Forward and Inverse Kinematics Using Pseudoinverse and Transposition Method for Robotic Arm DOBOT

Ondrej Hock and Jozef Šedo

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71417

Abstract

Kinematic structure of the DOBOT manipulator is presented in this chapter. Joint coordinates and end-effector coordinates of the manipulator are functions of independent coordinates, i.e., joint parameters. This chapter explained forward kinematics task and issue of inverse kinematics task on the structure of the DOBOT manipulator. Linearization of forward kinematic equations is made with usage of Taylor Series for multiple variables. The inversion of Jacobian matrix was used for numerical solution of the inverse kinematics task. The chapter contains analytical equations, which are solution of inverse kinematics task. It should be noted that the analytical solution exists only for simple kinematic structures, for example DOBOT manipulator structure. Subsequently, simulation of the inverse kinematics of the above-mentioned kinematic structure was performed in the Matlab Simulink environment using the SimMechanics toolbox.

Keywords: forward kinematics, inverse kinematics, Matlab Simulink simulation, robotic arm, Jacobian matrix, pseudoinverse method, SimMechanics

1. Introduction

Robots and manipulators are very important and powerful instruments of today's industry. They are making lot of different tasks and operations and they do not require comfort, time for rest, or wage. However, it takes many time and capable workers for right robot function [6].

The movement of robot can be divided into forward and inverse kinematics. Forward kinematics described how robot's move according to entered angles. There is always a solution for forward kinematics of manipulator. Solution for inverse kinematics is a more difficult problem than forward kinematics. The relationship between forward kinematics and inverse kinematics is illustrated in **Figure 1**. Inverse kinematics must be solving in reverse than forward kinematics. But we know to always find some solution for inverse kinematics of manipulator. There are

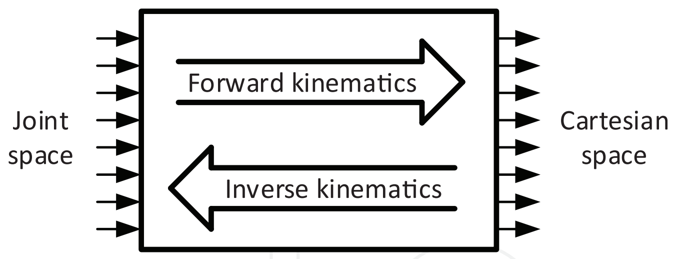

Figure 1. The schematic representation of forward and inverse kinematics.

only few groups of manipulators (manipulators with Euler wrist) with simple solution of inverse kinematics [8, 9].

Two main techniques for solving the inverse kinematics are analytical and numerical methods. In the first method, the joint variables are solved analytical, when we use classic sinus and cosine description. In the second method, the joint variables are described by the numerical techniques [9].

The whole chapter will be dedicated to the robot arm DOBOT Magician (hereafter DOBOT) shown in **Figure 2**. The basic parameters of the robotic manipulator are shown in **Figure 3** and its motion parameters are shown in **Table 1**.

This chapter is organized in the following manner. In the first section, we made the forward and inverse kinematics transformations for DOBOT manipulator. Secondly, we made the

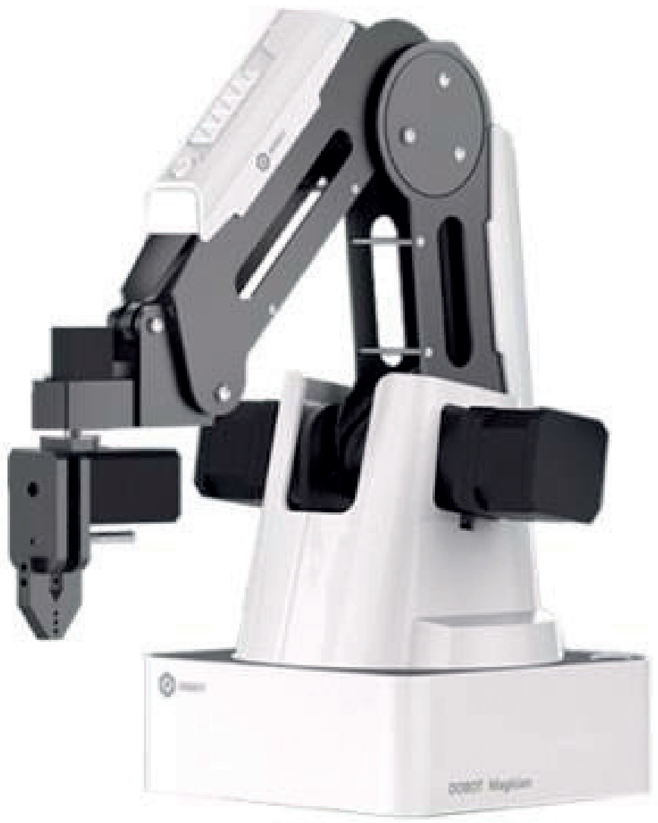

Figure 2. DOBOT Magician [10].

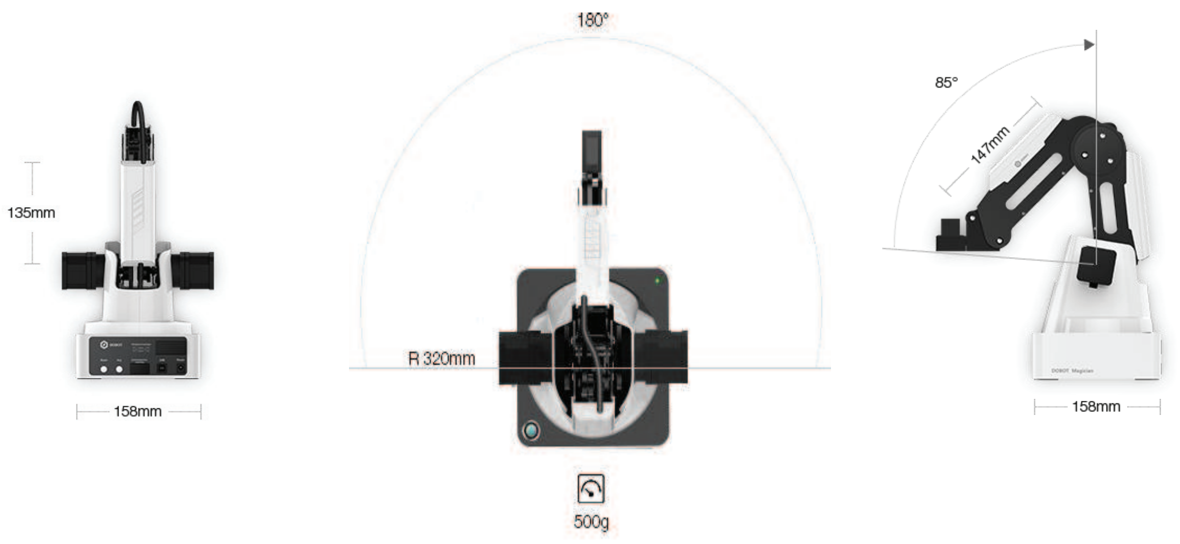

Figure 3. Simple specification of DOBOT [10].

Axis	Range	Max speed (250 g workload)
Joint 1 base	−90° to +90°	320°/s
Joint 2 rear arm	0° to +85°	320°/s
Joint 3 fore arm	−10° to +95°	320°/s
Joint 4 rotation servo	+90° to −90°	480°/s

Table 1. Axis movement of DOBOT Magician [10].

DOBOT Magician simulation in Matlab environment. Thirdly, we describe the explanation of Denavit-Hartenberg parameters. Finally, we made the pseudoinverse and transposition methods of Jacobian matrix in the inverse kinematics.

2. Kinematics structure RRR in 3D

Kinematic structure of the DOBOT manipulator is shown in **Figure 4**. It is created from three rotation joints and three links. Joint A rotates about the axis *z* and joints B and C rotate about the axis x_1.

Figure 5 shows a view from the direction of axis z and **Figure 6** shows a perpendicular view of the plane defined by z axis and line c.

Kinematic equations of the points B, C, and D, respectively:

$$x_0^B = 0 \tag{1}$$

$$y_0^B = 0 \tag{2}$$

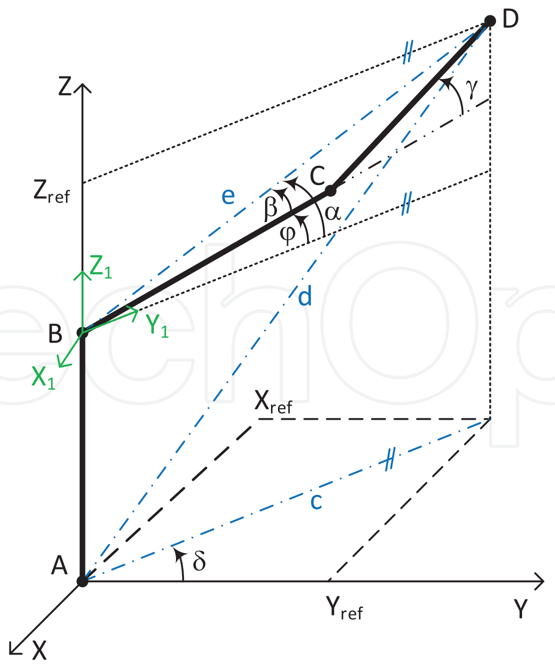

Figure 4. Representation of DOBOT manipulator in 3D view.

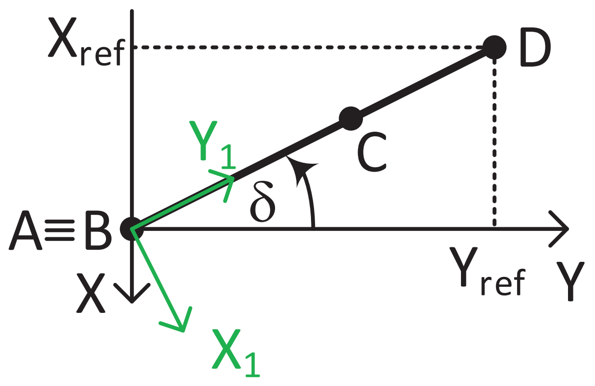

Figure 5. Representation of DOBOT footprint.

$$z_0^B = l_1 \tag{3}$$

$$x_0^C = -l_2.\cos\varphi_1.\sin\delta \tag{4}$$

$$y_0^C = l_2.\cos\varphi_1.\cos\delta \tag{5}$$

$$z_0^C = l_1 + l_2.\sin\varphi_1 \tag{6}$$

$$x_0^D = -(l_2.\cos\varphi_1 + l_3.\cos\varphi_2).\sin\delta \tag{7}$$

$$y_0^D = (l_2.\cos\varphi_1 + l_3.\cos\varphi_2).\cos\delta \tag{8}$$

$$z_0^D = l_1 + l_2.\sin\varphi_1 + l_3.\sin\varphi_2 \tag{9}$$

Where $\varphi_2 = \varphi + \gamma$.

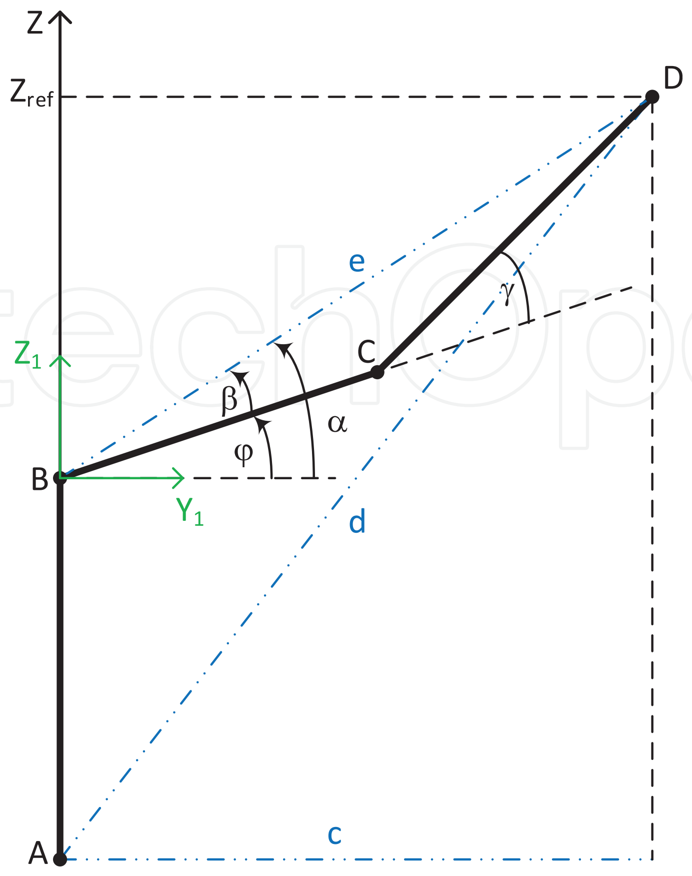

Figure 6. View of the plane defined by z axis and line c.

2.1. Forward kinematics

Forward kinematics task is defined by Eq. (10)

$$X = f(Q) \tag{10}$$

where

$$X = \begin{bmatrix} x_0^D \\ y_0^D \\ z_0^D \end{bmatrix} \tag{11}$$

X is position vector of manipulator endpoint coordinates.

$$Q = \begin{bmatrix} \varphi \\ \gamma \\ \delta \end{bmatrix} \tag{12}$$

Q is vector of independent coordinates: $\varphi = \varphi_1, \gamma, \delta$.

Because the function $X = f(Q)$ is nonlinear, it is difficult to solve the inverse task $Q = f(X)$ when looking for a vector of independent coordinates (rotation of individual manipulator joints) as a function of the desired manipulator endpoint coordinates. An analytical solution to the inverse task is possible only in the case of a relatively simple kinematic structure of the manipulator (see next chapter).

Therefore, the function $X = f(Q)$ linearized using Taylor series, taking into account only the first four (linear) members of the development:

$$x_0^D = x_0^D(\varphi,\gamma,\delta) \approx x_0^D(\varphi_0,\gamma_0,\delta_0) + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}(\varphi-\varphi_0) + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}(\gamma-\gamma_0) + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}(\delta-\delta_0) \tag{13}$$

$$y_0^D = y_0^D(\varphi,\gamma,\delta) \approx y_0^D(\varphi_0,\gamma_0,\delta_0) + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}(\varphi-\varphi_0) + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}(\gamma-\gamma_0) + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}(\delta-\delta_0) \tag{14}$$

$$z_0^D = z_0^D(\varphi,\gamma,\delta) \approx z_0^D(\varphi_0,\gamma_0,\delta_0) + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}(\varphi-\varphi_0) + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}(\gamma-\gamma_0) + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}(\delta-\delta_0) \tag{15}$$

After editing:

$$x_0^D(\varphi,\gamma,\delta) - x_0^D(\varphi_0,\gamma_0,\delta_0) = \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}(\varphi-\varphi_0) + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}(\gamma-\gamma_0) + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}(\delta-\delta_0) \tag{16}$$

$$y_0^D(\varphi,\gamma,\delta) - y_0^D(\varphi_0,\gamma_0,\delta_0) = \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}(\varphi-\varphi_0) + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}(\gamma-\gamma_0) + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}(\delta-\delta_0) \tag{17}$$

$$z_0^D(\varphi,\gamma,\delta) - z_0^D(\varphi_0,\gamma_0,\delta_0) = \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}(\varphi-\varphi_0) + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}(\gamma-\gamma_0) + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}(\delta-\delta_0) \tag{18}$$

We denoted:

$$\Delta x_0^D = x_0^D(\varphi,\gamma,\delta) - x_0^D(\varphi_0,\gamma_0,\delta_0) \tag{19}$$

$$\Delta y_0^D = y_0^D(\varphi,\gamma,\delta) - y_0^D(\varphi_0,\gamma_0,\delta_0) \tag{20}$$

$$\Delta z_0^D = z_0^D(\varphi,\gamma,\delta) - z_0^D(\varphi_0,\gamma_0,\delta_0) \tag{21}$$

$$\Delta\varphi = \varphi - \varphi_0 \tag{22}$$

$$\Delta\gamma = \gamma - \gamma_0 \tag{23}$$

$$\Delta\delta = \delta - \delta_0 \tag{24}$$

Then, we obtained:

$$\Delta x_0^D = \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\varphi + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\gamma + \left.\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\delta \tag{25}$$

$$\Delta y_0^D = \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\varphi + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\gamma + \left.\frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\delta \tag{26}$$

$$\Delta z_0^D = \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\varphi}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\varphi + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\gamma}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\gamma + \left.\frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\delta}\right|_{\varphi_0,\gamma_0,\delta_0}\Delta\delta \tag{27}$$

In matrix form:

$$\begin{bmatrix}\Delta x_0^D\\ \Delta y_0^D\\ \Delta z_0^D\end{bmatrix} = \begin{bmatrix}\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\varphi} & \frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\gamma} & \frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\delta}\\ \frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\varphi} & \frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\gamma} & \frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\delta}\\ \frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\varphi} & \frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\gamma} & \frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\delta}\end{bmatrix}_{\varphi_0,\gamma_0,\delta_0} \cdot \begin{bmatrix}\Delta\varphi\\ \Delta\gamma\\ \Delta\delta\end{bmatrix} \tag{28}$$

Where matrix:

$$J = \begin{bmatrix}\frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\varphi} & \frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\gamma} & \frac{\partial x_0^D(\varphi,\gamma,\delta)}{\partial\delta}\\ \frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\varphi} & \frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\gamma} & \frac{\partial y_0^D(\varphi,\gamma,\delta)}{\partial\delta}\\ \frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\varphi} & \frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\gamma} & \frac{\partial z_0^D(\varphi,\gamma,\delta)}{\partial\delta}\end{bmatrix}_{\varphi_0,\gamma_0,\delta_0} \tag{29}$$

is Jacobian matrix. We denoted:

$$\Delta X_0^D = \begin{bmatrix} \Delta x_0^D \\ \Delta y_0^D \\ \Delta z_0^D \end{bmatrix} \tag{30}$$

and

$$\Delta Q = \begin{bmatrix} \Delta \varphi \\ \Delta \gamma \\ \Delta \delta \end{bmatrix} \tag{31}$$

Then, we obtained the matrix equation, which represents linearized forward kinematics in incremental form:

$$\Delta X_0^D = J.\Delta Q \tag{32}$$

After we multiplied the Eq. (32) with inverse matrix J^{-1} from the left, we obtained the equation of inverse kinematics.

$$J^{-1}.\Delta X_0^D = J^{-1}.J.\Delta Q \tag{33}$$

$$J^{-1}.\Delta X_0^D = I.\Delta Q \tag{34}$$

Where I is the identity matrix. After that:

$$\Delta Q = J^{-1}.\Delta X_0^D \tag{35}$$

Derivative of the kinematic equations with respect to the independent coordinates for kinematic structure of DOBOT manipulator:

$$\frac{\partial x_0^D}{\partial \varphi} = [l_2.\sin\varphi + l_3.\sin(\varphi+\gamma)].\sin\delta \tag{36}$$

$$\frac{\partial x_0^D}{\partial \gamma} = l_3.\sin(\varphi+\gamma).\sin\delta \tag{37}$$

$$\frac{\partial x_0^D}{\partial \delta} = -[l_2.\cos\varphi + l_3.\cos(\varphi+\gamma)].\cos\delta \tag{38}$$

$$\frac{\partial y_0^D}{\partial \varphi} = -[l_2.\sin\varphi + l_3.\sin(\varphi+\gamma)].\cos\delta \tag{39}$$

$$\frac{\partial y_0^D}{\partial \gamma} = -l_3.\sin(\varphi+\gamma).\cos\delta \tag{40}$$

$$\frac{\partial y_0^D}{\partial \delta} = -[l_2.\cos\varphi + l_3.\cos(\varphi+\gamma)].\sin\delta \tag{41}$$

$$\frac{\partial z_0^D}{\partial \varphi} = l_2.\cos\varphi + l_3.\cos(\varphi + \gamma) \tag{42}$$

$$\frac{\partial z_0^D}{\partial \gamma} = l_3.\cos(\varphi + \gamma) \tag{43}$$

$$\frac{\partial z_0^D}{\partial \delta} = 0 \tag{44}$$

Jacobian matrix:

$$J = \begin{bmatrix} \frac{\partial x_0^D}{\partial \varphi} & \frac{\partial x_0^D}{\partial \gamma} & \frac{\partial x_0^D}{\partial \delta} \\ \frac{\partial y_0^D}{\partial \varphi} & \frac{\partial y_0^D}{\partial \gamma} & \frac{\partial y_0^D}{\partial \delta} \\ \frac{\partial z_0^D}{\partial \varphi} & \frac{\partial z_0^D}{\partial \gamma} & \frac{\partial z_0^D}{\partial \delta} \end{bmatrix} \tag{45}$$

2.2. Analytical Solution of the Inverse Kinematics of DOBOT manipulator

The following equations are derived from **Figure 4**.

$$c^2 = x^2 + y^2 \tag{46}$$

$$d^2 = c^2 + z^2 = x^2 + y^2 + z^2 \tag{47}$$

$$e^2 = l_2^2 + l_3^2 - 2l_2l_3\cos(\pi - \gamma) \tag{48}$$

$$\gamma = \pm\arccos\left(\frac{e^2 - l_2^2 - l_3^2}{2l_2l_3}\right) \tag{49}$$

$$e^2 = c^2 + (z - l_1)^2 \tag{50}$$

$$\varphi = \alpha - \beta \tag{51}$$

$$\alpha = arctg\frac{z - l_1}{c} = arctg\frac{z - l_1}{\sqrt{x^2 + y^2}} \tag{52}$$

$$l_3^2 = l_2^2 + e^2 - 2l_2.e.\cos\beta \tag{53}$$

$$\beta = \pm\arccos\left(\frac{l_2^2 + e^2 - l_3^2}{2l_2e}\right) \tag{54}$$

$$\varphi = arctg\frac{z - l_1}{\sqrt{x^2 + y^2}} \mp \arccos\left(\frac{l_2^2 + e^2 - l_3^2}{2l_2e}\right) \tag{55}$$

$$\delta = arctg\frac{-x}{y} \tag{56}$$

3. Simulation DOBOT Magician in Matlab environment

For simulation movement of the manipulator, Matlab Simulink environment and SimMechanics toolbox are suitable to use. Some blocks from SimMechanics toolbox are shown in **Figure 7**, which represents the model of DOBOT manipulator.

We used basic block from SimMechanics toolbox in simulation model:

- Joint actuator
- Revolute
- Body
- Body sensor
- Machine environment

The joint actuator block transfers the requested angles to the connected joint. The revolute block defined the rotation of body in space. The body block describes the parameters of body,

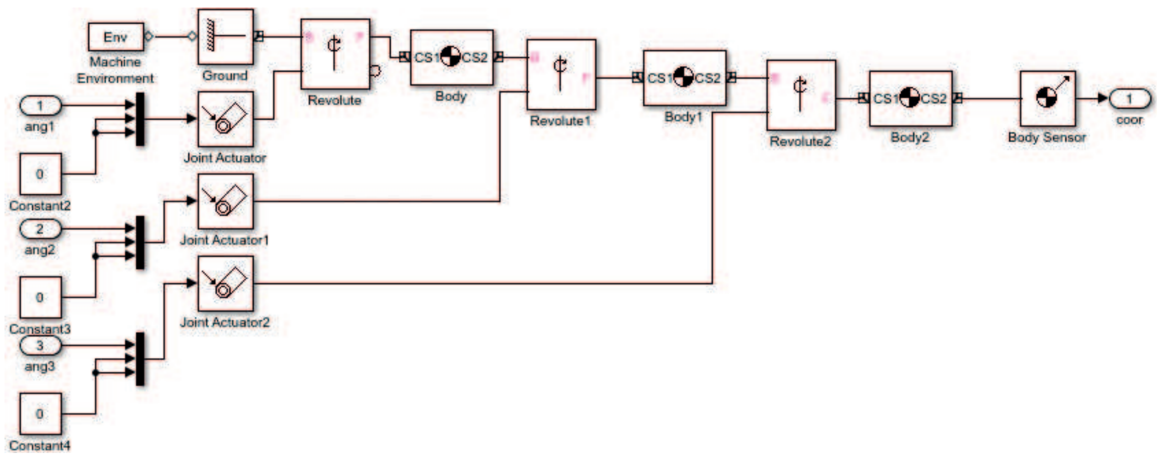

Figure 7. SimMechanics simulation model of DOBOT manipulator.

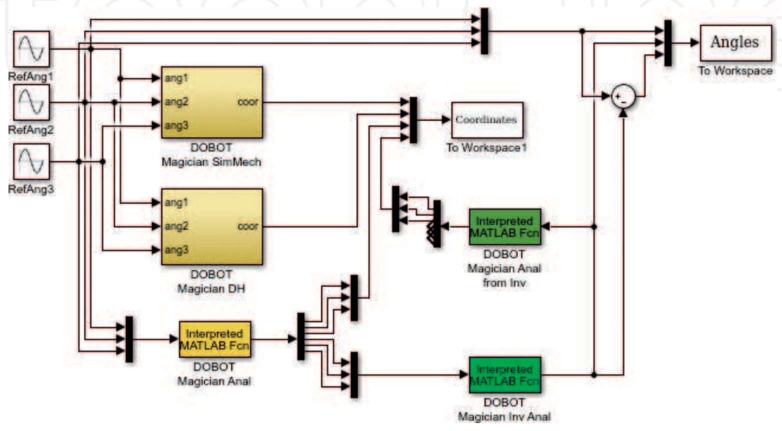

Figure 8. Simulation of DOBOT manipulator in Matlab environment.

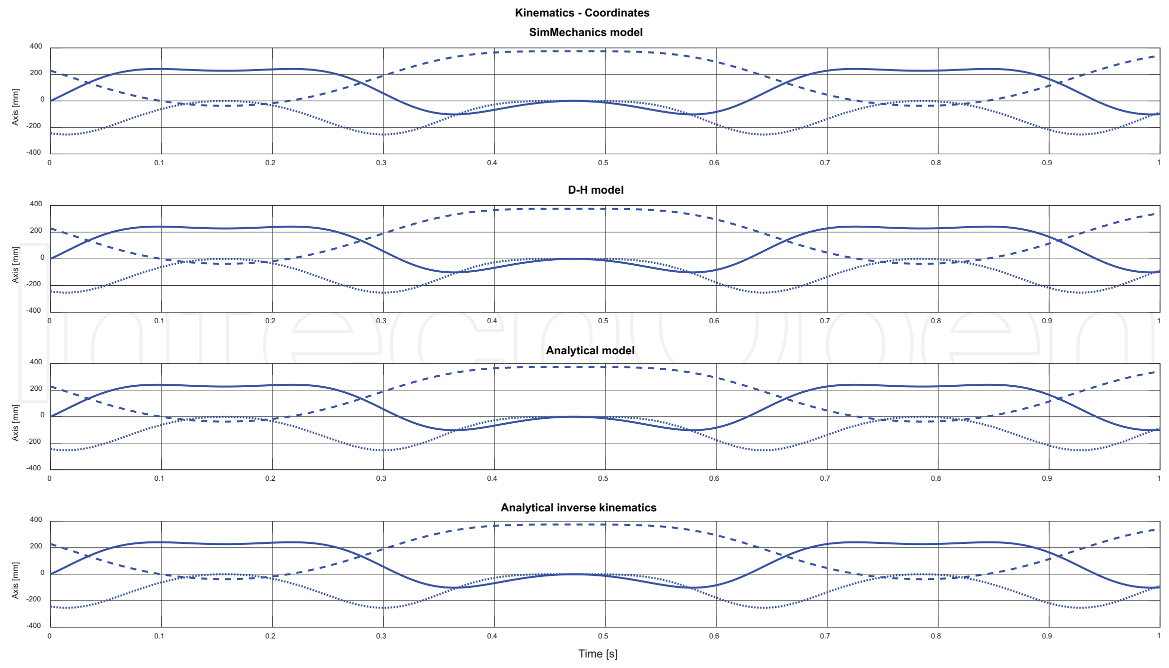

Figure 9. Endpoint coordinates of DOBOT manipulator.

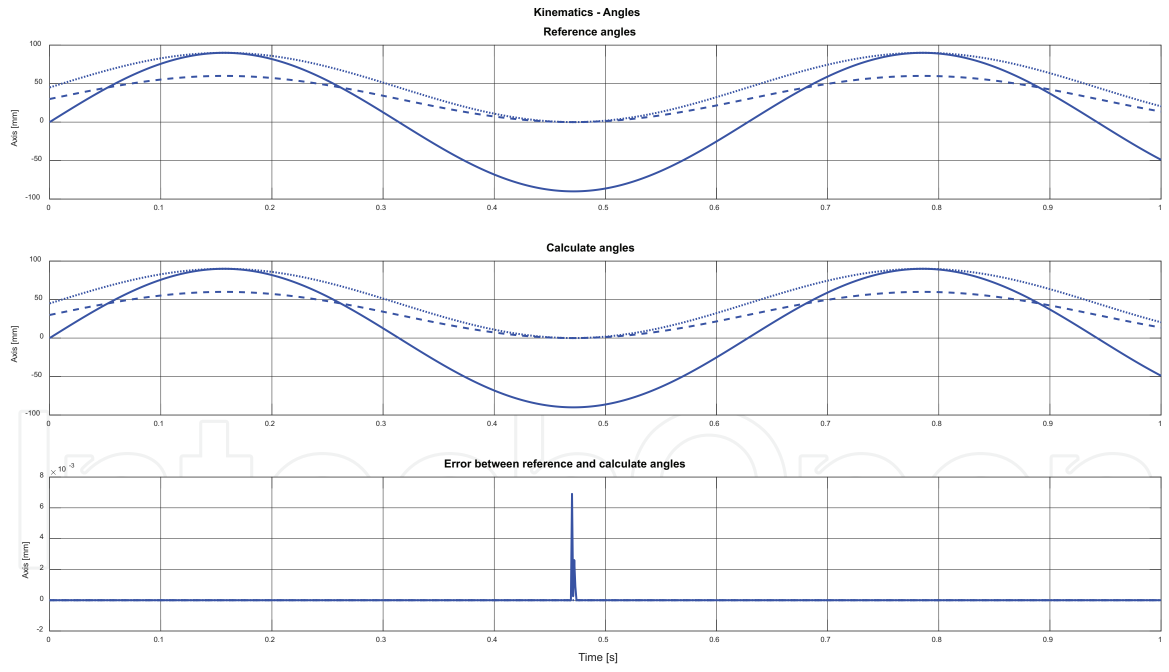

Figure 10. Reference angles, calculated angles, and error between these angles.

like dimension, inertia, etc. The body sensor block transfers the coordinates, velocity, and others to simulation, and the last block is the machine environment which defines the parameters of the environment in which the manipulator is located. You can find all necessary data about these blocks in [7].

The simulation model, shown in **Figure 8**, was designed for considerate results from SimMechanics model, model used D-H parameters and analytical model, which was described in previous chapter. Every result from these models is shown in **Figure 9**. The fourth part in **Figure 9** is the results from analytical simulation model of inverse kinematics. **Figure 10** represented the reference angles in the first part of the chart, calculated angles from analytical inverse kinematics model in the second part of chart, and finally the error between both angles. As we can see in **Figure 10**, the angles are same. This is proof that analytical model of DOBOT manipulator is usable for simulation and implementation to some DSP or microcontroller.

4. Denavit-Hartenberg parameters

The steps to get the position in using D-H convention are finding the Denavid-Hartenberg (D-H) parameters, building A matrices, and calculating T matrix with the coordinate position which is desired.

4.1. D-H parameters

D-H notation describes coordinates for different joints of a robotic manipulator in matrix entry. The method includes four parameters:

1. Twist angle α_{i}
2. Link length a_{i}
3. Link offset d_{i}
4. Joint angle θ_{i}.

Based on the manipulator geometry, twist angle and link length are constants and link offset and joint angle are variables depending on the joint, which can be prismatic or revolute. The method has provided 10 steps to denote the systematic derivation of the D-H parameters, and you can find them in [5] or [6].

4.2. A matrix

The A matrix is a homogenous 4 × 4 transformation matrix. Matrix describes the position of a point on an object and the orientation of the object in a three-dimensional space [6]. The homogenous rotation matrix along an axis is described by the Eq. (57) (**Figures 11–13**).

$$Rot_{z_{i-1}} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & 0 \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \sin\theta_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{57}$$

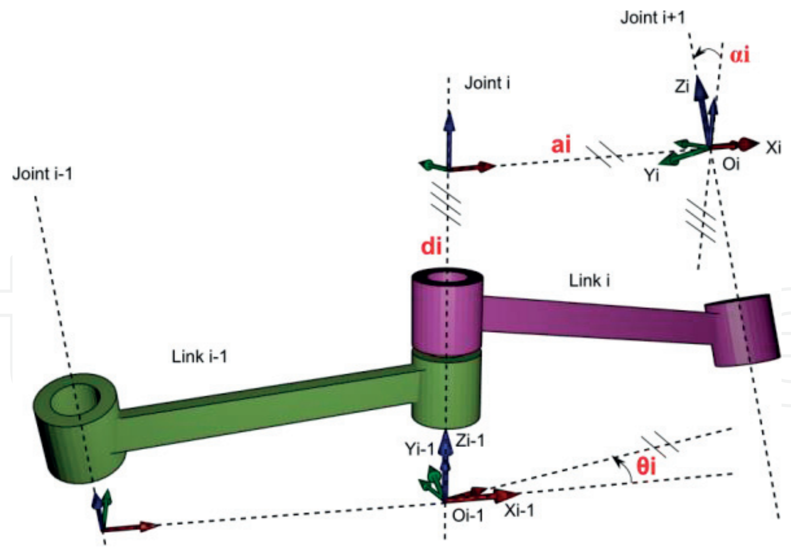

Figure 11. The four parameters of classic DH convention are $\theta_i, d_i, a_i, \alpha_i$ [4].

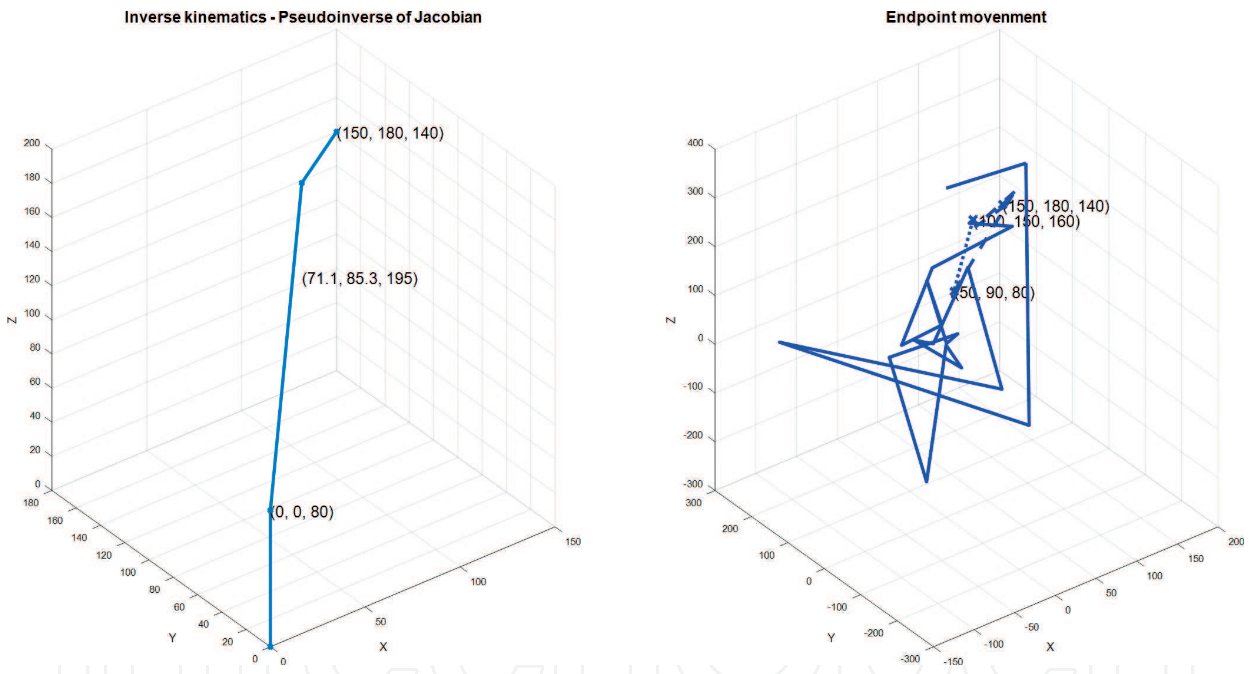

Figure 12. Simulation result of Jacobian matrix pseudoinverse in inverse kinematics model of the DOBOT manipulator.

The homogeneous translation matrix is described by Eq. (58).

$$Trans_{z_{i-1}} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{58}$$

In rotation matrix and translation matrix, we can find the four parameters θ_{i}, d_{i}, a_{i}, and α_{i}. These parameters derive from specific aspects of the geometric relationship between two coordinate

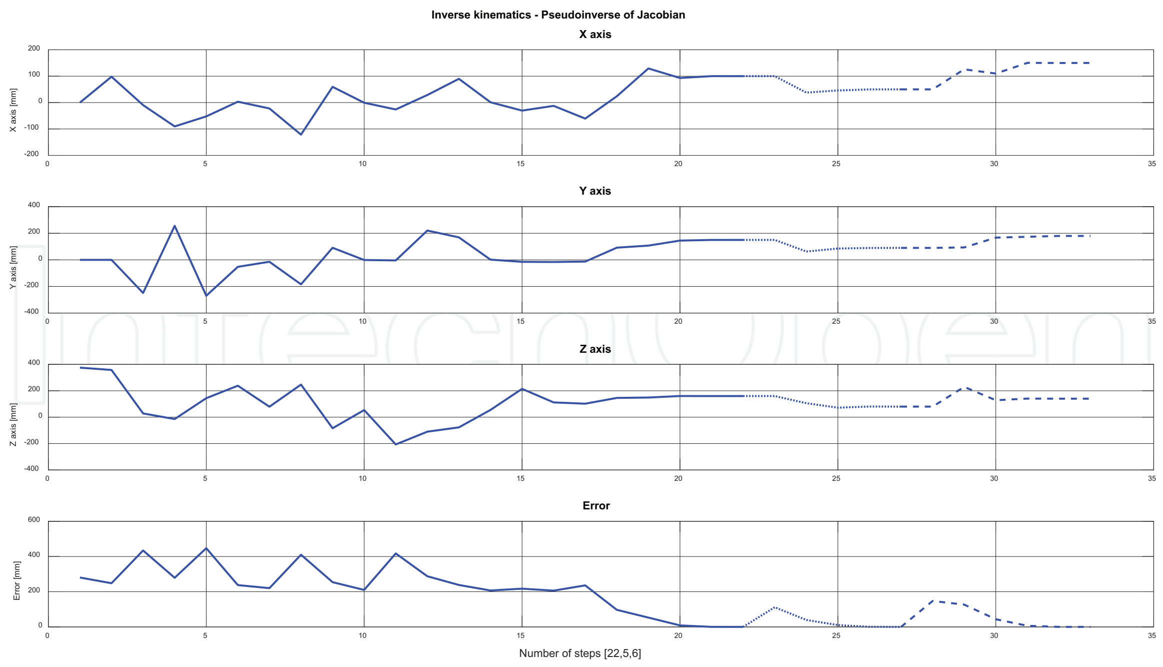

Figure 13. Simulation results of DOBOT axis and total error of coordinates for Pseudoinverse method.

frames. The four parameters are associated with link *i* and joint *i*. In Denavit-Hartenberg convention, each homogeneous transformation matrix A_i is represented as a product of four basic transformations as follows [6]:

$$T_i^{i-1} = Trans_{z_{i-1}}(d_i).Rot_{z_{i-1}}(\theta_i).Trans_{x_i}(r_i).Rot_{x_i}(\alpha_i) \tag{59}$$

D-H convention matrix is given in Eq. (60).

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & r_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & r_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{60}$$

The previous matrix can be simplified by following equation A_i matrix. The matrix A_i is composed from 3 × 3 rotation matrix R_i, 3 × 1 translation vector P_i, 1 × 3 perspective vector and scaling factor.

$$A_i = \begin{bmatrix} R_{i(3x3)} & P_{i(3x1)} \\ 0_{(1x3)} & 1 \end{bmatrix} \tag{61}$$

4.3. T matrix

The T matrix can be formulated by Eq. (62). The matrix is a sequence of D-H matrices and is used for obtaining end-effector coordinates. The T matrix can be built from several A matrices depending on the number of manipulator joints.

http://dx.doi.org/10.5772/intechopen.71417

$$T_0^3 = T_0^1 . T_1^2 . T_2^3 \tag{62}$$

Inside the T matrix is the translation vector P_i, which includes joint coordinates, where the X, Y, and Z positions are P_1, P_2, and P_3, respectively [6].

5. The pseudoinverse method

If the number of independent coordinates n (joint parameters) is larger than the number of reference manipulator endpoint coordinates m (three in Cartesian coordinate system for the point), it shows that a redundancy problem has occurred. In this case, it can exist in infinite combinations of independent coordinates for the only endpoint position. Jacobian matrix J has a size of m rows and n columns ($m \neq n$), i.e., J is a non-square matrix. In general, it cannot be computed inverse matrix from non-square matrix.

In order to solve inverse kinematics task for this case, pseudoinverse of Jacobian matrix (denotes J^+) is used. This method uses singular value decomposition (SVD) of Jacobian matrix to determine J^+.

Every matrix J can be decomposed with the usage of SVD to three matrices Eq. (63):

$$J = U \Sigma V^T \tag{63}$$

Where

J is $m \times n$ matrix.

U is $m \times m$ orthogonal matrix, i.e. $U^{-1} = U^T$.

V is $n \times n$ orthogonal matrix, i.e. $V^{-1} = V^T$.

Σ is $m \times n$ diagonal matrix, which contains singular values of matrix J on its major diagonal.

$$\begin{bmatrix} j_{11} & j_{12} & \cdots & j_{1n} \\ j_{21} & j_{22} & \cdots & j_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ j_{m1} & j_{m2} & \cdots & j_{mn} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mm} \end{bmatrix} . \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_d \end{bmatrix} . \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{n1} \\ v_{12} & v_{22} & \cdots & v_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1n} & v_{2n} & \cdots & v_{nn} \end{bmatrix} \tag{64}$$

Where $d = m$ for $m < n$ and $d = n$ for $m > n$, because Σ is a non-square matrix.

To determine matrices U and Σ, we multiply matrix J by its transpose matrix J^T from the right:

$$J.J^T = (U \Sigma V^T).(U \Sigma V^T)^T \tag{65}$$

$$J.J^T = U \Sigma V^T . V \Sigma^T U^T \tag{66}$$

$$J.J^T = U\Sigma\Sigma^T U^T \tag{67}$$

We multiply the above Eq. (67) by matrix U from the right:

$$JJ^T.U = U.\Sigma\Sigma^T.U^T U \tag{68}$$

$$JJ^T.U = U.\Sigma\Sigma^T \tag{69}$$

It leads to eigenvalue problem for JJ^T matrix. U is $m \times m$ square matrix, which contains eigenvectors of JJ^T matrix in its columns and $\Sigma\Sigma^T$ is diagonal matrix of eigenvalues $\lambda_1, \ldots, \lambda_m$.

To determine matrices V and Σ, we multiply matrix J by its transpose matrix J^T from the left:

$$J^T.J = \left(U\Sigma V^T\right)^T.\left(U\Sigma V^T\right) \tag{70}$$

$$J^T.J = V\Sigma^T U^T.U\Sigma V^T \tag{71}$$

$$J^T.J = V\Sigma^T\Sigma V^T \tag{72}$$

We multiply the above Eq. (72) by matrix V from the right:

$$J^T J.V = V.\Sigma^T\Sigma.V^T.V \tag{73}$$

$$J^T J.V = V.\Sigma^T\Sigma \tag{74}$$

It leads to eigenvalue problem for J^TJ matrix. V is $n \times n$ square matrix, which contains eigenvectors of J^TJ matrix in its columns and $\Sigma^T\Sigma$ is diagonal matrix of eigenvalues $\lambda_1, \ldots, \lambda_n$.

Matrices JJ^T and J^TJ are symmetric matrices and they have the same nonzero eigenvalues. Eigenvalues and eigenvectors of the real symmetric matrices are always real numbers and real vectors.

The eigenvalues are equal to square of the singular values: $\lambda_i = \sigma_i^2$, where $i = 1, \ldots, d$. The number of nonzero eigenvalues is $d = m$ for $m < n$ and $d = n$ for $m > n$. The number of zero eigenvalues is $|m - n|$.

When values of matrices U, Σ, and V were computed, we can determine pseudoinverse of Jacobian matrix as follows:

$$J^+ = \left(U.\Sigma.V^T\right)^{-1} \tag{75}$$

$$J^+ = V.\Sigma^+.U^T \tag{76}$$

Where

$$\Sigma^{+} = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_d} \end{bmatrix} \tag{77}$$

Now, we can solve inverse kinematics task for the cases, when Jacobian matrix is non-square:

$$\Delta Q = J^{+}.\Delta X \tag{78}$$

Pseudoinverse J^{+}, also called Moore-Penrose inverse of Jacobian matrix, gives the best possible solution in the sense of least squares [1].

6. The Jacobian matrix transpose method

We designed Jacobian matrix transpose method simulation [1–3]. The basic idea was written using Eq. (79). We used the transpose of Jacobian matrix, instead of the inverse of Jacobian matrix, in this method. We set $\Delta\theta$ equal to

$$\Delta\theta = \alpha J^{T}\ \vec{e} \tag{79}$$

Where α is:

$$\alpha = \frac{\left\langle \vec{e}, JJ^{T}\ \vec{e} \right\rangle}{\left\langle JJ^{T}\ \vec{e}, JJ^{T}\ \vec{e} \right\rangle} \tag{80}$$

Whole simulation is described by block diagram shown in **Figure 14**. In the first step, we defined requested error. This error represented difference between reference coordinates and actual coordinates. Error that we consider as unacceptable, we set to 200 μm. This is position repeatability of DOBOT. We calculate the increment of requesting angles $\Delta\theta$ in each iteration. In the first iteration, $\Delta\theta$ is equal to zero.

Figures 15 and **16** represent simulation result of DOBOT movement same as in simulation of Jacobian matrix pseudoinverse. Simulation was split on three parts. First part (solid line in chart) is movement from starting position to position (x, y, z) = (100, 150, 160) mm. Second part (dotted line in chart) is movement from previous position to position (50, 90, 80). And third part (dashed line in chart) is movement to position (150, 180, 140).

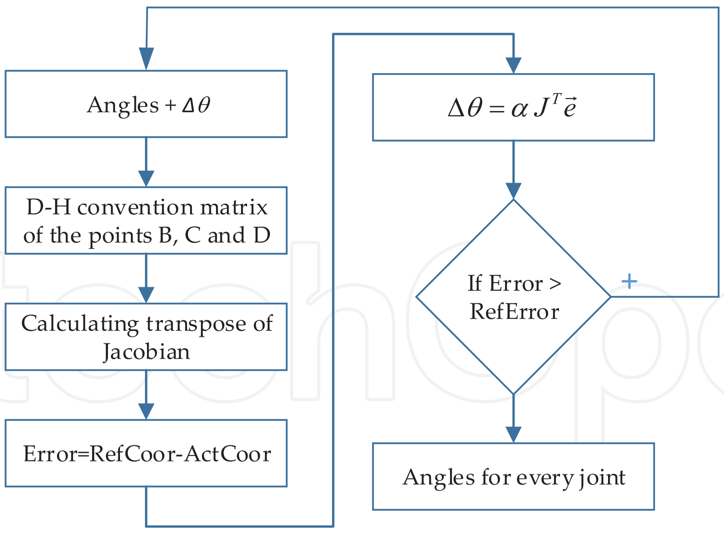
Angles + $\Delta\theta$
D-H convention matrix of the points B, C and D
Calculating transpose of Jacobian
Error=RefCoor-ActCoor
$\Delta\theta = \alpha J^T \vec{e}$
If Error > RefError
+
Angles for every joint

Figure 14. Block diagram of Jacobian matrix transpose method simulation.

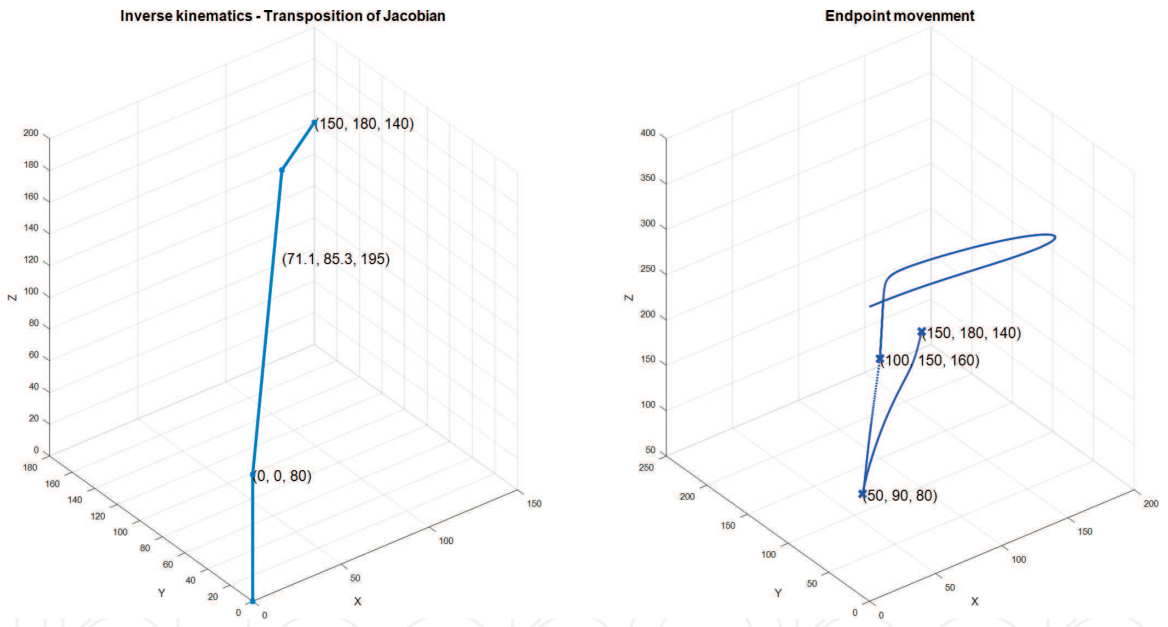

Figure 15. Simulation result of Jacobian matrix transposition in inverse kinematics model of the DOBOT manipulator.

7. Conclusion

As we can see in simulation results from previous subchapters, every method for inverse kinematics has some positives and negatives. Comparison of both methods is shown in **Table 2**. Pseudoinverse method is faster than transposition method, but is harder to implement in a DSP or a microcontroller. In Matlab environment, pseudoinverse method is easily made by the pinv() command. If we want to simplify inverse kinematics and we don't need fast calculating time, it is more readily to use transposition method. In the case of using DOBOT manipulator, it is considered to use the analytical model. In the case of more complicated manipulator, this method is inapplicable.

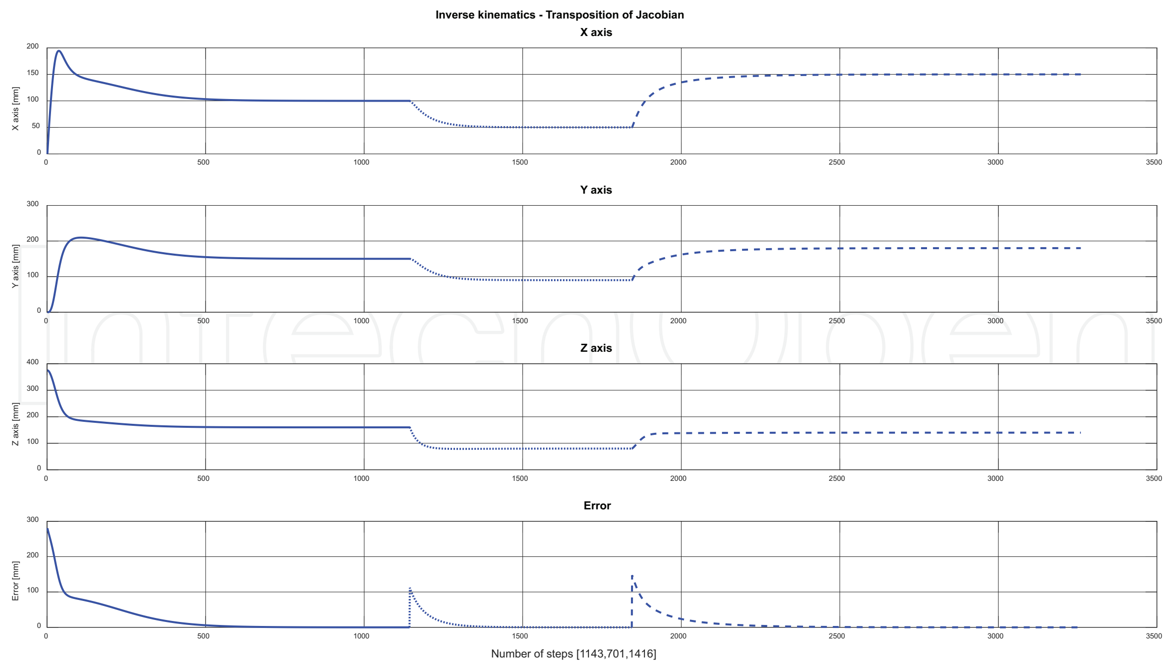

Figure 16. Simulation results of DOBOT axis and total error of coordinates for Transposition method.

Simulation part	Pseudoinverse method (number of iterations)	Transposition method (number of iterations)
Part 1 (solid line)	22	55
Part 2 (dotted line)	5	34
Part 3 (dashed line)	6	68

Table 2. Comparison of pseudoinverse and transposition method.

Comparison of both methods is shown in **Table 2**. As we can see in **Table 2**, the main criteria are number of iterations. Pseudoinverse method is much better, but only for simulation. If we can use this method in real-time application, like dSPACE from MathWorks® or implementation to DSP, we will not achieve such results like in **Table 2**. It is caused by using singular value decomposition (SVD), which is very demanding for a computation performance. In the other case, transposition of Jacobian matrix is much easier for implementation and need lower performance.

In the next research, we considerate the use suitable iterative method, like damped least squares. We also designed several implementation methods of Jacobian matrix transposition to DSP (TMS430, C2000™). It is very important to try more implementation methods for the most possible shortening of the calculation time.

Acknowledgements

The results of this work are supported by Grant No. APVV-15-0571: research of the optimum energy flow control in the electric vehicle system.

Author details

Ondrej Hock and Jozef Šedo*

*Address all correspondence to: jozef.sedo@fel.uniza.sk

Department of Mechatronics and Electronics, Faculty of Electrical Engineering, University of Žilina, Žilina, Slovakia

References

[1] Buss SR. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation. 2004;**17**:1-19

[2] Balestrino A, De Maria G, Sciavicco L. Robust control of robotic manipulators. In: Proceedings of the 9th IFAC World Congress; 1984;**5**:2435-2440

[3] Wolovich WA, Elliot H. A computational technique for inverse kinematics. In: Proceedings of the 23rd IEEE Conference on Decision and Control; 1984. pp. 1359-1363

[4] Denavit–Hartenberg [Internet]. 2017. Available from: https://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters [Accessed: 03-08-2017]

[5] Desai JP. D-H Convention, Robot and Automation Handbook. USA: CRC Press; 2005 ISBN: 0-8493-1804-1

[6] Rehiara AB. Kinematics of Adept Three Robot Arm. In: Satoru Goto, editor. Robot Arms. 2011. InTech. ISBN: 978-953-307-160-2. Available from: http://www.intechopen.com/books/robot-arms/kinematicsof-adeptthree-robot-arm [Accessed: 01-08-2017]

[7] The MathWorks, Inc., SimMechanics 2User's Guide [Internet], 2007. Available from: https://mecanismos2mm7.files.wordpress.com/2011/09/tutorial-sim-mechanics.pdf [Accessed: 08-04-2017]

[8] Bingul KS. The inverse kinematics solutions of industrial robot manipulators. In: IEEE Conference on Mechatronics; Istanbul, Turkey; 2004. pp. 274-279

[9] Serdar Kucuk, Zafer Bingul. In: Sam Cubero, editor. Robot Kinematics: Forward and Inverse Kinematics, Industrial Robotics: Theory, Modelling and Control. 2006. InTech. ISBN: 3-86611-285-8. Available from: http://www.intechopen.com/books/industrial_robotics_theory_modelling_and_control/robot_kinematics__forward_and_inverse_kinematics [Accessed 01-08-2017]

[10] WebPage [Internet], 2017. Available from: http://www.dobot.cc/ [Accessed: April 04, 2017]