Optimizing neural networks for pneumatic muscle actuator system identification: a cooperative coevolutionary approach

Nguyen Ngoc Son[1]*, Hoang Duc Quy[1], Tran Minh Chinh[1], Luu The Vinh[1]

[1]*Faculty of Electronics Technology, Industrial University of Ho Chi Minh City, Viet Nam,*
[*]*Corresponding author E-mail: nguyenngocson@iuh.edu.vn*
DOI: https://doi.org/10.64032/mca.v29i3.334

Abstract

This paper presents a cooperative coevolutionary optimization algorithm to overcome issues in gradient descent-based neural network, such as getting stuck in local minimum and slow convergence. The proposed method combines JAYA and a modified differential evolution (DE) techniques to optimize neural network weights. It works by splitting the population into two subpopulations, each focusing on optimizing different aspects of the network weights. The method's effectiveness is tested on two benchmark nonlinear dynamical systems and compared with existing methods. Results show that the neural network optimized by this approach achieves high accuracy and robustness. Finally, the practical applicability of this method is demonstrated by modeling the pneumatic muscle actuator (PMA) system using experimental data, where the PMA system is made up of Festo's MAS-10 N220 pneumatic artificial muscles and controlled with a DAQ NI 6221 card.

Keywords*: Coevolutionary algorithm; Optimized Neural network; Nonlinear system identification; Jaya algorithm; Differential evolution.*

Abbreviations

DE	Differential evolution
PMA	pneumatic muscle actuator
MSE	Mean square error
CoDEJA	Coevolutionary based DE and JAYA
CEAs	Coevolutionary algorithms

1. Introduction

Identifying nonlinear black-box models of dynamical systems involves finding system characteristics based on measured input and output signals by minimizing the error norm between measured data and model output. Recent research has explored various approaches, including feedforward neural networks [1], LSTM neural networks [2], neuro-fuzzy systems [3], and other hybrid intelligent methods [4]. Among these, optimizing weights and biases remains a critical task in neural networks. Over the past decade, gradient-based methods have been widely studied but still face limitations such as local minima and slow convergence, which degrade system identification performance.

To address these issues, evolutionary algorithms (EAs) have been explored for optimizing neural network weights and biases. For instance, Xiong et al. [5] applied differential evolution to optimize the number of hidden neurons and the regularization factor of hierarchical extreme learning machines for fault section diagnosis in large-scale power systems. Liu et al. [6] used differential evolution to train neural network-based control parameters for power electronic circuits. Nguyen et al. [7] proposed a modified differential evolution algorithm for optimizing neural networks in time series forecasting. Min et al. [8] introduced a neural network-optimized genetic algorithm for energy management strategies in hybrid electric vehicles under start-stop conditions. Nguyen et al. [9] applied the Jaya algorithm to optimize neural network weights for uncertain nonlinear system identification. Mao et al. [10] developed a biogeography-based optimization algorithm for multilayer perceptron networks in nonlinear system identification. However, EAs often struggle to find global solutions for large-scale problems within limited computational time.

Coevolutionary algorithms (CEAs) have been developed to address complex problems through a divide-and-conquer approach. Unlike EAs, which evolve a single population with one leading individual, CEAs evolve multiple populations simultaneously, each with multiple leading individuals. This enhances the balance between exploration and exploitation and helps prevent premature convergence.

For example, a co-evolutionary bird optimization algorithm combined with an online policy gradient method was introduced to solve continuous real-parameter optimization problems [11]. A knowledge-driven coevolutionary algorithm combined differential evolution (DE) and estimation of distribution algorithm through a cross-regional interactive learning mechanism to improve performance [12]. A modified competitive swarm optimizer (CSO) using a three-phase coevolutionary strategy was also proposed to enhance CSO performance [13].

Recently, CEAs have been applying to optimize neural network weights. Triumala et al. [14] introduced a multi-population cooperative neuro-evolution approach, while Gong et al. [15] combined cooperative coevolution with backpropagation. Wei et al. [16] developed a coevolutionary particle swarm optimization for manipulator control, and Liang et al. [17] proposed a cooperative coevolutionary JADE to evolve both topology and weights. Ayala et al. [18] applied differential evolution and harmony search for RBFNN optimization, and Xue et al. [19] combined differential evolution with Adam-based gradient descent for feedforward networks.

Jaya and differential evolution algorithms have been widely applied in areas such as scheduling [20], wireless sensor networks [21], and parameter estimation [22]. The Jaya

Received: 29 April 2025; Revised: 07 July 2025; Accepted: 15 July 2025.

algorithm offers advantages such as simplicity, efficiency, and no control parameters, making it suitable for complex problems. Differential evolution is also favored for its fast convergence and the need for only three control parameters.

Based on the above analysis, this paper presents a cooperative coevolutionary algorithm to optimize neural networks (named CoDEJA-NN), aiming to overcome local minimum and slow convergence issues in gradient-based methods. The proposed algorithm combines differential evolution and the Jaya algorithm by dividing the initial population into two subpopulations, each responsible for optimizing different aspects of the neural network. The method is evaluated through three nonlinear dynamical system identification tasks and compared against existing algorithms to verify its effectiveness.

The remainder of the paper is organized as follows. Section 2.1 introduces the PMA system configuration. Section 2.2 describes the proposed CoDEJA-NN model for system identification. Section 2.3 presents performance evaluation and discussion. Section 2.4 applies the proposed method to PMA system modeling. Section 3 concludes the paper.

2. Main body

2.1 Pneumatic muscle actuator (PMA) setup

A block diagram of the experimental pneumatic muscle actuator (PMA) architecture is shown in Fig.1, and the photo of the PMA system as Fig.2. The parameters of the PMA system are described in Table 1 and Table 2.

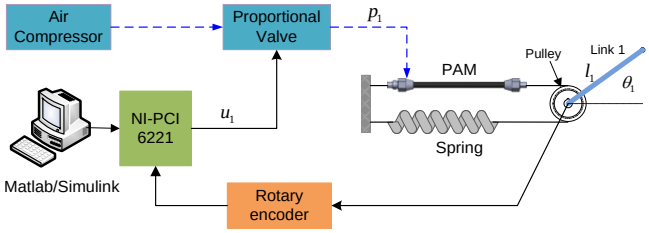

Figure 1: Diagram of the experimental PMA system

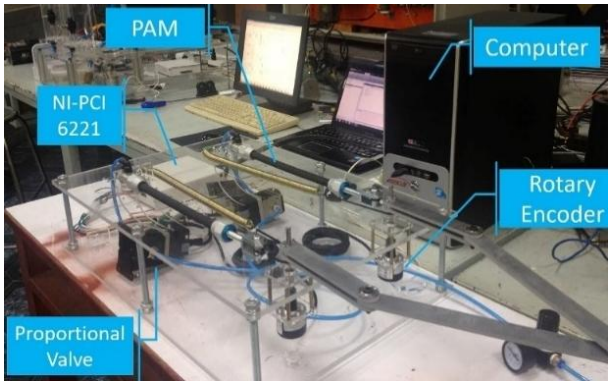

Figure 2: Diagram of the experimental PMA system

The PMA system is a 1-DOF motion platform actuated by PMAs, equipped with a coil spring for passive force compensation and a rotary encoder for angle measurement. Airflow is regulated by proportional valves (EV2500-008), and a National Instruments PCI 6221 card handles real-time data acquisition and control. Control algorithms are implemented in Matlab using the Real-Time Windows Target Toolbox with a sampling time of Ts = 0.01 s.

Table 1: Mechanical parameters of PMA system

Parameter	Notation	Value	Unit
Length link 1	l_1	15	*cm*
Distance between two rotary joints	d	20	*cm*
Diameter of pulley	δ	3	*cm*
Stiffness of spring	k	500	*N/m*

Table 2: Experimental device parameters of PMA system

No	Devices	Parameters
1	PMA: Festo MAS-10 N220	- Nominal length: 22 cm - Internal diameter: 10 mm - Operating pressure: 0-8 bar
2	Encoder: Autonics E40S6-3600	- Rotary encoder. - Resolution: 3600 pulses per revolution. - Power supply: 24 V ± 10%
3	DAQ card: NI PCI 6221	- Two 16-bit analog outputs (833 KS/s) - 24 digital I/O. - 16 Analog Inputs, 16-Bit, 250 KS/s
4	Proportional valse EV2500-008	- Power: DC 24 V (± 10% tolerance) - Control Pressure Range: 0 to 0.49 Mpa. - Input Signal: 0-10 V DC - Resolution: 0.5% of Full Scale (F.S.)

2.2 The proposed cooperative coevolutionary algorithms-based neural networks

2.2.1 *Neural networks training problem*

The structure of neural networks includes n inputs, q hidden neurons, and m outputs. In which, $\boldsymbol{X} = [x_{bias}, x_1, \dots, x_n]^T$ and $\boldsymbol{Y} = [y_1, .., y_m]^T$ are the input and output of neural networks, respectively. $\boldsymbol{V} = [v_{10}\, v_{11} \dots v_{1n}, \;\; v_{q0}, \dots \;\; v_{qn}]$ denotes the weights of the input layer and bias input. $\boldsymbol{W} = [w_{10}\, w_{11} \dots w_{1q}, \;\; w_{m0}, \dots \;\; w_{mq}]$ denotes the weights of hidden layer and bias. *f*(.) and *F*(.) are the active functions of the hidden and output layers.

The output of neural networks is defined as follows,

$$\hat{y}_l(k,\theta) = F_l\left(\sum_{i=1}^{q} w_{1i} f_i\left(\sum_{j=1}^{n} v_{ij} x_j(k) + v_{i0}\right) + w_{10}\right) \quad (1)$$

Where, $l \in [1, m]$. The weights are denoted as $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_D]$ which D is the number weights of hidden and output layers. The dimension of D is calculated as,

$$D = q(n+1) + m(q+1) \quad (2)$$

To optimize the weights and bias of neural networks $\boldsymbol{\theta}$ by minimizing the criteria mean square errors (MSE) which is defined as,

$$J(\theta, Z^N) = \frac{1}{2N}\sum_{k=1}^{N}\sum_{l=1}^{m}[y_l(k) - \hat{y}_l(k|\theta)]^T[y_l(k) - \hat{y}_l(k|\theta)] \quad (3)$$

Where, Z^N is the training data set which is defined by $Z^N = \{[\boldsymbol{X}(k), \boldsymbol{Y}(k)] | k = 1, \dots, N\}$. N is the number of sample data.

2.2.2 *Jaya algorithm-based neural network*

The Jaya algorithm was developed by R. Rao in 2015 [23] as a variant of the teaching-learning-based optimization method.

In Jaya algorithm, each search agent is called a particle $x_{j,i}$. Each particle moves toward the best solution $x_{j,\text{best}}$ and

away from the worst solution $x_{j,\text{worst}}$ in the search space. The position of the particle $x'_{j,i}$ is updated as follows,

$$x'_{j,i} = x_{j,i} + r_{1,j} \times \left(x_{j,\text{best}} - |x_{j,i}|\right) - r_{2,j} \times \left(x_{j,\text{worst}} - |x_{j,i}|\right) \quad (4)$$

Where $r_{1,j}$ and $r_{2,j}$ are two uniform functions which generate a random number within [0,1]. $|x_{j,i}|$ is the absolute value of the particle $x_{j,i}$.

2.2.3 *Differential evolution-based neural networks*

Differential Evolution (DE), proposed by R. Storn and K. Price in 1995 [24]. The DE process consists of four main steps as follows:

Step 1. Initialize a population of candidate solutions. A considered individual candidate is expressed as

$$\boldsymbol{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}] \quad (5)$$

Where G is the number of generations and $j = 1, 2, \ldots, D.$, $i = 1, 2, \ldots, NP$. NP is the size of the population and D is the size of the dimension variables of the optimization problem.

Step 2. Mutation. For each candidate solution, generate a donor solution by combining it with two other candidates in the population. A mutant vector $\boldsymbol{V}_{i,G} = [v_{1,i,G}, v_{2,i,G}, \ldots, v_{D,i,G}]$ is generated as Table 3. Where r_1^i, r_2^i, r_3^i and r_1^4 are integers randomly selected from [1;NP] such that $r_1^i \neq r_2^i \neq r_3^i \neq r_4^i \neq i$. The mutant coefficient F is selected from [0,1].

Table 3: Mutation scheme

DE/best/1	$v_{i,G} = x_{best,G} + F\left(x_{r_2^i,G} - x_{r_3^i,G}\right)$
DE/rand/1	$v_{i,G} = x_{r_1^i,G} + F\left(x_{r_2^i,G} - x_{r_3^i,G}\right)$
DE/best/2	$v_{i,G} = x_{best,G} + F\left(x_{r_1^i,G} + x_{r_2^i,G} - x_{r_3^i,G} - x_{r_4^i,G}\right)$
DE/rand/2	$v_{i,G} = x_{r_5^i,G} + F\left(x_{r_1^i,G} + x_{r_2^i,G} - x_{r_3^i,G} - x_{r_4^i,G}\right)$

Step 3. Crossover. To improve the variety of the population, the trial vector of *binôme* crossover $\boldsymbol{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, \ldots, u_{D,i,G}]$ can be defined as,

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & if\left(rd_{j,i}[0,1] \leq CR\right) \\ x_{j,i,G} & otherwise \end{cases} \quad (6)$$

where $rd_{j,i}[0,1]$ is a uniform generated in the range [0,1] and the crossover coefficient CR is selected from [0,1].

Step 4. Selection. To compare the trial solution $\boldsymbol{U}_{i,G}$ to the original solution $\boldsymbol{X}_{i,G}$ and replace $\boldsymbol{X}_{i,G}$ with the $\boldsymbol{U}_{i,G}$ if their fitness is better.

2.2.4 *Cooperative coevolutionary algorithms*

In this section, a cooperative coevolutionary algorithm (CoDEJA) is proposed by combining a modified differential evolution (MDE) and the Jaya algorithm. The population is divided into two subpopulations, each responsible for optimizing the weights and biases of the neural network. The flowchart of the CoDEJA-based neural network optimization is shown in Fig. 3.

In each subpopulation, the global solution is found by using the MDE algorithm and Jaya algorithm. The first subpopulation utilizes a modified differential evolution algorithm with self-adaptive mutation operators, including "rand/1" and "best/1". This allows for a balance between exploration and exploitation, enhancing the chances of finding the global solution. The detailed steps of the MDE algorithm are outlined in **Algorithm 1**. The second subpopulation leverages the Jaya algorithm to search for the global solution. The details of the Jaya algorithm are presented in **Algorithm 2**.

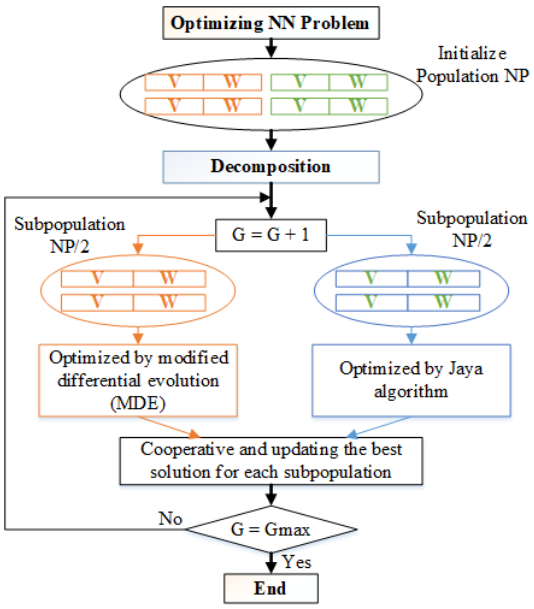

Figure 3: Flow chart of CoDEJA optimized neural networks

Algorithm 1: The pseudo-code of MDE

A. Input: NP/2, CR = 0.9, F = 0.5; the number weights of hidden and output layers D. The initial population $\boldsymbol{\theta}_{i,G} = [\theta_{1,i,G}, \theta_{2,i,G}, \ldots, \theta_{D,i,G}]$.
B. Output: The best solution
1: Evaluate the fitness for each individual in the subpopulation.
2: **for** i = 1 **to** *NP/2* **do**
3: j_{rand} = *randint*(1, *D*)
4: **for** j = 1 **to** *D* **do**
5: **if** rand[0,1] < *CR* or $j = j_{\text{rand}}$ **then**
6: **if** rand > *0.5* **then** *% using "rand/1"*
7: Select randomly $r_1 \neq r_2 \neq r_3 \neq i$, $\forall i \in \{1,2,\ldots,NP\}$ $u_{i,j} = x_{r1,j} + F \times (x_{r2,j} - x_{r3,j})$
8: **else**
9: Select randomly r1 ≠ r2 ≠best ≠ i, ∀i∈{1,2,…,NP} $u_{i,j} = x_{best,j} + F \times (x_{r1,j} - x_{r2,j})$
10: **end if**
11: **else**
12: $u_{i,j} = x_{i,j}$
13: **end if**
14: **end for**
15: **if** $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$
16: $\mathbf{x}_i = \mathbf{u}_i$
17: **else**
18: $\mathbf{x}_i = \mathbf{x}_i$
19: **end if**
20: **end for**

After that, following independent optimization within each subpopulation, information exchange and updates are facilitated: The "best" individuals from each subpopulation are exchanged, allowing them to contribute to the search process in the other subpopulation. Based on the exchanged information, the selection of "best" individuals within each

subpopulation is updated for the next generation. This cooperative approach aims to leverage the strengths of both the MDE and Jaya algorithms, while promoting knowledge transfer between subpopulations. The details of the CoDEJA algorithm are presented in **Algorithm 3**.

Algorithm 2: The pseudo-code of JAYA

A. Input: NP/2, the number weights of hidden and output layers D. The initial population $\boldsymbol{\theta}_{i,G} = [\theta_{1,i,G}, \theta_{2,i,G}, \ldots, \theta_{D,i,G}]$.
B. Output: The best solution

Evaluate the fitness in the subpopulation to find the best and worst solution.
for i = 1 **to** *NP/2* **do**
  j_{rand} = *randint*(1, *D*)
  for j = 1 **to** *D* **do**
    $u'_{j,i} = x_{j,i} + r_{1,j} \times (x_{j,\text{best}} - |x_{j,i}|) - r_{2,j} \times (x_{j,\text{worst}} - |x_{j,i}|)$
  end for
  if $f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i)$
    $\mathbf{X}_i = \mathbf{U}_i$
  else
    $\mathbf{X}_i = \mathbf{X}_i$
  end if
end for

Algorithm 3: The pseudo-code of CoDEJA

A. Input: NP, the number weights of hidden and output layers D. The initial population $\boldsymbol{\theta}_{i,G} = [\theta_{1,i,G}, \theta_{2,i,G}, \ldots, \theta_{D,i,G}]$.
B. Output: The global solution

Divide the population into two subpopulations
while(*G is not reached*) **do**
  Call Algorithm 1.
  Call Algorithm 2.
  Exchanged the "best" individuals from each subpopulation
  Selection of the "best" individuals updated for the next generation
end while

2.3 Benchmark dynamic systems identification

2.3.1 *Expeirmental setup*

In this section, the benchmark mechanical dynamic systems used to test the performance of the proposed CoDEJA-NN are presented. The characteristics of the benchmark dynamic system are summarized in Table 4.

Case study 1. MR Damper. The magneto-rheological (MR) damper dataset in Fig.4 was created by Dr. Akira Sano et al. [25] which concluded the velocity input v (cm/s) of the damper and the output f (N) of damping force measurements.

Case study 2. Piezoelectric actuator. The piezoelectric actuators dataset is provided by Prof. Micky Rakotondrabe which is a library in MATLAB with the name "idPiezoElectricData.mat". The dataset of Piezoelectric actuators is described in Fig.5. In which, the input v [Volt] is the voltage and the output d [μ.m] is the displacement of the actuator.

Remark 1. All algorithms are tested by using MATLAB 2023b on an Intel(R) Core (TM) i5-8400 CPU with a speed of 2.80GHz and 12 GB of RAM. The control parameters are chosen using the trial-and-error method, as shown in Table 4.

Remark 2. The dataset of benchmark dynamic systems is normalized by using the Min-Max technique to the range [-1,1] before the system identification process.

$$z_i = 2\frac{x_i - x_{min}}{x_{max} - x_{min}} - 1 \quad (8)$$

In which, z, x, x_{min}, x_{max} are normalized, current, minimum, and the maximum values in the dataset, respectively.

Table 4: The characteristics of benchmark dynamic system

Dataset	Samples	Range of input	Range of output
MR Damper	3,499	[-15.4970, 14.7925]	[-79.2771, 85.8684]
Piezoelectric actuator	10,000	[-71.5378, 71.1544]	[-71.8729, 71.7462]

Table 5: Parameters used in identification

Algorithm	Coefficients	Note
General	Number of runs Population dimension NP Amount of generations Range of weight values	10 times 2D 10.000 [-1,1]
DE [24]	Mutant factor, F Crossover factor, CR	0.5 0.9
JAYA [23]	No control parameters	-

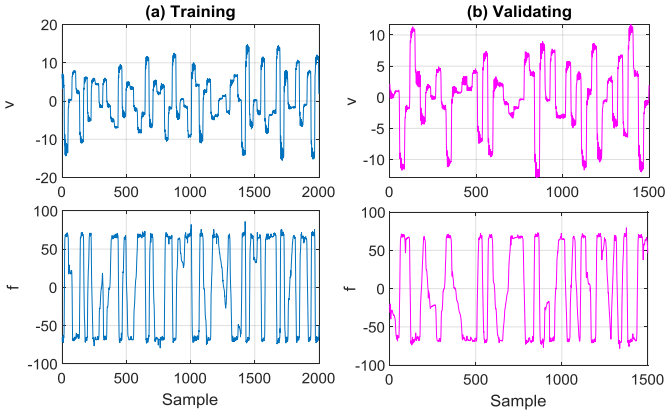

Figure 4: The dataset of MR Damper for identification

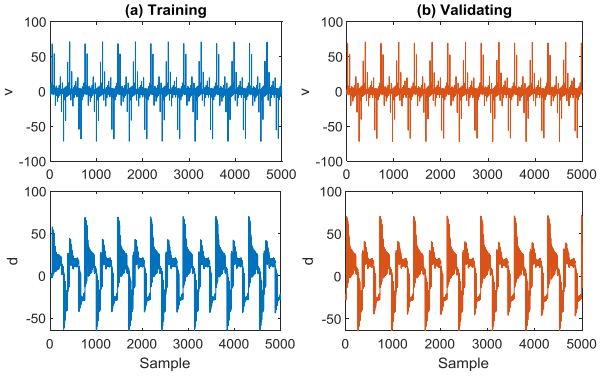

Figure 5: The dataset of Piezo for identification

2.3.2 *Results and disscussion*

Case study 1. MR Damper system identification

The input to the neural network is denoted by, $\boldsymbol{X} = [f(k-1), f(k-2), v(k-1), v(k-2), bias]^T$ and the output is denoted by $\boldsymbol{Y} = f(k)$. The total number of weights in the hidden and output layers is 43. The dataset shown in Fig. 4 is normalized to the range [-1, 1] before the system identification process.

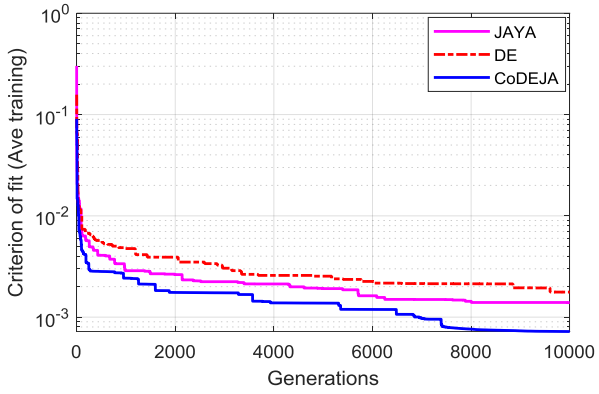

Figure 6: Criterion of fitness in the training process of MR Damper system

The identification results of MR Damper are described as follows. Fig.7 shows the convergence speed of DE, JAYA, and CoDEJA algorithms in the training process, while Fig.7 shows the performance identification of those algorithms in the validating process. Table 6 tabulates the criteria for performance in the training and validating process.

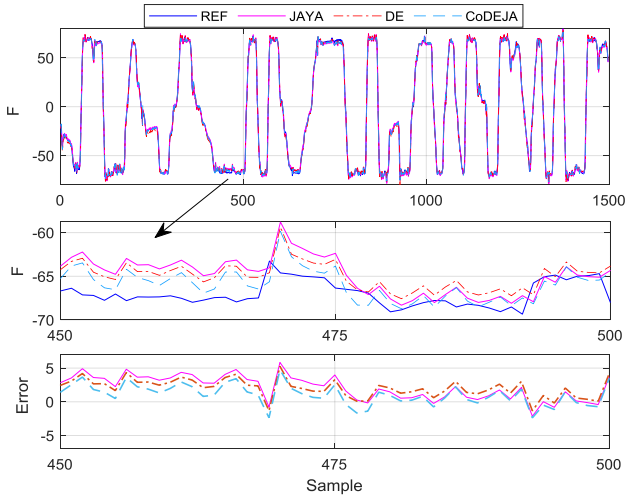

Figure 7: The performance identification in validating MR Damper system

Table 6: Performance identification of MR Damper system

Method	MSE					
	Training			Validation		
	Best	Worst	Average	Best	Worst	Average
CoDEJA	**4.6577**	**5.2429**	**4.9169**	**3.8408**	**4.3689**	**4.0872**
Jaya	7.6950	11.7246	9.4821	5.2692	6.1582	5.7681
DE	5.9106	18.9501	12.0524	4.7216	7.8881	6.3552

From Fig.6 and Fig.7, it can be observed that the convergence speed of CoDEJA is better than DE and JAYA. From Table 6, the MSE of CODEJA archives 4.9169 and 4.0872 in the training and validating process, while DE achieves 12.0524 and 6.3552, JAYA achieves 9.4821 and 5.7681.

Case study 2. Piezoelectric actuator

The neural model consists of six inputs, seven hidden neurons, and one output. The input to the neural network is denoted by, $\boldsymbol{X} = [d(k-1), d(k-2), d(k-3), v(k-1), v(k-2), v(k-3), bias]^T$ and the output is denoted by $\boldsymbol{Y} = d(k)$. The total number of weights in the hidden and output layers is 57. The dataset shown in Figure 6 is normalized to the range [-1, 1] before identification.

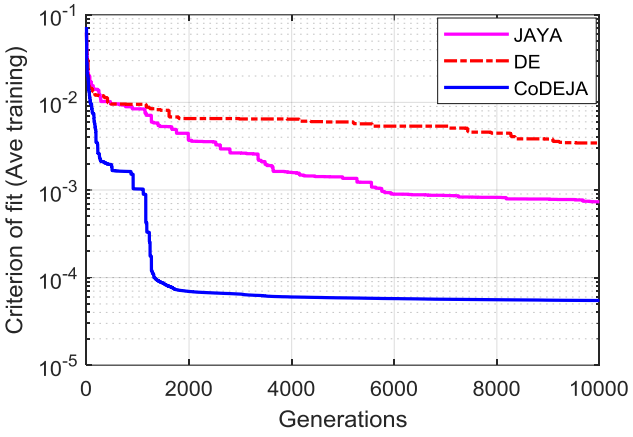

Figure 8: Criterion of fitness in training process of Piezo system

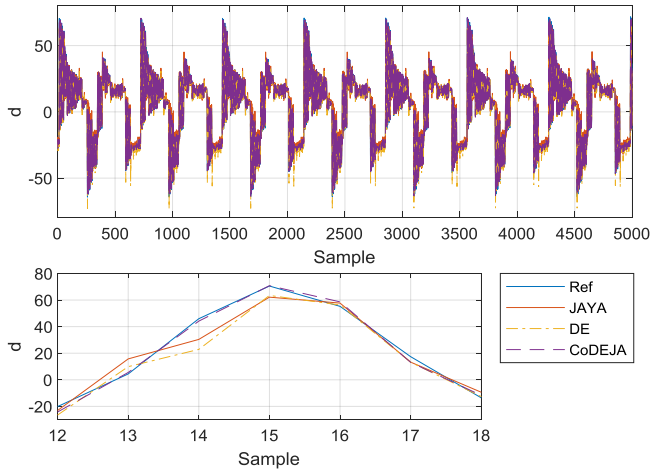

Figure 9: The performance identification in validating of Piezo system

Table 7: Performance identification of the Piezo system

Method	MSE					
	Training			Validation		
	Best	Worst	Average	Best	Worst	Average
CoDEJA	0.2391	0.2758	0.2533	0.2600	0.1483	0.2742
Jaya	0.5875	12.8411	3.396	0.6020	2.1090	1.5216
DE	1.3464	24.0183	15.9244	0.9192	5.3050	6.7693

The identification results of the Piezo system are described as follows. Fig.8 shows the convergence speed of DE, JAYA and CoDEJA algorithms in the training process, while Fig.9 shows the performance identification of those algorithms in the validating process. Table 7 tabulates the criteria's performance (i.e. minimum, maximum, and average) in the training and validating process.

From Fig.8, it can be seen that the convergence speed of CoDEJA yields a stronger performance than DE and JAYA algorithms. After 2000 generations, the CoDEJA converges to near zero. From Table 7, the MSE in the training and validating process of CoDEJA is 0.2533 and 0.2742, while DE yields 15.9244 and 6.7693, JAYA yields 3.396 and 1.5216.

In summary, through testing two different nonlinear benchmark systems, the CoDEJA algorithm has significantly improved quality compared to the DE and JAYA algorithms. The CoDEJA algorithm has combined the strengths of the DE and JAYA algorithms to balance the two aspects of local exploitation and global search to improve quality.

2.3.3 *Results compared with other methods*

In this section, the CoDEJA-optimized neural is compared to other studies that use the same datasets. Table 8 shows the results of CoDEJA-optimized neural with Cooperative RBFNN-DE-BHSA (binary harmony search algorithm) [26] in terms of best or minimum, worst or maximum, mean and standard deviation (Std.dev).

Table 8: Performance comparison of MR damper

Model	Regression input of neural	Best	Worst	Mean	Std.dev
CoDEJA-NN	$f(k-1), f(k-2)$ $v(k-1), v(k-2)$	3.840	4.369	4.087	0.201
RBFNN-DE-BHSA	$f(k-1), f(k-2)$ $v(k-1), v(k-2)$ $v(k-6), v(k-7)$	10.51	11.81	11.29	0.327

It can be seen that the identification performance of the CoDEJA-NN is better than the RBFNN- DE-BHSA model in terms of Mean and Std. dev criteria. The error of the CoDEJA-optimized Neural model is 4.0872 ± 0.2011, while the RBFNN- DE-BHSA model is 11.2856 ± 0.3273.

2.4 Modeling and identification of PMA system

First, training data is collected using the experimental PMA system to gather data on the applied voltage and the joint angle, as shown in Fig. 10. Here, $u_1(k)$ represents the applied voltage inputs, and $\theta_1(k)$ represents the joint angle output. This input-output data is used for both estimation (a) and validation (b) of the CoDEJA-optimized neural model.

Second. To Select model structure is as the input to the neural network defined by $\boldsymbol{X} = [\theta_1(k-1), \theta_1(k-2), \theta_1(k-3), u_1(k-1), u_1(k-2), u_1(k-3), bias]^T$ and the output is denoted by $\boldsymbol{Y} = \theta_1(k)$. The total weight of neural model in the hidden layers is 10.

Finally, the estimation and validation process are conducted to identify the PMA system. Figure 11 shows the performance identification of the PMA system on validating process. Figure 12 shows the histogram of identification errors. Figure 13 depicts the performance identification curve of the PMA system, where the horizontal axis is the voltage supplied to the PMA and the vertical axis is the rotation angle of the PMA.

The identification results show that the dynamic model of the PMA system performs very well in capturing the system's behavior. As shown in Figure 12, the model's prediction errors are mostly small and centered around -2.10^{-3}, with a distribution that is roughly symmetric and shaped like a bell. This indicates that the model has high accuracy, with no large errors. Although the slight shift to the left suggests a small tendency to underestimate, the narrow spread and absence of large outliers confirm the model's reliability.

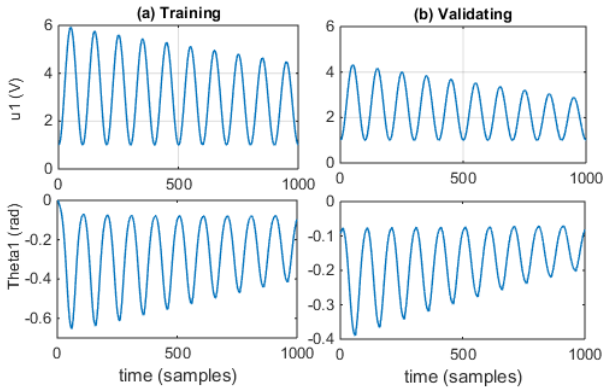

Figure 10: Collection of data for estimating and validating processes

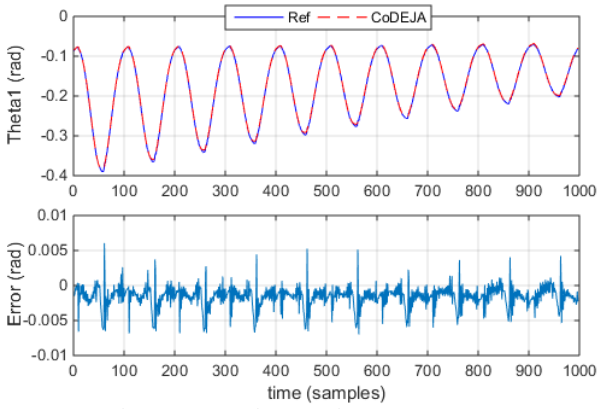

Figure 11: Performance identification of PMA system

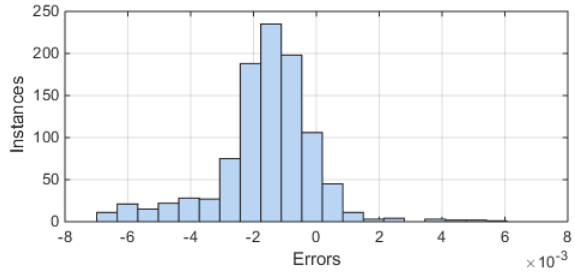

Figure 12: Histogram of identification errors

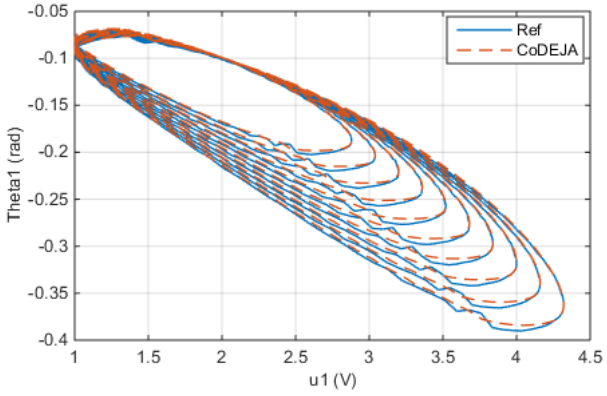

Figure 13: The performacne identification curve of PMA system

3. Conclusion

This paper introduces CoDEJA, a cooperative coevolutionary algorithm designed to simultaneously optimize the weights and biases of neural networks, enhancing both their search capabilities and convergence

speed. CoDEJA utilizes modified differential evolution (MDE) and JAYA algorithms to subdivide the population into two subpopulations, each responsible for optimizing a specific set of neural network parameters. To evaluate CoDEJA's performance, two nonlineear dynamic systems were used for testing. CoDEJA was compared against the classical DE and JAYA algorithms, as well as other existing approaches. The results demonstrate that CoDEJA-optimized neural networks achieve high accuracy and robustness. In addition, the proposed method has successfully applied modeling and identification of the PMA system.

Acknowledgement

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 107.01-2021.92.

References

[1] F. Abdollahi, H. A. Talebi, and R. V Patel, "Stable identification of nonlinear systems using neural networks: Theory and experiments," *IEEE/ASME Transactions On Mechatronics*, vol. 11, no. 4, pp. 488–495, 2006. https://doi.org/10.1109/TMECH.2006.878527

[2] F. Abdollahi, H. A. Talebi, and R. V Patel, "Stable identification of nonlinear systems using neural networks: Theory and experiments," *IEEE/ASME Transactions On Mechatronics*, vol. 11, no. 4, pp. 488–495, 2006. https://doi.org/10.1109/TMECH.2006.878527

[3] T.-H. Le, "Feed-Forward and Long Short-Term Neural Network Models for Power System State Estimation," *Acta Polytech. Hungarica*, vol. 21, no. 6, pp. 223–241, 2024. https://acta.uni-obuda.hu/Le_146.pdf

[4] M. Stefanoni, M. Takács, Á. Odry, and P. Sarcevic, "A Comparison of Neural Networks and Fuzzy Inference Systems for the Identification of Magnetic Disturbances in Mobile Robot Localization," *Acta Polytech. Hungarica*, vol. 22, no. 1, 2025. https://acta.uni-obuda.hu/Stefanoni_Takacs_Odry_Sarcevic_153.pdf

[5] G. Quaranta, W. Lacarbonara, and S. F. Masri, "A review on computational intelligence for identification of nonlinear dynamical systems," *Nonlinear Dynamics*, vol. 99, no. 2, pp. 1709–1761, 2020. https://doi.org/10.1007/s11071-019-05430-7

[6] X.-F. Liu, Z.-H. Zhan, and J. Zhang, "Resource-aware distributed differential evolution for training expensive neural-network-based controller in power electronic circuit," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6286–6296, 2021. https://doi.org/10.1109/TNNLS.2021.3075205

[7] N. N. Son and N. Van Cuong, "Neuro-evolutionary for time series forecasting and its application in hourly energy consumption prediction," *Neural Computing and Applications*, pp. 1–11, 2023. https://doi.org/10.1007/s00521-023-08942-x

[8] D. Min, Z. Song, H. Chen, T. Wang, and T. Zhang, "Genetic algorithm optimized neural network based fuel cell hybrid electric vehicle energy management strategy under start-stop condition," *Applied Energy*, vol. 306, p. 118036, 2022. https://doi.org/10.1016/j.apenergy.2021.118036

[9] N. N. Son, T. M. Chinh, and H. P. H. Anh, "Uncertain nonlinear system identification using Jaya-based adaptive neural network," *Soft Computing*, 2020, doi: 10.1007/s00500-020-05006-3.

[10] W. L. Mao, Suprapto, C. W. Hung, and T. W. Chang, "Nonlinear system identification using BBO-based multilayer perceptron network method," *Microsystem Technologies*, vol. 27, pp. 1497–1506, 2021. https://doi.org/10.1007/s00542-019-04415-1

[11] F. Zhao, T. Jiang, T. Xu, and N. Zhu, "A co-evolutionary migrating birds optimization algorithm based on online learning policy gradient," *Expert Systems with Applications*, vol. 228, p. 120261, 2023. https://doi.org/10.1016/j.eswa.2023.120261

[12] N. Zhu, F. Zhao, and J. Cao, "A knowledge-driven co-evolutionary algorithm assisted by cross-regional interactive learning," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 107017, 2023. https://doi.org/10.1016/j.engappai.2023.107017

[13] C. Huang, X. Zhou, X. Ran, Y. Liu, W. Deng, and W. Deng, "Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem," *Information Sciences*, vol. 619, pp. 2–18, 2023. https://doi.org/10.1016/j.ins.2022.11.019

[14] S. S. Tirumala, "Evolving deep neural networks using coevolutionary algorithms with multi-population strategy," *Neural Computing and Applications*, vol. 32, no. 16, pp. 13051–13064, 2020. https://doi.org/10.1007/s00521-020-04749-2

[15] M. Gong, J. Liu, A. K. Qin, K. Zhao, and K. C. Tan, "Evolving deep neural networks via cooperative coevolution with backpropagation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 420–434, 2020. https://doi.org/10.1109/TNNLS.2020.2978857

[16] L. Wei, L. Jin, and X. Luo, "A robust coevolutionary neural-based optimization algorithm for constrained nonconvex optimization," *IEEE Transactions on Neural Networks and Learning Systems*, 2022. https://doi.org/10.1109/TNNLS.2022.3220806

[17] J. Liang, G. Chen, B. Qu, C. Yue, K. Yu, and K. Qiao, "Niche-based cooperative co-evolutionary ensemble neural network for classification," *Applied Soft Computing*, vol. 113, p. 107951, 2021. https://doi.org/10.1016/j.asoc.2021.107951

[18] H. V. H. Ayala, D. Habineza, M. Rakotondrabe, and L. dos Santos Coelho, "Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks," *Appl Soft Comput*, vol. 87, p. 105990, 2020. https://doi.org/10.1016/j.asoc.2019.105990

[19] Y. Xue, Y. Tong, and F. Neri, "An ensemble of differential evolution and Adam for training feed-forward neural networks," *Information Sciences*, vol. 608, pp. 453–471, 2022. https://doi.org/10.1016/j.ins.2022.06.036

[20] F. Xie, L. Li, L. Li, Y. Huang, and Z. He, "A decomposition-based multi-objective Jaya algorithm for lot-streaming job shop scheduling with variable sublots and intermingling setting," *Expert Syst Appl*, vol. 228, p. 120402, 2023. https://doi.org/10.1016/j.eswa.2023.120402

[21] S. K. Chaurasiya, A. Biswas, A. Nayyar, N. Zaman Jhanjhi, and R. Banerjee, "DEICA: A differential evolution-based improved clustering algorithm for IoT-based heterogeneous wireless sensor networks," *International Journal of Communication Systems*, vol. 36, no. 5, p. e5420, 2023. https://doi.org/10.1002/dac.5420

[22] Z. H. Ding, Z. R. Lu, and F. X. Chen, "Parameter identification for a three-dimensional aerofoil system considering uncertainty by an enhanced Jaya algorithm," *Engineering Optimization*, vol. 54, no. 3, pp. 450–470, 2022. https://doi.org/10.1080/0305215X.2021.1872558

[23] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016. https://doi.org/10.5267/J.IJIEC.2015.8.004

[24] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997. https://doi.org/10.1023/A:1008202821328

[25] J. Wang, A. Sano, T. Chen, and B. Huang, "Identification of Hammerstein systems without explicit parameterisation of non-linearity," *International Journal of Control*, vol. 82, no. 5, pp. 937–952, 2009. https://doi.org/10.1080/00207170802382376

[26] H. V. H. Ayala, D. Habineza, M. Rakotondrabe, and L. dos Santos Coelho, "Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks," *Applied Soft Computing*, vol. 87, p. 105990, 2020. https://doi.org/10.1016/j.asoc.2019.105990