Modeling and Simulation of a 6 Degree of Freedom (DOF) Pick and Place Robotic Arm

Rohith Jayarajan
University of Maryland, College Park
(*e-mail:* rohith23@terpmail.umd.edu)

Contents

1 Abstract

The modeling of a 6 Degree-of-Freedom (DOF) robotic arm is presented in this work. The 6 Degree-of-Freedom (DOF) robotic arm is based on the KUKA KR6 R700 sixx HM-SC which is a compact robot. The robot arm consists of 6 joints, all of which are revolute, and 6 links between these joints. To understand the kinematics of the robotic arm, the forward, inverse and velocity kinematics are computed. The forward and inverse dynamics of this robot are tested and the torques are computed when an external force is applied to the robot links. To fulfill its purpose of being able to perform pick and place operations in a collaborative environment, path planning is also performed for this robotic arm setup.

2 Introduction

2.1 The Robotic Arm

Generally, a robotic arm is a programmable mechanical arm which is inspired by the functionality of the human arm. It consists of joints and links where each joint is connected to two links. The two types of joints commonly used for developing a robot arm are;

1. **Revolute Joint:** Such type of joints impart a robot with the ability to perform a rotational motion about the axis vertical to the arm axes.
2. **Prismatic Joint:** These joints allow the robot to perform a linear motion.

The number of joints determine the degrees of freedom of the robotic manipulator. The end-effector of the robotic arm can be designed depending on the tasks needed to be performed.

The robot modeled in this work is based one the KUKA KR6 R700 sixx HM-SC robotic arm. This choice is based on the fact that it belongs to the Agilus series; a line of compact robots designed for applications which require high working speeds. The HM in the name indicates that it is available as a Hygienic Machine; meaning that all its components are hygienic and safe for direct use with foodstuff, pharmaceutical substances, and similar sterile environments.

Fig. 1. KUKA KR 6 R700 sixx HM-SC (1).

2.2 The Simulation Tools

For modeling and simulation of the 6 Degree-of-Freedom robotic arm, MATLAB and Simulink were used. Simulink was used to design a functional 3-D representation of the 6-DOF robotic arm. The Robotics System Toolbox provided by MATLAB helped to compute and verify the kinematics and dynamics of the robot arm.

3 Background

The origin of robotic arms date back to drawings from da Vincis notebooks which suggest the design of a complex robotic arm with 4 DOFs and a controller with some programmability. In 1961, the first robotic arms for industrial use were developed by Unimate which then paved way for the famous PUMA arm. The Stanford arm by Scheinman designed in 1969 was one of the first robotic manipulators built for computer control (2).

The Denavit - Hartenberg (DH) convention introduced in 1955 by Jacques Denavit and Richard Hartenberg is a very useful method to link spatial kinematic chain frames using four parameters. This convention is widely used to model the kinematics of a robotic arm in most cases.

The DH convention gives us the forward kinematics. Forward kinematics is used to determine the position and orientation and of the end effector for given values of joint variables. And the reverse problem is that of determining values of joint variables given an end effector position. This is called inverse kinematics. Whereas velocity kinematics involves finding the linear and angular velocities of a given end effector using the Jacobian matrix. These three constitute the robot kinematics. Dynamics deals with the relationship between forces, torques and motion (5).

Robotic arms have since developed dramatically with several manufacturers like KUKA, Staubli, ABB, Fanuc, etc building complex yet simple to operate robotic manipulators. The manipulators being built span not only open loop chains like the Stanford arm or the work in this project but also closed chain robotic manipulators. The end effector of robotic arms is also being used today in much complex systems such as the da Vinci surgical robot for high precision surgery. The Mars Curiosity rover also had a robotic arm with three joints for manipulation tasks.

4 Assumptions

The assumptions used while modeling and designing the robot arm were:

1. All the surfaces are frictionless (sliding friction and viscous friction).
2. All the links are cylindrical in shape.
3. Joint interferences are neglected.
4. There is no backlash.
5. All joints have only a single degree of freedom.
6. The co-ordinate of the object to be picked with respect to the world frame is known.
7. The co-ordinate of the destination with respect to the world frame where object must be placed is known.

5 Implementation

The stages of the project can be split into the following parts:

5.1 Denavit Hartenberg (DH) Parameters

DenavitHartenberg (DH) convention is a commonly used convention used in robotics applications to select the frames of reference. In the DH convention, each homogeneous transformation is A_i is a product of 4 transformations which in order are- rotation about $z-axis$ by an angle θ, translation about $z-axis$ by distance d, translation about $x-axis$ by distance a followed by a rotation about $x-axis$ by an angle α. We can rewrite this in an equation form as

$$A_i = Rot_{z,\theta i} Trans_{z,di} Trans_{x,ai} Rot_{z,\alpha i} \tag{1}$$

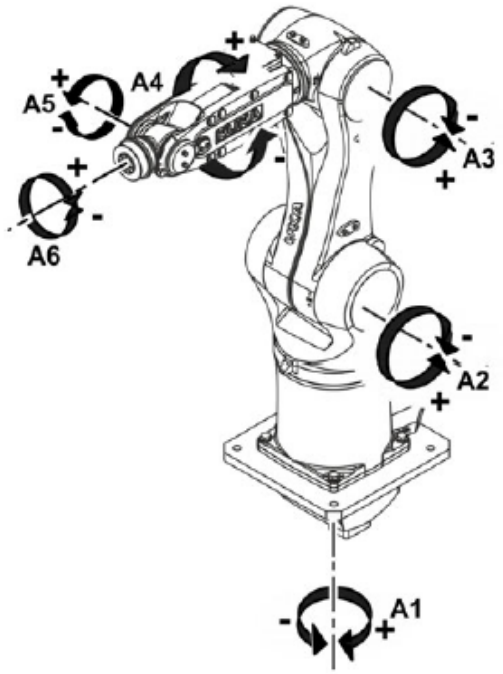

Fig. 2. Rotational axes of KUKA KR 6 R700 sixx HM-SC (1).

The values θ_i, a_i, d_i, and α_i are related to the link and joint i. Here, θ_i is called as the joint angle, a_i the link length, d_i the link offset and α_i is the link twist. The names given are related to the geometric relationship between the two coordinate frames. If a joint i is of the type revolute, then the joint variable is θ_i and if the joint is prismatic, then the joint variable d_i.

The DH table formed for the robot in this work is given by Table 5.1 using dimensions provided in Figure 3 and the rotation axes are shown in Figure 2.

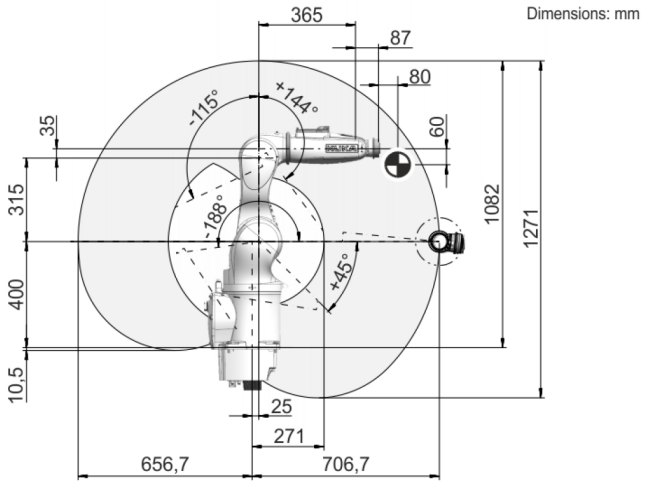

Fig. 3. Dimensional specifications of KUKA KR 6 R700 sixx HM-SC (1).

DH Table

Link	Transition	a_i	α_i	d_i	θ_i
1	0-1	25mm	90°	400mm	θ_1
2	1-2	315mm	0°	0	$\theta_2 + 90°$
3	2-3	35mm	90°	0	θ_3
4	3-4	0	-90°	365mm	θ_4
5	4-5	0	90°	0	θ_5
6	5-6	0	0°	0	θ_6

5.2 Designing the Robot Model using Simulink

Designing the model of the robot is necessary as without the 3-D model, it will be a lot tougher to see the results of our modeling in play. To design our 6-DOF arm, Simulink software has been used. It allows the user to create and manipulate 3-D objects and also observe the simulation.

The step by step construction of the Simulink model is given by:

1. The first and important blocks that need to be added to the Simulink model are Solver Configuration, World Configuration, and the Mechanism Configuration. All the other blocks of the model are connected to these three blocks.
2. After that, the base of the robot is built using a solid.
3. Construction of a Joint: Firstly, rigid transforms are used to change orientation and translation of the frame to attach the new joint in the right geometric manner. The, we attach the joint by using "Revolute Joint" from Simscape→Multibody→Joints in the Simulink Library Browser. Care must be taken that the joint is never accidentally placed inside a solid.

4. Addition of a Link: Once the joint is placed in our model, we need to add the link which should rotate (in our case) due to the previously added joint. Before adding the solid from Simscape→Multibody→Body Elements in the Simulink Library Browser, we should again change the frame by using rigid body transformation to accurately accommodate the solid.
5. Step 3 and 4 are repeated to build the entire 6 DOF arm model.
6. Now that we have all the joints and links in place, we need to provide some input to the joints so that our robot end- effector starts to move. For this we double click on the revolute joint block and select Torque→Automatically Computed and Motion→Provided by Input under the Actuation menu.
7. All we need to actuate the revolute joints is an input. We take this input from the workspace. To do this, we connect a "Constant" block to the revolute joint via a S-PS converter, both of which are present in the Simulink Library Browser. To setup the S-PS block, we double click it and make the following changes.Under Units, set "Input signal unit" to radian. Under Input Handling, change Filtering and derivatives→"Filter input, derivatives calculated" and Input filtering order→"Second-order filtering"
8. Set the simulation stop time to 100 units and then simulate the project. The robot will achieve the end-effector position based on the inputs given in the form of joint angles.

5.3 Designing the Robot Model using Robotics System Toolbox

The steps which follow describe the procedure used to design the 6 DOF arm using the Robotics System Toolbox which is optimized for robotics applications.

1. To create our robotic arm, we start by building the base of the robot which is the rigid body tree. This is done by using the RigidBodyTree() object in the robotic systems toolbox. All the joints and links are added to this rigid body tree.
2. Since we are using the DH convention for our problem, we have to specify the DH table we obtained in 5.1.
3. We now build each body of the robot and specify the joint and joint type along with it. All the joints for our robot is revolute, so add joints by using the .Joint('joint name', 'revolute') object.
4. Now we have to attach frames to each joints and links. This is where the DH parameters we specified in step 2 comes in play. Each joint is attached with a frame using the elements from the aforementioned DH table and using the 'dh' argument in the function setFixedTransform.
5. To complete the design of our 6 DOF arm we add each joint to the body we earlier mentioned in step 3 and link the bodies in order such that we have the base added first and the 6th body (6 joint and link) at the end.
6. Calling the function showdetails() gives us the details of our rigid body tree and the robot can be visualized using the show() function.

5.4 Forward Kinematics

Forward Kinematics is the problem where the end-effector position is to be computed when the joint angles of the robot manipulator are given. For calculating the Forward Kinematics, we use the DH parameters and construct the A-matrices using the equation 2

$$A_i = \begin{pmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

From our DH table, we get the values of A_1, A_2, A_3, A_4, A_5, and A_6. Using these values of A_i, we calculate the transformation matrices T_1, T_2, T_3, T_4, T_5, and T_6.

The value of T_6 has high importance for computing the forward kinematics. $T_6 = A_1 * A_2 * A_3 * A_4 * A_5 * A_6$ is given in the equation 3

$$T_6 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & o_x \\ r_{21} & r_{22} & r_{23} & o_y \\ r_{31} & r_{32} & r_{33} & o_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

The values o_x, o_y and o_z give us the end-effector coordinates. So, for any value of joint angles $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ and θ_6 we can compute the end-effector position as we get $[o_x, o_y, o_z]^T$ from T_6.

5.5 Inverse Kinematics

Inverse kinematics is the inverse problem where the joint angles of the manipulator are to be computed when the end-effector position is known. The inverse kinematics of the 6 DOF arm was computed using the *.InverseKinematics* System object function given in the Robotics System Toolbox. The *.InverseKinematics* object creates an inverse kinematics solver which computes the joints of the robot given the for a given end-effector position based on the rigid body tree model. The arguments that come inside the object are 'RigidBodyTree' and the value which specifies that we are computing inverse kinematics based on the rigid body tree we have created for our 6 DOF robotic manipulator arm.

5.6 Velocity Kinematics

The values of transformation matrices T_1, T_2, T_3, T_4, T_5, and T_6 computed for section 5.4 come into use again when we have to find out the velocity kinematics for the robotic manipulator. As we have only revolute joints in our robot, the equations to compute linear Jacobian and angular Jacobian are given in equations 4 and 5 respectively.

$$J_{vi} = z_{i-1} \times (o_n - o_{i-i}) \tag{4}$$

$$J_{\omega i} = z_{i-1} \tag{5}$$

for revolute joint i, where z is the coordinate of z axis and o is the coordinate of the end-effector origin for joint/link i.

The values z_0 and o_0 are $[0,0,1]^T$ and $[0,0,0]^T$ respectively. z_n and o_n are computed from the values in $T_n(1:3,3)$ and $T_n(1:3,4)$ respectively where $n = 1,2,3,4,5,6$.

The value of the velocity Jacobian matrix comes out as given in equation 6

$$J = \begin{pmatrix} J_v \\ J_\omega \end{pmatrix} \tag{6}$$

and the linear and angular velocities of the robotic arm are given by equations 7 and 8 respectively

$$v_n^0 = J_v \dot{q} \tag{7}$$

$$\omega_n^0 = J_\omega \dot{q} \tag{8}$$

5.7 Calculation of Forward Dynamics and Inverse Dynamics using Robotics System Toolbox

Computing forward and inverse dynamics for a robotic system is relatively easy using the functions given in the Robotics System Toolbox. Forward dynamics is used to calculate the motion of the robot using the internal and external forces which are known. Whereas. inverse dynamics is used to compute the internal forces, torques and external forces from the motion of the robot.

The steps to compute the forward dynamics using the MATLAB toolbox are:

1. The data format for all calculations of dynamics in the Robotics System Toolbox is either one of ”row” or ”column”. So the first step is to set the data format to type ”row”.
2. After this, the gravity is set and the home configuration of the robot is used for further processing.
3. Now for the robot to have an angular (joint is revolute) acceleration, it must experience a force. So we provide the force by using a function called externalForce() which takes in the robot model, the end-effector, the external forces applied, and the configuration of the robot as arguments.
4. Finally, the function forwardDynamics() is used to calculate the value of the joint (angular) accelerations of the robot.

The steps to compute the inverse dynamics using the MATLAB toolbox are:

1. As mentioned earlier, the data format is set to type ”row”, gravity is set, and the robot is given a random configuration.
2. The joint torques are calculated by using the function inverseDynamics() which takes the robot model and its configuration as arguments. This is the procedure to compute the inverse dynamics from a random static configuration.
3. To compute the joint torque which allow the robot to counter external forces, we can provide external forces on any link of our choice, or even our end-effector and then use our toolbox function.

5.8 Path Planning

The problem in path planning is to find the trajectory that connects the initial configuration to final configuration and at the same time satisfying the velocity, acceleration, and position constraints of the endpoints. In this project, a configuration is given to the robot and the robot is made to trace the path to a desired endpoint from this configuration. The steps followed to perform path planning using the Robotics System Toolbox are:

1. First, it is important to ascertain that we have a correctly designed robot with all the links, joints and the end-effector constructed using the DH convention.
2. Then once we know the start location and end location of the robots end-effector, we must define the trajectory that the robot end-effector should trace. A spline trajectory is used for this robot; which is defined as a continuous curve that passes through all the given set of points.
3. Inverse kinematics is used to find the solution for the configurations of the robot that achieves the given collection of points (end-effector coordinates).
4. All the joint configurations that achieves the end-effector position are stored and used afterwards to animate the solution for the robot tracing the trajectory.

6 Results

6.1 Design of the Robot Model using Simulink

The steps given in section 5.2 were followed and a Simulink design was created which is shown in Figure 5.

The links of robotic arm were colored so that all the 6 links could be easily visualized considering robot dimensions are in the range of millimeters. The 3-D model of our robot is shown in the figure 4 where the robot is in its home configuration i.e. all the joint angles have the value of 0°. The world coordinate frame axes are shown at the bottom left of the figure.

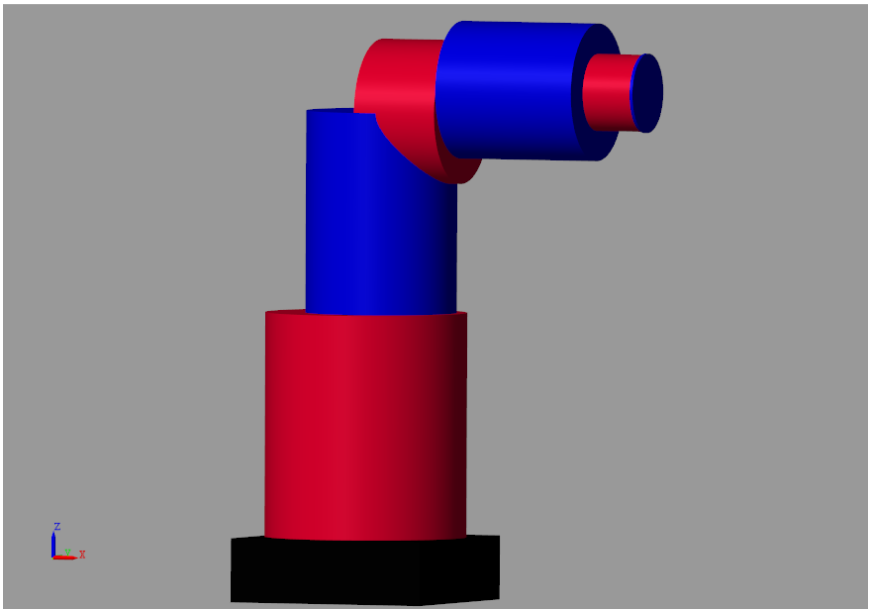

Fig. 4. Simulink 3-D model of the 6-DOF robot arm.

This model is an accurate representation of our robot arm except the fact that we are working under the assumption that all the links are cylindrical in shape. On giving sine wave input to these joints, the robotic arm model was observed to produce the desired rotations thereby changing the end effector position.

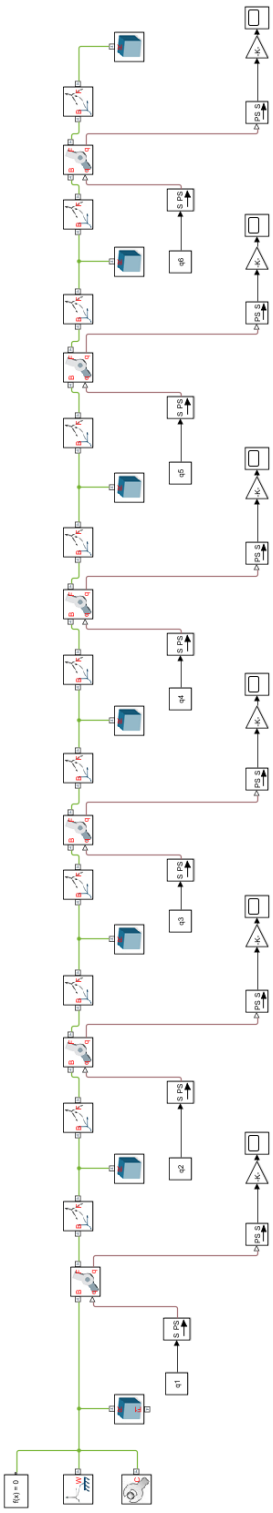

Fig. 5. Simulink block diagram.

6.2 Design of the Robot Model using Robotics System Toolbox

The steps given in section 5.3 were followed and a design of the robotic arm was created using the Robotics System Toolbox which is shown in Figure 6. The world coordinate axes are shown in the figure. This is a minimal representation but it simulates the behavior of the robotic arm accurately. We get this output using the show(SixDOFRobot) function which displays our robot design based on the DH parameters passed to the rigid body tree. The figure shows only four links but this is because of the fact that the coordinate frame is chosen in such a way that origins of 2 frames are taken at one point. Doing this simplifies our DH table and the calculations of A_I, T_i matrices also become much more easier.

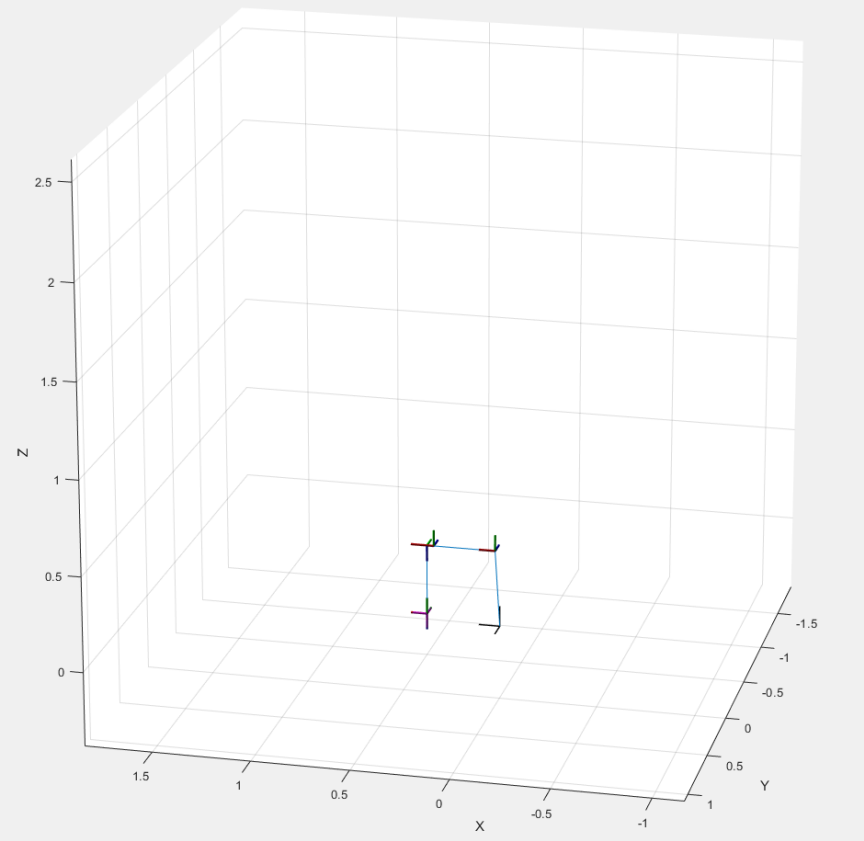

Fig. 6. Robotics System Toolbox model of the robot.

The showDetails() function gives details of the rigid tree body structure of our manipulator arm about the number of joints, name of body, joint name, joint type, parent (the component to which the body is added) name and the child (the next component added to a body) name. A tool body is attached to the center of last link so that we can calculate the kinematics for it which acts like our end-effector.

```
Command Window
----------------------
Robot: (7 bodies)

 Idx      Body Name    Joint Name    Joint Type      Parent Name(Idx)    Children Name(s)
 ---      ---------    ----------    ----------      ----------------    ----------------
   1        body_1         jnt_1      revolute              base(0)     body_2(2)
   2        body_2         jnt_2      revolute           body_1(1)      body_3(3)
   3        body_3         jnt_3      revolute           body_2(2)      body_4(4)
   4        body_4         jnt_4      revolute           body_3(3)      body_5(5)
   5        body_5         jnt_5      revolute           body_4(4)      body_6(6)
   6        body_6         jnt_6      revolute           body_5(5)      tool(7)
   7          tool          fix1         fixed           body_6(6)
----------------------
```

Fig. 7. Details of the rigid body tree design.

6.3 Forward Kinematics

The home configuration of the robot is which is given in Figure 4

Now to check the forward kinematics, we set the joint angles to the following values: $\theta_1 = -10°, \theta_2 = -20°, \theta_3 = -30°, \theta_4 = -40°, \theta_5 = -50°$, and $\theta_6 = -60°$. We observe the end effector position to be $[0.0629, -0.0111, 0.0308]^T$ which matches with the theoretical result we derive using the transformation matrix. The configuration of the robot and end-effector is shown in figure 8

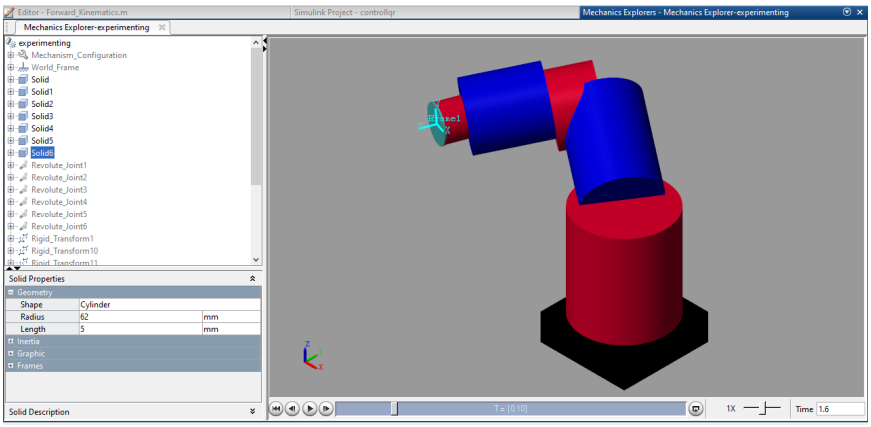

Fig. 8. Isometric view of the robot subject to
$\theta_1 = -10°, \theta_2 = -20°, \theta_3 = -30°, \theta_4 = -40°, \theta_5 = -50°$, and $\theta_6 = -60°$.

The values of the Transformation Matrix T_6, rotation matrix R_6 and end-effector position o_6 for joint variables $q_1, q_2, q_3, q_4, q_5, q_6$ are given in the Appendix from page 29 to page 38.

6.4 Inverse Kinematics

Inverse kinematics is computed by using the .InverseKinematics() object whose parameters are our 'RigidBodyTree' and the name of our rigid body tree. This object is equated to a variable. Finally we specify our end effector (which is the tool body), the transformation matrix T_6, weights for pose tolerances, and the initial guess. The weight is a 6 value vector.

The first three elements denote the weights on the error in orientation for the desired pose whereas the last 4 elements denote the error in xyz position for the desired pose. And initial guess as the name suggests is the guess of the configuration of robot which helps to guide the inverse kinematics solver.

The output we get from the inverse kinematics solver are 2 structure arrays- configSol and solInfo. solInfo retrns the solution information as a structure of the following fields-

1. Number of iterations the algorithm takes to solve the inverse kinematics
2. The number of times the algorithms got stuck in a local minimum and has to restart.
3. The magnitude of the solution's pose error compared to the specified end effectors pose.
4. Exit Flag
5. Status of the solve; whether it was a success or the best available solution to the problem.

The configSoln structure array is important as it consists the two important fields:

1. JointName - returns the name of the joint variable for which inverse kinematics has been solved.
2. JointPosition - returns the value of the solved joint variable.(in radian for joint angle)

So to verify if our solution for the inverse kinematics is correct, we take a random configuration with end-effector position of $[0.1241, 0.2535, 0.0634]^T$ shown in the Figure 9 as a Simulink model.

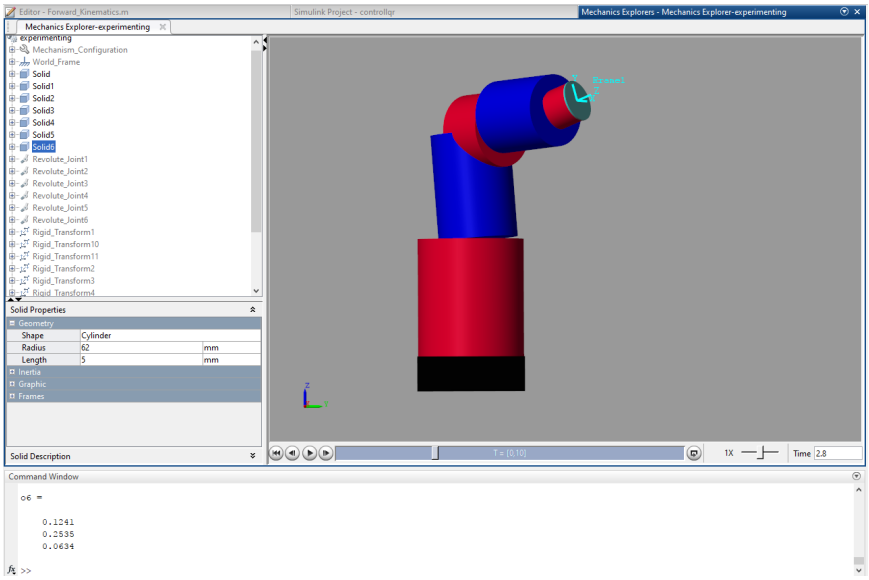

Fig. 9. Random configuration of robot- Simulink Model.

The inverse kinematics solver gives us the joint variables as $[1.1155, 0.0817, -0.3325, 2.5080, -0.3210, -0.9517]^T$ radians as shown in Figure 10 but we have to now verify if this is the correct inverse kinematics solution for our 6 DOF arm.

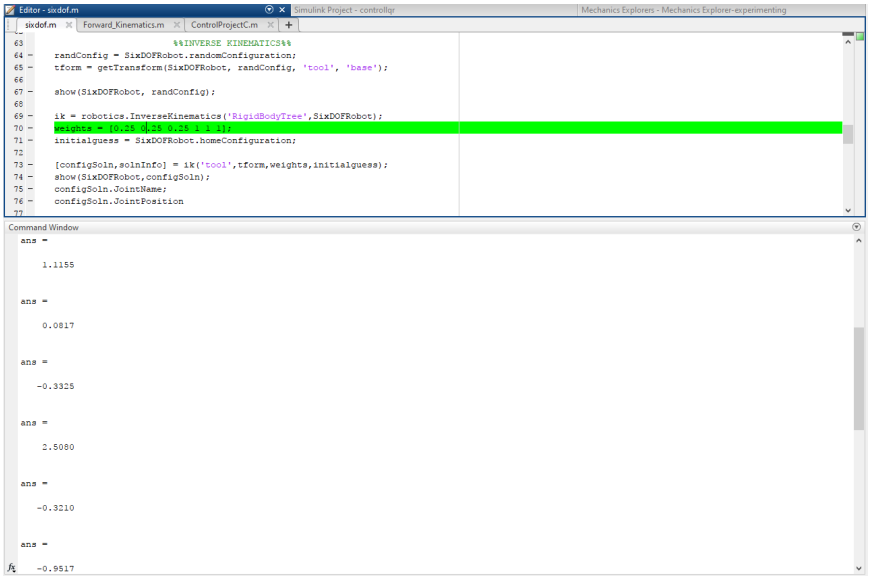

Fig. 10. Solution to the inverse kinematics for end-effector position of $[0.1241, 0.2535, 0.0634]^T$.

So to validate the results, the joint angle values are passed as θ_i values to the script written to compute forward kinematics. The solution is validated by the fact that the joint angle values calculated by inverse kinematics return us the correct end-effector position $[0.1241, 0.2535, 0.0634]^T$ for which the problem was solved. This is shown in Figure 11

Hence, inverse kinematics have been computed and verified. The outputs proving this result are on page 39 and 40.

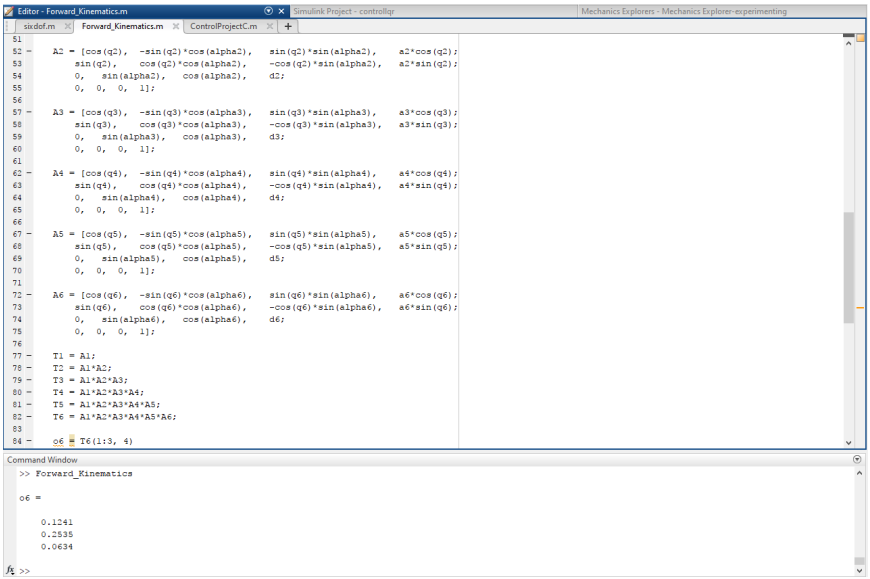

Fig. 11. Solution to the inverse kinematics for end-effector position of $[0.1241, 0.2535, 0.0634]^T$.

6.5 Velocity Kinematics

The steps mentioned in section 5.6 were used to solve for the velocity Jacobian of the 6-DOF robotic arm. The value of the Jacobain matrix for joint variables $q_1, q_2, q_3, q_4, q_5, q_6$ is given in the Appendix.

Using this Jacobian velocity matrix, we can compute the end-effector linear velocity as $v_n^0 = J_v \dot{q}$ and angular velocity as $\omega_n^0 = J_\omega \dot{q}$; where q is the 6-element vector consisting of the joint variables.

6.6 Forward Dynamics and Inverse Dynamics using Robotics System Toolbox

The steps given in 5.7 were followed. To compute the forward dynamics, for the home configuration of the robot, a force vector of $[0,0,0.3,0.2,0.9,0.8]$ is applied relative to the body frame of 'tool'. The resulting angular acceleration computed by the forward kinematics is give in figure 12

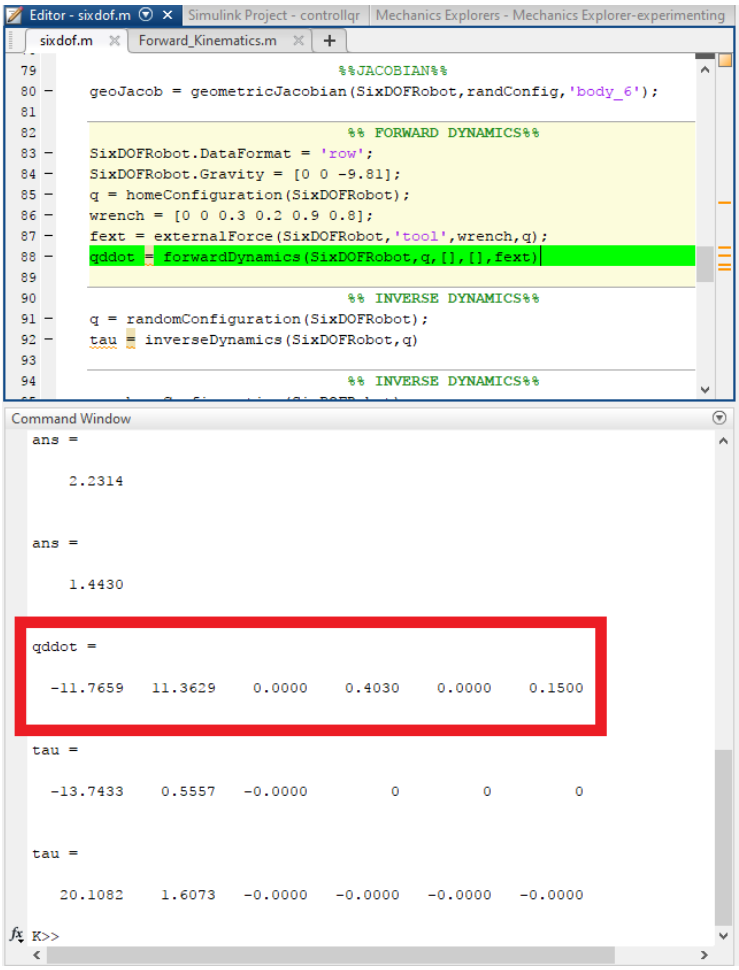

Fig. 12. Forward dynamics for home configuration and force vector $[0,0,0.3,0.2,0.9,0.8]$ applied relative to tool body of the robot.

In the case of inverse dynamics for static configuration of the robot, the joint torques were computed for a random robotic arm configuration and is shown in figure 13.

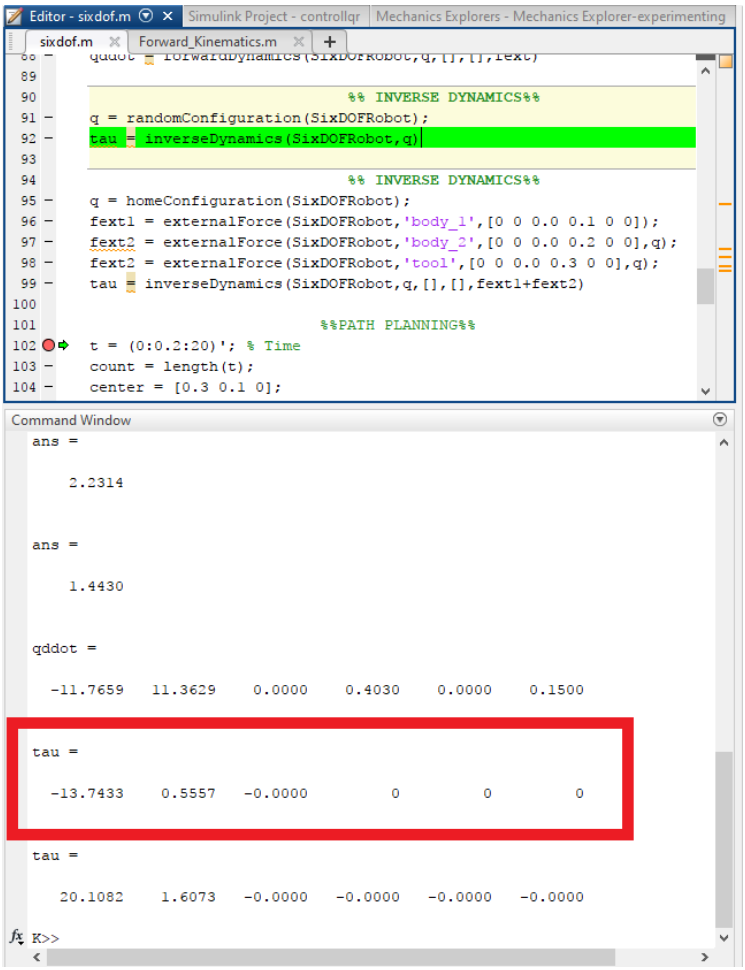

Fig. 13. Inverse dynamics for random static configuration of the robot.

When external forces $[0,0,0.0,0.1,0,0]$, $[0,0,0.0,0.2,0,0]$, $[0,0,0.0,0.3,0,0]$ are given to the robotic arm links 1,2 and tool respectively, the output torques shown in 14 should be applied to counter these forces.

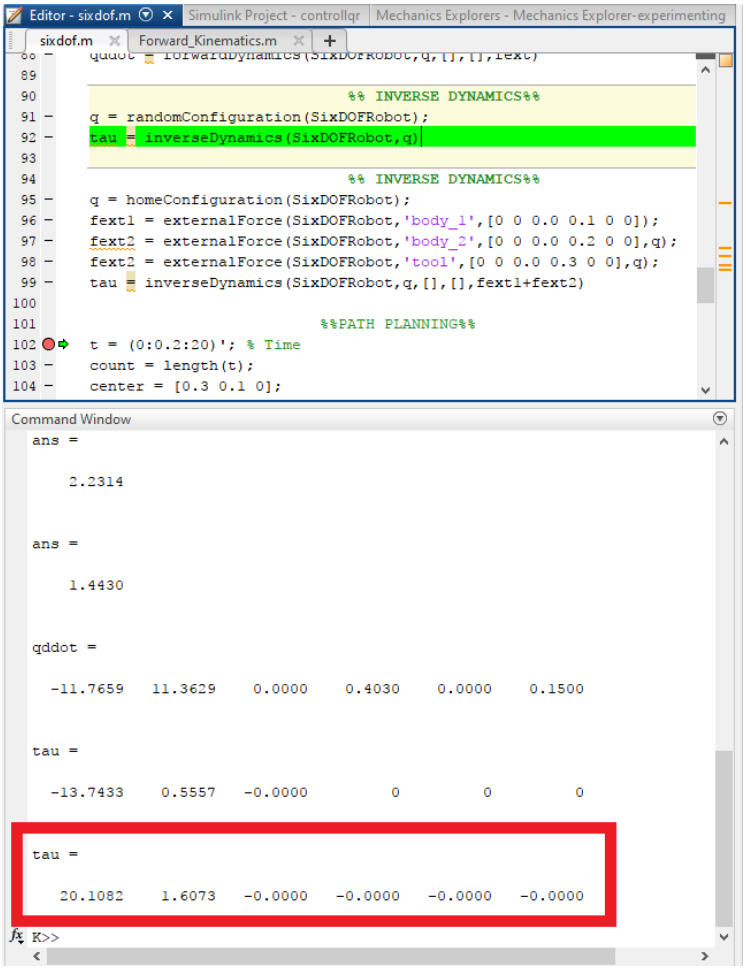

Fig. 14. Inverse dynamics of the robot to counter external forces.

6.7 Path Planning

Figure 15 is the animation plot of the robotic arm's end-effector and the curve in black color denotes the 2-D plot of the 3-D trajectory traced.

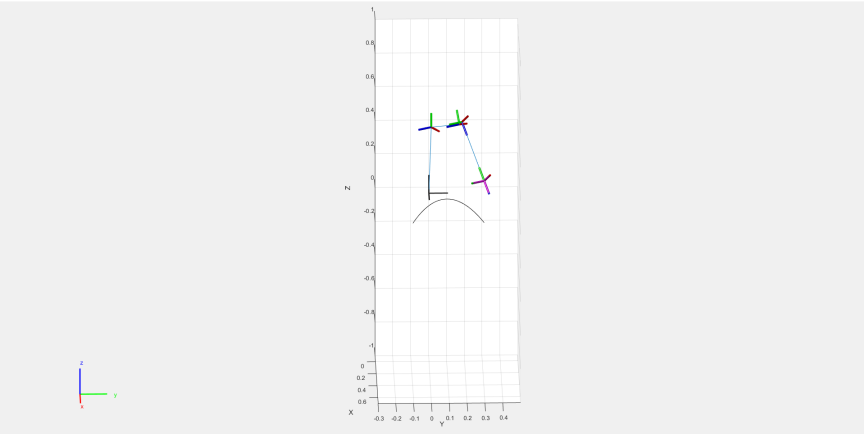

Fig. 15. Animation plot for path planning of the robot.

The animation plot confirms that the robot is able to trace the desired trajectory and also the ability to trace the desired trajectory indicates that the inverse kinematics solver is also correct.

7 Conclusion

This project successfully demonstrates the modeling of a 6-DOF robotic arm based on KUKA KR6 R700 sixx HM-SC manipulator arm. The designing of the robot was performed using both Simulink and the Robotics System Toolbox. The procedure followed to complete the modeling is explained with great detail in Chapter 5. The forward, inverse and velocity kinematics for the 6 DOF arm have been computed, tested and verified-

1. Forward Kinematics- For a given set of joint variables (angles) $\theta_i, i = 1,2,3,4,5,6$, the resulting configuration of the robot is obtained along with the end-effector position.
2. Inverse Kinematics- The inverse kinematics for the problem is solved; joint angles were solved for a given robot end-effector position. The accuracy of the solution for inverse kinematics was tested and proved by cross-verifying with the equation for forward kinematics.
3. Velocity Kinematics- Velocity Jacobian was calculated for the robotic arm and the equations for computing the linear and angular velocities of the end-effector were described.
4. Dynamics- The forward dynamics was computed using the Robotics System Toolbox which gave the values of joint accelerations based on the external force applied to the robotic arm. The inverse kinematics was calculated from the static configuration which gave us the value of the joint torques. Also the torque needed to counter act an external force was computed using the inverse kinematics solver.
5. Path Planning: Given a starting position and the desired end position of the robot end-effector, the path was traced by computing the inverse kinematics of all the points given by a trajectory. The ability to correctly trace the path is a testament to the success of this project.

8 Future Work

Firstly, the path planning concept used in this work can be replaced by using the concept of trajectory planning. Path planning is the procedure of finding and specifying the route from the initial position to the final position of the robot. Whereas, trajectory planning is the procedure of moving the robot arm from initial to final position using the velocity and acceleration information to deduce the smoothest and a faster trajectory.

The prospect of adding a vision sensor will impart the robotic arm with added capabilities to perform pick and place operations without explicitly specifying the coordinates of the object to be picked.

References

[1] https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/kr-agilus

[2] Moran, M.E. J Robotic Surg (2007) 1: 103. https://doi.org/10.1007/s11701-006-0002-x

[3] M. W. Spong, S. Hutchinson, M. Vidyasagar ,Robot Modeling and Control.

[4] MATLAB documentation https://www.mathworks.com/help/robotics/index.html

[5] ENPM 662 Lecture Notes by Prof. William S. Levine