Computed Torque Controller For A 6 DOF Robot Manipulator For Precise Motion Control.

Aniketh Reddy Seelam
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA 01605
Email: aseelam@wpi.edu

Marlon Scott
Electrical and Computer Engineering
Worcester Polytechnic Institute
Worcester, MA 01605
Email: mscott@wpi.edu

***Abstract*—This project deals with the control aspect of an ABB IRB 120 industrial robotic manipulator. It mainly deals with solving and verifying the Forward Kinematics, Inverse Kinematics, Dynamics and Control of the manipulator. We solved the kinematics of the robot and leveraged MATLAB's Robotics Toolbox for solving the dynamics of the robot. The motion control, being the major aspect, can be done in many different ways, out of which we chose to use Computed Torque controller. Overall, the end goal of the project was to enable the robot follow a rectangular trajectory requested by the user and visualize the results in RobotStudio. We used ABB RobotStudio for simulation of the robot and MATLAB for solving the Kinematics, Dynamics and Control.**

I. Introduction

Precise motion control is very important for various applications of industrial robots. It is important that the robot interacts with the objects in a smooth, precise and welcoming manner. This indicates the user that the robot is ready to interact with the environment in an appropriate way. For achieving the above said behavior, precise control of the speed and position of the robot is paramount. This ensures that the robot is not only safe for interaction with the environment, but also that the robot does not damage any object that it is interacting with.

This can be achieved using various control techniques such as Proportional Derivative (PD) [1], Proportional Integral Derivative (PID) [2], PD plus Feedforward, Computed Torque [3] or Model Predictive control [4]. We have decided to implement a Computed Torque controller due the advantages such as speed, simplicity and effectiveness.

MATLAB provides applications known as Add-Ons to extend its capabilities in engineering or mathematics towards a specific task. One such Add-On is known as the Robotics System Toolbox [5] (RST) which provides algorithms specifically useful for Robotic Applications. Since its introduction, RST has become more of a standard utility in the Robotics toolkit. Thus, we have decided to leverage RST for solving the dynamics of the robot and also to cross check our solution of Inverse Kinematics.

Use of actual hardware is prohibitively expensive and not accessible. Thus, we decided to leverage the availability of tools such as ABB RobotStudio to test, verify and visualize our developed algorithm. Figure 1 shows the ABB IRB 120 manipulator.

Fig. 1. ABB IRB 120 robot manipulator.

II. Problem Description

The problem that we tried to solve is to execute smooth rectangular trajectory following for point to point motion in the task space in a simulator. Another condition is that the robot's velocity at the corners of the rectangle is zero.Finally, it is necessary that the error between the desired and actual pose is minimal.

Given the technical specifications of the ABB IRB 120 Robot [6], we were required to develop the Forward and Inverse Kinematic (FK and IK) equations, as well as the Dynamical model. Development of the FK and IK equations will enable to us move from the joint space to the task space and vice versa. After the development of the dynamical model, we will be able to integrate our control technique. The instructions such as joint angles and the time(seconds) for the end-effector to travel to the instructed target need to be sent to our robot model in ABB RobotStudio to simulate our desired motion.

III. Kinematics

Kinematics is the study of the motion of the body including position, velocity and acceleration without considering the force causing the motion.

A. Forward Position Kinematics

The forward position kinematics generally referred as forward kinematics of a robot refers to the calculation of the position and orientation of its end-effector frame from the joint

angles. The end-effector frame is the frame of the robot at the end of its tool tip.

Referencing the product specification of the ABB IRB 120 robot manipulator, we assigned frames to the model joints as shown in Figure 2. $Frame_0(x_0, y_0, z_0)$ shown in the Figure 2 corresponds to the base frame of the robot where as $Frame_8(x_8, y_8, z_8)$ corresponds to the end-effector frame of the robot. The position at which $Frame_4, Frame_5, Frame_6$ and $Frame_7$ intersect is called the wrist position. The first three joint angles$(\theta_1, \theta_2, \theta_3)$ determine this wrist position.

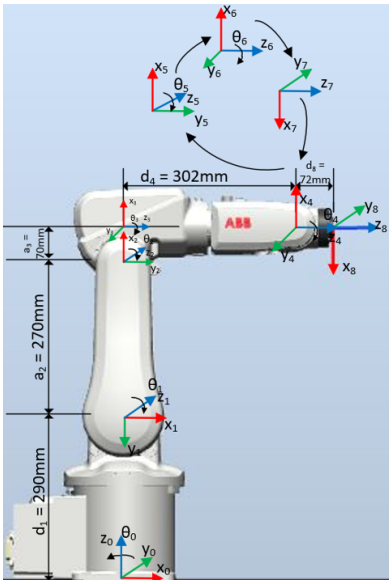

Fig. 2. ABB IRB 120 coordinate frames.

From these coordinate frames, the Denavit-Hartenberg parameters are developed as shown in Table I. The forward

DH Parameter Table				
Transformation	a(mm)	$\alpha(rad)$	d(mm)	$\theta(rad)$
T_{01}	0	$-\pi/2$	290	θ_1
T_{12}	270	0	0	$\theta_2 - \pi/2$
T_{23}	70	$-\pi/2$	0	θ_3
T_{34}	0	0	302	0
T_{45}	0	$\pi/2$	0	θ_4
T_{56}	0	$-\pi/2$	0	θ_5
T_{67}	0	0	0	$\theta_6 + \pi$
T_{78}	0	0	72	0

TABLE I
D-H PARAMETERS OF ABB IRB 120.

position kinematics returns a homogeneous transformation which contains a rotation and translation part. The translation part returns the position of the end-effector frame with respect to the base frame while the rotation part determines the orientation. The orientation is obtained by considering the ZYX Euler angle rotations. A generic homogeneous transformation of the robot to its end-effector looks like

$$T_{base-ee} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where x, y, z is the translation part which is nothing but the position of the end-effector.

B. Inverse Position Kinematics

Inverse position kinematics usually referred as just Inverse Kinematics is the exact opposite of forward position kinematics. The goal of inverse position kinematics is to obtain the joint configuration(s) which places the robot at the desired end-effector position and orientation. We used configuration(s) to remind that inverse position kinematics is generally a one-many solution for 6 DOF robots. The best joint configuration is picked based on range of motion and the joint limits. A 6 DOF robot with a wrist usually has at the maximum of 8 different solutions for Inverse Kinematics.

We have solved the Inverse Kinematics of ABB IRB 120 using Pieper's Solution. Pieper's approach works on the principle of separating the position solution for θ_1, θ_2 and θ_3 from the orientation solution to solve for θ_4, θ_5 and θ_6. Therefore, a geometrical approach is initially implemented to find the joint variables θ_1, θ_2 and θ_3 that determine the wrist position in space, while an analytical solution is applied to calculate the angles θ_4, θ_5 and θ_6 which describe the wrist orientation. The end-effector frame is usually a simple translation from the wrist's final frame.

1) Geometrical Solution For Wrist Position: Figure 3 shows the top view of the stick model of the ABB IRB 120 robot. Joint angle $\boldsymbol{\theta_1}$ is computed based on the wrist position (x, y) as shown below.

$$\boldsymbol{\theta_1} = atan2d(y, x) \tag{1}$$

Another solution for $\boldsymbol{\theta_1}$ which was not used for the purpose of this project is

$$\boldsymbol{\theta_1} = atan2d(y, x) + \pi \tag{2}$$

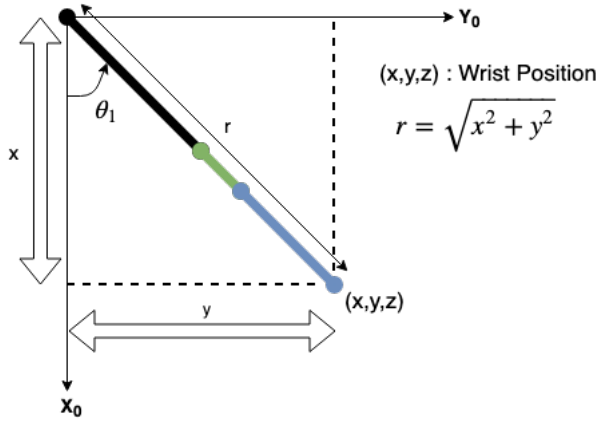

Fig. 3. ABB IRB 120 Stick Model Top View.

Figure 4 shows the side view of the stick model of the same robot. The links are color coded for clarity. The side view involves a lot more complexities and information than

the top view. The joint angles $\boldsymbol{\theta_2}$ and $\boldsymbol{\theta_3}$ are computed using information from Figure 3 and Figure 4.

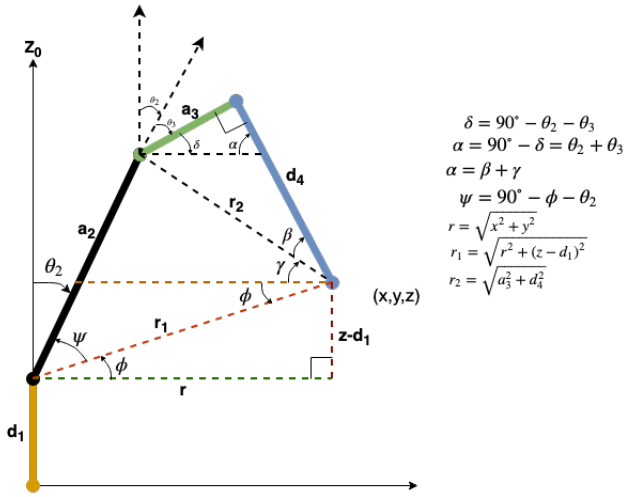

Fig. 4. ABB IRB 120 Stick Model Side View.

Cosine Rule shown in Figure 5 is used multiple times in this solution.

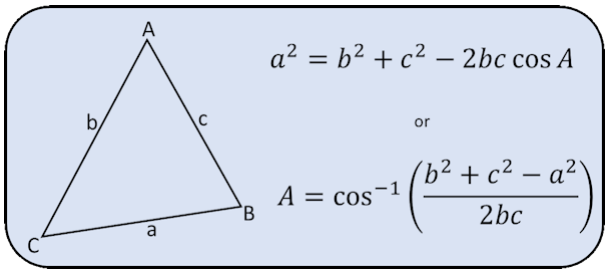

Fig. 5. Cosine Rule.

From Figure 4

$$\phi = atan2d(z - d_1, r) \tag{3}$$

$$\psi = acosd((a_2^2 + r_1^2 - r_2^2)/(2 * a_2 * r_1)) \tag{4}$$

$$\theta_2 = 90^\circ - \phi - \psi \tag{5}$$

Another solution for $\boldsymbol{\theta_2}$ is

$$\theta_2 = 90^\circ - \phi + \psi \tag{6}$$

To obtain solution for $\boldsymbol{\theta_3}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ must be solved first. Using Figure 4

$$\beta = atand(a3/d4) \tag{7}$$

$$\gamma = acosd((r_1^2 + r_2^2 - a_2^2)/(2 * r_1 * r_2)) - \phi \tag{8}$$

$$(or)\gamma = -acosd((r_1^2 + r_2^2 - a_2^2)/(2 * r_1 * r_2)) - \phi \tag{9}$$

Using $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ from above, $\boldsymbol{\theta_3}$ is obtained by

$$\theta_3 = \beta + \gamma - \theta_2 \tag{10}$$

Using $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$ and $\boldsymbol{\theta_3}$ obtained from here, the homogeneous transformation T_{04} till the wrist of the robot is obtained. The rotation part R_{04} of this transformation will be used to compute the joint angles $\boldsymbol{\theta_4}$, $\boldsymbol{\theta_5}$ and $\boldsymbol{\theta_6}$.

2) Analytical Solution For Wrist Orientation: The joint angles $\boldsymbol{\theta_4}$, $\boldsymbol{\theta_5}$ and $\boldsymbol{\theta_6}$ are determined from the rotation part R_{04} of the homogeneous transformation T_{04} obtained for the wrist position and the final orientation matrix computed from the input orientation angles. As we know, one of the inputs to the inverse kinematics problem are the orientations $\boldsymbol{A_x}$, $\boldsymbol{A_y}$ and $\boldsymbol{A_z}$ which correspond to the ZYX Euler angle rotation. Let this matrix be called $\boldsymbol{R_f}$ which is computed as shown below

$$R_f = Rot_z(A_z) * Rot_y(A_y) * Rot_x(A_x) \tag{11}$$

This R_f is also equivalent to the successive rotations due to the transformations from the base frame to the wrist. From the DH parameter table I, these successive rotations are nothing but the rotation part of T_{07} which is called R_{07} from here. The rotation matrix R_{04} which is due the successive rotations from base frame till $Frame_4(x_4, y_4, z_4)$ is obtained from T_{04} which was computed using $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$ and $\boldsymbol{\theta_3}$. To summarize, in short:

$$R_{07} = R_{04} * R_{47} = R_f \tag{12}$$

Using the above equation, we can clearly compute

$$R_{47} = inv(R_{04}) * R_f = (R_{04})^T * R_f \tag{13}$$

Here, both R_{04} and R_f are known which means R_{47} is nothing but a 3x3 matrix with all the elements known very precisely. Let this 3x3 matrix be represented as

$$R_{47} = \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \tag{14}$$

Also, R_{47} is nothing but the ZYZ Euler angle rotations at the wrist. This is equivalent to

$$R_{47} = Rot_z(\theta_4) * Rot_y(\theta_5) * Rot_z(\theta_6 + \pi) \tag{15}$$

Equation 15 can be expanded as

$$R_{47} = \begin{bmatrix} -c_4c_5c_6 + s_4s_6 & c_4c_5s_6 + s_4c_6 & c_4s_5 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 \\ s_5c_6 & -s_5s_6 & c_5 \end{bmatrix} \tag{16}$$

where $c_i = cos(\theta_i)$ and $s_i = sin(\theta_i)$.
Equating 14 and 16, we get

$$\theta_5 = atan2d(\pm\sqrt{g_{31}^2 + g_{32}^2}, g_{33}) \tag{17}$$

$$\theta_6 = atan2d(-g_{32}/sin(\theta_5), g_{31}/sin(\theta_5)) \tag{18}$$

$$\theta_4 = atan2d(g_{23}/sin(\theta_5), g_{13}/sin(\theta_5)) \tag{19}$$

For singular configurations i.e if $\theta_5 == 0^\circ or\, 180^\circ$,

$$if(\theta_5 == 0^\circ) \implies \theta_4 = 0^\circ, \theta_6 = atan2d(g_{12}, -g_{11}) \tag{20}$$

$$elseif(\theta_5 == 180^\circ) \implies \theta_4 = 0^\circ, \theta_6 = atan2d(-g_{12}, g_{11}) \tag{21}$$

This achieves the solution for Inverse Kinematics for the pose of the wrist. However, the input to the IK is not the pose

of the wrist but the end-effector. To get the pose of the wrist, the translation from the wrist to the end-effector's tip needs to be inverted. This translation in the DH table I is represented by T_{78} and the pose of the wrist is T_{07}. Using T_{08} and T_{78}, T_{07} is obtained as shown

$$T_{07} = T_{08} * (T_{78})^{-1} \tag{22}$$

The rotation and translation parts of T_{07} which correspond to the pose of the wrist are fed to the IK solver.

C. Forward Velocity Kinematics

Forward Velocity Kinematics is the problem of determining the cartesian velocity of the end-effector given the angular velocity of the joint angles. Geometric Jacobian(J) relates the cartesian velocity and the joint velocity as shown below

$$\dot{X} = J\dot{Q} \tag{23}$$

D. Inverse Velocity Kinematics

Inverse Velocity Kinematics is the problem of determining the joint angular velocities given the cartesian velocity of the end-effector. It is the opposite of Forward Velocity Kinematics. Geometric Jacobian(J) is again used to relate the joint velocities and the cartesian velocity as shown below

$$\dot{Q} = pinv(J) * \dot{X} \tag{24}$$

here **pinv** indicates pseudo-inverse as geometric jacobian **J** might not be a square matrix in all the cases.

IV. Dynamics

Dynamics also known as Kinetics is the relationship between the forces acting on a robot and the resulting motion of the robot. The general formula for the dynamical model of a robot is

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \tau \tag{25}$$

For the purpose of this project, as the involved robot is a complex model, we preferred to use MATLAB's Robotics System Toolbox to obtain the forward and inverse dynamics. A URDF(Unified Robot Description Format) model of the ABB IRB 120 robot is imported into MATLAB which is referred to as **IRB120** from here on. To enforce the effect of gravity on the robot, we need to define **IRB120**.Gravity. For the purposes of this project it was defined as

$$\textbf{IRB120}.Gravity = [0\ 0\ -9.80] \tag{26}$$

A. Forward Dynamics

Forward Dynamics of a robot is calculating the motion from known internal forces and/or torques and resulting reaction forces. Forward dynamics returns the joint accelerations given the joint torques and states. The joint accelerations are returned by the MATLAB function ***forwardDynamics*** [7]. It is used as shown below:

jointAccel = forwardDynamics(robot, configuration, jointVel, jointTorq) (27)

where *'configuration'* holds the current joint angles and *'robot'* is the imported urdf object **IRB120**.

B. Inverse Dynamics

Inverse Dynamics of a robot is calculating the joint torques required to perform the given motion. The joint torques are returned by the matlab function ***inverseDynamics*** [7] . It is used as shown below:

jointTorq = inverseDynamics(robot, configuration, jointVel, jointAccel) (28)

where *'configuration'* holds the current joint angles and *'robot'* is the imported urdf object **IRB120**.

V. Control

Control is the problem of controlling robots i.e to ensure a robot follows the commands of the user. There are many different control techniques which can be applied to achieve motion control. We have used **Computed Torque Control** due its simplicity and effectiveness at the same time.

A. Computed Torque Control

Computed Torque Control is based on the feedback linearization principle which is an approach that maps a nonlinear model into a linear closed-loop equation in terms of the state variables. The choice of the design parameters of the controller is trivial. The computed-torque control law is given by

$$\tau = M(\mathbf{q})[\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + K_v\dot{\widetilde{\boldsymbol{q}}} + K_p\widetilde{\boldsymbol{q}}] + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \tag{29}$$

where K_v and K_p are symmetric positive definite design matrices and $\widetilde{\boldsymbol{q}} = \boldsymbol{q_d} - \boldsymbol{q}$ denotes as usual, the error.

Equation 29 is computed from Equation 28 as shown below

$$\tau = massMatrix(IRB120, q)[K_v\dot{\widetilde{\boldsymbol{q}}} + K_p\widetilde{\boldsymbol{q}}] + inverseDynamics(IRB120, q, \dot{q}, \ddot{q}_d) \tag{30}$$

where **IRB120** is the imported IRB120 urdf object, massMatrix(**IRB120**,q) returns the mass matrix M(**q**) and the inverseDynamics method returns $M(\mathbf{q})\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$.

To close the loop, the torque obtained from Equation 30 is used in Equation 27 as shown below

$$\ddot{q} = forwardDynamics(IRB120, q, \dot{q}, \tau) \tag{31}$$

VI. Model Simulation

A. MATLAB

We created a MATLAB script that applies Computed Torque Control in the ODE45 solver for ABB IRB 120 robot. Our pre-determined targets are four corners of a rectangle that lie in the same x, y plane. We would like the robot to trace the rectangle with zero velocity at the corners, and travel from one corner to the other corner in 5 seconds making the total trajectory 20 seconds long. Our script also allows the user to substitute different targets for the four corner positions.

Once the ODE45 solution is obtained, MATLAB sends the six joint angles and the time to move to the next given configuration, to the model simulator, RobotStudio.

This information is sent via TCP IP communication protocol, where MATLAB acts as a client, and RobotStudio acts as the server.

B. ABB Robot Studio

All mathematically modeled motion derived in MATLAB for the ABB IRB 120 is visualized in ABB RobotStudio. We used the IRB 120 model available in the software library. We used the virtual controller provided by RobotStudio to enable the robot to move to the instructed joint positions.

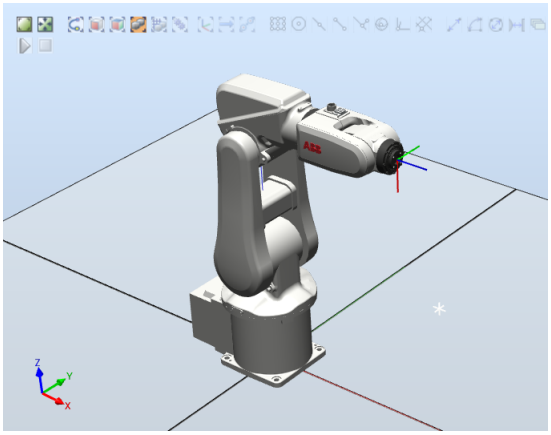

Fig. 6. ABB IRB 120 in home configuration in RobotStudio.

The instructions sent to the robot controller are written in RAPID, the programming language of ABB. We developed a RAPID program in RobotStudio that establishes communication with MATLAB, to send absolute joint angles and the time to complete the movement to the robot controller via TCP IP.

The instructions are sent in a data packet as a string. Each joint angle consists of 8 characters, including the decimal points. The time taken to move consists of 4 characters, including the decimal points.

Once all the data packets are received, the RAPID program extracts the values from these data packets and saves the numbers to the corresponding arrays for the joint angles and movement time. The robot controller is tasked immediately to move the joint's of the robot to the instructed angles within the instructed movement time.

VII. Results

The desired trajectory and the actual trajectory of the robot are shown in Figure 7 and Figure 8. As seen, the controller works as expected and the robot follows the desired trajectory.

Figure 9 shows a snapshot of the robot approaching one of the corners while following the trajectory.

Figure 10 shows joint angles vs time and Figure 11 shows joint velocities vs time while following the trajectory. As observed, the joint velocities are $0°/sec$ at 5,10,15 and 20 seconds when the robot approaches the corners. Overall, the graphs demonstrated successful execution of our controller by showing the ABB IRB 120 robot tracing the desired trajectory.

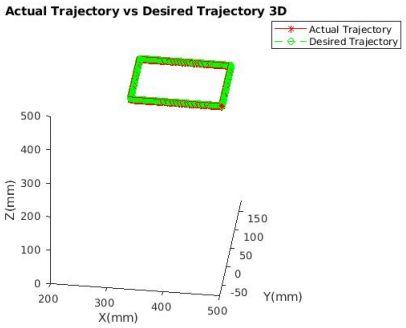

Fig. 7. Actual Trajectory vs Desired Trajectory in 3D.

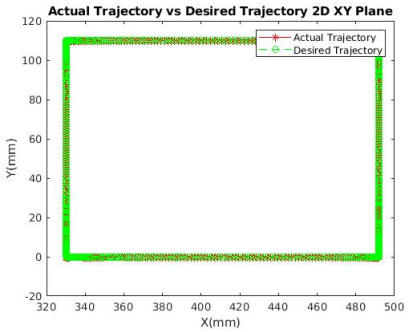

Fig. 8. Actual Trajectory vs Desired Trajectory in 2D X-Y Plane.

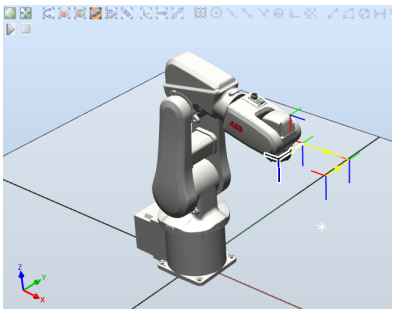

Fig. 9. ABB IRB 120 in action.

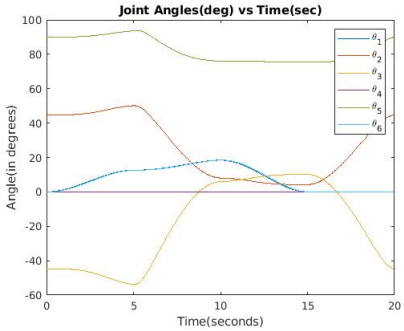

Fig. 10. Joint Angles during trajectory following.

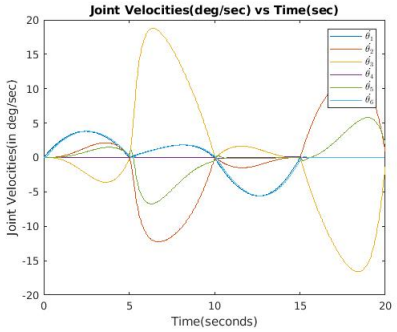

Fig. 11. Joint Velocities during trajectory following.

VIII. Conclusion

We have successfully developed and verified the Kinematics and Dynamics of the ABB IRB 120 robot and used them to develop a computed torque controller to follow a simple rectangular trajectory. The control input for each motor for every time stamp was obtained as well. We were also able to effectively send the calculated desired joint angles from MATLAB to RobotStudio and simulate the robot following the desired trajectory.

References

[1] Bullo, F. and Murray, R., "Proportional Derivative (PD) Control on the Euclidean Group, " in Proceedings of the third European Control Conference ECC 95. Rome, Italy, September 1995

[2] Ahmed Kharidege , Ding Jianbiao and Yajun Zhang, "Performance Study of PID and Fuzzy Controllers for Position Control of 6 DOF arm Manipulator with Various Defuzzification Strategies" MATEC Web of Conferences, ICMMR 2016.

[3] Grotjahn, M. and Heimann, B. (2000), "Symbolic Calculation of Robots Base Reaction-Force/Torque Equations with Minimal Parameter Set, " in: Morecki A., Bianchi G., Rzymkowski C. (eds) Romansy 13. International Centre for Mechanical Sciences (Courses and Lectures), vol 422. Springer, Vienna

[4] Timm Faulwasser, Tobias Weber, Pablo Zometa and Rolf Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot"

[5] MATLAB and Robotics System Toolbox 2019a, The MathWorks, Inc., Natick, Massachusetts, United States.

[6] ABB AB, Robotics, "Product Specification: IRB 120", Document ID: 3HAC035960-001 Revision: R. Se-721 68 Vsters Sweden.

[7] MATLAB and Robotics System Toolbox 2019a, The MathWorks, Inc., Natick, Massachusetts, United States. Retrieved from http://www.mathworks.com/help/robotics/ug/robot-dynamics.html