Inverse Dynamics Based Artificial Neural Network BFGS quasi-Newton backpropagation of 3 DOF Robotic Arm (IRB4400)

Saksorn Ruangtanusak

King Mongkut's University of Technology Thonburi
126 Pracha Uthit Rd, Bang Mot, Thung Khru, Bangkok 10140

ABSTRACT

In the field of robotic control, artificial intelligence has become the most investigated modern technology. To implement artificial intelligence for robots correctly, it's needed to have a trustable model of kinematics and dynamics. In this research, a 3-DOF sample of the industrial manipulator based on the ABB IRB 4400 model has been studied. Robot kinematics have been solved. For forward kinematics, the method uses the DH-parameter, and for inverse kinematics, it uses the geometric approach to get the end effector's final position. including with path-planning and path smoothing. For robot dynamics, the forward dynamics for controlling robot arm joints using feedforward control with a PI controller. while the inverse dynamic equation is derived by using Lagrange-Euler. Artificial neural networks (ANNs) solve the joint torque in a continuous path efficiently, overcoming the drawbacks of inverse dynamics. The innovative controller architecture outperforms prior strategies in terms of lowering position error in unusual situations and improving ANN accuracy in estimating robot joint angles. According to additional research, the use of artificial neural networks to inverse dynamics might considerably improve the performance of a 6-DOF robotics arm or a hyper-redundant manipulator in the future.

1. INTRODUCTION

Robotic control is now dominated by artificial intelligence. Additionally, it has several advantages in terms of performance, including precise control and minimal computation time. Kinematic and dynamic modeling are critical for robotic manipulator design, simulation, and control.

Kinematics describes the motion of the robotic manipulator without considering forces and torques causing this motion. The fundamental issue with robotic arm control is determining an accurate and reliable inverse dynamics solution. It establishes the relationship between the forces exerted on robotic manipulators and the acceleration generated by the manipulators. Dynamic equations are essential for robot motion modeling and control algorithm design. For dynamic analysis of robot manipulators, the Lagrange-Euler and Newton-Euler formulations are well-known.

The Lagrange-Euler approach solves these issues. Forces are stated in terms of kinetic and potential energy, which are scalar quantities that may be easily defined in terms of system coordinates, in the Lagrange-Euler technique. In real-time robot arm control, inverse dynamics calculations are required; however, solving inverse dynamics is computationally complicated and requires a long time to calculate.

This paper introduces a novel solving method for three-axis RRP manipulator robot based on ANN to be used in dynamic torque control. The training algorithm are BFGS quasi-Newton backpropagation. The dataset use in this paper are determine from inverse dynamics and forward dynamics. The ANN architecture is 4 hidden layers with tan-sigmoid activation function. In terms of performance, the inversed dynamics dataset outperforms the other in terms of accuracy and computation time. The paper is organized as follows. Kinematics, Dynamics, Implementation with ANNs, discussion of results, and concluding.

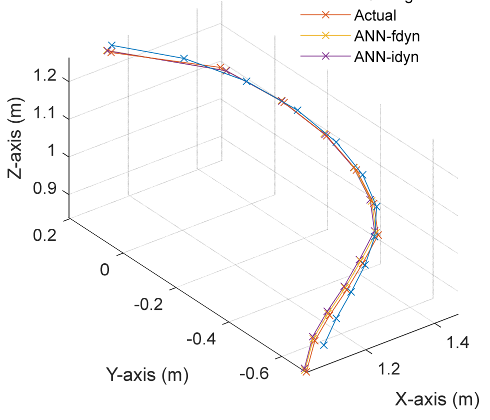


Figure 1: Example of path-planning

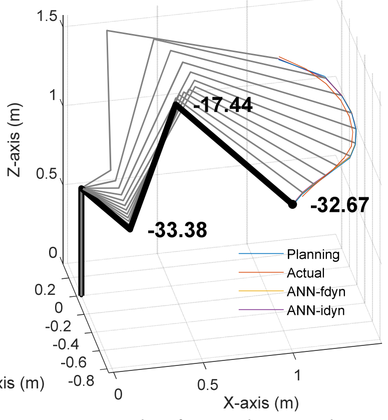


Figure 2: Example of 3D-robot simulation

2. KINEMATICS

2.1 Forward Kinematics

The Forward Kinematic (FK) matrices of the 6-DOF RRP type serial robot manipulator shown in Figure 2 are obtained in this section using the Denavit-Hartenberg (DH) method. [1], [2]. Transformation matrices is obtained based on the DH parameters shown in Table I.

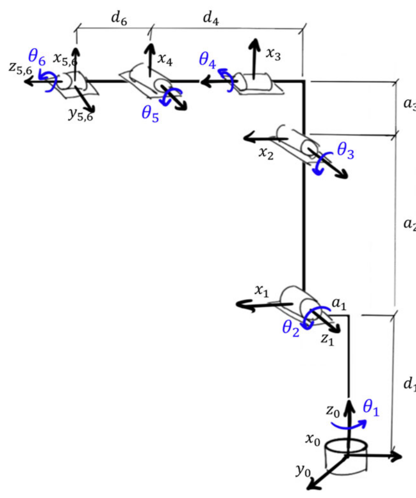


Figure 3: The RRP Type Robot of IRB-4400 and Co-ordinate Frames

Create the single link homogenous transformation matrix A$_i$

$$A_i = \mathrm{Rot}_{z,\theta i}\,\mathrm{Trans}_{x,di}\,\mathrm{Trans}_{x,\alpha i}\,\mathrm{Rot}_{x,\alpha i}$$

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where i is link number, $S\theta_i = \sin\theta_i, C\theta_i = \cos\theta_i$, θ_i is the joint rotation angle, a_i is the length of links, α_i is the twist angles, d_i is the link offsets, and θ is the joint angles. However, in this work, the robot arm has been simplified to a 3-DOF RRP Type. As a result, $\theta_1, \theta_2, \theta_3$ angles are zero.

Table I: DH-Parameter of IRB4400

Link	a_i	α_i	d_i	θ_i
1	a_1	$-90°$	d_1	θ_1^*
2	a_2	0	0	$\theta_2^* - 90°$
3	a_3	$-90°$	0	θ_3^*
4	0	$+90°$	d_4	θ_4^*
5	0	$-90°$	0	θ_5^*
6	0	0	d_6	θ_6^*

The system has six links homogeneous transformation matrix is calculated by multiplication of matrices as follows:

$$T_6^0 = A_1 A_2 A_3 A_4 A_5 A_6$$

2.2 Inverse Kinematics

Inverse Kinematics (IK) analysis is used to calculate the joint angles necessary to achieve the required position and orientation in Cartesian space. Inverse kinematics is more complex to solve than forward kinematics, and there is no global analytical solution method available. Each manipulator needs a particular method considering the system structure and restrictions. In this present work geometric approaches are selected.

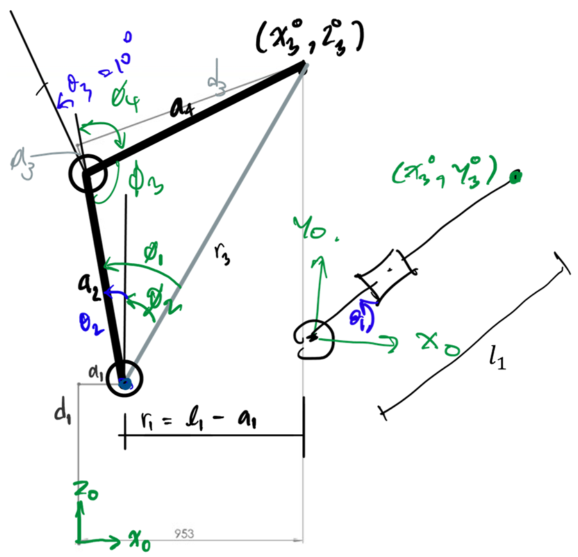


Figure 4: Inversed Kinematics drawing (Side view and Top view)

$$l_1 = \sqrt{(x_3^0)^2 + (y_3^0)^2}$$

$$\boldsymbol{\theta_1 = q_1 = atan2(y_3^0, x_3^0)}$$

$$\phi_1 = \cos^{-1}\left(\frac{a_3^2 + d_3^2 - a_2^2 - r_3^2}{-2a_2 r_3}\right)$$

$$\phi_2 = atan2(r_1, r_2)$$

From Geometric Side View

$$\theta_2 = \phi_1 - \phi_2$$

$$\boldsymbol{\theta_2 = q_2 = cos^{-1}\left(\frac{a_3^2 + d_3^2 - a_2^2 - r_3^2}{-2a_2 r_3}\right) - atan2(r_1, r_2)}$$

From Cosine Rule

$$\phi_3 = cos^{-1}\left(\frac{r_3^2 - r_4^2 - a_2^2}{-2r_4 a_2}\right)$$

From Geometric Side View

$$\phi_4 + \theta_3 = atan2(a_3, d_3) \qquad (1)$$

$$\phi_4 = \pi - \phi_3 \qquad (2)$$

Sub (2) *in* (1)

$$\theta_3 = \phi_3 - \pi + atan2(a_3, d_3)$$

$$\boldsymbol{\theta_3 = q_3 = cos^{-1}\left(\frac{r_3^2 - r_4^2 - a_2^2}{-2r_4 a_2}\right) - \pi + atan2(a_3, d_3)}$$

2.3 Trajectory Planning and Input Windows

This robot has a specify duty to draw any shape in a cartesian coordinate system. The input GUI are shown in Figure 4, which can point a coordinate in X,Y simultaneously.

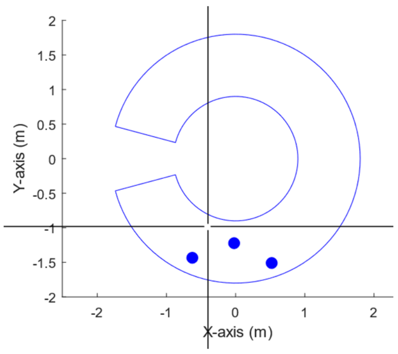


Figure 5: Path input windows (XY coordinate)

After getting a point coordinate, the robot needs a small increment of each angle. Thus, the program divides the input point into a smaller path. With a condition if-else. After getting a new finer path as shown in Figure 5.

```
if        R < 0.5; b=1
elseif    R < 1;   b=4
else               b=10
```

Where the implement is

$$R = \sqrt{X^2 + Y^2}\,;\ Interpolate\ Size = \frac{b}{R}$$

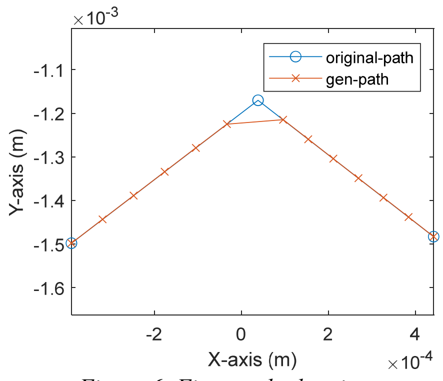


Figure 6: Finer path planning

2.4 Jacobian Matrix

The Jacobian matrix can be used to calculate the velocity relationship. The Jacobian matrix is a vector representation of the ordinary derivative of a scalar function. The Jacobian is a critical quantity when analyzing and controlling robot motion. [3] First, define linear velocity and angular velocity.

$$\begin{bmatrix} v_n^0(t) \\ \omega_n^0(t) \end{bmatrix} = \begin{bmatrix} J_v(q(t)) \\ J_\omega(q(t)) \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

$$= \begin{bmatrix} J_{v_1}(q(t)) & J_{v_2}(q(t)) & J_{v_3}(q(t)) \\ J_{\omega_1}(q(t)) & J_{\omega_2}(q(t)) & J_{\omega_3}(q(t)) \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

$$= \begin{bmatrix} z_1 \times (P_3^0 - P_0^0) & z_2 \times (P_3^0 - P_1^0) & z_3 \times (P_3^0 - P_2^0) \\ z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

Where as

$$z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix};\quad z_2 = z_3 = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix};\quad \psi = sin\left(\frac{d_3}{a_3}\right)$$

$$P_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix};\quad P_1 = \begin{bmatrix} a_1 c_1 \\ a_1 s_1 \\ d_1 \end{bmatrix};\quad P_2 = \begin{bmatrix} a_1 c_1 - a_2 s_2 c_1 \\ a_1 s_1 - a_2 s_2 c_1 \\ d_1 + a_2 c_2 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} a_1 c_1 + d_3 s_1 + a_2 c_1 c_2 + a_3 c_3 c_2 c_1 - a_3 s_3 s_2 c_1 \\ a_1 s_1 - d_3 c_1 + a_2 c_2 s_1 + a_3 c_3 c_2 s_1 - a_3 s_3 s_2 s_1 \\ d_1 + a_2 s_2 + a_3 c_2 s_3 + a_3 c_3 s_2 \end{bmatrix}$$

Thus, the Jacobian matrix is

$$J =$$

$$\begin{bmatrix} -s_1(a_1 - a_2 s_2 + r_4 c_{23\psi}) & -c_1(a_2 c_2 + r_4 s_{23\psi}) & -r_4 s_{23\psi} c_1 \\ c_1(a_1 - a_2 s_2 + r_4 c_{23\psi}) & -s_1(a_2 c_2 + r_4 s_{23\psi}) & -r_4 s_{23\psi} s_1 \\ 0 & -(c_1^2 + s_1^2)(a_2 s_2 - r_4 c_{23\psi}) & r_4 c_{23\psi}(c_1^2 + s_1^2) \\ 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix}$$

3. DYNAMICS

3.1 Inverse Dynamics (Lagrange)

The dynamical equations were symbolically derived with Lagrange Euler method [3]. From Jacobian matrix we know J_v matrix. Thus, we can create another J_{vc} matrix at centroid of each link as shown in .

$$J_{vc1} = [z_1 \times (p_{1c} - p_0); 0\,; 0]$$

$$J_{vc2} = [z_1 \times (p_{2c} - p_0);\ z_2 \times (p_{2c} - p_1)\,; 0]$$

$$J_{vc3} = [z_1 \times (p_{3c} - p_0); z_2 \times (p_{3c} - p_1)\,; z_3 \times (p_{3c} - p_2)]$$

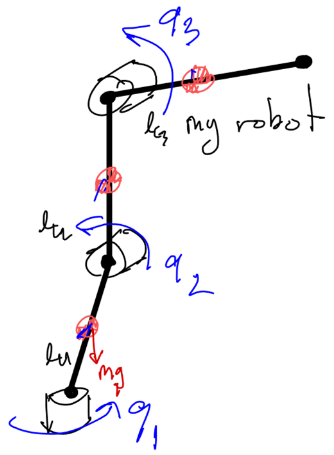


Figure 7: IRB4400 Simplified 3-link for Lagrange calculation

Determine Kinetics Energy in a translational part.

$$\frac{1}{2}mv_c^T v_c = \frac{1}{2}\dot{q}\{m_1 J_{vc1}^T J_{vc1} + m_2 J_{vc2}^T J_{vc2} + m_3 J_{vc3}^T J_{vc3}\}\dot{q}$$

Determine Kinetics Energy in a rotational part.

$$\frac{1}{2}\dot{q}^T\left\{I_1\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} + I_1\begin{bmatrix}1&1&0\\1&1&0\\0&0&0\end{bmatrix} + I_1\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\right\}\dot{q}$$

Determine potential energy of the robot

$$P = m_1 g h_1 + m_2 g h_2 + m_3 g h_3$$

Where as

$$h_1 = \frac{d_1}{2};\ h_2 = d_1 + \frac{a_2 s_2}{2};\ h_3 = d_1 + a_2 s_2 + r_4 s_{23\phi}$$

After having the parameters h, the potential energy of the robot can be calculated. The Lagrange function is given by

$$L = K - P = K_{trans} + K_{rot} - P$$

Thus, the dynamics equation as follow.

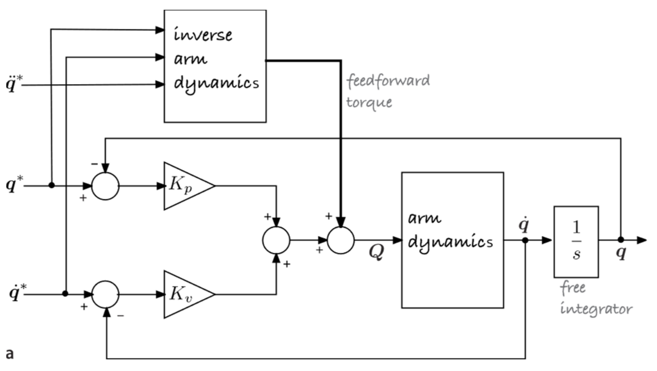
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i (i = 1,2,\dots,n)$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_1} - \frac{\partial L}{\partial q_1} = \tau_1$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_2} - \frac{\partial L}{\partial q_2} = \tau_2$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_3} - \frac{\partial L}{\partial q_3} = \tau_3$$

3.2 Forward Dynamics (Joint Control)



$$Q^* = \underbrace{M(q^*)\ddot{q}^* + C(q^*,\dot{q}^*)\dot{q}^* + F(\dot{q}^*) + G(q^*)}_{\text{feedforward}} + \underbrace{\left\{K_v(\dot{q}^* - \dot{q}) + K_p(q^* - q)\right\}}_{\text{feedback}}$$
$$= \mathcal{D}(q^*,\dot{q}^*,\ddot{q}^*) + K_v(\dot{q}^* - \dot{q}) + K_p(q^* - q)$$

Figure 8: Manipulator feedforward control structures [4]

Forward dynamics or joint control systems that calculate joint forces in order to maintain the desired trajectory of the robot end-effector despite variable dynamic characteristics or joint flexibility. From the equation as shown in Figure 7 can be simplified to

$$Q^* = D(\ddot{q}) + K_v(\dot{q}^* - \dot{q}) + K_p(q^* - q)$$

From motor mechanical drive, motor inertia (J) and motor electrical drive unit, armature resistance (R) or electrical gain are added to the control equation.

$$\tau R = \ddot{q}JR + K_v(\dot{q}^* - \dot{q}) + K_p(q^* - q)$$

Thus, the forward dynamics equation as follow.

$$\ddot{q}(\dot{q}, q) = \frac{K_v(\dot{q}^* - \dot{q})}{R \times J} + \frac{K_p(q^* - q)}{R \times J} - \frac{\tau}{J}$$

Other motor control parameter as shown below. The derivative gain, proportional gain

$$K_v = [1, 3, 3];\ K_p = [1, 200, 100]$$

$$J = 0.01\ kg \cdot m^2;\ R = 0.003\ \Omega$$

4. IMPLIMENTATION

4.1 Artificial Neural Network Approaches

Extensive studies presently employ a standard ANN architecture, as illustrated in Figure 8. To take use of the suggested method's advantages, this work employs standard ANN to solve the inverse dynamics. The training algorithm are BFGS quasi-Newton backpropagation. In this ANN, the elements in the input layer are three variables, which are the joint angle, q_i. and the output layer are three variables, which are the joint torque τ_i. In this present work, there are 2 dataset forward dynamics and inverse dynamics. For the forward dynamic dataset, it's needed to swap input and output.

$$\tau = ANN(q)$$

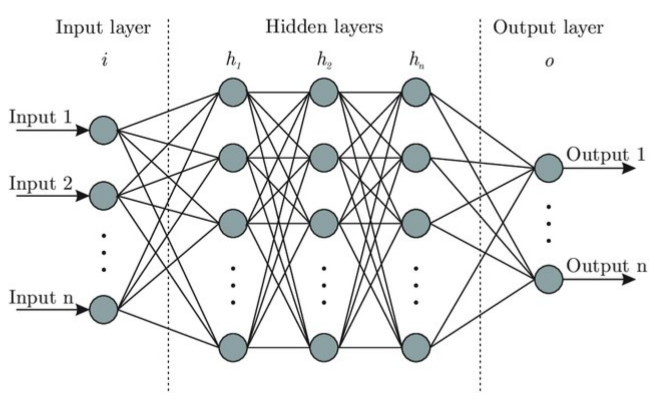


Figure 9: Artificial neural network architecture (ANN) [5]

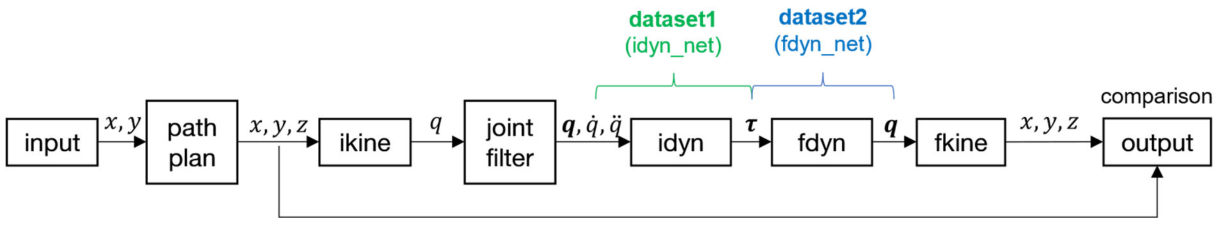


Figure 10: Program Flowchart

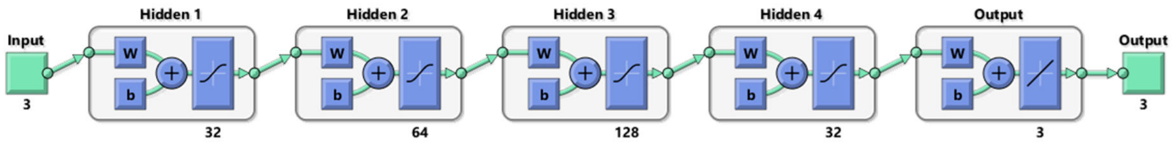


Figure 11: ANN architecture

4.2 Simulation Setup and ANN Training

From all the robot kinematic and robot dynamics can be combine as shown in Figure 9. First get the input in x, y coordinate, then use path planning algorithm to divides the input point into a smaller path. Inverse kinematics is used to create joints angle, q. Then, use path-filtering [6], [7] to smooth the joint angle as shown in Figure 11.

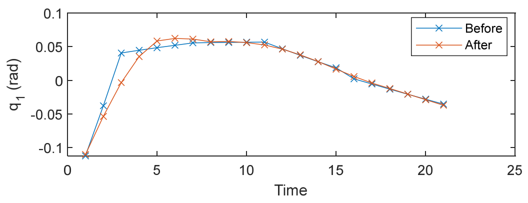


Figure 12: Path-filtering in path plan at joint 1

Inverse dynamics is used to create dataset1 and forward dynamics is used to create dataset2. Then, use forward kinematics to create cartesian space for comparing the results with the input from path plan.

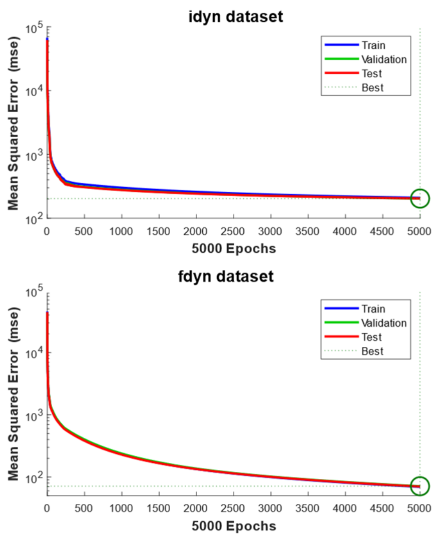


Figure 13: Performance of ANN traning

There are 50 image coordinates in the dataset, and each variable has a total of 8595 values. Divide the dataset into 7000 train instances and 1595 test instances, as shown in the diagram. The epoch for both datasets is set to 5000, while the aim is set to mean square error (MSE) = 0.

The ANN architecture is 4 hidden layers with tan-sigmoid activation function as shown in Figure 10. The hidden layers size is followed 32, 64, 128, 32. The performance of ANN training is shown in Figure 12.

5. RESULTS AND DISCUSSION

After the training of inverse dynamics dataset and forward dynamics dataset with ANN, the ANN model is generated into 2 function ANN-fdyn and ANN-idyn.

In this simulation, the robot rotates sinusoidally in the x and y axes while moving linearly in the z axis. To illustrate the helical motion route created by an ANN. ANN-fdyn and ANN-idyn are used to solve the inverse dynamics problem for robot motion. Figure 13 depicts the end effector's path configuration because of the robot arm movement.

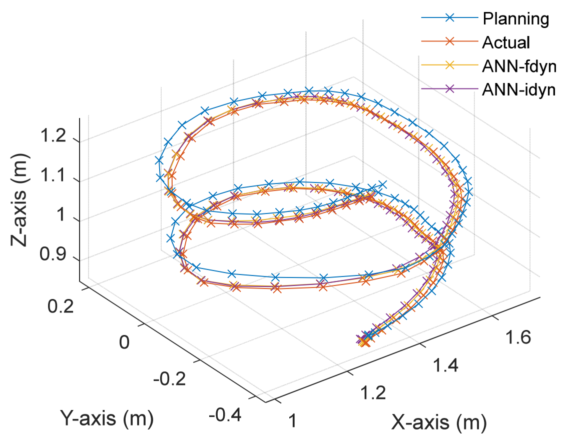


Figure 14: 3D Path compare

The motion of the robot arm in a Cartesian space depends on each joint angle. In Figure 14 the motion of robot joint angle is an enlargement of the timesteps period (0-10).

Figure 14 shows each joint angle result from actual (Inverse Dynamics) compared with both ANN methods.

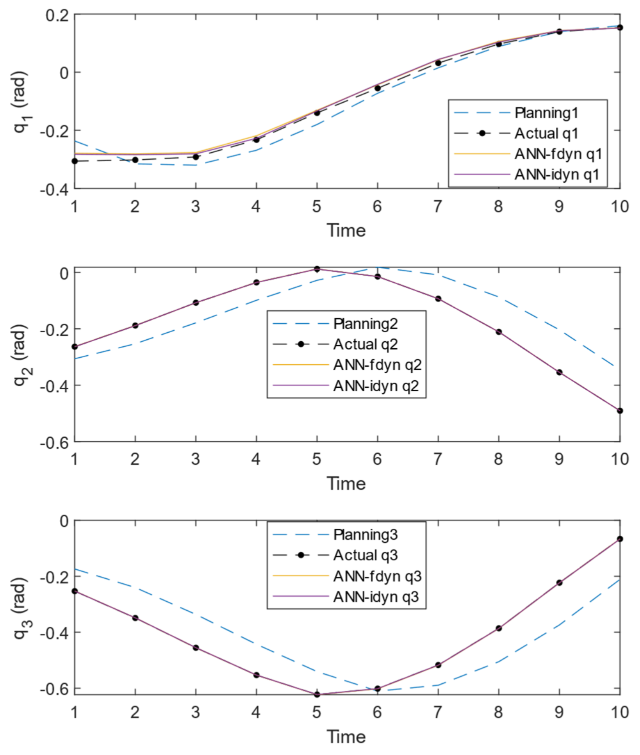


Figure 15: Robot joint angle ($q_1, q_2, and\ q_3$) with comparison with ANN-fdyn and ANN-idyn

The motion curves of ANN-idyn are more precise than the motion curves of ANN-fdyn in estimating the joint torque of the robot for desired positions and then calculating the position of the end-effector. The maximum error percentages are shown in Table II. The calculation time of traditional inverse dynamics is 0.243 s. On the other hand, both ANNs perform faster than traditional inverse dynamics. The time reduction percentage of ANN-fdyn is 45.21% and for ANN-idyn it is 237.5%.

Table II: Performance of robotic system by ANNs

Parameter	ANN-fdyn	ANN-idyn
P_x max error (%)	0.110	0.029
P_y max error (%)	2.827	1.075
P_z max error (%)	1.841	1.193
MSE	203.182	71.015
Time (s)	0.133	0.072

The errors are increased in some locations and decreased in others; the point with the highest error is the one where the ANN is predicting the joint torque, while the point with the lowest error is the one that is close to the training set samples in the ANN. As a result, this technology can be implemented in the exact movements of a robotic arm.

6. SUMMARY AND CONCLUSIONS

This study introduced an accurate and faster inverse dynamic based artificial neural network of 3-DOF RRP type serial rigid robot manipulator. In this paper, have separate into 3 main topics: Kinematics, Dynamics, and Implementations with ANNs.

Firstly, robot kinematics. The kinematics of industrial manipulator prototype was analyzed. The method using DH-parameter for forward kinematics and using geometric approach for inverse kinematics. Furthermore, the robot joint angle path is planning by divides the input path. With the help of path-filtering algorithm prevent non-smooth dynamic of robot.

Secondly, robot dynamic. The inverse dynamic equation is derived by using Lagrange-Euler method on MATLAB software. While the forward dynamics for control robot arm joint are using feedforward control [4]. With having 2 gains for proportional and derivative.

Thirdly, to implement kinematics and dynamics with ANNs to develop kinematics and robot dynamics in MATLAB for simulation.

In order to address the disadvantages of inverse dynamics controllers, artificial neural networks (ANNs) can solve the joint torque more quickly in a continuous route. In terms of training dataset selection, the inversed dynamics dataset outperforms the other in terms of accuracy and computation time (237.5 percent faster than conventional inverse dynamics).

For further studies, the use of artificial neural networks to inverse dynamics might significantly increase the performance of a 6-DOF robotics arm or a hyper redundant manipulator in the future.

REFERENCE

[1] Denavit J, Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices.," *Trans ASME J Appl Mech*, vol. 23, pp. 215-221., 1955.

[2] J. J. Craig, "Mechanics and Control Third Edition," p. 408.

[3] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar, *Robot Modeling and Control, 2nd Edition | Wiley*.

[4] P. Corke, *Robotics, Vision and Control*, vol. 73. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-20144-8.

[5] F. Bre, J. M. Gimenez, and V. D. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy Build.*, vol. 158, pp. 1429–1441, Jan. 2018, doi: 10.1016/j.enbuild.2017.11.045.

[6] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics Modelling, Planning and Control*. London: Springer London, 2009. doi: 10.1007/978-1-84628-642-1.

[7] Abraham. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures.," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964, doi: 10.1021/ac60214a047.

APPENDIX

Robot Properties

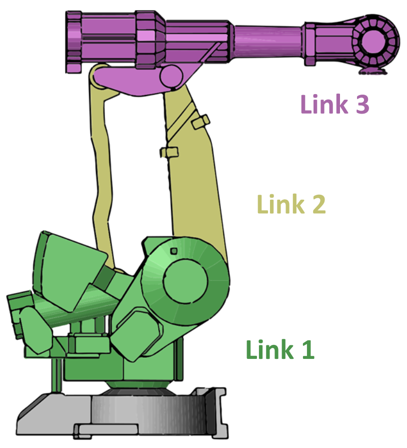


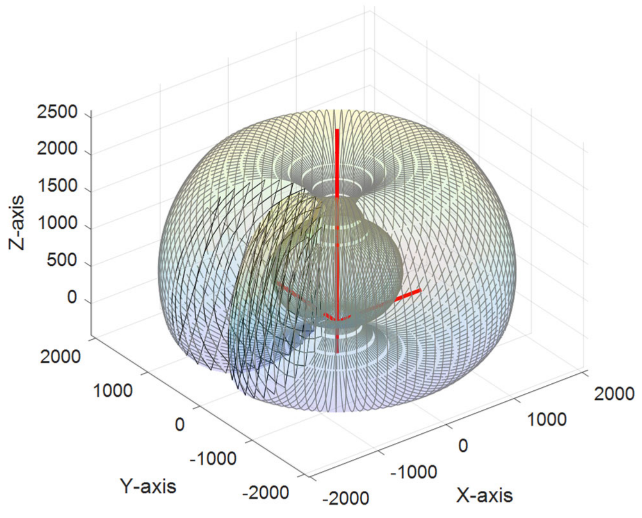
Figure 16: IRB4400 M94A Simplifired 3-DOF Link

The robot IRB4400 M94A Simplified 3-DOF properties are list in the Table II. The properties are determined from Solidworks 3D CAD.

Table III: IRB4400 M94A properties

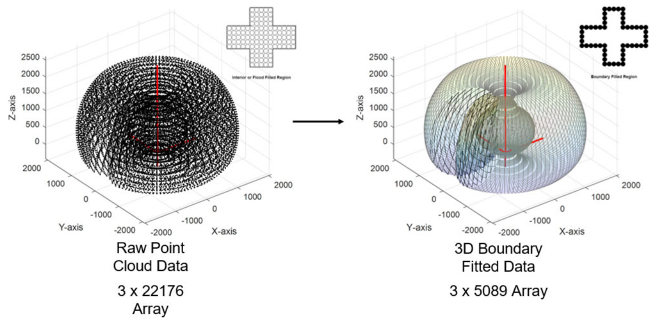
Link	Mass (kg)	Ixx (kg*m^2)	Iyy (kg*m^2)	Izz (kg*m^2)
1	131.2255	12.62	12.48	10.46
2	34.1066	2.696	3.106	0.746
3	48.2736	9.107	9.048	0.478

Robot Workspace



Boundary Filled Algorithm

4X Data point reduction


Raw Point Cloud Data
3 x 22176 Array
3D Boundary Fitted Data
3 x 5089 Array