Constructing Process Identification Simulator Based on Artificial Neural Network

Dr. Saad AL-Shaban, Dr. Saad AL-Sakafi* & Zainab Saeed Abood**

Received on: 27/7/2004
Accepted on: 3/1/2005

Abstract

The aim of this paper is to build a process simulator to aid the process and control engineers in design and simulation. The simulator is to be built of software modules. The basic modules are integrator and dead time, lag, and two-lag. Practical process built from the basic one are deadtime and integrator, deadtime and lag, deadtime and two-lag. The present work is achieved by using Levenberg-Marqardet algorithm with variable learning rate, the software used in this work is implemented by using Turbo (C language) with technical MATLAB version 6.0 package.

الخلاصة

تم استخدام الشبكات العصبية كأداة برمجية لتحديد أربعة أنظمة ، نظام التكامل،نظام التأخير الزمني،نظام التخلف الزمني، و نظام التخلف الزمني من الدرجة الثانية.مع ثلاث إشارات مدخلة للفحص (ramp, step, impulse).

ثم تم بناء ثلاثة أنظمة من الأنظمة الرئيسية ، حيث أطلق على الأنظمة المبنية منها نظام التأخير الزمني مع نظام التكامل، ونظام التأخير الزمني مع نظام التخلف الزمني، ونظام التأخير الزمني مع نظام التخلف الزمني من الدرجة الثانية.

Keywords: Process identification, Neural network, Levenberg-Marquardt Learning Algorithm, Process simulator, Integrator process, dead time process, lag process .

1-Introduction

Soft computing is a practical alternative for solving computationally complex and mathematically intractable problems especially system identification problems. The most popular constituents of the soft computing methodologies are the neural network *[1]*. Neural networks are non-linear black-box model structures, to be used with conventional parameter estimation methods. They have good general approximation capabilities for reasonable non-linear systems. When estimating the parameters in these structures, there is also good adaptability to concentrate on those parameters that have the most importance for the particular data set *[2]*, nonparametric identification process it is a technique that is demand in this work to estimate model behavior without necessarily using a given parameterized model set with correlation analysis, which estimates a system's impulse response *[3]*.

2-Neural Network and Learning Algorithm

The learning algorithm that is used to train the network is commonly levenberg-marquardt a version of back propagation algorithm used to train multi-layer feed forward networks based on nonlinear optimization technique by minimizing the sum of squares of errors (SSE) *[4]*. This method is based on an approximating the second

Department of Computer Eng. & Information Technology, University of Technology, Baghdad-IRAQ.

https://doi.org/10.30684/etj.24.2A.9
2412-0758/University of Technology-Iraq, Baghdad, Iraq

order derivatives by using the first order derivatives *[5]*, with batch update and variable learning rate, this method represents a simple heuristic strategy to increase the convergence speed of the back propagation algorithm, the flow chart that describe the algorithm that is used to achieve this work is shown in figure(1).the LM algorithm as follow *[6]*:-

Step1. Initialize the network weight to small random values and biases.

Step2. Present an input pattern. And calculate the output of the network.

$$a_1 = f_1(\sum w_1 p_1 + b_1)$$
$$a_2 = f_2(\sum w_2 a_1 + b_2)$$

Where

f_1, f_2 : Represents the activation functions of hidden and output layer respectively.

Step3. Calculate the elements of the Jacobian matrix associated with Input/output pairs by using this equation.

$$w(k+1) = w(k) + \left[J_k^T J_k + \mu I\right]^{-1} J_k^T e_k$$

Step4. Update the weight according to the following term.

$$J_{ij} = \frac{\Delta e_i}{\Delta w_j}$$

*Step5.*Stop if the network has convergence; else, go back to step 2.

3-Neural Identifier

Neural networks are used for identifying the behavior of a process simulator device, the data taken from the processes off-line, then applied to the neural network software tool figure (2). Four identifiers were used to identify four processes that describe in this device as follows:

NNI: -for integrator process.
NND: - for dead time process.
NNL: - for lag process
NNLL: - for two-lag process

After identifying these four basic processes, we can construct practical process from them. Then test signals were applied to test and establish the process behavior. Three-test signals are used impulse, step, and ramp.

3-1 NNI Process

The NNI structure details are shown in figure (3.a), one node in the input layer, one hidden layer with fifteen nodes and three nodes in the output layer. During training phase the training set of the signal is presented many times. A training corresponding to signal passing over the entire training set is called training epoch or training cycle. In order to have good mapping of a neural model to the plant response many experiments have been done including changes in the number of hidden units, and the momentum parameter one of the experiments represents good mapping sum squared error as shown in figure (3.b) shows the SEE Vs epochs and figure (3) c, d, e shows the actual output of the integrator process that is represented by (-) and the NNI output that is represented by (o), for three signals respectively.

The learning rate is varying according to the value of error before and after updating the parameters in each iteration. One advantage of Levenberg-Marguardate algorithm is that, not all the iteration in LM algorithm is used to update the network parameters, just the iteration, which decreases the error, is used in updating the network parameters. Therefore the error will never increase through the learning process, and hence it will have a stair-like performance surface *[5]*.

Simulation results showed that, LM training algorithm could reach any degree of accuracy with more iteration and degree of freedom in the hidden layers, equation (1) represent the NNI process:-

$$V_{out}(S)=\frac{1}{S}V_{in}(S) \qquad (1)$$

where:

$V_{in}(S)$: represents the input voltage.

$V_{out}(S)$: represents the output voltage.

$\frac{1}{RCS}$: represents the integral value.

S : represents the Laplace symbol.

3-2 NND Process

Neural network is used for identifying dead time process, the input for NND consists of (228) patterns each one has two values one for (D) value, (D) range is (0-9) NND trained for D=(1,5,7,10), the second value represents the input signal value. NND a structure detail is shown in figure (4.a), two nodes in input layer, and one hidden layer with ten nodes and one node in output layer. figure (4.b) shows the SEE that is reached by using LM on NND model figures(4) c, d, e show the response of the three signals, equation (4) represent the NND process :-

$$V_{out}(S)=V_{in}(S)e^{-DS} \qquad (2)$$

where:

$V_{in}(S)$: represents the input voltage.

$V_{out}(S)$: represents the output voltage.

D: represents the dead time value.

S: represents the Laplace symbol.

3-3 NNL Process

Neural network is used for identifying lag process, the input for NNL consists of (285) patterns each one has two values one for (τ) value, (τ) range (0-9), NNL trained for (τ=1,3,5,7,10), the second value represents the input signal time value. The structure detail is shown in figure (5.a), two nodes in input layer, and one hidden layer with fifty-five nodes and three nodes in output layer.

The sum squared error and the training epochs are shown in figure (5.b).The NNL output is shown in fig (5) c, d, e three signals with different τ values, equation (3) represent the NNL process:-

$$V_{out}(S)=\frac{1}{\tau S+1}V_{in}(S) \qquad (3)$$

where:

$V_{in}(S)$: represents the input voltage.

$V_{out}(S)$: represents the output voltage.

τ: represents the lag time value voltage.

S : represents the Laplace symbol.

3-4 NNLL Process

Neural network is used for identifying a two-lag process, the input for NNLL consists of (231) patterns each one has two values one for (τ) it ranges (0-9), NNLL trained for (τ=1,2,4,6,8,10) values, the second value represents the input signal value. The structure detail is shown in figure (6.a), two nodes in input layer, and two hidden layer, the first one with fifty -three nodes and the second one with fifty –two nodes with fifty-five nodes and three nodes in output layer. The sum squared error and the training epochs are shown in figure (6.b). The NNL output is shown in fig (6) c, d, e three signals with different τ values, equation (4) represent the NNLL process:-

$$V_{out}(S)=\frac{1}{\tau^2S^2+\tau S+1}V_{in}(S) \qquad (4)$$

4-Simulation Results

Four processes have been built using neural networks, they can be used to construct additional processes, these processes can be built and tested by applying one of the input tested signals ramp, step or impulse to identify the behavior of that process. Simulation results show the abilities of neural networks to adapt the new processes and to give good results. Three new processes are obtained NNDI, NNDL and NNDLL.

4-1 NNDI Process Simulate

This process was built from both dead time NND and integrator NNI processes, figure (6.a) shows this process. For impulse input with D=3second, NNDI gives sum square error (SSE=0.0068), impulse response is shown in figure (7.b). For step input with D=6 second, NNDI gives sum square error (SSE=0.0433), step response is shown in figure (7.c).For ramp input with D=2 second, the NNDI give sum square error (SSE=0.0160), Ramp response shown in figure (7.e).Figure (7.f) shows the error curves for 57 samples for three examples of signals that were selected to test the process, equation (5) represent the NNDI process:-

$$V_{out}(S) = \left(\frac{e^{-\tau S}}{S}\right) V_{in}(S) \qquad (5)$$

where:

$V_{in}(S)$: represents the input voltage.

$V_{out}(S)$: represents the output.

τ: represents the lag time value voltage.

S : represents the Laplace symbol.

4-2 NNDL Process Simulate

NND and lag NNL processes were used for constructing this process figure (8.a) shows this process. The NNDL gives approximation results which agree with the original one NNDI tested by applying three tested signals, to each dead time (D) and (τ) to each signal NNDL produced different sum square error For Impulse input with D=2 second, τ=1second, NNDL gives sum square error (SSE=3.4106e-5), impulse response is shown in figure (6.b). For step input with D=4 second, τ=9 second, NNDI gives sum square error (SSE=1.2301e-4), step response is shown in figure (8.c). For ramp input with D=2 second, τ=3second. The NNDI gives sum square error (SSE=1.1934e-4), ramp response is shown in figure (8.e).Figure (8.f) shows the error curves for 57 samples for three examples of signals that were selected to test the process, equation (6) represent the NNDL process:-

$$V_{out}(S) = \left(\frac{e^{-\tau S}}{\tau S + 1}\right) * V_{in}(S) \qquad (6)$$

4-3 NNDLL Process Simulate

NNDLL process was built from NND with two lags NNLL processes, figure (9.a) shows this process. The new process NNDLL gives approximation results which agree with the original one, NNDLL was tested by applying three tested signals, to each dead time (D) and (τ) to each signal NNDLL produced different sum square errors. For impulse input with D=2.5 second, τ=9.5 second. NNDLL gives sum square error (SSE=2.1471e-7), impulse response is shown in figure (9.b).For step input with D=5 second, τ=2second, NNDLL gives sum square error (SSE=2.8050e-6), step response is shown in figure (9.c).For ramp input with D=4second, τ=7second. NNDLL gives sum square error (SSE=2.0603e-6), , ramp response is shown in figure (9.d). Figure (9.e) shows the error curves for 57 samples for three examples signals that were selected to test the process, equation (7) represent the NNDLL process:-

$$V_{out}(S) = \left(\frac{e^{-\alpha S}}{\tau^2 S^2 + 2\tau S + 1}\right) * V_{in}(S) \qquad (7)$$

5-Conclusions

The following points outline the most important results that have been obtained in this work.

1-The Process simulator device here is replaced by four neural networks, so many problems are avoided such as:

- The maintenance problem: -Since NNs is implemented in software method and the process simulator device is a hardware tool, no maintenance is needed.

- The modification problem:-The process simulator device needs to replace any card for modification, but NNs needs small change in the written program. The time problem: The process simulator device uses wires to make a connection between signal and process.
This takes a time to prepare the process for testing while in computer program the selection is made in the program.
- The time problem: The process simulator device uses wires to make a connection between signal and process. This takes a time to prepare the process for testing while in computer program the selection is made in the program.

6-References

1-Efa, M. okyay,"*A comparative study of neural network structures in identification of non-linear systems*", 1997.

2-Sjoberg J. H. Hjalmarsson L. Ljung,"*Neural Networks in System Identification*", Department of Electrical Engineering Linkping University S-851 Linkping, Sweden 1989.

3-Pham Liu,"*Neural networks for identification prediction and control*", London:Springer-Verlag, 1995.

4-Rasha R. A. Izzat," *Neural networks for noise canceling applications*", Department of Computer Engineering, AL-Nahreen University 2000.

5-Eric W. Weisstein, "*Levenberg-Marquardt method*", From Math World-A Wolfram Web Resource..http://mathworld.wolfram.com/Levenberg-Marquardt method.html, 1999-2004 Wolfram Research,Inc.

6-Fredric M. Ham, Ph.D, Ivica Kostanic, *"Principles of neurocomputing for science and engineering"*, McGraw-Hill Higher Eduction ISBN 0-07118161.x, 2001.

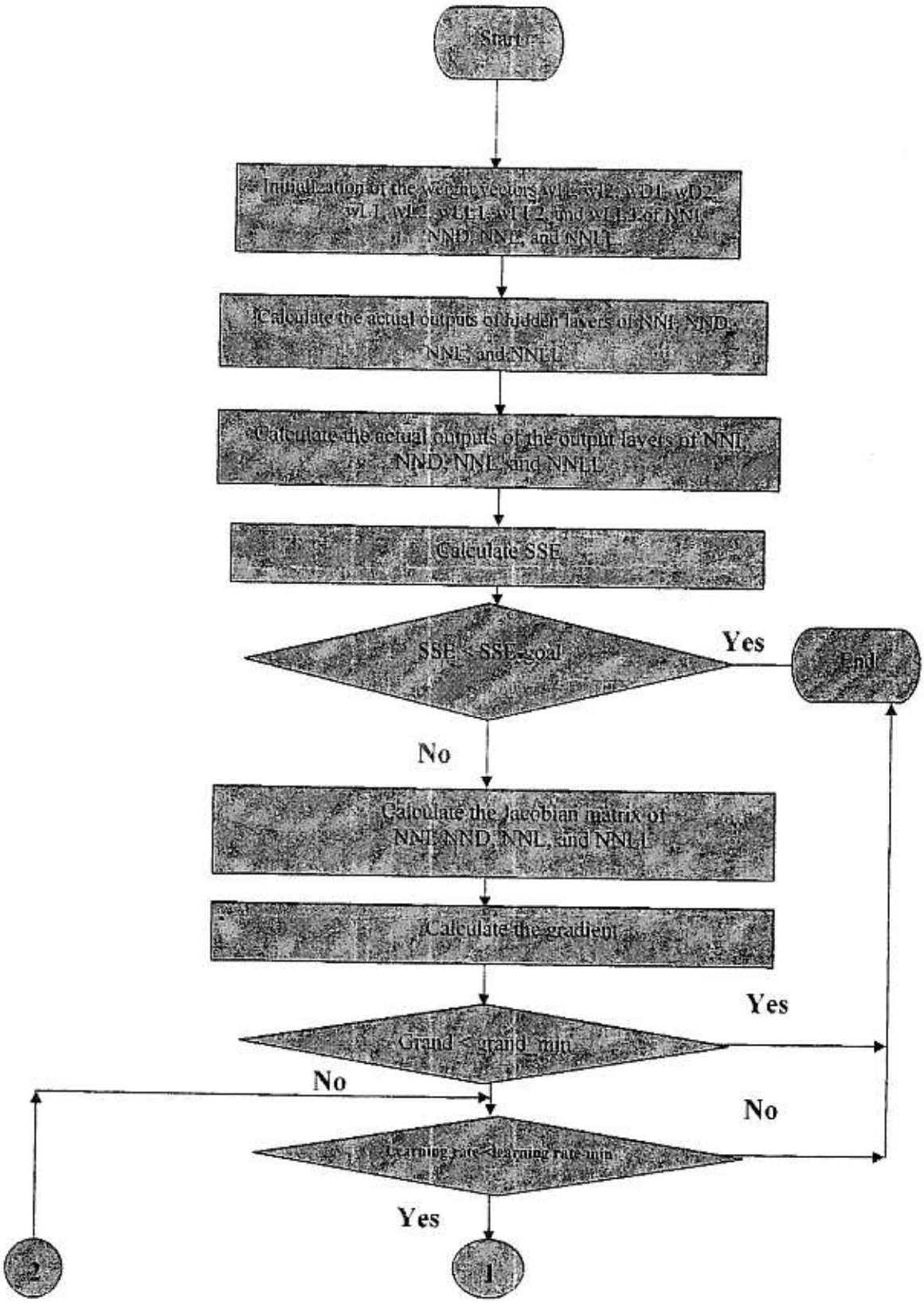
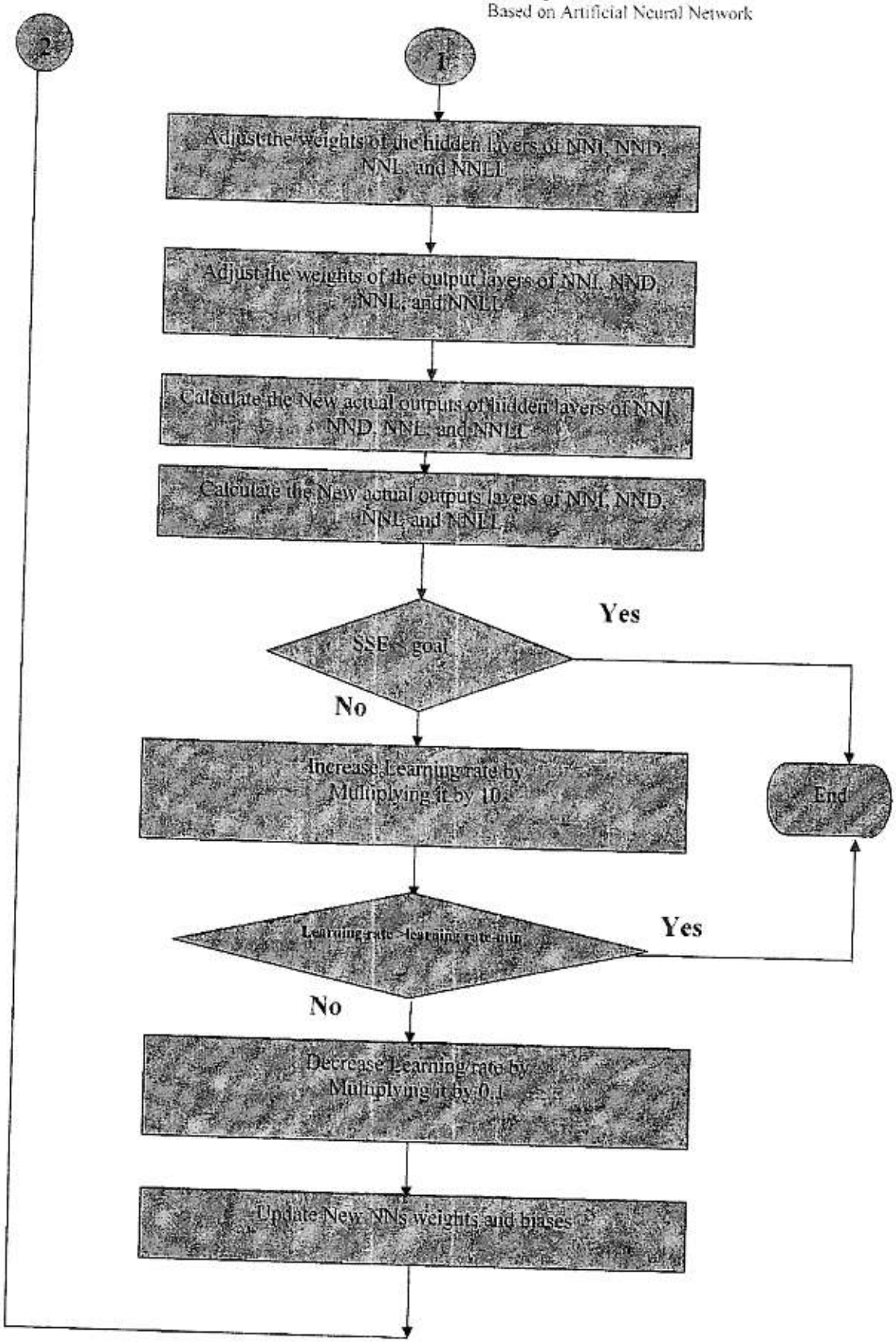


Fig (1) Flowchart
Of LM training algorithm with variable learning rate


2
1
Adjust the weights of the hidden layers of NNI, NND, NNL, and NNLL
Adjust the weights of the output layers of NNI, NND, NNL, and NNLL
Calculate the New actual outputs of hidden layers of NNI, NND, NNL, and NNLL
Calculate the New actual outputs layers of NNI, NND, NNL, and NNLL
SSE < goal
Yes
No
Increase Learning rate by Multiplying it by 10
End
Learning rate > learning rate min
Yes
No
Decrease Learning rate by Multiplying it by 0.1
Update New NNs weights and biases

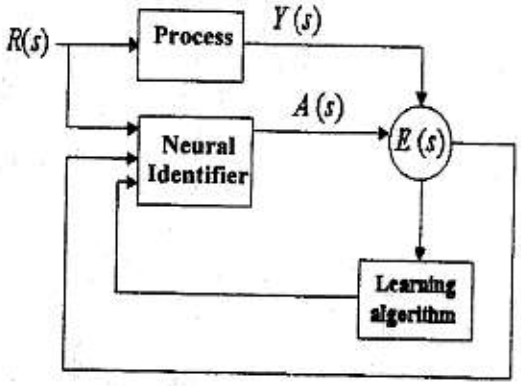


Fig (2) Neural identifier

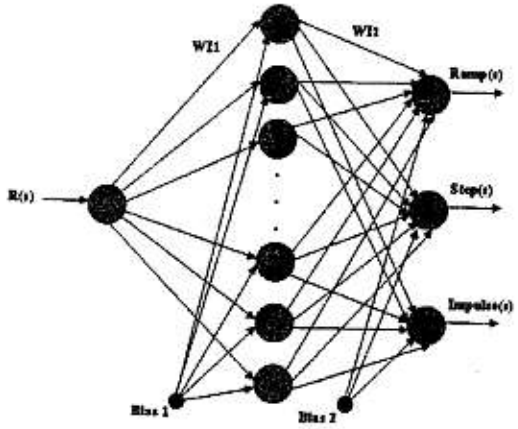


Fig.(3.a) NNI Process

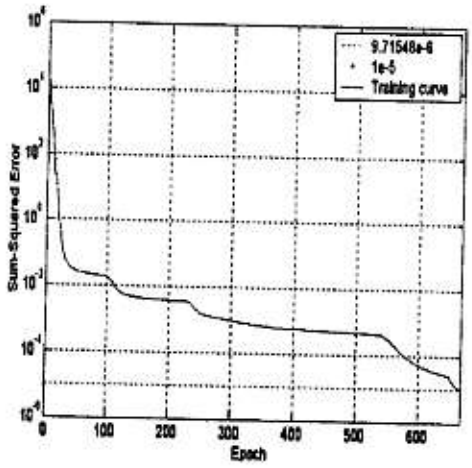


Fig. (3.b) SSE for NNI process

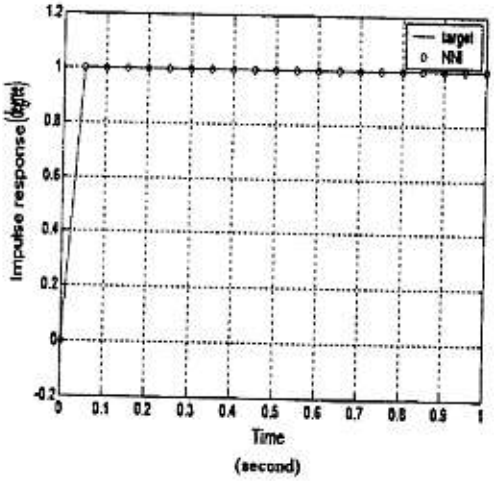


Fig. (3.c)Impulse response for NNI process

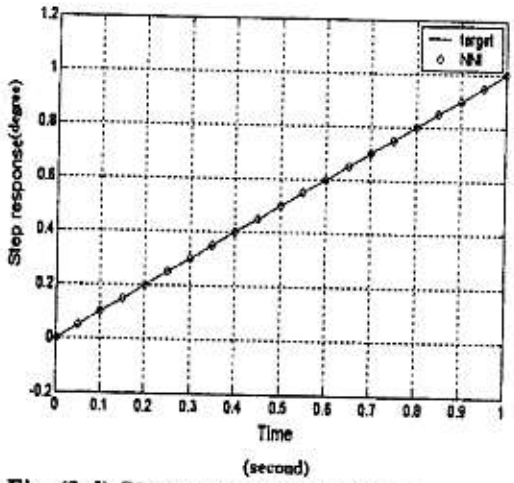


Fig. (3.d) Step response for NNI process

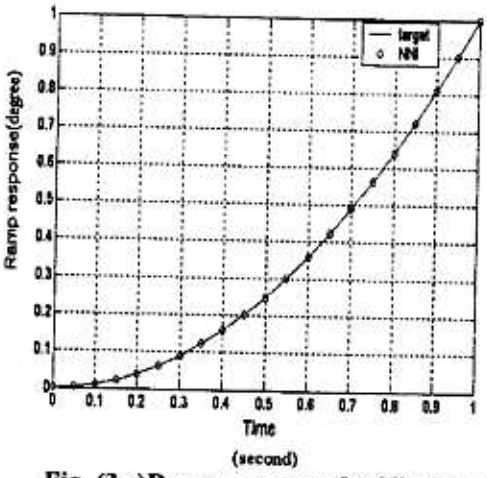


Fig. (3.e)Ramp response for NNI process

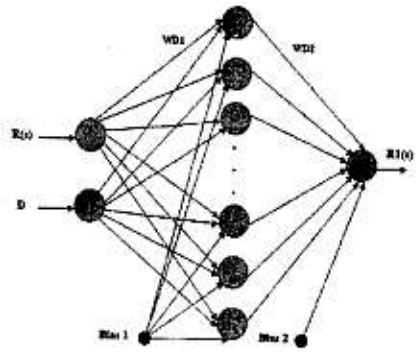


Fig.(4.a)NND Process

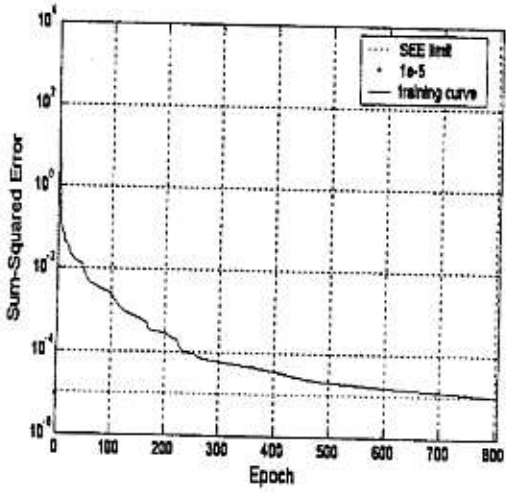


Fig. (4.b) SSE for NND process

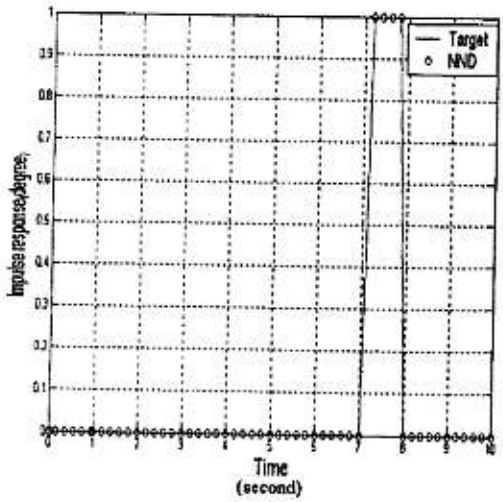


Fig. (4.c)Impulse response for NND process

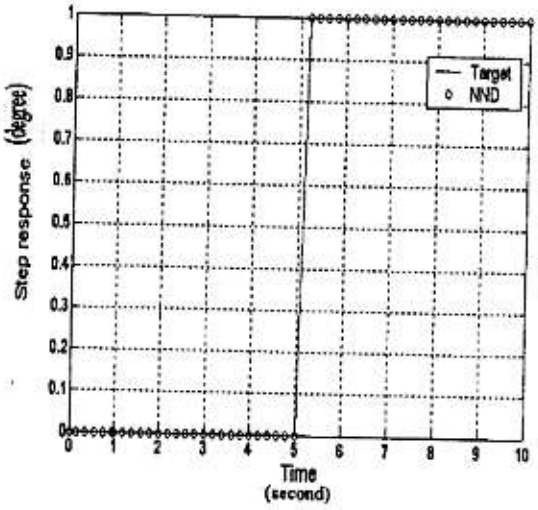


Fig. (4.d)Step response for NND process

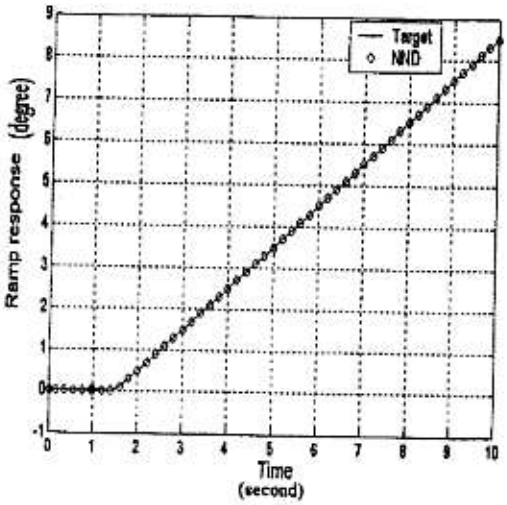


Fig. (4.e)Ramp response for NND process

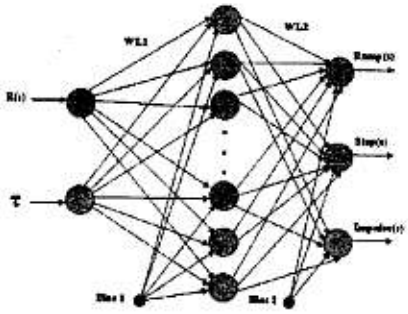


Fig.(5.a)NNL Process

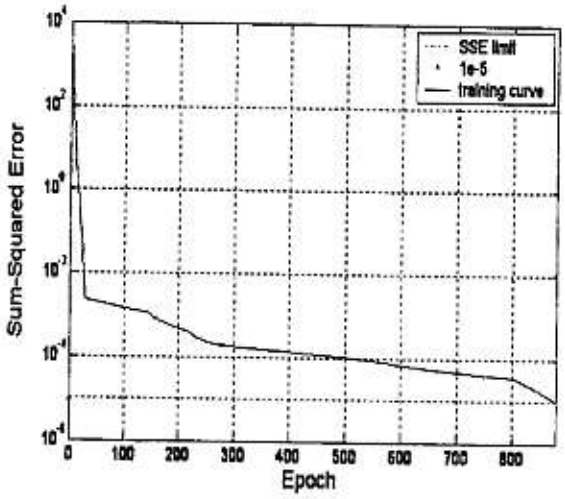


Fig.(5.b)SSE for NNL process

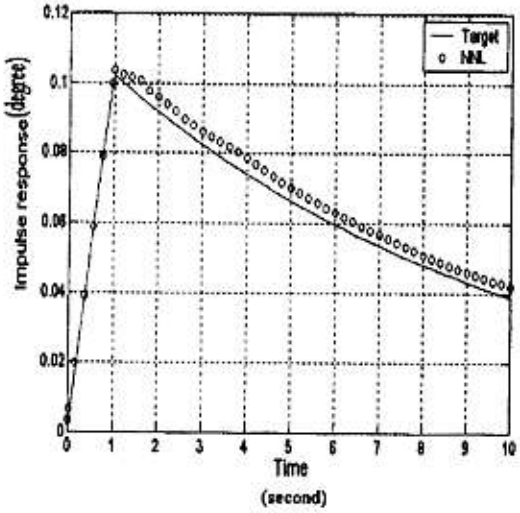


Fig.(5.c) Impulse response for NNL for τ=9.25s

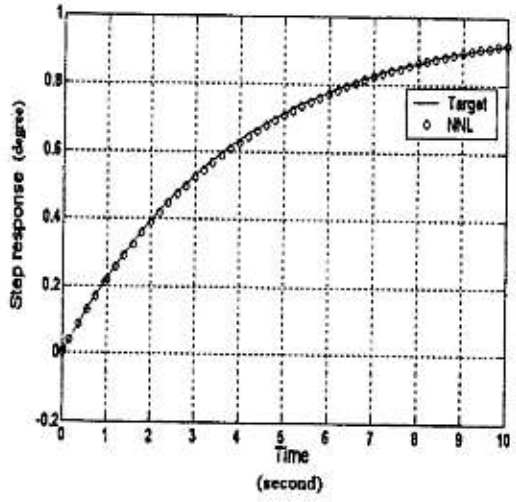


Fig.(5.d) Step response for NNL for τ=4s

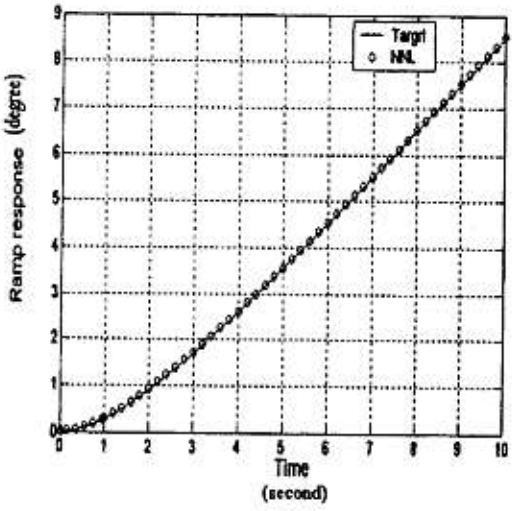
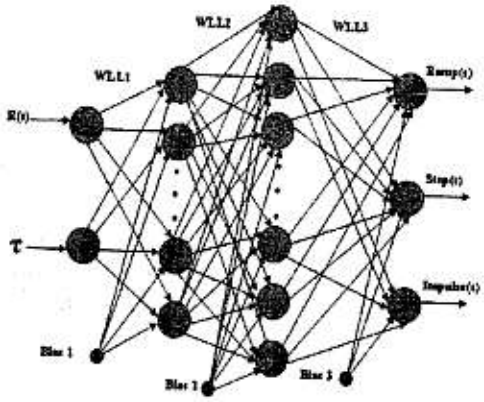


Fig.(5.e) Ramp response for NNL for τ=1.5s



Fig(6.a)NNLL Process

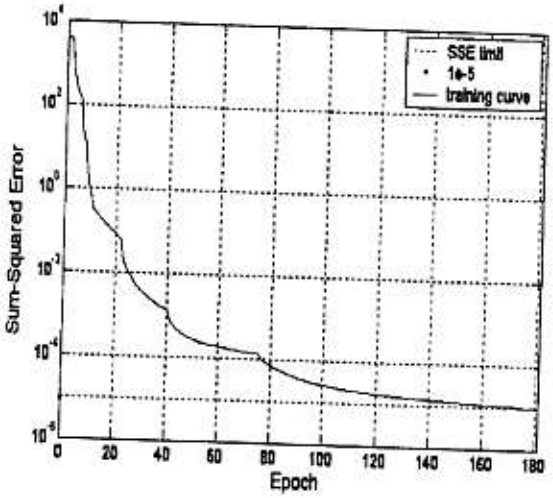


Fig.(6.b)SSE for NNLL process

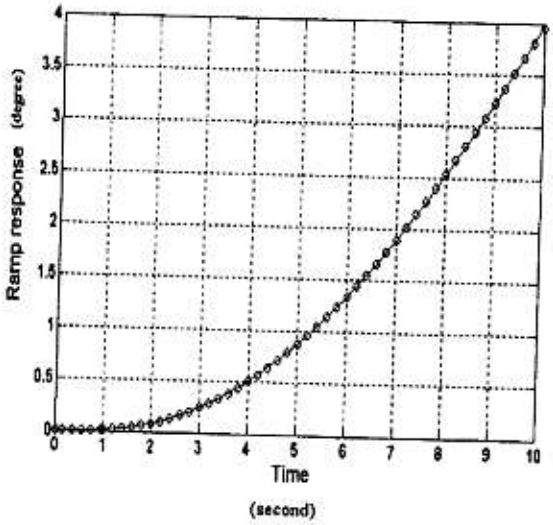


Fig (6.c) Ramp response after Lag τ =3.5s second

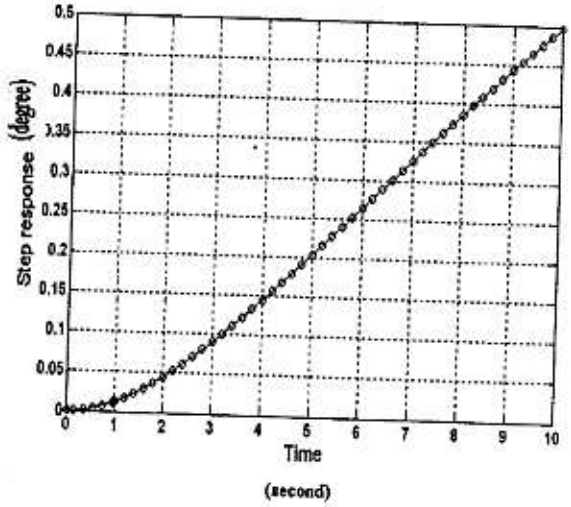


Fig (6.d) Step response after Lag τ =6 s

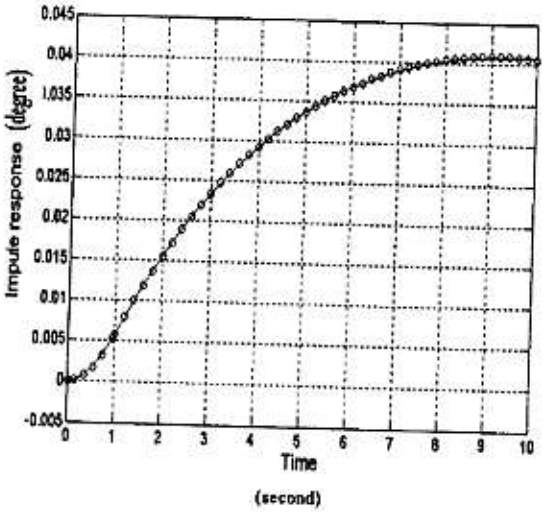


Fig.(6.e) Impulse response for NNL for τ=9s

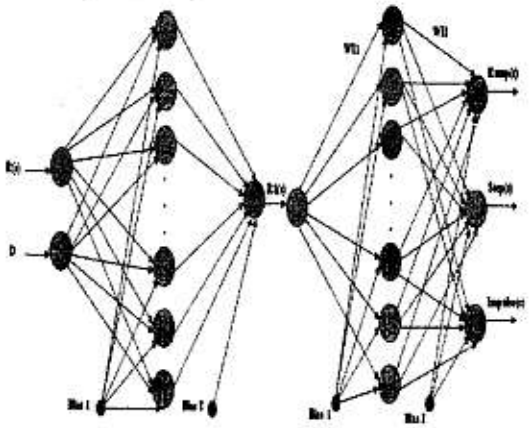


Fig. (7.a) NNDI process

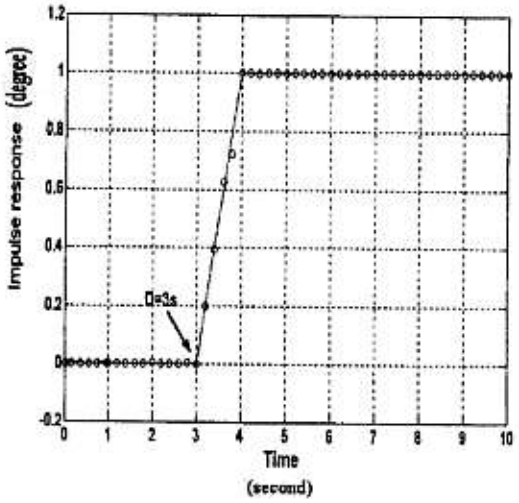


Fig. (7.b) Impulse response for NNDI process

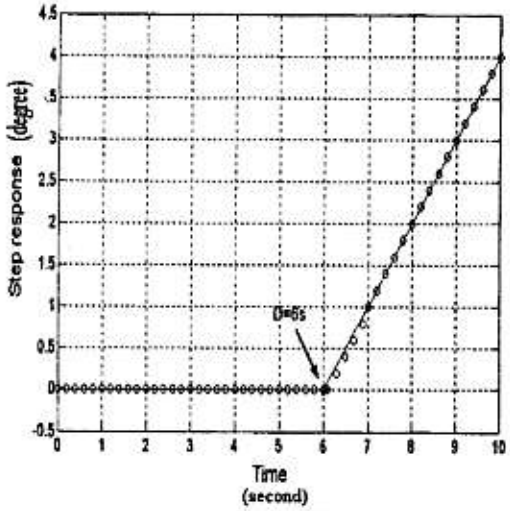


Fig. (7.c)Step response for NNDI process

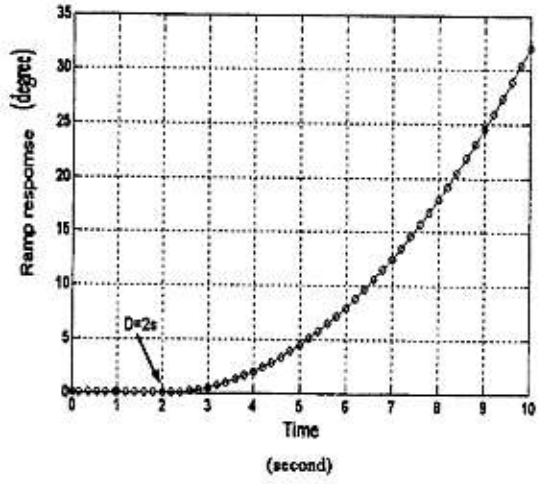


Fig. (7.d) Ramp response for NNDI

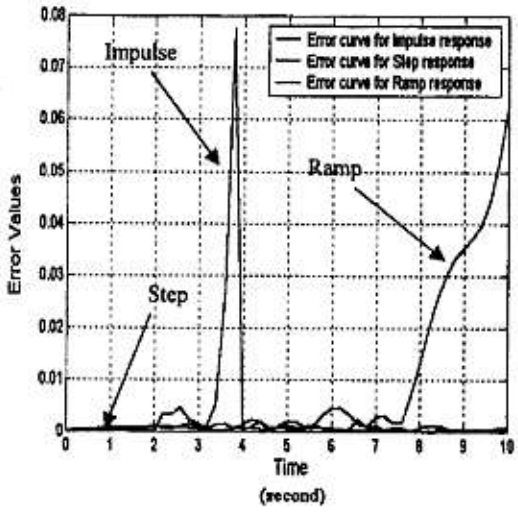


Fig. (7.e)Error curve for three tested signals

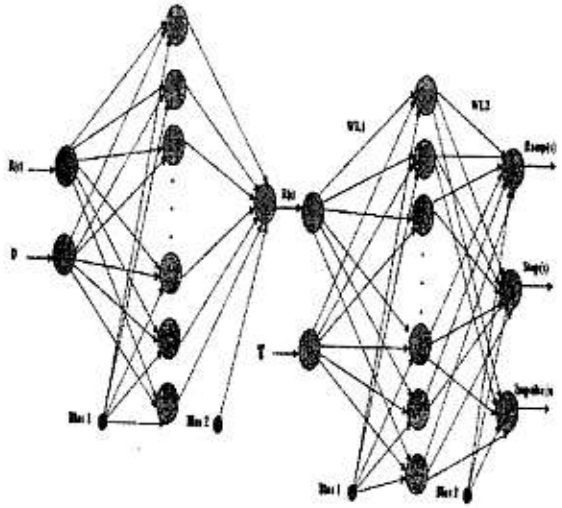


Fig. (8.a) NNDL process

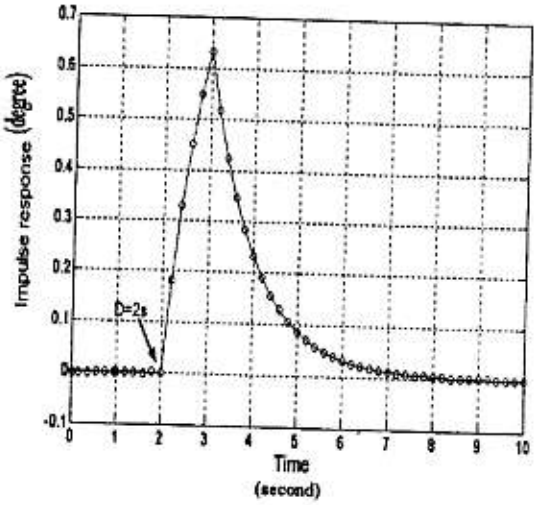


Fig. (8.b) Impulse response for NNDL

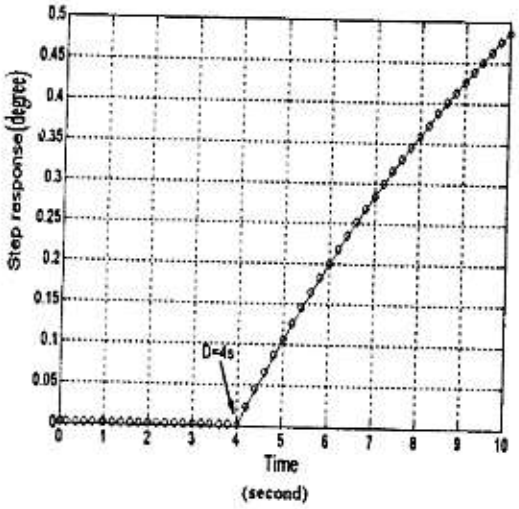


Fig. (8.c) Step response for NNDL signals

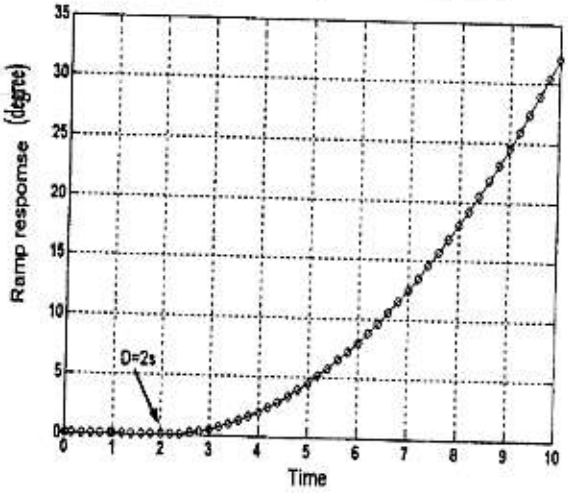


Fig. (8.d) Ramp response for NNDL

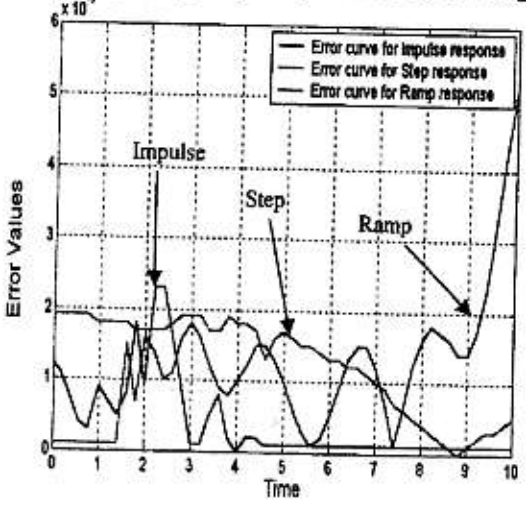


Fig. (8.e)Error curve for three tested signals

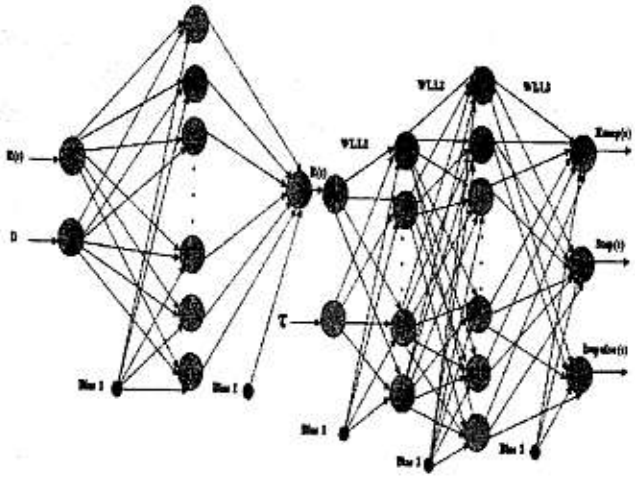


Fig. (9.a) NNDLL process

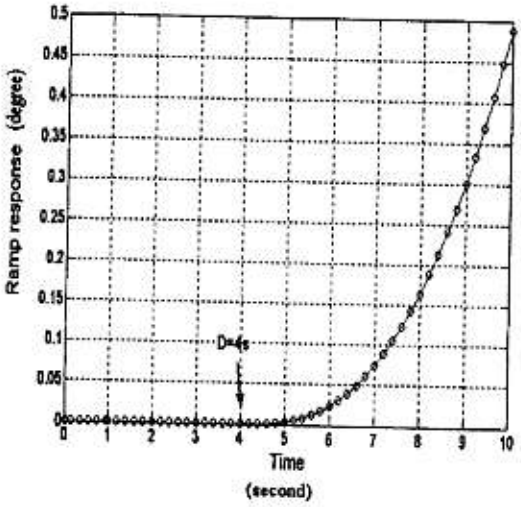


Fig. (9.b) Impulse response for NNDLL

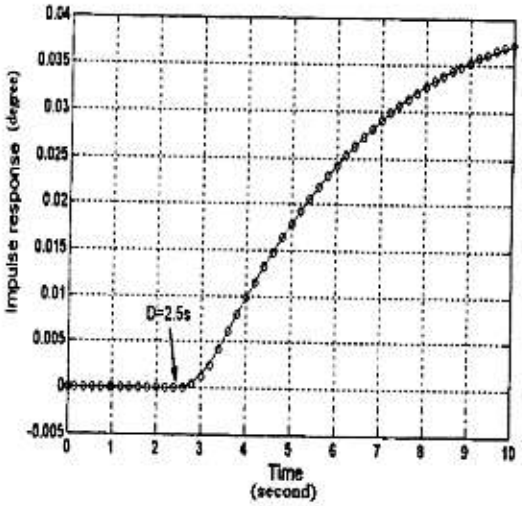


Fig. (9.c)Step response for NNDLL

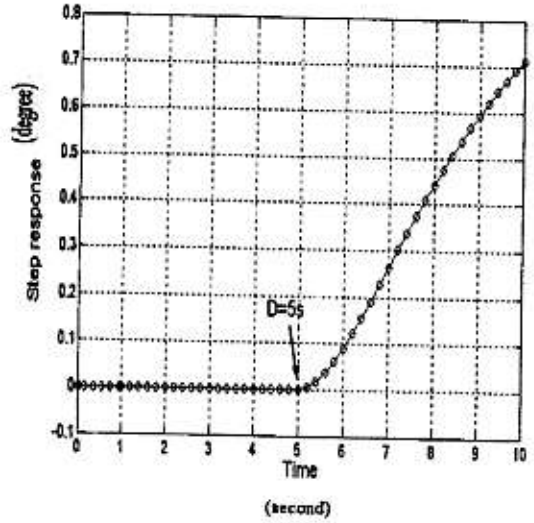


Fig. (9.d) Ramp response for NNDLL

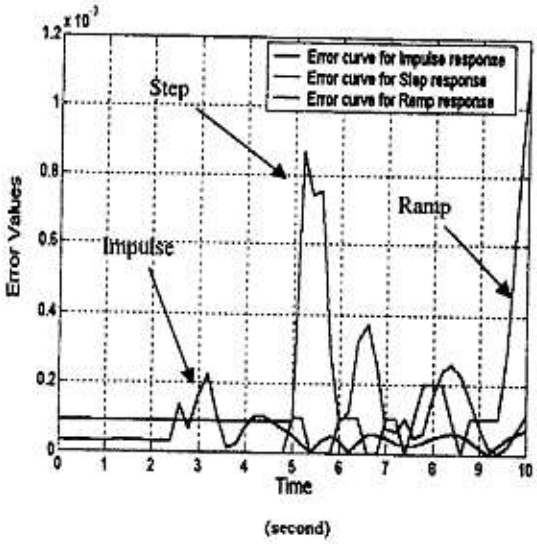


Fig. (9.e) Error curve for three signals