Nonlinear System Identification using Neural Networks and Trajectory-based Optimization

Hamid Khodabandehlou[1] [a] and M. Sami Fadali[2] [b]
[1]*Amgen Inc., One Amgen Centre Drive, Thousand Oaks, CA, U.S.A.*
[2]*Electrical and Biomedical Engineering Department, University of Nevada-Reno, NV, U.S.A.*

Keywords: System Identification, Neural Networks, Global Optimization, Nonlinear Benchmark.

Abstract: In this paper, we study the identification of two challenging benchmark problems using neural networks. Two different global optimization approaches are used to train a recurrent neural network to identify two challenging nonlinear models, the cascaded tanks and the Bouc-Wen system. The first approach, quotient gradient system (QGS), uses the trajectories of the nonlinear dynamical system to find the local minima of the optimization problem. The second approach, dynamical trajectory based methodology, uses two different nonlinear dynamical systems to find the connected components of the feasible region and then searches the regions for local minima of the optimization problem. Simulation results show that both approaches effectively identify the model of the cascade tanks and the Bouc-Wen model.

1 INTRODUCTION

Although engineering applications often require an accurate explicit mathematical model of the system, in many cases such a model is not available. While system models can, in theory, be derived using physical and mathematical principles, deriving such models is difficult in practice (Gautam, 2016)0. System identification is an alternative approach to modeling. System identification uses the input and output measurements of the system to derive its model. The model can be white, grey or black box (Gautam, 2016; Verdult, 2002).

Neural networks are a powerful tool for nonlinear system identification. Narendra and Parthasarathy showed that neural networks can identify and control nonlinear dynamical systems (Narendra et. al., 1990). Experimental results show that the neural network can effectively identify the forward and inverse transfer function (Yamada and Yabuta, 1993).

Efe and Kaynak studied the identification of nonlinear systems using feedforwad neural networks, radial basis function networks, Runge-Kutta neural networks and adaptive neuro-fuzzy inference systems, with application to a robotic manipulator (Efe and Kaynak, 1999). Other neural networks that have successfully identified nonlinear systems include Volterra polynomial basis function networks (Liu, 1998), wavelet networks and echo state networks (Khodabandehlou and Fadali, 2017), and partially recurrent neural networks (Pham and Liu, 1992, and 1995).

Coban proposed the Context Layered Recurrent Neural Network (CLRNN) for identification of linear and nonlinear dynamic systems (Coban, 2013). CLRNN is a multilayer recurrent neural network with a context layer. The context layer is a feedback from the first hidden layer to itself that improves the capability of the network to capture the linear behavior of the system.

Fully Recurrent Neural Networks (FRNN) are recurrent neural networks whose hidden layer nodes are all connected. FRNN's can effectively identify linear and nonlinear models. However, their complicated structure makes their training difficult and slow (Pearlmutter, 1995; Chren and Soo, 1996; Dongpo et. al., 2010). Backpropagation Through Time (BPTT) is an alternative to traditional error backpropagation for training recurrent neural networks. BPTT represents recurrent neural network as a multilayer feedforward network, then tunes its weights using backpropagation (Werbos, 1998;

[a] https://orcid.org/0000-0001-7386-2785
[b] https://orcid.org/0000-0002-3865-2499

Khodabandehlou, H. and Fadali, M.
Nonlinear System Identification using Neural Networks and Trajectory-based Optimization.
DOI: 10.5220/0007772605790586
In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019)*, pages 579-586
ISBN: 978-989-758-380-3

Stroeve, 1998, Hermans et. a., 2010). BPTT is computationally expensive and its computational time increases drastically with the size of the training dataset. It also suffers from vanishing gradient problem (Pascanu et. al., 2013). Regularization of the neural network weights is an approach to cope with the vanishing gradient problem (Pascanu et. al., 2013). Sutskever proposed Hessian free optimization to train recurrent neural networks and overcome the vanishing gradient issue (Sutskever et. al., 2011).

Williams and Zipser proposed Real Time Recurrent Learning (RTRL) for training recurrent neural networks. It does not need a precisely defined training interval its computational load is huge in large applications (Williams and Zipper, 1989). Generalized Long Short Term Memory (LSTM) is another approach for training second order recurrent neural networks (Monner and Reggia, 2012). The method is applicable to a wide range of networks and performs better than traditional LSTM.

Lu et. al. used low rank factorization to inspect redundancies in recurrent neural networks (Lu et. Al., 2016). They showed that using structured matrices and shared low-rank factors can effectively reduce the number of parameters of the standard LSTM without significantly increasing error.

Although a wide variety of algorithms have been used to train feedforward and recurrent neural networks, there are promising optimization approaches that have not been used for training and that have the potential to provide better system identification results. This paper explores the use of two trajectory-based methodologies for nonlinear system identification: the quotient gradient method and the dynamical trajectory-based approach. The quotient gradient method is a trajectory-based methodology to find the possible feasible solutions of the constraint satisfaction problem (CSP). Quotient gradient uses the trajectories of a stable nonlinear dynamical system to find the solutions of a constraint satisfaction problem (Lee and Chiang, 2001).

The Dynamical Trajectory Based approach (DTB) is another global optimization approach that is applicable to general constrained optimization problems. DTB uses the trajectories of two nonlinear dynamical systems, i.e. Projected Gradient System (PGS) and quotient Gradient System (QGS) to find disjoint components of the feasible region of the optimization problem and search those disjoint components for possible solutions of the optimization problem (Lee and Chiang, 2004).

In this study, we use the quotient the gradient method and DTB to train recurrent neural network and evaluate the performance of the networks on two of the challenging nonlinear system identification benchmarks. The first is the cascaded tank model, which is difficult to identify due to saturation and overflow in the tanks. The second is the Bouc-Wen model. The Bouc-Wen model is highly nonlinear model with hysteretic behaviour, which makes it challenging benchmark for system identification

The remainder of this paper is organized as follows: Section 2 presents the neural network structure. Section 3 describes the quotient gradient method and section 4 describes the dynamical trajectory based optimization approach. Section 5 describes the benchmark systems and Section 6 presents simulation results

2 NEURAL NETWORKS

In this study, we use a fully recurrent neural network with one hidden layer. The structure of the neural network is shown in Figure 1. For a network with n inputs, m hidden layer nodes, and t outputs, the input, internal state and output vector of the network, respectively, are:

$$\begin{aligned} \boldsymbol{u}(k) &= [u_1(k) \quad ... \quad u_n(k)]^T \\ \boldsymbol{z}(k) &= [z_1(k) \quad ... \quad z_m(k)]^T \\ \hat{\boldsymbol{y}}(k) &= [\hat{y}_1(k) \quad ... \quad \hat{y}_t(k)]^T \end{aligned} \tag{1}$$

The governing equation of the network is

$$\begin{aligned} \boldsymbol{z}(k) &= \boldsymbol{\psi}\big(W\boldsymbol{u}(k) + S\boldsymbol{z}(k-1)\big) \\ \hat{\boldsymbol{y}}(k) &= V\,\boldsymbol{z}(k) \end{aligned} \tag{2}$$

where $W_{m\times n}$, $S_{m\times m}$ and $V_{t\times m}$ are weight matrices.

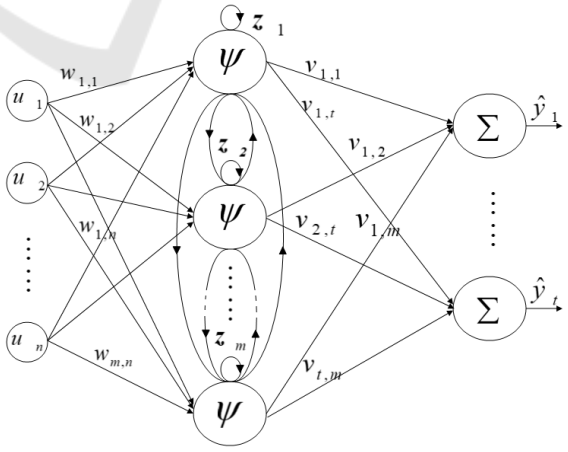

Figure 1: Internal structure of the neural network.

ψ is the activation function of the internal layer neurons and is the tangent hyperbolic function

$$\psi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3)$$

The network is trained to determine the optimal value of the weight matrices. The cost function for

$$SSE = \sum_{k=1}^{N} e(k)^T e(k) = \sum_{k=1}^{N} \big(\hat{y}(k) - y(k)\big)^T (\hat{y}(k) - y(k)) \quad (4)$$

N is the number of training samples. $\boldsymbol{y}$ is the output and $\hat{\boldsymbol{y}}$ is the output of neural network.

3 QUOTIENT GRADIENT METHOD

Neural network training is a nonlinear optimization problem. QGS is a nonlinear dynamical system to find local possible feasible solutions of the constraint satisfaction problem. Quotient gradient method transforms the constraint satisfaction problem into an unconstrained minimization problem, then uses QGS to find its local minima, which are the feasible solutions of the constraint satisfaction problem. Consider the following CSP:

$$C_I(y) < 0 \quad C_E(y) = 0, y \in R^{n-l} \quad (5)$$

where C_I are inequality constraints and C_E are equality constraints. All constraints are assumed smooth to guarantee the existence of the solution. This CSP can be transformed to the unconstrained minimization problem

$$\min_x f(x) = \frac{1}{2}\|H(x)\|^2, x = (y, s) \in R^n \quad (6)$$

$$H(x) = \begin{bmatrix} C_I(y) + \hat{S}^2 \\ C_E(y) \end{bmatrix} \in R^m, \quad \hat{S}^2 = (s_1^2, \dots, s_l^2)^T \quad (7)$$

where $\hat{S}$ is a set of slack variables to transform inequality constraints to equality constraints. Lee and Chiang showed that local minima of the unconstrained optimization problem are feasible solutions of the original CSP (Lee and Chiang, 2001). They introduced the QGS and showed that its equilibrium points are local minima of the unconstrained minimization problem, which are possible feasible solutions of the CSP. QGS for the unconstrained optimization problem is defined as

$$\dot{\boldsymbol{x}} = F(x) = -\nabla f(x) \coloneqq -D_x H(x)^T H(x) \quad (8)$$

Because QGS is a stable (Lee and Chiang, 2001), integrating it from any arbitrary point leads to an equilibrium point, which is local minimum of (5), and a feasible solution of (4). After finding the first equilibrium point, QGS must escape from its basin of attraction and enter the basin of attraction of another equilibrium point. This is done by backward integration of QGS in time until it reaches an unstable point. Thus, finding local minima of (5) reduces to a series of forward and backward integrations of QGS.

To train a neural network, we optimize the cost function (SSE) to find the optimal values of the weight matrices. To optimize SSE using the quotient gradient method, the weight matrices are partitioned as

$$V = \begin{bmatrix} \boldsymbol{v_1^T} \\ \vdots \\ \boldsymbol{v_m^T} \end{bmatrix}_{t \times m} \quad W = \begin{bmatrix} \boldsymbol{w_1^T} \\ \vdots \\ \boldsymbol{w_m^T} \end{bmatrix}_{m \times n} \quad S = \begin{bmatrix} \boldsymbol{s_1^T} \\ \vdots \\ \boldsymbol{s_m^T} \end{bmatrix}_{m \times m} \quad (9)$$

The vector of network parameters $\boldsymbol{x}$ is defined as

$$\boldsymbol{x} = [x_i]_{n_p \times 1} = [\boldsymbol{v_1}, .., \boldsymbol{v_m}, \boldsymbol{w_1}, \dots, \boldsymbol{w_m}, \boldsymbol{s_1}, \dots, \boldsymbol{s_m}]^T \quad n_p = m^2 + m \times (n + t) \quad (10)$$

Using the vector $\boldsymbol{x}$, the training set can be rewritten as

$$\boldsymbol{h}(\boldsymbol{x}) = [h_i(\boldsymbol{x})], i = 1,2, \dots, N \quad h_i(\boldsymbol{x}) = V\boldsymbol{\psi}\big(W\boldsymbol{u}(i) + S\boldsymbol{z}(i-1)\big) - y(i) \quad (11)$$

The QGS for training neural network can be constructed as

$$\dot{\boldsymbol{x}} = -\boldsymbol{f}(x) = -D_x \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{h}(\boldsymbol{x}) \quad (12)$$

where

$$D_x \boldsymbol{h}(x) = \begin{bmatrix} \dfrac{\partial h_1(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ \vdots \\ \dfrac{\partial h_N(\boldsymbol{x})}{\partial \boldsymbol{x}} \end{bmatrix}_{N \times n_p} \quad (13)$$

Finding optimal values of weight matrices becomes:

(1) Use training data and the vector of network parameters to construct the QGS system (12)
(2) Integrate the QGS forward in time until it reaches an equilibrium point.
(3) Integrate the QGS backward in time until it reaches an unstable equilibrium point.
(4) Integrate the QGS from the unstable point until it reaches another stable equilibrium point.

The process continues until it finds all equilibrium points of the QGS. The global optimum is the equilibrium with the lowest cost. The eigenvalues of the Jacobian matrix are a measure of equilibrium

instability during backward integration (Lee and Chiang, 2001; Khodabandehlou and Fadali, 2017).

4 DYNAMICAL TRAJECTORY BASED APPROACH

The quotient gradient method is a systematic approach to find the local minima of an optimization problem. The feasible region M in many optimization problems is the union of disjoint connected regions

$$M = \bigcup_i M_i \tag{14}$$

Dynamical trajectory based optimization is a systematic method to find the connected components of the feasible region and search those components for local minima. The approach has two phases: the quotient gradient system (QGS) to find connected components of the feasible region, and the projected gradient system (PGS) to search the feasible components for local minima 0(Lee and Chiang, 2001). Consider the following constrained minimization problem

$$\begin{aligned} &\min f(\boldsymbol{x}) \\ &\text{s.t. } \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \end{aligned} \tag{15}$$

Assume that $f(x) \in C^2(R^n, R^m)$ and $\boldsymbol{h}(\boldsymbol{x})$ are smooth to guarantee the existence of the solution. The inequality constraints can be incorporated in the optimization problem using slack variables. The feasible region of the optimization problem is

$$M := \{x \in R^n : \boldsymbol{h}(\boldsymbol{x}) = 0\} \tag{16}$$

4.1 PGS Phase

Searching connected components of the feasible region for local minima is an essential part of the DTB approach. DTB uses PGS to find multiple local minima in connected components of the feasible region. PGS is a stable nonlinear dynamical system whose equilibrium points are local minima of the optimization problem. After finding one local optimum, the trajectories of PGS can be used to move away from current local minimum and move toward another local minimum in the current component of the feasible region. The PGS is defined as

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}) = -\nabla f_{\text{proj}}(\boldsymbol{x}) \,, \boldsymbol{x} \in M \tag{17}$$

where $\nabla f_{\text{proj}}(\boldsymbol{x})$ is orthogonal projection of $\nabla f(\boldsymbol{x})$ on the tangent space of the feasible region. It can be shown than when $D\boldsymbol{h}(\boldsymbol{x}) = \partial \boldsymbol{h}(\boldsymbol{x})/\partial \boldsymbol{x}$ is non-singular, $\nabla f_{\text{proj}}(\boldsymbol{x})$ is defined as

$$\begin{aligned} &\nabla f_{\text{proj}}(\boldsymbol{x}) = [I - \\ &D\boldsymbol{h}(\boldsymbol{x})^T (D\boldsymbol{h}(\boldsymbol{x}) D\boldsymbol{h}(\boldsymbol{x})^T)^{-1} D\boldsymbol{h}(\boldsymbol{x})] \nabla f(\boldsymbol{x}) \end{aligned} \tag{18}$$

Every PGS trajectory converges to one of its stable equilibrium points, which is also a local optimum of (14). After finding one local minimum, PGS needs to escape from stability region of that local minimum and enter the stability region of another local minimum in the current component of the feasible region. This is achieved by backward integration of PGS until reaching a saddle point. Then by forward integration of PGS moves it towards another local optimal solution. By repeating this process, PGS finds all the local optimal solutions in the current component of the feasible region. The next step is moving toward another component of the feasible region, which is done in the QGS phase.

4.2 QGS Phase

To explore all the components of the feasible region, DTB approach needs to escape from current component and move to another component of the feasible region. DTB uses the trajectories of a nonlinear dynamical system to do this. The nonlinear dynamical system to explore the components of the feasible region, the QGS, is

$$\dot{\boldsymbol{x}} = -D\boldsymbol{h}(\boldsymbol{x})^T D\boldsymbol{h}(\boldsymbol{x}) \tag{19}$$

where $D\boldsymbol{h}(\boldsymbol{x})$ is the Jacobian of $\boldsymbol{h}$ at $\boldsymbol{x}$. Every QGS trajectory converges to one of its stable equilibrium manifolds and every stable QGS equilibrium manifold corresponds to a connected component of the feasible region. Therefore, to approach a connected component of the feasible region, the QGS is integrated until it reaches an equilibrium point. To escape from the current component of the feasible region and move toward another feasible component, the QGS is integrated backward in time until it reaches an unstable point, then integrated forward in time until it reaches another component of the feasible region. By invoking PGS and QGS phases repeatedly, the DTB approach finds multiple components of the feasible region and locates local optimal solutions. The local optimal solution with the lowest cost is the global optimal solution of the optimization problem.

Although training neural networks is an unconstrained optimization problem, constraints are needed for defining QGS. We define the constraints of the optimization problem as upper and lower bounds on the neural network weights. The

constraints are written in terms of the network parameters of (9) as

$$|x_i| \leq l_i, \qquad i = 1, \dots, n_p \tag{20}$$

Adding slack variables, $\boldsymbol{s}^T = [s_1, \dots, s_{n_p}]$, inequality constraints can be written as equality constraints

$$h_i(\boldsymbol{x}) = x_i^2 - l_i^2 + s_i^2 = 0, \; i = 1, \dots, n_p \tag{21}$$

The augmented vector of parameters is defined as

$$\boldsymbol{o} = [\boldsymbol{x}\, \boldsymbol{s}]^T{}_{(2\times n_p)\times 1} \tag{22}$$

Equation (14) can be rewritten in terms of $\boldsymbol{o}$ as

$$\begin{aligned} &\min f(\boldsymbol{o}) \\ &\text{s.t. } \boldsymbol{h}(\boldsymbol{o}) = \boldsymbol{0} \end{aligned} \tag{23}$$

The constraint set $D\boldsymbol{h}(\boldsymbol{o}) = [\partial h_i(\boldsymbol{o})/\partial \boldsymbol{o}]^T_{(n_p)\times(2\times n_p)}$ is always nonsingular and the PGS and QGS for training neural network are

PGS:

$$\begin{aligned} \dot{\boldsymbol{o}} = -(I - D\boldsymbol{h}(\boldsymbol{o})^T (D\boldsymbol{h}(\boldsymbol{o}) D\boldsymbol{h}(\boldsymbol{o})^T)^{-1} \\ \times D\boldsymbol{h}(\boldsymbol{o})) \nabla f(\boldsymbol{o}) \end{aligned} \tag{24}$$

QGS:

$$\dot{\boldsymbol{o}} = -D\boldsymbol{h}(\boldsymbol{o})^T \boldsymbol{h}(\boldsymbol{o}) \tag{25}$$

The steps to train a neural network using DTB are:

(5) Formulate the optimization problem of (23) using the cost function and the constraint set .
(6) Construct the PGS and QGS systems of (24), (25).
(7) Integrate QGS forward in time to reach an equilibrium point
(8) Integrate the PGS from equilibrium point of QGS until reaching an equilibrium point. This equilibrium point is a local optimal solution of the optimization problem.
(9) Integrate PGS backward in time until reaching a saddle point. Then integrate PGS forward in time to all other local optimal solution in current component of the feasible region
(10) Integrate QGS from last local optimal solution backward in time to reach an unstable point, then integrate it forward in time until reaching another stable point
(11) Go to step 4

This reduces training neural networks to repeatedly invoking PGS and QGS until a stopping criterion is satisfied. If the bounds on the network parameter are large, they do not pose a limitation on the solution (Lee and Chang, 2001, 2004).

5 BENCHMARK SYSTEMS

5.1 Cascade Tank Model

The cascade tank system is a challenging nonlinear benchmark for system identification. The system consist two tanks with a pump. Figure 2 shows the structure of the cascade tank model.

The pump feeds water into the upper tank and the lower tank has a free outlet. The system is not highly nonlinear during normal operation. However, with large water flow into the upper tank, overflow can occur in the upper tank. This overflow acts as an input-dependent process noise. Without overflow, the cascade tank is governed by

$$\begin{aligned} \dot{x}_1(t) &= -k_1\sqrt{x_1(t)} + k_4 u(t) + w_1(t) \\ \dot{x}_2(t) &= k_2\sqrt{x_1(t)} - k_3\sqrt{x_2(t)} + w_2(t) y(t) \\ y(t) &= x_2(t) + e(t) \end{aligned} \tag{26}$$

where $u(t)$ is the pump voltage, $x_1(t)$ and $x_2(t)$ are the states of the cascade tank system, $w_1(t)$, $w_2(t)$ and $e(t)$ are noise and k_1, k_2, k_3 and k_4 are system constants. $y(t)$ is the system output, i.e., the water level in the second tank.

The input is a multisine signal with frequencies from 0 to 0.0144 Hertz. Lower frequency inputs have larger amplitude than higher frequency inputs. The sampling period is $T_s = 4s$. A capacitive water level sensor is used to measure the water level and is a part of the system (Schoukens et al. 2015).

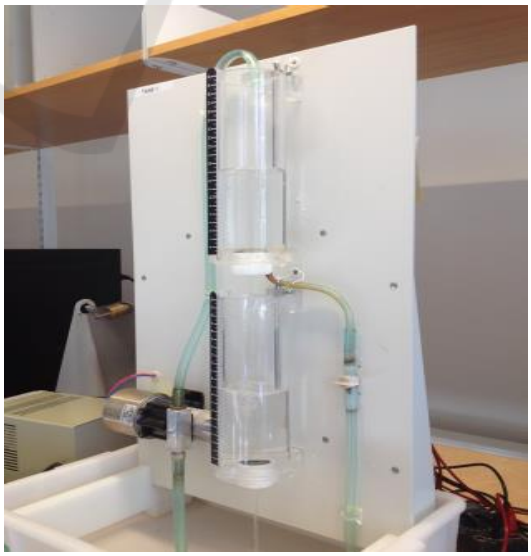

Figure 2: Cascade tank model structure (Schoukens et al. 2015).

5.2 Bouc-Wen Model

The Bouc-Wen model is a widely studied hysteresis model in mechanical and civil engineering (Ismail et. al., 2009, Khodabandehlou et. al., 2017). A hysteretic system has multiple stable equilibrium points; therefore its output can change based on the input history. This complicates the analysis and design of the system (Noël and Schoukens, 2006). The Bouc-Wen oscillator is governed by

$$m_L\ddot{y}(t) + r(y,\dot{y}) + z(y,\dot{y}) = s(t) \quad (27)$$

where $s(t)$ is the input, i.e. the external force, $y(t)$ is the displacement and m_L is the mass constant. $r(y,\dot{y})$ is the total restoring force and $z(y,\dot{y})$ is a history-dependent nonlinear term that determines the hysteretic property of the system. The static restoring force is

$$r(y,\dot{y}) = k_L y + c_L\dot{y} \quad (28)$$

where k_L is the linear stiffness coefficient and c_L is the viscous damping coefficient. $z(y,\dot{y})$ is

$$\dot{z}(y,\dot{y}) = \alpha|\dot{y}| - \beta(\gamma|\dot{y}||z|^{\nu-1} + \delta\dot{y}|z|^{\nu}) \quad (29)$$

α, β, γ, ν and δ are Bouc-Wen parameters that determine the shape and smoothness of the hysteresis loop. Table 1 shows the system parameters.

Table1: System parameters.

Parameter	Value
m_L	2
c_L	2
k_L	5×10^4
α	5×10^4
β	10^3
γ	0.8
δ	-1.1
ν	1

6 SIMULATION RESULTS

We use the quotient gradient and the DTB me to train a neural network for identification of benchmark problems and compare the results.

6.1 Cascaded Tank Modelling

For a fair comparison between QGS and DTB, the neural networks structure, including training data, input vector, hidden layer activation function and number of hidden layer nodes, is the same in all the simulations. All network parameters were initialized with random values from a zero-mean normal distribution with standard deviation $\sigma^2 = 0.1$. The optimal number of hidden layer nodes was found to be $m = 9$ and the network input is

$$\boldsymbol{u}(k) = [1, s(k), \dots, s(k-9), y(k-1), \dots, y(k-9)]^T \quad (30)$$

The target value for network output is $y(k)$. Figure 3 shows the validation data and the output of the trained networks. Figure 4 shows the validation error of the networks. Although both networks have excellent performance in the identification of the nonlinear model, the dynamical trajectory based method has a lower mean squared error than the QGS trained network for validation data. The mean squared error of the DTB trained network is $MSE = 0.0264$ while the mean squared error of QGS trained network is $MSE = 0.0312$. Although both approaches successfully identify the model of cascaded tanks, the dynamical trajectory based approach gives slightly better results. This is because it more accurately determines the local minima of the optimization problem.

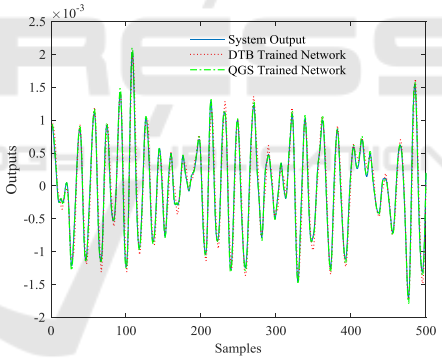

Figure 3: Validation outputs for the networks trained with DTB and QGS.

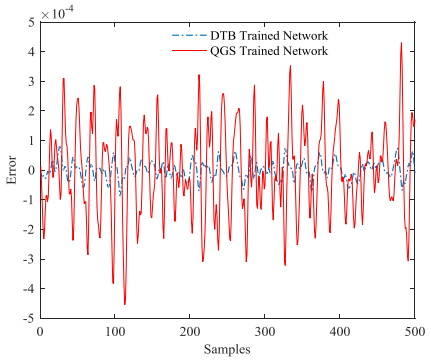

Figure 4: Validation error for the networks trained with DTB and QGS.

6.2 Bouc-Wen Model

For a fair comparison, all the parameters for QGS and DTB approach are assumed the same. The network parameters were initialized with random values from zero-mean normal distribution with standard deviation of $\sigma^2 = 0.1$. The optimal number of hidden layer nodes was found to be $m = 7$ and the neural network's input vector is assumed to be

$$\boldsymbol{u}(k) = [\, s(k), \dots, s(k-5), y(k-1), \dots, y(k-5)]^T \quad (31)$$

The target value for network output is $y(k)$. Figure 5 shows the validation output and the output of the trained networks. Figure 6 shows the validation error of the networks and shows that the dynamical trajectory based trained network has better performance than the QGS trained network. The mean squared error of the DTB trained network is $MSE = 2.3 \times 10^{-8}$. While the mean squared error of the QGS trained network is $MSE = 6.1 \times 10^{-8}$.

In our simulations, multiple runs of both algorithms with different initial values yields the same results, therefore it can be concluded that both methods are robust with respect to initial values. This is because both methods escape from stability region of the current equilibrium point and enter into stability region of another equilibrium point, therefore, they don't get trapped in local minima of the optimization problem. In both simulations, there is a steady state identification error. The error can be for coloured noise of the experimental data or suboptimal neural network structure and may be reduced by using different neural network.

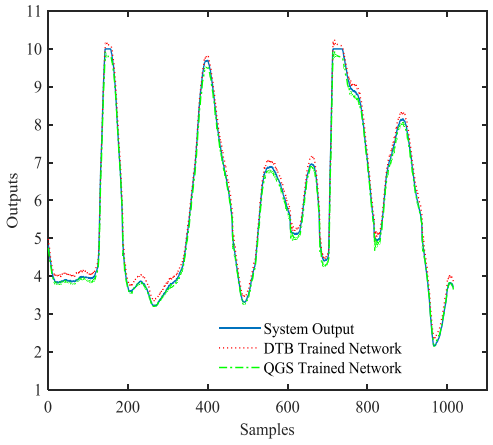

Figure 5: Test outputs for the networks trained with DTB and QGS.

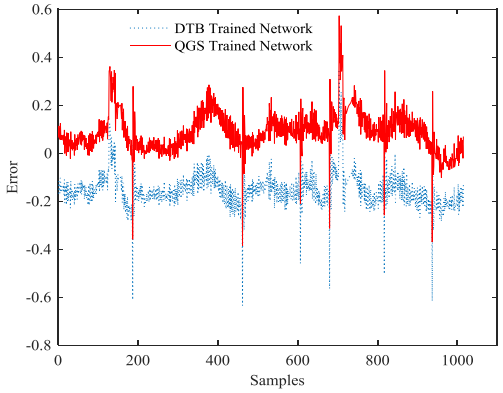

Figure 6: Test error of the networks trained with DTB and QGS.

7 CONCLUSION

In this study, we used two trajectory-based optimization approaches to train artificial neural networks for the identification of two nonlinear system identification benchmark problems: cascaded tanks and the Bouc-Wen model. Both approaches use trajectories of nonlinear dynamical systems to find optimal value of the neural network weights and were able to train neural network and efficiently identify nonlinear system models. Both approaches turn the neural network training problem into routine of forward and backward integration of nonlinear dynamical systems, therefore the training process is simple and straightforward. Although both approaches successfully identify the nonlinear models, the dynamical trajectory based approach has better performance at the expense of longer training time. Future work will include designing a neural network based controller for nonlinear systems.

REFERENCES

Chen, T. B., Soo, V. W., 1996. A comparative study of recurrent neural network architectures on learning temporal sequences. *In Proceedings of IEEE International Conference on Neural Network*, p.1945-1950.

Coban, R., 2013. A context layered locally recurrent neural network for dynamic system identification. *Engineering Applications of Artificial Intelligence*, 26, p.241–250.

Dongpo, X., Zhengxue, L., Wei, W., 2010. Convergence of gradient method for a fully recurrent neural network. *Soft Computing*, 14, p.245–250.

Efe, M. O., Kaynak, K., 1999. A comparative study of neural network structures in identification of nonlinear systems. *Mechatronics*, 9, p.287–300.

Gautam, P., 2016. System identification of nonlinear Inverted Pendulum using artificial neural network. *Recent Advances and Innovations in Engineering (ICRAIE)*, 2016 International Conference on.

Hermans, M., Dambre, J., Bienstman, P., 2015. Optoelectronic systems trained with backpropagation through time. *IEEE Trans. Neural Netw. Learn. Syst*., 26(7), p.1545-1550.

Ismail, M., Ikhouane, F., Rodellar, J., 2009. The hysteresis Bouc-Wen model, a survey. *Archives of Computational Methods in Engineering*, 16, p.161-188.

Khodabandehlou, H., Fadali, M. S., 2017. Echo State versus Wavelet Neural Networks: Comparison and Application to Nonlinear System Identification. *IFAC-Papers OnLine*, 50(1), p.2800-2805, doi: https://doi.org/10.1016/j.ifacol.2017.08.630

Khodabandolehlou, H., Pekcan, G., Fadali, M. S., Salem, M. A., 2017. Active neural predictive control of seismically isolated structures. *Structural Control and Health Monitoring*, 25(1), doi: https://doi.org/10.1002/stc.2061

Khodabandehlou, H., Fadali, M. S., 2017. A Quotient Gradient Method to Train Artificial Neural Networks. *In Proc. Int. Joint Conf. Neural Networks (IJCNN)*, Anchorage, USA

Lee, J., Chiang, H. D., 2001. Quotient gradient methods for solving constraint satisfaction problems. *IEEE Int. Symp. on Circuits and Systems*, Australia

Lee, J., Chiang, H. D., 2001. Computation of multiple type-one equilibrium points on the stability boundary using generalized fixed-point homotopy methods. *In Proc. of IEEE International Symposium on Circuits and Systems*, 2, p.361-364.

Lee, J., and Chiang, H. D., 2004. A Dynamical Trajectory-Based Methodology for Systematically Computing Multiple Optimal Solutions of General Nonlinear Programming Problems. *IEEE Trans. Automatic Control*, 49(6), p.888-899.

Liu, G. P., Kadirkamanathan, V., Billings, S. A., 1998. On-line identification of nonlinear systems using Volterra polynomial basis function neural networks. *Neural Networks*, 11, p.1645–1657.

Lu, Z., Sindhwani, V., Tara, N. S., 2016. Learning compact recurrent neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p.5960-5964.

Monner, D., Reggia, J. A., 2012. A generalized LSTM-like training algorithm for second-order recurrent neural networks. *Neural Networks*, 25, p.70-83.

Narendra, S. K., Parthasarathy, K. 1990. Identification and control of dynamical systems using neural networks. *IEEE Trans. on neural networks*, 1(1), p4-27.

Noël, J. P., Schoukens, M., 2016. Hysteretic benchmark with a dynamic nonlinearity. *In Workshop on Nonlinear System Identification Benchmarks*, p.7–14. Brussels, Belgium, Available: http://nonlinearbenchmark.org/#BoucWen

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. *In Proc of the 30th International Conference on Machine Learning*, Atlanta, GA, USA.

Pearlmutter, B. A., 1995. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Trans. Neural Networks*, 6(5), p.1212–1228.

Pham, D. T., Liu, X., 1992. Dynamic system modelling using partially recurrent neural networks. *Journal of systems engineering*, 2, p.90–97.

Pham, D. T., Liu, X., 1995. *Neural Networks for Identification, Prediction and Control*. Springer-Verlag, London

Schoukens, M., Mattson, P., Wigren, T., Noël, J. P., 2015. Cascaded tanks benchmark combining soft and hard nonlinearities. Available: http://nonlinearbenchmark.org/#Tanks

Stroeve, S., 1998. An analysis of learning control by backpropagation through time. *Neural Networks*, 11(4), p.709-721.

Sutskever, I., Martens, J., Hinton, G., 2011. Generating text with recurrent neural networks. *In Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, p.1017–1024.

Verdult, V., 2002. *Nonlinear system identification: a state-space approach*. Twente University Press.

Werbos, P. J., 1998. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1, p.339-356.

Williams, R. J. Zipper, D., 1989. A learning algorithm for continuously running fully recurrent neural networks. *Neural Computation*, 1, p.270-280.

Yamada, T., Yabuta, T., 1993. Dynamic system identification using neural networks. *IEEE Trans. on systems, man, and cybernetics*, 23(1), p204-211.