MATLAB Model

The Robotics Toolbox in MATLAB was used for the software approach to kinematic and dynamic problem. The toolbox contains many functions which can be used for kinematic, dynamic analysis and trajectory generation. IRB 6620 Model was created using 'rigidBodyTree' class as shown in Fig.1. rigidBodyTree is used to represent the connectivity of rigid bodies with joints.

```
Idx      Body Name    Joint Name    Joint Type      Parent Name(Idx)    Children Name(s)
---      ---------    ----------    ----------      ----------------    ----------------
  1             L1            J1      revolute              base(0)     L2(2)
  2             L2            J2      revolute                L1(1)     L3(3)
  3             L3            J3      revolute                L2(2)     L4(4)
  4             L4            J4      revolute                L3(3)     L5(5)
  5             L5            J5      revolute                L4(4)     L6(6)
  6             L6            J6      revolute                L5(5)     Gripper(7)
  7        Gripper            J7         fixed                L6(6)
```

Fig.1 IRB 6620 Tree Details

To create IRB 6620 using the Toolbox, the DH parameters from Table.2 are used as inputs. Links and joints are created using rigidBody and rigidBodyJoint classes respectively. setFixedTransform() function is used to feed in the DH parameter of a particular joint. It was made sure that the MATLAB Model accepts Modified DH parameters.

The joint limits of IRB 6620 were also added into the model using 'PositionLimits' instance of rigidBodyJoint class. The joints limits are depicted in Table.1.

Joint	Range (deg)
J1	[-170 170]
J2	[-65 140]
J3	[-180 70]
J4	[-300 300]
J5	[-130 130]
J6	[-300 300]

Table.1 IRB 6620 Joint limits

The Dynamic Properties like mass, CentreofMass and Inertial properties of the link are fed to the model. For visual purposes STL files of links were added into the model using addVisual() function after applying the required transformation between the Centre of STL file and joint coordinate frame.

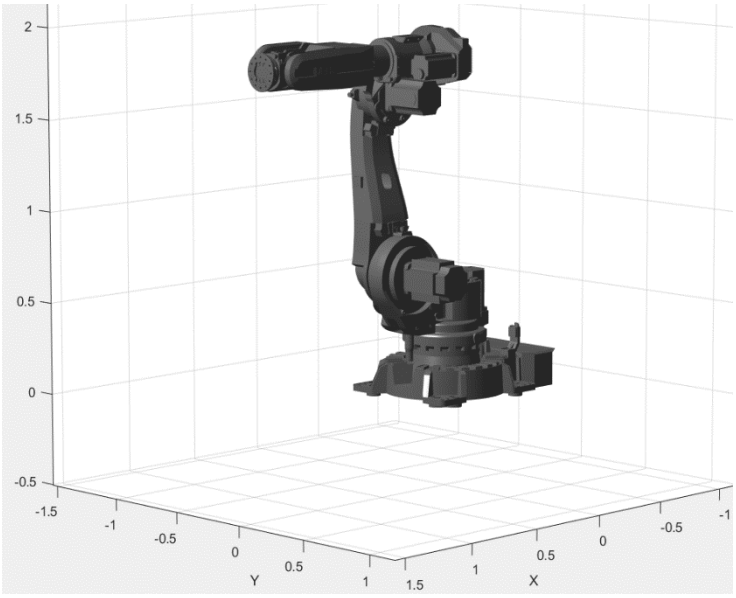
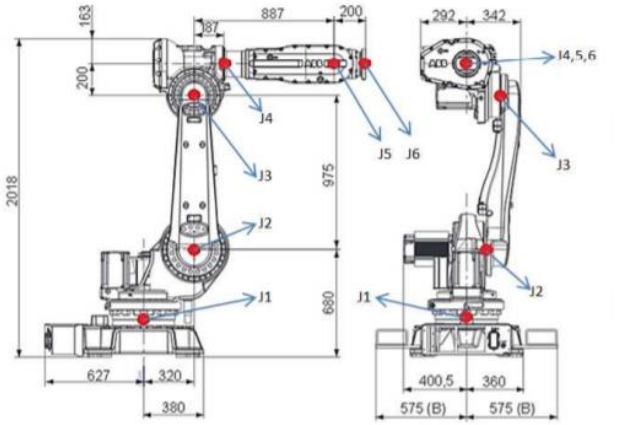


Fig.2 IRB 6620 MATLAB Model

Forward Kinematics

Forward Kinematics is computing end effector orientation and position using specified values of joint angles. In order to solve a forward kinematics problem, a methodology is followed. Firstly, local coordinate frames are attached to link joints. Then DH parameters are established for every link as shown in Table 1. For the given problem, links are defined using modified DH parameter convention.



To completely define a link 4 parameters are needed-

1. Link Length (a) – distance between Z_i and Z_{i+1} measured along X_i
2. Link Twist (α) – angle between Z_i and Z_{i+1} measured about X_i
3. Link offset (d) – distance between X_{i-1} and X_i measured along Z_i
4. Joint Offset (θ) – angle between X_{i-1} and X_i measured about Z_i

	$a\ (m)$	$\alpha\ (rad)$	$d\ (m)$	$\theta\ (rad)$
0-1	0	0	0.680	θ_1
1-2	0.320	$\pi/2$	0	θ_2
2-3	0.975	0	0	θ_3
3-4	0.200	$\pi/2$	0.887	θ_4
4-5	0	$-\pi/2$	0	θ_5
5-6	0	$\pi/2$	0	θ_6
6-7	0	0	0.200	0

Table 1. DH Parameters of IRB 6620

Now, homogenous transformation matrices are constructed to define frame (i) relative to frame (i-1). Equivalent transformation matrix is given by –

$$^{i-1}_{i}T = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)D_z(d_i)$$

Where R() and D() are the transformation matrices for rotation and translation respectively.

$$^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where $s\theta = \sin\theta$ and $c\theta = \cos\theta$

Then the kinematic equation of a serial robot containing N links is-

$$^{0}_{N}T = \prod_{i=1}^{N} {}^{i-1}_{i}T$$

The kinematic equation for IRB 6620 is given by-

$$^{0}_{7}T = {}^{0}_{1}T \times {}^{1}_{2}T \times {}^{2}_{3}T \times {}^{3}_{4}T \times {}^{4}_{5}T \times {}^{5}_{6}T \times {}^{6}_{7}T$$

$$
{}^{0}_{7}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_1 \\ 0 & 0 & -1 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\times \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & a_3 \\ 0 & 0 & -1 & -d_3 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Where,

$$r_{11} = c_6(c_5(c_1c_4c_{23} + s_1s_4) - s_5c_1s_{23}) + s_6(s_1c_4 - s_4c_1c_{23})$$

$$r_{12} = c_6(s_1c_4 - s_4c_1c_{23}) - s_6(c_5(c_1c_4c_{23} + s_1s_4) - s_5c_1s_{23})$$

$$r_{13} = c_5c_1s_{23} + s_5(c_1c_4c_{23} + s_1s_4)$$

$$r_{21} = c_6(c_5(s_1c_4c_{23} - c_1s_4) - s_5s_1s_{23}) - s_6(c_1c_4 + s_1s_4c_{23})$$

$$r_{22} = -c_6(c_1c_4 + s_1s_4c_{23}) - s_6(c_5(s_1c_4c_{23} - c_1s_4) - s_5s_1s_{23})$$

$$r_{23} = s_5(s_1c_4c_{23} - c_1s_4) + c_5(s_1s_{23})$$

$$r_{31} = c_6(c_5c_4s_{23} + s_5c_{23}) - s_6(s_4s_{23})$$

$$r_{32} = -c_6(s_4s_{23}) - s_6(c_5c_4s_{23} + s_5c_{23})$$

$$r_{33} = s_5c_4s_{23} - c_5c_{23}$$

$$p_x = r_{13}d_7 + a_1c_1 + a_2c_1c_2 + a_3c_1c_{23} + d_3c_1s_{23}$$

$$p_y = r_{23}d_7 + a_1s_1 + a_2s_1c_2 + a_3s_1c_{23} + d_3s_1s_{23}$$

$$p_z = r_{33}d_7 + d_1 + a_2s_2 + a_3s_{23} - d_3c_{23}$$

To get the homogenous transformation matrix of frame relative to another frame in MATLAB getTransform() function is used.

```
>> T = getTransform(IRB6620_mdh,[0  0 -pi/3 -pi/3 0 0],"Gripper")

T =

    0.2500    0.4330   -0.8660    0.4536
    0.8660   -0.5000   -0.0000   -0.0000
   -0.4330   -0.7500   -0.5000   -0.0367
         0         0         0    1.0000
```

Fig 3. Forward Kinematics of IRB 6620

The homogenous matrix obtained by this software numerical approach yields the same result as that of the analytical approach.

Inverse Kinematics

Inverse Kinematics analysis is finding the joint angles given a particular pose of the end effector. For IRB 6620, both analytical as well as simulation software approaches are used. The algebraic solution of IRB 6620 is presented first.

Let,

$$ {}^{0}_{7}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

To get dependence of θ_1 onto the left-hand side the following equation is used-

$$ [{}^{0}_{1}T]^{-1}\,{}^{0}_{7}T = {}^{1}_{2}T\,{}^{2}_{2}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T\,{}^{6}_{7}T \;\ldots\ldots\ldots (1) $$

Solving the above equation we get-

$$ \theta_1 = Atan2(p_y, p_x) $$

$$ \theta_3 = Atan2(a_3, d_3 + d_7) - Atan2(\pm K, \sqrt{{a_3}^2 + (d_3 + d_7)^2 - K^2}) $$

Where,

$$ K = \frac{(p_x c_1 + p_y s_1 - a_1)^2 + (p_z - d_1)^2 - {a_2}^2 - {a_3}^2 - (d_3 + d_7)^2}{2a_2} $$

Using method (1), pre multiplying the inverse of ${}^{0}_{3}T$, ${}^{0}_{4}T$ and ${}^{0}_{5}T$ to ${}^{0}_{7}T$ we can solve for $\theta_2, \theta_4, \theta_5, \theta_6$ respectively. We get the following equations from that-

$$ \theta_i = \tan^{-1}\frac{\sin\theta_i}{\cos\theta_i} \; where\; i = 2,4,5,6 $$

Additionally, there would be four more solutions obtained by flipping the wrist. The following solution is obtained.

$$ {\theta_4}^1 = \theta_4 + 180 $$

$$ {\theta_5}' = -\theta_5 $$

$$ {\theta_6}^1 = \theta_6 + 180 $$

To perform Inverse Kinematics in MATLAB two functions can be used, inverseKinematics() and generalizedInverseKinematics(). generalizedInverseKinematics() is used when there are more constraints apart from pose of the end effector. The results obtained is shown in Fig.4.

```
>> ik=inverseKinematics('RigidBodyTree',IRB6620_mdh);
>> pos=trvec2tform(via_points(:,1)');
>> [config,sol]=ik('Gripper',pos,ikweights,ikinitguess);
>> config

config =

    0.0000    0.4318    1.2217   -3.1416   -1.4836   -0.0000
```

Fig 4. Inverse Kinematics of IRB 6620

Static Velocity Calculation

Since manipulator is a chain of bodies, each one is capable of motion relative to its neighbor. Velocity of link i+1 will be addition of velocity of link i and velocity component added by joint i+1. While determining the motion of robot links, link frame 0 is taken as reference.

The angular velocity of link i+1 with respect to frame i+1 for revolute joint is –

$$ {}^{i+1}_{i+1}\omega = {}^{i+1}_{i}R\,{}^{i}_{i}\omega + \dot{\theta}_{i+1}\,{}^{i+1}_{i+1}\hat{Z} $$

The linear velocity of link i+1 with respect to frame i+1 for revolute joint is –

$$ {}^{i+1}_{i+1}v = {}^{i+1}_{i}R({}^{i}_{i}v + {}^{i}_{i}\omega \times {}^{i}_{i+1}P) $$

For simplicity in the calculation, linear and angular velocity are calculated at [0 pi/2 0 0 0 0] joint values. The angular and linear velocity of the base frame are zero.

$$ {}^{0}_{0}\omega = \begin{bmatrix}0\\0\\0\end{bmatrix} \qquad {}^{0}_{0}v = \begin{bmatrix}0\\0\\0\end{bmatrix} $$

$$ {}^{1}_{1}\omega = {}^{1}_{0}R\,{}^{0}_{0}\omega + \begin{bmatrix}0\\0\\\dot{\theta}_1\end{bmatrix} = \begin{bmatrix}0\\0\\\dot{\theta}_1\end{bmatrix} $$

$$ {}^{1}_{1}v = {}^{1}_{0}R({}^{0}_{0}v + {}^{0}_{0}\omega \times {}^{0}_{1}P) = \begin{bmatrix}0\\0\\0\end{bmatrix} $$

$$ {}^{2}_{2}\omega = \begin{bmatrix}0 & 0 & 1\\-1 & 0 & 0\\0 & -1 & 0\end{bmatrix}\begin{bmatrix}0\\0\\\dot{\theta}_1\end{bmatrix} + \begin{bmatrix}0\\0\\\dot{\theta}_2\end{bmatrix} = \begin{bmatrix}\dot{\theta}_1\\0\\\dot{\theta}_2\end{bmatrix} $$

$$ {}^{2}_{2}v = \begin{bmatrix}0 & 0 & 1\\-1 & 0 & 0\\0 & -1 & 0\end{bmatrix}\left(\begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}0\\0\\\dot{\theta}_1\end{bmatrix} \times \begin{bmatrix}a_1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\\-a_1\dot{\theta}_1\end{bmatrix} $$

Similarly,

$$ {}^{3}_{3}v = \begin{bmatrix}0\\a_2\theta_2\\-a_1\dot{\theta}_1\end{bmatrix} \qquad {}^{3}_{3}\omega = \begin{bmatrix}\dot{\theta}_1\\0\\\dot{\theta}_2+\dot{\theta}_3\end{bmatrix} $$

$$ {}^{4}_{4}v = \begin{bmatrix}d_3(\dot{\theta}_2+\dot{\theta}_3)\\-\dot{\theta}_1(d_3+a_1)\\-\dot{\theta}_2(a_3+a_2)-\dot{\theta}_3a_3\end{bmatrix} \qquad {}^{4}_{4}\omega = \begin{bmatrix}\dot{\theta}_1\\\dot{\theta}_2+\dot{\theta}_3\\\dot{\theta}_4\end{bmatrix} $$

$$ {}^{5}_{5}v = \begin{bmatrix}d_3(\dot{\theta}_2+\dot{\theta}_3)\\\dot{\theta}_2(a_3+a_2)+\dot{\theta}_3a_3\\-\dot{\theta}_1(d_3+a_1)\end{bmatrix} \qquad {}^{5}_{5}\omega = \begin{bmatrix}\dot{\theta}_1\\-\dot{\theta}_4\\\dot{\theta}_2+\dot{\theta}_3+\dot{\theta}_5\end{bmatrix} $$

$$ {}^{6}_{6}v = \begin{bmatrix}d_3(\dot{\theta}_2+\dot{\theta}_3)\\-\dot{\theta}_1(d_3+a_1)\\-\dot{\theta}_2(a_3+a_2)-\dot{\theta}_3a_3\end{bmatrix} \qquad {}^{6}_{6}\omega = \begin{bmatrix}\dot{\theta}_1\\\dot{\theta}_2+\dot{\theta}_3+\dot{\theta}_5\\\dot{\theta}_4+\dot{\theta}_6\end{bmatrix} $$

$$ {}^{7}_{7}v = \begin{bmatrix} d_3(\dot{\theta}_2 + \dot{\theta}_3) + d_7(\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_5) \\ -\dot{\theta}_1(d_3 + a_1) - d_7\dot{\theta}_1 \\ -\dot{\theta}_2(a_3 + a_2) - \dot{\theta}_3 a_3) \end{bmatrix} \qquad {}^{7}_{7}\omega = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_5 \\ \dot{\theta}_4 + \dot{\theta}_6 \end{bmatrix} $$

The Jacobian Matrix (J) maps the differential velocities between joint space and cartesian space. For an N axis manipulator, the end effector cartesian velocity if given by-

$$ V^7 = J^7(\theta)\dot{\theta} $$

Where,

$$ V^7 = \begin{bmatrix} \omega^7 \\ v^7 \end{bmatrix} $$

For IRB 6620 the Jacobian Matrix at [0 pi/2 0 0 0 0] joint values is –

$$ V^7 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & d_3 + d_7 & d_3 + d_7 & 0 & d_7 & 0 \\ -(d_7 + d_3 + a_1) & 0 & 0 & 0 & 0 & 0 \\ 0 & -(a_3 + a_2) & -a_3 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} $$

$$ J^0(\theta) = \begin{bmatrix} {}^{0}_{7}R & 0 \\ 0 & {}^{0}_{7}R \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & d_3 + d_7 & d_3 + d_7 & 0 & d_7 & 0 \\ -(d_7 + d_3 + a_1) & 0 & 0 & 0 & 0 & 0 \\ 0 & -(a_3 + a_2) & -a_3 & 0 & 0 & 0 \end{bmatrix} $$

$$ J^0(\theta) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & -1 & -1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -(a_3 + a_2) & -a_3 & 0 & 0 & 0 \\ (d_7 + d_3 + a_1) & 0 & 0 & 0 & 0 & 0 \\ 0 & d_3 + d_7 & d_3 + d_7 & 0 & d_7 & 0 \end{bmatrix} $$

The Jacobian matrix is computed with respect to the base frame. geometricJacobian() function is used to calculate the Jacobian Matrix of IRB 6620.

```
>> geometricJacobian(IRB6620_mdh,IRB6620_mdh.homeConfiguration,'Gripper')

ans =

   -0.0000         0         0    1.0000         0    1.0000
         0   -1.0000   -1.0000   -0.0000   -1.0000   -0.0000
    1.0000    0.0000    0.0000   -0.0000    0.0000   -0.0000
   -0.0000   -1.1750   -0.2000   -0.0000    0.0000   -0.0000
    1.4070    0.0000    0.0000   -0.0000    0.0000   -0.0000
         0    1.0870    1.0870   -0.0000    0.2000   -0.0000
```

Fig 5. Jacobian Matrix of IRB 6620

Note that from Fig.5 the bottom right 4 and 6 columns indicate that motion of Joint 4 and Joint 6 does not have any translational effect on the end effector.

$$\dot{\theta} = J^7(\theta)^{-1}V^7$$

The joint rates can be calculated only if the Jacobian Matrix is non-singular. Joint Values θ at which Jacobian Matrix becomes singular are known as singularities. Singularity is achieved when two robot links align in same orientation resulting in a reduction in the degree of freedom of the manipulator.

Dynamic Analysis

Dynamic analysis deals with the study of motion equations, the way in which manipulator responds when torque is applied by the actuators or external forces. Usually, there are two problems related to the dynamics of the manipulator. First one is Inverse Dynamics where given trajectory points, joint torques are calculated. Other is known as Forward Dynamics, where given joint torques, the resulting motion of the manipulator is calculated. The General equation of manipulator is given by-

$$\tau = M(\theta)\ddot{\theta} + B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] + G(\theta)$$

Where,

$M(\theta)$ is the $n \times n$ mass matrix of the manipulator.

$B(\theta)$ is a $n \times n(n-1)/2$ matrix of Coriolis coefficients.

$C(\theta)$ is a $n \times n$ matrix of centrifugal coefficients.

$G(\theta)$ is a $n \times 1$ matrix of gravity terms.

To perform inverse Dynamics in MATLAB, the mass properties, Centre of mass and Inertial Tensor have to be entered into the Robot Tree. The mass of different links is shown in Table. In MATLAB inverseDynamics() function is used to calculate joint torque given $\theta, \dot{\theta}$ and $\ddot{\theta}$. The initial configuration of robot is taken [0 0 -60 -60 0 0] and final joint values are [120 90 60 120 90 60]. The following results are obtained using MATLAB.

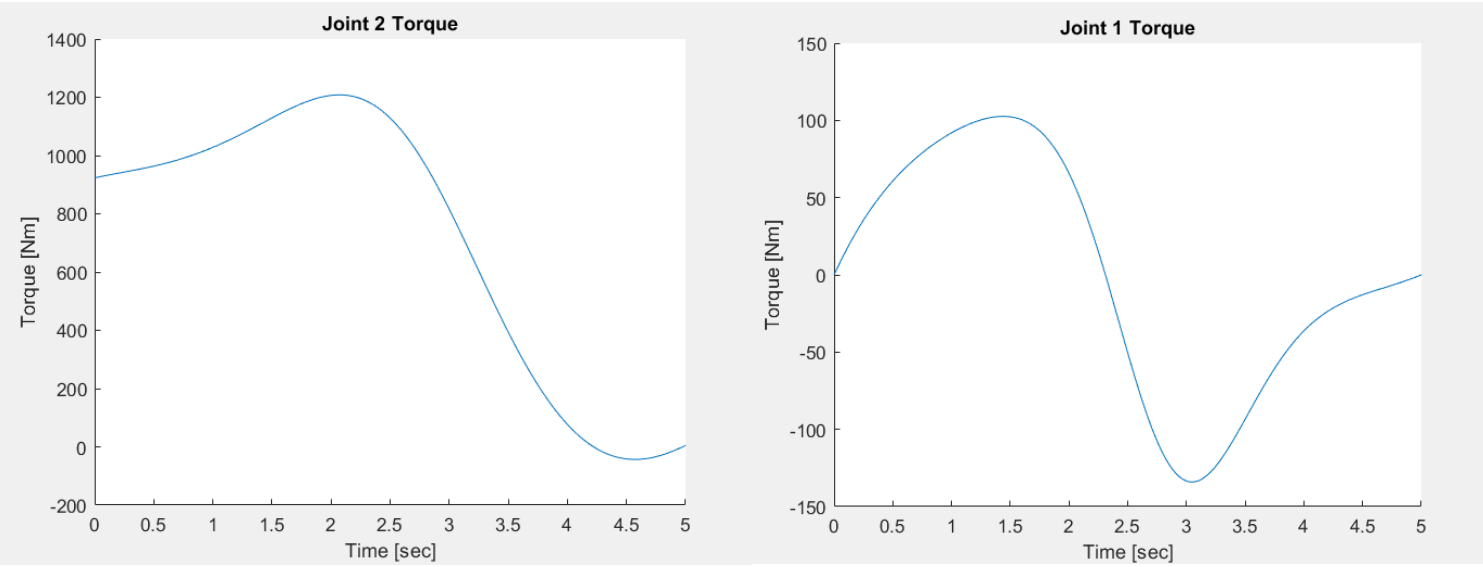


Fig 6. Torque of Joint 1 and 2

The torque is calculated for a quintic trajectory. Inverse Dynamics was also validated in RoboAnalyzer. RoboAnalyzer is an open source, 3D model-based software. To create IRB 6620, DH parameters are entered into the software. Changes were made in DH parameters according to the documentation of RoboAnalyzer. The results plotted are shown in Fig.7 Joint torque calculated by MATLAB yield the same result to that of RoboAnalyzer.

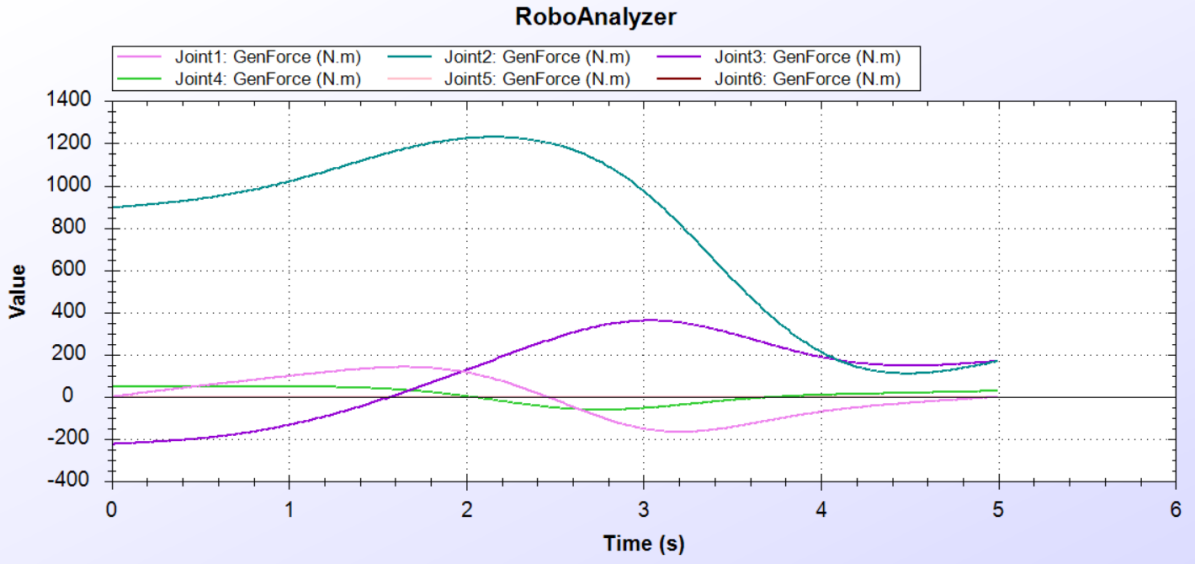


Fig 7. Joint Torques using RoboAnalyzer

Conclusion

In the above project detailed analytical approach validated with a numerical software approach of IRB 6620 is presented. Complete graphical data is provided to ease the transparency of the process and improve the analysis readability. For numerical validation, MATLAB and RoboAnalyzer software are used. Hence a cross-domain approach to Kinematic and dynamic modelling and analysis of a robot is investigated with supporting graphical and numerical results.