EAI Endorsed Transactions
on AI and Robotics

Research Article EAI.EU

Feedback Control Systems Stabilization Using a Bio-inspired Neural Network

Spyridon D. Mourtas[1], Vasilios N. Katsikis[1,*], Chrysostomos Kasimis[2]

[1]Department of Economics, Division of Mathematics and Informatics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece
[2]Department of Physics, Electronics Laboratory, University of Patras, Patras, GR-26504, Greece

Abstract

The proportional–integral–derivative (PID) control systems, which have become a standard for technical and industrial applications, are the fundamental building blocks of classical and modern control systems. In this paper, a three-layer feed-forward neural network (NN) model trained to replicate the behavior of a PID controller is employed to stabilize control systems through a NN feedback controller. A novel bio-inspired weights-and-structure-determination (BIWASD) algorithm, which incorporates a meta-heuristic optimization algorithm dubbed beetle antennae search (BAS), is used to train the NN model. More presicely, the BIWASD algorithm identifies the ideal weights and structure of the BIWASD-based NN (BIWASDNN) model utilizing a power sigmoid activation function while handling model fitting and validation. The results of three simulated trials on stabilizing feedback control systems validate and demonstrate the BIWASDNN model's exceptional learning and prediction capabilities, while achieving similar or better performance than the corresponding PID controller. The BIWASDNN model is compared to five other high-performing NN models, and a MATLAB repository is accessible in public through GitHub to encourage and enhance this work.

Received on 22 December 2021; accepted on 02 February 2022; published on 04 February 2022

Keywords: Beetle Antennae Search, Neural networks, PID controller, WASD, Feedback control systems

doi:10.4108/airo.v1i.17

1. Introduction

The proportional–integral–derivative (PID) controllers have been used successfully in process-controlled fields of industry such as machinery, metallurgy, power, and light industry since they first emerged decades ago [1]. The PID controller is a feedback-based control loop technique, and two of the main reasons for its continuous use are its simplicity of design and analysis, as well as its simplicity of implementation. However, PID controllers have downsides despite having a simple structure and being simple to comprehend and apply to many control systems [2]. Tuning the PID parameters, notably K_p, K_i, and K_d, determines the performance of the PID control system. Improper tuning will result in inferior or even unstable performance of the controlled system [3]. A neural network (NN) approach is presented in this research for substituting the PID controller with a NN feedback controller in control systems, resulting in equivalent or even better performance of the feedback controlled system. It is worth mentioning that the main advantage of a NN feedback controller over a PID controller is that it requires less computing during the feedback process because integration and derivatives are not used.

The rapid advancement of artificial intelligence, as well as modern electronics and information technologies, has resulted in a plethora of excellent theoretical research findings on artificial NN. Artificial NNs may be used to model and anticipate complicated problems and patterns, such as predicting summary statistics [4], diagnosis of breast cancer [5], financial time-series forecasting [6], optimize a financial portfolio [7], calculating specific matrices in maths [8], tracking robotic motion

*Corresponding author. Email: vaskatsikis@econ.uoa.gr

and mobile object [9, 10], solving perturbed time-varying underdetermined linear systems [11]. In general, determining the best structure of the NN is important and useful. Obtaining the ideal linking weights and number of hidden-layer neurons (HLNs), particularly in the generic multi-input NN, may significantly reduce computational complexity, increase hardware realization, and therefore improve the NN's efficiency [12]. One of the most significant and common feed-forward NN models is the error back-propagation (BP) training algorithm or its variations, which has extensive theoretical studies and real-world applications. BP algorithms are gradient-based iterative approaches that alter the artificial NN weights in a gradient-based descent direction to bring the input/output behavior into a desired mapping. BP-type NNs, in particular, appear to have the following flaws:
1. the probability of being trapped in some local minima;
2. difficulty selecting suitable learning rates (or, say, speed of training);
3. inability to design the optimal or smallest NN structure in a deterministic manner (or, say, high computational complexity).
As a result of the aforementioned inherent flaws, many improved BP-type algorithms have been developed and investigated. It is worth noting that many studies focus on the learning algorithm itself in order to improve the performance of BP-type NNs [13]. Nevertheless, the vast majority of improved BP-type algorithms have yet to overcome the aforementioned fundamental flaws. [14]. As a result, NNs determined from BP methods have a high computational complexity for obtaining the ideal connecting weights [13], and establishing the optimal NN structure is still a difficult process [14].

A number of weights-and-structure-determination (WASD) algorithms are presented as superior alternatives in [15] to prevail over the problems arising from BP algorithms and to define the appropriate NN structure for finer implementations. The WASD algorithm uses the weights-direct-determination (WDD) method to directly specify the optimal linking weights among the hidden and output layers while also acquiring the optimal amount of HLNs. Note that three major issues must be resolved during the design of a NN model for any application:
1. the activation function;
2. the number of HLNs (or, say, the structure);
3. the computation of connecting weights between two separate layers.
According to the previous analysis, obtaining the optimal connecting weights and the optimal number of HLNs for the multi-input NN are useful and important, especially in the general multi-input NNs, because they can considerably reduce the computational complexity and promote hardware realization. That is, they improve the efficiency of the NNs [15].

Another approach for improving the performance of artificial NNs is to use meta-heuristics. In this concept, the Beetle Antennae Search (BAS) algorithm has been used to optimize Elman NN [16], feed-forward high-dimensional NN [17], fog computing networks [18], and back-propagation NN [19]. In this paper, we also employ BAS to strengthen the proficiency of a WASD algorithm. It is worth mentioning that BAS is capable of effective global optimization and has been widely used in a variety of scientific domains in recent years, such as robotics [20], engineering [21], and finance [22–24]. In this research, a novel bio-inspired WASD (BIWASD) algorithm for training NN is developed by integrating the BAS and WASD techniques, and a three-layer feed-forward BIWASD-based NN (BIWASDNN) model is presented. The BIWASD algorithm identifies the ideal weights and structure of the BIWASDNN model utilizing a power sigmoid activation function (AF), while employing cross-validation to address bias and prevent being stuck in local optima during the training process. More particularly, the BIWASD algorithm discovers the ideal number of HLNs, as well as the best power of the AF at each HLN, to minimise the model's error throughout validation. The NN structure is optimized in this way, while the BIWASD method discovers the ideal weights. As a consequence, the computational cost is reduced even further than with conventional WASD approaches, that might require a large number of HLNs, and the cross-validation throughout the training phase improves the accuracy of the anticipated results even further. The BIWASD and the other WASD algorithms in [15] uses the WDD method and are liable for training the NN model, but the BAS algorithm incorporation, the power sigmoid AF, and the cross-validation throughout the training procedure are the differences. The results of three simulated trials on stabilizing feedback control systems validate and demonstrate the BIWASDNN model's exceptional learning and prediction capabilities, while achieving similar or better performance than the corresponding PID controller.

The following are the work's highlights:

- A NN approach to feedback control systems stabilization is proposed and investigated;
- A three-layer feed-forward BIWASDNN model is presented and studied, and a new BIWASD algorithm for training WASD-based NN model is proposed by merging the algorithms of BAS and WASD;
- Three simulated trials on stabilizing feedback control systems through a NN feedback controller

are presented. In these trials, the BIWASDNN model is compared against five other high-performance NN models, and the numerical stability of the BIWASD is investigated.

The following is the layout of the paper. Section 2 provides preliminaries on feedback control systems based on PID and the NN approach is described. Section 3 introduces the BIWASDNN model and its theoretical basis is analysed. Section 4 includes three simulated trials on stabilizing feedback control systems which examines the prediction ability of the BIWASDNN model on PID controllers output and the performance of NN feedback controller on feedback control systems stabilization. It also includes a comparison of the BIWASDNN model to five other high-performing NN models, as well as a concise overview and relevant information about the MATLAB repository, which is available on GitHub. Finally, in section 5, the final remarks are stated.

2. PID Feedback Control and NN Approach

Open loop control and closed loop (feedback) control are the two fundamental types of control loops. In open loop control, the controller's control action is independent of the process variable (PV), whereas in closed loop control, the controller's control action is dependent on process feedback in the form of the PV's value. A feedback loop assures that the controller exerts a control action to manage the PV to be identical to the reference input in a closed loop controller. Closed loop controllers are also known as feedback controllers because of this [25]. Control theory introduces feedback to prevail over the shortcomings of the open-loop controller. That is, a closed-loop controller employs feedback to control a dynamical system's states or outputs.

The PID controller is a typical feedback controller (or closed-loop controller) architecture. A PID controller computes an error value $e(t)$ as the difference between a desired setpoint and a measured PV on a continuous basis and makes a correction using proportional, integral, and derivative components. Note that these three components operate on the error signal to generate a control signal. They have been employed in almost all analogue control systems since the 1920s, and their theoretical understanding and application date back to that time. Assuming that $u(t)$ is the control signal sent to the system, $r(t)$ is the desired output, $y(t)$ is the measured output and $e(t) = r(t) - y(t)$ is the tracking error, the general form of a PID controller is as follows:

$$u(t) = \mathrm{K_p} e(t) + \mathrm{K_i} \int_0^t e(\tau)\mathrm{d}\tau + \mathrm{K_d}\frac{\mathrm{d}e(t)}{\mathrm{d}\tau}, \tag{2.1}$$

where $\mathrm{K_p}, \mathrm{K_i}, \mathrm{K_d} \in \mathbb{R}_0^+$ denote the coefficients for the proportional, integral, and derivative terms, respectively. Adjusting these three factors, frequently iteratively by tuning and without specialized knowledge of a plant model, yields the desired closed loop dynamics.

A NN feedback controller is presented in this section. To train the NN feedback controller, we must first discretize the entire feedback control system procedure. The purpose is to generate data that can be used to train the NN feedback controller. As a result, we construct a discrete-time PID controller ready for programming in Prop. 2.1, based on the continuous-time PID controller (2.1), where each of the terms is a discretized version of its equivalent continuous-time terms.

Proposition 2.1. The discrete version of (2.1) can be formulate as follows:

$$\begin{aligned} u(t_k) =& u(t_{k-1}) + \left(\mathrm{K_p} + \mathrm{K_i}\Delta t + \frac{\mathrm{K_d}}{\Delta t}\right) e(t_k) \\ &+ \left(-\mathrm{K_p} - \frac{2\mathrm{K_d}}{\Delta t}\right) e(t_{k-1}) + \frac{\mathrm{K_d}}{\Delta t} e(t_{k-2}), \end{aligned} \tag{2.2}$$

where Δt denotes the sampling period and k denotes the sample index.

Proof. Differentiating both sides of (2.1) employing Newton's notation yields:

$$\dot{u}(t) = \mathrm{K_p}\dot{e}(t) + \mathrm{K_i} e(t) + \mathrm{K_d}\ddot{e}(t),$$

then approximating the derivative terms, we have the following:

$$\begin{aligned} \frac{u(t_k) - u(t_{k-1})}{\Delta t} =& \mathrm{K_p}\frac{e(t_k) - e(t_{k-1})}{\Delta t} + \mathrm{K_i} e(t_k) \\ &+ \mathrm{K_d}\frac{\dot{e}(t_k) - \dot{e}(t_{k-1})}{\Delta t}. \end{aligned}$$

Approximating the rest derivative terms and then solving in terms of $u(t_k)$, it is easily observable that we obtain (2.2), hence completes the proof. □

Consider the single-input-single-output (SISO) control system of Fig. 1. The plant shown therein typically operates in continuous-time and it can be described by the following s-transfer function:

$$H(s) = G(s)U(s), \tag{2.3}$$

where $U(s), H(s)$ are the Laplace transforms of the input and output signals, respectively, and

$$G(s) = \frac{\sum_0^{n-1} b_i s^i}{\sum_0^n a_i s^i}, \tag{2.4}$$

where $G(s)$ is the transfer function of the plant with the coefficients $a_i, b_i \in \mathbb{R}$.

To convert the discrete-time signal $u(t_k)$ produced by the controller in (2.2) to a continuous-time piecewise

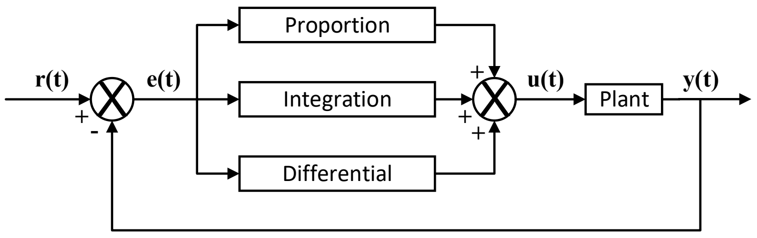


Figure 1. PID controller system structure.

constant signal, the zero-order-hold (ZOE) method is used [26]. In this way, the transfer function $H(s)$ is converted from s-domain to z-domain. Then, the difference equation of the z-transfer function $H(z)$ can be obtained as shown in Prop. 2.2.

Proposition 2.2. Considering the following z-transfer function:

$$H(z) = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} = \frac{y(z)}{u(z)}, \tag{2.5}$$

then its difference equation can be formulate as follows:

$$y(t_k) = -a_1 y(t_{k-1}) - a_0 y(t_{k-2}) + b_1 u(t_{k-1}) + b_0 u(t_{k-2}). \tag{2.6}$$

Proof. By cross multiplying (2.5), it can be reformulated as follows:

$$(z^2 + a_1 z + a_0)y(z) = (b_1 z + b_0)u(z)$$

or equivalent

$$z^2 y(z) + a_1 z y(z) + a_0 y(z) = b_1 z u(z) + b_0 u(z).$$

Taking the inverse transform of the above equation, we have that

$$z^2 y(z) = y(t_{k+2}), \quad a_1 z y(z) = a_1 y(t_{k+1}), \quad a_0 y(z) = a_0 y(t_k),$$
$$b_1 z u(z) = b_1 u(t_{k+1}), \quad b_0 u(z) = b_0 u(t_k),$$

and thus we have the following difference equation:

$$y(t_{k+2}) + a_1 y(t_{k+1}) + a_0 y(t_k) = b_1 u(t_{k+1}) + b_0 u(t_k).$$

Reducing each time index by 2 and then solving in terms of $y(t_k)$, it is easily observable that we obtain (2.6), hence completes the proof. □

Acquiring the signal $u(t)$ values of a PID controller during the control system stabilization process, the NN feedback controller in Fig. 2 can be trained to predict the signal $u(t)$ produced by the PID using the error $e(t)$ as input.

According to the aforementioned, the following Alg. 1 describes the whole discretized process both in the cases of a PID control system and a NN feedback control system.

Algorithm 1 Discretized process of a PID control system and a NN feedback control system.

```
Input: The desired output $r(t)$, the s-transfer function of the plant $H(s)$, the sampling period $\Delta t$ and $T$ the period end.
1: Convert the $H(s)$ from s-domain to z-domain.
2: Set $t = 2\Delta t$, $y(0:t) = 0$ and $e(0:t) = r(0:t)$
3: while $t \leq T$ do
4:     In the case of a PID controller, set $u(t)$ according to (2.2), and in the case of NN feedback controller, set $u(t)$ the NN prediction base on $e(t)$.
5:     Set $y(t)$ according to (2.6).
6:     Set $e(t) = r(t) - y(t)$.
7:     $t \leftarrow t + \Delta t$
8: end while
Output: The error $e(t)$, the signal $u(t)$ produced by the PID or the NN feedback controller, and the system output $y(t)$.
```

3. The BIWASDNN Model

This section introduces a three-layer feed-forward NN model with one input and n hidden layer neurons, as shown in Fig. 3. The first layer, more specifically, is the input layer, which receives and distributes X, i.e. the input, to the associated equal-weighted neuron in the second layer. The second layer, which includes no more than n power activated neurons, receives the value X as input from the first layer, and the corresponding AFs $F_v(X)$, $v \in [0, n-1] \subseteq \mathbb{Z}$, are power sigmoid functions with varying powers. The output layer, which contains a nonactivated neuron, is the final layer. The weight vector W is composed of the weights W_v in the neuron-to-neuron link among the neurons of the second and third layers, and is produced using the WDD method. BIWASDNN is the name of the NN model, which is trained using a novel BIWASD algorithm. Note that the BIWASD algorithm is responsible for finding the ideal weights W and the structure of the NN, i.e. finding the corresponding AFs $F_v(X)$ varying powers v. We will go through all of the details about the model's construction and structure in this section.

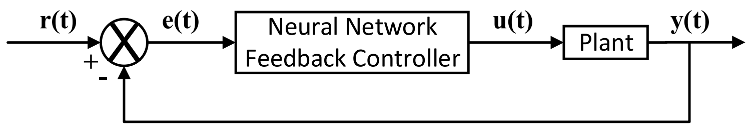


Figure 2. NN feedback controller system structure.

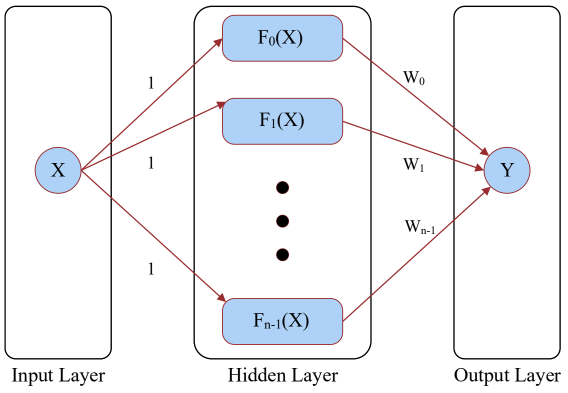


Figure 3. Structure of the BIWASDNN model.

3.1. WDD Method and AF

Machine learning is a computationally intensive process, and calculating iterative training errors is a difficult task. The WASD algorithm for NN training is implemented to reduce the computational cost of this process and simplify the network composition [15].

Comprehensive interpretations of important key theoretical underpinnings and analyses are provided here for the construction of the BIWASDNN. To start with the Taylor polynomial (TP) approximation theorem [27] is determined as below.

Theorem 3.1. Assume that K is a nonnegative integer and, on interval $[a,b]$, a target function $f(\cdot)$ has the $(K+1)$-order continuous derivative, then for $x \in [a,b]$,

$$f(x) = P_K(x) + R_K(x), \tag{3.1}$$

where $P_K(x)$ is a polynomial employed to approximate $f(x)$ and $R_K(x)$ is the error term.

Such that, with a fixed value $h \in [a,b]$, $f(x)$ may be approximated as below:

$$f(x) \approx P_K(x) = \sum_{i=0}^{K} \frac{f^{(i)}(h)}{i!}(x-h)^i, \tag{3.2}$$

where $f^{(i)}(h)$ signifies the value of the i-order derivative of $f(x)$ at the point h and $i!$ signifies the factorial of i. It is worth noting that $P_K(x)$ is the K-order TP of function $f(x)$.

Proposition 3.1. The TP approximation theorem may be used to approximate multivariable functions [27]. For a target function $f(x_1, x_2, \dots, x_g)$ with $(K+1)$-order continuous partial derivatives in an origin's neighborhood $(0,\dots,0)$ and g variables, the K-order TP $P_K(x_1,x_2,\dots,x_g)$ about the origin is:

$$P_K(x_1,x_2,\dots,x_g) = \sum_{i=0}^{K} \sum_{i_1+\cdots+i_g=i} \frac{x_1\cdots x_g}{i_1\cdots i_g}\left(\frac{\partial^{i_1+\cdots+i_g} f(0,\cdots,0)}{\partial x_1^{i_1}\cdots \partial x_g^{i_g}}\right), \tag{3.3}$$

where $i_1, i_2, \dots, i_g$ are nonnegative integers.

The following nonlinear function may be used to represent the relationship between the NN's input X and the output target Y:

$$Y = f(X). \tag{3.4}$$

Note that our approach is similar to the power activated NN in [15], which is inline with the K-order TP. As a result, considering the variable X and, in an origin's neighborhood $(0,\dots,0)$, the $(K+1)$-order continuous partial derivatives, the K-order TP $P_K(X)$ can map (3.4) as follows:

$$P_K(X) = \sum_{v=0}^{n-1} q_v w_v, \tag{3.5}$$

where $q_v = F_v(X) \in \mathbb{R}$ signifies a power function of all inputs and $w_v \in \mathbb{R}$ signifies the coefficient (or weight) for q_v.

The sigmoid AF is one of the most often used functions while it is most commonly used in models which call on us to anticipate the probability as a result. Due to its range, the sigmoid is the optimal choice because probability only occurs in the range of 0 to 1. Assuming that $X \in \mathbb{R}^{1\times N}$, the following power sigmoid AF is proposed and employed:

$$F_v(X) = \frac{e^{\odot X^{\odot v}}}{e^{\odot X^{\odot v}}+1}, \tag{3.6}$$

where $\odot$ and the superscript $()^{\odot}$ imply the Hadamard (or element-wise) product and the Hadamard exponential, respectively, X implies the function input, while the power value $v \in \mathbb{Z}^+$ and the range of (3.6) is $\left[\frac{1}{2}, 1\right)$. As a result, the HLNs of the K-order TP NN employ the AF of (3.6) to generate sigmoidal activation. It is worth mentioning that several AFs, such as Chebyshev and Euler polynomials, sine, square wave, and power are employed on WASD-based NN in [15, 28, 29].

Moreover, for a given number of samples $S \in \mathbb{N}$ with $X \in \mathbb{R}^S$, we set $q_{S,v} = F_v(X) \in \mathbb{R}^{S\times n}$ and, as a consequence, the input-activation matrix becomes

$$Q = \begin{bmatrix} q_{1,0} & q_{1,1} & \cdots & q_{1,n-1} \\ q_{2,0} & q_{2,1} & \cdots & q_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{S,0} & q_{S,1} & \cdots & q_{S,n-1} \end{bmatrix} \in \mathbb{R}^{S\times n}, \tag{3.7}$$

the weight vector $W = [w_0, w_1, \dots, w_{n-1}] \in \mathbb{R}^{n\times S}$ and the desired-output vector $Y \in \mathbb{R}^S$.

The weights of the BIWASDNN in Fig. 3 are then calculated using the WDD approach, rather than iterative weight training in traditional NNs [27]. From [15], the following lemma can be simply deduced.

Lemma 3.1. The steady-state weights of the K-order TP NN may be acquired forthrightly as below:

$$W = (Q^{\mathrm{T}}Q)^{-1}Q^{\mathrm{T}}Y = Q^{\dagger}Y, \tag{3.8}$$

where the superscripts $()^{\mathrm{T}}$, $()^{-1}$ and $()^{\dagger}$ signify the transpose operator, the inverse operator and the pseudo-inverse operator, respectively.

According to that, the matrix Q can be computed as proposed in Alg. 2.

Algorithm 2 Computing matrix Q.

```
Input: The data input $X$, the vector $N$ which comprises
    of the powers $v$ of each HLN.
1: $n \leftarrow$length$(N)$ and $m \leftarrow 1$
2: while $m \le n$ do
3:     $\hat{X} \leftarrow X^{\odot N(m)}$
4:     $Q(:,m) \leftarrow e^{\hat{X}} \odot (e^{\hat{X}} + 1)^{-1}$
5:     $m \leftarrow m + 1$
6: end while
Output: Matrix $Q$.
```

3.2. BIWASD Algorithm

The BIWASD algorithm, as well as the entire process of modelling and predicting with the BIWASDNN model, are discussed in depth in this subsection. BAS algorithm has been introduced in [30] and mimics the searching behavior of a beetle as presented in Fig. 4. Because of its simplicity, BAS permits novel optimisation algorithms to be developed (see [31–35]). As a result, the BIWASD algorithm uses the beetle searching behavior for finding the global minimum to specify the optimal weights of the NN.

The BIWASD algorithm is in charge of training the NN model. First, the data input is separated into two set of samples for fitting and validation. Note that this is the well-known cross-validation method, which is employed to evaluate the consistency of a machine learning model employing the validation data, as part of training. Validation tries to ensure that the model's performance generalizes beyond the training set because the validation set is separate from the training set. The parameter $p \in (0,1] \subseteq \mathbb{R}$ allows the user to specify the precise percentage difference between the fitting and validation sets. Supposing that H is the sample size of X, then the first $H_1 = pH$ samples of X are used for fitting the model and the last $H_2 = H - H_1$ samples for validation.

Second, the beetle searching behavior in Fig. 4 is adapted. The beetle searching behavior is described by the following random route when x is the beetle's position at t-th time and $f(x)$ is the smell strength, expressed as a fitness function:

$$z = \frac{\mathrm{rand}(n,1)}{\|\mathrm{rand}(n,1)\|},$$

$$Z = \frac{z}{2^{-52} + \|z\|}, \tag{3.9}$$

where n signifies the length of vector x and rand$(\cdot)$ signifies a random function. We set x_l to represent the left antennae and x_r to represent the right antennae, and their seeking behaviors are formulated as follows:

$$x_l = \mathrm{rnd}(x_t - Zd_t), \tag{3.10}$$

$$x_r = \mathrm{rnd}(x_t + Zd_t), \tag{3.11}$$

where rnd$(\cdot)$ denotes a function that outputs a number's value rounded up to the next integer, while the capacity to exploit is correlated to the sensing diameter of the antennae d. Moreover, the detecting behavior can be formulated as follows:

$$x_t = \mathrm{rnd}(x_{t-1} + S\delta_t \mathrm{sign}(f(x_r) - f(x_l))). \tag{3.12}$$

where sign$(\cdot)$ denotes a sign function, and δ signifies a size step that depicts the convergence speed after an increase in t throughout the searching. Last, the d and $delta$ update rules are the followings:

$$d_t = 0.95d_{t-1} + 0.001, \tag{3.13}$$

$$\delta_t = 0.95\delta_{t-1}. \tag{3.14}$$

It is worth noting that the aforementioned process is a converted BAS algorithm that returns only integer solutions.

Moreover, the fitness function was adjusted to identify the NN's ideal structure. Based on the results of (3.12), we do this by removing the HLNs that relate to a negative number. The fitness function then computes the matrix Q as shown in Alg. 2, and the corresponding

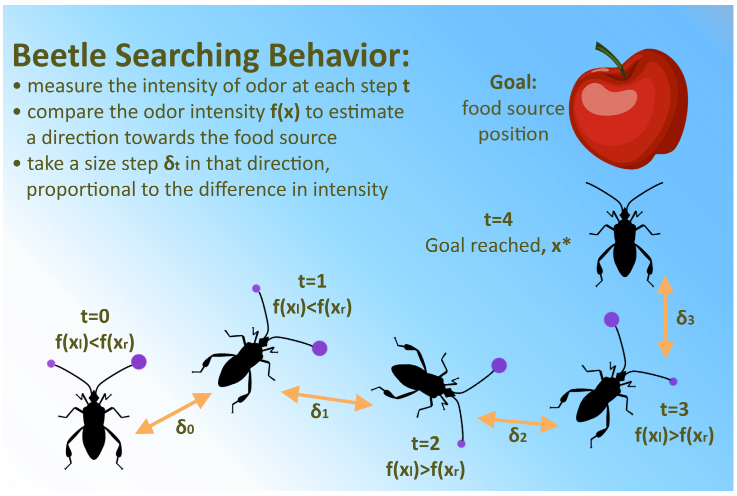

Beetle Searching Behavior:
• measure the intensity of odor at each step t
• compare the odor intensity f(x) to estimate a direction towards the food source
• take a size step δt in that direction, proportional to the difference in intensity
Goal: food source position
t=4
Goal reached, x*
t=0
f(xl)<f(xr)
t=1
f(xl)<f(xr)
t=2
f(xl)>f(xr)
t=3
f(xl)>f(xr)
δ0
δ1
δ2
δ3

Figure 4. Beetle searching behavior.

W of this Q matrix is created, as suggested in Lemma 3.1. More precisely, we set $W = Q^{\dagger}(H_1)Y(H_1)$, where $Q(H_1)$ and $Y(H_1)$ denote the Q matrix and the targets on the fitting set H_1. Then, the validation set H_2 predictions are calculated, and the root-mean-square error (RMSE) between these predictions and their target value is measured as shown in Alg. 3. That is, we set $\hat{Y}(H_2) = Q(H_2)W$, where $Q(H_2)$ and $\hat{Y}(H_2)$ denote the Q matrix and the predictions on the validation set H_2. The RMSE is an evaluation of accuracy used in statistics to compare forecasting errors of different models and is commonly used in machine learning as a cost function for regression problems. Also, the RMSE values closer to zero are finer, and is calculated as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{j=1}^{H}\left(Y_j - \hat{Y}_j\right)^2}{H}}, \tag{3.15}$$

where H denotes the samples' number, $\hat{Y}$ and Y denote the predicted and the target value, respectively. Provided a maximum number of HLNs n, the adapted BAS method identifies the ideal number of HLNs as well as the optimal AF power at each HLN, lowering the model's error throughout validation.

Algorithm 3 Fitness function.

```
Input: The data input $X$, the target $Y$, the numbers $H_1$ and $H_2$, the vector $x_t$ which comprises of the powers $v$ of each HLN.
1: procedure Fitness($X, Y, H_1, H_2, x_t$)
2:     Put in vector $N$ only the nonnegative elements of $x_t$.
3:     Calculate the matrix $Q$ through Alg. 2 under the vector $N$ .
4:     Find $W$ via WDD method for the first $H_1$ samples of $X$.
5:     Set $E_N$ the RMSE computed via (3.15) for the last $H_2$ samples of $X$ under $W$.
6: end procedure
Output: The RMSE $E_N$.
```

Last, the initial x_t at $t = 0$ in the aforementioned adapted BAS method must be a random vector $x_0 \in \mathbb{Z}^n$. However, we highly suggest setting as initial x_t the following vector:

$$\begin{aligned} x_0 =& [1 - \mathrm{rnd}(n/2), 2 - \mathrm{rnd}(n/2), 3 - \mathrm{rnd}(n/2), \\ & \dots, n - \mathrm{rnd}(n/2)]^{\mathrm{T}}, \end{aligned} \tag{3.16}$$

Based on [15], the starting structure of the NN employs the suggested powers v for rnd($n/2$) in number HLNs, while the process of adding/removing HLNs in the structure throughout the adjusted BAS iterations is made easier. BAS returns x^* at the end of the iterations, and we only set the values of $x^* \geq 0$ in N^*. Notice that N^* denotes the AF's optimal powers v at each NN HLN, and its length denotes the ideal number of HLNs. The BIWASD algorithm determines and returns the optimal W on the entire training data set, as well as the RMSE of the validation set, according to that N^*.

In conclusion, the BIWASD algorithm is used to train the NN model. It splits the data to fitting-validation sets and, stated a maximum HLN number n, sets the x_0 of

(3.16) as starting powers v of the AF at each HLN of the NN model and finds the optimal N^* by estimating the RMSE of the validation set. It is worth noting that the power sigmoid AF of (3.6) is not obligatory. This AF can be replaced with any of the AFs listed in [15]. Fig. 5a shows the whole procedure for the BIWASD algorithm, whereas Fig. 5b shows the entire process for modelling and predicting with the BIWASDNN model.

4. Experiments and MATLAB Repository

This section investigates the proficiency and the prediction ability of the BIWASDNN model on three simulated trials on stabilizing feedback control systems. It is equitable to compare the BIWASDNN model to certain other well-performing NN models because it is specifically built to handle only regression problems. The Hermite polynomial NN (HPNN) from [15], which is one of the best-performing WASD-based NN in solving regression problems, is one of the models, while the others include a MATLAB long short-term memory (LSTM) model, a MATLAB feed-forward (FF) model, and two models from MathWorks' Regression Learner App. As a consequence, the fine Tree (FTree) and squared exponential Gaussian process regression (SEGPR) models are used to tackle the regression problems which arise when stabilizing feedback control systems are considered. On the one hand, the MATLAB LSTM model has the following specs: optimizer = Adam, numHiddenUnits = 200; maxEpochs = 100; miniBatchSize = 20; LearnRate = 0.01. On the other hand, the MATLAB FF model has the following specs: hiddenSizes = 10. Furthermore, a short overview of the BIWASDNN's MATLAB repository is provided, along with some valuable information.

4.1. Simulated Trials

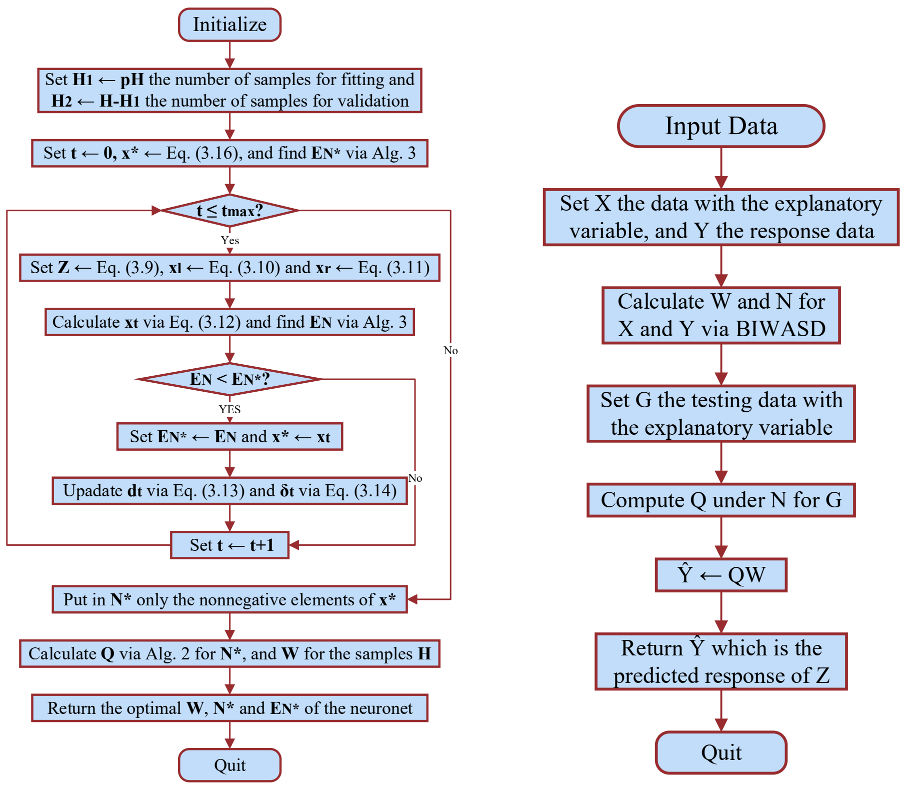
Table 1. Feedback control systems configuration.

Plant	$G(s)$	K_p	K_i	K_d
1	$\frac{1}{s^2+10s+20}$	350	300	50
2	$\frac{3}{5s^2+25s+50}$	200	200	100
3	$\frac{2}{10s^2+100s+10}$	200	200	100

The simulation trials employ Alg. 1 to generate three datasets containing samples of the tracking error $e(t)$ and the corresponding PID controller's signal $u(t)$. Each dataset corresponds to a PID feedback control system results in which the plant's s-transfer function and the PID parameters are declared in Tab. 1. Following the creation of each dataset, the BIWASDNN, HPNN, FTree, and SEGPR models are trained to produce the signal $u(t)$ using the error $e(t)$ as input. Setting the desired output $r(t) = 1$, the sampling period $\Delta t = 0.01$ and the period end $T = 20$, Alg. 1 outputs 2000 samples. Splitting the data 50-50% for training and testing the model, 1000 samples will be used to train the models and 1000 samples will be used to test them.

In the case of the BIWASDNN model, we set $p = 0.7$, splitting the training data to 70-30% for fitting and validating the model. It is interesting to note that the closer the p value gets to 1, the fewer the samples for validating the model will be, and the closer the p value gets to 0, the opposite occurs, lowering the model's prediction accuracy in both circumstances. As a consequence, for the model's optimal prediction accuracy, the optimal value of p must be set in $[0.6, 0.9]$. In addition, the BIWASD's parameters are set to $d_0 = 8$, $\delta_0 = 5$, the maximum iterations to $t_{\max} = 30$, and the maximum number of HLNs to $n = 30$. Assuming that the parameter $p \in [0.6, 0.9]$ and the parameter $t_{\max}$ specifies the maximum iterations, raising the parameter $t_{\max}$ will commonly lead to a high prediction accuracy for the model. In Figs. 6a-6c, the training procedure of the model is depicted through the RMSE for the datasets with plant 1, 2 and 3, respectively, where we observe that the BIWASD algorithm demands less than 30 iterations to converge to the optimal NN's structure. Furthermore, the BIWASD algorithm returned $N^* \in \mathbb{R}^{16}$ and, as a result, the optimal structure of the NN has 16 in number HLNs for the specific runs.

In Figs. 6d-6f, the results of the NNs models on the test data are depicted for the datasets with plant 1,2 and 3, respectively. The NN models' prediction capability on 1000 samples is great and practically identical in Figs. 6d and 6e, but the findings are not as good in Fig. 6f, where the NNs model's predictions have a tiny divergence from the target in some samples. The BIWASDNN, HPNN, Ftree and SEGPR models statistics are presented in Tab. 2 in which the coefficient of determination (R^2), mean absolute percentage error (MAPE), mean absolute error (MAE), root-mean-square error (RMSE), the average number of training iterations (NTI) and the average time consumption (TC) of the train and test prices for the datasets with plant 1, 2 and 3 are included. Note that following the notations used in the RMSE formula (3.15), MAE $= \frac{100}{H}\sum_{j=1}^{H}\left|Y_j - \hat{Y}_j\right|$ and MAPE $= \frac{100}{H}\sum_{j=1}^{H}\left|(Y_j - \hat{Y}_j)/Y_j\right|$. Checking the results of Tab. 2, it is clear that BIWASDNN has the lowest TC in all the datasets while the LSTM has the highest. In all datasets, all of the models have nearly identical statistics, with the exception of the LSTM model, which has poorer statistics. In addition, the FTree model's train data statistics are the best in the dataset with plant 3, while the BIWASDNN's model are the third best. However, the SEGRP model's test data


Initialize
Set H1 ← pH the number of samples for fitting and H2 ← H-H1 the number of samples for validation
Set t ← 0, x* ← Eq. (3.16), and find EN* via Alg. 3
t ≤ tmax?
Yes
No
Set Z ← Eq. (3.9), xl ← Eq. (3.10) and xr ← Eq. (3.11)
Calculate xt via Eq. (3.12) and find EN via Alg. 3
EN < EN*?
YES
No
Set EN* ← EN and x* ← xt
Upadate dt via Eq. (3.13) and δt via Eq. (3.14)
Set t ← t+1
Put in N* only the nonnegative elements of x*
Calculate Q via Alg. 2 for N*, and W for the samples H
Return the optimal W, N* and EN* of the neuronet
Quit
Input Data
Set X the data with the explanatory variable, and Y the response data
Calculate W and N for X and Y via BIWASD
Set G the testing data with the explanatory variable
Compute Q under N for G
Ŷ ← QW
Return Ŷ which is the predicted response of Z
Quit

(a) BIWASD algorithm.

(b) Complete process for modelling and predicting.

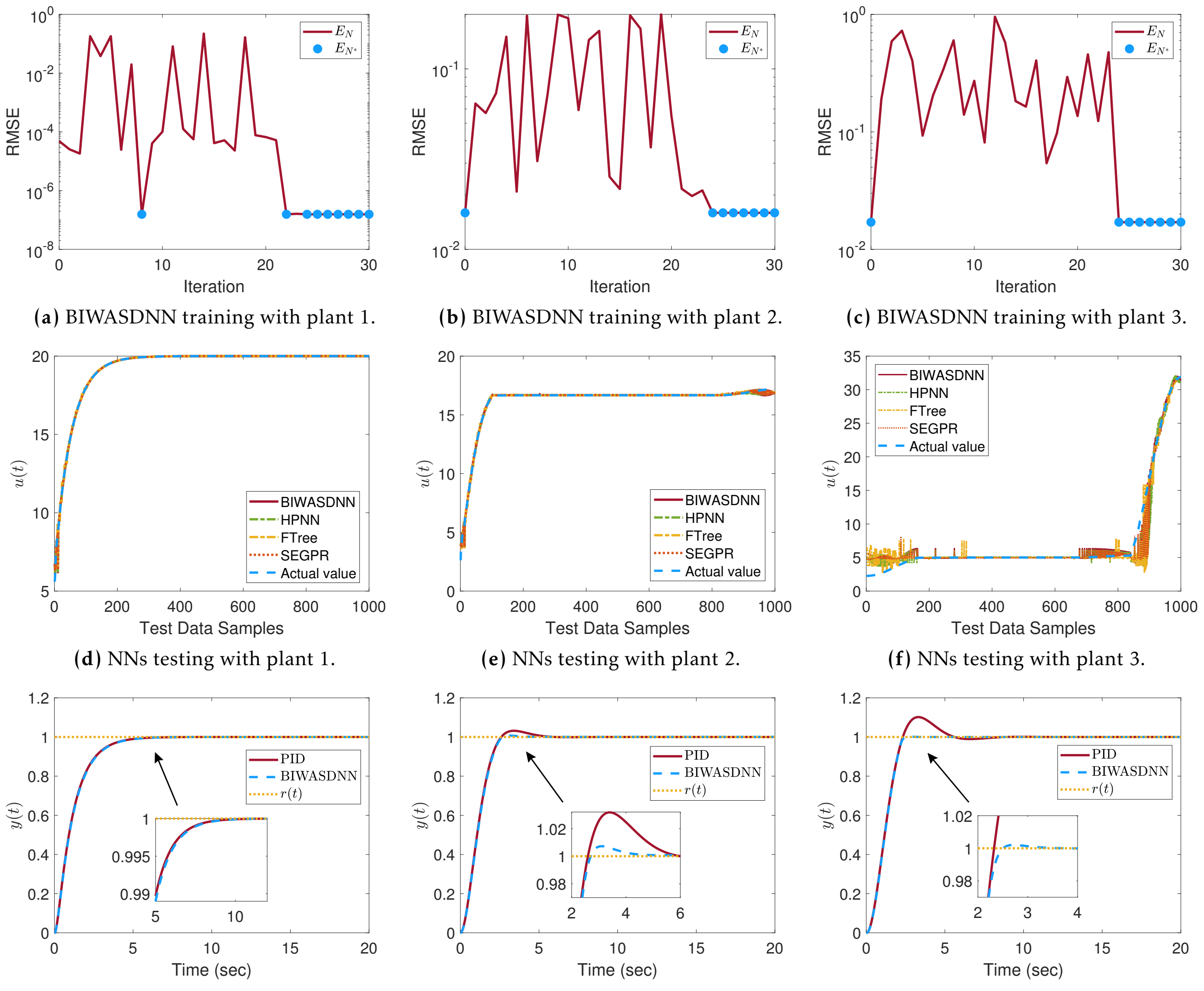
Figure 5. BIWASD algorithm and the entire process of BIWASDNN for modelling and predicting.

statistics are the best in the dataset with plant 3, while the BIWASDNN's model are the second best.

Because Tab. 2 also contains the NTI, some observations concerning algorithm convergence can be made. The BIWASDNN model requires less than 30 iterations to attain the highest prediction accuracy for all the datasets, as also recorded in Figs. 6a-6c, whereas the HPNN model requires 13 iterations. In the case of the LSTM, since the parameter maxEpochs has been set to 100, the model requires 100 iterations to attain the highest prediction accuracy for all the datasets, while the FF model requires 20, 50 and 130 iterations to attain the highest prediction accuracy for the datasets with plant 1, 2 and 3, respectively. Note that the models for FTree and SEGRP are not optimizable, hence the number of iterations is not available through MathWorks' Regression Learner App. According to the aforementioned observations, the HPNN model has the lowest NTI, followed by the BIWASDNN model, with the LSTM model having the highest. However, it is worth mentioning that the HPNN model has an average NTI/TC ration of 1/0.6923, while the BIWASDNN model has 1/0.0033. That is, the BIWASDNN model is 90 times faster to train than the HPNN model.

Now that the BIWASDNN model has been trained for each plant 1, 2 and 3, Alg. 1 is used with the NN feedback controller to stabilize the input signal of each control system. The results of each simulation trial are depicted in Figs. 6g-6i. We can see in Fig. 6g (plant 1 case) that the feedback control system with the NN feedback controller converges to the desired output in a similar fashion to the PID feedback control system. In contrast to the PID feedback control system in Figs. 6h (plant 2 case) and 6i (plant 3 case), the feedback control system with the NN feedback controller converges faster to the desired output. That is, the NN feedback controller has better performance than the PID controller in the cases of plant 2 and 3.

In general, according to Tab. 2 and the Figs. 6d-6f depicted results, the BIWASDNN model performed flawlessly in creating a model that may predict the PID controller's signal $u(t)$ taking as an input the corresponding tracking error $e(t)$, while its predicted capability is almost same to that of the HPNN, Ftree, and EGPR models on the specific datasets with plant 1, 2 and 3, and its TC being the lowest. Moreover, the NN feedback controller achieves similar or better performance than the corresponding PID controller.



(a) BIWASDNN training with plant 1. **(b)** BIWASDNN training with plant 2. **(c)** BIWASDNN training with plant 3.

(d) NNs testing with plant 1. **(e)** NNs testing with plant 2. **(f)** NNs testing with plant 3.

(g) BIWASDNN simulation with plant 1. **(h)** BIWASDNN simulation with plant 2. **(i)** BIWASDNN simulation with plant 3.

Figure 6. NNs training, testing and simulation results in Exp. 4.1.

4.2. BIWASD's Stability and MATLAB Repository

Numerical findings are presented in this subsection to prove the practicality of the suggested BIWASDNN model and the BIWASD algorithm on regression problems that involve datasets for training NN feedback controllers. The generalization capabilities of the suggested BIWASDNN equipped with the BIWASD algorithm are also investigated for completeness.

Because the input-layer weights and hidden-layer biases are produced at random, the final/optimal number of HLNs differs when the BIWASD algorithm is used multiple times. Each time, the BIWASDNN structure with a determined number of HLNs will not be the same. Each BIWASDNN model in Exp. 4.1 is trained and tested 100 times to ensure the stability of the suggested BIWASDNN equipped with the BIWASD method. The average length of N (Avg. len(N)) and the standard deviation of len(N) (σ(len(N))), are then computed 100 times along with the average values of R^2, MAPE, MAE and RMSE. Note that the average len(N) relates to the average optimal number of HLNs. The findings are presented in Tab. 3. Therein, comparing the average len(N) and σ (len(N)) for each BIWASDNN model in Exp. 4.1, we observe that the final/optimal number of HLNs specified by the BIWASD algorithm is considerably stable since the maximum σ (len(N)) is less than 0.56.

In general, when Tab. 3 and Tab. 2 are compared, the average R^2, MAPE, MAE and RMSE of BIWASDNN in Exp. 4.1 are almost identical. Furthermore, the BIWASD's stability on 100 runs is excellent, which makes the BIWASDNN's performance to be antagonistic or even better than the HPNN, Ftree, and SEGPR performances. Notice that the BIWASD algorithm not only determines the best BIWASDNN structure automatically and effectively, but also acquires the

Table 2. NN models' statistics (R^2, MAPE, MAE, RMSE), number of training iterations (NTI) and time consumption (TC).

NN Model	BIWASDNN						HPNN					
Data	TC	R^2	MAPE	MAE	RMSE	NTI	TC	R^2	MAPE	MAE	RMSE	NTI
Plant 1 Train	0.1 s	0.9954	0.2115	0.0121	0.1431	30	9 s	0.9954	0.2049	0.0111	0.1430	13
Plant 1 Test		0.9976	0.1048	0.0088	0.1016			0.9976	0.0988	0.0077	0.1024	
Plant 2 Train	0.1 s	0.9960	0.4589	0.0419	0.1274	30	10 s	0.9959	0.4766	0.0446	0.1288	13
Plant 2 Test		0.9968	0.3575	0.0395	0.1136			0.9966	0.3755	0.0422	0.1154	
Plant 3 Train	0.1 s	0.9377	13.1168	0.6824	1.6421	30	8 s	0.9313	13.0249	0.7376	1.7194	13
Plant 3 Test		0.9381	12.9333	0.6794	1.6374			0.9317	12.8155	0.7339	1.7137	
NN Model	Ftree						SEGRP					
Data	TC	R^2	MAPE	MAE	RMSE	NTI	TC	R^2	MAPE	MAE	RMSE	NTI
Plant 1 Train	1.9 s	0.9943	0.3463	0.0303	0.1593	-	6 s	0.9954	0.2164	0.0120	0.1438	-
Plant 1 Test		0.9966	0.2410	0.0277	0.1227			0.9977	0.1076	0.0086	0.0992	
Plant 2 Train	0.2 s	0.9957	0.5079	0.0433	0.1325	-	5.5 s	0.9962	0.4342	0.0381	0.1250	-
Plant 2 Test		0.9960	0.4316	0.0438	0.1265			0.9969	0.3294	0.0356	0.1101	
Plant 3 Train	0.1 s	0.9556	11.0420	0.5430	1.3983	-	4.5 s	0.9433	13.8469	0.6589	1.5671	-
Plant 3 Test		0.9377	12.5689	0.6266	1.6453			0.9439	13.6197	0.6557	1.5591	
NN Model	LSTM						Feed-forward					
Data	TC	R^2	MAPE	MAE	RMSE	NTI	TC	R^2	MAPE	MAE	RMSE	NTI
Plant 1 Train	32 s	-0.1182	7.7211	1.4283	1.4768	100	0.1 s	0.9952	0.2945	0.0257	0.1470	20
Plant 1 Test		-0.1303	7.7358	1.4296	1.4776			0.9971	0.1880	0.0220	0.1118	
Plant 2 Train	32 s	0.5151	7.7907	1.0647	1.1741	100	0.1 s	0.9956	0.4984	0.0447	0.1326	50
Plant 2 Test		0.5133	7.7907	1.0652	1.1706			0.9966	0.3842	0.0422	0.1146	
Plant 3 Train	32 s	0.8978	22.0061	1.3139	2.2116	100	0.2 s	0.9357	12.1905	0.6954	1.6634	130
Plant 3 Test		0.8975	22.0105	1.3137	2.2093			0.9361	11.9877	0.6918	1.6579	

optimal hidden-layer weights directly. These simulation trials demonstrated that the proposed BIWASDNN, which uses the BIWASD algorithm to approximate target functions, is efficient and effective.

Lastly, on GitHub, you can find the whole creation and implementation of the computational processes described in this work at this link:

`https://github.com/SDMourtas/BIWASDNN`,

where we created a MATLAB repository for stabilizing feedback control systems inline with Algs. 1-3 and the algorithms presented in the diagrams of Figs. 5a and 5b. For the simulation trials conducted in this section, the MATLAB repository contains detailed installation instructions and a complete implementation. Additionally, anyone can draw their own conclusions from their own tests by modifying the BIWASDNN model parameter values in the repository's main MATLAB function.

5. Conclusion

In this paper, the BIWASD algorithm is utilized to train a three-layer feed-forward NN model. The BAS and WASD techniques are combined to create the BIWASD algorithm for training NNs, and the BIWASDNN model is introduced. Using a power sigmoid AF, the BIWASD algorithm determines the proper weights and structure of the BIWASDNN while handling model fitting and validation. Three simulated trials on stabilizing feedback control systems have revealed the BIWASDNN model's learning and predicting performance. The results of the trials present the splendid precision and efficiency of the BIWASDNN model.

There are certain limits and suggestions that can be made about this work.
1. One disadvantage of this work is that the BIWASDNN model can only solve regression problems in machine learning, that narrowed the area of our investigation.
2. A different AF in the WDD process is proposed as a way to improve this research.
3. Applying akin methodologies to analogue filters for low-voltage operation and electronic adjustment of their frequency characteristics [36, 37], where the NN structure and training algorithms should be properly built for efficiency and prediction precision, may be a intriguing future research path.
4. A future potential work might be to investigate the utilization of a BIWASD-based NN in industrial applications [38, 39].

References

[1] SHAMSUZZOHA, M. [ed.] (2018) *PID Control for Industrial Processes* (IntechOpen). doi:10.5772/intechopen.69592.

[2] TABISH, M., KALAM, A. and ZAYEGH, A. (2019) Robot DC servo motor parameters estimation in a closed loop using BAT optimisation algorithm. In *2019 International Conference on Electrical,*

Table 3. BIWASD's stability on 100 repetitions.

NN Model	BIWASDNN					
Data	Avg. len(N)	σ(len(N))	Avg. R^2	Avg. MAPE	Avg. MAE	Avg. RMSE
Plant 1 Train	16.17	0.5514	0.9954	0.2019	0.0106	0.1431
Plant 1 Test			0.9976	0.0959	0.0072	0.1023
Plant 2 Train	16.01	0.1	0.9962	0.4460	0.0396	0.1248
Plant 2 Test			0.9970	0.3414	0.0371	0.1099
Plant 3 Train	16.02	0.2	0.9343	13.3563	0.6951	1.6735
Plant 3 Test			0.9347	13.1731	0.6921	1.6688

Communication, and Computer Engineering (ICECCE). doi:10.1109/icecce47252.2019.8940713.

[3] Sharma, K. and Palwalia, D.K. (2017) A modified PID control with adaptive fuzzy controller applied to DC motor. In *2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)*. doi:10.1109/icomicon.2017.8279151.

[4] Simos, T.E., Katsikis, V.N. and Mourtas, S.D. (2022) Multi-input bio-inspired weights and structure determination neuronet with applications in European Central Bank publications. *Mathematics and Computers in Simulation* **193**: 451–465. doi:10.1016/j.matcom.2021.11.007.

[5] Simos, T.E., Katsikis, V.N. and Mourtas, S.D. (2021) A fuzzy WASD neuronet with application in breast cancer prediction. *Neural Computing and Applications* doi:10.1007/s00521-021-06572-9.

[6] Simos, T.E., Mourtas, S.D. and Katsikis, V.N. (2021) Time-varying Black–Litterman portfolio optimization using a bio-inspired approach and neuronets. *Applied Soft Computing* **112**: 107767. doi:10.1016/j.asoc.2021.107767.

[7] Katsikis, V.N., Mourtas, S.D., Stanimirović, P.S., Li, S. and Cao, X. (2022) Time-varying mean–variance portfolio selection problem solving via LVI-PDNN. *Computers & Operations Research* **138**: 105582. doi:10.1016/j.cor.2021.105582.

[8] Katsikis, V.N., Stanimirović, P.S., Mourtas, S.D., Xiao, L., Karabasević, D. and Stanujkić, D. (2021) Zeroing Neural Network with Fuzzy Parameter for Computing Pseudoinverse of Arbitrary Matrix. *IEEE Transactions on Fuzzy Systems* : 1–1doi:10.1109/tfuzz.2021.3115969.

[9] Katsikis, V.N., Mourtas, S.D., Stanimirović, P.S. and Zhang, Y. (2021) Solving Complex-Valued Time-Varying Linear Matrix Equations via QR Decomposition With Applications to Robotic Motion Tracking and on Angle-of-Arrival Localization. *IEEE Transactions on Neural Networks and Learning Systems* : 1–10doi:10.1109/tnnls.2021.3052896.

[10] Katsikis, V.N., Mourtas, S.D., Stanimirović, P.S. and Zhang, Y. (2021) Continuous-Time Varying Complex QR Decomposition via Zeroing Neural Dynamics. *Neural Processing Letters* doi:10.1007/s11063-021-10566-y.

[11] Lu, H., Jin, L., Luo, X., Liao, B., Guo, D. and Xiao, L. (2019) RNN for solving perturbed time-varying underdetermined linear system with double bound limits on residual errors and state variables. *IEEE Transactions on Industrial Informatics* **15**: 5931–5942. doi:10.1109/tii.2019.2909142.

[12] Zhang, Y., Wang, Y., Li, W., Chou, Y. and Zhang, Z. (2016) WASD algorithm with pruning-while-growing and twice-pruning techniques for multi-input Euler polynomial neural network. *International Journal on Artificial Intelligence Tools* **25**(02): 1650007.

[13] Li, J., Cheng, J., Shi, J. and Huang, F. (2012) Brief Introduction of Back Propagation (BP) Neural Network Algorithm and its Improvement. In *Advances in Computer Science and Information Engineering* (Springer, Berlin, Heidelberg), *Advances in Intelligent and Soft Computing* **169**: 553–558. doi:10.1007/978-3-642-30223-7_87.

[14] Han, T., Lu, Y., Zhu, S. and Wu, Y.N. (2017) Alternating Back-Propagation for Generator Network. In Singh, S.P. and Markovitch, S. [eds.] *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA* (AAAI Press): 1976–1984. URL `http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14784`.

[15] Zhang, Y., Chen, D. and Ye, C. (2019) *Deep Neural Networks: WASD Neuronet Models, Algorithms, and Applications* (CRC Press).

[16] Gao, S., Zhang, Y., Zhang, Y. and Zhang, G. (2020) Elman neural network soft-sensor model of pvc polymerization process optimized by chaos beetle antennae search algorithm. *IEEE Sensors Journal* : 1doi:10.1109/JSEN.2020.3026550.

[17] Khan, A.H., Cao, X., Li, S., Katsikis, V.N. and Liao, L. (2020) BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer. *IEEE/CAA Journal of Automatica Sinica* **7**(2): 461–471.

[18] Li, X., Zang, Z., Shen, F. and Sun, Y. (2020) Task offloading scheme based on improved contract net protocol and beetle antennae search algorithm in fog computing networks. *Mobile Netw Appl* doi:10.1007/s11036-020-01593-5.

[19] Sun, Y., Zhang, J., Li, G., Wang, Y., Sun, J. and Jiang, C. (2019) Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes. *International Journal for Numerical and Analytical Methods in Geomechanics* **43**(4): 801–813. doi:10.1002/nag.2891.

[20] Cheng, Y., Li, C., Li, S. and Li, Z. (2020) Motion planning of redundant manipulator with variable joint velocity limit based on beetle antennae search algorithm. *IEEE Access* **8**: 138788–138799. doi:10.1109/ACCESS.2020.3012564.

[21] Li, X., Jiang, H., Niu, M. and Wang, R. (2020) An enhanced selective ensemble deep learning method for rolling bearing fault diagnosis with beetle antennae search algorithm. *Mechanical Systems and Signal Processing* **142**: 106752.

[22] Katsikis, V.N., Mourtas, S.D., Stanimirović, P.S., Li, S. and Cao, X. (2020) Time-varying minimum-cost portfolio insurance under transaction costs problem via beetle antennae search algorithm (BAS). *Applied Mathematics and Computation* **385**: 125453.

[23] Katsikis, V.N., Mourtas, S.D., Stanimirović, P.S., Li, S. and Cao, X. (2021) Time-Varying Mean-Variance Portfolio Selection under Transaction Costs and Cardinality Constraint Problem via Beetle Antennae Search Algorithm (BAS). *SN Operations Research Forum* **2**(18). doi:https://doi.org/10.1007/s43069-021-00060-5.

[24] Mourtas, S.D. and Katsikis, V.N. (2021) V-Shaped BAS: Applications on Large Portfolios Selection Problem. *Computational Economics* doi:10.1007/s10614-021-10184-9.

[25] DiStefano, J.J., Stubberud, A.R. and Williams, I.J. (2013) *Feedback and Control Systems*, Schaums outline series (McGraw-Hill Education), 3rd ed.

[26] Wang, L. (2020) *PID Control System Design and Automatic Tuning using MATLAB/Simulink* (Wiley-IEEE Press). doi:10.1002/9781119469414.

[27] Zhang, Y., Yu, X., Xiao, L., Li, W., Fan, Z. and Zhang, W. (2013) Weights and structure determination of articial neuronets. In *Self-Organization: Theories and Methods* (New York, NY, USA: Nova Science).

[28] Zeng, T., Zhang, Y., Li, Z., Qiu, B. and Ye, C. (2020) Predictions of USA Presidential Parties From 2021 to 2037 Using Historical Data Through Square Wave-Activated WASD Neural Network. *IEEE Access* **8**: 56630–56640. doi:10.1109/ACCESS.2020.2982192.

[29] Chen, L., Huang, Z., Li, Y., Zeng, N., Liu, M., Peng, A. and Jin, L. (2019) Weight and Structure Determination Neural Network Aided With Double Pseudoinversion for Diagnosis of Flat Foot. *IEEE Access* **7**: 33001–33008. doi:10.1109/ACCESS.2019.2903634.

[30] Jiang, X. and Li, S. (2017) BAS: Beetle Antennae Search Algorithm for Optimization Problems. *arXiv preprint* **abs/1710.10724**. URL `http://arxiv.org/abs/1710.10724`. `1710.10724`.

[31] Katsikis, V.N. and Mourtas, S.D. (2021) *Computational Management* (Springer, Cham.), *Modeling and Optimization in Science and Technologies* **18**, chap. Portfolio Insurance and Intelligent Algorithms, 305–323. doi:10.1007/978-3-030-72929-5_14.

[32] Katsikis, V.N. and Mourtas, S.D. (2021) Binary Beetle Antennae Search Algorithm for Tangency Portfolio Diversification. *Journal of Modeling and Optimization* **13**(1): 44–50. doi:10.32732/jmo.2021.13.1.44.

[33] Katsikis, V.N. and Mourtas, S.D. (2020) Optimal portfolio insurance under nonlinear transaction costs. *Journal of Modeling and Optimization* **12**(2): 117–124.

[34] Medvedeva, M.A., Katsikis, V.N., Mourtas, S.D. and Simos, T.E. (2020) Randomized time-varying knapsack problems via binary beetle antennae search algorithm: Emphasis on applications in portfolio insurance. *Math Meth Appl Sci* : 1–11doi:https://doi.org/10.1002/mma.6904.

[35] Khan, A.H., Cao, X., Katsikis, V.N., Stanimirovic, P., Brajevic, I., Li, S., Kadry, S. *et al.* (2020) Optimal Portfolio Management for Engineering Problems Using Nonconvex Cardinality Constraint: A Computing Perspective. *IEEE Access* : 1–1doi:10.1109/access.2020.2982195.

[36] Kasimis, C. and Psychalinos, C. (2012) Design of Sinh-Domain filters using complementary operators. *International Journal of Circuit Theory and Applications* **40**: 1019–1039. doi:10.1002/cta.769.

[37] Psychalinos, C., Kasimis, C. and Khateb, F. (2018) Multiple-input single-output universal biquad filter using single output operational transconductance amplifiers. *AEU - International Journal of Electronics and Communications* **93**: 360–367. doi:10.1016/j.aeue.2018.06.037.

[38] Luo, X., Zhou, M., Li, S., Wu, D., Liu, Z. and Shang, M. (2021) Algorithms of unconstrained non-negative latent factor analysis for recommender systems. *IEEE Transactions on Big Data* **7**(1): 227–240. doi:10.1109/TBDATA.2019.2916868.

[39] Luo, X., Zhou, M., Shang, M., Li, S. and Xia, Y. (2016) A novel approach to extracting non-negative latent factors from non-negative big sparse matrices. *IEEE Access* **4**: 2649–2655. doi:10.1109/ACCESS.2016.2556680, URL `https://ieeexplore.ieee.org/document/7457202/`.