2011 IEEE International Conference on Control Applications (CCA)
Part of 2011 IEEE Multi-Conference on Systems and Control
Denver, CO, USA. September 28-30, 2011

Fault Detection of an Engine Using a Neural Network Trained by the Smooth Variable Structure Filter

Ryan M. Ahmed, Mohammed A. El Sayed, S. Andrew Gadsden, and Saeid R. Habibi

Abstract— **A multilayered neural network is a multi-input, multi-output (MIMO) nonlinear system in which training can be regarded as a nonlinear parameter estimation problem by estimating the network weights. In this paper, the relatively new smooth variable structure filter (SVSF) is used for the training of a nonlinear multilayered feed forward network. The SVSF is a recursive sliding mode parameter and state estimator that has a predictor-corrector form. Using a switching gain, a corrective term is calculated to force the network weights to converge to within a neighbourhood of the optimal weight values. SVSF-based trained neural networks are used to classify engine faults on the basis of vibration data. Two faults are induced in a four-stroke, eight-cylinder engine. Furthermore, a comparative study between the popular back propagation method, the extended Kalman filter (EKF), and the SVSF is presented. Experimental results indicate that the SVSF is comparable with the EKF, and both methods outperform back propagation.**

I. Introduction

Back propagation (BP) used to be one of the most commonly used algorithm in the field of multilayer perceptron training [1]. It is a first-order stochastic gradient descent method that iteratively adjusts weights to minimize the output error in a supervised manner. However, since early versions of BP involve a constant learning rate, a slow speed of convergence is attained. In fact, several enhanced training algorithms have been developed through the literature to improve training performance, mapping accuracy and speed of convergence compared to the BP algorithm [2]. Most of these techniques like quasi-Newton and Levenburg-Marquardt demonstrate better performance as they involve second-order derivative information. In addition, these algorithms are implemented in a batch (multi-streaming) mode where weights are updated based on more than one training sample in the training set, in contrast with the conventional BP where weights are updated by involving only one training sample (a serial mode) [3]. Even though second-order algorithms have proven to outperform the classical first-order BP, they may suffer from poor convergence properties due to problems with local minima [2].

The Kalman filter (KF) is the most popular state estimation tool. It provides statistically optimal estimations for linear systems in the presence of Gaussian white noise. In the case of nonlinear systems, the extended Kalman filter is applied by linearizing the system around the latest state estimate at each time interval. An EKF-based neural network training technique was first introduced by Singhal and Wu in 1989 [4]. The EKF provides a powerful neural network training capability compared to conventional first-order gradient algorithms, such as the BP [2]. In literature, the EKF has been extensively applied for training of both feed-forward [5] and recurrent networks [6,7] in both a global form (GEKF) or in a decoupled form (DEKF). Although the EKF demonstrates a close performance compared to a second-order derivative, batch-based method, it can avoid local minima problems by encoding second-order information in terms of a state error covariance matrix [2]. Accordingly, the EKF represents an efficient and practical alternative to second-order training methods.

The recently proposed smooth variable structure filter (SVSF) provides a robust dynamic adaptation, high-rate of convergence, and can guarantee estimation stability for bounded uncertainties and noise levels [8]. The SVSF has been successfully applied for parameter and state estimation [9,10].

In this paper, the SVSF is used for feed-forward neural network training. The SVSF is applied in a global (GSVSF), multi-streaming mode. The GSVSF's performance is compared to the standard first and second-order derivative BP algorithms, as well as to the GEKF using an engine fault detection pattern classification problem. The paper is organized as follows: Section II provides a general description of the feed-forward multilayer perceptron, and Section III describes the global and decoupled extended Kalman filter-based training algorithms. Section IV-A provides an introduction to the SVSF, Section IV-B considers the formulation of the SVSF for neural networks training. Section V considers the application of SVSF training method to an engine experimental setup besides involves trained networks experimental results using the three training techniques discussed above. Section VI includes concluding remarks and recommendations for future work.

Manuscript submitted June 20, 2011.

The authors are with the Department of Mechanical Engineering at McMaster University, Hamilton, Ontario, Canada. Ryan M. Ahmed is an M.A.Sc. graduate, part-time Ph.D student, and a research engineer at Ford Motor Company (e-mail: ryan.ahmed@mcmaster.ca) interested in artificial intelligence, fault detection, and control systems. Mohammed El Sayed is a Ph.D. candidate (e-mail: abugabma@mcmaster.ca) with interests in fluid power systems, sliding mode control, and automation. S. Andrew Gadsden is a Ph.D. candidate, and both an ASME and IEEE student member. He is funded by the National Sciences and Engineering Research Council of Canada (NSERC) (e-mail: gadsdesa@mcmaster.ca). Dr. Saeid R. Habibi is a Professor and Chair in the Department of Mechanical Engineering at McMaster University (e-mail: habibi@mcmaster.ca).

II. Feed-Forward Multilayered Neural Network

A multilayer feed forward network consists mainly of a set of sensory units (input source nodes) that constitutes the input layer, one or more hidden layers and an output layer. As shown in Fig. 1, each node is connected to all nodes in the adjacent layer by links (weights), and computes a weighted sum of the inputs. An offset (bias) is added to the resultant sum followed by a nonlinear activation function application. The input signal propagates through the network in a forward direction on a layer-by-layer basis. Consequently, the network represents a static mapping between inputs and outputs.

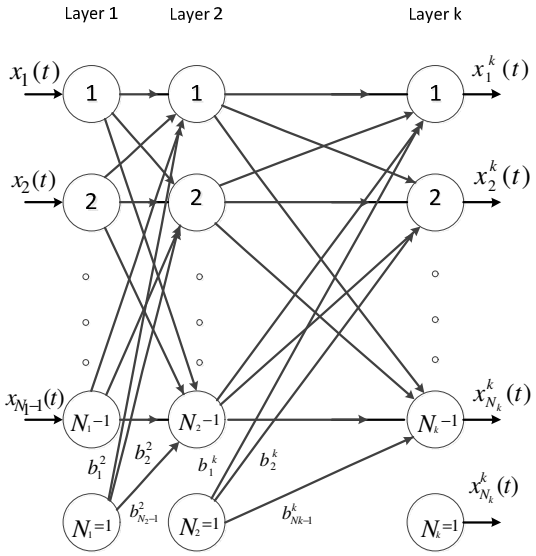


Fig. 1. Schematic diagram of feed-forward multilayer perceptron network [13]

Let k denote the total number of layers, including the input and output layers. $\mathrm{Node}(n,i)$ denotes the i^{th} node in the n^{th} layer, and $N_n - 1$ is the total number of nodes in the n^{th} layer. The operation of $\mathrm{node}(n+1,i)$ is described by the following equation:

$$x_i^{n+1}(t) = \varphi\Big(\sum_{j=1}^{N_n-1} w_{i,j}^n\, x_j^n(t) + b_i^{n+1}\Big) \qquad (1)$$

where $x_i^n(t)$ denotes the output of $node(n,j)$ for the t training pattern, $w_{i,j}^n$ denotes the link weight from $\mathrm{node}(n,j)$ to the $\mathrm{node}(n+1,i)$. b_i^n is the node offset (bias) for $\mathrm{node}(n,i)$. The function $\varphi(.)$ is a nonlinear sigmoid activation function that is commonly used for feedforward networks as it resembles the biological neurons operation. It is defined by:

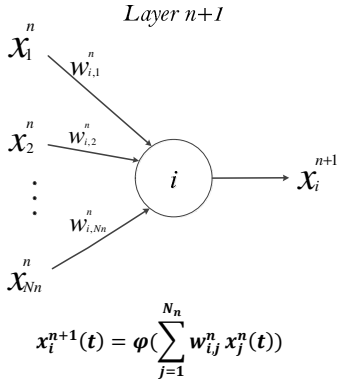
$$\varphi(w) = \frac{1}{1+e^{-aw}} \qquad a > 0 \text{ and } -\infty < w < \infty \qquad (2)$$

For simplicity, as shown in Fig. 2, the node bias is considered as a link weight by setting the last input N_n to $node(n+1,i)$ to value of one as follows:

$$x_{N_n}^n(t) = 1, \qquad 1 \le n \le k$$
$$w_{i,N_n}^n = b_i^{n+1}, \qquad 1 \le n \le k-1$$

Consequently, (1) can be rewritten in the following form:

$$x_i^{n+1}(t) = \varphi\Big(\sum_{j=1}^{N_n} w_{i,j}^n\, x_j^n(t)\Big) \qquad (3)$$



$$\boldsymbol{x_i^{n+1}(t) = \varphi\Big(\sum_{j=1}^{N_n} w_{i,j}^n\, x_j^n(t)\Big)}$$

Fig. 2. Node (n+1, i) representation

III. Global and Decoupled EKF-based NN training

The EKF has been tailored to train feed-forward neural networks by formulating the network as a filtering problem [11]. Accordingly, feed-forward multilayer perceptron network behaviour can be described by a nonlinear discrete-time state space representation [12]:

$$w_{k+1} = w_k + \omega_k \qquad (4)$$
$$y_k = C_k(w_k, u_k) + v_k \qquad (5)$$

Equation (4) represents the system equation. It demonstrates the neural network as a stationary system with an additional zero mean, white system noise ω_k with a covariance described by $[\omega_k \omega_l{}^T] = \delta_{k,l} Q_k$. Neural network weights and biases w_k are regarded as the system's state. Equation (5) is the measurement (observation) equation. It is a nonlinear equation relating network desired (target) response y_k to the network input u_k and network's weights w_k. The nonlinear function C_k represents the measurements matrix, which comprises the network overall transfer function. A zero-mean, white measurement noise v_k is added whose covariance is defined as $[v_k v_l{}^T] = \delta_{k,l} R_k$. As previously mentioned, the KF provides an optimal state estimate for linear systems. However, for this particular problem, the EKF will be implemented due to the nonlinearity present in the measurements.

Consider a feed-forward multilayer perceptron network with two hidden layers as shown in Fig. 3. All activation functions of the first, second and output layers are nonlinear

sigmoidal functions denoted as φ_I, φ_{II} and φ_o respectively. The network transfer function in terms of network weights, inputs and activation functions can be mathematically defined as:

$$y_i(k) = \varphi_o\left(W_{o_i}(k)\,\varphi_{II}\left(W_{II}(k)\,\varphi_I\left(W_I(k)u(k)\right)\right)\right) \quad \text{for } i=1,2..m \tag{6}$$

Where m denotes number of output neurons, W_I, W_{II}, W_o are group weight matrices for first hidden layer, second hidden layer and output layer, respectively. Linearization is performed by differentiating the network transfer function with respect to network synaptic weights (i.e.: deriving the Jacobian). The Jacobian matrix $C_{k|linearized}$ can be mathematically expressed as follows:

$$C_{k|linearized} = \begin{bmatrix} \frac{\partial y_1}{\partial W_1} & \frac{\partial y_1}{\partial W_2} & \cdots & \frac{\partial y_1}{\partial W_{N_T}} \\ \frac{\partial y_2}{\partial W_1} & \frac{\partial y_2}{\partial W_2} & & \frac{\partial y_2}{\partial W_{N_T}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_m}{\partial W_1} & \frac{\partial y_m}{\partial W_2} & \cdots & \frac{\partial y_m}{\partial W_{N_T}} \end{bmatrix} \tag{7}$$

where N_T denotes total number of synaptic weights (including bias), z specifies number of input neurons, by differentiating (6) with respect to different weight groups W_I, W_{II}, W_o and *for i, l=1, 2 ... m* get*:*

$$\frac{\partial y_i}{\partial W_{o_l}} = \begin{cases} \acute{\varphi}_o\left(W_{o_i}\varphi_{II}(W_{II}\varphi_I(W_Iu))\right)\varphi_{II}(W_{II}\varphi_I(W_Iu)), & if\ i=l \\ 0, & otherwise \end{cases} \tag{8}$$

$$\frac{\partial y_i}{\partial W_{II}} = \acute{\varphi}_o\left(W_{o_i}\varphi_{II}(W_{II}\varphi_I(W_Iu))\right)W_{o_i}\acute{\varphi}_{II}(W_{II}\varphi_I(W_Iu))\varphi_I(W_Iu) \tag{9}$$

$$\frac{\partial y_i}{\partial W_I} = \acute{\varphi}_o\left(W_{o_i}\varphi_{II}(W_{II}\varphi_I(W_Iu))\right)W_{o_i}\acute{\varphi}_{II}(W_{II}\varphi_I(W_Iu))W_{II}\acute{\varphi}_I(W_Iu)u \tag{10}$$

By augmenting (8), (9), and (10) in one matrix:

$$C_{k|linearized} = \left[\frac{\partial y}{\partial W_o} \quad \frac{\partial y}{\partial W_I} \quad \frac{\partial y}{\partial W_{II}}\right] \tag{11}$$

$C_{k|linearized}$ is the m-by-N_T measurement matrix of the linearized model around the current weight estimate.

The EKF-based neural network training introduced by Singhal and Wu is known as the global extended Kalman filter (GEKF) [4]. In GEKF algorithm, all network weights and biases are simultaneously processed and a second-order information matrix correlating each pair of network weights is involved and updated [2]. Consequently, the GEKF computational complexity is $O(m\,{N_T}^2)$ and storage requirements of $O({N_T}^2)$ is required which is relatively high compared to the standard BP algorithm.

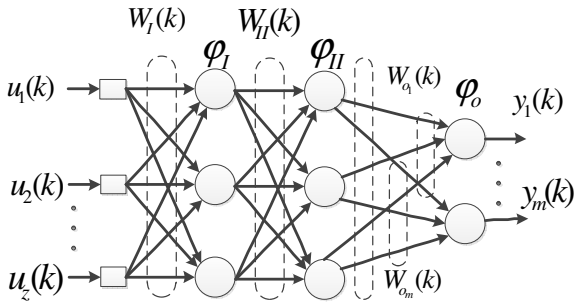


Fig. 3. Feed-forward multilayer perceptron with 'z' inputs, 2 hidden layers and 'm' outputs

The DEKF-based neural network training algorithm illustrated below represents the most general EKF-based neural network training form. The GEKF represents a special case of the DEKF by setting weight group number to one. Neural network training using the DEKF algorithm can be expressed as follows [13]:

$$\Gamma_k = [\sum_{i=1}^{g}(C_k^i)^T\,P_k^i\,C_k^i\ + R_k]^{-1} \tag{12}$$

$$K_k^i = P_k^i\ C_k^i\ \mathbf{\Gamma_k} \tag{13}$$

$$\alpha_k = d_k - \hat{d}_k \tag{14}$$

$$\hat{w}_{k+1}^i = \hat{w}_k^i + K_k^i\,\alpha_k \tag{15}$$

$$P_{k+1}^i = P_k^i - K_k^i\left(C_k^i\right)^T P_k^i + Q_k^i \tag{16}$$

where, the following are defined:

- **Γ** *m*-by-*m* matrix known as global scaling matrix (or global conversion factor).
- C n_i-by-*m* gradient matrix, it involves weights gradient w.r.t every output node.
- α *m*-by-1 matrix representing innovation which is the difference between desired (target) and actual network output.
- P n_i-by-n_i error covariance matrix.
- Q n_i-by-n_i process covariance matrix.
- K n_i-by-*m* Kalman gain matrix.
- $\hat{d}_k$ *m*-by-*1* output estimate.
- d_k *m*-by-*1* target (desired) output.
- R *m*-by-*m* measurement noise covariance matrix.

The above DEKF training algorithm operates in a serial mode fashion. In serial mode, one training sample is involved in error calculation, gradients computation and synaptic weight update. A problem known as *recency* phenomenon arises with serial mode when training tends to be influenced by the most recent samples [2]. Consequently, a trained network fails to remember former input-output mappings thus serial-mode training reduces training performance. The *recency* phenomenon can be circumvented using multi-streaming training technique, [14,15,16]. Multi-streaming EKF training allows multiple training samples to be batched and processed at the same time. It involves training *M* multiple identical parallel neural networks using

several training samples followed by a single weight update using the entire *M* networks errors. The above algorithm can be adjusted to multi-streaming mode by replacing *m* with *M x m* [3].

IV. Smooth Variable Structure Filter (SVSF) Neural Network Training

A. Introduction to the SVSF

In 2007, the smooth variable structure filter (SVSF) was introduced based on variable structure theory and sliding mode concepts [8]. It implements a switching gain to converge the estimates to within a boundary of the true states (i.e., existence subspace). In its present form, the SVSF has been shown to be stable and robust to modeling uncertainties and noise [17,18]. The SVSF method is model based and may be applied to differentiable linear or nonlinear dynamic equations. The original form of the SVSF as presented in [8] did not include covariance derivations. An augmented form of the SVSF was presented in [19], which includes a full derivation for the filter. The basic estimation concept of the SVSF is shown in the following figure.

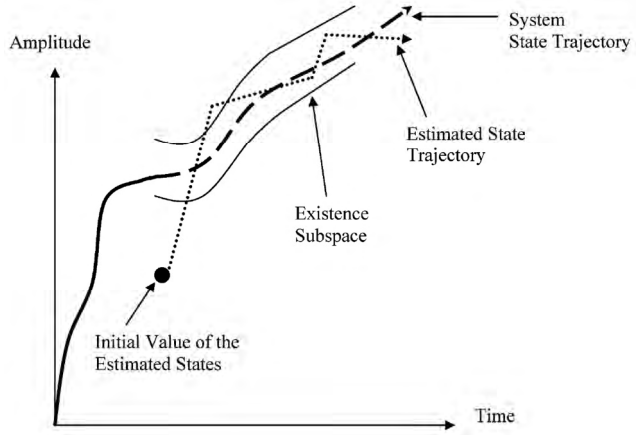


Fig. 4. SVSF estimation concept [8]

Further to an initial condition, the estimated state is forced to within a region around the state trajectory referred to as the existence subspace. The width of the existence subspace β is a function of the uncertain dynamics associated with the inaccuracy of the internal model of the filter and varies with time [8]. Typically this value is not exactly known but an upper bound may be selected based on a priori knowledge. Once the estimate enters the existence subspace, it is forced into switching along the system state trajectory. A saturation term may be used in this region to reduce the magnitude of chattering and smooth out the result.

The effect of the smoothing boundary layer is shown in Fig's. 5 and 6. When the smoothing boundary layer is defined larger than the existence subspace boundary, the estimated state trajectory is smoothed. However, when the smoothing boundary layer is too small, chattering remains indicating the uncertainties have been underestimated

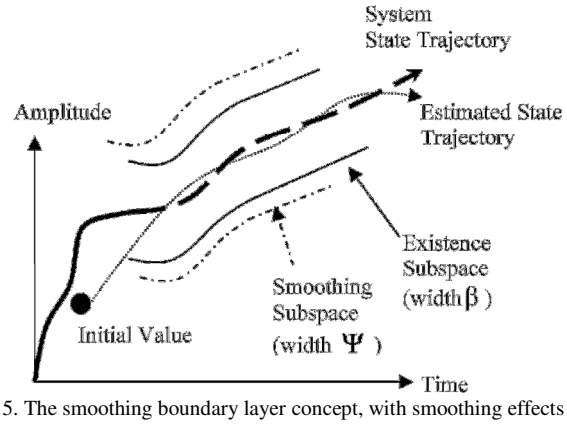


Fig. 5. The smoothing boundary layer concept, with smoothing effects [8]

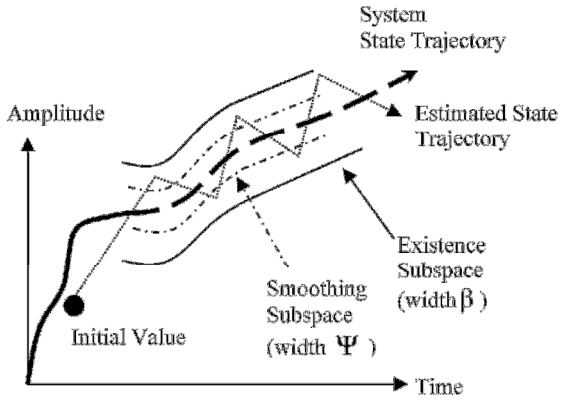


Fig. 6. The smoothing boundary layer concept, with chattering effects [8].

B. SVSF-Based Neural Networks Training

The SVSF can be applied for training nonlinear feed-forward neural networks through estimation of the network weights. The SVSF has been adapted to train feed-forward neural networks by visualizing the network as a filtering problem. This process is iterative and may be summarized by the following set of equations. The predicted state (or neural network weight) estimates $\hat{w}_{k+1|k}$ and state error covariances $P_{k+1|k}$ are first calculated respectively as follows:

$$\hat{w}_{k+1|k} = F\hat{w}_{k|k} + Gu_k \quad (17)$$

$$P_{k+1|k} = HP_{k|k}H^T + Q_k \quad (18)$$

Utilizing the predicted state estimates $\hat{w}_{k+1|k}$, the corresponding predicted measurements $\hat{z}_{k+1|k}$ and measurement errors $e_{z_{k+1|k}}$ may be calculated:

$$\hat{z}_{k+1|k} = H\hat{w}_{k+1|k} \quad (19)$$

$$e_{z_{k+1|k}} = z_{k+1} - \hat{z}_{k+1|k} \quad (20)$$

The SVSF gain is a function of: the a priori and a posteriori measurement errors $e_{z_{k+1|k}}$ and $e_{z_{k|k}}$, the smoothing boundary layer widths ψ, the 'SVSF' memory or convergence rate γ, as well as the linear measurement matrix H. For the derivation of the SVSF gain K_{k+1}, refer to [8,19]. The SVSF gain is defined as a diagonal matrix such that:

$$K_{k+1} = H^{+} diag\left[\left(\left|e_{z_{k+1|k}}\right| + \gamma\left|e_{z_{k|k}}\right|\right) \circ sat\left(\frac{e_{z_{k+1|k}}}{\psi}\right)\right] diag\left(e_{z_{k+1|k}}\right)^{-1} \quad (21)$$

This gain is used to calculate the updated state estimates $\hat{w}_{k+1|k+1}$ as well as the updated state error covariance matrix $P_{k+1|k+1}$:

$$\hat{w}_{k+1|k+1} = \hat{w}_{k+1|k} + K_{k+1} e_{z_{k+1|k}} \quad (22)$$

$$P_{k+1|k+1} = (I - K_{k+1}H)P_{k+1|k}(I - K_{k+1}H)^{T} + K_{k+1}R_{k+1}K_{k+1}^{T} \quad (23)$$

Finally, the updated measurement estimate $\hat{z}_{k+1|k+1}$ and measurement errors $e_{z_{k+1|k+1}}$ are calculated, and are used in later iterations:

$$\hat{z}_{k+1|k+1} = H\hat{w}_{k+1|k+1} \quad (24)$$

$$e_{z_{k+1|k+1}} = z_{k+1} - \hat{z}_{k+1|k+1} \quad (25)$$

The estimation process is stable and convergent if the following lemma is satisfied:

$$|e_k| > |e_{k+1}| \quad (26)$$

The proof, as defined in [8], is such that if one defines η_{k+1} as a random but bounded amplitude such that $\eta_{k+1} \leq \beta$, then $|e_k| > |e_{k+1}|$ applies to the phase where $|e_k| > \beta$ that is defined by the reachability phase. Let a Lyapunov function be defined such that $v_{k+1} = e_{k+1}^2$. Hence, the estimation process is stable if $(\Delta v_{k+1} = e_{k+1}^2 - e_k^2) < 0$. Furthermore, note that this stability condition is satisfied by (17) [8,20].

V. Experimental Results

A. Experimental Setup, Data Acquisition, and Analysis

The experimental setup involves a four stroke, 5.0 L, eight cylinder, Coyote-VP engine. The test is performed at FORD's Powertrain Research and Development Centre (PRDC). Vibration data has been recorded over four seconds using a charge-type piezoelectric accelerometer. The accelerometer has been attached to the engine lug in a premeditated position in order to detect faults of interest. Vibration data has been acquired using a PROSIG 5600 data acquisition system with built-in 16-bit analog-to-digital convertor card set at a sampling frequency of 32,768 Hz.

After data acquisition, the time domain data has been converted offline to the crank angle domain as shown in Fig. 7 using the cam identification (CID) sensor signal.

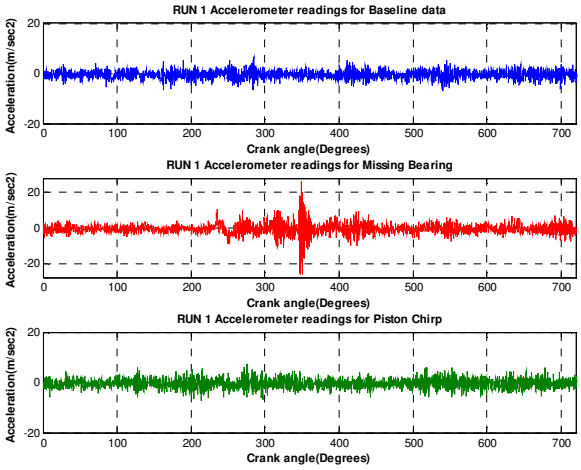


Fig. 7. Baseline, Missing bearing, and Piston Chirp vibration data in the crank angle domain

CID sensor is used to detect camshaft angle position. It's a non-contact sensor mounted on the engine and generate sinusoidal pulses at specific angles of $90^{\circ} - 120^{\circ} - 60^{\circ} - 120^{\circ} - 60^{\circ} - 180^{\circ} - 90^{\circ}$ per engine cycle. First sinusoidal pulse zero-crossing indicates that the first cylinder is 10° away from the top-dead-center (TDC). After transformation to the crank angle domain, data resampling is performed so that each engine cycle has the same number of points.

Two faults have been induced in the engine. They involve missing bearing fault (MB) and piston chirp (PC) fault. Vibration signals recorded from these two fault cases as well as the baseline fault-free engine case are used as a training data set for neural networks training.

B. SVSF, EKF, and BP Neural Networks Training

In this paper, fully connected feedforward multilayer perceptron with 281 input neurons representing vibration data points in the crank angle domain, two hidden layers with four neurons each, and three output units is used. The number of hidden layers and the number of hidden neurons are selected using trial and error to find the optimal results with minimal number of neurons. The trained network should be able to classify engines to either one of the two induced faults or to a baseline (fault-free) case as follows: (1, 0, 0: Baseline engine), (0, 1, 0: Piston Chirp fault detected), (0, 0, 1: Missing Bearing fault detected).

The test has been conducted through several runs, 30 engine cycles from each case stated before (i.e.: Baseline, MB, and PC) resulting in 90 training sets. Trained neural nets have been tested using 10 engine cycles from each case resulting in 30 testing set. Tesing data has never been used during network training.

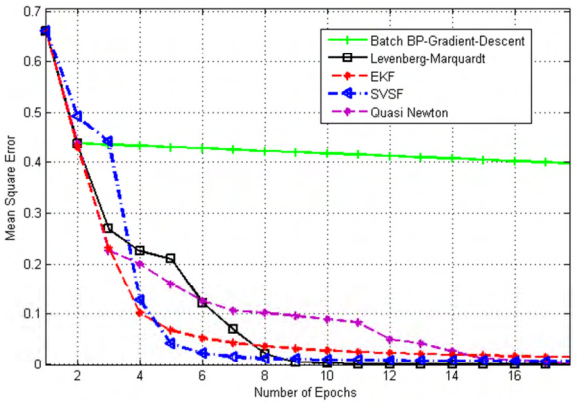


Fig. 8. Performance (Mean Squared Error) Variation vs. Number of Epochs

As mentioned earlier, networks were trained using the SVSF, EKF, batch first order, and second order BP algorithms (Levenberg-Marquardt and Quasi-Newton). Fig. 8 shows root mean square error (RMSE) variation for the first 17 epochs. The SVSF convergence properties outperform the conventional first order BP algorithms and it's comparable with the EKF and second order BP. At the sixth epoch, where training starts to stabilize, the SVSF reaches the least RMSE of 0.0221 followed by the EKF with RMSE of 0.0522. While at the 15th epoch, the LM achieves the least RMSE of $2.8e^{-8}$ followed by the SVSF with RMSE of 0.0065. Table I shows the trained networks RMSE using several training techniques after the 6th and the 15th epoch.

TABLE I
RMSE FOR VARIOUS TRAINING TECHNIQUES AFTER THE 6TH AND THE 15TH EPOCH

Training technique	RMSE, 6th	RMSE, 15th
Batch First-order BP	0.4276	0.4052
Levenberg-Marquardt	0.1215	$2.8e^{-8}$
Quasi-Newton	0.1261	0.0119
EKF	0.0522	0.0180
SVSF	0.0221	0.0065

Trained networks have been tested using 30 data sets (Fig.'s 9, 10, 11, 12 and 13), which demonstrate training and testing classification results for trained networks using first-order BP, second-order Levenberg-Marquardt, Quasi-Newton, EKF, and SVSF, respectively. Further details regarding the confusion matrix is attached in the Appendix. The EKF and SVSF are used in a global form and in multi-streaming fashion.

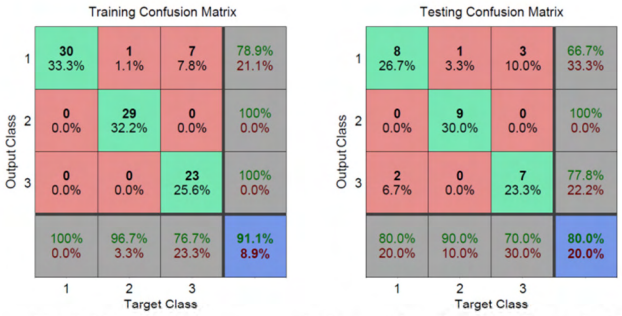


Fig. 9. Training and testing confusion matrices for batch first order BP

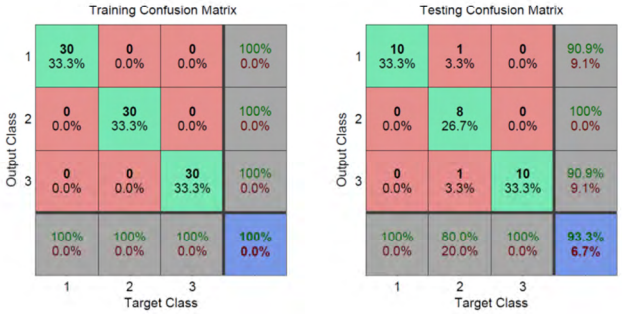


Fig. 10. Training and testing confusion matrices for Levenberg-Marquardt

Training and testing results after 30 epochs are summarized in Table II. The SVSF achieved the highest testing percentage in both training and testing along with Quasi-Newton followed by the EKF and Levenberg-Marquardt, and finally BP.

TABLE II
OVERALL TRAINING AND TESTING CLASSIFICATION RESULTS

Training technique	TRAINING %	TESTING %
First-order BP	91.1	80
Levenberg-Marquardt	100	93.3
Quasi-Newton	100	96.7
EKF	100	93.3
SVSF	100	96.7

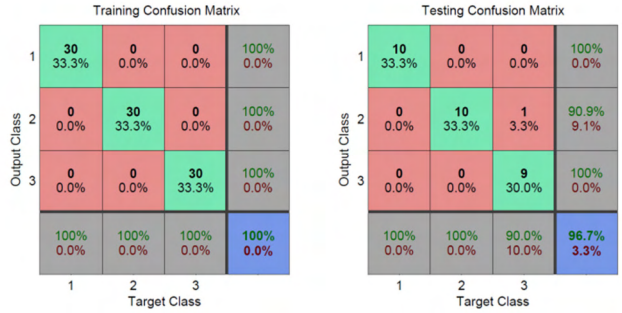


Fig. 11. Training and testing confusion matrices for Quasi-Newton

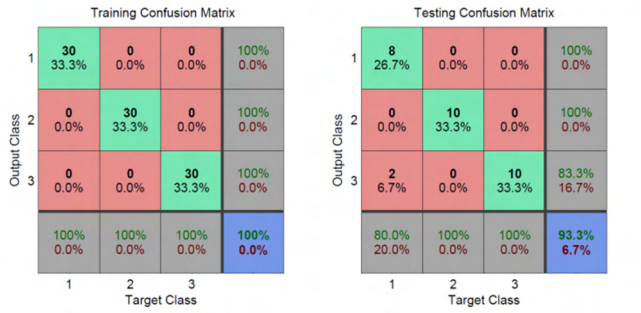


Fig. 12. Training and testing confusion matrices for EKF

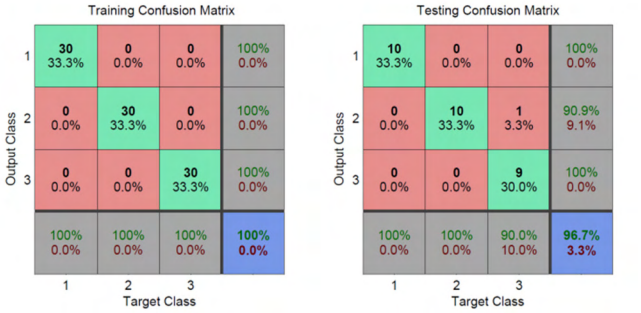


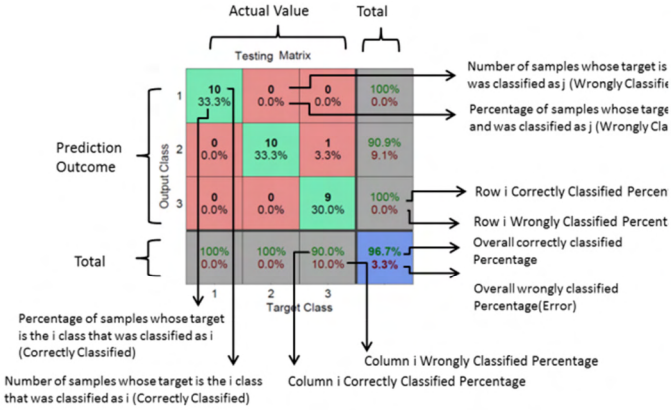
Fig. 13. Training and testing confusion matrices for SVSF

VI. Conclusion

In this research, the GSVSF has been successfully applied to train multilayered feed-forward networks and to detect and classify engine faults. The GSVSF demonstrated stability guarantee, fast speed of convergence, and high parameter estimation accuracy. The GSVSF training performance outperforms first order back propagation algorithms and it is comparable to the GEKF, and second order back propagation algorithms in terms of number of epochs and convergence speed. The GSVSF has shown better performance compared to GEKF in terms of generalization capability and in terms of RMSE variations during training. Further work involves application of the GSVSF to train recurrent multilayer perceptron (RMLP) networks for system identification.

VII. Appendix

The following figure demonstrates the confusion matrix in more detail.



VIII. Acknowledgement

The authors would like to thank Haran Arasaratnam (McMaster) for his help during data acquisition and analysis.

IX. References

[1] P. Werbos, "Backpropagation through time; what it does and how to do it," *Proceedings of the IEEE*, vol. 78, pp. 1550–1560, 1990.

[2] Simon Haykin, *Kalman filtering and neural networks*, 047136998504712215 46th ed., 2001.

[3] Felix Heimes, "Extended Kalman Filter Neural Network Training: Experimental Results and Algorithm Improvements," *IEEE*, vol. 0-7803-4778-1/98, 1998.

[4] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman algorithm," *Advances in Neural Information Processing Systems 1*, pp. 133–140, 1989.

[5] G.V. Puskorius and L.A. Feldkamp, "Decoupled extended Kalman filter training of feedforward layered networks," *Proc. IJCNNÕ91 I, Seattle*, pp. 771-777, 1991.

[6] R.J. Williams, "Training recurrent networks using the extended Kalman filter," *Proc. IJCNN'92 IV*, pp. 241-246, 1992.

[7] Pu Sun and Marko Kenneth, "Training recurrent neural networks for very high performance with the extended Kalman algorithm," *Intelligent Engineering Systems Through Artificial Neural Networks*, vol. 8, pp. 121-126, 1998.

[8] S. R. Habibi, "The Smooth Variable Structure Filter," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 1026-1059, 2007.

[9] S.R. Habibi and Burton R., "Parameter identification for a high-performance hydrostatic actuation system using the variable structure filter concept," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. v 129, n 2, pp. 229-235.

[10] S. Wang, S. Habibi, and Burton R., "A smooth variable structure filter for state estimation," *Control and Intelligent Systems*, vol. v 35, n 4, pp. 386-393, 2007.

[11] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs: NJ: Prentice-Hall, 1979.

[12] Trebaticky Peter and Jiri Pospichal, "Neural Network Training with Extended Kalman Filter Using Graphics Processing Unit," in *ICANN*, 2008, pp. 198-207.

[13] Youji Iiguni, Hideaki Sakai, and Hidekatsu Tokumaru, "A Real-Time Learning Algorithm for a Multilayered Neural Network Based on the Extended Kalman Filter," *IEEE transaction on signal processing*, vol. Vol. 40, No. 4., April 1992.

[14] L.A. Feldkamp and G.V. Puskorius, "'A signal processing framework based on dynamic networks with application to problems in adaptation, filtering and classification," in *IEEE*, 1998, pp. 2259-2277.

[15] Feldkamp L.A. and G.V. Puskorius, "Training controllers for robustness: multi-stream DEKF," in *IEEE international Conference on Neural Networks*, Orlando, 1994, pp. 2377-2382.

[16] Feldkamp L.A. and G.V. Puskorius, "Training of robust neurocontrollers," in *IEEE International Conference on Decision and Control*, Orlando, 1994, pp. 2754-2760.

[17] S. R. Habibi and R. Burton, "The Variable Structure Filter," *Journal of Dynamic Systems, Measurement, and Control (ASME)*, vol. 125, pp. 287-293, September 2003.

[18] S. R. Habibi and R. Burton, "Parameter Identification for a High Performance Hydrostatic Actuation System using the Variable Structure Filter Concept," *ASME Journal of Dynamic Systems, Measurement, and Control*, 2007.

[19] S. A. Gadsden and S. R. Habibi, "A New Form of the Smooth Variable Structure Filter with a Covariance Derivation," in *IEEE Conference on Decision and Control*, Atlanta, Georgia, 2010.

[20] P. A. Cook, *Nonlinear Dynamical Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.