doi: 10.5937/OTEH2402023T

Application of Screw Theory and its Implementation in Python for Controlling a Niryo One Manipulator

Vuk Todorović [1)]
Milan Blagojević [1)]
Nikola Nešić [1)]

The representation of a robot's configuration can be diverse, each with its advantages and disadvantages. In this paper, we approach the problem using screw theory, which states that all rigid-body motion can be represented with a rotation and translation along a screw axis. Using this as a basis, we'll be tackling the inverse kinematic problem of robots, which lacks standardized ways of being determined, unlike the forward kinematics problem. With that in mind, here we will show a combined approach of the analytic and numerical way of solving the inverse kinematic problem on a Niryo One robotic manipulator. The analytic solution is derived by simplifying the robot's structure and then using those results as initial guesses for the Newton-Raphson numerical method which may produce up to 8 possible solutions. The theoretical foundation is then implemented using the Python programming language after which the solution is sent to the robot via a Niryo One ROS API – pyniryo.

Key words: **screw theory, inverse kinematics, Paden-Kahan subproblems, Newton-Raphson method , Python, Niryo One.**

Introduction

IN the modern age, robotics is gaining an ever-increasing role in shaping modern life, hence why innovations are increasingly more directed towards this interdisciplinary scientific field. The structure of robots is complex and both individual and connected elements can significantly affect the whole structure and function of the mechanism. This is in favor of the fact that it is increasingly difficult to study one aspect of the robot's structure independently from the rest. Still, it is relatively easy to see how a dependent unit can affect the rest of the system and perform an analysis of its impact.

The essence of this paper lies in **screw theory** and its use for the analysis and synthesis of robot mechanics, as well as its application in controlling a **Niryo One manipulator** through the **Python** programming language and its publicly available libraries.

The work begins in the second chapter with some basic considerations that need to be taken into account, along with a theoretical movement model which we will follow and implement in an experiment.

In the third chapter, we get to the heart of this paper. After a brief overview of the task of determining a robot's inverse kinematics, we review the most important specifications of the Niryo One robot. In addition, the structure of the robot is displayed, the position of its joints, the type of communication and protocol, and **pyniryo** is specified as the Python library that allows us to apply ROS (Robot Operating System) from a high level of abstraction for controlling the robot. At the end, the **Paden-Kahan subproblems** and **Newton-Raphson method** are explained as they will be implemented to solve the **inverse kinematics** of the robot at the end of the chapter.

An experiment showing the operation of the robot and the application of the previous theory is presented in the fourth chapter.

The mechanics presented have been compiled together in the Python library by Lynch and Park [4]. Based on that, the authors have also developed a more refined library along with the code from the experiment and a more extensive version of this paper [1].

Niryo one manipulator

Basic Considerations

Niryo Robotics is a startup from France that designs and manufactures industrial robots, more precisely those robots are intended for personal and educational use purposes and the price is significantly lower than the average industrial robot whose price ranges between 20,000$ and 200,000$ [2]. The robot model used in this paper is from Niryo Robotics.

Apart from the open source software [7], the advantage of this robot is that the whole documentation can be found online for free at their website [6] and at GitHub [7].

Theoretical Movement Model

To showcase the proposed algorithm that this paper presents, in Table. 2.1 we have shown the planned sequental movements that the robot will perform.

Presented at the 11th International Scientific Conference on Defensive Technologies, Tara, Serbia, 9-11 October 2024.
[1)] Faculty of Technical Sciences, University of Pristina in Kosovska Mitrovica
Correspondence to: Vuk Todorović; e-mail: vuk.todorovic01@gmail.com

Table 2.1. Niryo One theoretical movement model with coordinates in the robots space frame

Actuation	Transformation matrix
1. Starting position (calculated from pyniryo [7])	$\begin{bmatrix} 0{,}540 & 0 & 0{,}841 & 91{,}039\cdot10^{-3} \\ 0 & 1 & 0 & 0 \\ -0{,}841 & 0 & 0{,}540 & 190{,}529\cdot10^{-3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$
2. Approach above the object	$\begin{bmatrix} 0 & 0 & 1 & 148\cdot10^{-3} \\ 0 & 1 & 0 & -148\cdot10^{-3} \\ -1 & 0 & 0 & 150\cdot10^{-3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$
3. Descent to the object	$\begin{bmatrix} 0 & 0 & 1 & 148\cdot10^{-3} \\ 0 & 1 & 0 & -148\cdot10^{-3} \\ -1 & 0 & 0 & 88\cdot10^{-3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$
4. Grip the object	N/A
5. Lift the object	Same as 2. Approach above the object
6. Approach above the objects destination	$\begin{bmatrix} 0 & 0 & 1 & 148\cdot10^{-3} \\ 0 & 1 & 0 & 148\cdot10^{-3} \\ -1 & 0 & 0 & 150\cdot10^{-3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$
7. Descent to the objects destination	$\begin{bmatrix} 0 & 0 & 1 & 148\cdot10^{-3} \\ 0 & 1 & 0 & 148\cdot10^{-3} \\ -1 & 0 & 0 & 88\cdot10^{-3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$
8. Release the object	N/A
9. Ascent from the object	Same as 6. Approach above the objects destination
10. Return to the start	Same as 1. Starting position

Robot Mechanics

The approach with which we'll explore robot mechanics is based on **screw theory** which states that all rigid body motion can be interpreted as a rotation and a translation along a screw axis resulting in a helicoid trajectory.

For the interested reader, an extensive resource is available at [3, 5] for screw theory while in this paper we'll only be using the results from those sources.

The main issue we will be discussing here is the inverse robot kinematics problem for our 6 degrees of freedom (DOF) [2, 8] robot. That is to say, given a homogeneous transformation matrix $\mathbf{T} \in SE(3)$ find the corresponding joint actuation angles $\boldsymbol{\theta} \in \mathbb{R}^n$,

$$f(\mathbf{T}) = \boldsymbol{\theta}. \quad (1)$$

The inverse kinematics problem may not always have a closed-form analytical solution or the solution is not favorable for usage in practical situations. Typically for 6 DOF, there are a finite number of solutions while for robots with more than 6 DOF, there are usually infinitely many solutions.

On the other hand, there are numerical methods that, if the initial guess is sufficient, will always converge to a valid solution if it exists. The problem is, how do we ensure convergence? This can be achieved by:

- Starting the robot's motion from a known configuration and slowly changing the end-effector's goal configuration with the frequency of the calculation,
- Use an approximate analytical solution as the initial guess.

This paper tries to achieve exactly the second point, find an approximate analytical solution, and use that as an initial guess for the numerical method. This is represented as Algorithm 2.

Technical Specifications

The robot's technical characteristics are available in Table 3.1. The joints and their position are given in Fig. 3.1.

Table 3.1. Niryo One technical specifications where the range of rotation of the actuators are from the Niryo One ROS API – pyniryo and the rest from [2, 7, 8]

Specification	Value
Mechanical specifications	
Degrees of Freedom	6
Weight	3,3 kg
Max Reach	440 mm
Max Payload	0,5 kg
Repeatability	±0,5 mm
Materials	Aluminium, PLA (3D printed)
Actuators	5 steppers and 2 servos
End effector – Gripper 1	
Weight	70 g
Length (gripper closed)	80 mm
Max opening width	27 mm
Picking distance from end-effector base	60 mm
Actuators range of rotation	
Joint 1	$-175{,}00000° \div 175{,}00000°$
Joint 2	$-109{,}44000° \div 36{,}670000°$
Joint 3	$-79{,}89000° \div 89{,}960000°$
Joint 4	$-174{,}76000° \div 175{,}00000°$

Joint 5	$-100,00000° \div 110,01000°$
Joint 6	$-144,96000° \div 144,96000°$
Electrical specifications	
Power Supply	$11V \div 6A$
Power Consumption	60W
Hardware	Arduino Mega, Raspberry Pi 3
Joint sensor	Magnetic sensor
Ports	4 USB, Ethernet
Software specifications	
Communication	Ethernet, Wi-Fi, Bluetooth, USB
User interface	Web app, Android app, iOS app, Gamepad
Programming interface	ROS, APIs, source code

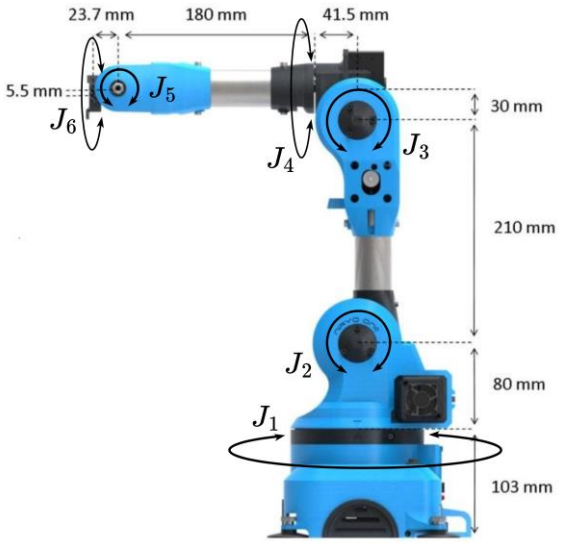


Figure 3.1. Niryo Ones The position of the joints and their respective axis of rotation where J_i denotes the i -th joint

Because we can connect and manage our robot in several ways (see Table 3.1), we must explicitly state which way will be used for our application:

- Communication – Ethernet TCP/IP connection,
- Control interface – Niryo One ROS API, i.e. **pyniryo**.

Ethernet is a reliable and relatively fast way of communication that requires a wired connection (in our case wireless is unnecessary anyway), while pyniryo is the latest available Python ROS API for the Niryo One. We could also apply directly ROS but that would introduce additional complexity and is not the aim of this paper. The development environment used is *VS Code* while the Python version used is 3.12.5.

Paden-Kahan Subproblems

The **Paden-Kahan** (**PD**) **subproblems** are a set of subproblems, that depend on the geometry of the robot, and that may be systemically used for determining a closed-form analytical solution for the inverse kinematics of the manipulator.

Before examining the subproblems, one more thing to note is that for each solution related to a particular joint angle θ_i, a valid solution is also

$$\theta = \theta_i \pm 2k\pi, \quad k \in \mathbb{Z} . \tag{2}$$

This may turn out to be important in cases where a robot's joint has an asymmetric joint range limit, e.g, let's say we have a robot whose joint ranges are $((-90°, 90°), (-225°, 45°), (-120°, 90°))$ but the solution vector $\boldsymbol{\theta} = (45°, 150°, -90°)$ is out of range, meaning that we must change our solution and say that $\boldsymbol{\theta} = (45°, -210°, -90°)$.

With that in mind, we'll explore how the subproblems are formulated, their solution, and the conditions under which a solution exists.

Subproblem 1. Rotation about a single axis

Problem. Let S be a zero-pitch screw axis $h = 0$ and $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ two points, find the corresponding rotation angle θ such that

$$e^{[S]\theta}\mathbf{p} = \mathbf{q} . \tag{3}$$

Solution. From the given subproblem it is evident that it corresponds to a rotation of the point $\mathbf{p}$ to the point $\mathbf{q}$ around the screw axis S and we are looking for the angle θ that satisfies this transformation.

The solution to this subproblem is given as

$$\theta = \operatorname{atan2}(\boldsymbol{\omega}_S \cdot (\mathbf{u}' \times \mathbf{v}'),\ \mathbf{u}' \cdot \mathbf{v}') \tag{4}$$

where:

$$\mathbf{u}' = \mathbf{u} - \operatorname{proj}_{\boldsymbol{\omega}_S} \mathbf{u} , \tag{5}$$

$$\mathbf{v}' = \mathbf{v} - \operatorname{proj}_{\boldsymbol{\omega}_S} \mathbf{v} , \tag{6}$$

$$\mathbf{u} = \mathbf{p} - \mathbf{r} , \tag{7}$$

$$\mathbf{v} = \mathbf{q} - \mathbf{r} , \tag{8}$$

and $\mathbf{r} \in \mathbb{R}^3$ is any point of the screw axis S. To determine this point, we'll first examine the parametric definition of the screw axis

$$S = \begin{bmatrix} \boldsymbol{\omega}_S \\ \mathbf{v}_S \end{bmatrix} = \begin{bmatrix} \hat{\omega} \\ -\hat{\omega} \times \mathbf{r} + h\hat{\omega} \end{bmatrix}$$

and considering that our screw axis is zero-pitch, we may write

$$\mathbf{v}_S = -[\boldsymbol{\omega}_S]\mathbf{r} \tag{9}$$

and thus

$$\mathbf{r} = -[\boldsymbol{\omega}_S]^{\dagger} \mathbf{v}_S . \tag{10}$$

While we may be able to solve for $\mathbf{r}$ analytically, it will always have an infinite amount of solutions (consider how the vector $\mathbf{r}$ points to a line), which may pose a problem since we'll need an explicit solution to implement it programmatically.

The subproblem has a solution when the following relations are satisfied:

$$\begin{matrix}\boldsymbol{\omega}_S \cdot \mathbf{u} = \boldsymbol{\omega}_S \cdot \mathbf{v} \\ \| \mathbf{u}' \| = \| \mathbf{v}' \| \Leftrightarrow \| \mathbf{u} \| = \| \mathbf{v} \|\end{matrix}. \quad (11)$$

In the case when $\mathbf{p} = \mathbf{q}$, there are an infinite amount of solutions to this subproblem.

Subproblem 2. Rotation about two subsequent axes

Problem. Let S_1 and S_2 be zero-pitch screw axes $h_1 = h_2 = 0$ that intersect and $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ two points, find the corresponding pair of rotation angles θ_1 and θ_2 such that

$$e^{[S_1]\theta_1} e^{[S_2]\theta_2} \mathbf{p} = \mathbf{q}. \quad (12)$$

Solution. This subproblem corresponds to two subsequent rotations of the point $\mathbf{p}$ to the point $\mathbf{q}$, first around the screw axis S_1 for an angle of θ_1 and then around the screw axis S_2 for an angle of θ_2.

In the case when the two screw axes overlap,

$$S_1 = \pm S_2 \Rightarrow S_1 \cdot S_2 = \pm \| S_1 \|^2 = \pm \| S_2 \|^2 \quad (13)$$

then we can rewrite (12) as

$$e^{[S_1]\theta_1} e^{[S_2]\theta_2} \mathbf{p} = e^{[S_1]\theta_1} e^{-[S_1]\theta_2} \mathbf{p} = e^{[S_1](\theta_1 - \theta_2)} \mathbf{p} = \mathbf{q} \quad (14)$$

that lends itself to PD 1 if we say that

$$\theta = \theta_1 \pm \theta_2. \quad (15)$$

All combinations of the two angles θ_1 and θ_2 that satisfy (15) can be used, meaning that we may simplify this further by saying that one angle equals zero.

When the screw axes intersect at a point, (12) can be rewritten as a set of two equations

$$\begin{matrix} e^{[S_1]\theta_1} \mathbf{p} = \mathbf{c} \\ e^{-[S_2]\theta_2} \mathbf{q} = \mathbf{c} \end{matrix} \quad (16)$$

such that they may be solved as PD 1 if we say that

$$\mathbf{c} = \mathbf{z} + \mathbf{r} \quad (17)$$

and $\mathbf{r} \in \mathbb{R}^3$ is the point of intersection between the two screw axes S_1 and S_2. Since, in the general case, if we use (10), we'll get two different points $\mathbf{r}_1 \neq \mathbf{r}_2$, there is a need to use another method for determining $\mathbf{r}$. The property of the vector cross-product $\mathbf{a} \times \alpha \mathbf{a} = \mathbf{0}$ allows us to write out (9) as a linear combination

$$-[\boldsymbol{\omega}_S]\mathbf{r}_i = -[\boldsymbol{\omega}_S](\mathbf{r} - \xi_i \boldsymbol{\omega}_S) = \mathbf{v}_S, \quad \xi_i \in \mathbb{R}, \quad i = 1, 2 \quad (18)$$

thus,

$$\mathbf{r} = \mathbf{r}_1 + \xi_1 \boldsymbol{\omega}_{S_1} = \mathbf{r}_2 + \xi_2 \boldsymbol{\omega}_{S_2} \quad (19)$$

and this can be solved as a linear system

$$\begin{bmatrix} \boldsymbol{\omega}_{S_1} & -\boldsymbol{\omega}_{S_2} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \mathbf{r}_1 - \mathbf{r}_2. \quad (20)$$

If the system has a solution, then the screw axes intersect at the point defined by (19), otherwise the axes do not intersect and this subproblem has no solution.

The other variable from (17) is defined as:

$$\mathbf{z} = \alpha \boldsymbol{\omega}_{S_1} + \beta \boldsymbol{\omega}_{S_2} + \gamma (\boldsymbol{\omega}_{S_1} \times \boldsymbol{\omega}_{S_2}), \quad (21)$$

$$\alpha = \frac{(\boldsymbol{\omega}_{S_1} \cdot \boldsymbol{\omega}_{S_2})(\boldsymbol{\omega}_{S_2} \cdot \mathbf{u}) - \boldsymbol{\omega}_{S_1} \cdot \mathbf{v}}{(\boldsymbol{\omega}_{S_1} \cdot \boldsymbol{\omega}_{S_2})^2 - 1}, \quad (22)$$

$$\beta = \frac{(\boldsymbol{\omega}_{S_1} \cdot \boldsymbol{\omega}_{S_2})(\boldsymbol{\omega}_{S_1} \cdot \mathbf{v}) - \boldsymbol{\omega}_{S_2} \cdot \mathbf{u}}{(\boldsymbol{\omega}_{S_1} \cdot \boldsymbol{\omega}_{S_2})^2 - 1}, \quad (23)$$

$$\gamma = \pm \sqrt{\frac{\| \mathbf{u} \|^2 - \alpha^2 - \beta^2 - 2\alpha\beta(\boldsymbol{\omega}_{S_1} \cdot \boldsymbol{\omega}_{S_2})}{\| \boldsymbol{\omega}_{S_1} \times \boldsymbol{\omega}_{S_2} \|^2}}, \quad (24)$$

where $\mathbf{u}$ and $\mathbf{v}$ are defined by (7) and (8) respectively.

The subproblem can either have two pairs of solutions, one pair of solutions, or no solutions for the angles θ_1 and θ_2. Two pairs of solutions are present when $\gamma \neq 0$, and one pair when (13) holds or $\gamma = 0$. No solutions exist when the square root from (24) is less than zero or certain conditions aren't met. The conditions for using this subproblem, in addition to the condition that the screw axes intersect, are (the implication of these conditions follows from (11) for (16)):

$$\begin{matrix} \boldsymbol{\omega}_{S_2} \cdot \mathbf{u} = \boldsymbol{\omega}_{S_2} \cdot \mathbf{z}, \\ \boldsymbol{\omega}_{S_1} \cdot \mathbf{v} = \boldsymbol{\omega}_{S_1} \cdot \mathbf{z}, \\ \| \mathbf{u} \|^2 = \| \mathbf{z} \|^2 = \| \mathbf{v} \|^2. \end{matrix} \quad (25)$$

Subproblem 3. Rotation to a given distance

Problem. Let S be a zero-pitch screw axis $h = 0$, $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ two points, and $\delta \geq 0 \in \mathbb{R}$ a real number, find the corresponding rotation angle θ such that

$$\| \mathbf{q} - e^{[S]\theta} \mathbf{p} \| = \delta. \quad (26)$$

Solution. From the given subproblem it is evident that it corresponds to rotating the point $\mathbf{p}$ such that after the transformation, it is a distance of δ from the point $\mathbf{q}$.

In the case that $\delta = 0$, (26) transforms into (3), i.e. PD 1

$$\| \mathbf{q} - e^{[S]\theta} \mathbf{p} \| = 0 \Rightarrow \mathbf{q} - e^{[S]\theta} \mathbf{p} = \mathbf{0} \Rightarrow e^{[S]\theta} \mathbf{p} = \mathbf{q}. \quad (27)$$

When $\delta > 0$ then we have the usual case with the solution given as

$$\begin{aligned} \theta = \operatorname{atan2}(\boldsymbol{\omega}_S \cdot (\mathbf{u}' \times \mathbf{v}'), \ \mathbf{u}' \cdot \mathbf{v}') \\ \pm \arccos\left(\frac{\| \mathbf{u}' \|^2 + \| \mathbf{v}' \|^2 - \delta'^2}{2 \| \mathbf{u}' \| \| \mathbf{v}' \|} \right) \end{aligned} \quad (28)$$

where

$$\delta'^2 = \delta^2 - | \boldsymbol{\omega}_S \cdot (\mathbf{p} - \mathbf{q}) | \quad (29)$$

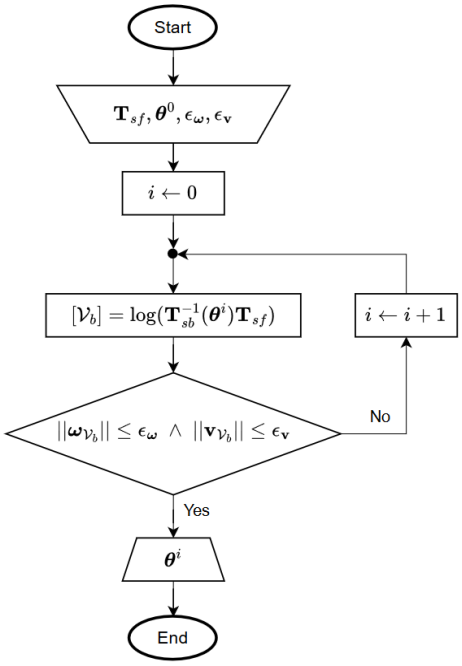
and the rest ($\mathbf{u}'$, $\mathbf{v}'$, $\mathbf{u}$, $\mathbf{v}$, $\mathbf{r}$) are defined by (5)÷(10).

Based on (28) we notice that this subproblem may have two or one solution based on the angle defined by the arccos argument. It is also evident that we don't have a solution if the arccos argument is outside of the function's domain. There may also be no valid solutions even if (28) yields a proper value which occurs if (26) is not satisfied.

Newton-Raphson Numerical Method

The numerical method that we'll be analyzing here is the **Newton-Raphson** method used for nonlinear root-finding. The algorithm for the method, defined in the $\{s\}$ frame, is

stated in Algorithm 1.



Algorithm 1. Newton-Raphson numerical method in the robots space frame

Niryo One Mechanics

The manipulator can be simplified and graphically represented as shown in Fig. 3.2 where we have a space frame $\{s\}$ and body frame $\{b\}$ set to standard positions with our robot model (which is per the pyniryo interface).

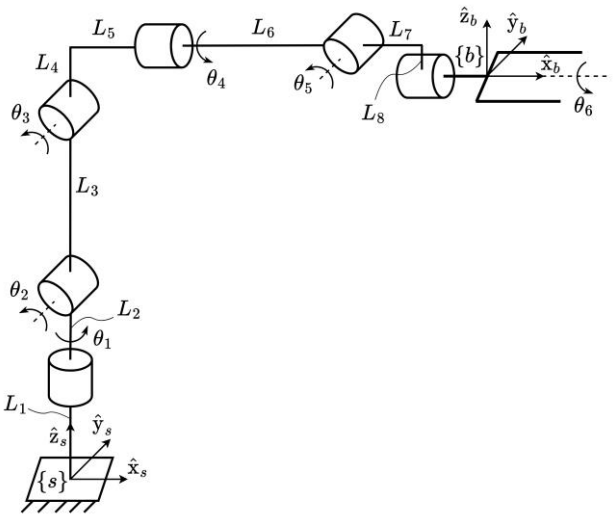


Figure 3.2. Kinematic diagram of the Niryo One manipulator

The zero configuration of the Niryo One manipulator is therefore

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & L_{57} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_{14} - L_8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where the values of the link lengths are given in Table 3.2. along with the definitions of the robots screw axes.

Table 3.2. Niryo One important kinematic values based on Fig. 3.1 and Fig. 3.2

Index number	Link length L in mm	Screw axis in the $\{s\}$ frame – S
1	103	(0, 0, 1, 0, 0, 0)
2	80	$(0, -1, 0, L_{12}, 0, 0)$
3	210	$(0, -1, 0, L_{13}, 0, 0)$
4	30	$(1, 0, 0, 0, L_{14}, 0)$
5	41.5	$(0, -1, 0, L_{14}, 0, -L_{56})$
6	180	$(1, 0, 0, 0, L_{14} - L_8, 0)$
7	23.7	/
8	5.5	/

As mentioned, we'll determine the inverse kinematics of the robot numerically, while using an approximate analytical solution as an initial guess analytically. Note that if we make an approximation and say that $L_8 \approx 0$, our manipulator closely resembles the structure of a PUMA robot for which the analytical solution is given in [5]. Therefore we will, by analogy, determine an approximate analytical solution to the PUMA robot.

We start from the expression of the direct kinematics in the $\{s\}$ frame according to our robot model

$$\prod_{i=1}^{6} e^{[\mathrm{S}_i]\theta_i} \mathbf{M} = \mathbf{T}_{sf},$$

and multiply both sides by $\mathbf{M}^{-1}$ to separate the unknowns from the known variables

$$\prod_{i=1}^{6} e^{[\mathrm{S}_i]\theta_i} = \mathbf{T}_{sf}\mathbf{M}^{-1} =: \mathbf{T}_1 \tag{30}$$

where

$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -L_{57} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L_8 - L_{14} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{31}$$

In four steps we will determine all of the joint angles.

Step 1 (solve for θ_3)

Multiply both sides with a vector that points to the point of intersection of the axes S_1, S_2 and S_3

$$\prod_{i=1}^{3} e^{[\mathrm{S}_i]\theta_i} \mathbf{q}_{456} = \mathbf{T}_1 \mathbf{q}_{456} \tag{32}$$

where the vector of intersection is defined as

$$\mathbf{q}_{456} = \begin{bmatrix} L_{56} \\ 0 \\ L_{14} \end{bmatrix}. \tag{33}$$

The previous relation (32) is a consequence of the fact

that if a vector $\mathbf{q}_S$ points to any point on a screw axis S, then the following holds true

$$e^{[S]\theta}\mathbf{q}_S = \mathbf{q}_S . \tag{34}$$

Next, we subtract both sides of (32) with the vector pointing to the intersection of the screw axes S_1 and S_2 to get

$$e^{[S_1]\theta_1}e^{[S_2]\theta_2}(e^{[S_3]\theta_3}\mathbf{q}_{456} - \mathbf{q}_{12}) = \mathbf{T}_1\mathbf{q}_{456} - \mathbf{q}_{12} \tag{35}$$

where the vector $\mathbf{q}_{12}$ is defined as

$$\mathbf{q}_{12} = \begin{bmatrix} 0 \\ 0 \\ L_{12} \end{bmatrix}. \tag{36}$$

Taking into account that homogeneous transformations preserve distances, norming both sides of (35) gives us

$$\| e^{[S_3]\theta_3}\mathbf{q}_{456} - \mathbf{q}_{12} \| = \| \mathbf{T}_1\mathbf{q}_{456} - \mathbf{q}_{12} \| =: \delta \tag{37}$$

which can be solved using the results from PD 3.

Step 2 (solve for θ_1 and θ_2)

Considering that, preceding this step, we have determined θ_3, we come back to (32) and after manipulating the relation we get

$$\begin{aligned} e^{[S_1]\theta_1}e^{[S_2]\theta_2}(e^{[S_3]\theta_3}\mathbf{q}_{456}) &= \mathbf{T}_1\mathbf{q}_{456} \\ \Rightarrow e^{[S_1]\theta_1}e^{[S_2]\theta_2}\mathbf{q}^{(1)} &= \mathbf{p}^{(1)} \end{aligned} \tag{38}$$

where

$$\mathbf{q}^{(1)} := e^{[S_3]\theta_3}\mathbf{q}_{456}, \tag{39}$$

$$\mathbf{p}^{(1)} := \mathbf{T}_1\mathbf{q}_{456}, \tag{40}$$

which can be solved using PD 2.

Step 3 (solve for θ_4 and θ_5)

After determining the first three angles which play a crucial role in the position of the robot's end-effector, we come back to (30) and separate the variables which are and aren't known

$$e^{[S_4]\theta_4}e^{[S_5]\theta_5}e^{[S_6]\theta_6} = e^{-[S_3]\theta_3}e^{-[S_2]\theta_2}e^{-[S_1]\theta_1}\mathbf{T}_1 \tag{41}$$

and multiplying both sides with the position vector $\mathbf{q}_6$ of the screw axis S_6 we get an equation that can be solved using PD 2

$$e^{[S_4]\theta_4}e^{[S_5]\theta_5}\mathbf{q}_6 = \mathbf{T}_2\mathbf{q}_6 =: \mathbf{p}_6 \tag{42}$$

where the position vector, which mustn't be equal to $\mathbf{q}_{456}$, is defined as

$$\mathbf{q}_6 = \begin{bmatrix} q_6 \\ 0 \\ L_{14} \end{bmatrix}, \quad q_6 \in \mathbb{R} \setminus \{L_{56}\} \tag{43}$$

and the matrix $\mathbf{T}_2$

$$\mathbf{T}_2 := e^{-[S_3]\theta_3}e^{-[S_2]\theta_2}e^{-[S_1]\theta_1}\mathbf{T}_1 . \tag{44}$$

Step 4 (solve for θ_6)

The last remaining angle can be calculated by separation in (30)

$$e^{[S_6]\theta_6} = \prod_{i=1}^{5}\exp(-[S_{6-i}]\theta_{6-i})\mathbf{T}_1 \tag{45}$$

and then multiply both sides by any vector $\mathbf{q}^{(2)}$ which doesn't point to any point on the screw axis S_6

$$e^{[S_6]\theta_6}\mathbf{q}^{(2)} = \prod_{i=1}^{5}\exp(-[S_{6-i}]\theta_{6-i})\mathbf{T}_1\mathbf{q}^{(2)} =: \mathbf{p}^{(2)} \tag{46}$$

where

$$\mathbf{q}^{(2)} \in \mathbb{R}^3 \setminus \{\mathbf{q}_6\}. \tag{47}$$

At the end of our analysis, we emphasize that the maximum number of different angle coordinates which achieve the same desired configuration is 8 due to the multiplicity of solutions in the equations (37), (38), and (42) . This can be represented by a graph as in Fig. 3.3.

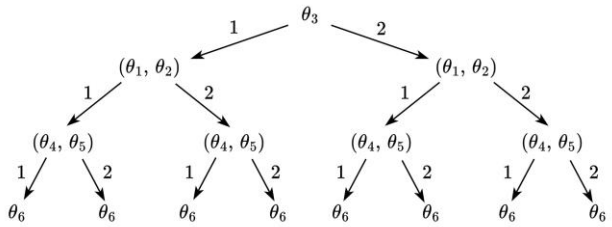
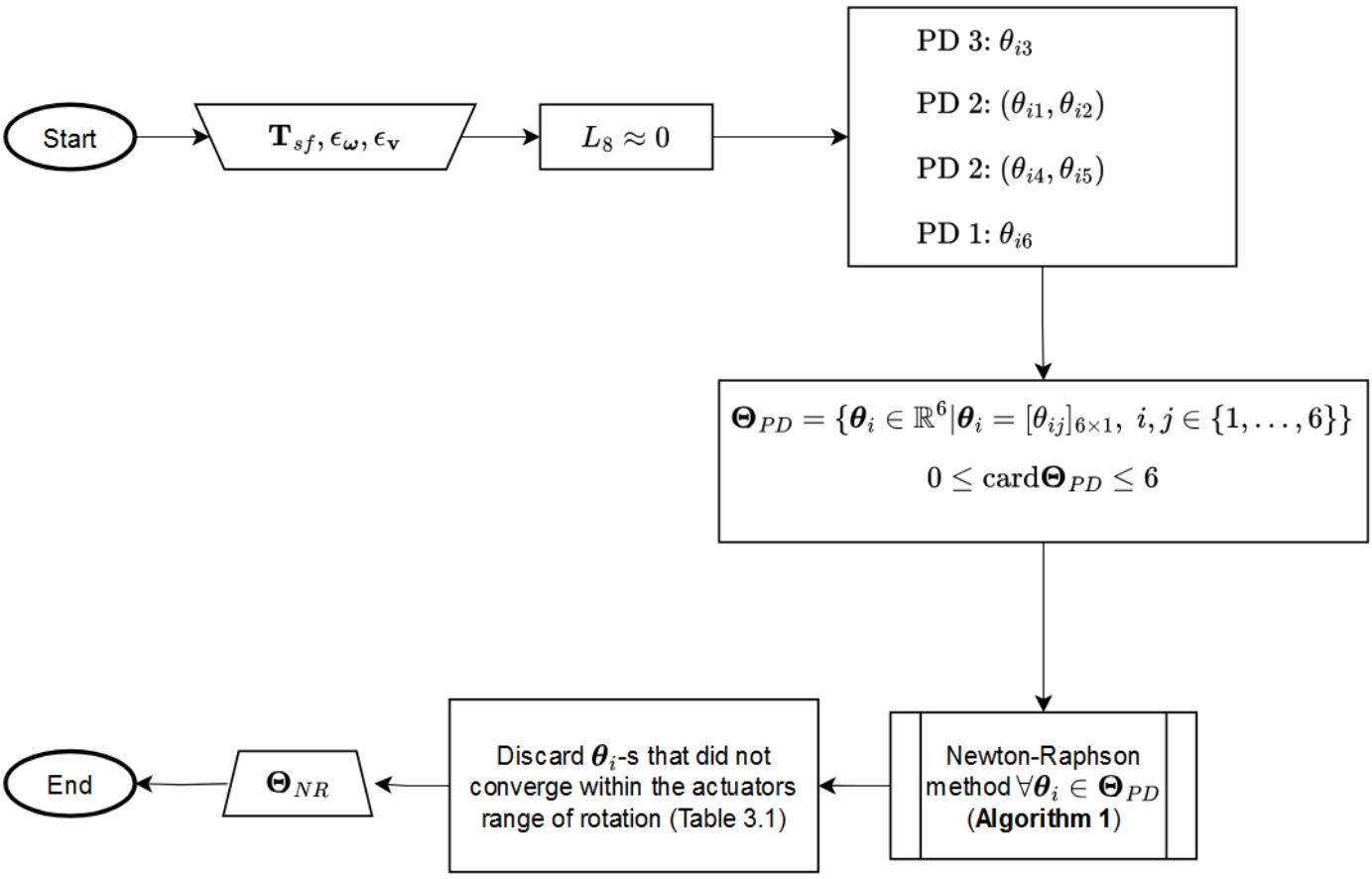


Figure 3.3. Graf representing PD approximate solutions where the nodes represent the angle coordinates and the branches lead to different sets of solutions

The graf from Fig. 3.3 represents the most general case when all 8 solutions exist. Most often, there are less than 8 solutions. This is because either the robot cannot reach a certain end-effector position and orientation with all 8 robot configurations or the algorithm converges to a certain configuration which would require the robot's actuators to move outside of their joint limit (see Table 3.1 for joint limits). If we have no PD solutions, then the desired configuration is outside the robot's workspace, otherwise it is inside.

Based on this chapter intended for determining Niryo One manipulator mechanics, an open-source Python library has been made and can be found on Github [1]. A larger portion of the library is related to general open-chain manipulators while only a single module is specialized for the Niryo One robot.

At the end, a graphical representation is shown on Algorithm 2.



Algorithm 2. Niryo One inverse kinematic algorithm. The sets $\mathbf{\Theta}_{PD}$ and $\mathbf{\Theta}_{NR}$ group the different valid solutions after applying the PD subproblems and Newton-Raphson method respectively

Experiment

Using the aforementioned method of determining Niryo One's inverse kinematics, we'll conduct an experiment that will serve as some validity to the effectiveness of this approach.

Problem. Displace the provided object using the Niryo One manipulator. The starting and final location of the object, measured in the robots $\{s\}$ frame, are $(148, -148, 0)$mm and $(148, 148, 0)$mm respectively.

Solution. The code used is available on the same GitHub page as the code for the entire library [1]. The robot is displayed doing its task in Fig. 4.1.

a) b) c)

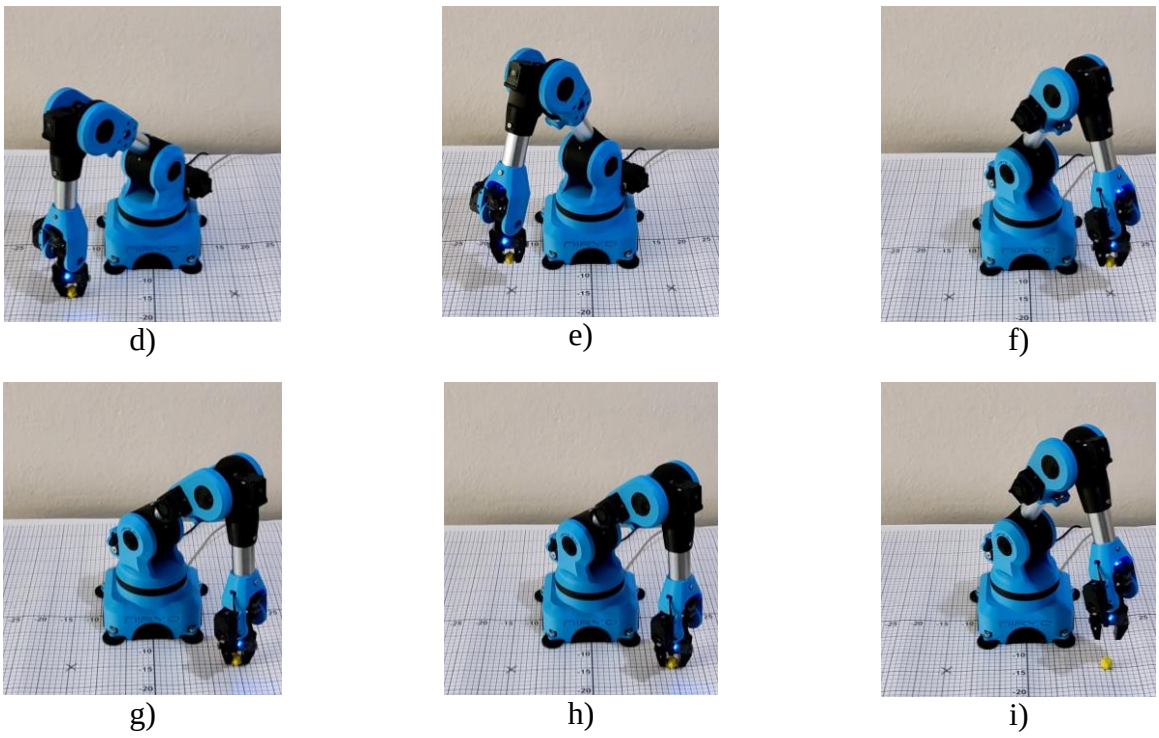


Fig. 4.1. Experiment: controlling the Niryo One manipulator. The experiment starts after the robot's calibration and ends when the robot returns to its starting position (which isn't shown here): a) Position 1 – starting position of the robot; b) Position 2 – approach above the object; c) Position 3 – descent to the object; d) Position 4 – gripping the object; e) Position 5 – lifting the object; f) Position 6 – approach above the objects destination; g) Position 7 – descent to the objects destination; h) Position 8 – release the object; i) Position 9 – ascent from the object.

Conclusion

Integration of various synergistic elements of robotic systems at a high theoretical and practical level can serve as a good example of engineering spirit, which was just performed in the work. Efficient calculation, clear formulation, easy implementation, lack of singularities, and the experiment on the Niryo One robot are evident indicators of the advantages of applying screw theory.

Another approach for determining a robot's inverse kinematics is based on the classic elimination theory from algebraic geometry, which is available at [5]. Python is a good choice for a fast and relatively simple implementation that can be refined at [1], to make the source code even more complete, robust, and efficient. In addition, the implementation in the C++ programming language would be ideal for a version of the source code that would be more suitable for the modern requirements of practical robotics. Before that, of course, it is also possible to delve deeper into the possibilities of ROS to more fully exploit the potential of the Niryo One robot.

Appendix: Mathematical Notation

The following notation is used:

- $\{s\}$ – space frame;
- $\{b\}$ – body frame;
- $L_{kn} = \sum_{i=k}^{n} L_i$ – sum of L_i link lengths;
- $\mathbf{x}_a = \begin{bmatrix} x_{a1} & x_{a2} & x_{a3} \end{bmatrix}^T \in \mathbb{R}^3$ – vector. If a subscript is present, then it defines the coordinate frame (e.g. coordinate frame a in this case) of the vector;
- $\hat{\mathbf{a}}_a = \mathbf{a}_a / \| \mathbf{a}_a \| \in \mathbb{R}^3$ – unit vector of its bolded equivalent;
- $\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ – the 3×1 zero vector;
- $\mathbf{a} \cdot \mathbf{b}, \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ – vector dot product;
- $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]\mathbf{b}, \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ – vector cross product (the $[*]$ operator is explained below);
- $\text{proj}_{\mathbf{b}}\mathbf{a} = (\mathbf{a} \cdot \hat{\mathbf{b}})\hat{\mathbf{b}}, \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^3$ – vector projection of $\mathbf{a}$ onto $\mathbf{b}$;
- $\mathbf{A}^{\dagger} \in \mathbb{R}^{m \times n}$ – the Moore-Penrose pseudoinverse;
- $\mathbf{R}_{ab} \in SO(3)$ – Rotation matrix from the special orthogonal group. If a subscript is present, then it defines the orientation of the second letter frame in the first letter frame (e.g. orientation of the frame $\{b\}$ in $\{a\}$ in this case). Otherwise, the matrix represents an operator;
- $[\mathbf{x}] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \in so(3), \quad \mathbf{x} \in \mathbb{R}^3$ – vector skew-symmetric matrix representation;
- $\mathbf{T}_{ab} = (\mathbf{R}_{ab}, \ \mathbf{p}_{ab}) \in SE(3)$ – homogeneous transformation matrix from the special Euclidean group. If a subscript is present, then it defines the orientation and position of the second letter frame in the first letter frame (e.g. position and orientation of the frame $\{b\}$ in $\{a\}$ in this case). Otherwise, the matrix represents an operator;

- $\mathrm{S}_a = \begin{bmatrix} \boldsymbol{\omega}_{\mathrm{S}_a} \\ \mathbf{v}_{\mathrm{S}_a} \end{bmatrix} \in \mathbb{R}^6, \quad \boldsymbol{\omega}_{\mathrm{S}_a}, \mathbf{v}_{\mathrm{S}_a} \in \mathbb{R}^3,$ – screw axis. If a $(\| \boldsymbol{\omega}_{\mathrm{S}_a} \| = 1) \vee (\| \boldsymbol{\omega}_{\mathrm{S}_a} \| = 0 \wedge \| \mathbf{v}_{\mathrm{S}_a} \| = 1)$ subscript is present, then it defines the coordinate frame (e.g. coordinate frame a in this case) of the screw axis;
- $[\mathrm{S}] = \begin{bmatrix} [\boldsymbol{\omega}_{\mathrm{S}}] & \mathbf{v}_{\mathrm{S}} \\ \mathbf{0}^T & 0 \end{bmatrix} \in se(3)$ – an $se(3)$ representation of the screw axis, equivalent to $[\mathbf{x}]$ explained before;
- $\mathrm{V}_a = \mathrm{S}_a \dot{\theta} = \begin{bmatrix} \boldsymbol{\omega}_{\mathrm{V}_a} \\ \mathbf{v}_{\mathrm{V}_a} \end{bmatrix}$ – twist. If a subscript is present, then it defines the coordinate frame (e.g. coordinate frame a in this case) of the twist;
- $[\mathrm{V}] = [\mathrm{S}]\dot{\theta} \in se(3)$ – an $se(3)$ representation of the twist, equivalent to $[\mathbf{x}]$ explained before;
- $\mathbf{M} \in SE(3)$ – zero/home configuration of an open chain manipulator in the $\{s\}$ frame;
- $\mathbf{J}_b(\boldsymbol{\theta}) \in \mathbb{R}^{6 \times n}$ – the Jacobian of a manipulator defined in the $\{b\}$ frame;
- $e^{[\mathrm{S}]\theta}\mathbf{x} = \mathbf{T}\mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{p}$ – an abuse of notation where $\mathbf{T}\mathbf{x}$ and $e^{[\mathrm{S}]\theta}\mathbf{x}$ are shorthand for $\mathbf{R}\mathbf{x} + \mathbf{p}$, and also $e^{[\mathrm{S}]\theta} = \mathbf{T}, \mathbf{T} = (\mathbf{R}, \mathbf{p}), \mathbf{x} \in \mathbb{R}^3$;
- $\operatorname{atan2}(x, y) = \begin{cases} \arctan\left(\frac{y}{x}\right) & x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & x < 0 \ \wedge \ y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & x < 0 \ \wedge \ y < 0, \\ \frac{\pi}{2} & x = 0 \ \wedge \ y > 0, \\ -\frac{\pi}{2} & x = 0 \ \wedge \ y < 0, \\ \text{undefined} & x = 0 \ \wedge \ y = 0 \end{cases},$

  $x, y \in \mathbb{R}$

  – a special function that is similar to the $\arctan(y/x)$ function but with a larger domain of $(-\pi, \pi]$.

References

[1] Library containing the source code for robot mechanics. URL: https://github.com/VuckoT/mehanika_robota.

[2] Kickstarter Niryo One project. URL: https://www.kickstarter.com/projects/niryo/niryo-one-an-open-source-6-axis-robotic-arm-just-f.

[3] Kevin M. Lynch and Frank C. Park. Modern Robotics: Mechanics, Planning, and Control. Cambridge University Press, 2017.

[4] Modern Robotics Python library. URL: https://github.com/NxRLab/ModernRobotics/tree/master/packages/Python.

[5] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. A Mathematical Introduction to Robotic Manipulation. CRC Press, 1994.

[6] Niryo One website. url: https://niryo.com/.

[7] Niryo Robotics on GitHub. URL: https://github.com/NiryoRobotics.

[8] Niryo Robotics. Niryo One Mechanical Specifications. 2018.

Received: 01.07.2024.
Accepted: 10.09.2024.

Primena teorije zavrtnja i njena implementacija u Pajtonu za upravljanje Niryo One manipulatorom

Položaj, odnosno, konfiguracija robota i njegova promena se može predstaviti na različite načine, od kojih svaki ima svoje prednosti i mane. U tu svrhu, u ovom radu se koristi teorija zavrtnja, koja kaže da se svako kretanje krutog tela može predstaviti jednom rotacijom i translacijom duž ose zavrtnja. Koristeći ovo kao osnovu, urađena je inverzna kinematika robota. Dok je procedura rešavanja direktne kinematike standardizovana, kod inverzne kinematike to nije slučaj. S obzirom na to, ovde je pokazan kombinovani pristup analitičkog i numeričkog načina rešavanja problema inverzne kinematike robotskog manipulatora Niryo One. Analitičko rešenje je izvedeno pojednostavljivanjem strukture robota, a zatim su ti rezultati korišćeni kao početno rešenje za Njutn-Rapsonovu numeričku metodu koja može da pronađe i do 8 validnih rešenja realnog robotskog manipulatora. Teorijska osnova se zatim implementira korišćenjem programskog jezika Python, nakon čega se rešenje šalje robotu preko Niryo One ROS API-a – pyniryo.

Ključne reči: **teorija zavrtnja, inverzna kinematika, Paden-Kahan podproblemi, Njutn-Rapsonova metoda, Pajton, Niryo One.**