method. This happens depending on the $\boldsymbol{p}_e(\boldsymbol{q})$ effector's position on the trajectory formed by the A_s points. In addition, the chart shows the error dependence quantified by the vector norm $\|\boldsymbol{\Delta}\|$ dependent on the position of the trajectory formed by the A_s points.

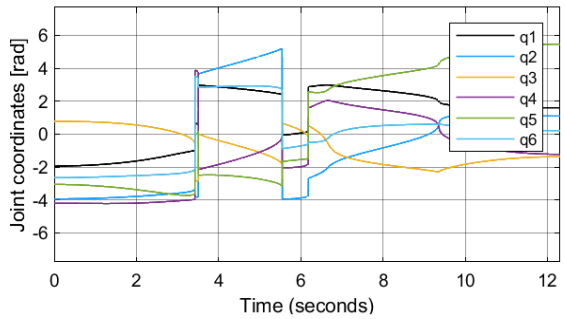

Figure 3. Vector value of joint coordinates $\boldsymbol{q}=[q_1, q_2, q_3, q_4, q_5, q_6]^T$ dependent on the $\boldsymbol{p}_e(\boldsymbol{q})$ effector's position on the trajectory formed by the A_s points. Computed by the inverse Jacobian method.

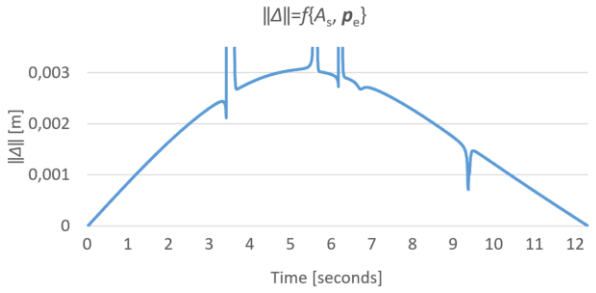

Figure 4. Graph showing the $\|\boldsymbol{\Delta}\|$ vector norm dependent on the position on the trajectory formed by the A_s points. Computed by the inverse Jacobian method.

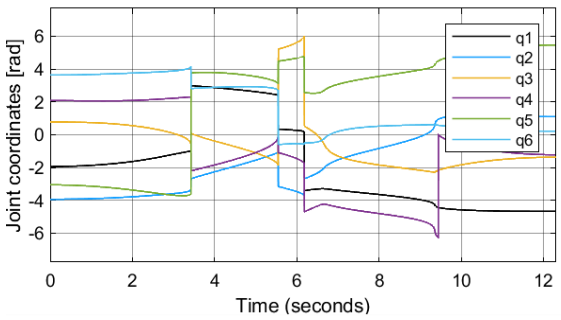

Figure 5. Vector value of joint coordinates $\boldsymbol{q}=[q_1, q_2, q_3, q_4, q_5, q_6]^T$ dependent on the $\boldsymbol{p}_e(\boldsymbol{q})$ effector position on the trajectory formed by the A_s points. Computed by the Pseudoinversion method

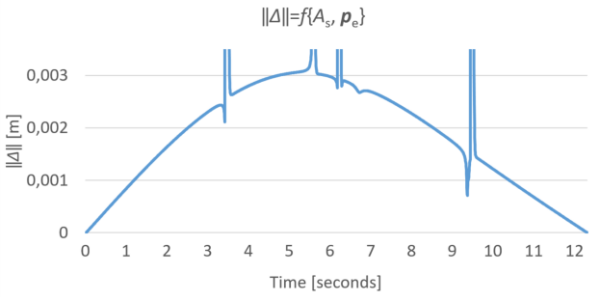

Figure 6. Graph showing the $\|\boldsymbol{\Delta}\|$ vector norm dependent on the position of the trajectory formed by the A_s points. Computed by the Pseudoinversion method

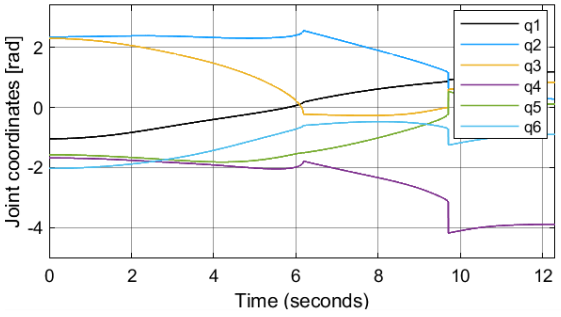

Figure 7. Vector value of joint coordinates $\boldsymbol{q}=[q_1, q_2, q_3, q_4, q_5, q_6]^T$ dependent on the $\boldsymbol{p}_e(\boldsymbol{q})$ effector position on the trajectory formed by the A_s points. Computed by the DLS method

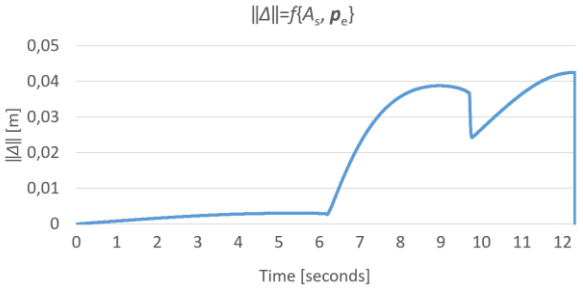

Figure 8. Graph showing the $\|\boldsymbol{\Delta}\|$ vector norm dependent on the position of the trajectory formed by the A_s points. Computed by the DLS method

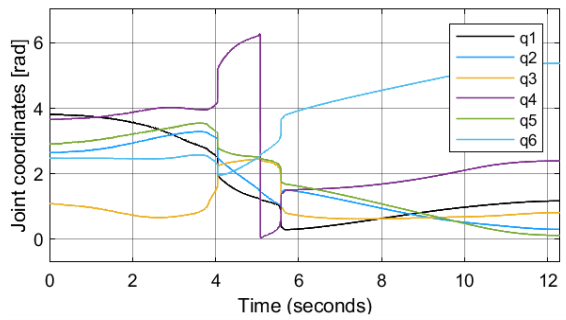

Figure 9. Vector value of joint coordinates $\boldsymbol{q}=[q_1, q_2, q_3, q_4, q_5, q_6]^T$ dependent on the $\boldsymbol{p}_e(\boldsymbol{q})$ effector position on the trajectory formed by the A_s points. Computed by the Transpose method

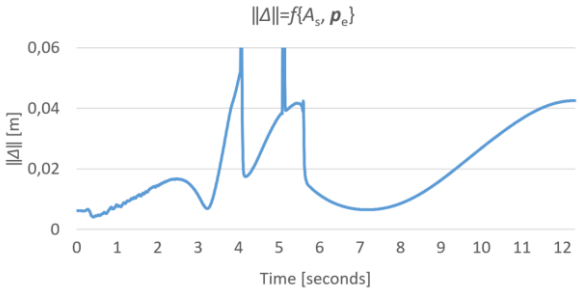

Figure 10. Graph showing the $\|\boldsymbol{\Delta}\|$ vector norm dependent on the position of the trajectory formed by the A_s points. Computed by the Transpose method

4 CONCLUSIONS

Based on the comparison of the methods in terms of the computation speed for this type of mechanism and the computation algorithm shown in Fig. 1 - part 1, 2, we can state that the fastest method was the Jacobian inversion method. However, if we were to create a structure with a different number of degrees of freedom from the modules, we would have to turn to a different method. The most suitable substitute for Jacobian inversion is pseudoinversion. In the DLS method (Figs. 7 and 8) we can notice the ability to avoid singularities (visible in the methods in Figs. 4 and 6) in the effector position corresponding to the sample at 3.5 and 5.5 seconds. Of course, this is subject to a suitable choice of the damping constant λ. In this case, it was chosen experimentally with regard to the highest possible computation speed and, at the same time, the smallest possible deviation from the original $\|\boldsymbol{\Delta}\|$ trajectory. As mentioned above, it is possible to "pull" the system out of the singularity to some extend with this method, at the expense of an increasing $\|\boldsymbol{\Delta}\|$ error. The cost of this "smooth" course is an exponential increase in the error from the singularity in the time of approx. 6.2 seconds almost 12 times - see Fig. 8. The error for the transpose method was relatively large. Due to this method, we were forced to increase the minimum allowable tolerance of joint coordinates vector from the value $\delta = 10^{-6}$ to the value $\delta = 10^{-5}$, for all methods, for the sake of objectivity. Otherwise, the computation time would be unbearable (it would take hours).

ACKNOWLEDGMENTS

This paper has been prepared with the support of the following grant projects:
APVV-18-0413 Modular architecture of the manufacturing technology structural elements.
KEGA 025TUKE-4/2019 Integrated teaching laboratory of virtual prototyping and experimental verification of the machine tools accuracy

REFERENCES

[Aristidou 2009] Aristidou, A., Lasenby, J. Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. Technical Report. Cambridge University, Engineering Department, 2009.

[Buss 2004] Buss, S.R. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. San Diego: University of California, Department of Mathematics, 2009.

[Coleman 1999] Coleman, T., Branch, M.A., Grace, A. Optimization toolbox. For Use with MATLAB. User's Guide for Matlab 5, Version 2, Release II, 1999.

[Grepl 2007] Grepl, R. The kinematics and dynamics of mechatronic systems. Brno University of Technology: Brno, 2007. ISBN 978-80-214-3530-8. (in Czech).

[Hock 2018] Hock, O., Sedo, J. Inverse kinematics using transposition method for robotic arm. In: 2018 ELEKTRO. IEEE, 2018, pp. 1-5.

[Hrubos 2016] Hrubos, M., et al. Searching for collisions between mobile robot and environment. International journal of advanced robotic systems, 2016, Vol. 13, DOI:10.1177/1729881416667500.

[Murray 1994] Murray, Richard M., et al. A mathematical introduction to robotic manipulation. Boca Raton: CRC press, 1994. ISBN 0-8493-7981-4.

[Ondocko 2020] Ondocko, S., et al. Position forward kinematics of 6-DOF robotic arm. Acta Mechanica Slovaca 2020, Vol. 24, Issue 2, pp. 30-36. DOI: 10.21496/ams.2020.01.

[Ondocko et al. 2020] Ondocko, S., et al. Inverse Kinematics Data Adaptation to Non-Standard Modular Robotic Arm Consisting of Unique Rotational Modules. Applied Sciences 2021, Vol. 11, Issue 3, pp. 1-15. DOI: 10.3390/app11031203.

[Ondocko 2021] Ondocko, S., Stejskal, T., Svetlik, J., Hrivniak, L., Sasala, M. Processing of Inversion Kinematics Data in Matlab for Robotic Arm Composed of Urm Modules. In: Proceedings of the 18th International Scientific Conference of Doctoral Students of Engineering Faculties of Technical Universities and Colleges, Novus Scientia 2021, Kosice, 28.1.2021, pp. 202-206, ISBN 978-80-553-3798-2. (in Slovak)

[Otto 2005] Otto, S., Denier, J.P. An introduction to programming and numerical methods in MATLAB. Springer Science & Business Media. 2005. ISBN 1852339195.

[Paden 1986] Paden, B. Kinematics and Control Robot Manipulators. PhD thesis. Berkeley: University of California, Department of Electrical Engineering and Computer Sciences, 1986.

[Peiper 1968] Peiper, D.L. The kinematics of manipulators under computer control. Stanford University CA, Department of Computer Science, 1968.

[Siciliano 2007] Siciliano, B., Khatib, O. (Eds.). Springer handbook of robotics (2nd Edition). Springer, 2016. ISBN 978-3-319-32550-7.

[Stofa 2019] Stofa, M. Experimental development of rotary modules for the construction of serial kinematic

structures in manufacturing technology. Kosice: Technical university, Dissertation thesis, 2019.

[Svetlik 2013] Svetlik, J., Demec, P., Semjon, J., Rotational kinetic module with unlimited angle of rotation. Robotics in theory and practise, Book Series: Applied Mechanics and Materials, 2013, Vol. 282, pp. 175-181. DOI: 10.4028/www.scientific.net/AMM.282.175.

[Svetlik2 2013] Svetlik, J., Demec, P. Methods of Identifying the Workspace of Modular Serial Kinematic Structures. Book Series: Applied Mechanics and Materials, 2013, Vol. 309, pp. 75-79. DOI: 10.4028/www.scientific.net/AMM.309.75.

[Svetlik 2016] Svetlik, J., Stofa, M., Pituk, M. Prototype development of a unique serial kinematic structure of modular configuration. MM Science Journal, 2016, pp. 994-998.

[Zhao 2007] Zhao, Y., Huang, T., Yang, Z. A successive approximation algorithm for the inverse position analysis of the general serial manipulators. The International Journal of Advanced Manufacturing Technology, 2007, Vol. 31, No. 9-10, pp. 1021-1027. DOI: 10.1007/s00170-005-0271.

CONTACTS:

Stefan Ondocko, Ing.
Technical University of Kosice,
Faculty of Mechanical Engineering,
Department of Manufacturing Machinery and Robotics,
Letna 9, Kosice, 042 00, Slovakia
Telephone: +421 55 602 3238,
E-mail: stefan.ondocko@tuke.sk