International Journal of Mathematical, Engineering and Management Sciences
Vol. 6, No. 6, 1694-1708, 2021
https://doi.org/10.33889/IJMEMS.2021.6.6.100

Forward Kinematics of Delta Manipulator by Novel Hybrid Neural Network

Mahesh A. Makwana
Department of Food Engineering,
College of Food Processing Technology & Bio Energy,
Anand Agricultural University, Anand, Gujarat, India.
Corresponding author: maheshmakwana@aau.in

Haresh P. Patolia
Mechanical Engineering Department,
Birla Vishvakarma Mahavidyalaya, V. V. Nagar, Gujarat, India.
E-mail: hppatolia@bvmengineering.ac.in

(Received on April 4, 2021; Accepted on November 8, 2021)

Abstract
For the parallel configuration of the robot manipulator, the solution of Forward Kinematics (FK) is tough as compared to Inverse Kinematics (IK). This work presents a novel hybrid method of optimizing an Artificial Neural Network (ANN) specifically Multilayer Perceptron (MLP) with Genetic Algorithm (GA) and Step-wise Linear Regression (SWLR) to solve the complex FK of Delta Parallel Manipulator (DPM). The joint space angular positional data has been iterated using IK to generate point cloud of Cartesian space positional data. This data set is highly random and broad which leads to higher-order nonlinearity. Hence, normalization of the dataset has been done to avoid outliers from the dataset and to achieve better performance. The developed ANN based MLP gave a mean square error of 0.0000762 and an overall R^2 value of 0.99918. Finally, the proposed network has been simulated to solve FK of the parallel manipulator and to check its efficacy. For given joint angles, the proposed network predicted positional values which are in good approximation with known trajectory solved by standard analytical method.

Keywords- Forward kinematics, Parallel manipulator, Artificial Neural Network, Genetic Algorithm, Feed forward back propagation.

1. Introduction
Parallel manipulator is one of the highly demanded robots in manufacturing, dairy, food and agriculture industries. It possesses advantages like high acceleration, high precision, low inertia and high accuracy over serial manipulator. Only disadvantage of the parallel configuration has its small work volume. But in micro manufacturing this limitation can be ignored (Uzunovic et al., 2013). For the better understanding of work volume, robot performance and manipulator control, the FK and IK problems must be solved faster and accurately. Solution of FK of parallel manipulator is highly complex and time consuming when compared to serial manipulator.

Due to their utilization capacity in different industries parallel manipulators were highly investigated in the past. DPM is one of the types of parallel robot which is widely used in industries. Delta robot was invented in the early 1980s by Prof. Reymond Clevel, Laboratory of Robotics System, Swiss Federal Institute of Technology, Switzerland. Then after many modified delta robots have been developed for different applications. Robot movement can be predicted by incorporating FK and trajectory planning techniques (Ang et al., 2014). Analytical solution of FK for DPM tends to number of solutions with tedious computations. Its solution usually involves systems of nonlinear equations which are highly coupled and in general have no closed form and unique solution. It requires the solution of multiple coupled nonlinear algebraic equations which brings multiple valid solutions (Sadjadian et al., 2005). This problem can be overcome using ANN which converge with efficient single solution with less time and memory consumption.

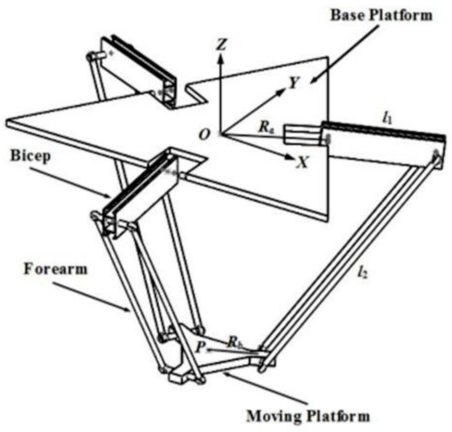
ANNs, due to their extreme flexibility and the capability of mapping nonlinearity within complex dataset may be explored for many potential applications in the field of robotics. Many efforts have been made on applications of neural networks for various types of parallel manipulators (Parikh and Lam, 2005; Sang and Han, 1999; Zhang and Lei, 2011). Supervised ANN can certainly learn the input-output data to find the non-linear relationships which are inherent in the training data (Elsheikh et al., 2013). Many types of neural networks namely MLP, Radial Basis Function (RBF), Support Vector Machine (SVM) are considered to solve the FK of parallel manipulator. MLP takes less time for the solution as compared to the other methods. No significant difference can be found in single hidden layer MLP and double hidden layer MLP (Ghasemi et al., 2017; Ramanababu et al., 2013). Back Propagation Neural Network (BPNN) has maximum potentiality to be fundamental topology for solving kinematics problems of manipulator which is easy to implement and tends to converge in good approximations (Tang et al., 2014). Further, BPNN is an appropriate method for solving the forward kinematics because of its various advantages, such as high efficiency and high convergence ratios (Zhang et al., 2019).

Even for real control of parallel robot ANN can be used to solve FK (Lv et al., 2017). ANN is possible to implement in a hardware form, which can result in a multi-fold reduction in the solution time for the same accuracy level (Parikh and Lam, 2009). ANN with a hybrid approach using GA to model the parallel system and for solving FK and IK problems of the Parallel Continuum Robot gives acceptable precision and is considered one of the best approach to model the parallel system (Wu et al., 2017). Apart from FK, ANN can also solve IK and the same should be applied to design a controller which gives advantages over the existing techniques for minimizing the position error (Almusawi et al., 2016). FK solution method which uses ANN that relies on training with a certain number of iterations can further be opted to determine workspace of the robot (Youssef et al., 2019). Designing a parallel robot with maximum workspace, both ANN and GA approaches serve as Optimum Algorithm (López et al., 2021).

The paper follows, structural configuration of delta robot, explanation of IK and generation of input-output data-set to be used in the ANN training. Next, the development of MLPs, optimization using GA and its simulation as well as validation. Towards the end conclusion and future scopes have been discussed.

2. Model Description of Delta Type Parallel Manipulator

The DPM, as shown in Figure 1, is made of three parallel kinematic chains connected to end-effector attached with moving platform. Each chain is consisting of bicep and forearm which in turn connected to moving platform. Bicep is actuated by an actuator fixed to the manipulator base plate. All actuators are placed at 120º apart from each other. Moving platform can be manipulated by angular motions of the biceps and connected forearms. In configuration, biceps are active limbs whereas forearms are passive limbs. Parallelogram design of manipulator ensures the parallel movement of moving platform. The robot is of Revolute-Universal-Universal (RUU) joint configuration. Robot under study has been developed in-house as shown in Figure 2. The geometrical parameters of the robot defined as base platform radius R_a= 147.2 mm, moving platform radius R_b= 73.6 mm, bicep length l_1= 150 mm and forearm length l_2= 380 mm. Throughout the analysis, the geometric center of the manipulator base *O* has been considered as an origin of the absolute coordinate frame. The geometric center of the moving platform *P*, is the end-effector position with respect to reference *O*.



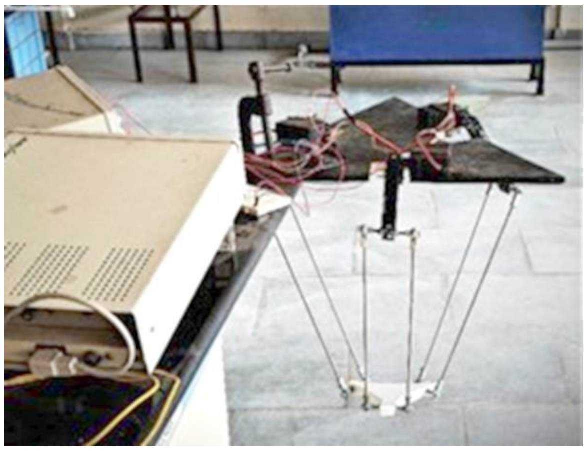
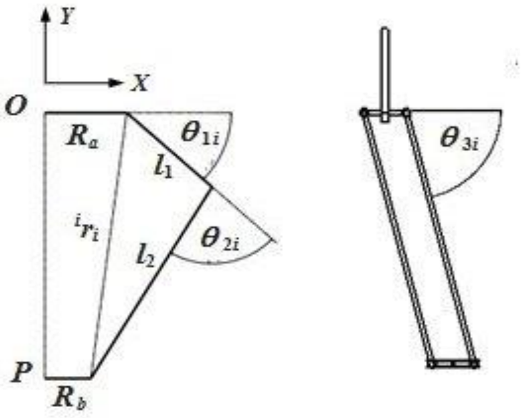


Figure 1. Description of D-PM.

Figure 2. Developed model of D-PM.

3. Inverse Kinematics of Delta Robot

IK is mathematical technique to calculate joint space parameters of robotic manipulator by feeding coordinate space parameters, which means calculating angular rotations of links using given Cartesian coordinates. For given D-PM, IK calculates angular position vector for biceps expressed by $\theta_{1i} = (\theta_{11}, \theta_{12}, \theta_{13})$ using x, y, z coordinates of end-effector $P = (p_x, p_y, p_z)$. Here θ_{1i} is the angle between vector R_a and bicep l_1 and θ_{2i} is the angle between bicep l_1 and forearm l_2 as shown in Figure 3. The clockwise rotation from *X*-axis has been considered positive angular movement, θ_{3i} is rotation of forearm in *YZ*-plane. As shown in Figure 4, α_1 is angle between *X*-axis to position of actuator A_1, α_2 is angle between positions of actuators *X*-axis and A_2 and α_3 is angle between actuators *X*-axis and A_3.



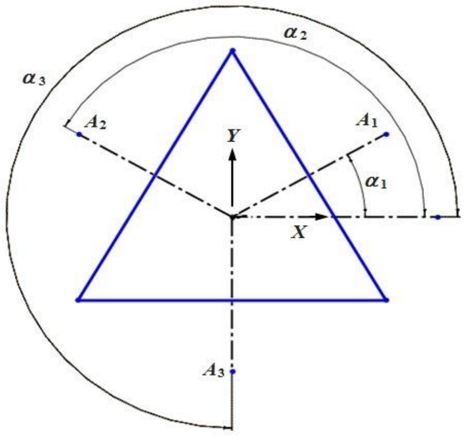


Figure 3. Geometrical parameters of D-PM.

Figure 4. Absolute coordinate system.

The end-effector position *P* can be defined with reference to point *O* as,

$$p = l_{1i} + l_{2i} + R_{ai} - R_{bi} \text{ where, } i = 1, 2, 3 \tag{1}$$

The radius of base platform and moving platform, $R_a = |R_{ai}|$ and $R_b = |R_{bi}|$ respectively measured with reference to point *O*. The angular measurement taken about the *Z*-axis, both for base and moving platform (Brinker et al., 2015).

The transformation matrix of angle α_i about the Z-axis is given as,

$$^{i}T_o = \begin{bmatrix} \cos\alpha_i & \sin\alpha_i & 0 \\ -\sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

An auxiliary vector $^{i}r_i$ as shown in Figure 3, pointing along the links l_{1i} and l_{2i} in between the position of the actuated joint R_{ai} and the corresponding joint R_{bi} connecting the moving platform and the forearm has been defined as:

$$^{i}r_i = -^{i}r_{ai} + ^{i}p + ^{i}r_{bi} = \begin{bmatrix} -R_a + R_b \\ 0 \\ 0 \end{bmatrix} + ^{i}T_o \cdot p \tag{3}$$

$$= \begin{bmatrix} -R_a + R_b + p_x \cos\alpha_i + p_y \sin\alpha_i \\ -p_x \sin\alpha_i + p_y \cos\alpha_i \\ p_z \end{bmatrix} \tag{4}$$

So, as per the vector loop notation vector $^{i}r_i$ can be described as:

$$^{i}r_i = ^{i}l_{1i} + ^{i}l_{2i} = \begin{bmatrix} l_{1i} \cdot \cos\theta_{1i} \\ 0 \\ l_{1i} \cdot \sin\theta_{1i} \end{bmatrix} + \begin{bmatrix} l_{2i} \cdot \sin\theta_{3i} \cdot \cos(\theta_{1i} + \theta_{2i}) \\ l_{2i} \cdot \cos\theta_{3i} \\ l_{2i} \cdot \sin\theta_{3i} \cdot \sin(\theta_{1i} + \theta_{2i}) \end{bmatrix} \tag{5}$$

The actuator A_1 rotates about negative *Y*-axis by angles θ_{1i} relate to the direction of rotation. Hence, auxiliary angle θ_{3i} can be determined by using Eqs. (4) and (5) and the second component of the resulting relationship is,

$$\theta_{3i} = \arccos\left[\frac{-p_x \sin\alpha_i + p_y \cos\alpha_i}{l_{2i}}\right] \tag{6}$$

with the help of r_i

$$^{i}r_{iy}{}^{2} + ^{i}r_{iz}{}^{2} = l_{1i}^2 + l_{2i}^2 + 2l_{1i} \cdot l_{2i} \sin\theta_{3i} \cdot \cos\theta_{2i} \tag{7}$$

The secondary passive angle between bicep and forearm is given by

$$\theta_{2i} = \text{acos}\left[\frac{{}^i r_{ix}{}^2 + {}^i r_{iy}{}^2 + {}^i r_{iz}{}^2 - l_{1i}^2 - l_{2i}^2}{2l_{1i}\cdot l_{2i}\cdot \sin\theta_{3i}}\right] \tag{8}$$

where, 0 < θ_{2i} < 180°. With these angles, the actuation angle can be derived as,

$$\sin\theta_{1i} = -\frac{-l_{1i}\cdot{}^i r_{iz} - l_{2i}\cdot s\theta_{3i}\cdot c\theta_{2i}\cdot{}^i r_{iz} + l_{2i}\cdot s\theta_{3i}\cdot s\theta_{2i}\cdot{}^i r_{ix}}{2l_{1i}\cdot l_{2i}\cdot s\theta_{3i}\cdot c\theta_{2i} + l_{1i}^2 + l_{2i}^2 - l_{2i}^2\cdot c^2\theta_{3i}} \tag{9}$$

$$\cos\theta_{1i} = \frac{l_{1i}\cdot{}^i r_{ix} - l_{2i}\cdot s\theta_{3i}\cdot s\theta_{2i}\cdot{}^i r_{iz} + l_{2i}\cdot s\theta_{3i}\cdot c\theta_{2i}\cdot{}^i r_{ix}}{2l_{1i}\cdot l_{2i}\cdot s\theta_{3i}\cdot c\theta_{2i} + l_{1i}^2 + l_{2i}^2 - l_{2i}^2\cdot c^2\theta_{3i}} \tag{10}$$

From Eqs. (9) and (10), the angle $\tan\theta_{1i}$ is obtained as,

$$\theta_{1i} = a\tan - \frac{-l_{1i}\cdot{}^i r_{iz} - l_{2i}\cdot s\theta_{3i}\cdot c\theta_{2i}\cdot{}^i r_{iz} + l_{2i}\cdot s\theta_{3i}\cdot s\theta_{2i}\cdot{}^i r_{ix}}{l_{1i}\cdot{}^i r_{ix} - l_{2i}\cdot s\theta_{3i}\cdot s\theta_{2i}\cdot{}^i r_{iz} + l_{2i}\cdot s\theta_{3i}\cdot c\theta_{2i}\cdot{}^i r_{ix}} \tag{11}$$

4. Multilayer Perceptron

Many past works show that MLP are very much capable of approximating an XOR operator as well as many other non-linear functions. MLP is a type of ANN made up of more than one hidden layer. The basic configuration is input layer to output layer and in between them it consists of arbitrary numbers of hidden layers. Mathematically expressed as:

$$P = \varphi\sum_{i=1}^{n} W_i\theta_i + b \tag{12}$$

where, P is the output, W is the weightage applied to each neuron, θ is an input, b is bias and φ is an activation function. In supervised learning problems, MLP can train set of input-output data and learn to make regression to find best possible correlation between data to minimize error. BPNN is used to make those weightage and bias adjustments relative to the error. Performance of MLP can be determined by Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Square Error Regression (MSEReg) and Sum Square Error (SSE).

In this work, MLP with Feed Forward BPNN with different combinations have been used for fitting highly nonlinear dataset. Then, to improve the performance of MLP optimization using GA has been employed. For hidden layers Activation function <TANSIG> and output layer <PURELIN> have been used. Gradient decent with momentum weightage and bias learning function has been used for adaption learning. The first criterion of activation function is easy and fast convergence of the network. <TANSIG> is a hyperbolic tangent sigmoid transfer function which can map between +1 to -1 and hence good choice to map the negative inputs. Negative and the zero inputs will be mapped near zero as shown in Figure 5. Moreover, the advantage over the sigmoid function is that its derivative is steeper, which means it will be more efficient because it has a wider range for faster learning and grading. Function can be expressed mathematically as:

$$\varphi = \text{tansig}(x) = \frac{2}{1+e^{-2x}} - 1 \tag{13}$$

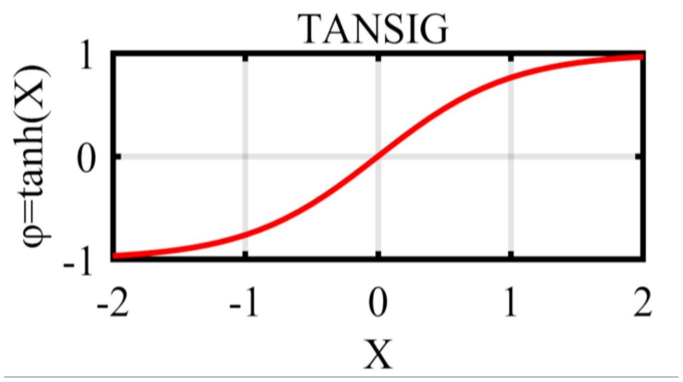


Figure 5. Hyperbolic tangent function.

The most appropriate activation function for the output neuron(s) of a Feed Forward Neural Network used for regression problems is a linear activation, even the data initially normalised. For output layer, <PURELIN> function $f(u)=u$ has been used. MSE is chosen to check the performance of the network. The approximated Hessian matrix H is given as,

$$H = J^T \cdot J \tag{14}$$

Here, Hessian matrix H must be invertible. The gradient can be computed as

$$g = J^T e \tag{15}$$

In above equation, J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weightages and biases and e is a vector of network errors. The complexity to calculate Jacobian matrix through a standard BPNN technique is much less than computing the hessian matrix. Levenberg–Marquardt algorithm also known as Damped Least Squares (DLS) introduces another approximation to Hessian matrix represented by following equation,

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e \tag{16}$$

In equation (16), μ is always positive and called as combination coefficient and I is an identity matrix. When the value of scalar μ is zero, it becomes Newton's method which uses the approximate hessian matrix. When μ is large, it becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error Minima, so the aim is to shift toward Newton's method as quickly as possible. Thus, μ is decreased after each successful iteration and increased only when a tentative step would increase the performance function. In this way, the performance function is always reduced at each iteration of the algorithm. If the combination coefficient μ is very large, it can be interpreted as the learning coefficient in the steepest descent method.

$$a = \frac{1}{\mu} \tag{17}$$

The adaptive learning function <learngdm>, calculates the weightage change dW for a given neuron from the input *P* and error *E*, the weightage (or bias) W, learning rate l_r, and momentum constant mc, according to gradient descent with momentum. The weightage change dW is given as,

$$dW = mc \cdot dW_{prev} + (1 - mc) l_r \cdot g \cdot W \tag{18}$$

The gradient descent algorithm takes a long time to find the minimum value of the error function when selecting very small learning rate. This defeats the purpose of gradient descent, which uses a computationally efficient method for finding the optimal solution. The standard gradient descent procedure uses a fixed learning rate 0.01 that is determined by trial and error. A default value typically works for standard multi-layer neural networks. The same value has been chosen for training and it works well to obtain best fit.

5. Training of MLP

To obtain joint space angular data, Matlab code has been generated to solve standard equations of IK and subsequently has been simulated for iterative generation of 5367 coordinate points which are within the reachable workspace. After normalizing of obtained dataset, joint space angular data has been fed to MLP as an input and corresponding coordinate points as an output. Input-output dataset has been represented in Figure 6.

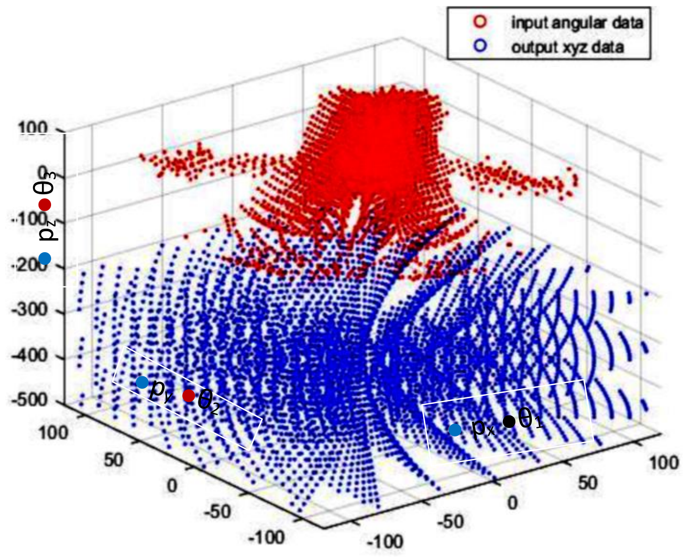


Figure 6. Graphical representation of input and output data.

Initially, neural network has been iterated with single hidden layer and other methods viz, NARX based wavelet neural network (WNN) and Radial Basis Neural Network (RBNN). But due to high nonlinear relationship among input-output dataset of system under study, required performance could not be achieved. Hence, two hidden layer MLP has been tried with different combination of neurons in each layer with training parameters as Table 1.

Table 1. Training parameters.

Min_Gradient	μ	μ_decrease	μ_increase	μ_max	epoch
1×10^{-07}	0.001	0.1	10	1×10^{10}	1000

As the input-output dataset is very complex, multiple hidden layers of neurons are needed to be learned with high levels of accuracy. Moreover, this hybridization approach is meant to improve positional accuracy and precision of manipulator and thus high efficiency in terms of network performance is expected. So, by choosing two hidden layers, the combination of performance parameters viz., MSE and R^2 are more desired and solve the original purpose.

Generation of an objective function should be carefully carried out for efficient and comprehensible optimisation. Hence, the number of neurons for each layer is taken after some pre-trials to check MSE. Performance of different neuron combination and its effect on the performance of MLPs are shown in Table 2. For further improvement in performance GA optimization has been adopted.

Table 2. Iteration of MLP with different neurons combination.

ANN	HL1 Neurons	HL2 Neurons	MSE	R-square	Time (min)	Epoch
MLP1	20	--	0.011500	0.99119	0.8	60
MLP2	30	--	0.006350	0.99564	2.16	79
MLP3	40	--	0.004360	0.99572	2.12	72
MLP4	05	05	0.055700	0.97148	2.03	78
MLP5	06	05	0.042800	0.97726	2.52	110
MLP6	10	05	0.019300	0.98736	1.41	63
MLP7	20	05	0.004230	0.99767	3.50	136
MLP8	10	10	0.002510	0.99803	1.51	68
MLP9	10	15	0.000981	0.99817	4.37	158
MLP10	20	10	0.001040	0.99774	1.34	50
MLP11	05	10	0.011800	0.99353	2.53	108
MLP12	10	20	0.000901	0.99921	2.41	86
MLP13	10	25	0.000532	0.99952	1.02	105
MLP14	10	30	0.001480	0.99834	1.42	45
MLP15	20	30	0.000136	0.99980	6.03	103
MLP16	20	20	0.000775	0.99879	1.25	34
MLP17	30	10	0.000575	0.99914	3.03	81
MLP18	30	20	0.000477	0.99929	1.18	21
WNN-NARX	10	--	0.067600	0.96384	50	18
WNN NLIO	10	--	0.118000	0.92616	1.6	27
RB1	89	--	0.026000	0.97212	301.85	50
RB2	50	--	0.026000	0.97212	123.58	50

6. Algorithm for Optimising Neurons in Hidden Layers

A nature inspired evolutionary algorithm called GA was originally developed by John Holland in 1960. It involves an initialization method, fitness function to evaluate each chromosome, natural selection, crossover, and mutation operators (Lamini et al., 2018). The objective is to minimize mean square error which depends on number of neurons in hidden layers. Two depended variables have been chosen as n_1 and n_2 which are number of neurons in hidden layer 1 and hidden layer 2 respectively. An objective function can be written as,

$$f(x) = f(n_1, n_2) \tag{19}$$

6.1 Objective Function

The required fitness function also called as objective function for this study has been developed by nonlinear regression which can be mathematically expressed as:

$$f(x) = a + bn_1 + cn_2 + dn_1^2 + en_1n_2 + fn_2^2 + gn_1^3 + hn_1^2n_2 + in_1n_2^2 + jn_2^3 + kn_1^3n_2 + ln_1^2n_2^2 + mn_1n_2^3 + nn_2^4 \quad (20)$$

where, coefficients are;

$a = 0.1946$; $b = -0.04477$; $c = 0.01018$; $d = 0.004642$; $e = -0.001447$; $f = -0.00223$;
$g = -0.0001412$; $h =$ -0.0002465; $i = 0.0005365$; $j = -4.794 \times 10^{-05}$; $k = 1.37 \times 10^{-05}$;
$l = -1.92 \times 10^{-05}$; $m = 6.415 \times 10^{-07}$; $n = 5.139 \times 10^{-07}$.

6.2 Constraint Equation

Similarly, constraint equation has been developed based on performance of different combinations of neurons as Table 2. For constraint other consideration has been taken as both the variables are positive integers. An inequality constraint function active at constraint boundary can be expressed as,

$$n_1 + n_2 \leq 51 \quad (21)$$

6.3 Genetic Optimisation Result

Using GA obtained values of optimized number of neurons for hidden layer 1 and hidden layer 2 are represented in Table 3. Minimum penalty value is –7.12321. These values have been applied to train new MLP which shows improved performance.

Table 3. Optimised result.

Minimum Penalty Value	**Hidden Layer 1 (n_1)**	**Hidden Layer 1 (n_2)**
-7.12321	30	21

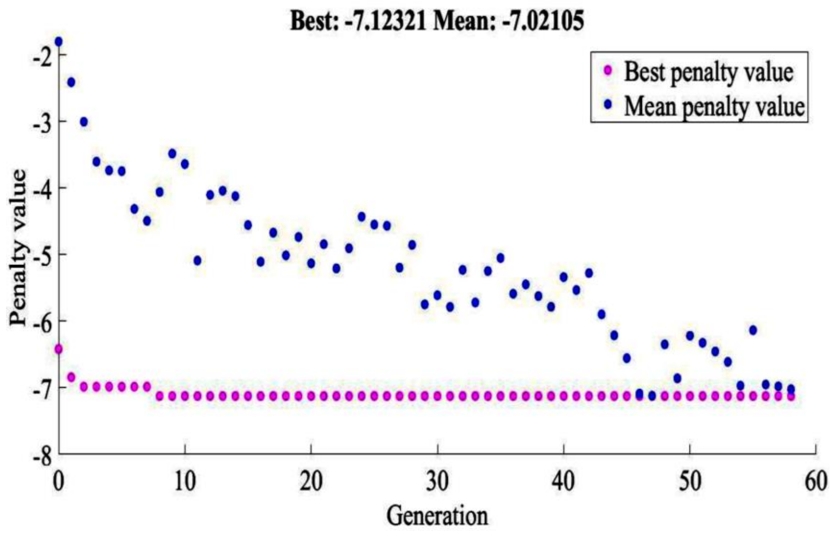


Figure 7. Penalty value versus generation.

For handling the constraint of evolutionary optimization problem penalty function can panelized infeasible solutions by reducing their fitness values as they violate the constraints. Figure 7 shows best and mean penalty values with each generation. For best feasible solution penalty value is –7.12321 at 58th iteration and mean penalty value is –7.2105.

Optimized solutions for both independent variables n_1 and n_2 are 30 and 21 respectively as shown in Figure 8.

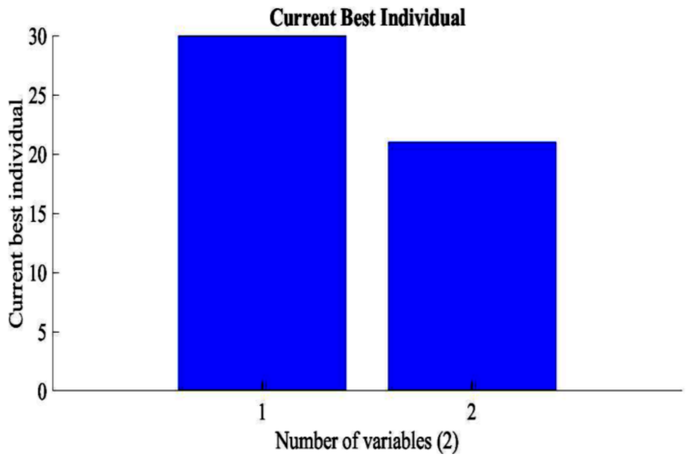


Figure 8. Best individuals plot.

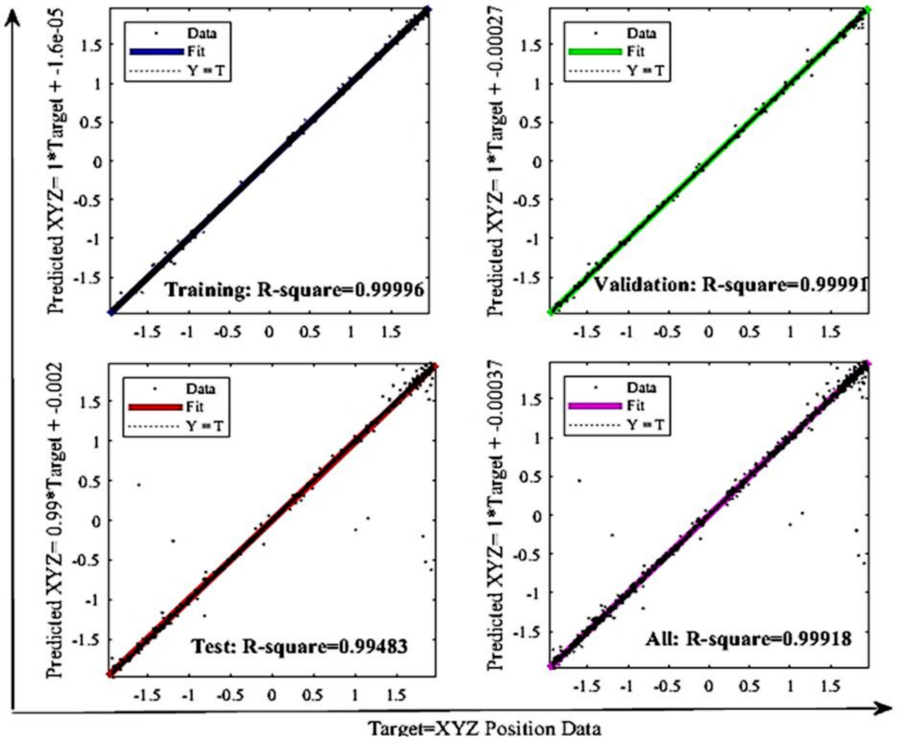


Figure 9. Regression plot of developed MLP.

7. Results and Discussion

New MLP having optimized number of neurons after applying GA has been trained. Certainly, it gave better performance for system under study as in 169 epoch trained within 6 minutes 53 seconds gave the overall R^2 value of 0.99918. MSE is 0.0000762 which is considered best among all trained ANN. Corresponding values of R^2 for Training, Validation and Testing is 0.99996, 0.99991 and 0.99483 which can be shown in Figure 9.

Performance plot as shown in Figure 10, the convergence of MSE over each epoch and for the training the value of MSE is 0.0000762 and for validation 0.00018035 at epoch 163.

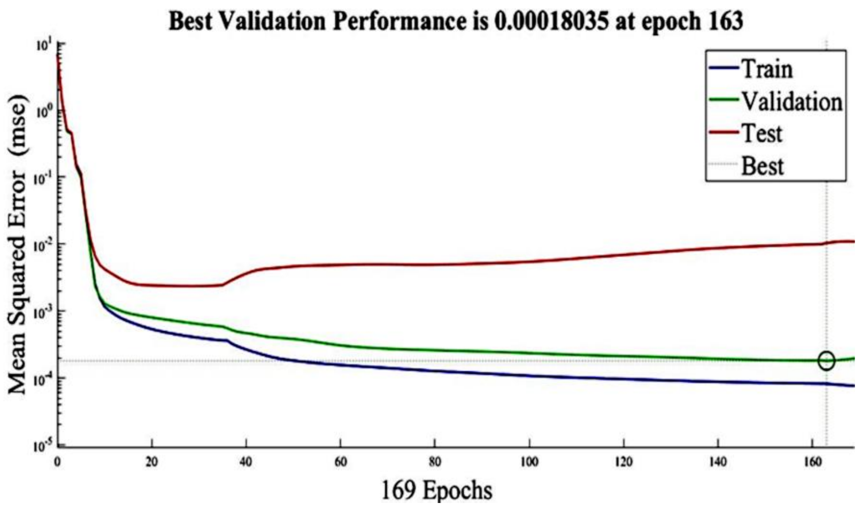
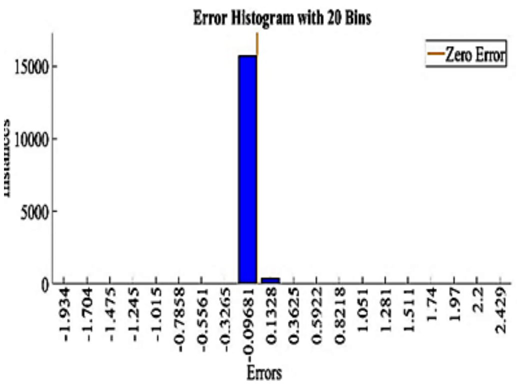


Figure 10. Performance chart of developed MLP.

Figure 11 depicts error histogram of MLP. The total error range is divided into 20 smaller bins which are the vertical bars on the plot. *Y*-axis represents the number of samples of output dataset. Error histogram is centered at –0.09681.



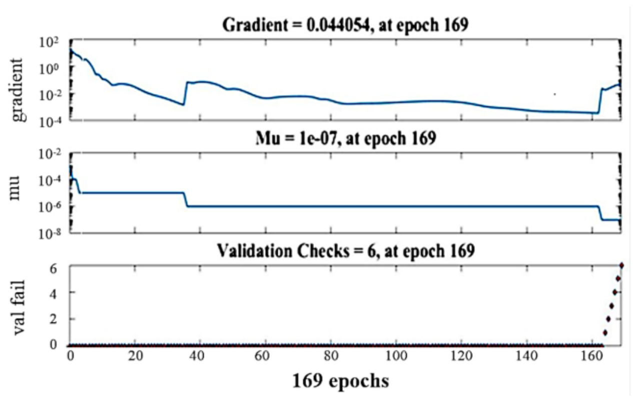


Figure 11. Error histogram of MLP.

Figure 12. Training statistics of MLP.

From Figure 12, it has been found that gradient at 169th epoch for training is 0.044054 with value of combination coefficient (μ) 1×10^{-07} is desired. All required performance parameters are shown in Table 4. As per hypotheses, GA optimization of hidden layer neurons drastically improves the performance of MLP and statistically at par with performance of un-optimized MLPs.

Table 4. Final results of optimised MLP.

Performance criteria	**Value**
Overall Mean Square Error (MSE)	0.0000762
Training Performance	0.99996
Testing Performance	0.99483
Validation Performance	0.99991
Overall R value	0.99918
Best validation performance (epoch)	163
Error Histogram centred at (bell curve)	-0.09681
Total epoch runs	169

Though the hybrid approach gives good performance in solving the FK, there are certain constraint as there is no specific rule for determining the structure of ANN. Moreover, only through trial-and-error appropriate network structure can be achieved. Fine tuning of all the parameters for the GA, like mutation rate, elitism percentage, crossover parameters, fitness normalisation/selection parameters etc., is often just trial and error. For GA, designing a fitness function and getting the representation and operators right can be difficult. It has to perform very carefully otherwise results may be inefficient or incomprehensible.

8. Validation of MLP

Finally, developed optimized MLP has been validated for given input of angular data. These inputs are solved analytically to generate trajectory within constrained workspace. Also, FK has been solved using developed MLP. The output data obtained from MLP is normalized data. So, to derive the required real values of de-normalized output SWLR has been used. Both the trajectories overlap as shown in Figure 13. Obtained results are very much accepted within constrain angular positions for system under study.

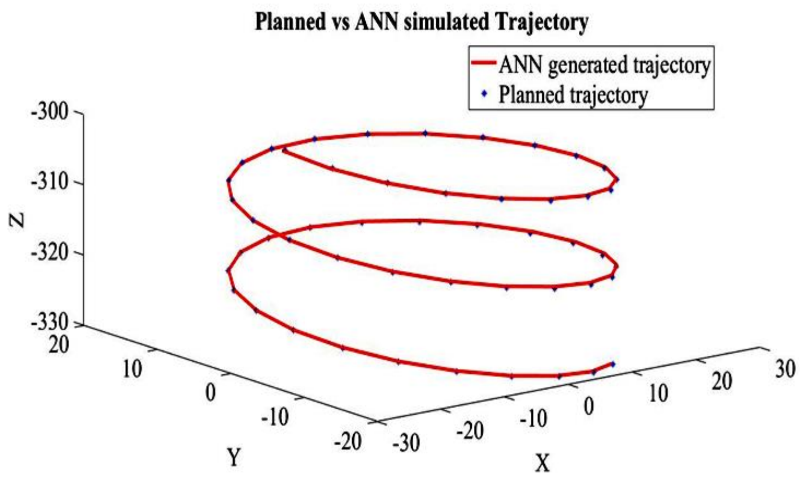


Figure 13. Desired and MLP solved trajectories.

9. Conclusion and Future Scope

The solution to the FK problem in parallel manipulators is essential for a closed-loop control system. But it has been known that close form single solution of FK comprises of nonlinear equation is very difficult and could lead to several multiple solutions. In this paper solution of the problem has been tried to solve with genetically optimized MLP. Presented hybrid GA-MLP Algorithm can generate the best estimation of FK of the manipulators under study. For application in robotics where precise movement of an end effector requires, MLP with optimized neurons in each layer using GA turns in to more accurate FK solution. Furthermore, when the configuration is at kinematic singularities, the proposed approach can provide solution for the problem with minimal errors. Alternatively, this approach can be applied to find FK solutions at singularity points. Later on, singular points can be avoided when implementing actual hardware to achieve better control of the manipulator.

The work mainly used new approach to solve FK problem of delta robot. No one could have potentially researched about it to an "exact" focus before optimizing number of neurons to improve performance of ANN. Hybridization of optimization techniques with neural network improves the results. Main advantage of this approach is to find single solution. This approach can be applied for easy controlling of system in real time to have precise control of parallel manipulator. This approach is also recommended and applicable for other class of robotic manipulators and complex mechanical systems for solving different complex non-linear equations to obtain kinematics and dynamics.

As discussed, different types of ANN and machine learning techniques have very large scope of application in robotics and allied branches. Hence, for class of parallel manipulator different areas viz. dynamics mapping, performance of manipulator, trajectory planning and hardware implementation of developed MLP will be explored. Furthermore, to solve FK some hybrid techniques and combination of optimization techniques will be explored.

Conflict of Interest
The authors confirm that there is no conflict of interest for this publication.

Acknowledgement
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors would like to thank the editor and anonymous reviewers for their comments that help improve the quality of this work.

References

Almusawi, A.R.J., Dülger, L.C., & Kapucu, S. (2016). A new artificial neural network approach in solving inverse kinematics of robotic arm (denso VP6242). *Computational Intelligence and Neuroscience*, *2016*, 1-10. https://doi.org/10.1155/2016/5720163.

Ang, C.K., Tang, S.H., Mashohor, S., & Arrifin, M.K.A.M. (2014). Solving continuous trajectory and forward kinematics simultaneously based on ANN. *International Journal of Computers, Communications and Control*, *9*(3), 253-260. https://doi.org/10.15837/ijccc.2014.3.112.

Brinker, J., Corves, B., & Wahle, M. (2015). A comparative study of inverse dynamics based on clavel's delta robot. In *2015 14th International Federation for the Promotion of Mechanism and Machine Science World Congress* (pp. 89-98). IFToMM. Taipei, Taiwan. https://doi.org/10.6567.

Elsheikh, A., Showaib, E., & Asar, A.E.M. (2013). Artificial neural network based forward kinematics solution for planar parallel manipulators passing through singular configuration. *Advances in Robotics & Automation 2*(2), 1-6. https://doi.org/10.4172/2168-9695.1000106.

Ghasemi, J., Moradinezhad, R., & Hosseini, M.A. (2017). Kinematic synthesis of parallel manipulator via neural network approach. *International Journal of Engineering*, *30*(9), 1319-1325. https://doi.org/10.5829/ije.2017.30.09c.04.

Lamini, C., Benhlima, S., & Elbekri, A. (2018). Genetic algorithm based approach for autonomous mobile robot path planning. *Procedia Computer Science*, *127*, 180-189. https://doi.org/10.1016/j.procs.2018.01.113.

López, E.G., Yu, W., & Li, X. (2021). Optimum design of a parallel robot using neuro-genetic algorithm. *Journal of Mechanical Science and Technology*, *35*(1), 293-305. https://doi.org/10.1007/s12206-020-1229-6.

Lv, W., Tao, L., & Hu, Y. (2017). On the real-time calculation of the forward kinematics of a suspended cable-driven parallel mechanism with 6-degree-of-freedom wave compensation. *Advances in Mechanical Engineering*, *9*(6), 1-17. https://doi.org/10.1177/1687814017706264.

Parikh, P.J., & Lam, S.S. (2009). Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy. *International Journal of Advanced Manufacturing Technology*, *40*(5-6), 595-606. https://doi.org/10.1007/s00170-007-1360-x.

Parikh, P.J., & Lam, S.S.Y. (2005). A hybrid strategy to solve the forward kinematics problem in parallel manipulators. *IEEE Transactions on Robotics*, *21*(1), 18-25. https://doi.org/10.1109/TRO.2004.833801.

Ramanababu, S., Raju, V.R., & Ramji, K. (2013). Neural network solutions of forward kinematics for 3RPS parallel manipulator. In *2013 proceedings of the 1st International and 16th National Conference on Machines and Mechanisms* (pp. 1064-1070). AMM & IFToMM. IIT Roorkee, India.

Sadjadian, H., Member, H.D.T., & Fatehi, A. (2005). Neural networks approaches for computing the forward kinematics of a redundant parallel manipulator. *International Journal of Computational Intelligence*, *2*(1), 40-47. https://doi.org/10.5281/zenodo.1328934.

Sang, L.H., & Han, M.C. (1999). Estimation for forward kinematic solution of Stewart platform using the neural network. In *1999 Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients* (Vol. 1, pp. 501-506). IEEE. Kyongju, Korea. https://doi.org/10.1109/iros.1999.813053.

Tang, S.H., Ang, C.K., Ariffin, M.K.A.B.M., & Mashohor, S.B. (2014). Predicting the motion of a robot manipulator with unknown trajectories based on an artificial neural network. *International Journal of Advanced Robotic Systems*, *11*(10), 1-9. https://doi.org/10.5772/59278.

Uzunovic, T., Golubovic, E., Baran, E.A., & Sabanovic, A. (2013). Configuration space control of a parallel Delta robot with a neural network based inverse kinematics. In *2013 8th International Conference on Electrical and Electronics Engineering (ELECO)* (pp. 497-501). IEEE. Bursa, Turkey. https://doi.org/10.1109/ELECO.2013.6713892.

Wu, G., Shi, G., & Shi, Y. (2017). Modeling and analysis of a parallel continuum robot using artificial neural network. In *2017 IEEE International Conference on Mechatronics(ICM)* (pp. 153-158). IEEE. Churchill, VIC, Australia. https://doi.org/10.1109/ICMECH.2017.7921096.

Youssef, A., Bayoumy, A.M., Rostom, M., & Tolbah, F.A. (2019). Modelling and simulation of 3DOF parallel manipulator using artificial neural network. In *2019 18th International Conference on Aerospace Sciences & Aviation Technology*. *IOP Conference Series: Materials Science and Engineering* (Vol. 610, pp. 1-10). IOP. Kobry Elkobbah, Cairo, Egypt. https://doi.org/10.1088/1757-899x/610/1/012080.

Zhang, D., & Lei, J. (2011). Kinematic analysis of a novel 3-DOF actuation redundant parallel manipulator using artificial intelligence approach. *Robotics and Computer-Integrated Manufacturing*, *27*(1), 157-163. https://doi.org/10.1016/j.rcim.2010.07.003.

Zhang, H., Fang, H., Zhang, D., Luo, X., & Zou, Q. (2019). Forward kinematics and workspace determination of a novel redundantly actuated parallel manipulator. *International Journal of Aerospace Engineering*, *2019*, 1-14. 4769174. https://doi.org/10.1155/2019/4769174.