Solving forward and inverse kinematics problem for a robot arm (2DOF) using Fuzzy Neural Petri Net (FNPN)

Wael Hussein Zayer¹, Zahraa Ali Maeedi², Abdul Jabber Fathel Ali²
¹Department of Electromechanical, Southern Technical University, Iraq
²Department of Electrical Engineering, Wasit University, Iraq

Abstract.- At a time the robots become an important part of our lives by carrying them many tasks that require strength and accuracy, controlling these robots and giving them high accuracy and performance has become the focus of attention of workers in this field. In this research, one of the problems of controlling the robot arm is represented, where the problems of the forward and inverse kinematics have been addressed and find solutions to it using the artificial intelligence algorithm fuzzy Neural Petri net. The Matlab program was used to simulate the forward and reverse kinematics equations and then use the simulation results as data (inputs, outputs) for the proposed algorithm and run the program to update weights and give the required results depending on the ratio mean square error given.

Keywords: Robotic arm control, Forward kinematic, inverse kinematics, FNPN controller.

1. Introduction
Nowadays, the interest in the robot and its applications is expanding quickly are reaching over all parts, Where it used in a wide range of applications, from manufacturing to surgery and handling of hazardous materials. The robot arm is mechanical device consists of links and joints that give strength and solidness [1], as it has a large of applications it should have effective control. Robot arm control is a problem in controlling the mechanical system and not a problem in controlling the individual motors in the robot arm. Motion control methods of the robotics have always presented an extensive research area for the engineers and researchers in the field of designing control system due to the developments in using intelligent control methods. A broad number of control techniques is used like model-oriented control and used to control the positions of robots, like PID controller, Fuzzy logic control, and Neural Networks. These techniques required a specific mathematical model or ideal control model, which is very complexing. In the opposite, UN model-oriented control doesn’t require accurate information about parameters, like mechanical parts or the operating parts of the robot. This makes the design process simple [2]. One of these methods is using artificial intelligence tools in controlling the kinematics problem of the arm.

2. Objective
The main objective of this research is to find a solution to the problem of the kinematics of the forward and reverse kinematics by using one of the algorithms of artificial intelligence, using fuzzy neural Petri net algorithm that combined the properties of Petri net in its simple representation and learning procedure of the neural network to find the end-effector position in the forward kinematic and finding the joint angles in the inverse kinematic.
3. Forward Kinematics of 2DOF Robot Manipulator

Kinematics is the studying of movement concerning a reference framework without consideration of forces or different parameters that affect the movement. The analytic of the robot arm's spatial movement as time function is the fundamental worry of the kinematics, especially the connection between the position and direction of the arm's with the estimations of the joints' directions. A robot arm, as referenced previously, comprises of "inflexible bodies" called links associated by joints. The links and joints of the controller structure a kinematics chain which is open toward one side and associated with the ground of the other. The end-effector, or hand, or gripper, is associated with the free end, the arm's extreme, and the control goal of the robot framework is to situate the end-effector at a specific area [3]. The problem of the forward kinematics is to decide the position and direction of the arm's extreme. Using a transformation matrix T that relates the position and direction of the arm's extreme with the coordinates of joints [3]. The forward kinematics of the simple 2R planar robot manipulator shown in Figure 1. Position vector of end effector point P equal to:

\[ P_x = l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \]  
\[ P_y = l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \]  

\[ D = \frac{p x^2 + p y^2 - l_1^2 - l_2^2}{2 l_1 l_2} \]  

\[ \theta_2 = \tan^{-1}\left( \frac{\sqrt{1-D^2}}{D} \right) \]  
\[ \theta_1 = \tan^{-1}\left( \frac{px}{py} \right) - \tan^{-1}\left( \frac{l_2 \sin(\theta_2)}{l_1 + l_2 \cos(\theta_2)} \right) \]  

4. Inverse Kinematics of 2DOF Robot Manipulator

The inverse kinematics is the opposite process to the forward kinematics whereby given the desired position for the arm and then found the joint angles that gives these locations. The problem here there is more than one solution to find these locations different angles at a different time can give the same position. The inverse kinematic equations [5] are represented in the following:

5. Dynamics of 2DOF

Dynamics is the study of movement. It portrays why and how a movement happens when forces are applied to huge bodies. The dynamic equations of a robot arm are derived using the Lagrange-Euler
formulation to explain the problems involved in dynamic modeling [4]. The dynamic equations of two degrees of freedom arm of Figure 1 in terms of $\theta_1$ and $\theta_2$ are:

$$\tau_1 = \left(\frac{m_1 l_1^2}{3} + \frac{m_2 l_2^2}{3} + m_2 l_1 l_2 c \theta_2\right) \dot{\theta}_1 + \left(\frac{m_2 l_2^2}{3} + \frac{m_2 l_1 l_2}{2} c \theta_2\right) \dot{\theta}_2 - \frac{m_2 l_1 l_2}{2} s \theta_2 \dot{\theta}_2^2$$

$$m_2 l_1 s \theta_2 \dot{\theta}_1 \dot{\theta}_2 + \frac{m_1 m_l_1}{2} c \theta_1 + \frac{m_2 l_2}{2} c \theta_1 + m_2 g l_1 c \theta_1$$

$$\tau_2 = \left(\frac{m_2 l_2^2}{3} + \frac{m_2 l_1 l_2}{2} c \theta_2\right) \dot{\theta}_1 + \frac{m_2 l_2^2}{3} \ddot{\theta}_2 + \frac{m_2 l_2^2}{2} s \theta_2 \dot{\theta}_1^2 + \frac{m_2 g l_2}{2} c \theta_1$$

Where $l_1, l_2$ are the link lengths, $m_1, m_2$ are the masses of each link and are the torque for each link and letters $s$ and $c$ are representing the functions sin and cos. For the simulation modeling the parameter values taken for (2DOF) robot arm presented in table 1 with initial values equal to zero for $\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2$. Figure 2 represents the simulation dynamic model of (2DOF) robot arm, while Figure 3 and Figure 4 represented the simulation model of forward and inverse kinematics.

### Table 1. Parameter values for the dynamic model.

| links | $\tau$ | $m$ | $l$ | $g$ |
|-------|--------|-----|-----|-----|
| 1     | 3N.m   | 1Kg | 1m  | 9.81 m/s$^2$ |
| 2     | 1N.m   | 1 Kg| 1m  | 9.81 m/s$^2$ |

Figure 2. The simulation dynamic model of (2DOF) robot arm.
6. Fuzzy Neural Petri Net (FNPN)
The fuzzy neural Petri net used the stricture of the Petri net and the learning procedure of neural network whereby fuzzification the input place and using the back-propagation algorithm to update weights. Figure 5 shows the structure of the fuzzy neural Petri net. The Petri net consists of input places, transition and output places and arc that connected these places by each other and separated by the transitions. In fuzzy neural Petri net, the input places represented the input layer and the transitions represented the hidden layer also the output places represented the output layer [6]. The FNPN equations are illustrated as follows:
Figure 5. Fuzzy Neural Petri Net structure.

$$P_j = f(\text{Input}(j))$$ (8)

Where $f$ is a Gaussian membership function

$$\mu_j = \exp\left(-\frac{(x_j - c_{ij})^2}{2s_{ij}^2}\right)$$ (9)

Where $c_{ij}, s_{ij}$ are the mean and standard deviation of the Gaussian function.

$$Z_i = \prod_{j=1}^{n} W_{ij} S(\frac{r_{ij}}{P_j})\quad j = 1,2,\ldots,n; i = 1,2,\ldots,\text{hidden}$$ (10)

$$r_{ij} \rightarrow P_j = \begin{cases} \frac{P_j}{r_{ij}}, & \text{if } r_{ij} > P_j \\ 1, & \text{otherwise} \end{cases}$$ (11)

$$Z_i = \prod_{j=1}^{n} W_{ij} V_{kj} \quad \frac{P_j}{r_{ij}}, \text{if } r_{ij} > P_j$$ (12)

Where $W_{ij}$ is the weight between the $j$-th input places and the $i$-th transition and $r_{ij}$ is the threshold level associated with the level of marking of the $j$-th input places and the $i$-th transition and $Z_i$ is the activation level of the $i$-th transition.

$$Y_k = f \sum_{i=1}^{\text{NoofTransition}} V_{ki}(Z_i), \quad j = 1,2,\ldots,n$$ (13)

Where $Y_k$ is the marking level of the $k$-th output place and equal to the weighted sum of the activation level $Z_i$ and the associated connections $V_{ki}$. $f$ is a nonlinear function between $[0,1].$ The update of the parameters is derived using the back propagation algorithm where the summation of the squared error is used to optimize the weights and threshold[6].

$$E = \frac{1}{2} \sum_{k=1}^{m} (t_k - Y_k)^2$$ (14)

Where $t_k$ is the $k$-th target, $Y_k$ is the $k$-th output. The nonlinear function that used is a standard sigmoid described as:

$$Y_k = \frac{1}{1+\exp(-\sum_{i} v_{ki})}$$ (15)

According to the gradient method, the parameters are update

$$\text{param(iter + 1)} = \text{param(iter)} - \alpha \nabla_{\text{param}} E$$ (16)

Where $\alpha$ is the learning rate coefficient and $\nabla_{\text{param}} E$ is a gradient of the performance index $E$ concerning the network parameters, and iter is the iteration counter. The parameters update is deriving as following:
\[ \Delta V_{kl} = -\zeta \frac{\partial E}{\partial V_{kl}} \quad (17) \]
\[ \frac{\partial E}{\partial V_{kl}} = (\frac{\partial E}{\partial Y_k})(\frac{\partial Y_k}{\partial V_{kl}}) \quad (18) \]
\[ \frac{\partial E}{\partial Y_k} = -(t_k - Y_k) \quad (19) \]
\[ (\frac{\partial Y_k}{\partial V_{kl}}) = Y_k(1 - Y_k)Z_i = (-\exp(-Z_i V_{kl})Z_i)/(1 + \exp (-Z_i V_{kt})) \quad (20) \]
\[ \Delta V_{kl} = -\zeta [\exp(-Z_i V_{kl})Z_i(t_k - Y_k)/(1 + \exp (-Z_i V_{kt}))] \quad (21) \]
\[ V_{kl}(\text{iter} + 1) = V_{kl}(\text{iter}) - \alpha \Delta V_{kl} \quad (22) \]
For \( k = 1, 2, \ldots, m; \ i = 1, 2, \ldots, \noof\text{hidden transitions} \)
\[ \Delta r_{ij} = -\zeta \frac{\partial E}{\partial r_{ij}} \quad (23) \]
\[ \frac{\partial E}{\partial r_{ij}} = (\frac{\partial E}{\partial Y_k})(\frac{\partial Y_k}{\partial Z_i})(\frac{\partial Z_i}{\partial r_{ij}}) \quad (24) \]
\[ \frac{\partial E}{\partial Y_k} = -(t_k - Y_k) \quad (25) \]
\[ \frac{\partial Y_k}{\partial Z_i} = Y_k(1 - Y_k)V_{kl} \quad (26) \]
\[ \frac{\partial Z_i}{\partial r_{ij}} = A[\frac{\partial}{\partial r_{ij}}(W_{ij} + (r_{ij} \rightarrow P_j) - W_{ij}(r_{ij} \rightarrow P_j))] = A(1 - W_{ij}) \frac{\partial}{\partial r_{ij}}(r_{ij} \rightarrow P_j) \quad (27) \]
Where
\[ A = \prod_{i=1}^{n} [W_{il}S(r_{li} \rightarrow P_l)] \]
\[ \frac{\partial}{\partial r_{ij}}(r_{ij} \rightarrow P_j) = \frac{\partial}{\partial r_{ij}} \begin{cases} P_j, & \text{if } r_{ij} > P_j \\ r_{ij}, & \text{otherwise} \end{cases} \]
\[ r_{ij}(\text{iter} + 1) = r_{ij}(\text{iter}) - \alpha \Delta r_{ij} \quad (29) \]
For \( j = 1, 2, \ldots, n; \ i = 1, 2, \ldots, \noof\text{hidden transitions} \)
\[ \Delta W_{ij} = -\zeta \frac{\partial E}{\partial W_{ij}} \quad (30) \]
\[ \frac{\partial E}{\partial W_{ij}} = (\frac{\partial E}{\partial Y_k})(\frac{\partial Y_k}{\partial Z_i})(\frac{\partial Z_i}{\partial W_{ij}}) \quad (31) \]
For \( k = 1, 2, \ldots, m; \ i = 1, 2, \ldots, \noof\text{hidden transitions}; j = 1, 2, \ldots, n \)
\[ \frac{\partial E}{\partial Y_k} = -(t_k - Y_k) \quad (32) \]
\[ \frac{\partial Y_k}{\partial Z_i} = Y_k(1 - Y_k)V_{kl} = (-\exp(-Z_i V_{kl})V_{kl})/(1 + \exp (-Z_i V_{kt})) \quad (33) \]
\[ \frac{\partial Z_i}{\partial W_{ij}} = A[\frac{\partial}{\partial W_{ij}}(W_{ij} + (r_{ij} \rightarrow P_j) - W_{ij}(r_{ij} \rightarrow P_j))] \]
\[ = A(1 - (r_{ij} \rightarrow P_j)) \quad (34) \]
Where
\[ A = \prod_{i=1}^{n} [W_{il}S(r_{li} \rightarrow P_l)] \]
\[ r_{ij} \rightarrow P_j = \begin{cases} P_j, & \text{if } r_{ij} > P_j \\ r_{ij}, & \text{otherwise} \end{cases} \]
\[ \Delta W_{ij} = -\zeta (t_k - Y_k)(\exp(-Z_i V_{kl})V_{kl})/(1 + \exp (-Z_i V_{kt})) \prod_{i=1}^{n} [W_{il}S(r_{li} \rightarrow P_l)](1 - \begin{cases} P_j, & \text{if } r_{ij} > P_j \\ r_{ij}, & \text{otherwise} \end{cases} \]
\[ W_{ij}(\text{iter} + 1) = W_{ij}(\text{iter}) - \alpha \Delta W_{ij} \quad (36) \]
7. RESULT AND DISCUSSION
To analyze the movement of the robotic arm and solve the problem of forward and reverse kinematics, an artificial intelligence algorithm is used combined the feature of Neural network, Fuzzy logic and Petri Net where Petri Net is a useful choice for the discrete events system where the system represents no matter who much it complex in a simple graphical based on the logical expression to describe the system problem, where based on the conditions that put the event or process may happening or not happening, and the ability of fuzzy logic to make decisions and the learning ability of the neural network represented by back-propagation algorithm all together in an algorithm known FNPN. The Matlab program was used to perform simulations as follows

➢ First, data generation, whereby using the simulation library to create a model that simulates the mathematical equations of the dynamic model and the mathematical equations of the forward and backward kinematics and save the simulation results for the three models and use them as inputs and outputs for the proposed program then used it to compare them with the results of the proposed program as following, from the dynamic model the value of joints angles is taken where in our case we have two angles and these angles are used as input (subsystem) to the kinematics forward model to get the desired position, Figure 6 shows the simulation result of the theta 1 and theta 2 which is the input to the forward model. In the inverse kinematics, the position which gets in the forward model is used as input (subsystem) to the inverse model to get the joints angles that give this position which may not necessarily be the same angles that used in the beginning and save the result from the workspace as an excel sheet.

![Figure 6. Joint angles Theta 1 and Theta 2 from simulation against time.](image)

➢ Secondly, calculating the desired values represented by the desired position in the case of the forward kinematic, where the simulation results are entered as follows: The inputs to the program are the angles that give this position and the outputs to the program are the desired position. In the case of reverse kinematics, the inputs to the program are the position and the outputs are the angles that gave this position. In both cases, five hidden layers are used and the value of the mean square error that used was 0.00001. Finally, the result of the program was compared with the result from the simulation, Figures 7 and 8 show the comparison of the results and from them, one can observe the values were close to the desired values for both simulated cases of (forward and inverse) kinematics.
➢ **Thirdly** the Tables 2 and 3 show the results from the proposed program and compared them with the result of the simulation for two cases (forward and inverse) kinematics wherein table 2 theta1 and theta 2 are the inputs and px and py are the outputs desired for the forward kinematic and the predicted px and py from FNPN algorithm the opposite process is happening in Table 3 where the px and py are the inputs and the theta1 and theta2 are the desired outputs for the inverse kinematic and theta1 and theta2 predicted from FNPN algorithm.

![Graph 1](image1)

**Figure 7.** Compared between desired values and FNPN predicted in the forward kinematics.

![Graph 2](image2)

**Figure 8.** Comparison between desired values and FNPN predicted in the inverse kinematics.

| Table 2 | Table 3 |
|--------|--------|
| Comparison between the desired values of Figure 7. | Comparison between desired values & FNPN predicted in the inverse kinematics FNPN predicted in the forward kinematics |
## From simulation (desired) From FNPN (predicted)

| S.NO | px | py | input | target | output | error | th1     | th2     | th1     | th2     | error th1 | error th2 | The error |
|------|----|----|-------|--------|--------|-------|---------|---------|---------|---------|-----------|-----------|-----------|
| 1    | 2  | 0  | 0.7496 | -0.0939 | 0.7496 | -0.0939 |
| 2    | 2  | -3.77E-10 | -1.08E-08 | -0.547 | 0.547 | 0.476 |
| 3    | 2  | -3.66E-01 | -2.20E-06 | 0.351 | 2.57E-04 | -0.25083 | -0.7275207 |
| 4    | 2  | -9.06E-06 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 5    | 2  | -4.98E-02 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 6    | 2  | 0.993516 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 7    | 2  | 0.81537 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 8    | 2  | 0.190775 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 9    | 2  | 0.856039 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

## From input (target) From output (target) From RMSE(predicted)

| S.NO | px | py | input | target | output | error | th1     | th2     | th1     | th2     | error th1 | error th2 | The error |
|------|----|----|-------|--------|--------|-------|---------|---------|---------|---------|-----------|-----------|-----------|
| 1    | 2  | 0  | 0.7496 | -0.0939 | 0.7496 | -0.0939 |
| 2    | 2  | -3.77E-10 | -1.08E-08 | -0.547 | 0.547 | 0.476 |
| 3    | 2  | -3.66E-01 | -2.20E-06 | 0.351 | 2.57E-04 | -0.25083 | -0.7275207 |
| 4    | 2  | -9.06E-06 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 5    | 2  | -4.98E-02 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 6    | 2  | 0.993516 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 7    | 2  | 0.81537 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 8    | 2  | 0.190775 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 9    | 2  | 0.856039 | 0.00000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

### Notes
- ISC-AET 2020
- Journal of Physics: Conference Series 1773 (2021) 012009
- doi:10.1088/1742-6596/1773/1/012009
- IOP Publishing

---

### Table Data

- **From simulation (desired)**
  - **From FNPN (predicted)**
  - **From input (target)**
  - **From output (target)**
  - **From RMSE (predicted)**

### Key
- **S.NO**: Sequence Number
- **px**, **py**: Position Coefficients
- **input**: Input Values
- **target**: Target Values
- **output**: Output Values
- **error**: Error Values
- **th1**, **th2**: Angle Coefficients
- **error th1**, **error th2**: Error Coefficients
- **The error**: Total Error

### Formula
- Various mathematical formulas and equations are used to calculate the values in the tables.

---

### Additional Information
- ISC-AET 2020 Conference Proceedings
- Journal of Physics: Conference Series 1773 (2021)
- DOI: 10.1088/1742-6596/1773/1/012009
- IOP Publishing
8. CONCLUSIONS
Dependence on mathematical models in calculating and simulating forward and inverse kinematics may not be within the required level as some of these models fail correct simulation of the movement of the robot arm. Therefore, the fuzzy neural Petri net algorithm was used, as it relies only on input and output data to determine the structure of the model. The network can always be updated and get better results when obtaining new data, as well as depending on the error rate to reach the required accuracy. The use of this algorithm to solve the problem of the forward and inverse kinematics gave good results and a similar approach to the results taken from simulations.

REFERENCES:
[1] Okubanjo, A. A., Oyetola, O. K., Osifeko, M. O., Olaluwoye, O. O., & Alao, P. O 2017 Modeling of 2-DOF Robot Arm and Control. FUTO JNLS, vol3, pp. 80-92
[2] Abbas, Z. 2018 Motion Control of Robotic Arm Manipulator Using PID and Sliding Mode Technique (Doctoral dissertation, CAPITAL UNIVERSITY).
[3] Rosales, E. M., &Gan, Q. 2002 CSM-413 Forward and Inverse Kinematics Models for a 5-dof Pioneer 2 Robot Arm.
[4] AYTEN, K. K., &Dumlu, A. 2018 Real-Time Implementation of Sliding Mode Control Technique for Two-DOF Industrial Robotic Arm. IğdırÜniversitesi Fen BilimleriEnstitüsü Dergisi, 8(4), 77-85.
[5] Jazar, R. N 2010 Theory of applied robotics: kinematics, dynamics, and control. Springer Science & Business Media.
[6] Ali, A. A., &Mosa, G. S 2017 NEURAL FUZZY PETRI NET BASED ARABIC PHONEME CLASSIFIER WITH MFCC FEATURE EXTRACTION. QalaaiZanist Journal, vol 2, 375-389.