NUMERICAL OPTIMIZATION – INVERSE KINEMATICS IN ROBOTIC MANIPULATORS

Tanyaradzwa Chinyai

# Introduction

## Abstract:

This paper investigates the efficiency of the Newton-Raphson method in solving inverse kinematics (IK) problems for robotic manipulators compared to analytical solutions. We demonstrate through computational experiments that the Newton-Raphson method offers a faster and more versatile approach to solving the IK. A case is made particularly for high-degree-of-freedom (DOF) robots where analytical solutions may not be readily available are computationally expensive. By contrasting the iterative nature of the Newton-Raphson method with the simultaneous computation of joint variables in analytical solutions, we highlight the superiority of the former in terms of computational speed and applicability to complex robot geometries. Our findings underscore the practical utility of numerical optimization methods in robotics, offering insights into their efficacy for solving IK problems in real-world applications. This review also makes an analysis of the disadvantages of using the Newton method for these robotic applications and how they are mitigated in industry according to the literature.

Keywords: Inverse kinematics; Optimization; Numerical approach; Analytical approach

## Review

The term robot is used to describe different things to different people. The term "robot" made its debut in the 1920 Czech science fiction play "Rossum's Universal Robots" by Karel Čapek, with the term coined by his brother Josef. In Czech, "robot" originally referred to serf labor but colloquially conveyed the notion of hard work or drudgery. The play featured artificial people or androids, a theme echoed in subsequent robot narratives where the robots often rebel, leading to dire consequences for humanity. Isaac Asimov's renowned robot series, spanning from 1950 to 1985, delved into the complexities of human-robot interactions and morality. These stories introduced robots with "positronic brains" programmed with the "Three Laws of Robotics," which have profoundly influenced later literature and films, shaping public perceptions of robots. [1] Cinema and fiction have strongly influenced what many people perceive a robot to be and its capabilities. Considering scientists have come far in developing the tools and mathematical mechanisms, which bring these perceptions to fruition; in reality, the practice of robotics is far behind their popular conception. It is, however, certain that this technology will be an important part of our lives in the coming years. This makes it important for researchers to leverage the current technological advancements to continue digging into the operation of robots and developing faster and more efficient ways to make them work.

This paper focuses on the kinematic algorithms used in the operation of industrial robot arms or manipulators. A manipulator is a mechanical device used to perform various tasks by manipulating objects within its workspace. It typically consists of a series of connected segments called *links*, which are actuated by *motors* or other actuators, and joints that allow relative motion between adjacent links. The end of the manipulator, known as the end effector, is used to interact with objects or perform tasks.

Here are some key characteristics and components of a robotic manipulator:

***Links***: Rigid segments that make up the structure of the manipulator. They are connected by joints and can vary in length and shape depending on the manipulator's design.
***Joints***: Mechanisms that enable relative motion between adjacent links. Common types of joints include revolute (rotational), prismatic (linear), and spherical (rotational with multiple axes).
***End Effector***: The tool or device attached to the end of the manipulator that interacts with objects or performs tasks. End effectors can include grippers, tools, sensors, cameras, or other specialized equipment.
***Degrees of Freedom (DOF)***: The number of independent ways the manipulator can move. Each joint typically contributes one degree of freedom to the manipulator's overall motion.
***Kinematics***: The study of motion without considering the forces that cause it. Robot kinematics involves determining the position, velocity, and acceleration of the manipulator's end effector based on the joint angles or displacements.
***Dynamics***: The study of forces and torques that cause motion. Robot dynamics involves understanding how external forces and torques affect the motion of the manipulator, as well as the internal forces and torques exerted by the actuators.

In order to design and operate a manipulator, the following steps need to be followed:

1. A forward Kinematic equation of the position of the joints needs to be deduced. Literature suggests several ways of approaching this (some methods are robust and popular, and some are still in development) including the Denavit-Hartenberg convention, Product of Exponentials (PoE) Method and geometric method. For this paper we worked with the geometric method for its simplicity.
2. Establish the method for solving inverse kinematic (IK). During the operation, depending on the application, welding, placing, etc. the operator, programmer will know the position which the robot hand needs to move to. These IK solutions translate the position of the end effector (hand) to the positions of the joints within the described motion. The solution to the IK can either be a closed form algebraic solution which is ideal for non-redundant robots - robots whose degree of freedom (rotational axes) directly corresponds

to the number of movements it can perform. The other kind of solution is a ***numerical solution***- the main subject of this paper. The inverse kinematics problem can be conceptualized as the task of refining the joint coordinates until the forward kinematics aligns with the desired pose. In a more formal sense, this can be viewed as an optimization problem, aiming to minimize the discrepancy between the forward kinematic solution and the desired position of the robot arm.

3. After having the inverse kinematics, construct the Jacobian matrix.
4. Establish the LaGrange equation in order to know the torques and forces required.
5. Design the controller accordingly.

While the above list covers significant aspects of the robotic design process, it's important to note that it's not exhaustive. However, it does encapsulate the fundamental concepts of the process, which designers typically consider. Literature often acknowledges the importance of considering all the mentioned aspects during the design phase.

# Method

The objective is to formulate the forward kinematic for a particular case of a 2 DOF robot then to calculate the inverse kinematic analytically and numerically using MATLAB. Then compare the speed of each application.
To demonstrate the usefulness of the numerical solution to the IK problem, the popular Newton-Raphson was chosen for a simple 2 DOF Planar manipulator (shown in Figure 1 below).

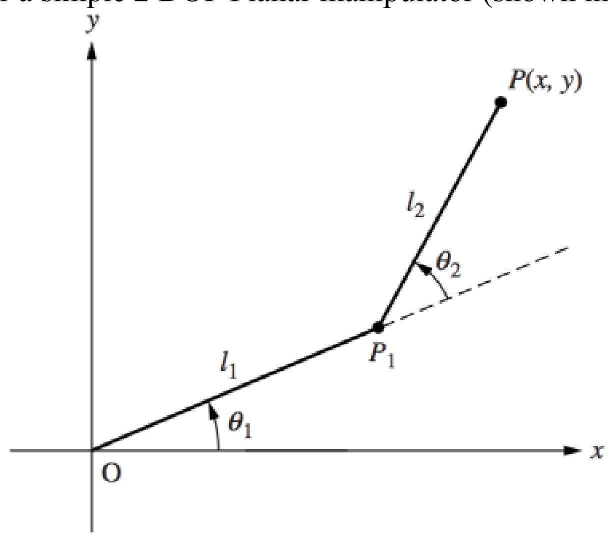


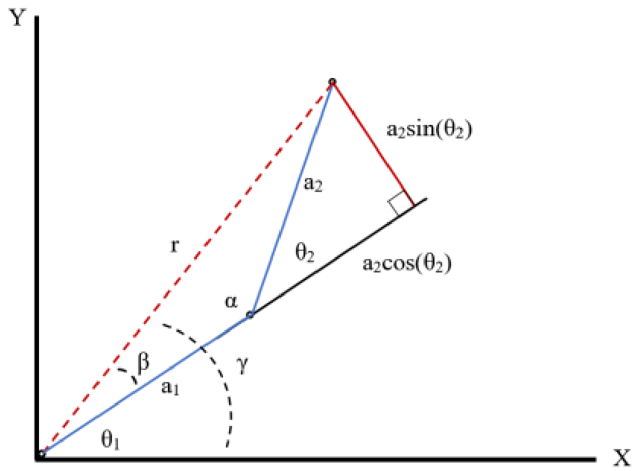

*Figure 1: Geometry of 2 DOF Planar Robot*

## Forward kinematic:

This is given by $P(x, y)$ which describes the x and y positions of the end effector with respect to the joint angles $\theta_1$ and $\theta_2$:

$$P(x,y) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) \\ l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2) \end{bmatrix} \tag{1}$$

## Analytical solution




If we wish to move the arm to a desired position, $\begin{bmatrix} x_d \\ y_d \end{bmatrix}$ it is required that the controller know the values of the joint angles. Analytically, the two equations for x and y are solved simultaneously. From figure 2 above, it can be deduced that:

$$\sin(\theta_2) = \sqrt{(1-\cos^2(\theta_2)^2)} \tag{2}$$

$$\gamma = \text{atan}\left(\frac{y_d}{x_d}\right) \quad \theta_1 = \gamma - \beta \tag{3}$$

From the above equations, the analytical inverse kinematic is found by:

$$\theta_2 = \mp\text{acos}\left(\frac{x^2+y^2-a_1^2-a_2^2}{2a_1a_2}\right) \tag{4}$$

$$\theta_1 = \text{atan}\left(\frac{y}{x}\right) \mp \text{atan}\left(\frac{a_2 sin\,(\theta_2}{a_1 + a_2\,cos(\theta_2)}\right) \tag{5}$$

Once these closed form equations are found, the values of $\theta_1$ and $\theta_2$ can be found at any point in the motion of the robot. Since the trigonometric functions are periodic and give a pair of solutions with each calculation; the solutions are subjected to constraints that filter out the solutions that are not in the operating range of the joint actuators.

## The Numerical Solution

The Numerical solution uses the Newton-Raphson method to minimize the error value which is the difference between the desired joint angles $\begin{bmatrix}\theta_{1D}\\ \theta_{2D}\end{bmatrix}$ and the current joint angles $\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}$. It is derived from the Newton's method and even though the two algorithms are often used interchangeably, they refer to slightly different things. The Newton-Raphson algorithm is a specific implementation of Newton's method for finding the roots of a function. It is expressed as follows:

$$\begin{bmatrix}\theta_{1(n+1)}\\ \theta_{2(n+1)}\end{bmatrix} = \begin{bmatrix}\theta_{1(n)}\\ \theta_{2(n)}\end{bmatrix} - \left(\left(\frac{\delta F(x)}{\delta x}\right)^{-1} |_{\theta_n}\right)\ (x - x_n) \tag{6}$$

The partial derivative evaluated at each angle is known as the Jacobian matrix, $J$, and the difference $(x - x_n)$ is the error value, which we will call $Err$. Therefore, the algorithm can be rewritten as:

$$\begin{bmatrix}\theta_{1(n+1)}\\ \theta_{2(n+1)}\end{bmatrix} = \begin{bmatrix}\theta_{1(n)}\\ \theta_{2(n)}\end{bmatrix} - (J^{-1} \cdot Err) \tag{7}$$

The Jacobian is the partial derivative with respect to $\theta_1$ and $\theta_2$ of the forward kinematic – eq(1). It is expressed in matrix form as:

$$J = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{8}$$

And the $Err$ value is equal to:

$$Err = \begin{bmatrix} x - x_d \\ y - y_d \end{bmatrix} = \begin{bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) - x_d \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) - y_d \end{bmatrix} \tag{9}$$

From equation (7), the value of the joint angles is calculated and contrasted against the speed of the analytical equation.

# Application

References

[1] https://link.springer.com/book/10.1007/978-3-319-54413-7

https://link.springer.com/article/10.1007/s11768-010-0011-1

https://link.springer.com/chapter/10.1007/978-3-030-40690-5_23