Kinematics & Dynamics Library for Baxter Arm

Aakash Murugan, Ashwin Sahasrabudhe, Akshay Kumar, Chaitanya Perugu, Sanjuksha Nirgude

***Abstract*— The Baxter robot is a 14-DOF dual-arm standard research platform used widely in research tasks along with the Baxter SDK provided by the developers, Rethink Robotics. Despite the ubiquitous use of the robot, the official software support is sub-standard. Especially, the native IK service has a low success rate and is often inconsistent. This unreliable behavior makes Baxter difficult to use for experiments and the research community is in need of a more reliable software support to control the robot. We propose to create a Python based library supporting the kinematics and dynamics of the Baxter robot with extended options for arm control and cross-platform usability.**

***Keywords* — Baxter Research Robot, Manipulator Kinematics, Iterative IK, Dynamical Model, Redundant Manipulator**

I. INTRODUCTION

A. Introduction to Baxter

Baxter is a dual arm humanoid robot developed by Rethink Robotics Inc., with 7 degrees of freedom on each arm manipulator, hence, falling under the *kinematically redundant* category. Baxter's arms are actuated by Serial Elastic Actuators (SEAs) which provide inherent compliance for safety purposes. Baxter has three offsets in its arm kinematic structure. Each 7-DOF arm has a 2-DOF offset shoulder, a 2-DOF elbow and a 3-DOF offset wrist. Both arms include angle position and joint torque sensing. Due to the offsets, no three consecutive frames meet at a common origin, hence, according to Pieper's principle, there is no analytical solution for Inverse Pose Kinematics (IPK) of Baxter's arm manipulator [8].

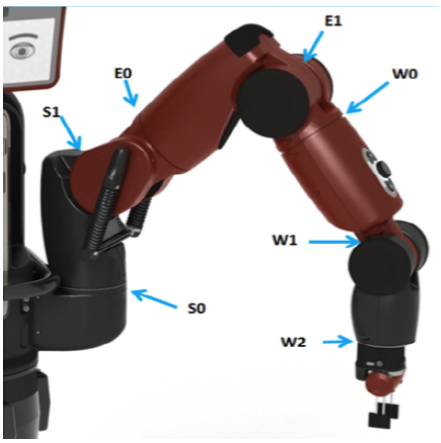

Fig. 1: Baxter's 7-DOF arm

In Fig. 1 the shoulder joints are represented by S0 (shoulder roll) and S1 (shoulder pitch), elbow joints by E0 (elbow roll) and E1 (elbow pitch), and wrist joints by W0 (wrist roll), W1 (wrist pitch) and W2 (wrist roll).

Planning a robot's motion requires understanding of the relationship between the actuators we can control and the robot's resulting position in the environment. *Forward Kinematics* of a robotic arm is the process used to find the position of the end-effector of the robot using the knowledge of the angle of each joint. If we need to find the angle of each joint for a particular end-effector position, we need to invert this relationship. This process is known as *Inverse Kinematics*. *Dynamics* of a robot provides the relationship between actuations and contact forces, acceleration and the resulting motion trajectories. Through this project, we make this process easier for the user of the Baxter robot by developing a Python library to perform the forward and inverse pose kinematics and dynamics of a well-known humanoid robot, Baxter.

B. Kinematics

Robot mechanisms consist of number of rigid bodies connected by joints. The position and orientation of these rigid bodies in space are together termed as a *pose*. The robot kinematics describes the pose, velocity and acceleration of the rigid bodies that make up their robot mechanism. A *kinematic joint* is a connection between two bodies that allows for relative motion. Kinematics of the robot consists of two processes, forward pose kinematics and inverse pose kinematics. In forward pose kinematics of a serial manipulator, we find the position and orientation of the end-effector frame of interest from the joint values. In the process of inverse kinematics, we find the values of joint positions that result in the given values of position and orientation of the end-effector relative to the base frame.

C. Dynamics

Dynamics of a robot describes the relationship between joint actuator torques and the resulting motion. The dynamic equations of motion form a basis of many computations in mechanical design, controls and simulation [9]. This equation of motion consists of joint space positions, velocity, acceleration and force vectors.The common form of a serial manipulator's dynamical model is in the following state space form.

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau$$

In the equation, q denotes the vector of joint angles, $M(q)$ is the inertia matrix, $C(q,\dot{q})$ denotes the Coriolis matrix and the $G(q)$ represents the gravitational vector. The equation equates to the vector of actuator torques. There are two

All the authors are with the Department of Robotics Engineering, Worcester Polytechnic Institute, USA

approaches towards the dynamical modeling of a system, the Euler-Lagrange approach and the Newton-Euler approach. In Euler-Lagrange model, the links are considered together and the model is obtained analytically using kinetic and potential energy of the system.The Newton-Euler method forms an equation with recursive solution. The Forward Kinematics equation defines a function between space of Cartesian poses and the space of joint poses. The velocity relationship are determined by the Jacobian of this function [10].

II. BACKGROUND

The first step to controlling a robot is to understand the mathematical model of the system. Baxter has been designed to operate safely in collaboration with humans, and, as a result, has become one of the most used research platforms in robotics labs. Consequentially, many groups has performed research on Baxter's kinematics and dynamics. Silva, et al., explored the forward kinematics of Baxter's arms, and the workspace defined by the arm span [1]. Smith, et al., derived dynamic equations for the arms, validated by measuring torques throughout pre-defined trajectories [2]. Silva and Ju, et al. also derived kinematic models of the arms and modeled them in MATLAB [3].

The forward kinematics of a serial manipulator is a very well-established concept in robotics research, the most commonly used technique being Denavit-Hartenberg parameters [4]. Inverse Kinematics is more complicated, as there is no universal mechanism to derive the inverse kinematics equations of a manipulator. In fact, closed-form solutions do not even exist for redundant (7-DOF or more) manipulators. Most work on inverse kinematics of redundant robots focus on iterative or numerical approaches [5] [6] [7], which suffer from their own share of issues. Dr. Williams II of Ohio University penned an excellent article discussing the 7-DOF Baxter arm kinematics [8].

The dynamics of Baxter's arm has been a subject of research too, albeit less than its kinematics. The mathematical model of Baxter's dynamics is extremely complicated, with over "half a million coefficients", as reported by Yang, et al. [9]. "Advanced Technologies in Modern Robotics Applications" [10] also discusses the Euler-Lagrange method for dynamical modeling of Baxter, which we will use as our primary reference for this purpose.

III. PROJECT GOALS

We are planning to develop our own software package for the Baxter 7-DOF arm. We will perform kinematics analysis and dynamic modeling of the 7-DOF arm to create a software library that can achieve consistent and reliable performance and would help other robotics researchers working with Baxter. We have defined our expected goals and reach goals as in Table 1.

The expected goals consist of Forward Kinematics, Inverse Kinematics, calculation of MCG matrices and Jacobians. If everything goes well, we would like to extend our project to include Position and Trajectory control, Motion Planning and ROS support.

TABLE I: PROJECT GOALS

Expected Goals	Reach Goals
Forward Kinematics	Position Control
Inverse Kinematics	Trajectory Control
M C G Matrices	Motion Planning
Velocity Kinematics	ROS Support

A. Forward Kinematics

In order to compute the forward kinematics for a robot mechanism in a systematic manner, one should use a suitable kinematics model. We are planning to compute the forward pose and velocity kinematics of the 7-DOF arm. For the inverse Kinematics, since 7 DOFs introduce redundancy in the arm, we would like to freeze one arm joint and then work on the inverse kinematics of 6-DOF arm as the first goal. The next goal is to take the redundancy into consideration and use the complete 7-DOF arm for the calculation of inverse pose and velocity kinematics.

The library will be in such a way that it will output in different formats as specified by the user. It would have the ability to output translation vector, transformation matrix, rotation matrix, Euler angles and quaternions. We are planning to perform kinematics and dynamics analysis to one arm of the Baxter robot and then use it for the second arm as the second arm is simply a mirror image of the first arm.

B. Dynamics

We need the Mass-inertia, Coriolis co-efficient and the Gravity co-efficient matrix for the computation of the dynamical model of the robot manipulator. For the calculation of these matrices, we plan to use Euler-Lagrange method with the help of mechanical parameters from the URDF files. The Jacobian is used for calculating the velocity forward and inverse kinematics of the Baxter robot manipulator. We would calculate the Jacobian by differentiating the Forward Kinematics Equations.

C. Controls

If possible, we will use the dynamical model of the system to design the controller for the position and trajectory control of the system. For the position control, we are planning to use PD/PID control with shiver prevention and gravity compensation, whereas for the trajectory control, we are planning to develop computed torque control. Trajectory control is used for starting from a configuration and reaching a goal via a given set points.

IV. CURRENT WORK

The Baxter robot is endowed with a proficient SDK provided by its makers, Rethink Robotics. The native support uses Gazebo as the simulation environment. Moreover, it has several interfaces for tele-operation of the simulation model as well as the real-life model of Baxter. Here, we discuss our approaches to solve the various expected and reach goals that we have put forth for the project. The various subsections discuss the methodology, tools and resources that we used.

A. Denavit-Hartenberg Parameters

DH parameters are a set of four variables that define the spatial relationship between two valid coordinate frames. These variables are d (translation along old z), θ (rotation about old z), a (translation along new x) and α (rotation about new x). These parameters are combined using the DH framework to derive the transformation matrix that links the two coordinate frames through matrix multiplication.

$$T_n^{n-1} = Rot_z(\theta) \cdot Trans_z(d) \cdot Trans_x(a) \cdot Rot_x(\alpha)$$

The table below shows DH parameters as calculated for the Baxter robot's left arm.

TABLE II: DH PARAMETERS FOR BAXTER LEFT ARM

Link	d (m)	θ (rad)	a (m)	α (rad)
S0-S1	0.27035	θ_0	0.069	$-\pi/2$
S1-E0	0	$\theta_1 + \pi/2$	0	$\pi/2$
E0-E1	0.36435	θ_2	0.069	$-\pi/2$
E1-W0	0	θ_3	0	$\pi/2$
W0-W1	0.37429	θ_4	0.01	$-\pi/2$
W1-W2	0	θ_5	0	$\pi/2$
W2-EE	0.229525	θ_6	0	0

B. Forward Pose Kinematics

Baxter's forward pose kinematics (FPK) equations give the 6-DOF pose (3 position and 3 orientation) of the end-effector as a function of the seven joint angles of the arm manipulator. Computing the DH parameters is the first step towards deriving the closed-form FPK equations.

Forward kinematics is calculated using the transformation matrices. The transformation matrix from the base to the tip frame using the several intermediate transformation matrices is given as:

$$T_7^0 = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_4^3 \cdot T_5^4 \cdot T_6^5 \cdot T_7^6$$

The transformation matrices used in the above equation make use of the DH parameters as given in the generalized formula for Transformation matrix T_n^{n-1}. The final end-effector position equations as a function of the joint angles, derived from Forward Kinematics results are given below.

$$X_{ee}^b = d_1.c_0 - l_4(c_5(s_0s_2s_3 - c_0c_1c_3 + c_0c_2s_1s_3) + s_5(c_4(c_0c_1s_3 + c_3s_0s_2 + c_0c_2c_3s_1) + s_4(c_2s_0 - c_0s_1s_2))) - l_3(s_0s_2s_3 - c_0c_1c_3 + c_0c_2s_1s_3) - d_3.s_4(c_2s_0 - c_0s_1s_2) + l_2.c_0c_1 - d_2.s_0s_2 - d_3.c_4(c_0c_1s_3 + c_3s_0s_2 + c_0c_2c_3s_1) + d_4.c_6(s_5(s_0s_2s_3 - c_0c_1c_3 + c_0c_2s_1s_3) - c_5(c_4(c_0c_1s_3 + c_3s_0s_2 + c_0c_2c_3s_1) + s_4(c_2s_0 - c_0s_1s_2))) + d_4.s_6(s_4(c_0c_1s_3 + c_3s_0s_2 + c_0c_2c_3s_1) - c_4(c_2s_0 - c_0s_1s_2)) - d_2.c_0c_2s_1$$

$$Y_{ee}^b = l_4.(c_5(c_1c_3s_0 + c_0s_2s_3 - c_2s_0s_1s_3) - s_5(c_4(c_1s_0s_3 - c_0c_3s_2 + c_2c_3s_0s_1) - s_4(c_0c_2 + s_0s_1s_2))) + d_1.s_0 + l_3(c_1c_3s_0 + c_0s_2s_3 - c_2s_0s_1s_3) + d_4.s_6(s_4(c_1s_0s_3 - c_0c_3s_2 + c_2c_3s_0s_1) + c_4(c_0c_2 + s_0s_1s_2)) + d_3s_4(c_0c_2 + s_0s_1s_2) + d_2.c_0s_2 + l_2.c_1s_0 - d_3.c_4(c_1s_0s_3 - c_0c_3s_2 + c_2c_3s_0s_1) - d_4.c_6(s_5(c_1c_3s_0 + c_0s_2s_3 - c_2s_0s_1s_3) + c_5(c_4(c_1s_0s_3 - c_0c_3s_2 + c_2c_3s_0s_1) - s_4(c_0c_2 + s_0s_1s_2))) - d_2.c_2s_0s_1$$

$$Z_{ee}^b = l_1 - l_2.s_1 - d_2.c_1c_2 - l_3.c_3s_1 - l_3.c_1c_2s_3 - l_4.c_3c_5s_1 + d_3.c_1s_2s_4 + d_3.c_4s_1s_3 - d_3.c_1c_2c_3c_4 - l_4.c_1c_2c_5s_3 + d_4.c_1c_4s_2s_6 + d_4.c_3c_6s_1s_5 + l_4.c_1s_2s_4s_5 + l_4.c_4s_1s_3s_5 - d_4.s_1s_3s_4s_6 - l_4.c_1c_2c_3c_4s_5 + d_4.c_1c_2c_3s_4s_6 + d_4.c_1c_2c_6s_3s_5 + d_4.c_1c_5c_6s_2s_4 + d_4.c_4c_5c_6s_1s_3 - d_4.c_1c_2c_3c_4c_5c_6$$

The above equations can be used directly to get the end effector positions by feeding in the inputs for the joint angles. The terms c_i and s_i in the equations denote $cos(\theta_i)$ and $sin(\theta_i)$ respectively, while l_i and d_i refer to the lengths and offsets at each set of links. Note that d_4 for Baxter is zero.

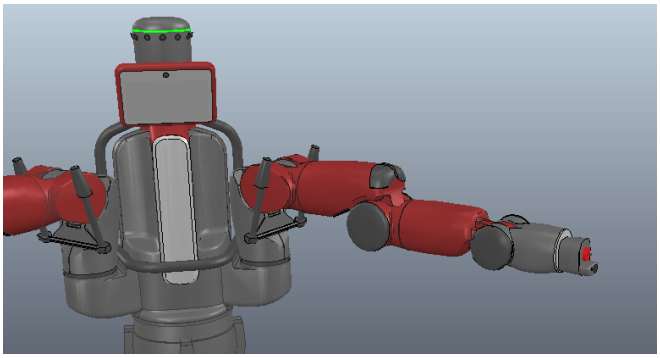

Fig. 2: Actual Baxter simulation model

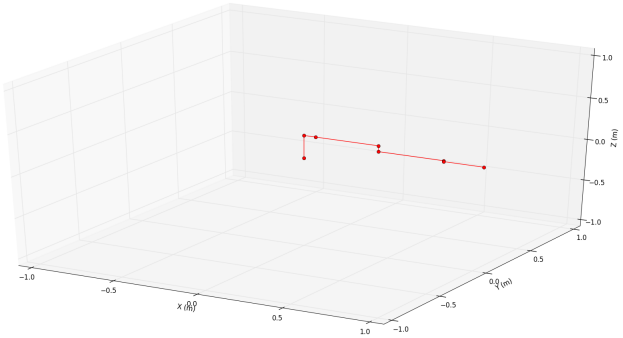

Fig. 3: Our representative skeleton model

C. Skeleton Model

In order to visualize our conceived methodology and test its working, we devised a skeletal model of the Baxter arm. Based on the DH parameters, we obtained a 3D stick diagram of the arm using the *matplotlib* library in Python, showing the Cartesian configuration (position and orientation) of the various links in a given joint space configuration. We ensured its credibility by visually comparing the same with the Baxter model in simulation provided by Rethink Robotics. Figure 2 and 3 show the Baxter arm in the home configuration in V-REP simulation environment and corresponding visualization of the developed skeletal model.

D. Workspace Analysis

Baxter arm's redundancy and kinematic structure enables it to get rid of singularities, evidently making it capable of having a large workspace. Despite the joints of Baxter not

having full 360 degree range of motion, the seven degrees of freedom and the presence of a non-spherical wrist make it possible for the robot to eventually get to any point within the bounding canopy formed by maximum reach position of the EE over complete range of motion for the joints.

We exploit the developed skeletal model to obtain the canopy of end-effector positions that encase the complete workspace. The flow-chart to demonstrate workspace calculation is shown in Figure 4. However, since we weren't able to incorporate the self-collision space in the rudimentary skeletal model, the resultant workspace tends to have those regions that are practically not reachable by the Baxter arm. Figure 5 shows the workspace derived by our model.

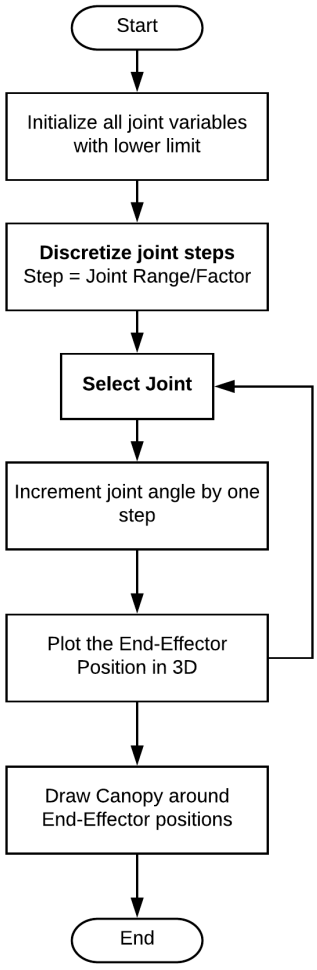

Fig. 4: Workspace Calculation

E. Jacobian Matrix

While forward position kinematics enables us to calculate the pose of the end-effector at all times, velocity kinematics is much more useful and powerful in the context of manipulator control. Velocity kinematics relates the joint velocities and the end-effector velocities. Interestingly, although the

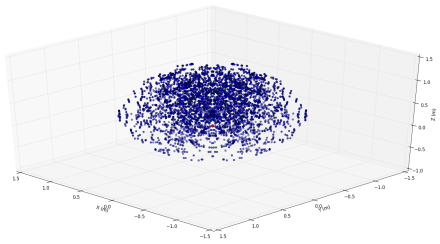

Fig. 5: Workspace Diagram of Baxter's left arm

forward pose kinematics of a robot manipulator is non-linear, the velocity kinematics is always linear.

The established approach to mapping the velocities in the joint space and task space is the Jacobian matrix. The Jacobian is a $n \times m$ matrix (where n is the dimension of the task space and m is the joint space dimension), computed through the partial differentiation of the FPK equations, as shown below.

$$J(q) = \begin{bmatrix} A(q) \\ B(q) \end{bmatrix} = \begin{bmatrix} \frac{\partial x_t(q)}{\partial q_1} & \frac{\partial x_t(q)}{\partial q_2} & \dots & \frac{\partial x_t(q)}{\partial q_m} \\ \xi_1 z_1 & \xi_2 z_2 & \dots & \xi_m z_m \end{bmatrix}$$

Here, $x_t(q)$ is the $n \times 1$ vector-valued FPK equation, z_k is the k^{th} joint rotation axis in the base frame and ξ_k represents the type of the k^{th} joint (0 for prismatic and 1 for revolute).

The arm manipulator of the Baxter research robot has 7 joints, all of them revolute. Hence, the joint space dimension m is 7 and the task space dimension n is 6 while all ξs are 1. For the purpose of this project, computing the Jacobian matrix for Baxter has been done through MATLAB Symbolic Math Toolbox since it is extremely complicated to compute by hand.

F. Velocity Kinematics

The Jacobian matrix is central to the velocity kinematics of a serial manipulator. Let x represent the 6×1 end-effector pose in task space and q refer to the 7×1 vector of joint angles. Then, $\dot{x}$ and $\dot{q}$ are the velocities in the task space and the joint space respectively. The Jacobian matrix $J(q)$ relates the two velocities as follows:

$$\dot{x} = J\dot{q}$$
$$\dot{q} = J^{\dagger}\dot{x}$$

Here, $J^{\dagger} = J^T(JJ^T)^{-1}$ is the Moore-Penrose pseudoinverse of J, a more generalized matrix inverse for non-square matrices. The above equations form the set of velocity kinematics equations, the first representing the forward velocity kinematics (FVK) while the second is the inverse velocity kinematics (IVK).

The Jacobian matrix $J(q)$ plays a big role in the kinematic and dynamic analysis of a serial manipulator, hence, computing it is an important step. Since it relates the task space velocities and the joint space velocities, the Jacobian can be used to measure the *manipulability* of a joint pose, i.e. the ability to apply forces or velocities in different directions.

Singularities are joint poses which do not allow for task space velocities in certain directions, essentially reducing the effective degrees of freedom of the manipulator. The Jacobian provides a simple way to measure the closeness of a joint pose to a singularity, thereby, providing ample warning to stay away from such configurations. Such measures are called *manipulability indices*.

One of the most widely-used is the *Yoshikawa's manipulability index*, defined as $\sqrt{|JJ^T|}$ which goes to zero as the manipulability of the joint pose decreases. As part of our library implementation, we compute the Yoshikawa's index whenever we (re)compute the Jacobian matrix. This enables us to warn the users whenever Baxter's joint pose is approaching a singular configuration.

G. 6-DOF Inverse Pose Kinematics

Before going towards control of the 7-DOF redundant Baxter arm, we first work on only 6 degrees of freedom by freezing the joint E0. In this section, we perform inverse kinematics on 6-DOF Baxter robot arm. As we have locked the joint E0 we have $\theta_3 = 0$. Additionally, we take another reasonable approximation of $L_5 = 0$. We also make use of a new parameter L_h to simplify the results, defined as

$$L_h = \sqrt{L_2^2 + L_3^2}$$

We perform the forward position kinematic of the 6-DOF arm. This is got by substituting the values as above in the FPK equations. We get the end-effector pose T_6^0 from the matrix as follows.

$$T_6^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_6^0 \\ r_{21} & r_{22} & r_{23} & y_6^0 \\ r_{31} & r_{32} & r_{33} & z_6^0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Our 6-DOF analytical solution basically maps from this Transformation matrix T_6^0 to the joint angles $\theta_1, \theta_2, \theta_4, \theta_5, \theta_6, \theta_7$. The position vector in this case is given as

$$\{P_6^0\} = \begin{bmatrix} x_6^0 \\ y_6^0 \\ z_6^0 \end{bmatrix} = \begin{bmatrix} c_1(L_1 + L_h c_2 + L_4 c_{24}) \\ s_1(L_1 + L_h c_2 + L_4 c_{24}) \\ -L_h s_2 - L_4 s_{24} \end{bmatrix}$$

Therefore the joint angles $\theta_1, \theta_2, \theta_4$ are solved analytically as follows

$$\theta_1 = atan2(y_4^0, x_4^0)$$

$$\theta_{2_{1,2}} = 2\tan^{-1}(t_{1,2})$$

$$\theta_{4_{1,2}} = atan2(-z_6^0 - L_h s_{2_{1,2}}, \tfrac{x}{c_1} - L_1 - L_h c_{2_{1,2}}) - \theta_{2_{1,2}}$$

where the variables E, F, G are

$$E = 2L_h(L_1 - \tfrac{x}{c_1})$$

$$F = 2L_h z$$

$$G = (\tfrac{x^2}{c_1^2} + L_h^2 + L_1^2 - L_4^2 + z^2 - 2\tfrac{L_1 x}{c_1})$$

and the variable $t_{1,2}$ is

$$t_{1,2} = \tfrac{-F \pm \sqrt{E^2 + F^2 - G^2}}{G - E}$$

Therefore, we get two possible solutions for the angles from the translational components as shown in the table below.

Sol. 1	θ_1	θ_{2_1}	θ_{4_1}
Sol. 2	θ_1	θ_{2_2}	θ_{4_2}

These two solutions share the same common θ_1 and (θ_2, θ_4) pairs corresponding to the elbow up and elbow down configurations.

Now for the solutions of the joint angles θ_5, θ_6 and θ_7 we take use of the rotational matrix R_6^0 and the previous results $(\theta_1, \theta_2, \theta_4)$ into consideration. Since θ_5, θ_6 and θ_7 are found only in the R_6^3, we find R_6^3 from R_6^0 as follows:

$$R_6^3(\theta_5, \theta_6, \theta_7) = [R_3^0(\theta_1, \theta_2, \theta_4)]^T [R_6^0] = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

$$R_6^3(\theta_5, \theta_6, \theta_7) = \begin{bmatrix} -s_5 s_7 + c_5 c_6 c_7 & -s_5 c_7 - c_5 c_6 s_7 & c_5 s_6 \\ s_6 c_7 & -s_6 s_7 & -c_6 \\ c_5 s_7 + s_5 c_6 c_7 & c_5 c_7 - s_5 c_6 s_7 & s_5 s_6 \end{bmatrix}$$

Therefore, using R_6^3, we can find θ_5, θ_6 and θ_7 from the equations given below.

$$\theta_5 = atan2(R_{33}, R_{13})$$

$$\theta_6 = atan2(\tfrac{R_{21}}{c_7}, -R_{23})$$

$$\theta_7 = atan2(-R_{22}, R_{21})$$

H. Iterative Solvers for Redundant IPK

1) Jacobian Pseudoinverse: Computation of the pseudoinverse for non-square Jacobian matrices gives us the inverse velocity kinematics for the robot. The position kinematics is then obtained by integrating the velocity kinematics over several time steps. The pseudoinverse approach to iterative IK starts with taking the joint angle positions for the current configuration as the seed angles for integration over time.

The algorithm is run repeatedly with a value of $\dot{x}$ taken as a small vector in the direction of the vector joining the current end-effector position to the target end-effector position. Thereafter, using the obtained Jacobian Pseudoinverse, we compute the joint angular velocity. Integrating it over a constant value of time-step and comparing the obtained final Cartesian pose with the target pose, we gradually converge to the latter. This process of computation and comparison until convergence makes this approach an iterative technique. Algorithm 1 shows the pseudo-code for this algorithm.

Algorithm 1: Pseudoinverse Method

PROCEDURE PseudoInverse($x_{des}, q_{seed}, stepSize$)
$x \leftarrow FPK(q_{seed})$
$q \leftarrow q_{seed}$
repeat
$\Delta x \leftarrow x_{des} - x$
$\dot{x} \leftarrow \frac{\Delta x}{||\Delta x||} \times stepSize$
$q \leftarrow q + J^{\dagger}(q)\dot{x}$
$x \leftarrow FPK(q)$
until $||x_{des} - x|| < \varepsilon$

2) Cyclic Coordinate Descent: Inverse kinematics for serial manipulators has always been a challenging task. It has been an area of research for long, especially for redundant robots with more than 6 degrees of freedom. The proposed task is more challenging when the number of variables (joint angles) is less than the number of equations derived from the input pose and orientation of the robot.

In the Cyclic Coordinate Descent method, changes are made in joint values, one joint per iteration until the end-effector pose and orientation does not converge to the input value. The variation made in joint angle values for joints farthest from the base does not reflect on the complete chain. Such joints are tried first while gradually moving towards joints proximal to the base. This make the process slow in order to make their effects reflect on the subsequent links. Since it does not involve the Jacobian matrix, this method is free from issues of matrix inversion and is also free from singularities. Figure 6 shows the flowchart for CCD.

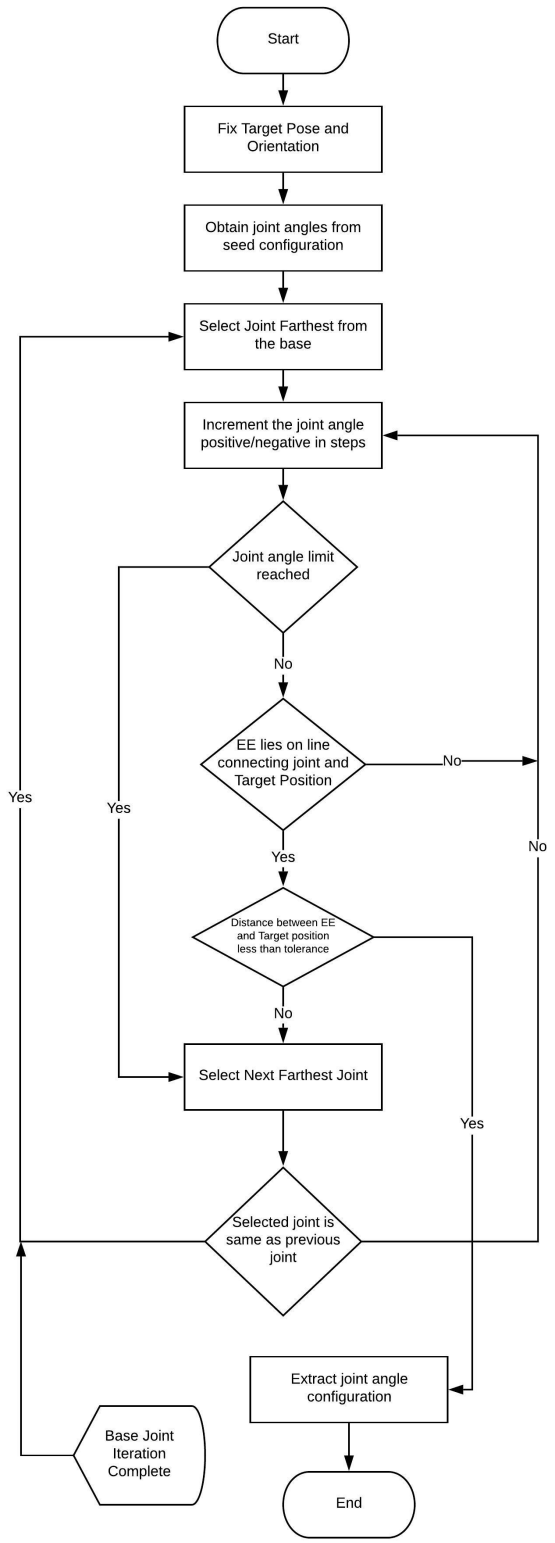

Fig. 6: CCD Algorithm Implementation

V. Future Work

A. Dynamics - MCG Matrices

The mass-inertia matrices, Coriolis effect coefficients and gravity coefficients are three major components needed to define the dynamical aspects a robot. The mass matrix for an individual link of the manipulator is defined as:

$$M = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

The Baxter robot URDF provided by Rethink Robotics already has these values available for each individual links. We intend to verify the same using the software *MeshLab*. To ensure accuracy of the values, we check if the already provided values are for the collision model of the robot (defines individual links as cylinders and spheres) or the visual models (defines the exact shape and curvature of the robot).

Coriolis effect is generally neglected for smaller systems to merely circumvent the intricacies involved with the solution. However, for a serial manipulator with several links like for Baxter, we can't neglect the Coriolis effect. As analyzed by other researchers, there are over half-a-million coefficients for Baxter. We propose to use the Euler-Lagrangian method to solve for these values and also to calculate the values in the gravity matrix.

B. Trajectory and Position Control

While trajectory control deals with real-time tracking of each instant of the system-state and ensuring that it matches the desired instantaneous state; position control is mainly concerned with the final condition of the system and tries to ensure that it matches the desired value. Trajectory control minimizes instantaneous errors while position control tends to modify the control signals only to minimize the steady-state error.

The trajectory control input has a feedback component related to the error in current and desired instantaneous state and a feedforward component related to the reference input. A general representation of the input is given as:

$$u = k_1 e + u^d(t)$$

C. ROS Support

We propose to code our library with an object-oriented approach with the ability to work either as a standalone system or as a plugin service for several platforms like MoveIt! and Klamp't or as a ROS package. We intend to make the library suitable for reuse and extend its applicability across various motion-planning and control platforms. The table below shows the remaining tasks and proposed timeline for the same.

VI. Future Timeline

Task No.	Task Name	Duration (weeks)
1	M C G Matrices	2
2	Reach Goals	
2.1	Trajectory and Position Control	1
2.2	ROS Support	1

We reiterate the subtasks we have completed till date in the following list.

1) Forward Pose Kinematics
2) Skeleton visualization and workspace analysis
3) Velocity Kinematics
4) 6-DOF Inverse Pose Kinematics
5) 7-DOF Inverse Pose Kinematics
    a) Jacobian Pseudoinverse
    b) Cyclic Coordinate Descent

References

[1] L. E Silva, T. M. Tennakoon, M. Marques, and A. M. Djuric, Baxter Kinematic Modeling, Validation and Reconfigurable Representation, 2016, vol. 2016April, no. April.

[2] A. Smith, C. Yang, C. Li, H. Ma, and L. Zhao, Development of a dynamics model for the Baxter robot, in 2016 IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2016, 2016, pp. 12441249.

[3] Z. Ju, C. Yang, and H. Ma, Kinematics modeling and experimental verification of baxter robot, in Proceedings of the 33rd Chinese Control Conference, 2014, pp. 85188523.

[4] R. S. Hartenberg J. Denavit "A kinematic notation for lower pair mechanisms based on matrices" J. Appl. Mech. vol. 77 no. 2 pp. 215-221 Jun. 1955.

[5] Kazerounian, K. (1987). On the numerical inverse kinematics of robotic manipulators. Journal of mechanisms, transmissions, and automation in design, 109(1), 8-13.

[6] Beeson, P., & Ames, B. (2015, November). TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on (pp. 928-935). IEEE.

[7] Aristidou, A., & Lasenby, J. (2011). FABRIK: a fast, iterative solver for the inverse kinematics problem. Graphical Models, 73(5), 243-260.

[8] R.L. Williams II, Baxter Humanoid Robot Kinematics, Internet Publication, https://www.ohio.edu/mechanical-faculty/williams/html/pdf/BaxterKinematics.pdf, April 2017.

[9] Yang, C., Ma, H., & Fu, M. (2016). Advanced technologies in modern robotic applications. Springer Singapore.

[10] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. Robot Dynamics and Control