Mathematical Model of Robot Melfa RV-2SDB

Peter Papcun and Ján Jadlovský

Department of Cybernetics and Artificial Intelligence (DCAI),
Faculty of Electrical Engineering and Informatics (FEEI), Technical University of Košice,
Letná 9, 042 00 Košice, Slovakia

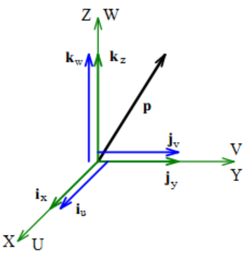
Abstract. This article deals with the methodology of processing direct kinematics for a robotic arm. The paper describes basic matrices of direct kinematics and basic relations. A method of application and calculation of direct kinematics for industrial robot is also presented. The last part is the application of general inverse kinematics algorithm. This mathematical model is verified in the article.

1 Introduction

The article examines the mathematical model of the robotic arm MELFA RV-2SDB, it describes the general methodology of creating the robotic arm's mathematical model. The second chapter reminds basic matrices which are dedicated for calculation of the direct kinematics model. The next chapter describes the procedure of creating the mathematical model in steps. The fourth chapter describes the general algorithm for the inverse kinematics of the robotic arm. The fifth chapter compares the mathematical model and the real model of the robotic arm experimentally.

2 Direct Kinematics

This chapter reminds basic matrices for the calculation of direct kinematics. Mathematical descriptions of these matrices are described in [1]. This chapter summarizes all of the basic matrices only. Basic kinematics matrix:

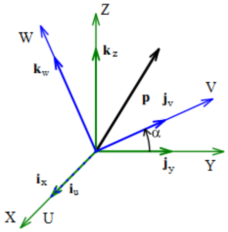


$$\underline{R} = \begin{bmatrix} \underline{i}_x \cdot \underline{i}_u & \underline{i}_x \cdot \underline{j}_v & \underline{i}_x \cdot \underline{k}_w \\ \underline{j}_y \cdot \underline{i}_u & \underline{j}_y \cdot \underline{j}_v & \underline{j}_y \cdot \underline{k}_w \\ \underline{k}_z \cdot \underline{i}_u & \underline{k}_z \cdot \underline{j}_v & \underline{k}_z \cdot \underline{k}_w \end{bmatrix}$$

Fig. 1. Basic kinematics

P. Sinčák et al. (eds.), *Emergent Trends in Robotics and Intelligent Systems*,
Advances in Intelligent Systems and Computing 316, DOI: 10.1007/978-3-319-10783-7_16

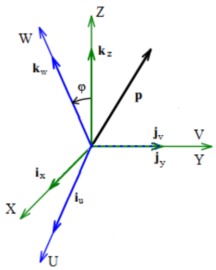
Rotation around axis X:



$$\underline{R}x,\alpha=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\alpha & -\sin\alpha\\ 0 & \sin\alpha & \cos\alpha\end{bmatrix}$$

Fig. 2. Rotation around axis X

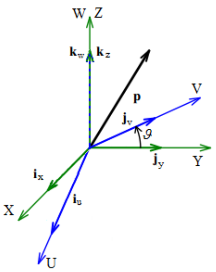
Rotation around axis Y:



$$\underline{R}y,\varphi=\begin{bmatrix}\cos\varphi & 0 & \sin\varphi\\ 0 & 1 & 0\\ -\sin\varphi & 0 & \cos\varphi\end{bmatrix}$$

Fig. 3. Rotation around axis Y

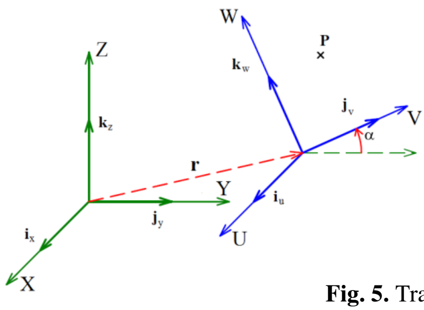
Rotation around axis Z:



$$\underline{R}z,\vartheta=\begin{bmatrix}\cos\vartheta & -\sin\vartheta & 0\\ \sin\vartheta & \cos\vartheta & 0\\ 0 & 0 & 1\end{bmatrix}$$

Fig. 4. Rotation around axis Z

Translation:



$$P=\left[\begin{array}{ccc|c} & & & r_x\\ & \underline{R} & & r_y\\ & & & r_z\\ \hline 0 & 0 & 0 & 1\end{array}\right]$$

Fig. 5. Translation

3 Industrial Robot

This chapter describes the methodology of the creating and calculation of the direct kinematics mathematical model. This methodology is applied on the industrial robot MELFA RV-2SDB made by Mitsubishi corporation. We propose this methodology.

The First Step: Present Technological Parameters of the Robotic Arm:
Direct kinematics calculation needs to know accurate dimensions of the robotic arm and its maximal rotation:

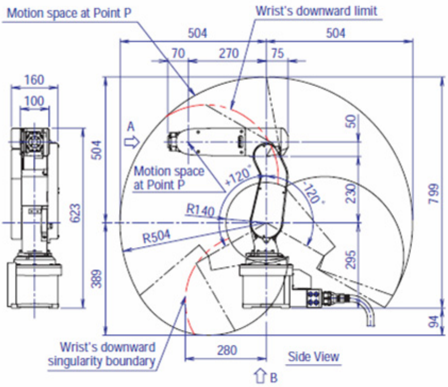


Fig. 6. Dimensions of robotic arm MELFA RV-2SDB

The Second Step: Kinematics Structure of the Robotic Arm:
Calculation also needs to know the kinematics structure of the robotic arm. This 3D kinematics structure is drawn with respect to the figure of the industrial robot and then the figure is removed and leaving only 3D kinematics structure of the robotic arm:

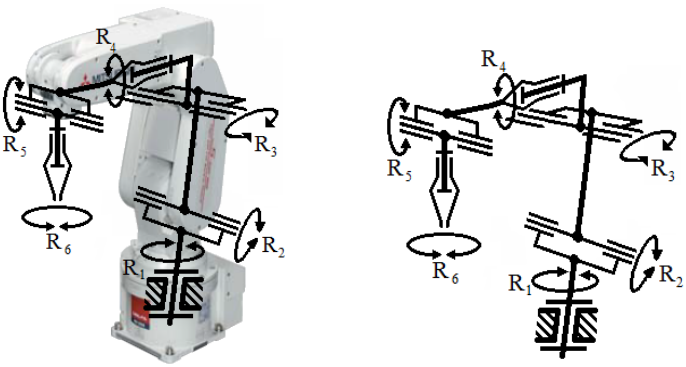


Fig. 7. 3D kinematics structure of industrial robot MELFA RV-2SDB

The 3D kinematics structure is redrawn to the classic kinematics structure for its simplification. The classic kinematics structure has arms marked with letters from *a* to *f* and joints R_1 - R_6 marked with Greeks letters (representing the angles) $\alpha, \beta, \gamma, \delta, \varepsilon, \varphi$ in the following figure:

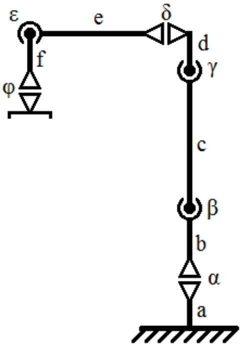


Fig. 8. Classic kinematics structure of the industrial robot MELFA RV-2SDB

Important information can be read from the scheme in Figure 8 and the dimensions of the robotic arm in Figure 6 (dimension a can be ignored because the nearest kinematics mechanism is rotary in axis z):

a = 0 mm	d = 50 mm
b = 295 mm	e = 270 mm
c = 270 mm	f = 70 mm

The next important information is the maximum range of joints:

$\alpha \in <-240°; 240°>$	$\delta \in <-200°; 200°>$
$\beta \in <-120°; 120°>$	$\varepsilon \in <-120°; 120°>$
$\gamma \in < 0°; 160°>$	$\phi \in <-360°; 360°>$

The third step: Determining the zero position of the robotic arm:

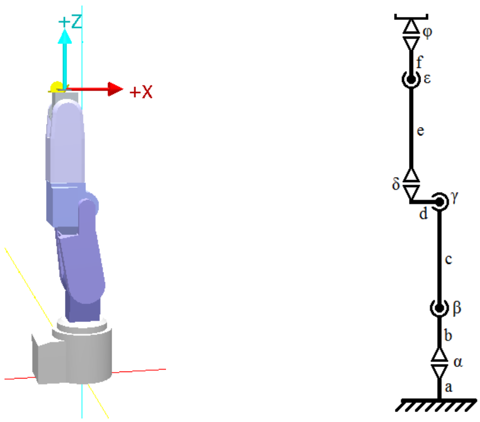


Fig. 9. Kinematics scheme of robotic arm if all angles have size 0°

The Fourth Step: Transformation Matrices:

Transformation matrices can be determined by the main information from the first three steps. They are created according to Figure 9 from the beginning of coordination system to the robot's end point.

Joint R1 is resolved to the first (because dimension a is ignored), this is the rotary motion around axis z (1), then the matrix calculation moves under joint R2 (2):

$$A = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1) \qquad B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & b \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Joint R_2 causes rotary motion around axis *y* (3), then it moves under joint R_3 (4):

$$C = \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3) \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

Joint R_3 rotates around axis *y* (5) and calculation moves under joint R_4 (6):

$$E = \begin{pmatrix} \cos\gamma & 0 & \sin\gamma & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\gamma & 0 & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5) \qquad F = \begin{pmatrix} 1 & 0 & 0 & -d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (6)$$

Joint R4 rotates around axis z (7) and it moves under joint R5 (8):

$$G = \begin{pmatrix} \cos\delta & -\sin\delta & 0 & 0 \\ \sin\delta & \cos\delta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (7) \qquad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & e \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (8)$$

Joint R_5 causes rotary motion around axis *y* (9), then it moves to last joint R_6 (10):

$$I = \begin{pmatrix} \cos\varepsilon & 0 & \sin\varepsilon & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varepsilon & 0 & \cos\varepsilon & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (9) \qquad J = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & f \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (10)$$

Last joint R_6 causes rotary motion around axis *z* (11):

$$K = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (11)$$

The fifth step: The composite homogeneous transformation matrix:

The composite homogeneous transformation matrix is calculated from all transformations matrices, which is calculated in the fourth step:

$$T = A.B.C.D.E.F.G.H.I.J.K \tag{12}$$

Resultant composite homogeneous transformation matrix:

$$T = \begin{pmatrix} n_x & o_x & a_x & x \\ n_y & o_y & a_y & y \\ n_z & o_z & a_z & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{13}$$

Equations (14) are simplified (sin(x) → sx):

$$\begin{aligned}
x &= f\,(c_\alpha c_\beta c_\gamma c_\delta s_\varepsilon - c_\alpha s_\beta s_\gamma c_\delta s_\varepsilon - s_\alpha s_\delta s_\varepsilon + c_\alpha c_\beta s_\gamma c_\varepsilon + c_\alpha s_\beta c_\gamma c_\varepsilon) + \\
&\quad + e\,(c_\alpha c_\beta s_\gamma + c_\alpha s_\beta c_\gamma) - d(c_\alpha c_\beta c_\gamma - c_\alpha s_\beta s_\gamma) + c\, c_\alpha s_\beta \\
y &= f(s_\alpha c_\beta c_\gamma c_\delta s_\varepsilon - s_\alpha s_\beta s_\gamma c_\delta s_\varepsilon + c_\alpha s_\delta s_\varepsilon + s_\alpha c_\beta s_\gamma c_\varepsilon + s_\alpha s_\beta c_\gamma c_\varepsilon) + \\
&\quad + e(s_\alpha c_\beta s_\gamma + s_\alpha s_\beta c_\gamma) - d(s_\alpha c_\beta c_\gamma - s_\alpha s_\beta s_\gamma) + c\, s_\alpha s_\beta \\
z &= f\,(c_\beta c_\gamma c_\varepsilon - s_\beta c_\gamma c_\delta s_\varepsilon - c_\beta s_\gamma c_\delta s_\varepsilon - s_\beta s_\gamma c_\varepsilon) + e(c_\beta c_\gamma - s_\beta s_\gamma) + d\,(s_\beta c_\gamma + c_\beta s_\gamma) \\
&\quad + c\, c_\beta + b
\end{aligned}$$

$$\begin{aligned}
n_x &= c_\alpha c_\beta c_\gamma c_\delta c_\varepsilon c_\varphi - c_\alpha s_\beta s_\gamma c_\delta c_\varepsilon c_\varphi - s_\alpha s_\delta c_\varepsilon c_\varphi - c_\alpha c_\beta s_\gamma s_\varepsilon c_\varphi - c_\alpha s_\beta c_\gamma s_\varepsilon c_\varphi + \\
&\quad + c_\alpha s_\beta s_\gamma s_\delta s_\varphi - c_\alpha c_\beta c_\gamma s_\delta s_\varphi - s_\alpha c_\delta s_\varphi \\
o_x &= c_\alpha s_\beta s_\gamma c_\delta c_\varepsilon s_\varphi - c_\alpha c_\beta c_\gamma c_\delta c_\varepsilon s_\varphi + s_\alpha s_\delta c_\varepsilon s_\varphi + c_\alpha c_\beta s_\gamma s_\varepsilon s_\varphi + c_\alpha s_\beta c_\gamma s_\varepsilon s_\varphi + \\
&\quad + c_\alpha s_\beta s_\gamma s_\delta c_\varphi - c_\alpha c_\beta c_\gamma s_\delta c_\varphi - s_\alpha c_\delta c_\varphi \\
a_x &= c_\alpha c_\beta c_\gamma c_\delta s_\varepsilon - c_\alpha s_\beta s_\gamma c_\delta s_\varepsilon - s_\alpha s_\delta s_\varepsilon + c_\alpha c_\beta s_\gamma c_\varepsilon + c_\alpha s_\beta c_\gamma c_\varepsilon
\end{aligned}$$

$$\begin{aligned}
n_y &= s_\alpha c_\beta c_\gamma c_\delta c_\varepsilon c_\varphi - s_\alpha s_\beta s_\gamma c_\delta c_\varepsilon c_\varphi + c_\alpha s_\delta c_\varepsilon c_\varphi - s_\alpha c_\beta s_\gamma s_\varepsilon c_\varphi - s_\alpha s_\beta c_\gamma s_\varepsilon c_\varphi + \\
&\quad + s_\alpha s_\beta s_\gamma s_\delta s_\varphi - s_\alpha c_\beta c_\gamma s_\delta s_\varphi + c_\alpha c_\delta s_\varphi \\
o_y &= s_\alpha s_\beta s_\gamma c_\delta c_\varepsilon s_\varphi - s_\alpha c_\beta c_\gamma c_\delta c_\varepsilon s_\varphi - c_\alpha s_\delta c_\varepsilon s_\varphi + s_\alpha c_\beta s_\gamma s_\varepsilon s_\varphi + s_\alpha s_\beta c_\gamma s_\varepsilon s_\varphi + \\
&\quad + s_\alpha s_\beta s_\gamma s_\delta c_\varphi - s_\alpha c_\beta c_\gamma s_\delta c_\varphi + c_\alpha c_\delta c_\varphi \\
a_y &= s_\alpha c_\beta c_\gamma c_\delta s_\varepsilon - s_\alpha s_\beta s_\gamma c_\delta s_\varepsilon + c_\alpha s_\delta s_\varepsilon + s_\alpha c_\beta s_\gamma c_\varepsilon + s_\alpha s_\beta c_\gamma c_\varepsilon
\end{aligned}$$

$$\begin{aligned}
n_z &= s_\beta s_\gamma s_\varepsilon c_\varphi - s_\beta c_\gamma c_\delta c_\varepsilon c_\varphi - c_\beta s_\gamma c_\delta c_\varepsilon c_\varphi - c_\beta c_\gamma s_\varepsilon c_\varphi + s_\beta c_\gamma s_\delta s_\varphi + c_\beta s_\gamma s_\delta s_\varphi \\
o_z &= s_\beta c_\gamma c_\delta c_\varepsilon s_\varphi + c_\beta s_\gamma c_\delta c_\varepsilon s_\varphi - s_\beta s_\gamma s_\varepsilon s_\varphi + c_\beta c_\gamma s_\varepsilon s_\varphi + s_\beta c_\gamma s_\delta c_\varphi + c_\beta s_\gamma s_\delta c_\varphi \\
a_z &= c_\beta c_\gamma c_\varepsilon - s_\beta c_\gamma c_\delta s_\varepsilon - c_\beta s_\gamma c_\delta s_\varepsilon - s_\beta s_\gamma c_\varepsilon
\end{aligned} \tag{14}$$

Matrix components x, y and z are the end point coordinates of the industrial robot. Rotation matrix is assembled from vectors n, o and a. The rotation of the coordination system is calculated from this rotation matrix. Presented below are the general rotation matrixes (15) around all axes:

$$R_C = \begin{pmatrix} \cos\tau & -\sin\tau & 0 \\ \sin\tau & \cos\tau & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\rho & 0 & \sin\rho \\ 0 & 1 & 0 \\ -\sin\rho & 0 & \cos\rho \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix} =$$
$$\begin{pmatrix} \cos\tau.\cos\rho & \cos\tau.\sin\rho.\sin\omega - \sin\tau.\cos\omega & \cos\tau.\sin\rho.\cos\omega + \sin\tau.\sin\omega \\ \sin\tau.\cos\rho & \sin\tau.\sin\rho.\sin\omega + \cos\tau.\cos\omega & \sin\tau.\sin\rho.\cos\omega - \cos\tau.\sin\omega \\ -\sin\rho & \cos\rho.\sin\omega & \cos\rho.\cos\omega \end{pmatrix} \tag{15}$$

ω – Rotation around axis x
ρ – Rotation around axis y
τ – Rotation around axis z

All axes rotations are deduced from equation (15):
Rotation around axis y, angle ρ:

$$n_z = -\sin\rho \tag{16}$$

$$\rho = \arcsin(-n_z) \tag{17}$$

Rotation around axis z, angle τ:

$$n_y = \sin\tau.\cos\rho \tag{18}$$

$$\tau = \arcsin\left(\frac{n_y}{\cos(\arcsin(-n_z))}\right) \tag{19}$$

Rotation around axis x, angle ω:

$$o_z = \cos\rho.\sin\omega \tag{20}$$

$$\omega = \arcsin\left(\frac{o_z}{\cos(\arcsin(-n_z))}\right) \tag{21}$$

4 Inverse Kinematics

Robot inverse kinematics is calculated by algorithm which we have proposed. This algorithm uses mathematical model of the robot's direct kinematics and Newton approximation method with the use of Jacoby matrix.

Inputs of the algorithm are an initial state, desired position and parameter δ. The initial state is the actual rotation state of the joints. The desired position consists of coordinates and axes rotations which the robotic arm has to be in. Parameter δ is described below. The shift has to be defined as zero before the start of the main loop of the algorithm.

The algorithm includes two functions. These functions are not shown in the flowchart for lack of space in this paper. These functions are AA (angle adjust) and DK (direct kinematics). AA function adjusts angles in the range from -360° to 360°. DK inputs are angles of the joints rotation and the output is vector with coordinates and axes rotation. The algorithms main loop includes two very important calculates: Jacoby matrix and shift. The classic formula for Jacoby matrix is (22):

$$J = \begin{pmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \cdots & \frac{df_1}{dx_n} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} & \cdots & \frac{df_2}{dx_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{df_m}{dx_1} & \frac{df_m}{dx_2} & \cdots & \frac{df_m}{dx_n} \end{pmatrix} \tag{22}$$

We modified the formula (22) to a discrete form. The discretization of derivation is different. Discrete Jacoby matrix consist of vectors which represent a difference between temporary positions and positions after minimal modification (δ) in the angle of one robotic joint. If the robot has 6 degrees of freedom and 6 joints, then the resultant Jacoby matrix has size 6x6. Discrete Jacoby matrix (Function DK is used in the formula for simplification):

$$J = \left[(DK(\alpha,\beta,\gamma,\delta,\varepsilon,\varphi))' \quad (DK(\alpha,\beta,\gamma,\delta,\varepsilon,\varphi))' \quad \cdots \quad (DK(\alpha+\Delta,\beta,\gamma,\delta,\varepsilon,\varphi))'\right] - \\ -\left[(DK(\alpha+\Delta,\beta,\gamma,\delta,\varepsilon,\varphi))' \quad (DK(\alpha,\beta+\Delta,\gamma,\delta,\varepsilon,\varphi))' \quad \cdots \quad (DK(\alpha,\beta,\gamma,\delta,\varepsilon,\varphi+\Delta))'\right] \tag{23}$$

Inverse kinematics algorithm:

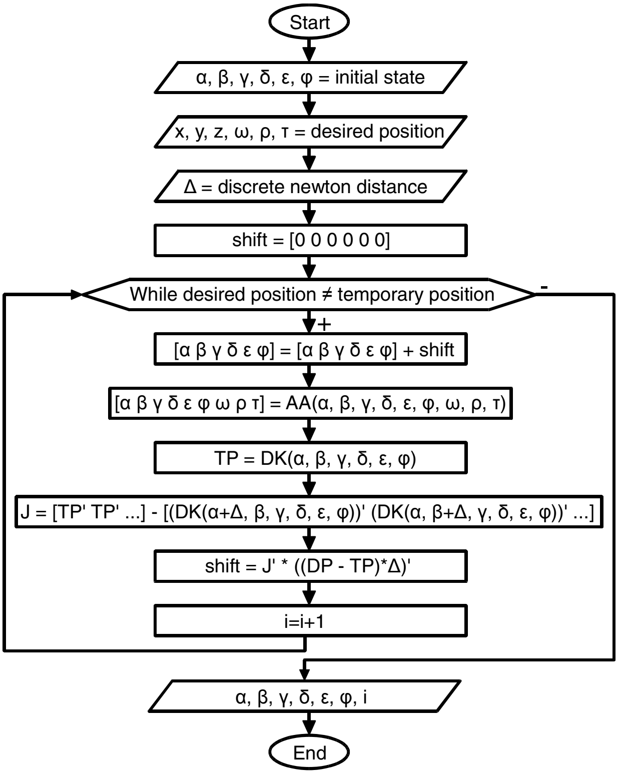


Fig. 10. Inverse kinematics algorithm

Shift is calculated through the basic formula:

$$dq = J^{-1}dp \tag{24}$$

Discretization of this formula (DP – vector of desired position, TP – vector of temporary position, s - shift):

$$s = J^{-1}(\Delta(DP - TP))' \tag{25}$$

The algorithm output is the angles vector of joint rotation and the number of loop iterations. It is important to point out that the robot's movement begins after the algorithm finishes; robot's movement never begins while an algorithm is running. Temporary position is not a real position, only an auxiliary position necessary for calculation.

5 Experimental Verification

These results were verified on the real industrial robot. Direct kinematics is verified by entering a random motor (joint) rotation to the real model and after the completion of the operation the actual position of the robot's endpoint is read from the robot controller. The same random joint rotation is calculated by composite homogeneous transformation matrix in MATLAB. These results are compared in Table 1:

Legend to table:

α, β, γ, δ, ε, ϕ: joint rotation of industrial robot

X, Y, Z: real position (coordinate) of robot's endpoint
A, B, C: real rotation of coordination system of robot's endpoint
x, y, z: calculated position (coordinate) of robot's endpoint
ω, ρ, τ: calculated rotation of coordination system of robot's endpoint

Table 1. Verification of calculated direct kinematics – selected measurement

č.	α[deg.]	β[deg.]	γ[deg.]	δ[deg.]	ε[deg.]	ϕ[deg.]
	X[mm]	Y[mm]	Z[mm]	A[deg.]	B[deg.]	C[deg.]
	x[mm]	y[mm]	z[mm]	ω[deg.]	ρ[deg.]	τ[deg.]
1.	87,47	-45,92	98,93	85,70	58,00	121,00
	-56,81	55,22	676,16	11,46	-74,13	-43,06
	-56,8141	55,2198	676,1553	11,4617	-74,127	-43,057
2.	225	110	70	-90	-90	300
	-138,68	-237,68	-53,66	-90	30	-45
	-138,6844	-237,6793	-53.6646	-90	30	-45
3.	21,8	-1,32	102,84	0	78,48	21,8
	250	100	450	180	0	180
	249,9950	99,9951	450,0048	180	0	180

On Table 1 it can be seen that the direct kinematics of an industrial robot MELFA RV-2SDB is calculated correctly. There is more verification but the results are same; this is the reason why the results are not in the table. Inverse kinematics is verified, too with excellent results (calculated joints rotation were the same as the real joints rotation).

6 Conclusion

This paper describes the process of creating a mathematical model of robotic arms. This mathematical model refers to the direct and inverse kinematics model. The process of creating the direct kinematics model was tested on 4 robotic arms (Mitsubishi MELFA RV-2SDB – model DCAI, Robkovia – model DCAI, SEF – Model DCAI and OWI 535 Robotic Arm). The algorithm of inverse kinematics was tested on 2 robotic arms (Mitsubishi MELFA RV-2SDB and SEF). We use the Newton approximation method for the inverse kinematics calculation but there are many other methods, for example Taylor expansion of the transformation matrix, Analytic solution, BFS (Broyden, Fletcher, Shanno) method and Vector method of inverse transformation. We use the mentioned method because we considered this method reliable and result needs maximally 8 iterations.

Acknowledgment. This work has been supported by the Scientific Grant Agency of Slovak Republic under project Vega No.1/0286/11 Dynamic Hybrid Architectures of the Multiagent Network Control Systems – 50%. This work has been supported by the Research and Development Operational Program for project: University Science Park Technicom for innovative applications with knowledge technology support, ITMS code 26220220182, co-financed by the ERDF – 50%.

References

1. Krokavec, D.: Data processing in robotics (Spracovanie údajov v robotike), Košice (1985) ISBN 85-629-85
2. Mostýn, V., Krys, V.: Mechatronics of industrial robots (Mechatronika průmyslových robotů), Ostrava (2012) ISBN 978-80-248-2610-3
3. Božek, P., Barborák, O., Naščák, Ľ., Štollmann, V.: Specialized robotic systems (Špecializované robotické systémy), Bratislava (2011) ISBN 978-80-904766-8-4
4. Jadlovský, J., Papcun, P.: Optimizing industry robot for maximum speed with high accuracy. In: Procedia Engineering, Zemplínska Šírava, vol. 48, pp. 533–542 (2012) ISSN 1877-7058
5. Papcun, P., Čopík, M., Ilkovič, J.: Robot control integrated in flexible production system (Riadenie robota integrovaného v pružnom výrobnom systéme). In: Electro Scope, vol. 2012(2), Plzeň, pp. 1–9 (2012) ISSN 1802-4564