MATLAB - Based Robot Analysis, Simulation and Visualization of da-Vinci Surgical Robot

Aashirwad Patel
dept.of Robotics
Worcester Polytechnic Institute
acpatel@wpi.edu

Ajith Kumar Ganesan Govindasamy
dept.of Robotics
Worcester Polytechnic Institute
aganesangovindas@wpi.edu

Chinmay Burgul
dept.of Robotics
Worcester Polytechnic Institute
cmburgul@wpi.edu

Joshua Amrith Raj Caleb Chanthi Raj
dept.of Robotics
Worcester Polytechnic Institute
jcalebchanthiraj@wpi.edu

***Abstract*—The paper aims to present a detailed insight into the kinematic and dynamic analysis of one of the Patient Side Manipulators (PSM) of a renowned robotic surgical system, the Da-Vinci Surgical Robot. The model construes a double parallelogram 5-link remote centre of motion mechanism. The kinematic and dynamic model is developed using conventional Danavit-Hartenburg (DH) notation, and Euler-Lagrange every based method.**

***Index Terms*—Minimally Invasive Surgery (MIS) Robot, da-Vinci Surgical Robot, Forward Kinematics, Inverse Kinematics, Work-space estimation, Dynamics and Singularity.**

I. Introduction

The da-Vinci Surgical System, a type of Minimally Invasive Surgical (MIS) robot, was formally created by an American company Intuitive Surgical after the approval from Food and Drug Administration (FDA) in 2000. It is designed to aid surgeon who operates from a console to perform complex surgeries using minimally invasive approach. The console provides a comfortable platform to the surgeon that helps prevent fatigue over long periods of surgical procedures.

A typical da-Vinci Surgical System consists of a master console with two 7-d.o.f. master tool manipulator (MTM) arms with camera viewport, a patient cart set up joint with two to three 6-d.o.f. patient side manipulators (PSM) and one 5-d.o.f. endoscope camera manipulator (ECM), and a high-performance vision stereoscopic camera endoscope.

The first joint, the yaw, is not motor actuated and is usually set up manually by the surgeon before surgery and it stays stationary throughout. The second and third joints contributes to pitch and roll to the Cartesian movement. The third joint is a double four-bar linkage mechanism consisting of 5 links that forms a pitch intersecting the yaw axis of the first joint. This intersection constructs a virtual remote centre of motion (RCM), whose generation is one of the requisite terms for a MIS robot. The fourth joint is prismatic joint that assists in tool insertion. Hence, these joints completely form the entire Cartesian coordinate system. The wrist, comprising of the remaining three joints increases the dexterity of the manipulator. It consists of pitch, roll, yaw and opening and closing of the scissor-like gripper.

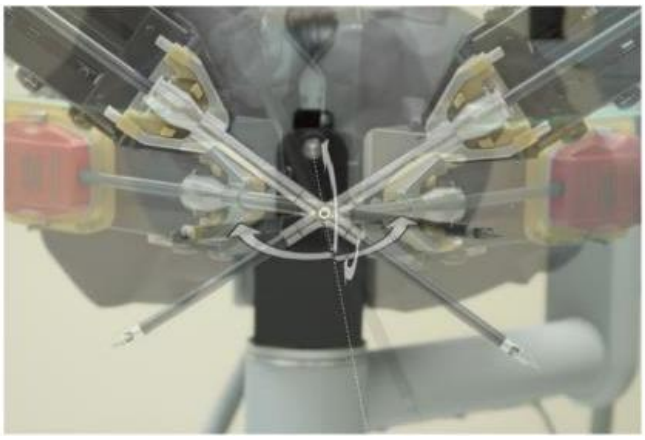


Fig. 1. Remote center of motion mechanism of the Patient side Manipulator source : dVRK Manual

II. Body

In these projects we analyzed the kinematics and dynamics of the DaVinci Manipulator. As mentioned in the introduction the DaVinci is not a pure serial manipulator it consists a double four bar linkage at the third joint, which makes the conventional approach of analyzing the kinematics and dynamics irrelevant. But assuming third joint as a simple rotary joint will solve the complexity. So, in this project we have assumed the third joint as a simple rotary joint and used the conventional approach of DH- Parameter to find the forward kinematics of the manipulator.

The uniqueness of these manipulator is the third frame is having an angular offset with the second frame. We had to introduce a variable () to parametrize the offset in DH table. After finding the forward kinematics from the DH table we visualized the work volume of the Manipulator by moving first three joints with a defined limit. The fourth Joint in the

Frame	Parent Frame	a	α	d	θ
1	0	0	$-\pi/2$	0.1524	$\pi/2$
2	1	0	$-\pi/2$	0.0296	$-\pi/2 + q_1$
3	2	0.150	0	0	$-\beta + q_2$
4	3	0.516	0	0	$\beta + \pi/2 - q_2$
5	4	0.2881	0	0	$-\beta + q_2$
6	5	-0.0430	$\pi/2$	0	$-\pi/2$
7	6	0	0	q_3	$\pi/2$
8	7	0	$\pi/2$	0	$\pi/2 + q_4$
9	8	0.0091	$-\pi/2$	0	$\pi/2 + q_5$
10	9	0.0102	0	0	q_6
11	10	0	$-\pi/2$	0	$-\pi/2$
C1*	2	0	0	0	q_2
C2-int*	2	0	$-\pi/2$	0	$-\pi/2 - \beta + q_2$
C2*	C2-int	0	0	δq_3	0

Fig. 2. Denavit-Hartenberg Parameters: Conventional

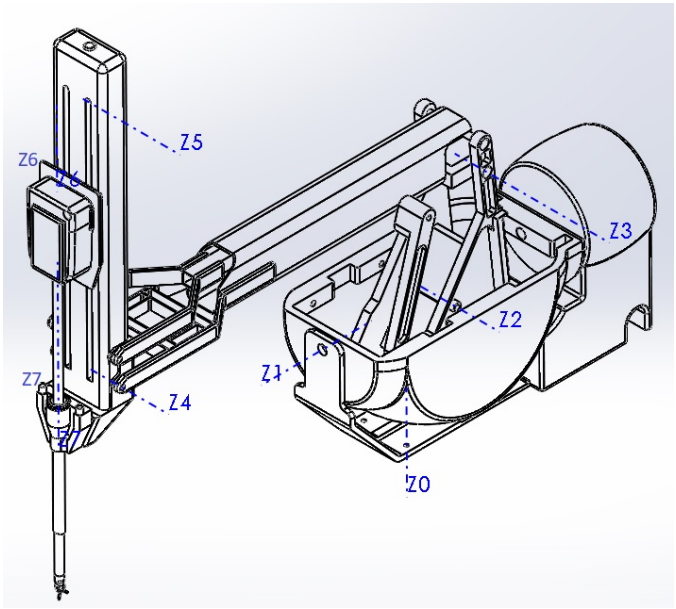


Fig. 3. PSM Manipulator

manipulator is only used during the surgical procedures for processes like injecting probes, suturing procedures and has very high scaled down ratio to operate with MTM. And the fifth, sixth and seventh joint is a spherical end-effector. So, we visualized the work volume with only first three joints.

Finding the Inverse Kinematics of a 6 DOF serial manipulator is a bit technically challenging task. Finding the Inverse kinematics of a hybrid of serial and parallel manipulator like DaVinci is a real challenging task. And the traditional approach to find IK, the geometric approach will fail here. We have not found any related works for IK solution of DaVinci because it is a designed to be a teleoperated robot and not designed to work autonomously.

A. Inverse Kinematics using Error Minimization:

Position: The four-bar linkage present in the DaVinci does not make it a pure serial manipulator. This makes solving the inverse kinematics geometrically very difficult. We therefore

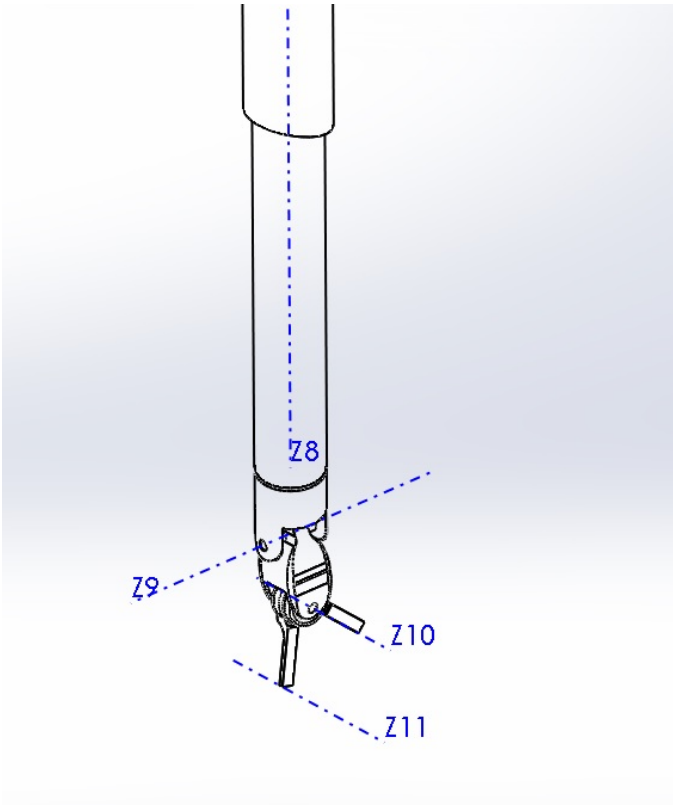


Fig. 4. Sperical Joint at End Effector

decided to go with another less known method for solving the Inverse Kinematics. This method minimizes the error which is defined as the difference between desired position of the end effector (x_d) and the position calculated with the joint variables using the forward kinematics (x). The initial values of the joint variables are selected arbitrarily.

$$e = x_d - x \tag{1}$$

To minimize the error this method makes use of a mathematical technique known as gradient descent optimization. This states that a function f(x) decreases fastest in the opposite direction of its gradient. Therefore, to reach the minimum value of f(x) we update x such that

$$X_{n+1} = X_n - \gamma_n(X_n), n > 0 \tag{2}$$

In any manipulator the joint variables and position is related by:

$$J(q) - \Delta q = \Delta x \tag{3}$$

$$\Delta q = J(q)^{-1}\Delta x \tag{4}$$

Hence, we must minimize the function:

$$||J(q)\Delta q - \Delta x||^2 \tag{5}$$

Simplifying this we get our step length to be $\Delta q = 2J(q)^T\Delta x$. By updating q sequentially, it converges to the inverse kinematics solution.

Orientation: To solve for the orientation, we use decoupling. The DaVinci PSM arm has 11 joints with 6 and 3 active and passive joints respectively. The revolute joints present at joints 8, 9 and 10 can be approximated as a single spherical joint present at the end of link 7. By solving the position inverse kinematics, we get the values of $q_1,\ q_2\ and\ q_3$. To solve for $q_4, q_5\ and\ q_6$, we use the equation: $R_{10}^{8} = (R_8^0)^T R$. Where R is the rotation matrix of the end effector.

Jacobian: The DaVinci manipulator has 6 active joints of which five are revolute and one is prismatic. The Jacobian is a 6x6 matrix and consists of two parts $J_v\ and\ J_w$ for the translational component and rotational component respectively. The elements of J_v is given by $\frac{\partial \vec{x}}{\partial q_i}$ where $\vec{x}$ is the position of the end effector. The elements in J_w is given by $\rho \vec{z_{i1}}$ Where ρ_i is 1 if the joint i is revolute and 0 of it is prismatic. Our final Jacobian Matrix looks something like this

$$\begin{vmatrix} \frac{\partial \vec{x}}{\partial q_1} & \frac{\partial \vec{x}}{\partial q_2} & \frac{\partial \vec{x}}{\partial q_3} & \frac{\partial \vec{x}}{\partial q_4} & \frac{\partial \vec{x}}{\partial q_5} & \frac{\partial \vec{x}}{\partial q_6} \\ \vec{z_0} & \vec{z_1} & \vec{0} & \vec{z_4} & \vec{z_5} & \vec{z_6} \end{vmatrix}$$

The manipulator encounters singularity when the value of the determinant Jacobian matrix is zero. Because of the complex equations involved we were not able to find the set of joint variables for which the robot encounters singularity.

Some unconventional approaches like finding IK using relative vector, or using Jacobian matrix exists. But we have not focused on the Inverse Kinematics part till now in this project. Using the forward kinematics we have derived a Jacobian matrix which will help in relating the joint velocities, forces, torques with the end-effector velocities, forces and torques. This Jacobian matrix will also help in finding the Singularities of the manipulator.

Finding the Dynamics using conventional methods of Euler Lagrange was also not possible without the assumption of third joint as a simple rotary joint. We have considered the masses as the point masses located at the end of the links and derived the dynamic equations using Euler-Lagrange method. And, we have derived the mathematical equation for velocity and acceleration for all the joints. The dynamic equations include the force and torque equations of each joints. We have used these equations to plot the change in forces and torques with the change in joint actuation. We have also used the urdf files created from AIM lab to do the trajectory/motion planning of the on Moveit API from ROS framework.

B. Trajectory Planning:

Trajectory planning involves moving from point A to point B while avoiding collisions over time. We form a polynomial as a function of time such that the constraints are satisfied. There are two cases of trajectory planning point to point trajectory and via points trajectory. In point to point trajectory planning we consider only the initial and final positions while planning the trajectory. In via points trajectory, we consider a set of points which must lie on the trajectory along which the robot should move.

Spline Function:

The joint variables are polynomial functions in time. The trajectory of the end effector can be approximated by using a piecewise polynomial called splines. The degree of the polynomial depends on the complexity of the task. For example, in the case of via points trajectory planning, as the number of points through which the robot must pass increases, the degree of the polynomial increases. An example for finding the trajectory between two points is shown below.

Quintic Spline: q(t) = $\mathrm{a}_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4^t 4 + a_5 t^5$

$$q(t0) = q0\ \&\ q(tf) = qf \tag{6}$$

$$\dot{q}(t0) = v0\ \&\ \dot{q}(tf) = vf \tag{7}$$

$$\ddot{q}(t0) = a0\ \&\ \ddot{q}(tf) = af \tag{8}$$

Using these conditions, we can find the optimum trajectory. To find the trajectory to be followed by the robot we first get the coordinates of the initial and final points. We then solve for the joint variables at these points using inverse kinematics. Depending on the number of boundary conditions we adopt a suitable polynomial. Suppose, we have n boundary conditions we consider a n-1 degree polynomial. Applying the boundary constraints, we get a set of equations in terms of the unknown coefficients of the polynomial. We solve these equations and compute the unknown coefficients.

Trajectory Planning for a da-Vinci robotic System: For doing point to point trajectory planning for a da-Vinci robot we consider the two points as $x_i = [0.6, 0.4, 10]$ and $x_f = [1.6, 0.4, 0]$. We also assume zero velocities at the initial and final positions. Using inverse kinematics, we find the values of joint variables at initial and final positions. We are only concerned about the position of the end effector therefore do not find the values of theta4, theta5 and theta6. Since we have 4 boundary constraints we solve for the coefficients of a 3 degree polynomial in time for theta1, theta2 and d3.

The graphs for joint variables, joint velocities and joint accelerations are shown below.

Simulating the DaVinci Robot: We have done a simulation of the DaVinci robot using MATLAB. We consider the motion of the end effector to be in a straight line. At each point in the trajectory we solve the inverse kinematics to get the value of joint variables theta1, theta2 and d3. A few screenshots of the simulation are shown below. This can be extended for any curve on 3-D space, provided the points lie within the workspace of the DaVinci robot.

III. Performance Experiments

As we have mentioned earlier, one of the characteristic features of the DaVinci robot is the parallelogram structure. This plays a vital role in the rigidity especially in a surgical robot like DaVinci where accuracy is necessary to execute the desired task. To experiment this, we assume an error of 0.1

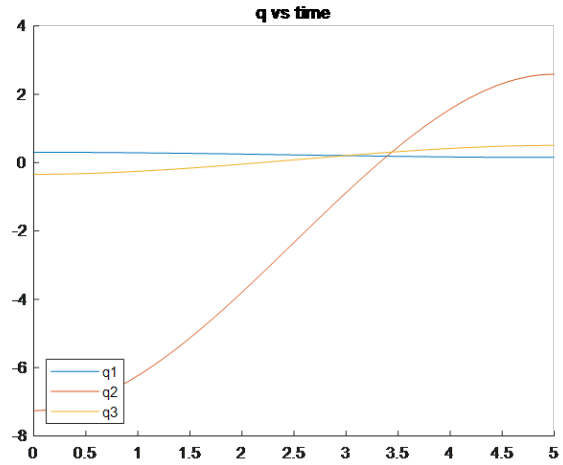


Fig. 5. Joint angle vs Time

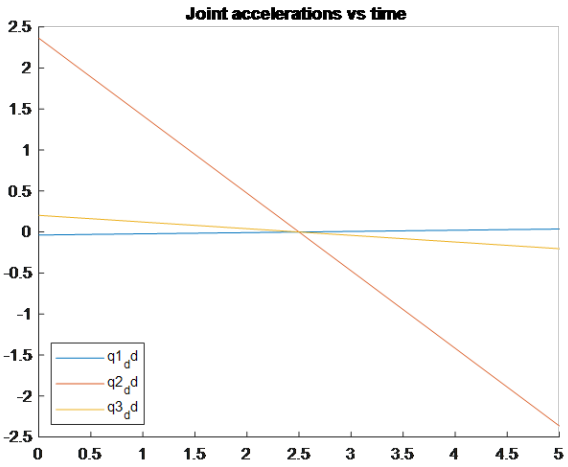


Fig. 6. Joint Velocity vs time

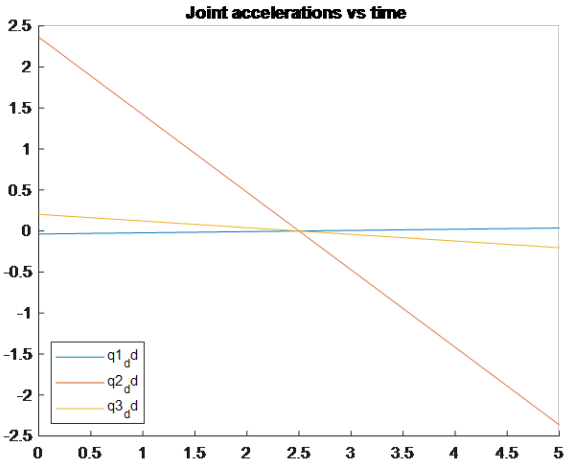


Fig. 7. Joint Acceleration vs time

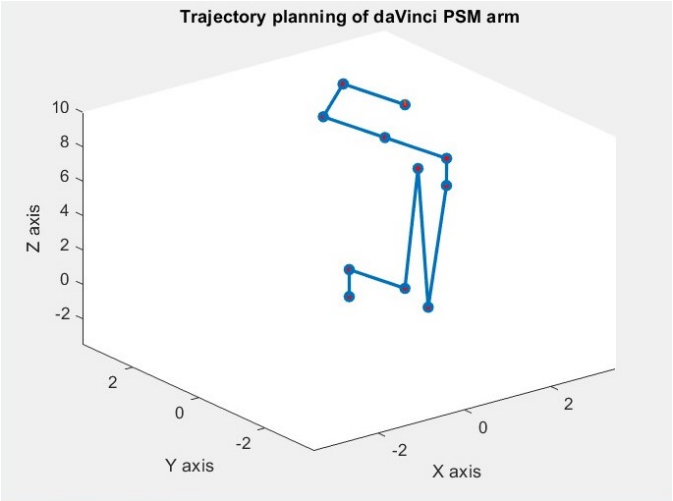
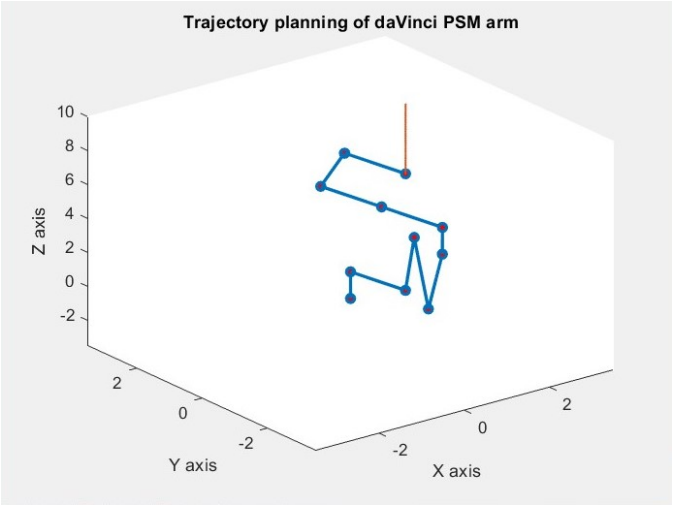
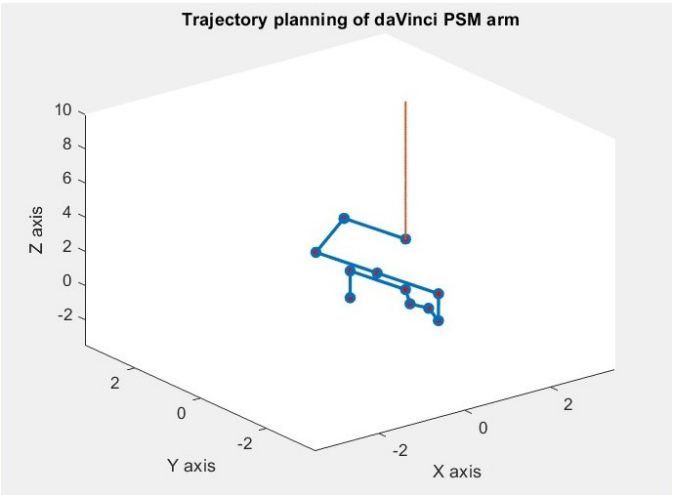


Fig. 8. Trajectory Simulation



g

Fig. 9. Trajectory Simulation



g

Fig. 10. Trajectory Simulation

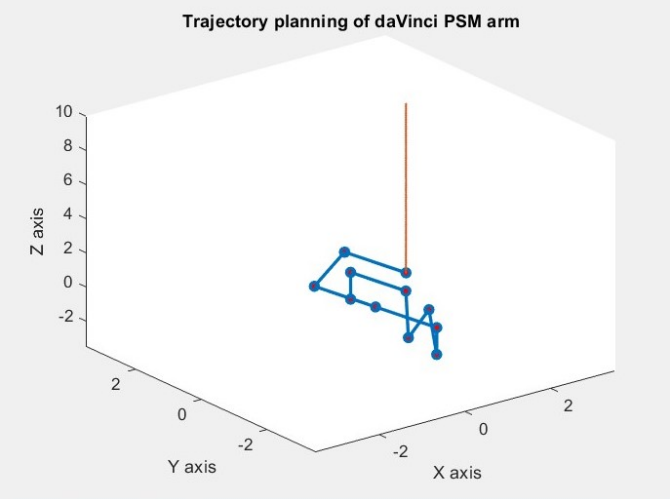


Fig. 11. Trajectory Simulation

radians in 2 and observe the effect in the final transformation matrix. As shown in the table below the change is very small. This is due to the parallelogram structure present.

TABLE I
CHANGE IN POSITION DUE TO ERROR

θ_2	X	Y	Z
Without Error	2.2322	0.3428	2.0459
Error = 0.1 rad	2.2293	0.3424	1.9459

A. *Effect of Joint Variables on Actuator Forces*

The actuator forces for a set of values for: $\theta_1, \theta_2, d_3, \theta_4, \theta_5, \theta_6, \dot{\theta_1}, \dot{\theta_2}, \dot{d_3}, \dot{\theta_4}, \dot{\theta_5}, \dot{\theta_6}, \ddot{\theta_1}, \ddot{\theta_2}, \ddot{d_3}, \ddot{\theta_4}, \ddot{\theta_5}, \ddot{\theta_6}$

TABLE II
ACTUATOR FORCES

T1 (x 10-3 Nm)	T2 (x 10-3 Nm)	T3 (N)	T4 (x 10-3 Nm)	T5 (x 10-3 Nm)	T6 (x 10-3 Nm)
37.6800	-78.3657	-1.8340	37.6800	29.3871	37.6800

To see the effect of each joint variables, we plot graphs varying one of the joint variables keeping other variables constant. We assume the variation joint variables with respect to time as $\theta(t) = t^3$.

CONCLUSIONS

We have solved the forward kinematics of the DaVinci robot by finding its DH parameters. The workspace of the manipulator was plotted using MATLAB. Using the transformation matrices, we found the Jacobian of the manipulator. For the dynamic analysis we followed the Euler-Lagrangian approach. We found the kinetic and potential energies and derived the Lagrangian equations. Using this we found the forces and

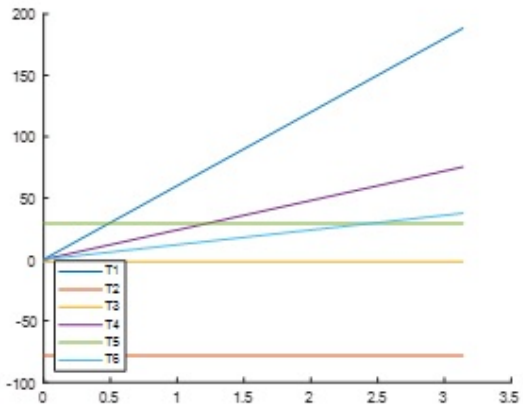


Fig. 12. Torque Vs. θ_1

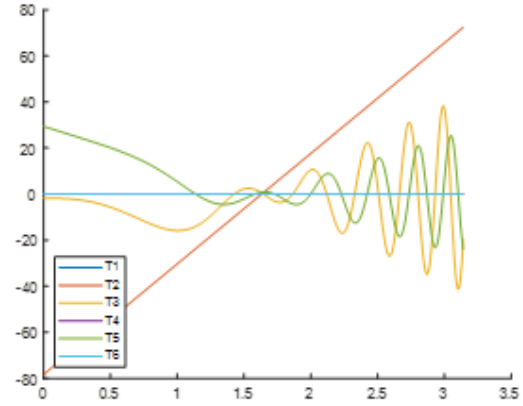


Fig. 13. Torque Vs. θ_2

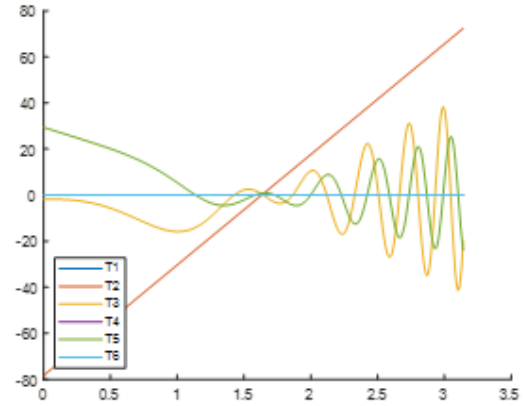


Fig. 14. Torque Vs. d_3

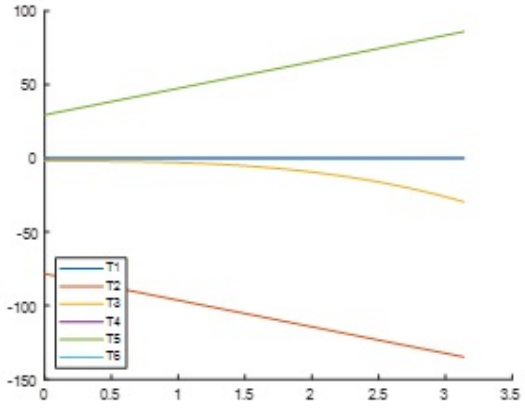


Fig. 15. Torque Vs. θ_4

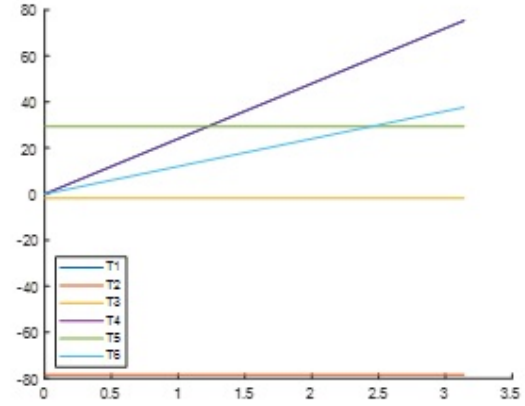


Fig. 16. Torque Vs. θ_5

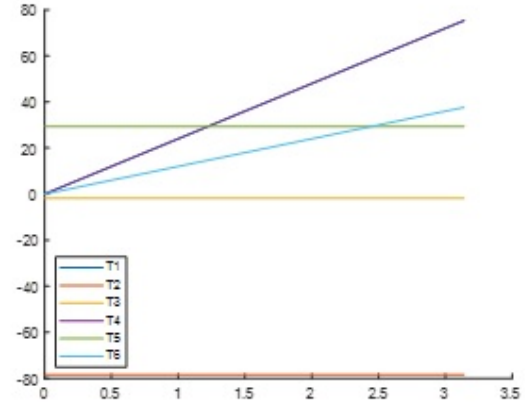


Fig. 17. Torque Vs. θ_6

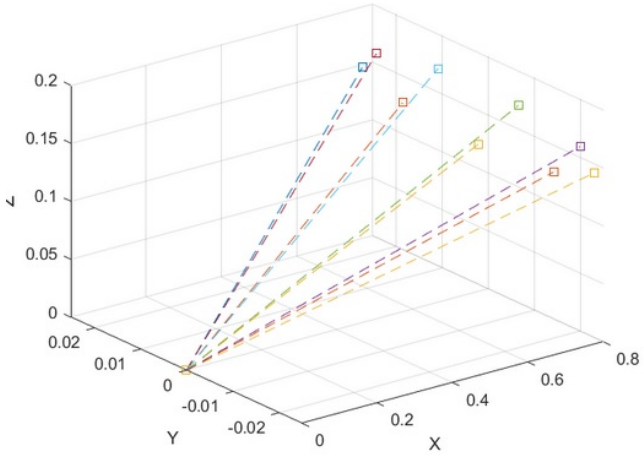


Fig. 18. $Varying\ \theta_1\ from\ -\pi/2\ to\ \pi/2$

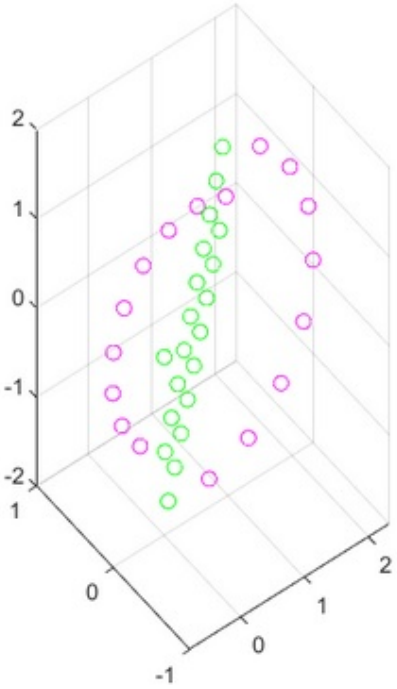


Fig. 19. $Varying\ \theta_1\ from -\pi/2\ to\ \pi/2\ and\ varying\ d_3\ from\ 0\ to\ 3$

torques which is to be generated by each actuator in the robot to get the desired forces and toque on the end-effector or the cartesian forces.

FUTURE WORK

Future work will be devoted to solving the inverse Kinematics and planning the trajectory of this robot. We will present a 3-D visualization of the motion of the robot and trajectory planning using Move-it API in ROS framework. Using these

results, we will extend this work by simulating an application of the robot.

REFERENCES

[1] R. A. Gondokaryono, Cooperative Object Manipulation with Force Tracking on the da Vinci Research Kit, Automation and Interventional Medicine (AIM) Lab, Worcester Polytechnic Institute, Worcester, Massachusetts, August 2018.

[2] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. Robot modeling and control, volume 3. Wiley New York, 2006.

[3] A. Munawar, Implementation of a Surgical Robot Dynamical Simulation and Motion Planning Framework, Automation and Interventional Medicine (AIM) Lab, Worcester Polytechnic Institute, Worcester, Massachusetts, May 2015.