SOLVING FORWARD AND INVERSE KINEMATICS FOR ABB IRB 1410: THEORETICAL AND ANALYTICAL APPROACH

Abstract. A general theory for modeling industrial robots is discussed and performing forward kinematics in a generic manner using DH parameters was done. Velocity kinematics was also done using a theoretical approach. Following this, Inverse kinematics was solved using one of the theoretical approaches for the spherical wrist industrial manipulator ABB IRB 1410. This was done by building an analytical model for the first three joints followed by the other revolute joints and a spherical wrist. Finally, an analytical approach for inverse Kinematics using computational techniques and concept of Jacobian was done and was found to be accurate.

Keywords: Inverse Kinematics, Jacobian, Spherical Wrist.

1 Introduction

In today's era, we are living in a place where mathematical tools and concepts of Linear Algebra is an integral part in solving complex non-linear equations and Artificial Intelligence is the tool to make it easier. Every massive discovery in the field of sciences and mathematics has been possible because of data combined with the power of computing. With the assistance of tools, the capability of learning and solving takes a bigger leap to further simplification in understanding the concepts. The transformation of theoretical non-linear equations to easily solvable computational problems such as Jacobian formulation and iterative gradient descent has changed many industries. It is now helping them manufacture technology; making it accessible and economical.

Robots are considered as the basis for competitive manufacturing which purposes to yield high quality, productivity at nominal cost. All the necessary requirements for robots were initially developed with industrial applications in cognizance. These robots usually consist of a jointed arm/linked arm and an end effector. This manipulators in 3 or more axes are either fixed on surface or mobile for automation purpose. One major advantage of humans working with the industrial robot is that, each can accomplish what they are good at. For instance, robots do repetitive or risky works, while humans perform complex works and describe the work of the robot.

Other than Industrial purpose robots, there are other types of robots being used in many fields such as in Medical, Agriculture, Military, Health care, food Preparation, Travel etc. Transportation, handling and shipping robots are becoming essential to have for the retailers and logistics companies, due to which the packages are shipped to the

destination source earlier than expected. Robot-Assisted surgeries helps doctors to humans to recover from an injury. Automobile manufacturers such as Tesla, Ford, BMW are all working on the next wave of transport, which lets us to sit back and enjoy the ride, just with the combination of data science and robotics.

In this work, the process of modelling our robot using DH Parameters is discussed. The industrial manipulator or robot that will be used as the primary example will be ABB IRB 1410. It is a 6-axis industrial manipulator with a spherical wrist and is modelled for automation industries that manufacture products. It also communicates widely with other manipulators and is in sync with their activities. The manipulator is popular for its robust build and tenacity. It implicitly yields a longer lifespan without sporadically checking for maintenance and noise levels. The robot has a long reach. The flexible, but slim wrist provides immaculate performance and operation even in difficult and crowded locations. The ability to adjust position and the speed of each process is complemented by the compact design, that helps us in acquiring adequate accuracy with low loss. This makes it ideal for Arc Welding and Material Handling applications.

Inverse kinematics is the numerical interaction of computing joint variables that is expected to put the kinematic chain in a given position and orientation local to the position of the base of the manipulator. Kinematic chain motion analysis is the primary step to model any industrial manipulator. It helps us to study and determine the position of all joints and links in a manipulator or system. This information is also required to analyze controlled paths.

Thus, this work was inspired by a closely related work by Michael Norrlof, Department of Electrical Engineering at Linkoping University, where he has explained the modelling of ABB IRB 1400 and has clearly elaborated about modelling the industrial robot. Here, kinematic analysis was done using various inverse kinematic techniques and the kinematic equations are for defining the configuration of the framework. The nature and constraints on the independent parameters of the robot can be defined using non-linear loop equations. This can be collectively coined as the degree of freedom of the manipulator.

Several techniques for solving inverse kinematics have been used to find the desired configuration of the end-effector. All techniques can be broadly classified under three broad classifications: Analytical techniques, numerical techniques and hybrid techniques. Analytical techniques involve finding the feasible configurations as a function of the link lengths and rotation constraints, pertinent to the manipulator. However, we have to bank upon few assumptions to compute the configuration. Though these solutions exist for many kinematic chains, it’s not existent for all. However, it is widely used as they do not suffer from problem that arise due to singularity. Analytical approaches include Decoupling technique and Inverse Transform Technique. In this paper, we have elaborated the use of Decoupling technique.

There are many methods of modelling and obtain a numerical solution for inverse kinematics problems. The best ways involve iterative optimization and converging close to actual solution with a minor error less than a tolerance value. Other forms of solutions to solve include gradient descent techniques and Newton Raphson Techniques. In this work, we have used the Inverse Jacobian Technique.

There have also been some attempts to solve these problems using data driven techniques. Since Neural Networks are popular to solve gradient descent problems using backpropagation, this possibility has also been explored. The recent advances in image processing and motion detection have generated enough data that paved a way for these methods to solve the problem. The primary purpose to use these methods is to explicitly use observed positions and orientations so that the end effectors automatically find the feasible position learnt from the input.

Other methods to solve include some emerging hybrid methods that involve a combination of above discussed techniques. Some of these techniques involve use of frameworks that apply concepts of linear Algebra and optimization. These include solutions using interpolation of radial basis function as done by Charles Rose, Peter Sloan and Michael Cohen in their work. Similarly, Lee and Shin formulated an Inverse Kinematics solving approach that uses a hybrid numerical optimization technique, based on penalty configuration, and an analytical method meticulously planned to decrease the epochs for numerical optimization.

The solution to the inverse kinematics problem is imperative to plan the trajectory of motion of manipulators. Solving the inverse kinematics problem can be modified to solving a system of non-linear equations. Solving non-linear equations is difficult because of the high dependency of variables on each another. The slightest change in any one of the variables results in changes in the others' as well. Complex kinematic structures of robots make it difficult to find analytical inverse kinematics solutions. Though closed formed solutions have been derived by writing the Lagrangian multiplier function for the joint variables and optimizing it for a given end effector configuration subject to the joint constraints. However, the problem of singularity may creep in and give erroneous solutions.

Numerical inverse kinematics solutions are highly sensitive to the initial conditions and their convergence depends on the same. Inverse operations on matrices like in the Newton- Raphson method adds to computational expense. Adopting to numerical approach like the Newton - Raphson method leads us to dealing with singularities. Singularities occur if the determinant of the Jacobian matrix is zero. Other problems include a starting assumption, that may be far from actual solution leading to unwarranted iterations and converging only at one solution at a time out of several solutions.

In this work, we have used the decoupling technique as a part of theoretical solving and Inverse Jacobian Technique/ Newton-Raphson Method as a numerical technique. The core idea behind Jacobian technique is that, given an initial assumption for the joint

variables and try to converge the solution iteratively. Main goal is to minimize the error between the theoretical value and assumed value).

To do that we consider the norm of the difference between them. In decoupling technique, we decompose the transformation matrix into two matrices - position matrix and orientation matrix. Generally, in 6-R manipulators consists of a spherical wrist, the first three joints determine the position of the end effector and last three determine the orientation of the end effector. Since it's an Inverse kinematics problem, end effector co-ordinates are known to us, so the orientation matrix of the manipulator can be found. Finally, by applying mathematical rules/projections(trigonometry) we have determined all the joint variables.

2 Homogenous Transformation

While considering a rigid body in 3D space with a local frame moving and rotating in global co-ordinate frame, we can relate the two frames using the equation:

$$G_B = {}^{G}R_B * B_R + G_D \tag{1}$$

This can also be interpreted as:

$$G_B = {}^{G}T_B * B_R \tag{2}$$

Where:

$${}^{G}T_B = \begin{bmatrix} {}^{G}R_B & G_d \\ 0 & 1 \end{bmatrix} \tag{3}$$

The Homogenous transformation matrix is a 4x4 matrix that helps to co-relate a homogenous position from one frame to another. This extension of vector as the column space of a square matrix is just a way to simplify numeric calculations. The additional appended value to the three coordinate systems is called a scaling factor. For all practical problems, we will consider the value as unity.

This homogenous matrix can be further divided into a translation and a rotation matrix that enables us to represent the motion as a pure translation followed by pure rotation.

$${}^{G}T_B = {}^{G}D_B * {}^{G}R_B \neq {}^{G}R_B * {}^{G}D_B \tag{4}$$

Unlike the traditional rotation matrix to rotate about the Eigen vector axis in space, its inverse of the transformation matrix is not equal to its transpose and the matrix is no longer orthogonal.

$$ {}^{B}T_{G} = {}^{G}T_{B}^{-1} = \begin{bmatrix} {}^{G}R_{B}^{T} & -{}^{G}R_{B}^{T}G_{d} \\ 0 & 1 \end{bmatrix} \tag{5} $$

3 Denavit Hartenberg Parameters

In robotics, the Denavit Hartenberg parameters are four parameters primarily affiliated to a convention that is used to describe the position and orientation of the distal end of each link in space. These joints are part of a linked kinematic chain or manipulator and each of them is attached to its corresponding imaginary reference frame. This was introduced to make a universal coordinate frames for open kinematic chains.

A series manipulator with have n joints and n+1 links if we start numbering from the fixed link with numbers 0 and 1 for links and joints respectively. Thus, the opposite end of every link 'i' is attached to link 'i+1' at joint 'i+1' and the proximal end is attached to link i with the help of joint i. Every joint will have their own four DH parameters and their values determine the orientation of a link in space. The four DH parameters are:

1. Link length (denoted as a)
2. Link Twist (denoted as $\boldsymbol{\alpha}$)
3. Joint Offset (denoted as d)
4. Joint Angle (denoted as θ)

Determining the orientation of the local frame using DH convention is an easy task. The z-axis is coaxial with the axis of rotation of the joint 'i+1'. The x-axis is defined along the common normal between its own axis and the previous coordinate system's axis of rotation. the y-axis can be obtained with the cross product of the x-axis and z-axis using the right-hand thumb rule. Following the convention, we can determine the DH parameter table for ABB IRB 1410 as follows:

S No.	Link Length(a)	Link Twist ($\boldsymbol{\alpha}$)	Joint Offset (d)	Joint Angle (θ)
1	0	90	475	Θ
2	150	0	0	Θ
3	600	90	0	Θ
4	120	-90	720	Θ
5	0	90	0	Θ
6	0	0	85	Θ

Table 1. DH parameter table for ABB IRB 1410

The only parameter that is not design intrinsic is the joint angle. Hence, during the motion of the manipulator, the only parameter that we can tune during position analysis of every link is the joint angle of each DH parameter in the chain.

4 Analyzing Forward Kinematics

Forward kinematics basically helps us to determine the final end effector configuration by using the details of the position and orientation of each link. That is, given the position and orientation of all link in space, we can successively find the co-ordinates of the distal end of each link iteratively. While modelling the manipulator and performing forward kinematic analysis, we consider the manipulator as an open kinematic chain. This implies that the manipulator is formed by joining a set of links at the joints.

The crux of following DH convention lies in the fact that we can represent the position and orientation in a kinematic chain using the homogenous transformation matrix. It is not constant, but can be achieved as a function of the DH parameters and the transformation matrix can be written as:

$$ {}^{G}T_{B} = {}^{G}D_{Z,d} * {}^{G}R_{Z,\theta} * {}^{G}D_{x-1,a} * {}^{G}R_{x-1,\alpha} \tag{6} $$

Thus, the matrix that maps the successive local coordinates in a kinematic chain using homogenous transformation matrix with DH parameters as a function is:

$$ {}^{i-1}T_{i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & x_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & 0 \\ 0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7} $$

The generic code to do it in Python and MATLAB is included in Appendix 8.1.

5 Introduction to Velocity Kinematics

For velocity analysis, we can consider the final end effector orientation with some vector x. Intuitively, the velocity can be expressed as the derivative of x:

$$ \dot{x} = \frac{dx}{dt} \in R_m \tag{8} $$

Using this, we can express the forward kinematics as:

$$ x(t) = f(\theta(t)) \tag{9} $$

where θ is the function parameter of joint variables. Using the product rule of derivatives, we can further simplify the derivative as:

$$\dot{x}(t) = J(\theta(t))\frac{d\theta}{dt} \tag{10}$$

where J is called the Jacobian. The Jacobian matrix represents the partial derivative function or gradient of the joints end-effector velocity.

Writing the columns of J(θ) as:
$J_1(\theta), J_2(\theta), J_3(\theta), J_4(\theta), J_5(\theta), J_6(\theta)$ and the tip velocity as v_{tip}, we can derive it as an expression obtained by linear combination of columns of the Jacobian and joint velocity as follows:

$$\begin{aligned} v_{t_1 p} = {} & J_1(\theta) w_1(t) + J_2(\theta) w_2(t) + J_3(\theta) w_3(t) \\ & + J_4(\theta) w_4(t) + J_5(\theta) w_5(t) + J_6(\theta) w_6(t) \end{aligned} \tag{11}$$

When the joint angles become collinear i.e., if theta values are 0 or 180 degrees, then the Jacobian becomes a singular matrix. Such orientations are hence called as singularities. They are identified as a condition where the end effector is unable to move along an axis in which it's DOF is lost. Thus, as long as $J_1(\theta), J_2(\theta), J_3(\theta), J_4(\theta), J_5(\theta), J_6(\theta)$ are not collinear, it is feasible to have v_{tip} in any direction in the frame.

The code to compute all joint velocities using Jacobian in MATLAB is included in Appendix 8.2.

6 Introduction to Inverse Kinematics

Theoretically, Inverse Kinematics is just the opposite of forward kinematics. Given a kinematic chain's end effector position and orientation in space, we have to determine the angle in which each link has to be aligned to achieve the optimum end effector configuration. But the problem is not as simple as it sounds.

The functions that map the position and orientation of joints to its corresponding DH parameters is in terms of sines and cosines. Thus, it's mandatory to solve a system of non-linear equations to get the result. The variables of these equations are configuration parameters of the chain and the independent variables are same as the degree of freedom of the system. Essentially, the problem reduces to solving these set of equations and finding a feasible solution.

There are many ways to solve them theoretically and iteratively using computational techniques. Theoretical ways include Decoupling technique and Inverse Transform

Technique. Though these solutions exist for many kinematic chains, it's not existent for all. If the degree of freedom of manipulator exceeds the degree of freedom of the end effector (a manipulator with more than 6 DOF's in 3D space), there exists infinite solutions due to infinite linear combinations of dependent parameters.

ABB IRB 1410 is a 6 DOF manipulator in 3D space. Thus, we have 8 possible solutions. We can practically map it to three configurations: Left-handed or right-handed, elbow up or down (above or below shoulder level) and wrist orientation. Since each of them have two possibilities, we get $2^3 = 8$ theoretical solutions to achieve the required end effector position orientation.

6.1 Decoupling Technique

The decoupling technique is the most widely used analytical approach to solve inverse kinematics for robots with a spherical wrist. The joints that contain two coinciding DH local axes in the kinematic chain are referred as the wrist. The wrist joints are always revolute along two local axes. It's wise to attach a spherical wrist to the manipulator end to allow the optimum configuration of the end-effector. The intersection of the joints in a spherical wrist is called the wrist center point.

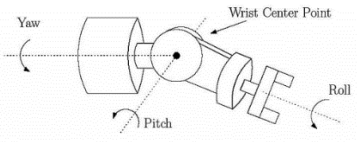

Fig. 1. Lateral View of Wrist Center Point

The overall homogenous transformation matrix of the manipulator can be decomposed into two matrices, one for position of the end effector multiplied with the other which denotes the orientation of the end effector.

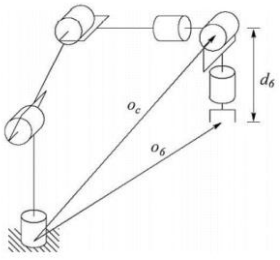

Fig. 2. Elbow Manipulator with Spherical Wrist

Given the 4 x 4 homogenous transformation *H*,

$$H = \begin{bmatrix} R_0^6 & o_6 \\ 0 & 1 \end{bmatrix} \tag{12}$$

According to the spherical wrist assumption, the end effector position will be determined by the first three joints and the end effector orientation will be determined by the last three joints. The process of decoupling is done as follows:

$$R_0^6 = R_0^3 R_3^6 \tag{13}$$

$$R_3^6 = \left[R_0^3 \right]^{-1} R_0^6 \tag{14}$$

$$R_3^6 = \left[R_0^3 \right]^{T} R_0^6 \tag{15}$$

Given the end effector position, $O_c = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$ we find R_3^0

θ_1 is the projection of the end effector onto the x-y plane.

$$\theta_1 = \tan^{-1} \frac{y_c}{x_c} \tag{16}$$

θ_3 is the projection of the end effector on the link 1 and link 2 plane.

$$\cos(\theta_3) = \frac{s^2 + r^2 - a_2^2 - a_3^2}{2a_2 a_3} \tag{17}$$
$$r^2 = x_c{}^2 + y_c{}^2; \qquad s^2 = z_c^2$$

θ_2 is the projection of the end effector on the link 1 and link 2 plane.

$$\theta_2 = \tan^{-1} \frac{s}{r} - \tan^{-1} \frac{a_3 \sin(\theta_3)}{a_2 + a_3 \cos(\theta_3)} \tag{18}$$
$$r^2 = x_c{}^2 + y_c{}^2; \qquad s^2 = z_c^2$$

Inverse orientation is obtained using the final orientation matrix and the rotation matrix of the first three joints.

$$ {}^{3}R_{6} = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_6S_4 - C_4C_5S_6 & C_4S_5 \\ C_4S_6 + C_5C_6S_4 & C_4S_6 - C_5S_6S_4 & S_4S_5 \\ -C_6S_5 & S_5S_6 & C_5 \end{bmatrix} \tag{19} $$

Thus expression for theta 4 to 6 is as follows:

$$ \theta_4 = \tan^{-1}\left(\frac{r_{23}}{r_{13}}\right) \tag{20} $$

$$ \theta_5 = \cos^{-1}\left(r_{33}\right) \tag{21} $$

$$ \theta_6 = \sin^{-1}\left(\frac{r_{32}}{\sin\left(\theta_5\right)}\right) \tag{22} $$

The shortcoming of the decoupling technique is the heavy dependence of variables on each other. As we see in the result obtained, the slightest change of value in any of the previously calculated joint variables will lead to vast differences in other joint variables.

The code to compute the optimal configuration using symbolic computation in MATLAB is included in Appendix 8.3.

6.2 Newton Raphson Technique

There are many methods of modelling and obtain a numerical solution for inverse kinematics problems. The best ways involve iterative optimization and converging close to actual solution with a minor error less than a tolerance value. In this work, we have used the Inverse Jacobian Technique.

It is a simple and effective way to find solution for a set of non-linear equations. It involves considering an initial assumption for the values of the target variables and iteratively converge to the solution. This is done by using the inverse of the Jacobian of the governing functions wrt. the independent parameters of these functions. In each iteration, we try to minimize the error between the assumption and the actual value by minimizing the norm of the difference between them.

In case of Inverse Kinematics, we know the configuration of the end effector. Thus, we know all the elements of the Homogenous matrix. We can also theoretically determine the end effector coordinate using the dependent parameter. Since joint angle is the tunable for all DH frames, the homogenous transformation matrix can be expressed as

a function of all thetas of the successive frames. Each element of the 4x4 matrix is a function for the Jacobian matrix and we can construct the matrix as the partial derivative wrt. each joint angle.

For ABB IRB 1410, we have six joints, implying we have six joint angles from θ_1 to θ_6. The assumption has to be a 6x1 for all values of theta and the Jacobian will be 16x6 matrix. The function can be expressed as a flattened version of the 4x4 HT matrix that will yield a 16x1 matrix. This has to be equated to the expected end effector position and orientation. To solve iteratively, we just substitute all the above values in the formula below:

$$x_{i+1} = x_i - J\left(x_i\right)^{-1} * F\left(x_i\right) \tag{23}$$

To check for convergence, we need to check when the difference between x_{i+1} and x_i is lesser than a desired threshold or if we exceed a desired number of iterations. Since we have multiple solutions, that is the robot is redundant, then the pseudo inverse finds a solution vector that has the smallest length among solutions. If robot achieves singularity configuration, then pseudo inverse minimizes error in the final equation.

The code to compute the same optimal configuration using random initialization and iteration in MATLAB is included in Appendix 8.4.

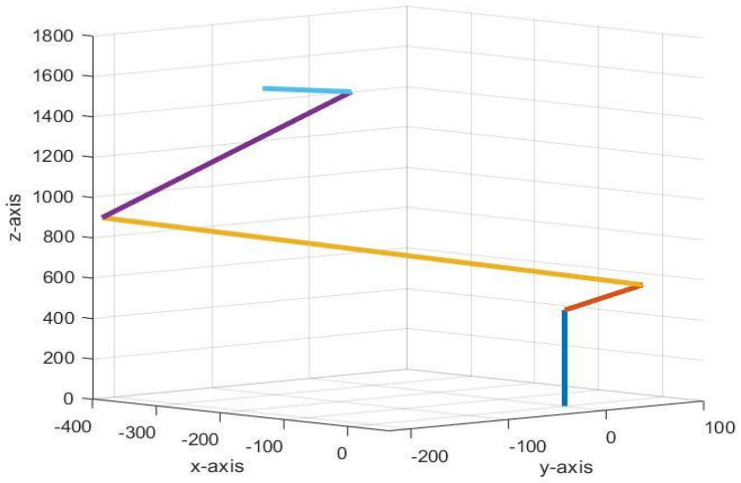

Fig. 3. Skeletal Frame of ABB IRB 1410

(Source: MATLAB output snippet)

7 Conclusion

Inverse Kinematics helps us in determining the correct angles in which we should position our manipulator so that we can reach the final required end effector co-ordinates. Integrating the theoretical solution using computational techniques can broaden its applications. Various fields can advance in areas such as automation, animation control, interactive manipulation, collision avoidance, self-driven cars and optimal configuration finding. Thus, this paper helps us understand that integrating the theory and practical applications in such arenas will open the domain of exploring a lot more in other areas.

8 Appendix

8.1 Robot specific and generic python code to implement forward kinematics:

```
import numpy as np
import matplotlib.pyplot as plt
class Forward_Kinematics:
    def __init__(self,joints):
        self.j = joints
    def rotateX(self,theta):
        return np.ar-
ray([[1,0,0,0],[0,np.cos(np.deg2rad(theta)),-
np.sin(np.deg2rad(theta)),0],[0,np.sin(np.deg2rad(theta))
,np.cos(np.deg2rad(theta)),0],[0,0,0,1]])

    def rotateZ(self,theta):
        return np.array([[np.cos(np.deg2rad(theta)),-
np.sin(np.deg2rad(theta)),0,0],[np.sin(np.deg2rad(theta))
,np.cos(np.deg2rad(theta)),0,0],[0,0,1,0],[0,0,0,1]])
    def TransX(self,val):
        mat = np.identity(4)
        mat[0][3]=val
        return mat

    def TransZ(self,val):
        mat = np.identity(4)
        mat[2][3]=val
        return mat
```

```
    def homoMat(self):
        dh_par = self.DH_Parameters()
        return np.matmul(np.mat-
mul(self.TransZ(dh_par[2]),self.ro-
tateZ(dh_par[3])),np.mat-
mul(self.TransX(dh_par[0]),self.rotateX(dh_par[1])))

 def finPos(self,mat):
        print(f"DH Parameters: \n {mat}" )
        finHomo = np.identity(4)
        homo_mat = np.zeros((4,4))
        for i in range(self.j):
            homo_mat = self.homo_Mat(mat[i,:])
            finHomo = np.matmul(finHomo,homo_mat)

        return np.round(finHomo,3)

#Robot Specific Case
IRB_1410 = np.array([[0, 90, 475, 30],[150, 0, 0, 60],[60
0, 90, 0, 90] ,[120, -
90, 720, 30],[0, 90, 0, 90],[0, 0, 85, 60]])
#DH parameters
total_joints = 6
model1 = Forward_Kinematics(total_joints)
model1.finPos(IRB_1410)

#Generic Case
total_joints = int(input("Enter total num-
ber of joints in Manipulator: "))
model2 = Forward_Kinematics(total_joints)
DH_params = np.zeros((total_joints,4))
for i in range(total_joints):
    print(f"Enter DH Parameters for Joint {i+1}:")
    for j in range(4):
        DH_params[i][j] = float(input())

print("Dh Parameters for Model: \n",DH_params)
model2.finPos(DH_params)
```

8.2 Code to determine velocity of End effector in MATLAB:

```
clc;clear all;close all;
syms th1 th2 th3 th4 th5 th6;
IRB_1410 = [0 90  475 th1; 150 0 0 th2; 600 90 0 th3; 120
-90 720 th4;0 90 0 th5;0 0 85 th6]
```

```
symbols = [th1; th2; th3; th4; th5; th6];
zeroTsix= eye(4);
for i=1:6
zeroTsix = zeroTsix * homoFromDH (IRB_1410(i,1),
IRB_1410(i,2), IRB_1410(i,3), IRB_1410(i,4));
end
angles = [30;60;90;30;90;60];
prac_vals = double(subs(zeroTsix,symbols,angles))
posVec = [ zeroTsix(13) zeroTsix(14) zeroTsix(15)]
velJacob = jacobian(posVec,symbols)
velJacob = double(subs(velJacob,symbols,angles))
angVel = [10;20;10;0;-10;0];
endVel = velJacob*angVel

function mat = zRotHomo(ang)
    mat = [cosd(ang) -sind(ang) 0 0; sind(ang) cosd(ang)
0 0;0 0 1 0;0 0 0 1];
end

function mat = xRotHomo(ang)
    mat = [1 0 0 0;0 cosd(ang) -sind(ang) 0 ;0 sind(ang)
cosd(ang) 0;0 0 0 1];
end

function homo = homoFromDH(a,alpha,d,theta)
homo = transl(0,0,d) * zRotHomo(theta) * transl(a,0,0) *
xRotHomo(alpha);
end
```

8.3 Theoretical model to determine solution to Inverse Kinematics using decoupling technique in MATLAB:

```
clc;clear all;close all;

d=0;
T1=dh_param(30,90,0,0.475);
T2=dh_param(60,0,0.15,0);
T3=dh_param(30,90,0.6,0);
T4=dh_param(60,-90,0.12,0.72);
T5=dh_param(30,90,0,0);
T6=dh_param(60,0,0,0.085+d); %d is the gripper length

final_matrix = T1*T2*T3*T4*T5*T6;
T03=T1*T2*T3;            % transfomation matrix from
frame 0 to frame 3(translation)
```

```
T36=T4*T5*T6;            % transfomation matrix from
frame 3 to frame 6(orientation)
rot_mat=final_matrix(1:3,1:3);
pos_mat=final_matrix(1:3,end);
a=T03(1,4);              % x coordinate of end effector
b=T03(2,4);              % y coordinate of end effector
c=T03(3,4);              % z coordinate of end effector

theta1=atan2d(b,a);
D=(a^2 +b^2 +(c-0.475)^2 - 0.6^2 -0.12^2)/(2*0.6*0.12);

theta3=abs(acosd(D));
theta2=atan2d(abs(sqrt((c-0.475)^2)),abs(sqrt((a^2
+b^2))))-
atan2d(0.12*sind(theta3),(0.6+0.12*cosd(theta3)));
inv_0R3=inv((T1*T2*T3));
RHS=inv_0R3(1:3,1:3)*rot_mat;
theta4=atan2d(RHS(2,3),RHS(1,3));
theta5=acosd(RHS(3,3));
theta6=asind(RHS(3,2)/sind(theta5));

disp("Theta1: "+theta1)
disp("Theta2: "+theta2)
disp("Theta3: "+theta3)
disp("Theta4: "+theta4)
disp("Theta5: "+theta5)
disp("Theta6: "+theta6)

function dh=dh_param(theta,alpha,a,d)
dh=transl(0,0,d)*trotz(theta,'deg')*transl(a,0,0)*trotx(a
lpha,'deg');
 end
```

8.4 Numerical Solution to determine solution to Inverse Kinematics using Iterative Newton Raphson Technique in MATLAB:

```
clc;clear all;close all;
syms th1 th2 th3 th4 th5 th6;
IRB_1410 = [0 90  475 th1; 150 0 0 th2; 600 90 0 th3; 120
-90 720 th4;0 90 0 th5;0 0 85 th6]
symbols = [th1; th2; th3; th4; th5; th6];
zeroTsix= eye(4);
for i=1:6
zeroTsix = zeroTsix * homoFromDH (IRB_1410(i,1),
IRB_1410(i,2), IRB_1410(i,3),IRB_1410(i,4));
```

```
end
angles = [30;60;90;30;90;60];
prac_vals = double(subs(zeroTsix,symbols,angles))
func = reshape(zeroTsix-prac_vals,[16,1])

%vary your assumption within permitted joint angle range
%assumption = [30 ;90; 60; 90; 30; 45];
assumption = [0 ;90; 0; -90; 60; 30];
tol = 0.001;numIter = 100;
jac = jacobian(reshape(zeroTsix,[16,1]),symbols);
for i=1:numIter
    hist = assumption;
    mat = double(subs(jac, symbols,assumption));
    func_val = double(subs(func,symbols,assumption));
    assumption = assumption - pinv(mat)*func_val;
    if(norm(hist-assumption)<tol)
        break
    end
end
disp("Final Values:")
assumption = mod(assumption,360)
disp("Iterations taken: ")
disp(i)
calculated = double(subs(zeroTsix,symbols,assumption))
IRB_1410 = double(subs(IRB_1410,symbols,assumption))
x = linspace(-1500,1500,3001);
y = linspace(-1500,1500,3001);
[X,Y]=meshgrid(x,y);
HT= eye(4);
x_start = 0; y_start = 0; z_start = 0;

for i=1:6
HT = HT * homoFromDH(IRB_1410(i,1), IRB_1410(i,2)
,IRB_1410(i,3), IRB_1410(i,4));
plot3([x_start HT(13)],[y_start HT(14)],[z_start HT(15)],
'MarkerSize',7); grid on;hold on
    x_start = HT(13);y_start = HT(14);z_start = HT(15);
end

hold off

function mat = zRotHomo(ang)
    mat = [cosd(ang) -sind(ang) 0 0; sind(ang) cosd(ang)
0 0;0 0 1 0;0 0 0 1];
end
```

```
function mat = xRotHomo(ang)
    mat = [1 0 0 0;0 cosd(ang) -sind(ang) 0 ;0 sind(ang)
cosd(ang) 0;0 0 0 1];
end

function homo = homoFromDH(a,alpha,d,theta)
    homo = transl(0,0,d) * zRotHomo(theta) *
transl(a,0,0) * xRotHomo(alpha);
end
```

9 References

1. A Generalized Matrix Inverse with Applications to Robotic Systems Bo Zhang and Jeffrey Uhlmann Dept. of Electrical Engineering & Computer Science University of Missouri-Columbia
2. Linear algebra applied to Kinetic Control of Mobile Manipulators; Victor H. Andaluz, Edison R Sasig, William D. Chicazia and Paola M. Velasco.
3. Moravec, Hans Peter. "Robot". Encyclopedia Britannica, 4 Feb. 2021.
4.) Ijeoma, Muzan & Faisal, Tarig & Al-Assadi, Hayder & Solihin, Mahmud Iwan. (2012). Implementation of Industrial Robot for Painting Applications. Procedia Engineering. 41. 1329-1335. 10.1016/j.proeng.2012.07.318.
5. A. Grau, M. Indri, L. L. Bello and T. Sauter, "Industrial robotics in factory automation: From the early stage to the Internet of Things," IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, 2017, pp. 6159-6164, doi: 10.1109/IECON.2017.8217070.
6. Modeling of Industrial Robots, Michael Norloff, Dept. of Electrical Engineering, Linkoping University.
7. Inverse kinematics techniques in Computer graphics: A survey. By: A Aristidou, J. Lasenby, Y. Chrysanthou, A. Shamir ; Computer graphics Forum, 2017
8. A closed form Solution for inverse Kinematics of Robot Manipulators with Redundancy Pyung H Chang, IEE Journal of robotics and automation, October 1987
9. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm, Adrian-Vasile Duka, 7th International Conference Interdisciplinary in engineering, 2013
10. A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators Li-Chun Tomy Wang and Chih Cheng Chen, IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. I, NO. 4, AUGUST 1991
11. Artist-Directed Inverse- Kinematics Using Radial Basis Function Interpolation, Charles F Rose, Peter-Pike J. Sloan, Michael F. Cohen, Eurographics Vol. 20 Number 3, 2001.
12. Kucuk, S., & Bingul, Z. (2014). Inverse kinematics solutions for industrial robot manipulators with offset wrists. Applied Mathematical Modelling, 38(7-8), 1983–1999.
13. Li, Y., Wei, Y., & Chu, Y. (2015). Research on Solving Systems of Nonlinear Equations Based on Improved PSO. Mathematical Problems in Engineering, 2015, 1–13.
14. For pictures:
    feng.stafpu.bu.edu.eg/Electrical%20Engineering/823/crs-14863/Files/lecture7_InverseKinematics.pdf