Műszaki Tudományos Közlemények vol. 9. (2018) 71–74.
https://doi.org/10.33894/mtk-2018.09.13
Hungarian: https://doi.org/10.33895/mtk-2018.09.13,
http://eda.eme.ro/handle/10598/30295

SOLUTION OF THE INVERSE KINEMATIC TASK OF A ROBOT-ARM BASED ON A „QUASI-DIFFERENTIAL" FIXED-POINT TRANSFORMATION METHOD

Tamás FAITLI[1], József TAR[2]

[1] *Óbuda University, Donát Bánki Faculty of Mechanical and Safety Engineering, Budapest, Hungary, faitlit@gmail.com*

[2] *Óbuda University, Antal Bejczy Center for Intelligent Robotics, Budapest, Hungary, tar.jozsef@nik.uni-obuda.hu*

Abstract

While the forward kinematic task of robots can be solved easily through homogenous transformation matrices, the inverse kinematic task leads to difficulties as the construction of the system becomes more complex. In this paper, a solution has been worked out for a three Degree-of-Freedom (DOF) robot-arm based on recent research, by the use of a novel, fixed-point transformation based technique.

Keywords: *robot-arm, inverse kinematics, fixed-point theorem, „quasi-differential" solution.*

1. Introduction

The inverse kinematic task of an open kinematic structure, that is, determining the required joint coordinates for a desired position of the end-effector, is usually a difficult task. The desired location of the end-effector can be achieved in infinitely many configurations („poses") for a redundant, multiple DOF system. The traditional method of solving this task usually consists of a matrix inversion operation, which cannot be determined when the robot is at its kinematic singularities. During motion, this can cause rapid „yank" types of movement, which might damage itself or its environment. In [1], a new method has been worked out, which transforms the problem into a fixed-point task, one that does not include the matrix inversion operation. This way, we can avoid the usual difficulties in the kinematic singularities, and nearby in the so-called ill-conditioned areas.

1.1. In short about the „quasi-differential" fixed-point transformation based method

Let us consider an array $q \in \mathbb{R}^n, n \in \mathbb{N}$, which contains the joint coordinates of an n DOF open kinematic chain, and an array $x \in \mathbb{R}^m, m \in \mathbb{N}$ which contains the Cartesian workshop-coordinates of certain points of the robot-arm. Furthermore, let us take the parameter $s \in [s_i, s_f] \subset \mathbb{R}$, which might be the time itself, or a function of the time. This way, the *x(s)* function describes the nominal motion.

Let us suppose, that a nonlinear, differential real function $f: \mathbb{R}^n \to \mathbb{R}^n$ is given as *x=f(q)*, where the task is to find q based on a known x value. This task was transformed in the following way in [2]. Let us take a differentiable function $F(\xi): \mathbb{R} \to \mathbb{R}$, which has an "attractive fixed-point" $F(\xi_\star) = \xi_\star$. We can create a sequence using *F(ξ)* and the repeating signals:

$$q_{s,i+1} = \left[F\left(A \parallel f\left(q_{s,i}\right) - x \parallel + \xi_\star\right) - \xi_\star\right] \frac{f\left(q_{s,i}\right) - x}{\parallel f\left(q_{s,i}\right) - x \parallel} + q_{s,i} \qquad (1)$$

where the norm is calculated based on the Frobenius formula, and $A \in \mathbb{R}$ is a constant parameter. In case of an iteration k, when $q_{s,k} = q_\star$ satisfies the condition $x = f(q_\star)$, the equation *(1)* will lead to $q_{s,k+1} = q_{s,k}$, that is, $q_\star$ will be the solution and a fixed-point as well. As long as we can guarantee that *(1)* is convergent, we can insert a

desired value *x(s)* into the algorithm, which will get close to the fixed-point of the function after several iterations. That is, $q_{s,i}$ will go to the solution of the problem, and we can identify the required joint coordinates q_s.

The constant parameter *A* is important in order to guarantee the convergence, which is usually a small, signed value. It is also important to clarify that the method expects that in the following time instant s+1 the coordinates $q_{s,i}$ are nearby the fixed-point, which can be assumed from the fact that we know the initial state and position of the robot, and from that point it is moving step by step.

In addition, the method suggests using an approximate Jacobian matrix that is conducive to the convergence, and at the same time avoids further problems when the configuration of the robot-arm leads to a Jacobian matrix which is not quadratic. The suggestion is using a modified $\tilde{x} \equiv J^T(q)x = J^T f(q)$ instead of the array *x*.

2. Implementation

The realization of the method and the examination of its potential were achieved through simulations due to its availability and its simplicity.

2.1. System under investigation

The model of the robot-arm can be seen in **Figure 1.** The more-detailed description of forward kinematics and its derivation can be found in **[3]**.

As long as the aim is the trajectory tracking of the end effector, it is enough to know its Cartesian-coordinates in order to implement the method. We can easily calculate these through forward kinematics. (In order to simplify the Jacobian matrix and the description of the Cartesian-coordinates, let us introduce the following notations. Let $c_i \equiv \cos(q_i)$, and $c_{ij} \equiv \cos(q_i + q_j)$, and in similar manner let us introduce their s_i,s_{ij} sinus variations.)

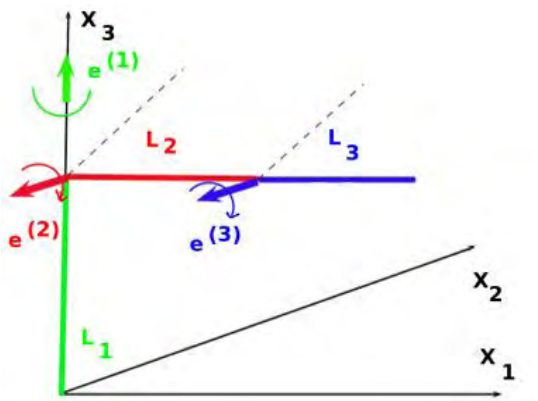

Figure 1. *The kinematic model of the robot-arm under investigation in the „default position”*

$$\begin{pmatrix} x_{3v} \\ y_{3v} \\ z_{3v} \end{pmatrix} = \begin{pmatrix} L_3c_1c_{23} + c_1c_2L_2 \\ L_3s_1c_{23} + s_1c_2L_2 \\ -s_{23}L_3 - s_2L_2 + L_1 \end{pmatrix} \quad (2)$$

By taking the corresponding partial derivatives in (2) we can obtain the Jacobian matrix of the robot, that is

$$J = \begin{pmatrix} -L_3c_{23}s_1 - L_2c_2s_1 & -L_3c_1s_{23} - L_2c_1s_2 & -L_3c_1s_{23} \\ L_3c_1c_{23} + c_1c_2L_2 & -L_3s_1s_{23} - s_1s_2L_2 & -L_3s_1s_{23} \\ 0 & -L_3c_{23} - L_2c_2 & -L_3c_{23} \end{pmatrix} \quad (3)$$

2.2. Simulation

The simulation was processed by the help of a self-developed „Julia” script called *fmtu_simulation.jl*.

For function *F(ξ)* in equation *(1)* we used *F(ξ) = ξ/2 + D*. The other parameters required by the simulation can be seen in **Table 1.**

The nominal trajectories were generated by a simple sinusoidal function that is the robot-arm is moving around continuously in a “scan” type of movement. First, there is a loop when we calculate and save into the memory the generated nominal trajectory. Then we can simply insert that as the desired trajectory into the algorithm introduced above. We allow the algorithm to calculate the required joint coordinates on its own. Finally, the program plots all the calculated data.

Table 1. *The parameters of the simulation*

Length of 1st link L_1 [m]	1.5
Length of 2nd link L_2 [m]	0.8
Length of 3rd link L_3 [m]	0.5
A	-1.0
D	0.3

2.3. Results

The results of the simulation can be seen in the following **Figures 2**, **3.** and **4.**

It is clearly visible that the tracking of the Cartesian-coordinates are fairly accurate, while the calculated joint coordinates differ somewhat from the nominal joint coordinates. This is due to the fact that some parts of the trajectory can be reached through different arm configurations, in other words, the solution that the method offers

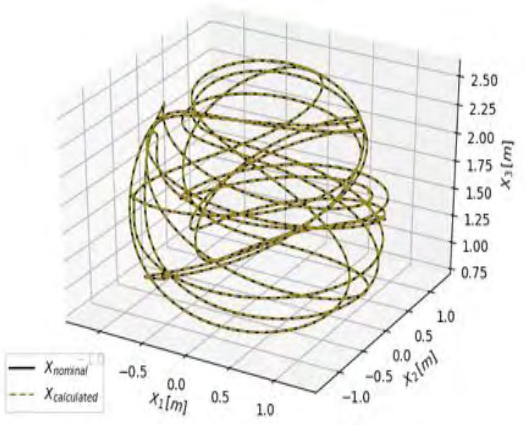

Figure 2. *3D plot of the nominal and the calculated Cartesian-coordinates*

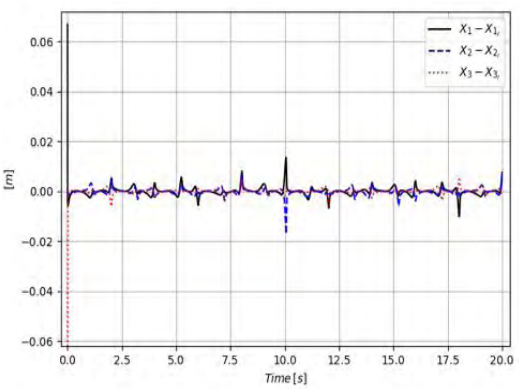

Figure 3. *Trajectory tracking error*

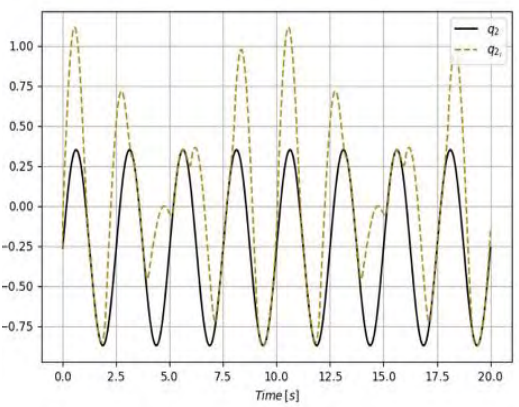

Figure 4. *The nominal and the calculated q_2 joint coordinates*

is realized in tracking via different configurations than the configurations we generated for the trajectory originally. The separation of the two paths can happen in the kinematic singularities.

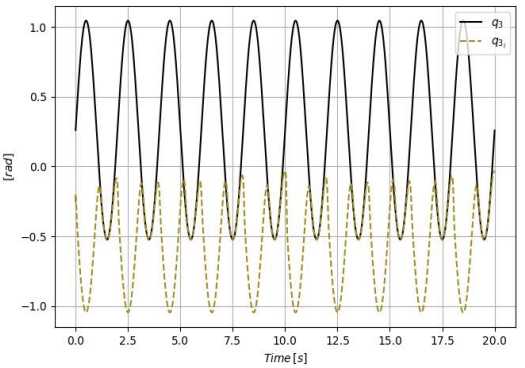

Figure 5. *The nominal and the calculated q3 joint coordinates*

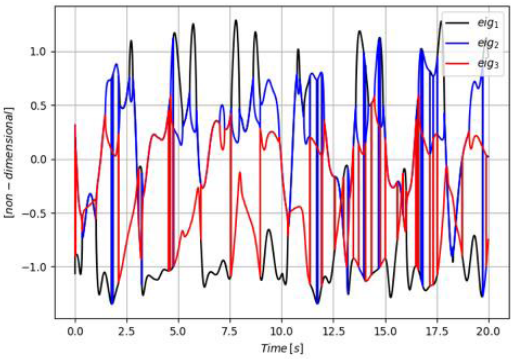

Figure 6. *The real part of the eigenvalues of the Jacobian matrix*

It is worth examining the eigenvalues of the Jacobian matrix during the simulation. It can be plotted with minimal extension in the program code.

It can be seen that there are several points where at least one of the eigenvalues is zero, which is where the robot-arm goes through several kinematic singularities during the motion.

3. Conclusion

This “quasi-differential” approach to solving the inverse kinematic task can be implemented easily, its parameters can be set relatively fast through simulations. It gives fairly accurate trajectory tracking even in the kinematic singularities and nearby those points as well.

In the case that we wish to achieve more precise tracking of the joint coordinates, (for example, if we want to get back the same configurations as the generated one), we can extend the array x with the end point of the second link, and also extend the corresponding Jacobian. This way we can get closer to the desired solution.

Acknowledgments

This paper has been supported by the “Ministry of Human Capacities” application number ÚNKP-17-1-I, and application number NTP-HHTDK-17-0047.

References

[1] B. Csanádi, J. K. Tar, J. F. Bitó: *Matrix inversion-free quasi-differential approach in solving the inverse kinematic task*. In Proc. of the 17th IEEE International Symposium on Computational Intelligence and Informatics (CINTI 2016), November 17-19, 2016, Budapest, Hungary, 61–66.

[2] A. Dineva, J. K. Tar, A. Várkonyi-Kóczy, V. Piuri: *Adaptive control of underactuated mechanical systems using improved "Sigmoid Generated Fixed Point Transformation" and scheduling strategy*. In Proc. of the 14th IEEE International Symposium on Applied Machine Intelligence and Informatics, January 21-23, 2016, Herlany, Slovakia, 193–197.

[3] T. Faitli: *Robotkar dinamikai szabályozásának szimulációs összehasonlító vizsgálata "Fixpont transzformációs adaptív" és "Fixpont transzformáción alapuló modell referenciás adaptív (MRAC)" szabályozóval*, ÓE-BGK Kari TDK Conference, November 15, 2017, Budapest, Hungary.