Inverse Kinematics and Dynamics Control for the da Vinci Research Kit's Patient Side Manipulator

Emmanuel Jayaraju	ejayaraju@wpi.edu
Aditya Mehrotra	amehrotra@wpi.edu
Ritwik Pandey	rpandey@wpi.edu
Haoying Zhou	hzhou6@wpi.edu

An end-term report presented for the
RBE 501: Robot Dynamics course

Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
December 13, 2021

1 Abstract

This project is based on the da Vinci Reasearch Kit (dVRK) [1], an open-source surgical robotic platform (see Fig. 1) obtained from the first-generation of Intuitive Surgical System. Its controllers and software were developed at the Laboratory for Computational Sensing and Robotics (LCSR) at Johns Hopkins University and the Automation and Interventional Medicine (AIM) lab at Worcester Polytechnic Institute [5]. The aim of this project is to derive the complete dynamic model for the Patient Side Manipulator (PSM) of the dVRK and then calculate the inverse dynamics of the system to move it from a given start pose to a goal pose. Further, to verify the dynamic model and the calculated inverse dynamics, our model was used to simulate the PSM on the Asynchronous Multi-Body Framework (AMBF), developed by the AIM lab at WPI.

2 Introduction

Intuitive Surgical's da Vinci robot is a well-known system for real world surgery and is used by several hospitals for minimally invasive surgery [1]. As shown in Fig. 1, the da Vinci robot is generally comprised of four sub-systems, namely, a pair of Master Tool Manipulators (MTMs), three Patient Side Manipulators (PSMs), one slave Endoscope Camera Manipulator (ECM) and a Setup Joint (SUJ) Cart [3]. However, the focus of this project is on the PSMs only. Particularly in developing an accurate dynamics model that can be used to calculate the inverse dynamics required to follow a specific pre-planned trajectory.

This work details the progress in development of the kinematic and dynamic models. As a first step, the robot was modeled on MATLAB using the Robotics Toolbox [2] in multiple configurations. This model is based on the DH parameters provided in the dVRK user manual. This is helpful in quickly visualizing the robot and verifying the calculations with the existing functions from the toolbox. After developing the kinematic model of the manipulator, the Product of Exponentials method was used to solve for the forward kinematics for the robot. Furthermore, the inverse kinematics were solved using a combination of Newton-Raphson and Gradient method. The dynamics model was created by assigning link frames to each link, calculating the transforms required to move from one one link frame to another, and calculating the spatial inertia matrices for each link. The inverse dynamics of the system was calculated using the Recursive Newton-Euler algorithm. Finally the system was simulated both on MATLAB and the AMBF.

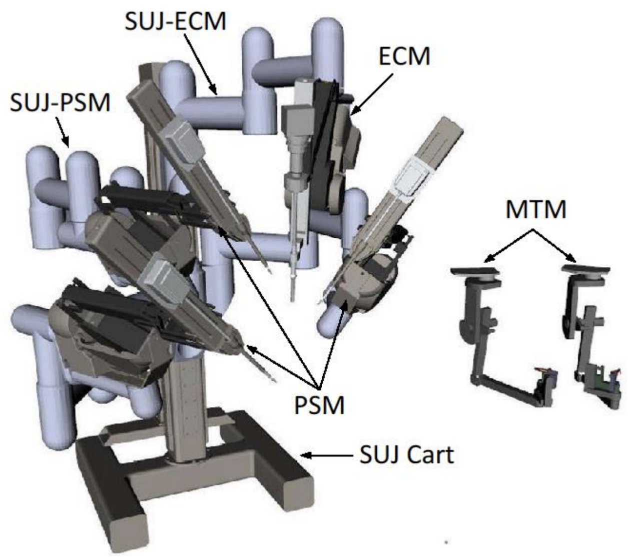


Figure 1: The da Vinci Research Kit

3 Models

3.1 Kinematic Modelling

3.1.1 Under-actuated Model

The dVRK is an under-actuated model with a closed-loop parallelogram mechanism. This model is in accordance with the work of Wang et al. [7]. Table 1 represents the modified DH parameters for the under-actuated model and Fig. 2 shows its visual representation. Here, $L_{2L3} = 0.04009$ m, $L_{2H1} = 0.14454$ m, $L_{2L2} = 0.516$ m, $L_3 = 0.04009$ m, $L_{tool} = 0.4162$ m, $L_{pitch2yaw} = 0.0091$ m. In this mechanism, the passive joints 2' 2", 2"" are dependent on position of joint 2.

Links	a	d	α	Offset
Link1	0	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
Link2	0	0	-$\frac{\pi}{2}$	-$\frac{\pi}{2}$
Link2'	L_{2L3}	0	0	$\frac{\pi}{2}$
Link2"	L_{2H1}	0	0	$\frac{\pi}{2}$
Link2""	L_{2L2}	0	0	0
Link3	L_3	0	-$\frac{\pi}{2}$	-$(L_{RCC} - L_{2H1})$
Link4	0	L_{tool}	0	0
Link5	0	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
Link6	$L_{pitch2yaw}$	0	-$\frac{\pi}{2}$	$\frac{\pi}{2}$
Link7	0	0	0	0

Table 1: Under-actuated Model Modified DH Parameters

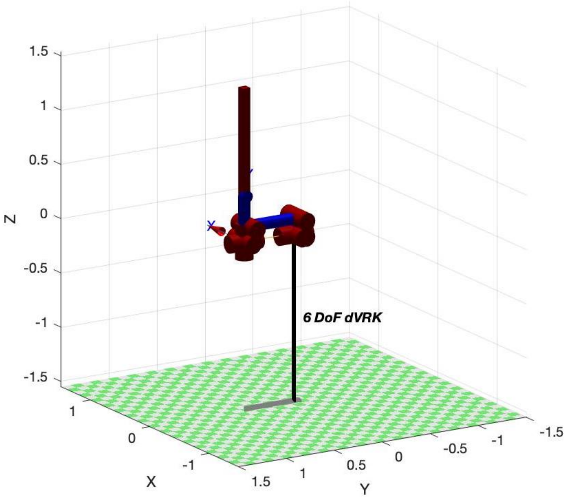


Figure 2: Under-actuated model representation

However, to keep the modelling simple while using the Robotics Toolbox on Matlab, a simplified model was proposed by ignoring the passive joints as described below.

3.1.2 Simplified Model

This is a customized model of the dVRK's PSM. The frame assignments and relevant dimensions are as shown in the Fig. 3. Table 2 represents the modified DH parameters for the model.

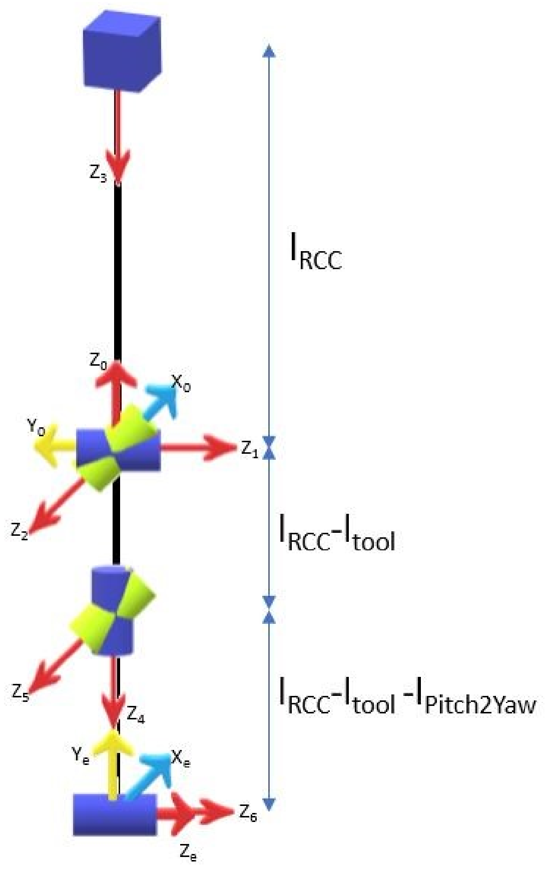


Figure 3: WPI Simplified Model

Links	a	d	α	Offset
Link1	0	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$
Link2	0	0	-$\frac{\pi}{2}$	-$\frac{\pi}{2}$
Link3	0	0	$\frac{\pi}{2}$	-L_{RCC}
Link4	0	L_{tool}	0	0
Link5	0	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
Link6	$L_{pitch2yaw}$	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
Link7	0	0	0	π

Table 2: Simplified Model Modified DH Parameters

Then, the rotation matrix (R_{ee}) and translation (p_{ee}) of the PSM end effector were calculated to be:

$$R_{ee} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, p_{ee} = \begin{bmatrix} 0 \\ 0 \\ L_{RCC} - L_{tool} - L_{pitch2yaw} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.0065 \end{bmatrix}$$

and thus, the transformation matrix from the origin of the global frame to the end effector frame can be given as:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.0065 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Screw-Axis representation of the manipulator was calculated as:

$$S = \begin{bmatrix} 0 & -1.0000 & 0 & 0 & -1.0000 & 0 & 0 \\ -1.0000 & 0 & 0 & 0 & 0 & -1.0000 & -1.0000 \\ 0 & 0 & 0 & -1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0065 & 0.0065 \\ 0 & 0 & 0 & 0 & -0.0156 & 0 & 0 \\ 0 & 0 & -1.0000 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.1.3 AMBF Simulation Model

This representation of the manipulator was customized further to solve for the dynamics using the AMBF simulator by moving the space frame to a new position as described in Fig. 4. This model is scaled and here, $L_{RCC} = 0.4318$ m, $L_{tool} = 0.4162$ m, $L_{pitch2yaw} = 0.0091$ m, $L_{yaw2CtrlPnt} = 0.0102$ m.

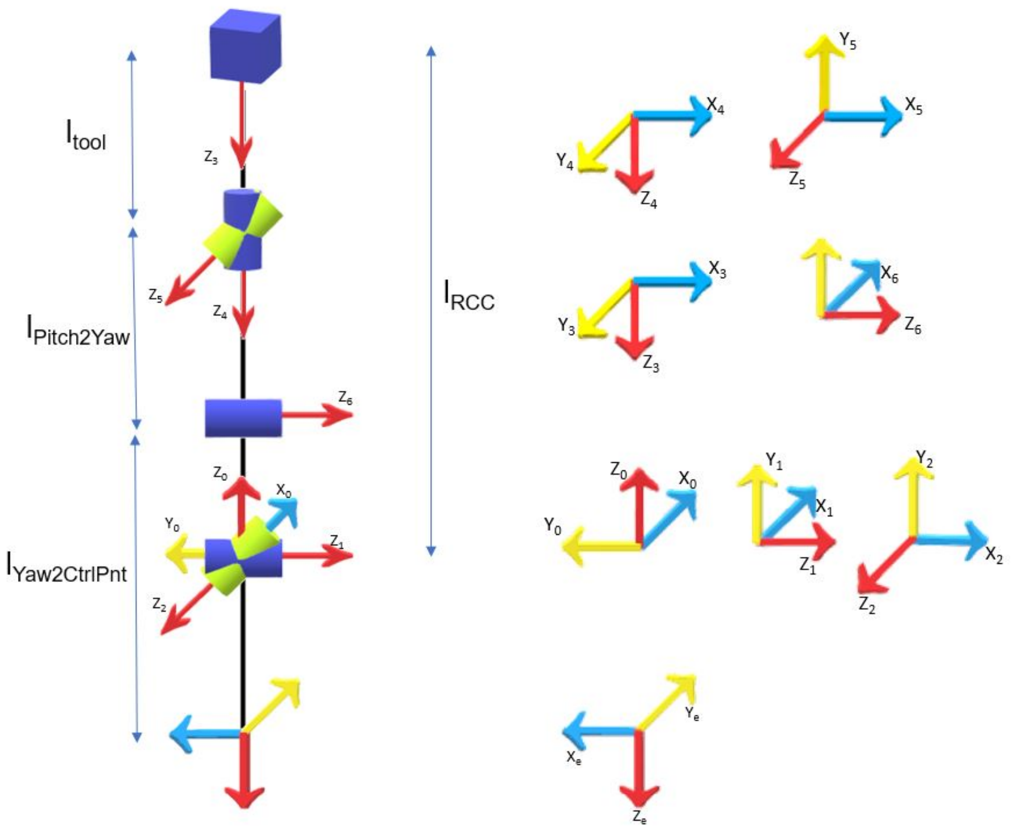


Figure 4: AMBF Simulation Model

The rotation matrix (R_{ee}) and translation (p_{ee}) of the PSM's end effector can be calculated:

$$R_{ee} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, p_{ee} = \begin{bmatrix} 0 \\ 0 \\ L_{RCC} - L_{tool} - L_{pitch2yaw} - L_{yaw2CtrlPnt} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0.0037 \end{bmatrix}$$

and thus, the transformation matrix from the origin of the global frame to the end effector frame can be written as:

$$M = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -0.0037 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Screw-Axis representation of the robot was calculated as:

$$S = \begin{bmatrix} 0 & -1.0000 & 0 & 0 & -1.0000 & 0 & 0 \\ -1.0000 & 0 & 0 & 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & -1.0000 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0.065 & 0 \\ 0 & 0 & 0 & 0 & -0.0156 & 0 & 0 \\ 0 & 0 & -1.0000 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.2 Dynamic Modelling

The following calculations are for the above model (Sec. 3.1.3) to perform Dynamics on Matlab.

Pose of link 1 in link frame$\{0\}$:

$$M_{01} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose of link 2 in link frame$\{1\}$:

$$M_{12} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose of link 3 in link frame$\{2\}$:

$$M_{23} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & (L_{RCC}/2) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose of link 4 in link frame$\{3\}$:

$$M_{34} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & (L_{tool} - L_{RCC})/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose of link 5 in link frame$\{4\}$:

$$M_{45} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose of link 6 in link frame$\{5\}$:

$$M_{56} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -(L_{tool} + L_{pitch2yaw})/2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pose of link 7 in link frame$\{6\}$:

$$M_{67} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1.000 & -(Lpitch2yaw/2) + L_{yaw2ctrlPnt} \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The above frame assignments for links can be validated to be correct by comparing the product of all the above Mxx with the homogeneous matrix 'M' derived earlier for the end effector. So, to compute the link frame for end effector with respect to the space frame:

$$M_7 = M_{01} \times M_{12} \times M_{23} \times M_{34} \times M_{45} \times M_{56} \times M_{67}$$

$$M_7 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -0.0037 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It can be observed that the result above is equal to the homogeneous transformation matrix (M) from the space frame to the end effector frame.

Next, the Rotational Inertia Matrix and Spatial Inertia Matrix for a link are defined as,

$$I_b = \begin{bmatrix} \sum m_i(y_i^2 + z_i^2) & -\sum m_i x_i y_i & -\sum m_i x_i z_i \\ \dots & \sum m_i(x_i^2 + z_i^2) & \dots \\ \dots & \dots & \sum m_i(x_i^2 + y_i^2) \end{bmatrix}$$

$$G_b = \begin{bmatrix} I_b & 0_{3\times3} \\ 0_{3\times3} & mI_{3\times3} \end{bmatrix}$$

The body inertia for each link was generated using the *Blender* software. Using these inertia values and the masses of each links, here's the spatial inertia matrices for all links.

m_1 = 5 kg; m_2 = 5 kg; m_3 = 2 kg; m_4 = 1 kg; m_5 = 1 kg; m_6 = 1 kg; m_7 = 2 kg

Spatial Inertia Matrix for link 1:

$$G1 = \begin{bmatrix} 0.2458 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2458 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0150 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_1 \end{bmatrix}$$

Spatial Inertia Matrix for link 2:

$$G2 = \begin{bmatrix} 13.7647 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.7060 & 0 & 0 & 0 & 0 \\ 0 & 0 & 13.870 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_2 \end{bmatrix}$$

Spatial Inertia Matrix for link 3:

$$G3 = \begin{bmatrix} 0.09508 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2877 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2418 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_3 \end{bmatrix}$$

Spatial Inertia Matrix for link 4:

$$G4 = \begin{bmatrix} 1.1920 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.1920 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0009 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_4 \end{bmatrix}$$

Spatial Inertia Matrix for link 5:

$$G5 = \begin{bmatrix} 0.0008 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0028 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0028 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_5 \end{bmatrix}$$

Spatial Inertia Matrix for link 6:

$$G6 = \begin{bmatrix} 0.0002 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0002 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0002 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_6 \end{bmatrix}$$

Spatial Inertia Matrix for link 7:

$$G7 = \begin{bmatrix} 0.0009 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0005 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0008 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_7 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_7 \end{bmatrix}$$

4 Methods

4.1 Kinematics

This section explains the mathematical and kinematic principle behind the corresponding functions.

For an angular velocity $\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$, the skew matrix $[\omega]$ is:

$$[\omega] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{1}$$

The rotation matrix can be calculated using the exponential coordinates and Rodrigues' formula:

$$R = e^{[\omega]\theta} = I + \sin\theta[\omega] + (1 - \cos\theta)[\omega]^2 \tag{2}$$

where I is the identity matrix and $[\omega]$ is calculated from Equation 1.

A screw axis (special case of twist) is represented in following format:

$$S = (\omega, v) \tag{3}$$

where ω is the angular velocity and v is the linear velocity. Here, $v = -[\omega]p + \dot{p}$, p is the translation vector indicating the position.

According to the derivation described in [6], for a revolute joint:

$$T = e^{[S]\theta} = \begin{bmatrix} e^{[\omega]\theta} & (I\theta + (1-\cos\theta)[\omega] + (\theta - \sin\theta)[\omega]^2)v \\ 0 & 1 \end{bmatrix} \tag{4}$$

where I is the identity matrix for $\mathbb{R}^{3\times3}$ and $[\omega]$ is calculated from Equation 1.

And for a prismatic joint:

$$T = e^{[S]\theta} = \begin{bmatrix} I & v\theta \\ 0 & 1 \end{bmatrix} \tag{5}$$

where I is the identity matrix for $\mathbb{R}^{3\times3}$.

The forward kinematics can be calculated using the screw axes:

$$T = (\prod_{i=1}^{N} e^{[S_i]\theta_i})M \tag{6}$$

where N is the number of joints, S_i is the screw axis of joint i, θ_i is the i_{th} joint variable, M is the home configuration of the end effector.

According to the derivation described in [6], the adjoint transformation can be calculated as follows:

$$\begin{aligned} T &= \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \\ Ad_T(v) &= \begin{bmatrix} R & [p]R \\ 0 & R \end{bmatrix} v \end{aligned} \tag{7}$$

where $[p]$ is calculated via Equation 1.

Subsequently, the adjoint transformation can be used to calculate the space Jacobian matrix as following:

The i_{th} column of the space Jacobian matrix is:

$$\begin{aligned} J_{S_1} &= S_1 \\ J_{S_i}(\theta) &= Ad_{e^{[S_1]\theta_1} \ldots e^{[S_{i-1}]\theta_{i-1}}}(S_i) \qquad for\ i \neq 1 \end{aligned} \tag{8}$$

The analytical jacobian was used to converge to the generated trajectory.

$$\begin{aligned} \dot{p} &= J_a(\theta)\dot{\theta} \\ J_S &= \begin{bmatrix} J_{Sw} \\ J_{Sv} \end{bmatrix}^T \\ \Rightarrow J_a &= J_{Sv} - [p]J_{Sw} \end{aligned} \tag{9}$$

The following equation describe the Levenberg-Marquadt algorithm used for achieving optimal results in Inverse Kinematic.

$$\begin{aligned} \Delta Q &= J_a^{+}(S_{current} - S_{target}) \\ J_a^{+} &= J_a^T(JJ^T + \lambda^2 I)^{-1} \end{aligned} \tag{10}$$

where λ is the damping factor and I is the identity matrix for $\mathbb{R}^{n\times n}$

To get the optimal results, these methods were used in a combination where if the error in the pose is greater than 0.1, the Gradient method is used, whereas on lower values, the Newton-Raphson method takes over.

4.2 Dynamics

Further ahead, the Recursive Newton Euler (RNE) method was used for solving the inverse dynamics of the manipulator.

The adjoint matrix is used to transform the twist from one the space frame to the body frame.

$$\begin{aligned} \nu_s &= Ad_T \nu_b \\ \nu_b &= Ad_T^{-1} \nu_s \end{aligned} \tag{11}$$

The relationship between effort and the force exerted by the end effector on the environment can be give as,

$$\tau = J^T(\theta) F_{tip} \tag{12}$$

The Lie Bracket of twist is defined as,

$$[ad_\nu] = \begin{bmatrix} [\omega] & 0_{3\times3} \\ [\nu] & [\omega] \end{bmatrix} \tag{13}$$

So, the equation of motion with the terms defined above is

$$F_b = G_b \dot{\nu_b} - [ad_\nu]^T G_B \nu_b \tag{14}$$

For transforming the pose of the links from one frame to another, the following relationship is used

$$\begin{aligned} M_{i-1,i} &= M_{i-1}^{-1} M_i \\ M_{i,i-1} &= M_i^{-1} M_{i-1} \end{aligned} \tag{15}$$

To identify all screw axes in the local body frame,

$$\begin{aligned} A_i &= Ad_{M_i^{-1}}(S_i) \\ T_{i-1,i}(\theta) &= M_{i-1,i} e^{[A_i]\theta_i} \end{aligned} \tag{16}$$

With the above relations defined, the forward iteration of the RNE can be given as

$$\begin{aligned} \text{Velocity } \nu_i &= A_i \dot{(\theta_i)} + [Ad_{T_{i,j-1}}]\nu_{i-1} \\ \text{Acceleration } \dot{\nu_i} &= A_i \ddot{\theta_i} + [ad_{\nu_i}] A_i \dot{\theta_i} + [Ad_{T_{i,j-1}}]\dot{\nu_{i-1}} \end{aligned} \tag{17}$$

Whereas, the Backward Iteration of RNE is:

$$\begin{aligned} \text{Wrench } F_i &= G_i \dot{\nu_i} - [ad_{\nu_i}]^T G_i \nu_i + [Ad_{T_{i+1,i}}]^T F_{i+1} \\ \text{Torque } \tau_i &= F_i^T A_i \end{aligned} \tag{18}$$

Furthermore, to generate the trajectory that the end effector follows, the quintic polynomial funciton shown below was used.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_0 & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix} \tag{19}$$

5 Experimental Results

5.1 Kinematics

The computation for the forward kinematics of the robot is straight-forward. The output of the robot pose in the home configuration is shown in Fig. 5.

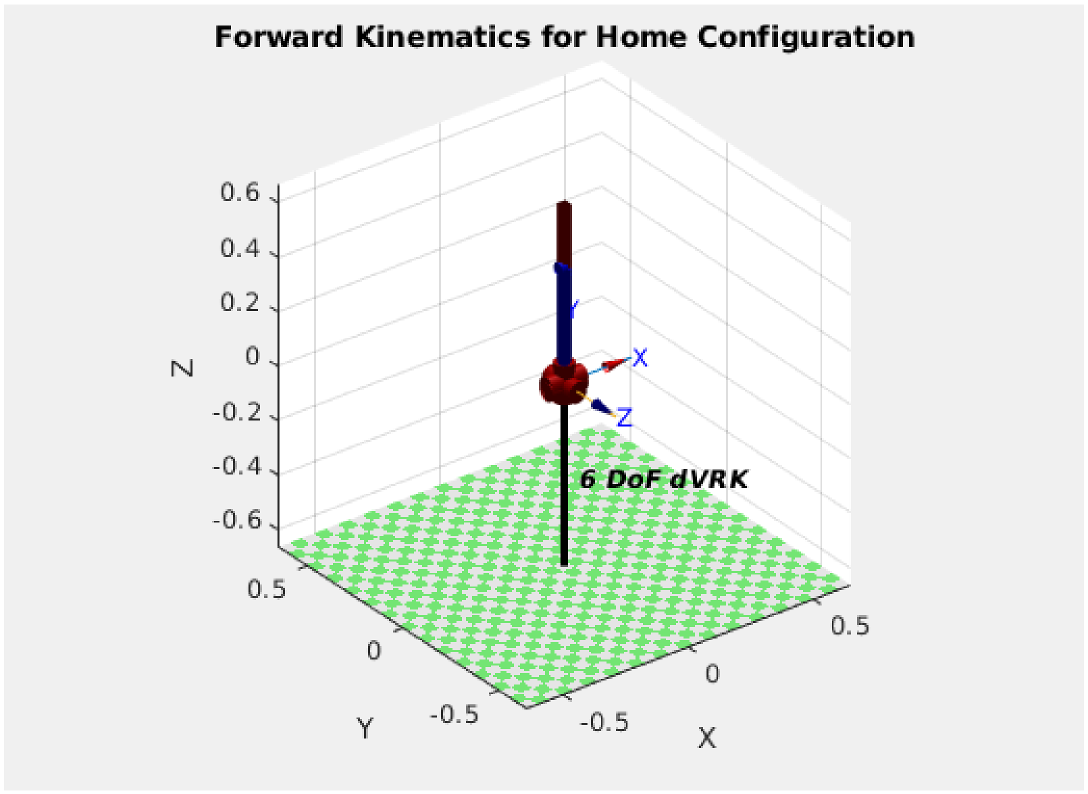


Figure 5: Simplified model in the home configuration

To demonstrate the output of the Inverse Kinematics for the robot, a 3D crown-shaped trajectory (see Fig. 6) was generated using the following set of equations:

$$a = 0.1; \qquad b = 0.05; \qquad c = 0.1; \qquad \theta = [-\pi, \pi]$$

$$x = a \times sin(\theta)$$

$$y = a \times sin(\theta)$$

$$z = b \times sin(3\theta) - c$$

This trajectory was then mapped with the end effector for both the models of the robot. Since the passive joints in the under-actuated model mimic the active joint 2, the joint values obtained from the simplified model is used to simulate the under-actuated model as well. Fig. 7a and Fig. 7b present the Simplified and under-actuated models tracing the path respectively.

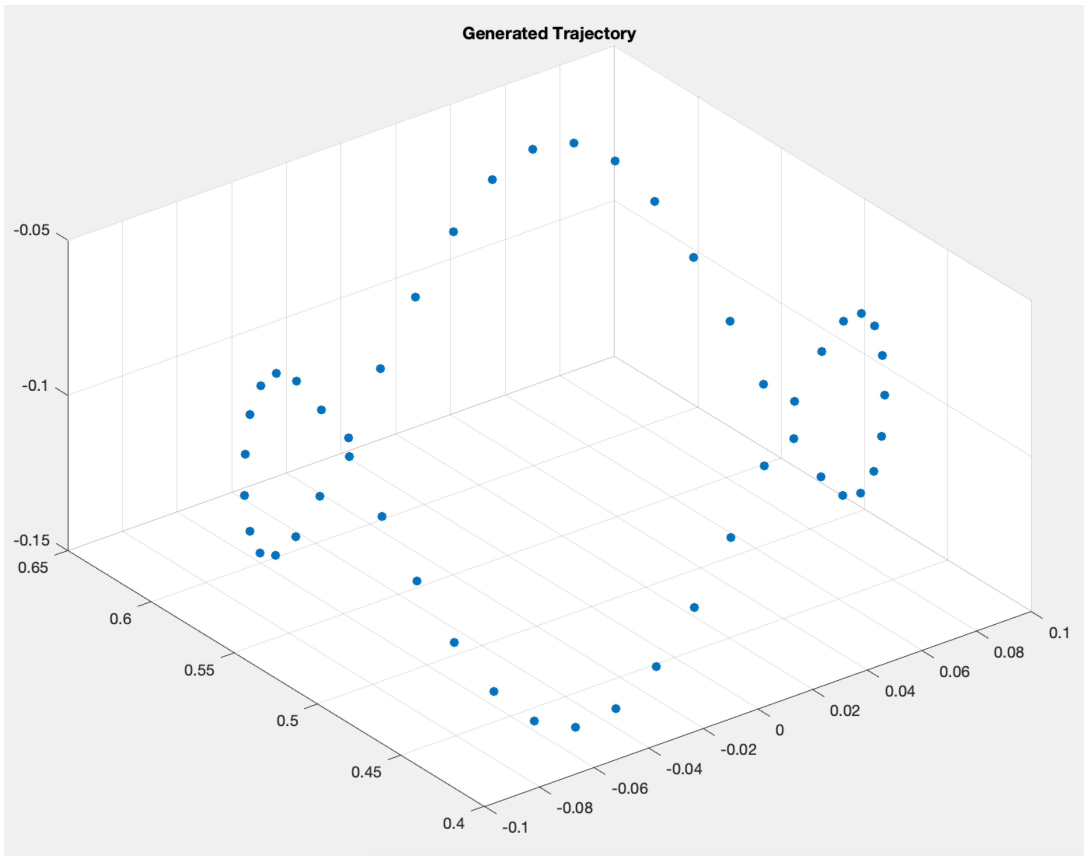
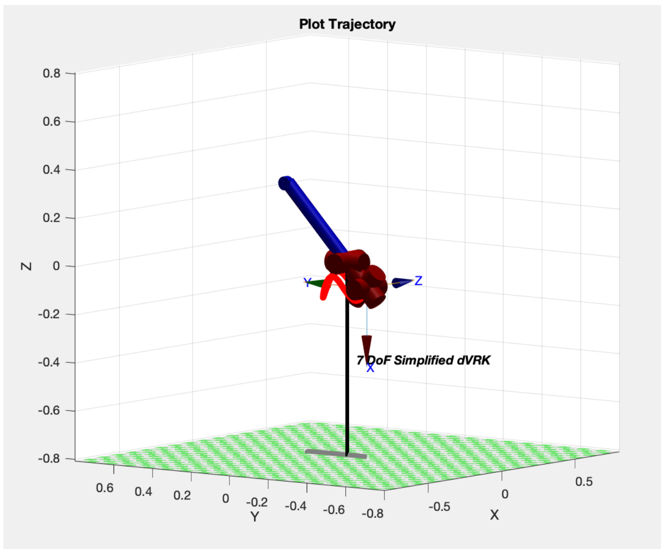
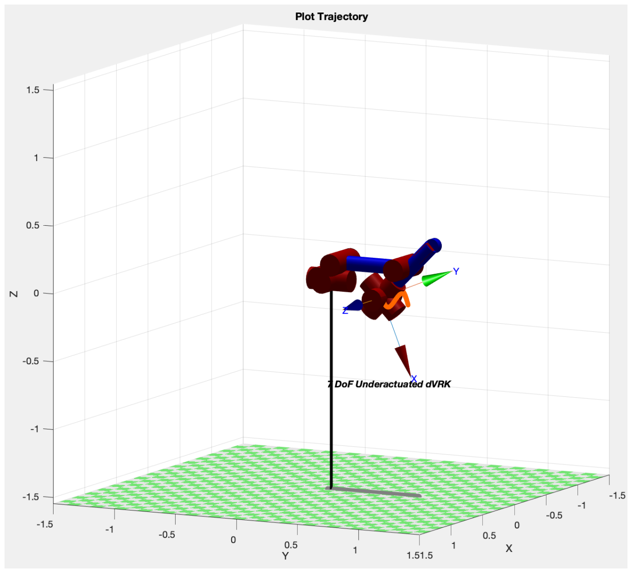


Figure 6: Generated trajectory to be followed



(a) Simplified Model mapping the trajectory



(b) Under-actuated Model mapping the trajectory

Figure 7: Experimental Results of Kinematic Modeling of the dVRK

Further, an error in pose against time (in secs) plot was recorded (see Fig. 8) for the Inverse Kinematics calculated for the Simplified Model.

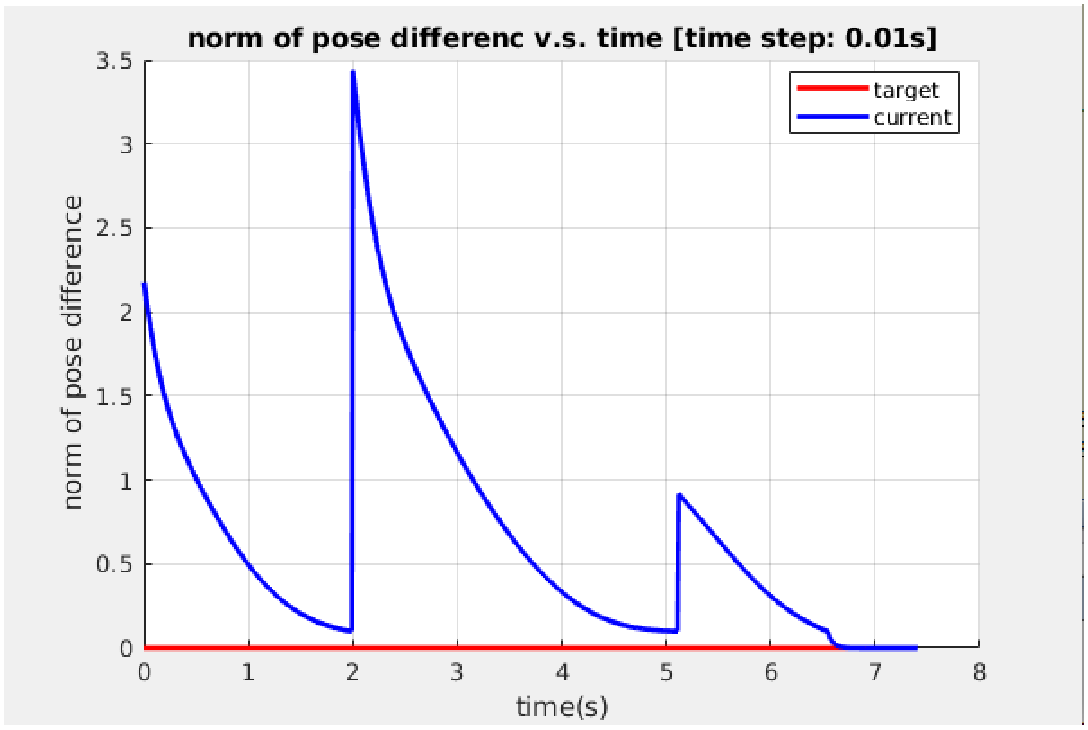
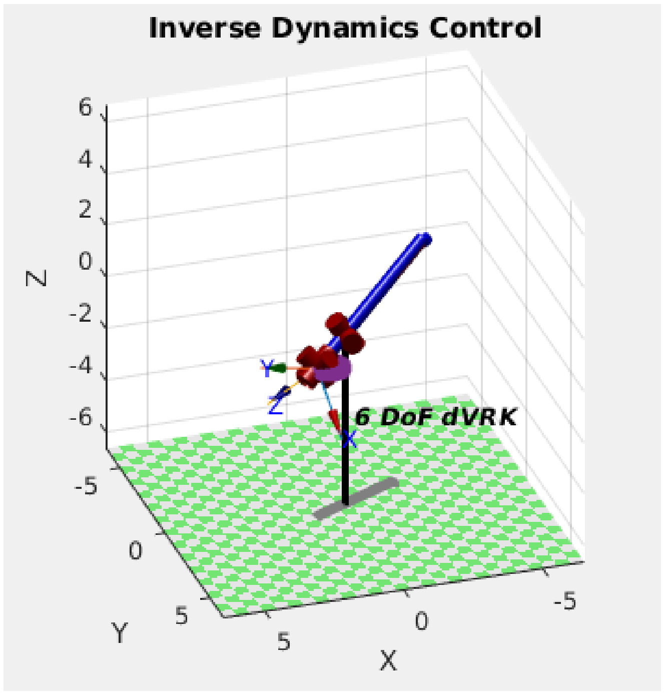


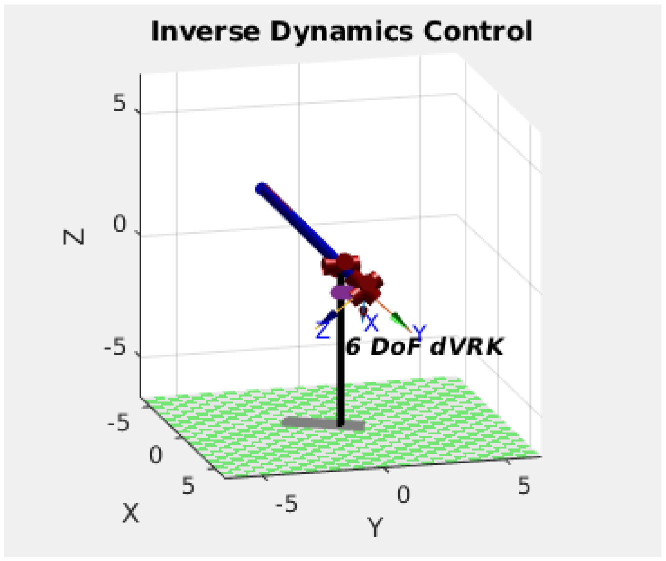
Figure 8: Convergence of error in pose

5.2 Dynamics

The dynamics of the model was worked out on the AMBF simulation model. To demonstrate the output of the Inverse Dynamics for the robot, two trajectories namely, a circle and a line was generated. Fig. 9a and Fig. 9b presents the AMBF simulation model tracing the circle and the line path respectively.



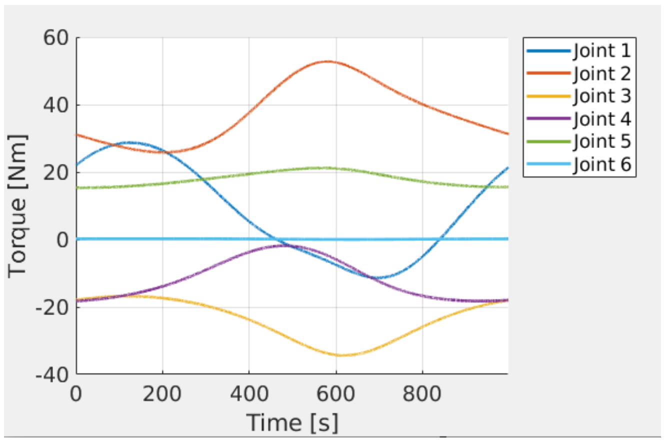
(a) AMBF Simulation Model mapping the circle trajectory



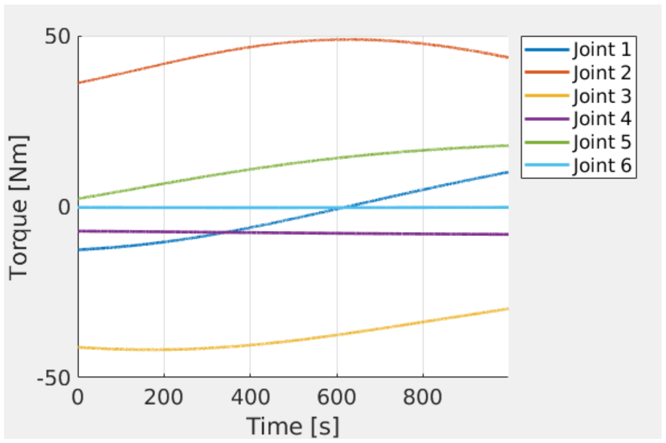
(b) AMBF Simulation Model mapping the line trajectory

Figure 9: Trajectory Tracking using Inverse Dynamics Control for the dVRK's PSM

Fig. 10a and Fig. 10b presents the joint torques profile obtained to track the said trajectories.



(a) Joints Torque Profile for the circle trajectory



(b) Joints Torque Profile for the line trajectory

Figure 10: Joint Torque Profiles for Trajectory Tracking

Furthermore, the results of Inverse Dynamics obtained from the MATLAB script were applied to the PSM on AMBF simulator. For this work, a Python based client node that interfaces with the PSM object on the AMBF simulator was used. This communication is shown at a higher level in Fig. 11. The PSM at their home configuration in AMBF simulator looks as shown in Fig. 12.

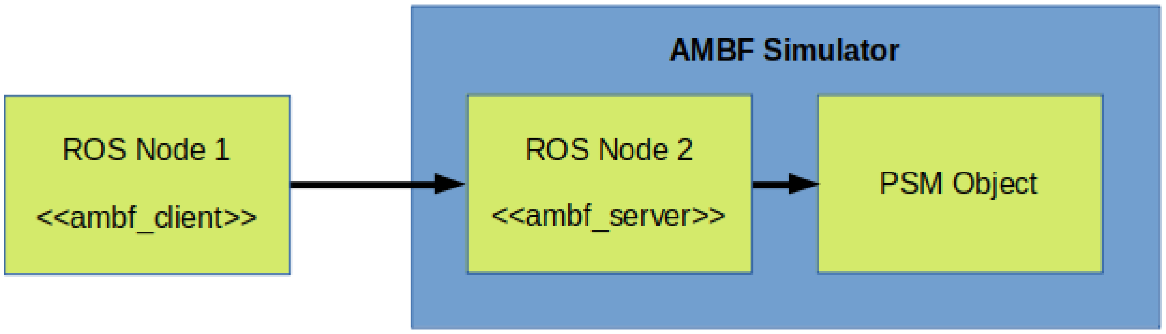


Figure 11: High-level view of interaction with AMBF simulator

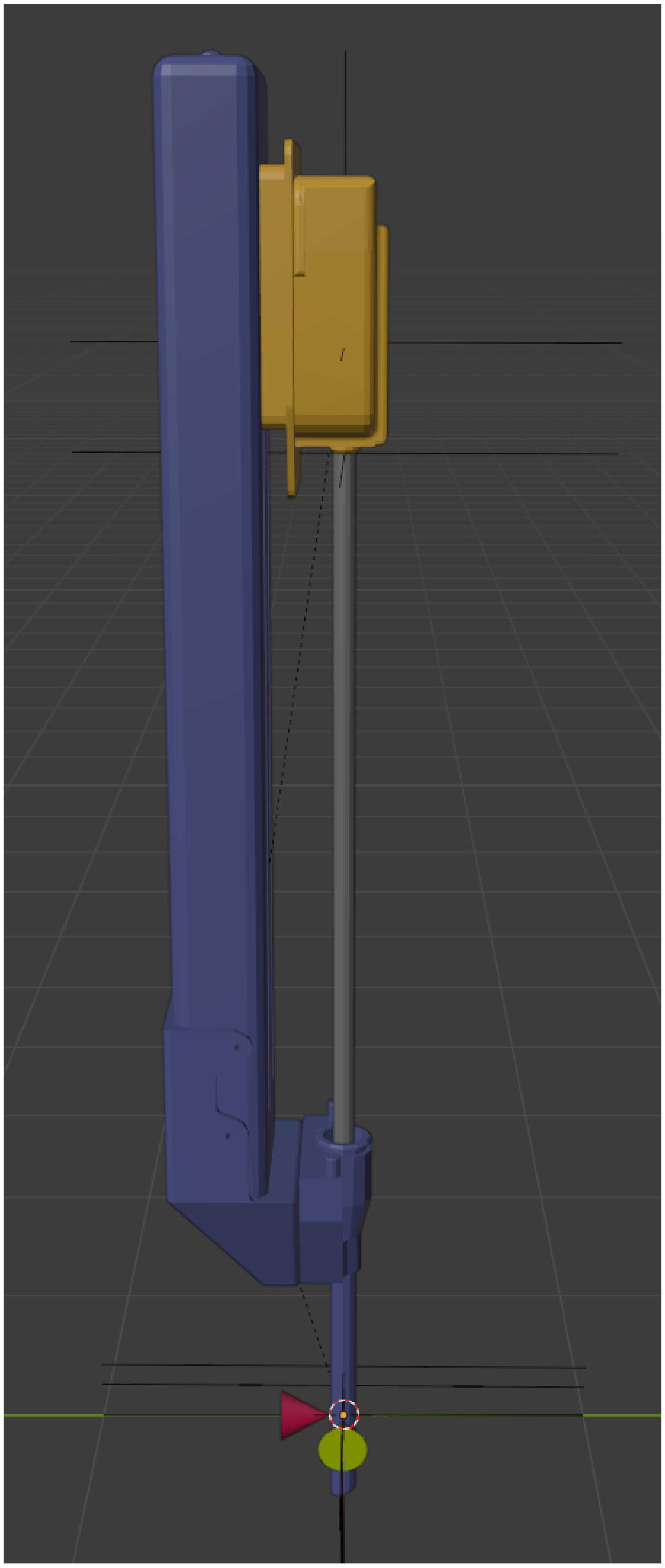


Figure 12: View of PSM in home configuration on the AMBF simulator

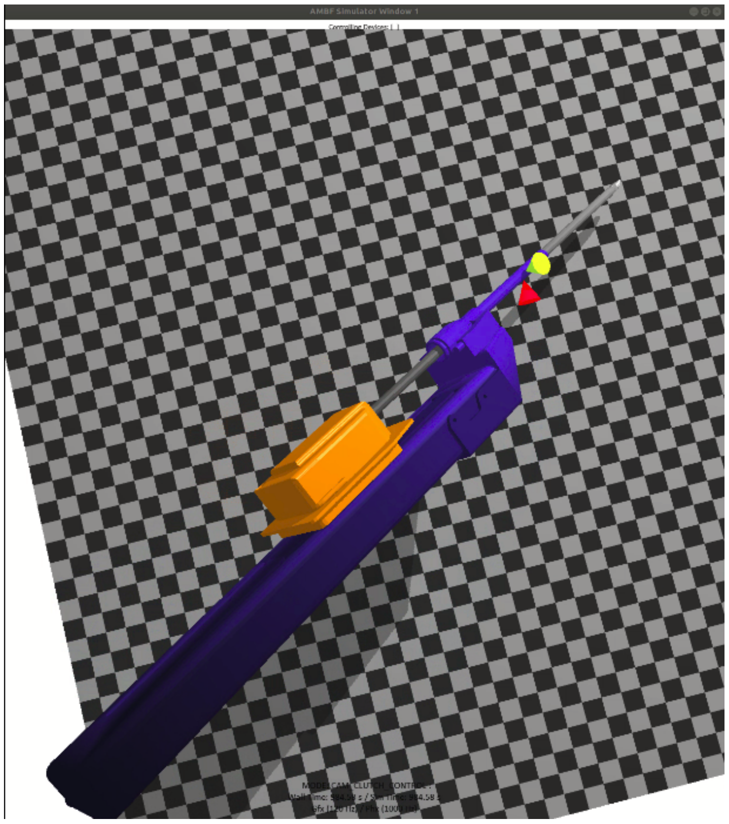
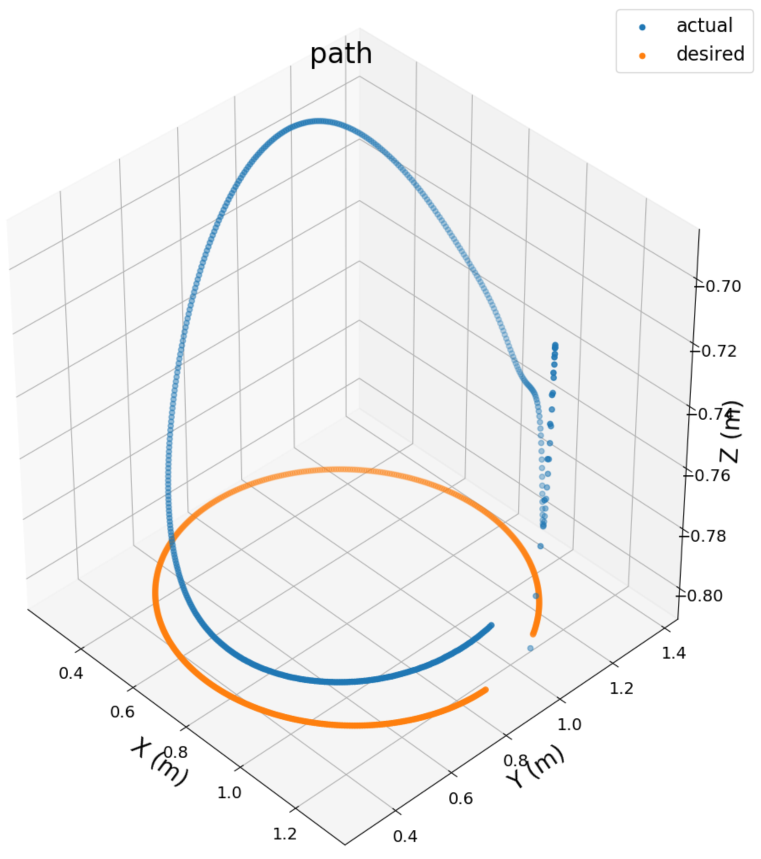


Figure 13: PSM simulation on AMBF simulator in action

Fig. 14 presents the quantitative data of trajectory tracking achieved on the AMBF simulator for IK.



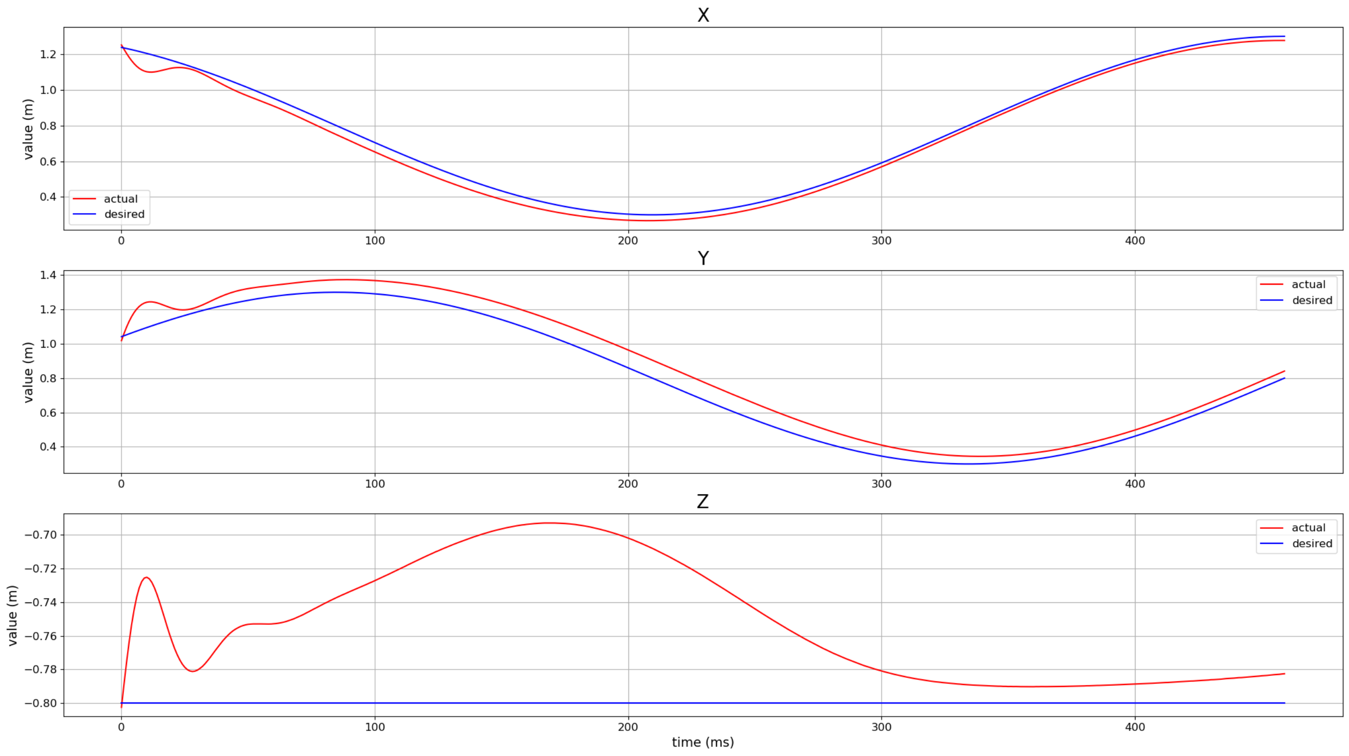


Figure 14: Trajectory Tracking using Inverse Kinematics Control for the dVRK's PSM

6 Discussion & Conclusions

In the first half of this project, forward and inverse kinematics were calculated for the PSM. The next phase of the project involved development of the dynamic model and calculation of the inverse dynamics of the manipulator. The recursive Newton-Euler approach was used to compute the inverse dynamics for a pre-defined path. The poses calculated using inverse kinematics in Matlab were exported to a .csv file. The output was communicated with the AMBF simulator through the ROS communication framework since AMBF allows interfacing with the PSM with a ROS based client wrapper around it. The results achieved for the dynamics control on the PSM wasn't satisfactory. However, that is one of the focus for our future work. Overall, while the goals that were set at the beginning of this study were successfully achieved, possibilities for improvement remain.

7 Future Work

An adaptive active inference controller (AIC) can be designed for the PSM. Most control systems require accurately modelled parameters and dynamics. They also require significant tuning every time the parameters, payloads, or the environment change. The AIC utilises the Free Energy Framework [4] proposed by Karl Friston. This allows the system to work around inaccuracies in the models and adapt to rapid changes in the environment. An added advantage is that very minimal tuning is required to transfer this controller from a simulated environment to a physical real-world environment. The performance of this controller can be compared to that of a simple PID controller.

In order to remove the dependency on MATLAB scripts for real-time simulation, the work done on MATLAB would ported to C++ with help of the Rigid Body Dynamics Library. The controller described above could be integrated to this code-base. As of now, only path planning has been implemented on the AMBF simulator, trajectory planning is in scope of the future work.

8 Authors' Contributions

- **Emmanuel Jayaraju**: methodology, formal analysis, writing – original draft preparation;
- **Aditya Mehrotra**: methodology, formal analysis, writing – original draft preparation;
- **Ritwik Pandey**: methodology, formal analysis, writing – original draft preparation;
- **Haoying Zhou**: conceptualization, methodology, formal analysis, writing – original draft preparation.

References

[1] da vinci research kit wiki community. https://research.intusurg.com/index.php/Main_Page. (Accessed on 09/23/2021).

[2] Robotics toolbox - peter corke. https://petercorke.com/toolboxes/robotics-toolbox/. (Accessed on 10/29/2021).

[3] J Bodner, H Wykypiel, G Wetscher, and T Schmid. First experiences with the da vinci™ operating robot in thoracic surgery. *European Journal of Cardio-thoracic surgery*, 25(5):844–851, 2004.

[4] Karl Friston. The free-energy principle: A rough guide to the brain? *Trends in Cognitive Sciences*, 13(7): 293–301, 2009. doi: 10.1016/j.tics.2009.04.005.

[5] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. An open-source research kit for the da vinci® surgical system. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 6434–6439. IEEE, 2014.

[6] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[7] Yan Wang, Radian Gondokaryono, Adnan Munawar, and Gregory Scott Fischer. A convex optimization-based dynamic model identification package for the da vinci research kit. *IEEE Robotics and Automation Letters*, 4 (4):3657–3664, 2019.

Appendix A Source code

All source code is in the following GitHub repository:
https://github.com/JackHaoyingZhou/RBE501_final_project
If you come across any issues in accessing to the repository, please contact Haoying Zhou via email.