Chapter

Kinematics of Serial Manipulators

Ivan Virgala, Michal Kelemen and Erik Prada

Abstract

This book chapter deals with kinematic modeling of serial robot manipulators (open-chain multibody systems) with focus on forward as well as inverse kinematic model. At first, the chapter describes basic important definitions in the area of manipulators kinematics. Subsequently, the rigid body motion is presented and basic mathematical apparatus is introduced. Based on rigid body conventions, the forward kinematic model is established including one of the most used approaches in robot kinematics, namely the Denavit-Hartenberg convention. The last section of the chapter analyzes inverse kinematic modeling including analytical, geometrical, and numerical solutions. The chapter offers several examples of serial manipulators with its mathematical solution.

Keywords: algorithm, inverse kinematics, Jacobian, manipulator, optimization, redundant, robot

1. Introduction and basic definitions

In the following sections, this chapter will deal with direct and inverse kinematics of open-chain multibody systems consisting of rigid bodies. The whole problematics is analyzed from the view of robotics. Each manipulator or mechanism investigated in this chapter will be of serial kinematic structure (open chain).

Open-chain multibody systems are mechanically constructed by connecting a set of bodies, called links, by means of various types of joints. In general, the joints can be passive or active. The joints, which are moved by actuators, are active joints.

In general, from the view of robotics, there are two tasks in kinematics:

- Forward kinematics—the forward kinematics problem represents relationship between individual joints of investigated robot and end-effector.

- Inverse kinematics—the problem of inverse kinematics is as follows: given a desired configuration of end-effector of robot, find the joint angles that achieve that configuration.

Before these terms are explained and demonstrated by some study cases, we have to mention the basic definitions, necessary for the further analyses.

Degrees of freedom (DOF): is the smallest number of coordinates needed to represent the robot configuration. Thus, the number of DOF equals to the dimension of configuration space.

Joint space: Let us define all the joint variables in a vector $\mathbf{q} = [q_1, q_2, \ldots, q_n]^T \in \mathbb{Q} \subset \mathbb{R}^N$. The set $\mathbb{Q}$ we call the so-called joint space and it contains all the possible values, which joint variables may acquire.

Workspace: Workspace is a subset of the Euclidean space $\mathbb{E}$, in which the robot executes its tasks. From the view of robotics, workspace is the set of all the points that mechanism may reach in Euclidean space $\mathbb{E}$ by end-effector. The workspace can be categorized as follows [1, 2]:

Maximal workspace—it is defined as locations that can be reached by end-effector at least with one orientation.

Inclusive-orientation workspace—it is defined as locations that can be reached by end-effector with at least one orientation among a range of orientations (maximal workspace is particular case).

Constant-orientation workspace—it is defined as location that can be reached by the end-effector with fixed orientation of joints.

Total-orientation workspace—it is defined as location that can be reached by the end-effector with any orientation.

Dexterity workspace—it is defined as location that can be reached by the end-effector with any orientation and without kinematic singularities.

Task space–space of positions and orientations of the end-effector frame. The workspace is a subset of task space that the end-effector frame can reach [3].

2. Rigid body motion

Rigid motion of an object is a motion that preserves distance between points [4]. Rigid body is a set of particles such that the distance between any two particles remains constant in time, regardless of any motions of the body or forces exerted on the body. If we consider **p** and **q** as two points on rigid body, while rigid body moves, **p** and **q** must satisfy $\|\mathbf{p}(t) - \mathbf{q}(t)\| = \|\mathbf{p}(0) - \mathbf{q}(0)\| = constant$, see **Figure 1**.

Let us consider an object, described as a subset O of $\mathbb{R}^3$. Then a motion of object (rigid body) is represented by mapping $f(t) : O \rightarrow \mathbb{R}^3$. This mapping describes how the points of this object move as a function of time, relative to some fixed coordinate system.

Let the inertial reference frame be $O = \{x_r, y_r, z_r\}$ and $\mathbf{i}_r, \mathbf{j}_r, \mathbf{k}_r$ represent unit vectors of the reference frame. The vector **p** can be expressed with respect to inertial reference frame $O = \{x_r, y_r, z_r\}$ by the following equation

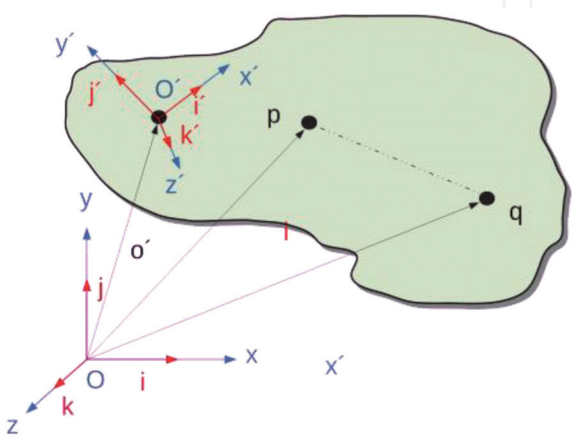

Figure 1.
Rigid body.

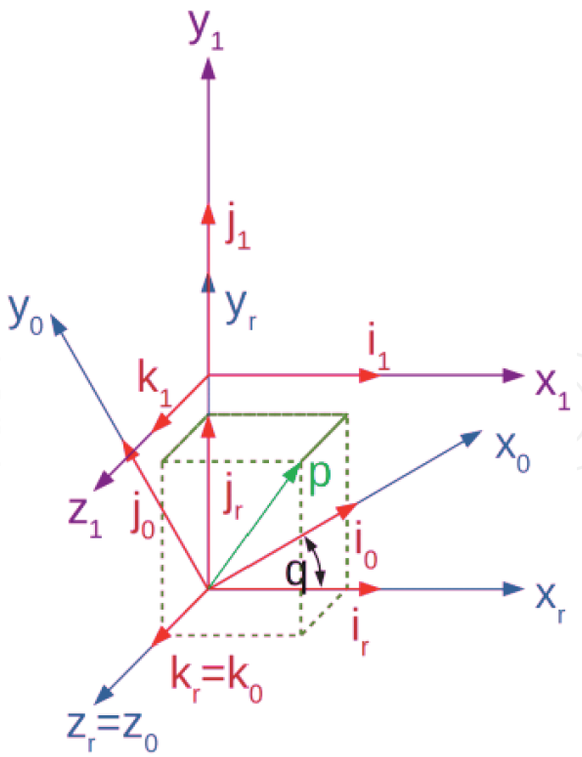

Figure 2.
Rotation of coordinate system.

$$\mathbf{p} = p_{xr}\mathbf{i}_r + p_{yr}\mathbf{j}_r + p_{zr}\mathbf{k}_r \tag{1}$$

where $\mathbf{p} = \left[p_x, p_y, p_z\right]^T \in \mathbb{R}^3$. Coordinates of vector $\mathbf{p}$ can be also expressed as its projections in directions of individual unit vectors as scalar product. In order to find the relation, the vector $\mathbf{p}$ needs to be expressed in coordinates $O_1 = \{x_1, y_1, z_1\}$

$$\begin{aligned} p_{xr} &= \mathbf{i}_r\,\mathbf{p} = \mathbf{i}_r p_{x1}\mathbf{i}_1 + \mathbf{i}_r p_{y1}\mathbf{j}_1 + \mathbf{i}_r p_{z1}\mathbf{k}_1 \\ p_{yr} &= \mathbf{j}_r\,\mathbf{p} = \mathbf{j}_r p_{x1}\mathbf{i}_1 + \mathbf{j}_r p_{y1}\mathbf{j}_1 + \mathbf{j}_r p_{z1}\mathbf{k}_1 \\ p_{zr} &= \mathbf{k}_r\,\mathbf{p} = \mathbf{k}_r p_{x1}\mathbf{i}_1 + \mathbf{k}_r p_{y1}\mathbf{j}_1 + \mathbf{k}_r p_{z1}\mathbf{k}_1 \end{aligned} \tag{2}$$

which can be rewritten in matrix form

$$\begin{bmatrix} p_{xr} \\ p_{yr} \\ p_{zr} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_r\mathbf{i}_1 & \mathbf{i}_r\mathbf{j}_1 & \mathbf{i}_r\mathbf{k}_1 \\ \mathbf{j}_r\mathbf{i}_1 & \mathbf{j}_r\mathbf{j}_1 & \mathbf{j}_r\mathbf{k}_1 \\ \mathbf{k}_r\mathbf{i}_1 & \mathbf{k}_r\mathbf{j}_1 & \mathbf{k}_r\mathbf{k}_1 \end{bmatrix} \begin{bmatrix} p_{x1} \\ p_{y1} \\ p_{z1} \end{bmatrix} \tag{3}$$

that is $\mathbf{p}_b = \mathbf{R}_{r1}\mathbf{p}_1$. The meaning of this term is as follows. Coordinates of the vector $\mathbf{p}$ expressed in $O_1 = \{x_1, y_1, z_1\}$ are computed to $O = \{x_r, y_r, z_r\}$ so that they are left multiplied by transformation matrix $\mathbf{R}_{r1}$.

As can be seen in **Figure 2**, coordinate system x_0, y_0, z_0 is rotated with respect to coordinate system x_r, y_r, z_r by angle q around the axis z_r. By consideration of previous equations; and by consideration of the facts that scalar product of two perpendicular vectors equals zero, scalar product of two parallel unit vectors is one, and scalar product of concurrent unit vectors is $\cos\alpha$; and by assuming that $\cos\left(\frac{\pi}{2} \pm \alpha\right) = \mp\sin\alpha$, that is

$$\mathbf{i}^T\mathbf{i} = 1, \mathbf{j}^T\mathbf{j} = 1, \mathbf{k}^T\mathbf{k} = 1$$
$$\mathbf{i}^T\mathbf{j} = 0, \mathbf{j}^T\mathbf{k} = 0, \mathbf{k}^T\mathbf{i} = 0$$

one can obtain the following rotation matrix

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{4}$$

where $\mathbf{R}_x$ is a rotation matrix for rotation around the x-axis by angle α. Subsequently, rotation matrices can be also be expressed for rotation around y-axis and z-axis

$$\mathbf{R}_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{5}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Since rotation matrix $\mathbf{R}$ is an orthogonal matrix, for this reason

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}_3 \tag{7}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix. Considering the case when there is displacement of local coordinate system and at the same time also its rotation, it would be expressed as

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \mathbf{R}_{axis,angle} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{8}$$

Eq. (8) represents a system of three equations, which will be extended by fourth equation 1 = 0 + 0 + 0 + 1, which is

$$\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R}_{axis,angle} & \vdots & p_x \\ & & \vdots & p_y \\ \cdots & \cdots & \vdots & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{9}$$

3. Forward kinematics

The forward kinematic model determines the position and orientation of the end-effector relating to base frame of the mechanism or to global coordinate system (GCS).

3.1 Open kinematic chain

We will focus on robots, which contain a set of links connected together by joints. The joints are usually revolute or prismatic or they can be more complex,

such as socket joint or ball joint [5]. Within this chapter will be considered only revolute and prismatic joints, which have only a single degree-of-freedom motion. Let us consider a mechanism with N links connected together by $N-1$ joints. The i-th joint connects link $i-1$ to link i. The number of the joints starts with 1 and ends with $N-1$. The next consideration for the following mathematical model is that the first link is connected to the base fixed to inertial reference frame, while the last link is free and able to move.

The i-th joint is associated with joint variable q_i, while q_i may contain θ_i and d_i for revolute and prismatic joints, respectively. The local coordinate frame is attached to each link, so to i-th link is attached to I_i frame, $I_i = \{O_i, x_i, y_i, z_i\}$. When a mechanism performs any motion in its workspace, the coordinates of each point on i-th link are constant with respect to their coordinate frame $I_i = \{O_i, x_i, y_i, z_i\}$.

Let $\mathbf{A}_i$ be a homogeneous transformation matrix, which holds position and orientation of frame $I_i = \{O_i, x_i, y_i, z_i\}$ with respect to $I_{i-1} = \{O_{i-1}, x_{i-1}, y_{i-1}, z_{i-1}\}$. It should be noticed that values of matrix $\mathbf{A}_i$ are not constant, but they change with changing configuration of the mechanism. In general, a homogeneous transformation matrix expressing the position and orientation of $I_j = \{O_j, x_j, y_j, z_j\}$ with respect to $I_i = \{O_i, x_i, y_i, z_i\}$ is called a transformation matrix ${}^{i}\mathbf{T}_j$. We can also define the following matrix

$$\mathbf{H} = \begin{bmatrix} {}^{0}\mathbf{R}_n & {}^{0}\mathbf{o}_n \\ 0 & 1 \end{bmatrix} \tag{10}$$

where ${}^{0}\mathbf{R}_n$ is a 3×3 rotation matrix with and ${}^{0}\mathbf{o}_n$ is a 3×1 vector expressing position and orientation of end-effector (the last point of mechanism) with respect to inertial reference frame (base of mechanism). Eq. (10) can then be written as

$$\mathbf{H} = {}^{0}\mathbf{T}_n = \prod_{i=1}^{N} \mathbf{A}_i \tag{11}$$

while $\mathbf{A}_i$ equals

$$\mathbf{A}_i = \begin{bmatrix} {}^{i-1}\mathbf{R}_i & {}^{i-1}\mathbf{o}_i \\ 0 & 1 \end{bmatrix} \tag{12}$$

3.2 Denavit–Hartenberg convention

For the computation of forward kinematics for open-chain robot according to Eq. (11), a general approach was derived in order to determine the relative position and orientation of two consecutive links. This approach determines two frames attached to two links (rigid bodies) and computes the coordinate transformations between them [6].

For utilization of the Denavit-Hartenberg convention, some rules need to be observed. Let us consider **Figure 3**. Let axis i represent the axis connecting link $i-1$ and link $i+1$. In order to define link frame i, the procedure is as follows. First of all, the axis z_i and axis z_{i-1} are chosen. Next, origin O_i is located at the intersection of axis z_i with the common normal to axes z_i and z_{i-1}. By this step be get points O_i and O_i'. The common normal of these two axes is a minimum distance between them. Subsequently, the axis x_i is chosen along the common normal to axes z_{i-1} and z_i in

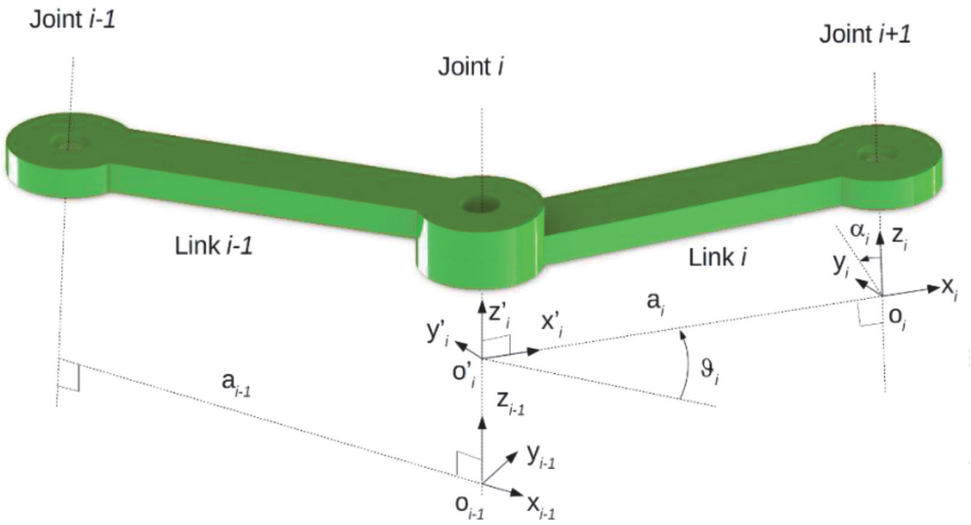

Figure 3.
Denavit-Hartenberg approach.

the direction from joint i to the joint $i+1$. In the last step, axis y_i is chosen so as to complete a right-handed frame.

After these steps, the link frames have been established and now the position and orientation of frame i with respect to frame $i-1$ can be determined by following DH parameters [7]:

- a_i: distance between the points O_i and O_i'
- d_i: distance between O_{i-1} and O_i' along the axis z_{i-1}
- α_i: angle between the axes z_{i-1} and z_i about axis x_i (positive direction—counter-clockwise rotation)
- ϑ_i: angle between axes x_i and x_{i-1} (positive direction—counter-clockwise rotation)

It should be also noted that parameters a_i and α_i are always constant, because they depend on the geometric aspect of mechanism. Considering the two other parameters d_i and ϑ_i, depending on the joint type, one is constant and other one may change as follows:

- Revolute joint: ϑ_i is the joint variable and d_i is constant
- Prismatic joint: d_i is the joint variable and ϑ_i is constant

In general, six parameters are necessary in order to describe the position and orientation of a rigid body in the 3D space. Based on previously mentioned facts, we can say about DH convention that only four parameters are required by assuming that the axis x_i intersects z_{i-1}, and that axis x_i is perpendicular to z_{i-1}.

3.2.1 Example of forward kinematics using the Denavit-Hartenberg convention

Let us consider some kind of industrial robot, namely SCARA (Selective Compliance Assembly Robot Arm) robot, which has RRP structure. Its kinematic structure is shown in **Figure 4**.

Considering the basic principles of the Denavit-Hartenberg convention introduced in the previous section, we are able to introduce D-H parameters, see **Table 1**.

Based on DH parameters, which are obvious from **Figure 4**, particular homogeneous transformation matrices can be established.

$$
{}^{0}\mathbf{A}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\,(q_1) & -sin\,(q_1) & 0 & 0 \\ sin\,(q_1) & cos\,(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}
$$

$$
{}^{0}\mathbf{A}_1 = \begin{bmatrix} cos\,(q_1) & -sin\,(q_1) & 0 & L_1 cos\,(q_1) \\ sin\,(q_1) & cos\,(q_1) & 0 & L_1 sin\,(q_1) \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{1}\mathbf{A}_2 = \begin{bmatrix} cos\,(q_2) & -sin\,(q_2) & 0 & 0 \\ sin\,(q_2) & cos\,(q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}
$$

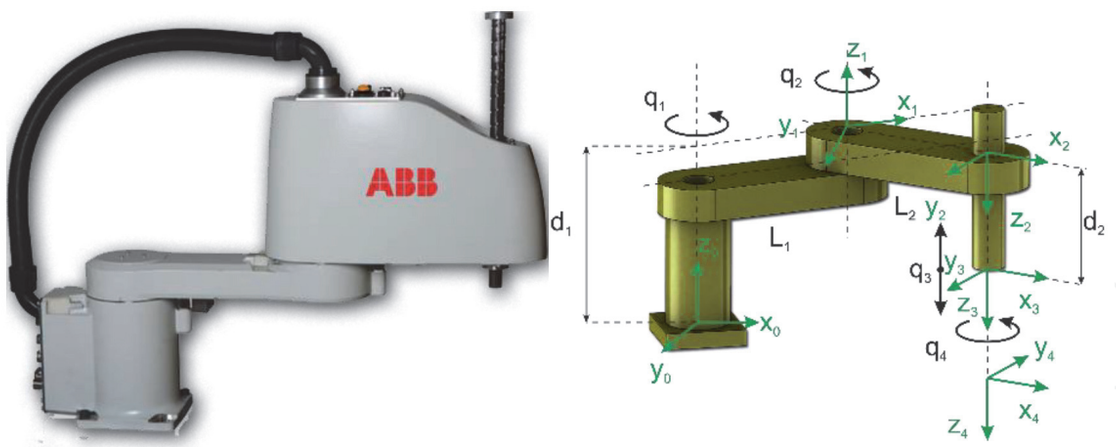

Figure 4.
SCARA robot.

Link	a_i	α_i	d_i	ϑ_i
1	L_1	0	d_1	q_1
2	L_2	0	0	q_2
3	0	π	$d_2 + q_3$	0
4	0	0	0	q_4

Table 1.
Denavit-Hartenberg parameters.

$$
{}^{1}\mathbf{A}_2 = \begin{bmatrix} cos\,(q_2) & -sin\,(q_2) & 0 & L_2\,cos\,(q_2) \\ sin\,(q_2) & cos\,(q_2) & 0 & L_2\,sin\,(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{2}\mathbf{A}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\,(\pi) & -sin\,(\pi) & 0 \\ 0 & sin\,(\pi) & cos\,(\pi) & d_2 + q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}
$$

$$
{}^{3}\mathbf{A}_4 = \begin{bmatrix} cos\,(q_4) & -sin\,(q_4) & 0 & 0 \\ sin\,(q_4) & cos\,(q_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}
$$

So, the final transformation matrix is

$$
{}^{0}\mathbf{T}_4 = {}^{0}\mathbf{A}_2\,{}^{1}\mathbf{A}_2\,{}^{2}\mathbf{A}_3\,{}^{3}\mathbf{A}_4 \tag{17}
$$

$$
{}^{0}\mathbf{T}_4 = \begin{bmatrix} & & & p_x \\ & \mathbf{R} & & p_y \\ & & & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}
$$

By vector ${}^{0}\mathbf{o}_4 = \begin{bmatrix} p_x p_y p_z \end{bmatrix}^T$ is defined position of end-effector of SCARA manipulator with respect to its base inertial reference frame.

4. Inverse kinematics

The solution exists only if the given end-effector position and orientation are in dexterous workspace of the solved mechanism. While the forward kinematic model is expressed as

$$
\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{19}
$$

where $\mathbf{f}$ is a function defined between joint space $\mathbb{R}^n$ and workspace $\mathbb{R}^m$, which maps the joint position variables $\mathbf{q} \in \mathbb{R}^n$ to the position/orientation of the end-effector of mechanism, the inverse kinematic model is based on

$$
\mathbf{q} = \mathbf{f}^{-1}(\mathbf{x}) \tag{20}
$$

where $\mathbf{q} \in \mathbb{R}^n$ and $\mathbf{x} \in \mathbb{R}^m$. In the case of the forward kinematic model, end-effector position and orientation are computed for various kinds of mechanisms like manipulators, in a unique manner, for example, by above-mentioned transformation matrices. The inverse kinematic problem is more complex and finding the solution could be in many cases very complicated. While forward kinematics has a closed-form solution, an inverse kinematics in most cases does not have a closed-form solution. A forward kinematic model has a unique solution, while an inverse

kinematic model may have multiple solutions or infinite number of solutions, especially for kinematically redundant mechanisms. In order to obtain a closed-form solution, there are two main approaches, namely algebraic approach and geometric approach.

4.1 Closed-form solution of inverse kinematics

Let us consider a two-link mechanism moving in the 2D plane, see **Figure 5**.

Considering the forward kinematic model, while the angles of joints ϑ_1 and ϑ_2 are given, the aim is to find the position of end-effector $\mathbf{x}_E = [x\, y]^T \in \mathbb{R}^m$. The forward kinematic model can be easily determined by the following equations

$$x = l_1 \cos\vartheta_1 + l_2 \cos(\vartheta_1 + \vartheta_2) \tag{21}$$

$$y = l_1 \sin\vartheta_1 + l_2 \sin(\vartheta_1 + \vartheta_2) \tag{22}$$

Now, the inverse kinematic problem is to find angles ϑ_1 and ϑ_2, while the end-effector position x and y are given by vector $\mathbf{x}_E = [x\, y]^T \in \mathbb{R}^m$.

$$c = x^2 + y^2 \tag{23}$$

$$\alpha = atan2(y, x) \tag{24}$$

$$\vartheta_2 = \pm arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \tag{25}$$

$$\beta = arccos\left(\frac{L_1^2 - L_2^2 + c^2}{2L_1c}\right) \tag{26}$$

$$\vartheta_1 = \alpha - \beta \tag{27}$$

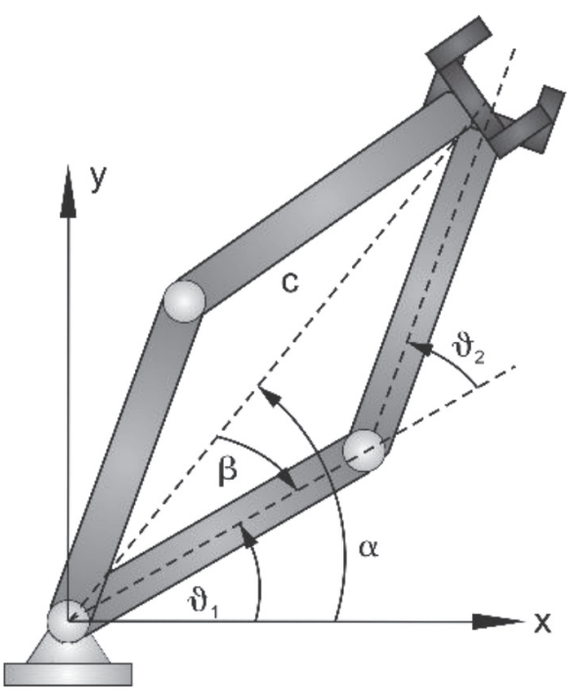

Figure 5.
Inverse kinematics solution for two-link mechanism.

As can be seen in **Figure 5**, the presented configuration of the mechanism has two solutions in inverse kinematics. These two solutions are based on signum in Eqs. (25) and (27).

5. Differential kinematics

5.1 Analytical Jacobian

In order to describe the relation between joint angles and end-effector configuration, often the relation between the joint and end-effector velocities is used. Let us consider a set of coordinates $\mathbf{x} \in \mathbb{R}^m$, their velocity is $\dot{\mathbf{x}} = d\mathbf{x}/dt \in \mathbb{R}^m$. Then, we can apply Eq. (19). Then, one obtains

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \frac{d\mathbf{q}}{dt} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{28}$$

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial q_1} & \cdots & \frac{\partial x_m}{\partial q_n} \end{bmatrix} \tag{29}$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{m \times n}$ is the analytical Jacobian matrix, which is very often used in kinematics and dynamics of robotic systems. Jacobian reflects differences between joint and configurations space of the investigated mechanism. In robotics, Jacobian is often used for several purposes such as for the definition of the relation between joint and configuration space, definition of the relation between forces/torques between spaces, the study of kinematic singularities, the definition of numerical solution for inverse kinematic problem, and the study of manipulability properties. We can look at Jacobian from a different perspective. Particular Jacobian columns represent the influence of i-th joint on the end-effector velocity.

The following example will demonstrate the derivation of analytical Jacobian for a three-link mechanism (**Figure 6**).

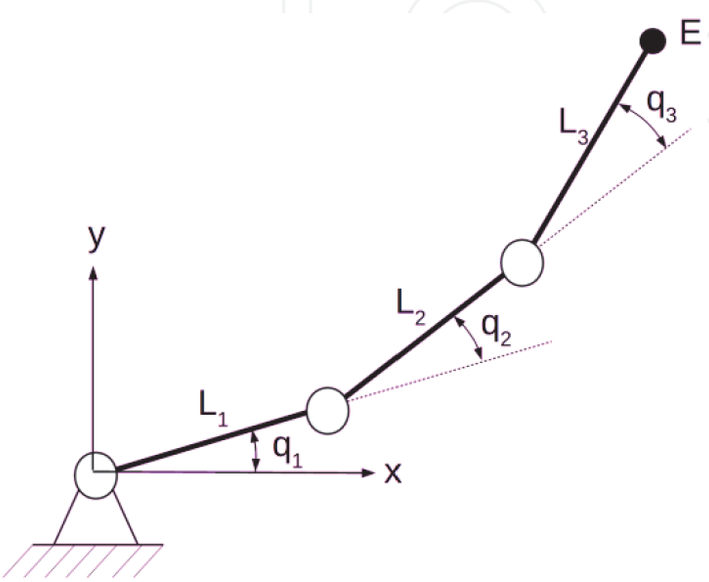

Figure 6.
Three-link mechanism.

In order to utilize Eq. (29), we need to define position of point $\mathbf{E} = \left[x_E y_E\right]^T \in \mathbb{R}^2$.

$$x_E = L_1 \cos q_1 + L_2 \cos\left(q_1 + q_2\right) + L_3 \cos\left(q_1 + q_2 + q_3\right) \tag{30}$$

$$y_E = L_1 \sin q_1 + L_2 \sin\left(q_1 + q_2\right) + L_3 \sin\left(q_1 + q_2 + q_3\right) \tag{31}$$

Assuming Eqs. (30) and (31), Jacobian according to Eq. (29) will be matrix $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{2\times 3}$

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial x_E}{\partial q_1} & \frac{\partial x_E}{\partial q_2} & \frac{\partial x_E}{\partial q_3} \\ \frac{\partial y_E}{\partial q_1} & \frac{\partial y_E}{\partial q_2} & \frac{\partial y_E}{\partial q_3} \end{bmatrix} \tag{32}$$

where the elements of the Jacobian matrix are

$$\frac{\partial x_E}{\partial q_1} = -L_1 sin\left(q_1\right) - L_2 sin\left(q_1 + q_2\right) - L_3 sin\left(q_1 + q_2 + q_3\right)$$

$$\frac{\partial x_E}{\partial q_2} = -L_2 sin\left(q_1 + q_2\right) - L_3 sin\left(q_1 + q_2 + q_3\right)$$

$$\frac{\partial x_E}{\partial q_3} = -L_3 sin\left(q_1 + q_2 + q_3\right)$$

$$\frac{\partial y_E}{\partial q_1} = L_1 cos\left(q_1\right) + L_2 cos\left(q_1 + q_2\right) + L_3 cos\left(q_1 + q_2 + q_3\right)$$

$$\frac{\partial y_E}{\partial q_2} = L_2 cos\left(q_1 + q_2\right) + L_3 cos\left(q_1 + q_2 + q_3\right)$$

$$\frac{\partial y_E}{\partial q_3} = L_3 cos\left(q_1 + q_2 + q_3\right)$$

5.2 Geometric Jacobian

Besides analytically expressing the Jacobian, we can express it by a geometric approach. To establish function $\mathbf{f}(\mathbf{q})$ in closed-form, a symbolic formalism is necessary, which could be difficult from the view of implementation. For this reason, a different way of Jacobian expression, the so-called geometric Jacobian, was developed. The geometric Jacobian can be obtained by consideration of rotational velocity vector $\boldsymbol{\omega}$. Let us consider link according to **Figure 6**. The Jacobian can be expressed as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_v \\ \mathbf{J}_\omega \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{v1} & \dots & \mathbf{J}_{vn} \\ \mathbf{J}_{\omega 1} & \dots & \mathbf{J}_{\omega n} \end{bmatrix} \tag{33}$$

The first term expresses the effect of $\dot{q}_1$ on linear velocity $\mathbf{v}$ and the second term expresses the effect on the rotational velocity $\boldsymbol{\omega}$. Thus,

$$\begin{aligned} \mathbf{v} &= \mathbf{J}_{v1}\dot{q}_1 + \dots + \mathbf{J}_{vn}\dot{q}_n \\ \boldsymbol{\omega} &= \mathbf{J}_{\omega 1}\dot{q}_1 + \dots + \mathbf{J}_{\omega n}\dot{q}_n \end{aligned} \tag{34}$$

That is, the analytical Jacobian differs from the geometrical Jacobian for the rotational part. Considering the revolute joint, the i-th column of Jacobian can be computed as

$$\begin{bmatrix} \mathbf{J}_{vi} \\ \mathbf{J}_{\omega i} \end{bmatrix} = \begin{bmatrix} {}^0\mathbf{z}_{i-1} \times \left({}^0\mathbf{p}_n - {}^0\mathbf{p}_{i-1}\right) \\ {}^0\mathbf{z}_{i-1} \end{bmatrix} \tag{35}$$

For prismatic joint, the i-th column of Jacobian can be computed as

$$\begin{bmatrix} \mathbf{J}_{vi} \\ \mathbf{J}_{\omega i} \end{bmatrix} = \begin{bmatrix} {}^0\mathbf{z}_{i-1} \\ \mathbf{0} \end{bmatrix} \tag{36}$$

where ${}^0\mathbf{p}_n$ is the end-effector position defined in transformation matrix ${}^0\mathbf{T}_n$ defined in the previous section. Next, ${}^0\mathbf{p}_{i-1}$ is the position of frame $I_{i-1} = \{O_{i-1}, x_{i-1}, y_{i-1}, z_{i-1}\}$, defined in transformation matrix ${}^0\mathbf{T}_{i-1}$. Finally, ${}^0\mathbf{z}_{i-1}$ is the third column of rotation matrix ${}^0\mathbf{R}_{i-1}$, while ${}^0\mathbf{R}_{i-1} = {}^0\mathbf{R}_1(q_1){}^1\mathbf{R}_2(q_2) \ldots {}^{i-2}\mathbf{R}_{i-1}(q_{i-1})$.

The following example will demonstrate the derivation of geometric Jacobian for a two-link mechanism (**Figure 7**).

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{z}_0 \times \left(\mathbf{p}_2 - \mathbf{p}_0\right) & \mathbf{z}_1 \times \left(\mathbf{p}_2 - \mathbf{p}_1\right) \\ \mathbf{z}_0 & \mathbf{z}_1 \end{bmatrix} \tag{37}$$

where $\mathbf{p}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$, $\mathbf{p}_1 = \begin{bmatrix} L_1 \cos q_1 \\ L_1 \sin q_1 \\ 0 \end{bmatrix}$, $\mathbf{p}_2 = \begin{bmatrix} L_1 \cos q_1 + L_2 \cos\left(q_1 + q_2\right) \\ L_1 \sin q_1 + L_2 \sin\left(q_1 + q_2\right) \\ 0 \end{bmatrix}$ and rotational axes are $\mathbf{z}_0 = \mathbf{z}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. By solving Eq. (37), we get

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} -L_1 \sin q_1 - L_2 \sin\left(q_1 + q_2\right) & -L_2 \sin\left(q_1 + q_2\right) \\ L_1 \cos q_1 + L_2 \cos\left(q_1 + q_2\right) & L_2 \cos\left(q_1 + q_2\right) \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{38}$$

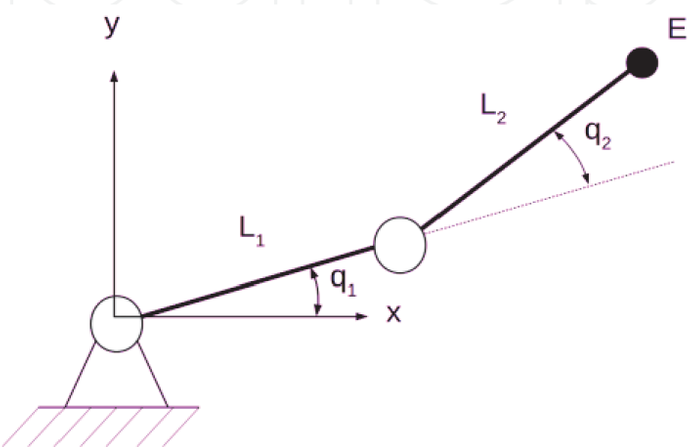

Figure 7.
Two-link mechanism.

Now, the Jacobian is a 6×2 matrix and its maximum rank is 2. That is, at most two components of angular/linear end-effector velocity can be independently assigned. In this case, when orientation is not required, only first two rows are considered.

5.3 Kinematically redundant manipulators

The next approach to inverse kinematic solution we want to focus on is numerical solutions. Nevertheless, many times, it is hard to find a closed-form solution for inverse kinematics; the basic kinematics of industrial robots have developed an approach to solve it. The problems arise with nonconventional kinematic structures, especially with kinematically redundant manipulators. A kinematically redundant manipulator has more number of DOFs than is absolutely necessary to perform the desired task. For example, conventional industrial robot has usually six DOFs, by which it is able to reach any point in its workspace. By adding an additional DOF, this robot becomes kinematically redundant due to this additional DOF.

A numerical solution is usually used when a closed-form solution for $\mathbf{q}$ does not exist or is difficult to find. In this section, we will focus on kinematically redundant mechanisms. Considering the dimension of joint space n and dimension of task space m, for kinematically redundant mechanisms $n > m$. The level of redundancy can be expressed by $r = n - m$. Kinematic redundancy is used for many tasks such as kinematic singularities avoidance, obstacle avoidance, joint limits avoidance, increasing the manipulability in specified directions, minimizing the energy consumption, minimum of motion torques, optimizing execution time, etc. As can be seen, kinematic redundancy allows many optimization tasks to be solved. On the other hand, kinematic redundancy brings some disadvantages as well; for example, a greater structural complexity of construction caused by many of DOFs (mechanical, actuators, sensors), which have an influence on final cost of this kind of mechanism. Next field of potential disadvantages is the field of control, due to complicated algorithms for inverse kinematic computation or motion control. From this reason redundant manipulators could be difficult in real-time control.

There are many approaches within numerical solution of inverse kinematics, which are still in focus of research. Most approaches deal with Jacobian matrix in order to find a linear approximation to the inverse kinematic problem. Among the most used of them are damped least squares (DLSs), Jacobian transpose, and damped least squares with singular value decomposition (SVD) [8, 9].

Another kind of approach is the approach based on Newton methods [6]. The aim of these algorithms is to find the final configuration of joints with focus on minimization problem. For this reason, the final motion of robot is smooth. This family of methods includes methods such as Powell's method [10] or Broyden, Fletcher, Goldfarb and Shanno (BFGS).

A very well-known and used method for inverse kinematics of kinematically redundant mechanisms is the so-called cyclic coordinate descent (CCD) algorithm [11]. The CCD method is a very simple and at the same time a very strong method. It is a heuristic iterative method with low computational cost per iteration. Next very know heuristic iterative method is FABRIK (Forward And Backward Reaching Inverse Kinematics) [12, 13]. The FABRIK method minimizes the system error by adjusting each joint angle one at a time.

Most of inverse kinematics numerical methods could be divided into two classes: linearization algorithms and minimization algorithms. Concerning the linearization algorithms, the idea is piecewise linearization of nonlinear inverse kinematic problem, which is based on the Jacobian matrix. An example of this kind of method is the Jacobian transpose method. In minimization algorithms, the idea is to formulate some cost function, which will be minimized, for example cyclic coordinate descent

algorithm. Besides the mentioned methods, there are many other such as pseudoinverse methods, such as the Levenberg–Marquardt damped least squares methods, quasi-Newton and conjugate gradient methods, neural network and artificial intelligence methods.

The basic technique is based on Eq. (39)

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{39}$$

The above relation can be inverted to so-called Jacobian control method

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q})\dot{\mathbf{x}} \tag{40}$$

which leads to joint velocity vector $\mathbf{q}$ with minimum norm. The term $\mathbf{J}^{\dagger}$ represents the Moore-Penrose pseudoinverse given by $\mathbf{J}^{\dagger} = \mathbf{J}^{T}\left(\mathbf{J}\mathbf{J}^{T}\right)^{-1}$ for kinematically redundant mechanisms where $m < n$ $\left(\mathbf{J}\mathbf{J}^{\dagger} = \mathbf{I}\right)$ or by $\mathbf{J}^{\dagger} = \left(\mathbf{J}^{T}\mathbf{J}\right)^{-1}\mathbf{J}^{T}$ for $m > n$ $\left(\mathbf{J}^{\dagger}\mathbf{J} = \mathbf{I}\right)$.

A very common method on which the solution is based is the Newton-Rhaphson method. The Newton-Rhaphson method is a root-finding algorithm that produces approximations of the roots of a real-valued function. The method starts with differentiable function f defined for a real variable x, derivative of function f', and initial guess x_0 for a root of function f. If these assumptions are satisfied and initial guess is close, then

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{41}$$

Talking about numerical motion optimization of kinematically redundant mechanisms, there are two approaches. The first approach deals with local methods, which are solved typically online, and the second one with global methods, which require quantity of computation. For this reason, the global methods are computed usually offline.

One of the commonly used local methods is the family of null-space methods. This method uses the extension of Eq. (40) and gives

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q})\dot{\mathbf{x}} + \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)\dot{\mathbf{q}}_0 \tag{42}$$

where $\dot{\mathbf{q}}_0 \in \mathbb{R}^n$ is an arbitrary joint space velocity vector, chosen according to desired behavior; so it is chosen for optimization purposes. Next, $\mathbf{I} \in \mathbb{R}^{n \times n}$ is the identity matrix. The term $\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)$ represents the orthogonal projection matrix in the null space of $\mathbf{J}$, $\mathbf{J}^{\dagger}(\mathbf{q})\dot{\mathbf{x}}$ is orthogonal to $\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)\dot{\mathbf{q}}_0$. For this reason, $\mathbf{q} = 0$. Physically, this term corresponds to self-motion, where the combined joints motion generates no motion in the task space (no motion of end-effector). So, the term $\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)$ is symmetric and idempotent ($\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)^2 = \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)$). Also, it ensures $\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)^{\dagger} = \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)$. The inverse kinematic solution expressed by Eq. (42) is equivalent to solving a quadratic programming (QP) problem based on $\mathbf{H}(\dot{\mathbf{q}}) = \min_{\dot{\mathbf{q}}} \frac{1}{2}\left(\dot{\mathbf{q}} - \dot{\mathbf{q}}_0\right)^{T}\mathbf{W}\left(\dot{\mathbf{q}} - \dot{\mathbf{q}}_0\right)$ subjected to $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$.

Now the question is, how the vector $\dot{\mathbf{q}}_0$ can be chosen. One of the basic ways to choose it is the so-called projected gradient method $\dot{\mathbf{q}}_0 = \nabla_q \mathbf{H}(\mathbf{q})$. Supposing that $\dot{\mathbf{x}} = 0$, that is, the mechanism performs only self-motion, it can be written $\dot{\mathbf{q}} = \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)\nabla_q \mathbf{H}$; so, $\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)\nabla_q \mathbf{H} = 0$ is a necessary condition of constrained optimality. Based on these facts, an objective function can be chosen for some optimization of motion:

- Manipulability—maximize the distance from kinematic singularities

$$\mathbf{H}(\mathbf{q}) = \sqrt{det\left[\mathbf{J}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\right]} \tag{43}$$

- Joint limit avoidance—minimize the distance from the middle of joints range

$$\mathbf{H}(\mathbf{q}) = \frac{1}{2}\sum_{i=1}^{n}\left(\frac{q_i - q_i^-}{q_{M,i} - q_{m,i}}\right)^2 \tag{44}$$

where $q_i \in \left[q_{m,i}, q_{M,i}\right]$ and $q_i^- = \frac{q_{M,i} - q_{m,i}}{2}$.

- Obstacle avoidance—maximize the minimum distance to obstacle

$$\mathbf{H}(\mathbf{q}) = \min_{\substack{\mathbf{a} \in robot \\ \mathbf{b} \in obstacle}} \|\mathbf{a}(\mathbf{q}) - \mathbf{b}\|^2 \tag{45}$$

where $\mathbf{a}(\mathbf{q})$ represents points on investigated mechanism and $\mathbf{b}$ represents points on the obstacle.

Example: Let us consider a planar six-link robot connected by six revolute joints. All links have the same length $L = 100\ mm$. The purpose of the simulation is path tracking (circle form) described by matrix $\mathbf{X}_{path} = \begin{bmatrix} \mathbf{x}_d \\ \mathbf{y}_d \end{bmatrix} \in \mathbb{R}^{m \times p}$, where p is the number of geometric points of the desired path, while $\mathbf{x}_d = -2.5L + L\cos\varphi$, $\mathbf{y}_d = L\sin\varphi$, $\varphi \in \{0, \ldots, 2\pi\}$ assuming the step of φ increase to be 0.2. That is, $\mathbf{x}_d = \left[x_1, \ldots, x_p\right] \in \mathbb{R}^p$ and $\mathbf{y}_d = \left[y_1, \ldots, y_p\right] \in \mathbb{R}^p$. From the matrix $\mathbf{X}_{path}$ will be in each path point determined the desired point $\mathbf{e}_p = \left[x_p y_p\right]^T \in \mathbb{R}^m$. Since, there is consideration of planar task with focus on end-effector position, the task space is $m = 2$. The expected solution assumes only primary solution without any secondary tasks. For inverse kinematic solution, damped least squares method, which avoids many of pseudoinverse method's problems, will be used. This method is also known as the Levenberg–Marquardt method, which arises from cost function $\|\mathbf{J}\Delta\mathbf{q} - \Delta\mathbf{x}\|^2 + \lambda^2\|\Delta\mathbf{q}\|^2$ where $\lambda \in \mathbb{R}$ is a non-zero scalar constant. By minimizing this term, one obtains

$$\Delta\mathbf{q} = \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}\right)^{-1}\Delta\mathbf{x} \tag{46}$$

where $\mathbf{I}$ $\mathbb{R}^{m \times m}$ is the identity matrix. The simulation will work according to the following algorithm.

Algorithm: Inverse kinematic model for 6-link manipulator.

1: Set desired path for end-effector by matrix $\mathbf{X}_{path}$ and parameters such as L, λ.
2: FOR step = $1 \rightarrow p$.
3: Set the desired position of end-effector from $\mathbf{X}_{path} \Longrightarrow \mathbf{e}_p = \left[x_p y_p\right]^T \in \mathbb{R}^m$
4: **WHILE** $\mathbf{x}_{actual} \neq \mathbf{e}_p$
5: Compute Jacobian
6: Compute $\mathbf{x}_{actual} = \left[x_{act} y_{act}\right]^T \in \mathbb{R}^m$
7: Compute $\Delta\mathbf{q} = \mathbf{q} + \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}\right)^{-1}\Delta\mathbf{x}$

8: $\mathbf{q} = \Delta\mathbf{q}$
9: **END WHILE**
10: END FOR

The results of the simulation can be seen in **Figures 8** and **9**. **Figure 8** presents the motion of planar robot. The aim is to tracking of path with circle shape (green color) by end-effector of the robot. The robot has a fixed frame in point [0,0]. The initial position of all joints is $\mathbf{q} = [0\ 0\ 0\ 0\ 0\ 0]^T$.

During the simulation, no restriction such as joint limit was considered. The simulation was done with a tolerance ±5 mm. **Figure 9** presents the variation of individual robot joints during the path tracking.

Example: Let us consider a planar 20-link robot connected with revolute joints. The links have the same length $L = 16.75\,mm$. The aim is to move the end-effector from its initial position to the end position by tracking the desired path. We will consider two cases. The first one considers free robot environment, the second one considers obstacles in the robot environment. The second solution will also consider kinematic singularities avoidance task and joint limit avoidance task.

Now, let us consider the cost function dealing with all mentioned secondary tasks [10].

$$\mathbf{H} = \|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}\|^2 + \|\mathbf{J}_c\dot{\mathbf{q}} - \mathbf{x}_c\|^2 + \|\mathbf{J}_L\dot{\mathbf{q}} - \mathbf{x}_L\|^2 + \|\rho\dot{\mathbf{q}}\|^2 \tag{47}$$

After mathematical adjustment, we get the final formula for kinematic control

$$\dot{\mathbf{q}} = \left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \mathbf{J}_c^T\mathbf{W}_c\mathbf{J}_c + \mathbf{J}_L^T\mathbf{W}_L\mathbf{J}_L + \mathbf{W}_s\right)^{-1}\left(\mathbf{J}^T\mathbf{W}\dot{\mathbf{x}}\right) \tag{48}$$

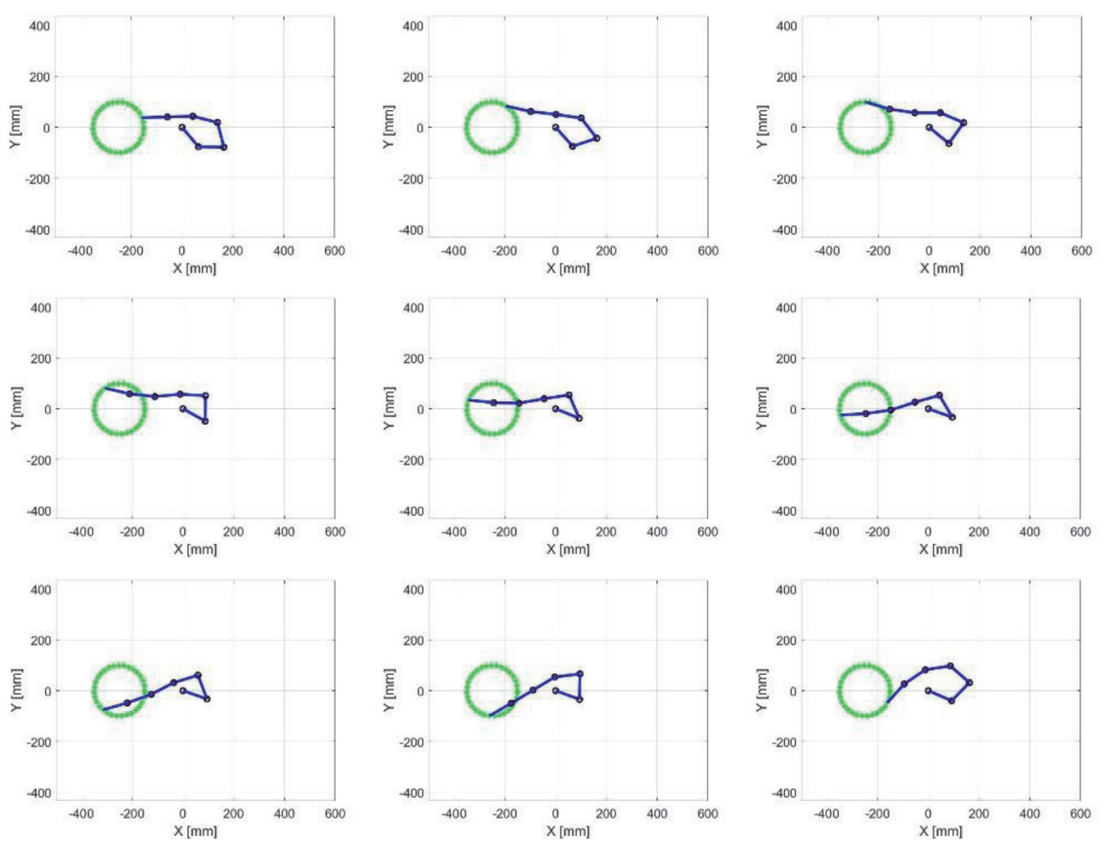

Figure 8.
Simulation of inverse kinematics for six-link manipulator.

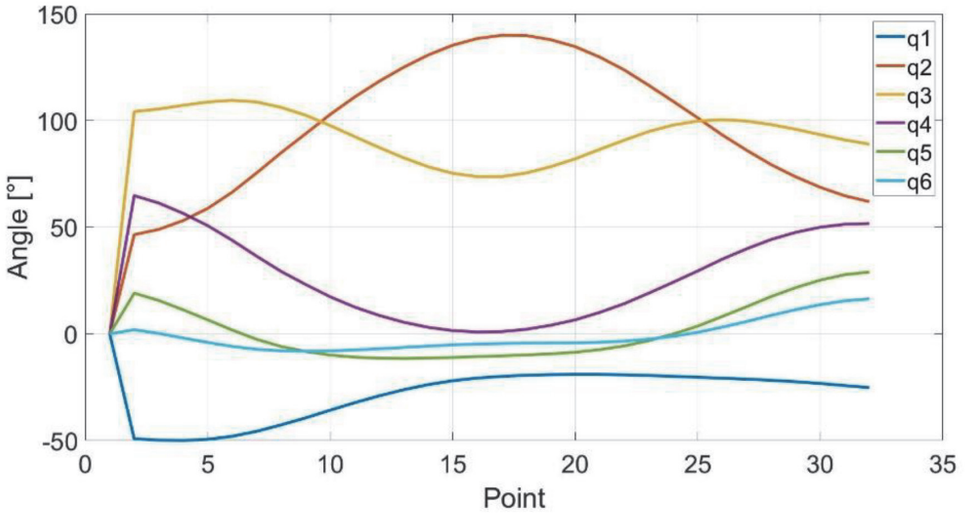

Figure 9.
Variations of joint angles.

where $\mathbf{W}$, $\mathbf{W}_c$, $\mathbf{W}_L$, and $\mathbf{W}_s$ are in our case 20 × 20 weight matrices of primary task, obstacle avoidance task, joint limit avoidance task, and kinematic singularities avoidance task, respectively. The setting of weight matrices is subjective. For example, if obstacle avoidance task should have higher priority above primary task by 100 times, the matrix $\mathbf{W}_c$ would consist of values 100 while $\mathbf{W}$ consist of values 1.

Let us consider individual secondary tasks. The joint limit avoidance task deals with the range of motion of individual manipulator links. How many joints will have their limit range is up to us. Of course, in real robots, usually all joints are limited in their motion. There are several ways on how to model the range of joint motion. In this case, an approach with changing of value of weight variable W_{li} based on joint position will be used. If the joint is in admissible range, the value of the weight variable is set to be zero. When the joint reaches the boundary of its range motion, the value of the weight variable increases. When the joint reaches a value outside its admissible range, the value of the weight variable increases to its maximum. This approach can be expressed by Eq. (49)

$$\mathrm{W_{li}} = \begin{cases} \mathrm{W_w} \leftarrow \mathrm{q_i} < \mathrm{q_{imin}} \\ \dfrac{\mathrm{W_w}}{2}\left\{1 + \cos\left[\pi\left(\dfrac{\mathrm{q_i} - \mathrm{q_{imin}}}{\rho_\mathrm{i}}\right)\right]\right\} \leftarrow \mathrm{q_{imin}} \le \mathrm{q_i} \le \mathrm{q_{imin}} + \rho_\mathrm{i} \\ 0 \leftarrow \mathrm{q_{imin}} + \rho_\mathrm{i} < \mathrm{q_i} < \mathrm{q_{imax}} - \rho_\mathrm{i} \\ \dfrac{\mathrm{W_w}}{2}\left\{1 + \cos\left[\pi\left(\dfrac{\mathrm{q_{imax}} - \mathrm{q_i}}{\rho_\mathrm{i}}\right)\right]\right\} \leftarrow \mathrm{q_{imax}} - \rho_\mathrm{i} \le \mathrm{q_i} \le \mathrm{q_{imax}} \\ \mathrm{W_w} \leftarrow \mathrm{q_i} > \mathrm{q_{imax}} \end{cases} \tag{49}$$

The value of the weight variable has to be set for every joint of the manipulator that needs to be limited in the range of motion. Individual weight variables W_{li} where $i \in \{0, \ldots, N\}$ are parts of the final weight matrix of the joint limit avoidance task $W_l \in \mathbb{R}^{n \times n}$. The final weight matrix $\mathbf{W}_\mathrm{l}$ is the diagonal matrix [14]:

$$\mathbf{W}_l = \begin{bmatrix} W_{l1} & & & & \\ & W_{l2} & & & \\ & & W_{l3} & & \\ & & & \ldots & \\ & & & & W_{ln} \end{bmatrix} \tag{50}$$

The weight matrix $\mathbf{W}_l$ is used with the corresponding Jacobian matrix $\mathbf{J}_L \in \mathbb{R}^{n \times n}$. The Jacobian matrix for the joint limit avoidance task is $\mathbf{J}_L = \partial \mathbf{e} / \partial \mathbf{q}$. If a particular joint does not consider the joint limit avoidance task, the value of $\mathbf{J}_L$ is set to be zero; otherwise it is set to be one. The limit of all joints in motion of the manipulator investigated in this study is set to be ±100°. Different way of joint limit control is according to above mentioned Eq. (44).

During the obstacle avoidance task, the control system investigates the relation between manipulator links and obstacles in their environment. In general, this task can be solved from two views. At first, one group of obstacles can represent static obstacles or other robots in an investigated environment. The second group of obstacles can be represented by dynamic obstacles, which means that these obstacles change their position relating to global reference system in the time. Of course, the second group of obstacles is more difficult from the view of control in comparison with static obstacles. It is more difficult especially in the cases of requirements for real-time control [15].

The aim of obstacle avoidance is to prevent the collision between any part of the manipulator and potential obstacles, other robots, or collision with itself. Again, there are many methods on how to control robot motion at a safe distance from other objects, regardless of whether the obstacles have regular or irregular shape. For simplification, the irregular shapes are usually replaced by appropriate regular shape. This simplification can also significantly simplify the mathematical model and obtaining the numerical solution can be faster and the solution more stable. One of the methods on how to simplify irregular shapes is to replace all irregular shapes by a cylinder, with the obstacle being situated in the center of the cylinder, see **Figure 10**. The diameter of the cylinder determines the distance of influence of this obstacle.

At first, we set the coordinate of an obstacle in the task space as $\mathbf{s}_o$. The projection of the line from the i-th joint of the manipulator link to the center of a cylinder (obstacle) on the i-th link is:

$$\mathbf{p}_i = \mathbf{e}_i^T (\mathbf{s}_0 - \mathbf{s}_i) \tag{51}$$

The coordinate of the link point with potential to get into collision is:

$$\mathbf{s}_{ci} = \mathbf{s}_i + \mathbf{p}_i \mathbf{e}_i \tag{52}$$

The distance between the potential point of collision and the center of the cylinder is:

$$\mathbf{d}_{ci} = \|\mathbf{s}_{ci} - \mathbf{s}_0\| \tag{53}$$

Subsequently, the unit vector of the potential point of collision can be given by:

$$\mathbf{u}_i = \frac{\mathbf{s}_{ai} - \mathbf{s}_0}{\mathbf{d}_{ci}} \tag{54}$$

Now, the Jacobian matrix for the obstacle avoidance task can be given. The i-th row of the Jacobian matrix can be expressed as

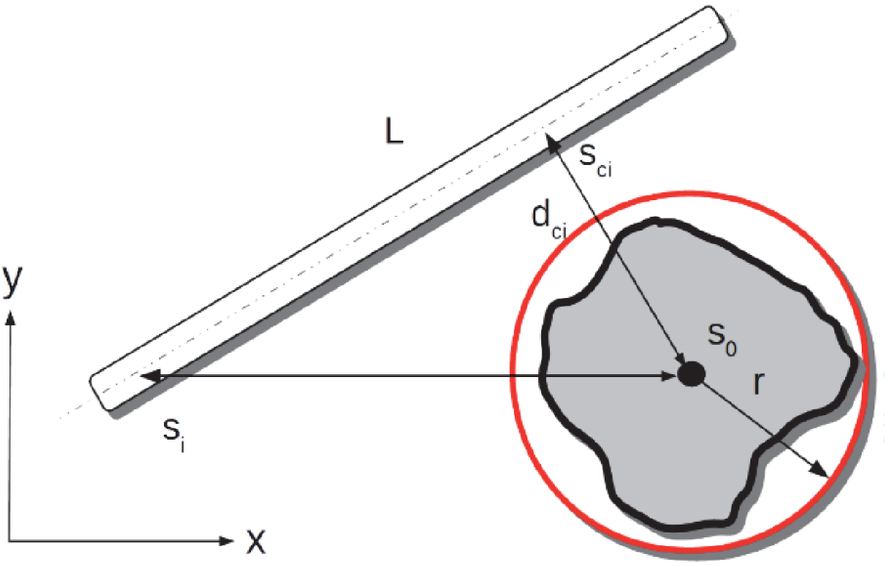

Figure 10.
Relation between manipulator link and obstacle.

$$\mathbf{J}_{ci} = -\mathbf{u}_i^T \mathbf{J}_{s_{ci}} \tag{55}$$

where matrix $J_{s_{ci}}$ is:

$$\mathbf{J}_{s_{ci}} = \frac{\partial \mathbf{s}_{ci}}{\partial \mathbf{q}} \tag{56}$$

The Jacobian matrix J_c consists of submatrices J_{ci}. The dimension of the Jacobian matrix is $J_c \in R^{c\times c}$, where c represents the number of manipulator links that could collide with the obstacles.

For numerical simulation, we will use following algorithm: Inverse kinematic model for 20-link manipulator [14].

Algorithm: Inverse kinematic model for 20-link manipulator

1: CYCLE WHILE 1
2: Determination of new required vector $x_d \in R^m$ from the matrix of planned path $P \in R^{r\times 2}$
3: CYCLE WHILE 2
4: Computation of Jacobian matrix J (damped least squares method)
5: Determination of actual end-effector position in the task space $x \in R^m$ with actual generalized variables $q \in R^n$
6: Computation of general equation

$$\dot{q} = \left(J^{-1}WJ + J_c{}^{-1}W_cJ_c + J_l{}^{-1}W_lJ_l + W_s\right)^{-1}\left(J^TW\dot{x}\right)$$

7: $q = q_{previous} + \dot{q}dt$
8: $q_{previous} = q$
9: IF $x_d = x$ THEN
END CYCLE WHILE 2
ELSE
CYCLE WHILE 2 continues
END IF
10: END CYCLE WHILE 2
11: END CYCLE WHILE 1

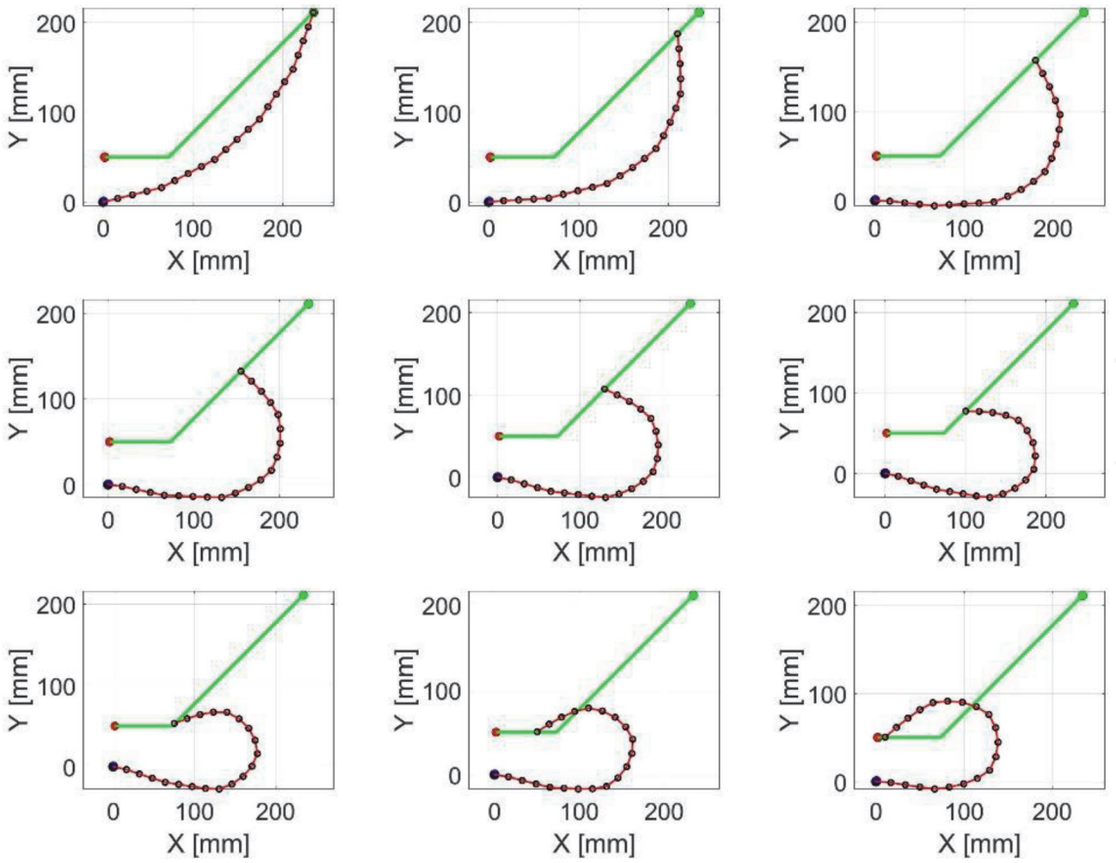

Figure 11.
Simulation of 20-link manipulator in free environment.

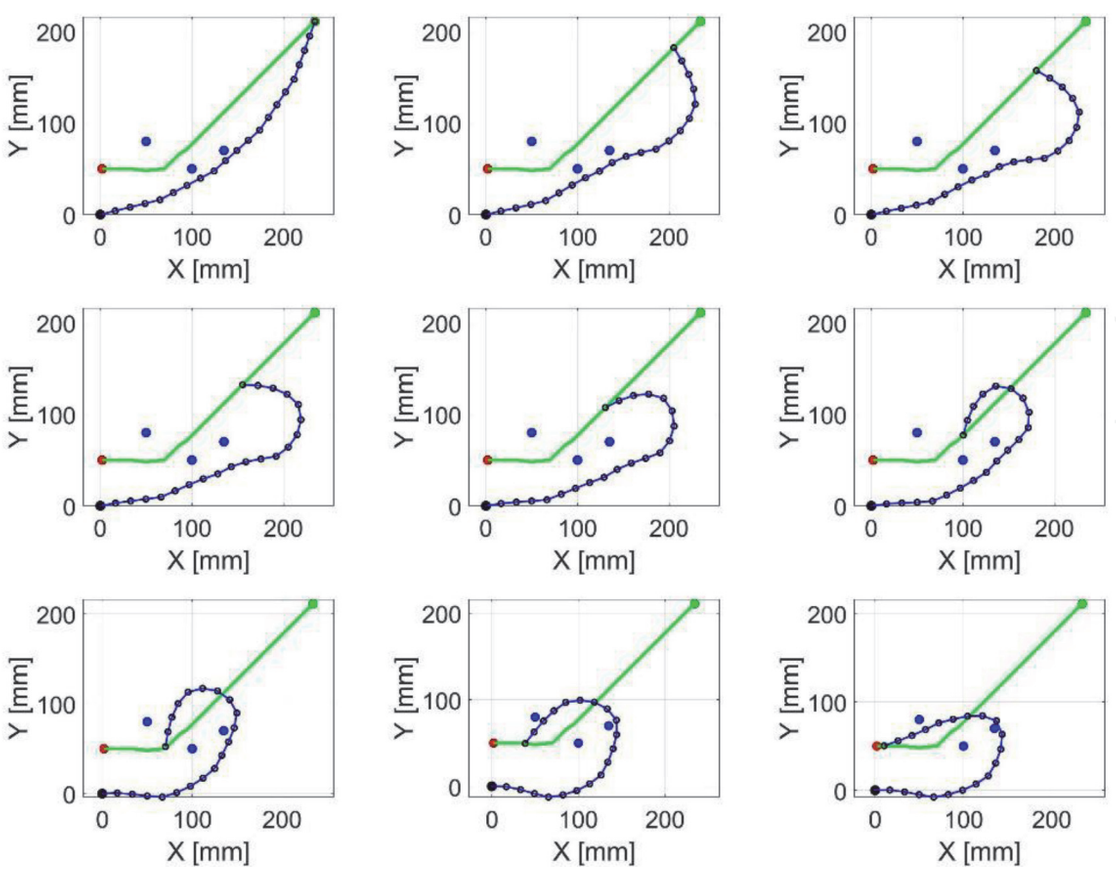

Figure 12.
Simulation of 20-link manipulator in constrained environment.

In **Figure 11** can be seen the simulation of motion of a 20-link manipulator, which moves in free environment without any obstacles. This case also does not consider any joint limit avoidance task. The aim is to move the end-effector of manipulator by predefined path (green color).

Figure 12 depicts a different situation. A 20-link manipulator moves according to predefined path between four obstacles. The solver also assumes joint limit avoidance task for all 20 joints. From this figure can be seen the difference in "self-motion" in joint space, while it does not affect the motion of end-effector in task space. The end-effector always tracks the same path (by neglecting end-effector orientation).

The second case is significantly difficult from the view of computational complexity in comparison with the case without any constraints in motion.

6. Conclusion

This chapter was focused on forward and inverse kinematics of open-chain mechanisms, namely manipulators with serial kinematic structures. We have introduced rigid motion and subsequently we have focused on forward kinematics. The chapter presents a kinematic model of SCARA by the well-known Denavit-Hartenberg convention. Within inverse kinematics was introduces several methods, including analytical, geometrical, and numerical. The last section dealt with modeling of kinematically redundant planar robots. At first, we introduced a six-link planar robot with focus on numerical solution using DLS. The last example presented a 20-link redundant manipulator moving between the obstacles. The solution includes, besides the primary solution, secondary tasks such as singularity avoidance, joint limit avoidance, and obstacle avoidance.

Acknowledgements

The authors would like to thank to Slovak Grant Agency VEGA 1/0389/18 "Research on kinematically redundant mechanisms" and to KEGA 018TUKE-4/2018 "Implementation of new technologies and education methods in area of controlling systems for improvement of educational level and practical experiences of graduate students of Mechatronics study program" and also to KEGA 030TUKE-4/2020 "Transfer of knowledge from the field of industrial automation and robotics to the education process in the teaching program mechatronics."

IntechOpen

Author details

Ivan Virgala*, Michal Kelemen and Erik Prada
Technical University of Košice, Košice, Slovakia

*Address all correspondence to: ivan.virgala@tuke.sk

IntechOpen

References

[1] Li XJ, Cao Y, Yang DY. A numerical-analytical method for the computation of robot workspace. In: The Proceedings of the Multiconference on "Computational Engineering in Systems Applications". Beijing; 2006. pp. 1082-1086. DOI: 10.1109/CESA.2006.4281805

[2] Dash AK, Chen IM, Yeo SH, Yang G. Workspace generation and planning singularity-free path for parallel manipulators. Mechanism and Machine Theory. 2005;**40**(7):776-805. DOI: 10.1016/j.mechmachtheory.2005.01.001

[3] Lynch KM, Park FC. Modern Robotics – Mechanics, Planning, and Control. Cambridge University Press; 2017. ISBN: 9781107156302

[4] Murray RM, Li Z, Sastry SS. A Mathematical Introduction to Robotic Manipulation. United States, Boca Raton, Florida: CRC Press; 1994. ISBN-13: 978-0849379819

[5] Trebuňa F, Virgala I, Pástor M, Lipták T, Miková Ľ. An inspection of pipe by snake robot. International Journal of Advanced Robotic Systems. 2016;**13**(5). DOI: 10.1177/1729881416663668

[6] Kucuk S, Bingul Z. Inverse kinematics solutions for industrial robot manipulators with offset wrists. Applied Mathematical Modelling. 2014;**38**(7–8): 1983-1999. DOI: 10.1016/j.apm.2013.10.014

[7] Siciliano B, Sciavicco B, Villani L, Oriolo G. Robotics: Modeling, Planning and Control. Springer; 2009. ISBN: 978-1-84628-641-4. DOI: 10.1007/978-1-84628-642-1

[8] Wampler CW. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 1986;**16**(1):93-101

[9] Baillieul J. Kinematic programming alternatives for redundant manipulators. In: Proceedings of the IEEE International Conference on Robotics and Automation. St. Louis, MO, USA; 1985. pp. 722-728

[10] Powell MJD. A Hybrid Method for Nonlinear Equations. Gordon & Breach Science Publishers, Inc.; 1970

[11] Wang L-CT, Chen CC. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. IEEE Transactions on Robotics and Automation. 1991;**7**(4):489-499

[12] Aristidou A, Lasenby J. FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem. Amsterdam, Netherlands: Elsevier; 2010. Submitted to Graphical Models

[13] Kelemen M, Virgala I, Lipták T, Miková Ľ, Filakovský F, Bulej V. A novel approach for a inverse kinematics solution of a redundant manipulator. Applied Sciences. 2018;**8**:2229

[14] Virgala I, Lipták T, Miková Ľ. Snake robot locomotion patterns for straight and curved pipe. Journal of Mechanical Engineering. 2018;**68**(2). DOI: 10.2478/scjme-2018-0020

[15] Fahini F. Autonomous Robots – Modelimg, Path Planning, and Control. Springer; 2009. ISBN: 978-0-387-09537-0. DOI: 10.1007/978-0-387-09538-7