A neural network based adaptive controller for end effector tracking of redundant manipulators

Muhammad Idrees
Faculty of Electrical and Computer Engineering
Univeristy of Ontariotech
Email: muhammad.idrees@ontariotechu.net

Elena Villaobos Herra
Faculty of Electrical and Computer Engineering
Univeristy of Ontariotech
Email: elena.villalobosherra@ontariotechu.net

Rakhshanda Rukham
Faculty of Electrical and Computer Engineering
Univeristy of Ontariotech
Email: rakhshanda.rukham@ontariortechu.net

***Abstract*— This document explores the neural network based adaptive controller for redundant manipulators developed in [1], aiming to implement it for the trajectory tracking of an end-effector in the x, y z cartesian space. The control scheme is based on estimating the robot dynamic matrices with neural networks and changing the neural network parameters using adaptive control theory. The robot model is a 4 degrees of freedom redundant manipulator and it was implemented in MATLAB Simulink. After implementing the robot model and starting to implement the controller, a dimension mismatch presented a significant obstacle in the implementation of the adaptation laws for the neural network parameters. Due to this obstacle, the adaptation laws were not fully implemented. The results obtained show the end-effector following a trajectory similar in shape to the desired trajectory, but without tracking it. It is concluded that further analysis needs to be performed with the stability analysis that the authors did in order to resolve the mismatch in the dimensions.**

Keywords*— *End-effector, adaptive control, neural network

I. Introduction

In recent years, robotics technology has been expanding rapidly, and it has become a crucial part of several industries. However, the dynamic model of robots is challenging to obtain due to inaccuracies resulting from changing the end effector's tools or increasing and decreasing the loadings. There can be unmodeled disturbances as well. Therefore, adaptive estimation comes as a useful tool to estimate the dynamics of the manipulator accurately. Moreover, unmodelled disturbances can further contribute to tracking errors. Our project endeavors to tackle these challenges by creating a neural network-focused adaptive controller designed to track end-effector movements of superfluous manipulators [1].

To effectively gauge the manipulator's dynamics - specifically, the M, C, and H matrices that are not easy to model flawlessly, we used an adaptive estimation technique. Given the intricacies of the dynamic model of the manipulator, accurately modeling it can be a challenging feat, especially when considering the changes that can occur due to factors such as modifications to the tool at the end effector, varying loads, or unpredictable interferences [2]. Thus, adaptive estimation becomes a great tool to assess the manipulator's dynamics. Here we proposed a neural network-based adaptive controller for end-effector tracking of redundant manipulators developed in [1]. The project utilizes a 4 DOF redundant manipulator with x, y, z control. The first joint is a prismatic one for extended reach while the rest three are revolute. To model the robot for trajectory tracking control, it is necessary to model the position and velocity kinematics, followed by the torque model.

In this project, the DH (Denavit Hardenberg) schematic model and DH table of the PRRR manipulator are shown, and the Jacobian and Hessian are implemented in MATLAB. This project aims to contribute to improving the precision and accuracy of redundant manipulators. This can have various applications in various industries, leading to more efficient and cost-effective operations. Additionally, it can inspire further studies and innovations in the field of robotics, which can ultimately lead to better automation and technology.

II. Robot Model

The robot used for trajectory tracking is a 4 DOF redundant manipulator with x, y, z control. The robot has a first prismatic joint for extended reach, and the three other joints are revolute. To model the robot for trajectory tracking control, it is necessary to model the position and velocity kinematics followed by the torque model.

A. Forward Kinematics model

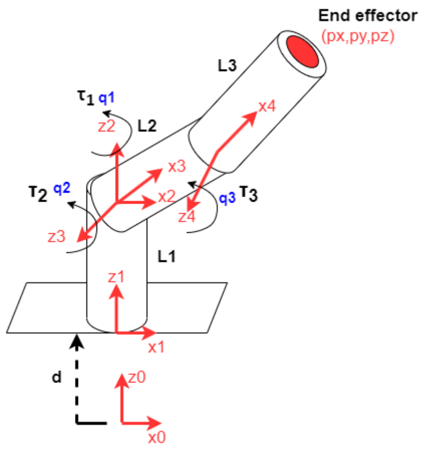

Fig. 1. *DH Schematic of PRRR manipulator*

The DH (Denavit Hardenberg) schematic model and DH table of the PRRR manipulator are shown in Figure 1 and Table I respectively.

TABLE I. DH TABLE

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d	0
2	0	0	L_1	q_1
3	90^o	0	0	q_2
4	0	L_2	0	q_3
5	0	L_3	0	0

The transformations matrices between link frames are:

$${}^0_1T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^1_2T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^2_3T = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^3_4T = \begin{bmatrix} c_3 & -s_3 & 0 & L_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^4_5T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using relative transformations, the transformation matrix between frame {0} and frame {5} can be established as:

$${}^0_5T = {}^0_1T\ {}^1_2T\ {}^2_3T\ {}^3_4T\ {}^4_5T$$

$${}^0_5T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & c_1(L_2c_2 + L_3c_{23}) \\ r_{21} & r_{22} & r_{23} & s_1(L_2c_2 + L_3c_{23}) \\ r_{31} & r_{32} & r_{33} & L_1 + d + L_2s_2 + L_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The end effector position in frame {0} can thus be obtained from the last column of ${}^0_1T$ matrix:

$$p_x = c_1(L_2c_2 + L_3c_{23})$$
$$p_y = s_1(L_2c_2 + L_3c_{23})$$
$$p_z = L_1 + d + L_2s_2 + L_3s_{23}$$

B. *Velocity analysis and Jacobian*

The end effector velocity can be equated to the joint space velocity using the Jacobian relationship. This is given by:

$$\dot{x} = J\dot{q}$$

The Jacobian matrix for the PRRR configuration can be obtained by finding the derivative of the end effector positions as:

$$J = \frac{\delta x}{\delta q} = \begin{bmatrix} \frac{\delta x}{\delta d} & \frac{\delta x}{\delta q_1} & \frac{\delta x}{\delta q_2} & \frac{\delta x}{\delta q_3} \\ \frac{\delta y}{\delta d} & \frac{\delta y}{\delta q_1} & \frac{\delta y}{\delta q_2} & \frac{\delta y}{\delta q_3} \\ \frac{\delta z}{\delta d} & \frac{\delta z}{\delta q_1} & \frac{\delta z}{\delta q_2} & \frac{\delta z}{\delta q_3} \end{bmatrix}$$

$$J = \begin{bmatrix} 0 & -s_1(L_2c_2 + L_3c_{23}) & -c_1(L_2s_2 + L_3s_{23}) & -c_1L_3s_{23} \\ 0 & c_1(L_2c_2 + L_3c_{23}) & -s_1(L_2s_2 + L_3s_{23}) & -s_1L_3s_{23} \\ 1 & 0 & L_2c_2 + L_3c_{23} & L_3c_{23} \end{bmatrix}$$

C. *Acceleration analysis and Hessian*

Hessian matrix relates the task space acceleration vector to joint space acceleration vector given by:

$$\ddot{x} = J\ddot{q} + H\dot{q}$$

The Hessian matrix is the time derivative of the Jacobian matrix. For the PRRR configuration, it can be defined as:

$$\dot{J} = H = \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \end{bmatrix}$$

The hessian matrix terms are:

$$H_{11} = 0$$

$$H_{12} = -\mathrm{s}_1\big(-\mathrm{L}_2\mathrm{s}_2\dot{\mathrm{q}}_2 - \mathrm{L}_3\mathrm{s}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)\big) - \mathrm{c}_1\dot{\mathrm{q}}_1(\mathrm{L}_2\mathrm{c}_2 + \mathrm{L}_3\mathrm{c}_{23})$$

$$H_{13} = \mathrm{c}_1\big(-\mathrm{L}_2\mathrm{c}_2\dot{\mathrm{q}}_2 - \mathrm{L}_3\mathrm{c}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)\big) - \mathrm{s}_1\dot{\mathrm{q}}_1(-\mathrm{L}_2\mathrm{s}_2 - \mathrm{L}_3\mathrm{s}_{23})$$

$$H_{14} = \mathrm{c}_1\big(-\mathrm{L}_3\mathrm{c}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)\big) - \mathrm{s}_1\dot{\mathrm{q}}_1(-\mathrm{L}_3\mathrm{s}_{23})$$

$$H_{21} = 0$$

$$H_{22} = \mathrm{c1}\big(-\mathrm{L}_2\mathrm{s}_2\dot{\mathrm{q}}_2 - \mathrm{L}_3\mathrm{s}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)\big) - \mathrm{s}_1\dot{\mathrm{q}}_1(\mathrm{L}_2\mathrm{c}_2 + \mathrm{L}_3\mathrm{c}_{23})$$

$$H_{23} = \mathrm{s}_1\big(-\mathrm{L}_2\mathrm{c}_2\dot{\mathrm{q}}_2 - \mathrm{L}_3\mathrm{c}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)\big) + \mathrm{c}_1\dot{\mathrm{q}}_1(-\mathrm{L}_2\mathrm{s}_2 - \mathrm{L}_3\mathrm{s}_{23})$$

$$H_{24} = \mathrm{s}_1\big(-\mathrm{L}_3\mathrm{c}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)\big) + \mathrm{c}_1\dot{\mathrm{q}}_1(-\mathrm{L}_3\mathrm{s}_{23})$$

$$H_{31} = 0$$

$$H_{32} = 0$$

$$H_{33} = -\mathrm{L}_2\mathrm{s}_2\dot{\mathrm{q}}_2 - \mathrm{L}_3\mathrm{s}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)$$

$$H_{34} = -\mathrm{L}_3 \mathrm{s}_{23}(\dot{\mathrm{q}}_2 + \dot{\mathrm{q}}_3)$$

D. Dynamic model

The dynamics model of a manipulator in closed form can be written in joint space as:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + H(q,\dot{q})$$

Where, $M(q)$ is the inertia matrix dependent on the joint positions, $C(q,\dot{q})$ is the normal and centrifugal acceleration matrix and is dependent on the joint positions and velocities while, $H(q,\dot{q})$ contains the gravity and friction dynamics. Gravity is dependent on the joint position, while the frictional dynamics can be more complex depending on joint positions as well as velocities. The τ vector represents the torques provided to each joint. Since the trajectory tracking is performed in the task space, the dynamic model can be represented in the task space as:

$$\tau^* = M^*(q)\ddot{x} + C^*(q,\dot{q})\dot{x} + H^*(q,\dot{q})$$

The transformed matrices can be computed using the following relationships:

$$M^*(q) = J^{-T} M J^{-1}$$
$$C^*(q) = J^{-T}\left(C J^{-1} - M J^{-1} \dot{J} J^{-1}\right)$$
$$H^*(q,\dot{q}) = J^{-T} H$$
$$\tau^* = J^{-T}\tau$$

III. Control law

The control law developed in [1] will be summarized in this section. First, the following error terms are defined:

$$e = x - x_d$$
$$\dot{e} = \dot{x} - \dot{x}_d$$
$$s = \dot{e} + \Lambda e$$
$$\dot{x}_r = \dot{x}_d - \Lambda e$$
$$\ddot{x}_r = \ddot{x}_d - \Lambda \dot{e}$$

Where Λ is a diagonal positive matrix, x is a vector with the x, y, z positions of the end-effector and x_d correspond to the desired x, y, z positions of the end-effector.

.

The overall equation of the control law can be defined as:

$$\tau^* = \widehat{M}^*(q)\ddot{x}_r + \hat{C}^*(q,\dot{q})\dot{x}_r + \widehat{H}(q,\dot{q}) - K_v s + \omega$$

Where the subscript $\widehat{\ }$ in this equation represents that the term is being estimated by a neural network, K_v is a positive gain matrix and ω is a compensation signal defined as follows:

$$\omega = -K_\theta\left(\|\hat{\theta}\| + \|\theta\|_{max}\right)s$$

Where $\hat{\theta}$ contains is a matrix with all of the neural network estimated parameters that will be defined later $(\hat{\alpha}_{1,2}, \beta_{1,2}, \gamma_{1,2})$ and θ contains all of the final neural network parameters. The term $\|\theta\|_{max}$ is an estimation that can only be chosen in a heuristic way according to the authors. Finally, K_θ is defined as an approximation constant.

To compute the control law, the estimations of M, C and H must be obtained. These estimations come from tree neural networks with the following structure:

$$\widehat{M}^*(q) = \hat{\alpha}_1^T \varphi(\hat{\alpha}_2^T q)$$

$$\hat{C}^*(q,\dot{q}) = \hat{\beta}_1^T \varphi\left(\hat{\beta}_2^T q, \dot{q}\right)$$

$$\widehat{H}^*(q,\dot{q}) = \hat{\gamma}_1^T \varphi(\hat{\gamma}_2^T q, \dot{q})$$

The authors used Lyapunov stability analysis to obtain the adaptation laws of each of the neural network terms. The results of their analysis derived the following expressions:

$$\dot{\hat{\alpha}}_1 = -\kappa\Gamma_M\|s\|\hat{\alpha}_1 - \Gamma_M\left(\varphi\left(\hat{\alpha}_2^T z\right) + \varphi'\left(\hat{\alpha}_2^T z\right)\hat{\alpha}_2^T z\right)\ddot{x}_r s^T$$

$$\dot{\hat{\alpha}}_2 = -\kappa\Gamma_M\|s\|\hat{\alpha}_2 - \Gamma_M z\ddot{x}_r s^T \hat{\alpha}_1^T \varphi'\left(\hat{\alpha}_2^T z\right)$$

$$\dot{\hat{\beta}}_1 = -\kappa\Gamma_C\|s\|\hat{\beta}_1 - \Gamma_C\left(\varphi\left(\hat{\beta}_2^T \acute{z}\right) + \varphi'\left(\hat{\beta}_2^T \acute{z}\right)\hat{\beta}_2^T \acute{z}\right)\dot{x}_r s^T$$

$$\dot{\hat{\beta}}_2 = -\kappa\Gamma_C\|s\|\hat{\beta}_2 - \Gamma_C \acute{z}\dot{x}_r s^T \hat{\beta}_1^T \varphi'\left(\hat{\beta}_2^T \acute{z}\right)$$

$$\dot{\hat{\gamma}}_1 = -\kappa\Gamma_H\|s\|\hat{\gamma}_1 - \Gamma_H\left(\varphi\left(\hat{\gamma}_2^T \acute{z}\right) + \varphi'\left(\hat{\gamma}_2^T \acute{z}\right)\hat{\gamma}_2^T \acute{z}\right)s^T$$

$$\dot{\hat{\gamma}}_2 = -\kappa\Gamma_H\|s\|\hat{\gamma}_2 - \Gamma_H \acute{z} s^T \hat{\gamma}_1^T \varphi'\left(\hat{\gamma}_2^T \acute{z}\right)$$

Where with κ being a positive gain and $\Gamma_{\{M,C,H\}}$ being positive matrices.

IV. Simulink modelling

The Simulink model is divided into three sections. Section 1 models the dynamics of the robot, section 2 models the desired trajectory block while Section 3 models the control law part. The overall Simulink model is shown in Figure 2.

A. Robot dynamic model block

The dynamic model of the robot is constructed from rigid body blocks, rigid transformations, prismatic and revolute joints. The Links of the robot are modelled using cylinders while the base of the robot is modelled in SolidWorks and then imported into the Simulink environment. The robot dynamic model block is shown in Figure 3.

B. Desired trajectory block

The desired trajectory block is constructed from repeating sequence blocks. The desired end effector position, velocity and acceleration are imported into MATLAB workspace along with the time stamps. This block is shown in Figure 4.

C. Control law block

The control law block is constructed from blocks including gain, matrix transpose, matrix multiply, constant blocks and MATLAB function blocks. The control law block

takes the joint space position, velocity, and the desired trajectory to estimate the torque for the motion of the manipulator. The neural network adaptation rules are implemented in MATLAB function block which updates the neural network parameters at each time stamp. The updated neural network parameters are then used to compute the M^*, C^* and H^* matrices. It's important to note that the author's adaptation laws for the neural network parameters were not fully implemented due to mismatches in the dimension analysis of the components. Further analysis of the stability analysis needs to be performed in order to resolve this problem.

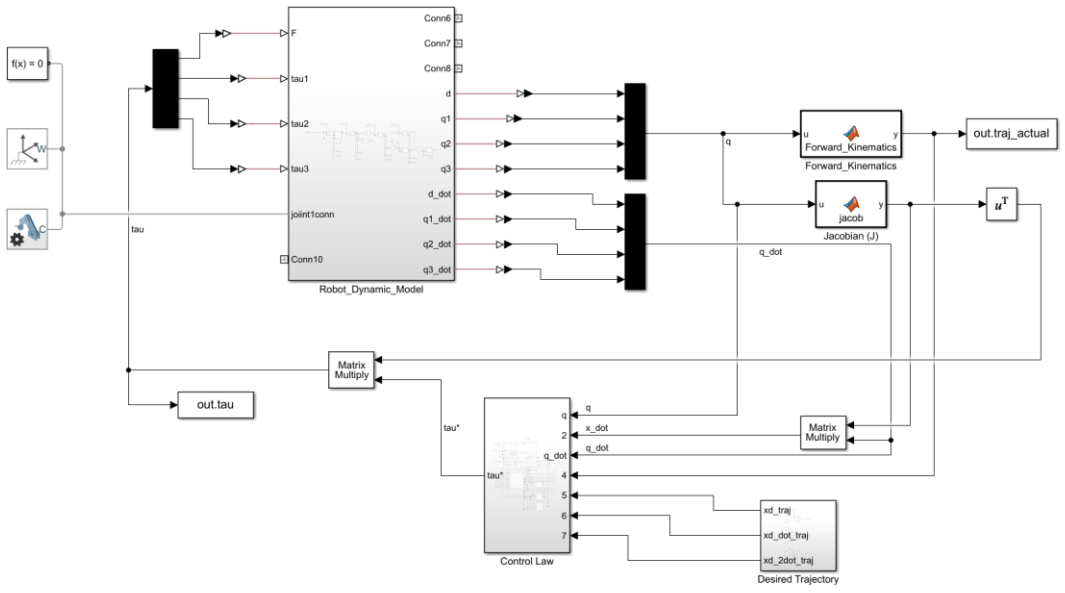

Fig. 2. *Simulink block diagram including robot model, control law and desired trajectory blocks.*

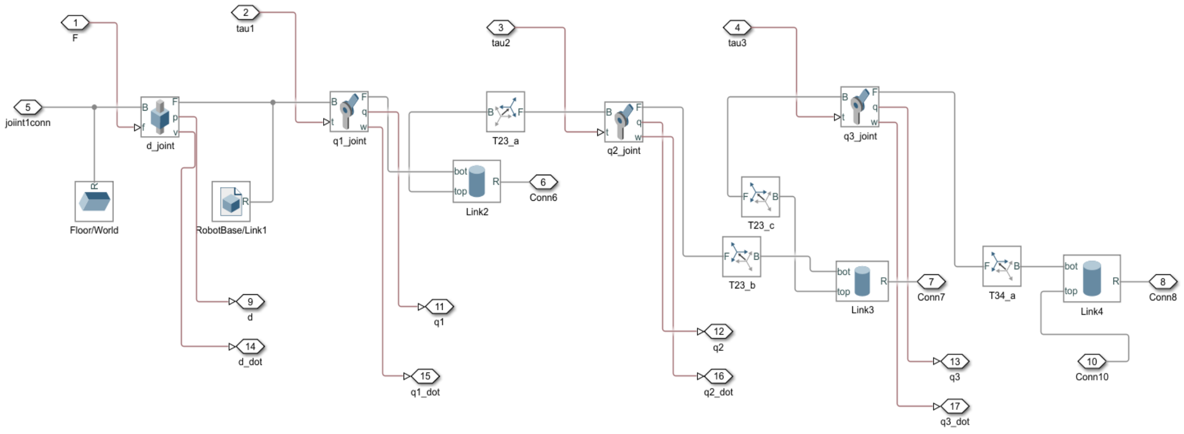

Fig. 3. *Simulink model of the robot dynamics.*

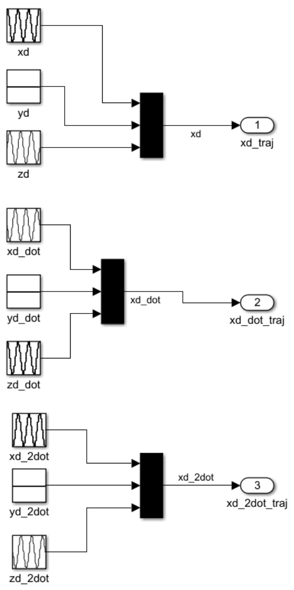

Fig. 4. *Simulink model for desired trajectory*.

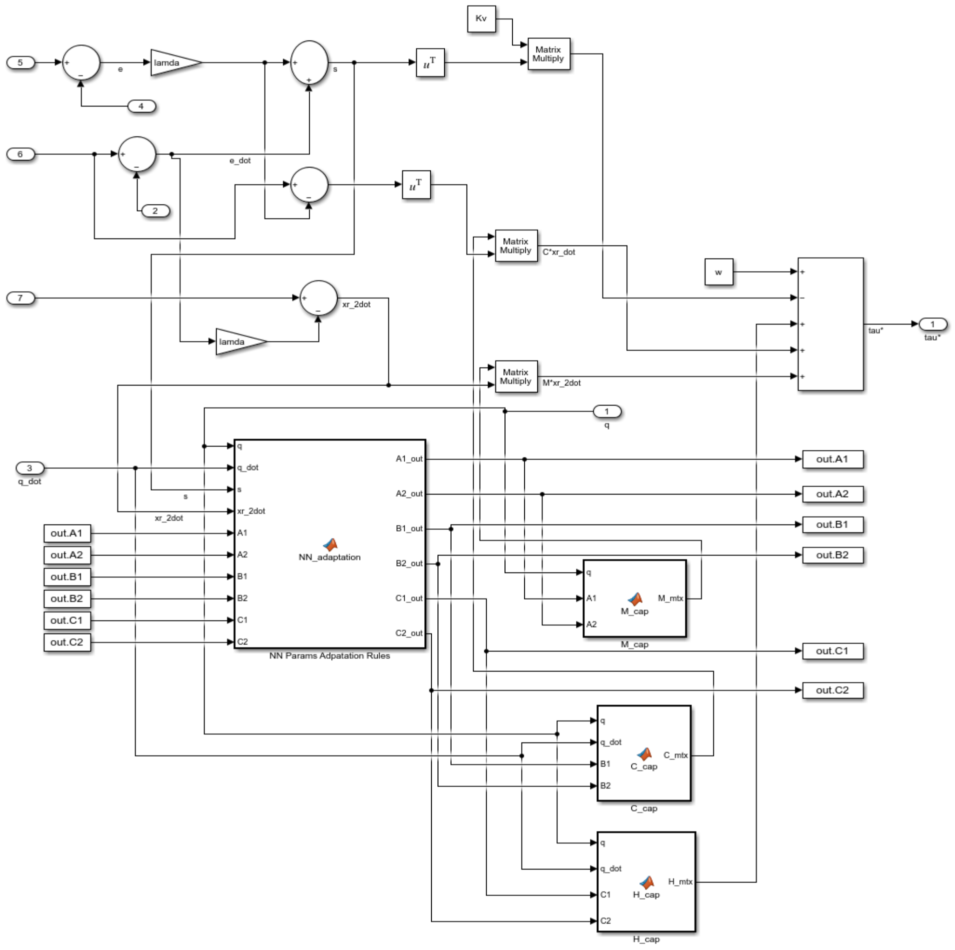

Fig. 5. *Simulink model for control law implmentation*.

V. Results and Discussions

A. Desired Trajectory

The desired trajectory is a circular one in the xz plane. A circle of radius r with center point $(O_x, 0, O_z)$ can be defined by using the following equations.

$$p_x = O_x + r\cos\theta$$
$$p_y = 0$$
$$p_z = O_z + r\sin\theta$$

Similarly, the velocity and acceleration of the end effector for a circular trajectory can be defined as:

$$v_x = -v\sin\theta$$
$$v_y = 0$$
$$v_z = v\ \cos\theta$$

$$a_x = -a\cos\theta$$
$$a_y = 0$$
$$a_z = -a\sin\theta$$

If the end effector is moving with a constant speed v, the acceleration vector only has normal term given by $a = \frac{v^2}{r}$. The desired trajectory for $r = 0.1,\ O_x = 0.1, O_x = 0.1, v = 0.1\ \ and\ \theta = [0\ 2\pi]$ is shown below in Figures 6-8.

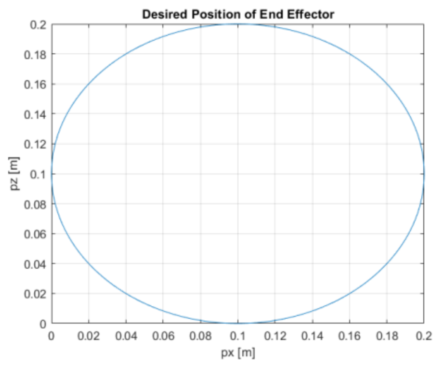

Fig. 6. *Desired Position of end effector*

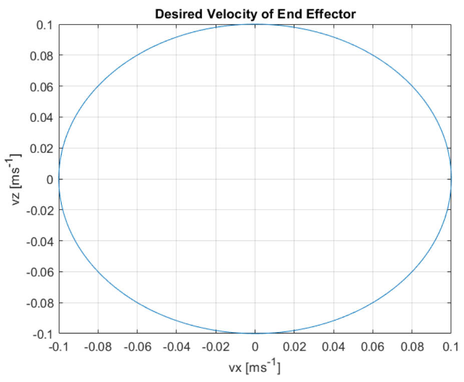

Fig. 7. *Desired Velocity of end effector*

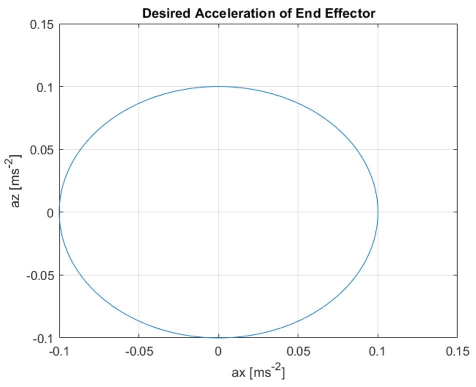

Fig. 8. *Desired Acceleartion of end effector*

B. Robot parameters and controller settings

Before analyzing the performance of the controller, it is necessary to define the dynamic parameters of the robot. These are shown in Table II.

TABLE II. Robot Dynamic Parameters

Parameter	***Desription***	***Value***
L_1	Link 1 length	0.2 m
L_2	Link 2 length	0.2 m
L_3	Link 3 length	0.3 m
M	Mass of robot base	5 kg
m_1	Link 1 mass	1 kg
m_2	Link 2 mass	0.5 kg
m_3	Link 3 mass	0.25 kg
B	Friction coefficient for prismatic joint	0.001 Ns/m
b_1	Friction coefficient for revolute joint 1	0.001 Ns/rad
b_2	Friction coefficient for revolute joint 2	0.001 Ns/rad
b_3	Friction coefficient for revolute joint 3	0.001 Ns/rad

Similarly, the controller parameters including $\lambda, K_v, \omega\ and\ \kappa$ are defined in Table III.

TABLE III. Controller Settings

Parameter	***Desription***	***Value***
λ	Error correction gain	1.2
K_v	Controller gains for x, y, z tracking	$\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1.5 \end{bmatrix}$

ω	Compensation term for torque	$\begin{bmatrix}0\\0\\0\end{bmatrix}$
κ	Adaptation strength for NN parameters	5

C. *Desired and actual trajectory*

After the simulation is run for a time of 4 s, the results are plotted. It is observed that the is making the robot follow a trajectory that is circular. The radius of the tracking circle is, however, larger than the desired one. The error remains stable in the time interval [1 2.5] s while it diverges later as shown in the error plot (Figure 9).

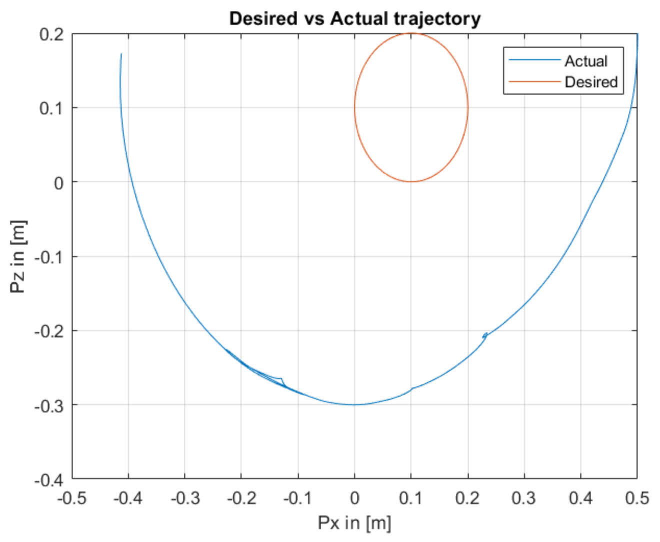

Fig. 9. *Desired vs Actual trajectory*

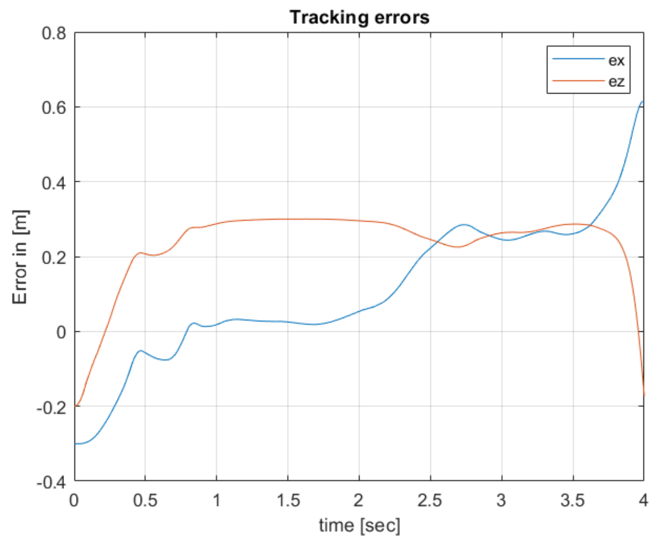

Fig. 10. *Tracking errors in x and z direction*

D. *Joint space positions*

The joint space positions change continuously updating the end effector position at each time instance. It is observed that the controller performs excellent in keeping the robot base grounded despite the large inertial forces exerted on it due to the motion of the robot arms. The joint space position plots are shown in figure 11.

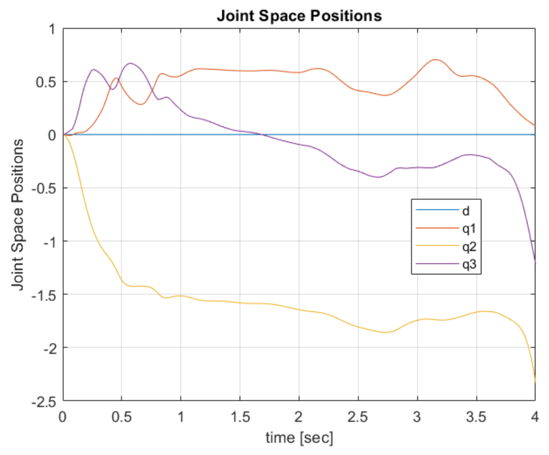

Fig. 11. *Joint Space Positions*

E. *Controller torques*

As the simulation runs, the torque curves (figure 12) evolve that indicates the controller is responding to the tracking error and changes in the M, C and H matrices. The torques adjust the position of the joints continuously to achieve the desired trajectory. The force on the base is initially large to keep the base at the ground which is confirmed by the d parameter in the joint space position plot in figure 11. The force is then constantly adjusted subject to the varying inertial forces on the robot base. The joint torques of the revolute joints are also seen to be continuously changing to get the desired trajectory tracking at the end effector of the robot.

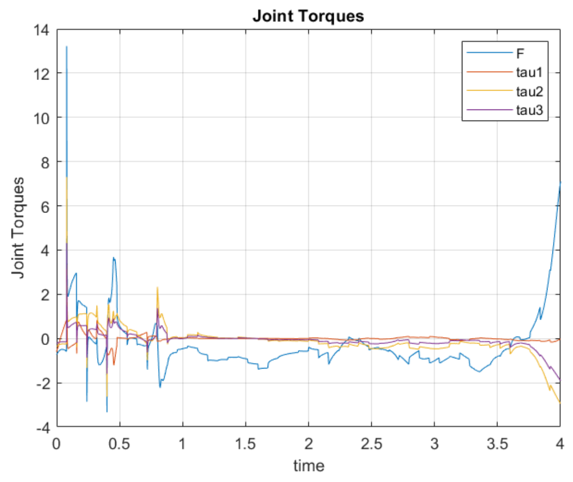

Fig. 12. *Joint Space Torques*

VI. Conclusion

The concept of estimating neural network parameters with adaptive control theory was explored in this report. The controller results were not as successful as expected, since the end-effector is following a circular trajectory but not tracking the exact desired trajectory (a circle with a smaller radius). The main cause of this behavior is that the author's adaptation laws for the neural network parameters were not implemented fully, since the dimension analysis of the adaptation laws presented a mismatch that was unable to be resolved. Further analysis of the Lyapunov stability analysis needs to be conducted in order to derive successful adaptation laws for the neural network parameters and achieve results similar to those the authors presented.

Even though the adaptation laws were not implemented in full, the modelling scheme and the controller structure was built, so if the dimension mismatch problem can be resolved, testing it with the rest of the model is a simpler task.

VII. Reference

[1] B. Daachi and A. Benallegue, "A Neural Network Adaptive Controller for End-effector Tracking of Redundant Robot Manipulators," Journal of Intelligent and Robotic Systems, vol. 46, pp. 245-262, 2006. DOI: 10.1007/s10846-006-9060-6.

[2] R. Barua, S. Kumar, A. Mallik, and A. Singh, "Experimental Analysis the Dynamic Model of 3DOF Robotic Arm," in Proceedings of the 6th International Conference on Robotics, 2021, pp. 20-26.