# AN ADAPTIVE NEWTON-LIKE TRAINING ALGORITHM FOR NONLINEAR FILTERS WHICH HAVE EMBEDDED MEMORY

*M.Rabinowitz, G.M.Gutt and G.F.Franklin*

Gravity Probe B, Stanford University, Stanford, CA 94306, USA

## ABSTRACT

We discuss herein filtering problems involving the emulation of nonlinear systems which have embedded dynamics and where the truth model consists only of an input and output sequence. It has been shown [1] that steepest descent training algorithms such as *Backpropagation-Through-Time* (BPTT) are locally $H^\infty$ optimal for applications where training inputs vary at each weight update, or *training epoch*. However, when the same data set is used for several epochs, as is typical for nonlinear system identification, BPTT is suboptimal, and a new adaptive Gauss-Newton technique, which generates updates closer to the Newton update direction, is preferable. We discuss the application of the technique to IIR and FIR filter architectures for pre- and post-compensating nonlinear systems. Comparisons to canonical methods, namely BPTT, Kalman Filtering, Gauss-Newton, and Broyden-Fletcher-Goldfarb-Shanno (BFGS) are favorably made.

## 1. BACKGROUND

Over the last few decades, linear dynamic system identification has evolved into a rigorous field [2, 3]. However, nonlinear system identification is still a nascent science, demonstrating a need for new ideas and techniques. We'll consider systems consisting of cascaded linear and nonlinear components, where only an input-output sequence is available: Wiener proposed the use of functional series which were orthogonal for white Gaussian inputs [4]. Such polynomial-type functional sets, which modeled nonlinear dynamics as a series of convolutions of increasing order, had been previously described by Volterra [5]. Identification of systems using finite-order Volterra-Series expansions have been investigated for deterministic, Gaussian and Non-Gaussian inputs [6, 7, 8]. While the Volterra-based techniques accommodate a vast class of nonlinear systems, the characteristic functions or *Volterra Kernels* provide excessive degrees of computational freedom, and so place excessively stringent requirements on the quality and quantity of identification data. Several block-oriented approaches, which incorporate knowledge of the system architecture to reduce the number of free parameters, have been addressed in the literature - see [9] for a structured overview. In the last decade, Backpropagation (BP) has emerged as a standard technique for training multi-layer adaptive filters to implement static functions, to operate on tapped-delay line inputs, and in recursive filters where the desired outputs of each filter layer are known - [10, 11, 12]. The principle of static BP was extended to networks with embedded memory via *backpropagation-through-time* (BPTT) [13, 14]. In essence, BPTT is a steepest descent algorithms, applied successively to each layer in a nonlinear filter. BBTT can be improved upon by a Newton-like technique.

## 2. OVERVIEW OF THE METHOD

Before delving into algorithmic details, we describe in fig. 1 the flow diagram according to which the components of the method are combined. The solid lines indicate the flow of data between steps; the dashed lines indicate the flow of control. In order to limit the filter's degrees of computational freedom [15, 16, 17, 18] the desired dynamic nonlinear operator is decomposed to linear dynamic elements, and nonlinear elements. Based upon this decomposition, we taylor-make a filter architecture, represented by function $h$, and consisting of adaptive static nonlinear components, and adaptive linear components. Next, using an initial set of filter parameters $\mathbf{w}$, and architecture $h$, we propagate a training input sequence $\{u_n\}$ through the filter to obtain an output sequence $\{\hat{y}_n\}$. An error signal $\{e_n\}$ is created by subtracting from $\{\hat{y}_n\}$ the desired output sequence $\{y_n\}$. Using $h$, $\{e_n\}$, and $\mathbf{w}$, a single matrix $\nabla_{\mathbf{w}} h$ is constructed which relates each network parameter to the error it produces over the full sequence $\{e_n\}$. Using $\nabla_{\mathbf{w}} h$, $\{e_n\}$, and a learning rate $\mu$, an adaptive Newton-like update algorithm is used to determine a weight update vector $\Delta\mathbf{w}$. A temporary parameter set $\mathbf{w}_t$ is created by summing $\Delta\mathbf{w}$ and $\mathbf{w}$. Using $\mathbf{w}_t$ and $h$, the training input sequence $\{u_n\}$ is again propagated through the filter to obtain output sequence $\{\hat{y}_{t_n}\}$. $\{e_{t_n}\}$ is created by differencing $\{\hat{y}_{t_n}\}$ and $\{y_n\}$. Costs are computed from the error sequences, such as $J = \sum_n e_n^2$ and $J_t = \sum_n e_{t_n}^2$. If $J > J_t$, $\mathbf{w} \leftarrow \mathbf{w}_t$, $\mu$ is increased by a constant factor, and the input sequence is again propagated through the filter. Otherwise, $\mu$ is decreased by a constant

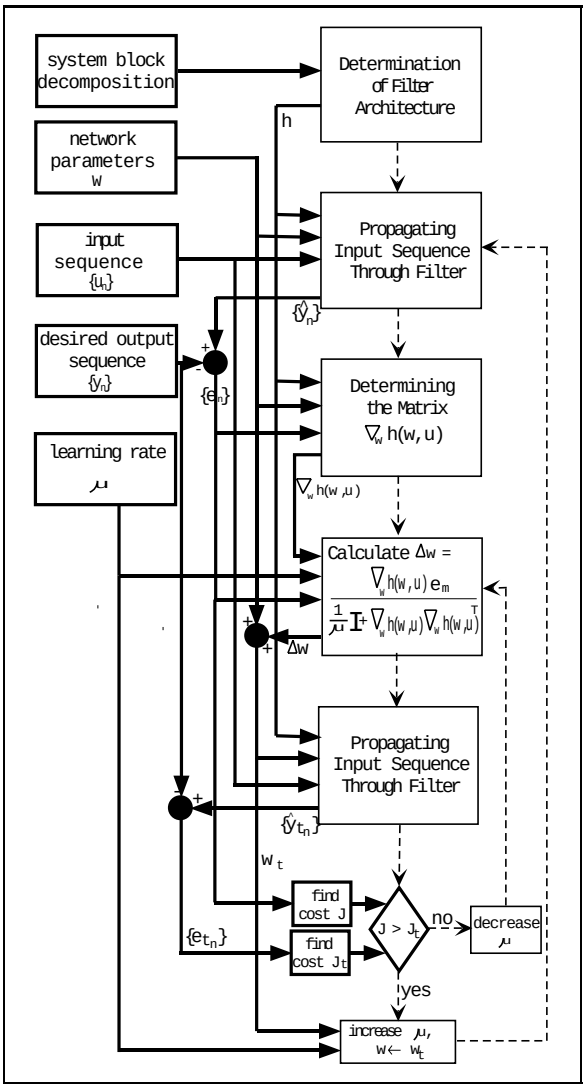


Figure 1: Overview of the Method

factor and $\Delta \mathbf{w}$ is recalculated.

## 3. THE NEWTON-LIKE TRAINING ALGORITHM

In this section we provide some rationale for the adaptive training algorithm - a more thorough derivation may be found in [16]. We seek, at each training *epoch*, an iterative update of the weights vector. At epoch $k$, one may describe the sequence of desired training outputs as a vector:

$$\mathbf{y}_k = h(\mathbf{w}^*, \mathbf{u}_k) + \mathbf{v}_k \tag{1}$$

where $\mathbf{w}^*$ is the vector of ideal weights which training seeks to discover, $\mathbf{u}_k$ is a vector of inputs at epoch $k$ [1] and $\mathbf{v}_k$ is a vector of unknown disturbances. If the data in $\mathbf{u}_k$ and $\mathbf{y}_k$ are sampled from real-world continuous signals, then $\mathbf{v}_k$ can be regarded as a vector of noise on the ideal system inputs or sampled outputs. Alternatively, $\mathbf{v}_k$ can be regarded as error resulting from the architectural imperfections of the network, such as emulating IIR linear dynamics with a tapped delay line. For a weights estimate at epoch $k$, $\mathbf{w}_{k-1}$, we define a cost function in terms of the error in the absence of disturbance $\mathbf{v}_k$:

$$J(\mathbf{w}_k, \mathbf{u}_k) = \frac{1}{2}\left(h(\mathbf{w}_k, \mathbf{u}_k) - h(\mathbf{w}, \mathbf{u}_k)\right)^T \left(h(\mathbf{w}_k, \mathbf{u}_k) - h(\mathbf{w}, \mathbf{u}_k)\right) \tag{2}$$

By evoking the mean value theorem [19], it can be shown [16] that as $\mathbf{w}_k \to \mathbf{w}^*$, we may approximate cost as:

$$\hat{J}(\mathbf{w}_k, \mathbf{u}_k) = \frac{1}{2}\left(\mathbf{e}_{mk} - \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k)^T (\mathbf{w}_k - \mathbf{w}_{k-1})\right)^T \left(\mathbf{e}_{mk} - \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k)^T (\mathbf{w}_k - \mathbf{w}_{k-1})\right) \tag{3}$$

We find the derivative of this cost with respect to $\mathbf{w}_k$.

$$\nabla_{\mathbf{w}} \hat{J}(\Delta \mathbf{w}_{k-1}) = -\nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k) \left(\mathbf{e}_{mk} - \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k)^T (\mathbf{w}_k - \mathbf{w}_{k-1})\right) \tag{4}$$

Suppose one demands that the step taken at time $k-1$, $\Delta \mathbf{w}_{k-1} = \mathbf{w}_k - \mathbf{w}_{k-1}$, be co-linear with the gradient of the quadratic cost estimate at the new location, $\mathbf{w}_k$, to which we are stepping:

$$\Delta \mathbf{w}_{k-1} = -\mu_k \nabla_{\mathbf{w}} \hat{J}(\mathbf{w}_k, \mathbf{u}_k) = \mu_k \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k) \left(\mathbf{e}_{mk} - \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k)^T \Delta \mathbf{w}_{k-1}\right) \tag{5}$$

For a quadratic cost, this restriction would achieve updates which move more directly towards the quadratic cost minimum than a steepest descent update. In our case, since $\mu_k$ can be freely adjusted, we can solve (5) for the update step:

$$\Delta \mathbf{w}_{k-1} = \left(\frac{1}{\mu_k} + \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k) \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k)^T\right)^{-1} \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k) \mathbf{e}_{mk} \tag{6}$$

As one decreases the bandwidth of the input signal $\mathbf{u}_k$, or increases the degrees of freedom in the filter architecture, one increases the condition number of the matrix

$$\nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k) \nabla_{\mathbf{w}} h(\mathbf{w}_{k-1}, \mathbf{u}_k)^T \tag{7}$$

If $\mu_k$ is too large the poor conditioning of the inverse in equ. (6) can detrimentally effect the update direction. Conversely, if $\mu_k$ is too small, convergence is retarded. Consequently, we allow $\mu_k$ to vary based on the performance of the training updates as described in fig. 1.

[1] The subscript $k$ on $\mathbf{y}_k$ and $\mathbf{u}_k$ indicates that the inputs, and desired output set can be changed at each epoch, however, for applications with non-quadratic cost these are typically held constant over several epochs.

## 4. FINDING THE FULL DERIVATIVE MATRIX $\nabla_{\mathbf{W}} H$ FOR A GENERAL IIR FILTER ARCHITECTURE

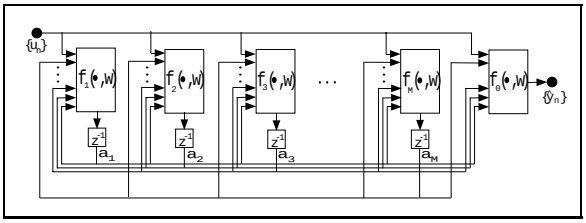


Figure 2: Generic Nonlinear Recursive Filter Architecture

The general approach described here applies to a vast range of filter architectures. It can be considerably refined for computational efficiency for specific architectures such as FIR Volterra networks [16, 20]. The generic architecture of this discussion is displayed in fig. 2. The system has $M$ states, represented at time $n$ by the state vector $\mathbf{a}_n = [a_{1\,n} \ldots a_{M\,n}]^T$ [2]. The subsequent value of each state in the filter or system is some function of the current states, the inputs and the set of parameters within the network

$$\mathbf{a}_{n+1} = \begin{bmatrix} a_{1\,n+1} \\ \vdots \\ a_{M\,n+1} \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{a}_n, u_n, \mathbf{w}) \\ \vdots \\ f_M(\mathbf{a}_n, u_n, \mathbf{w}) \end{bmatrix} \tag{8}$$

where $\mathbf{w} = [w_1 \ldots w_V]^T$ is a vector of length $V$, containing all adaptable parameters in the network, and $\{u_n\}$ $n = 1 \ldots N$ is the input sequence to the system. For the sake of clarity, we assume that the states of the system are all 0 before excitation with the input sequence $\{u_n\}$. Each output of the filter is generated by the function $\hat{y}_n = f_0(\mathbf{a}_n, u_n, \mathbf{w})$. The sequence of outputs is a vector of length $N$ which we denote $h(\mathbf{w}, \mathbf{u})$. Our task is to determine the full derivative matrix $\nabla_{\mathbf{w}} h(\mathbf{w}, \mathbf{u})$, which relates each weight to error produced over the full time sequence.

In order to calculate the dependence of some output $\hat{y}_n$ on $\mathbf{w}$, we employ a partial derivative expansion with respect to the state vector:

$$\nabla_{\mathbf{w}} \hat{y}_n = \sum_{i=1}^{n} \nabla_{\mathbf{a}_i} \hat{y}_n \nabla_{\mathbf{w}} \mathbf{a}_i \tag{9}$$

The term $\nabla_{\mathbf{w}} \mathbf{a}_i$ in equ.(9) can be directly calculated

$$\nabla_{\mathbf{w}} \mathbf{a}_i = \begin{bmatrix} \frac{\partial a_{1\,i}}{\partial w_1} & \cdots & \frac{\partial a_{1\,i}}{\partial w_V} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{M\,i}}{\partial w_1} & \cdots & \frac{\partial a_{M\,i}}{\partial w_V} \end{bmatrix} \tag{10}$$

[2] For this section, we have dropped any indication of the training epoch, $k$, simply for notational clarity

$$= \begin{bmatrix} \frac{\partial f_1(\mathbf{a}_i, u_i, \mathbf{w})}{\partial w_1} & \cdots & \frac{\partial f_1(\mathbf{a}_i, u_i, \mathbf{w})}{\partial w_V} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{a}_i, u_i, \mathbf{w})}{\partial w_1} & \cdots & \frac{\partial f_M(\mathbf{a}_i, u_i, \mathbf{w})}{\partial w_V} \end{bmatrix}$$

In order to calculate the term $\nabla_{\mathbf{a}_i} \hat{y}_n$ in equ. (9), we again apply a partial derivative expansion as follows:

$$\nabla_{\mathbf{a}_i} \hat{y}_n = \nabla_{\mathbf{a}_n} \hat{y}_n \nabla_{\mathbf{a}_{n-1}} \mathbf{a}_n \ldots \nabla_{\mathbf{a}_i} \mathbf{a}_{i+1} \tag{11}$$

Each of the terms in (11) can now be computed:

$$\nabla_{\mathbf{a}_i} \mathbf{a}_{i+1} = \begin{bmatrix} \frac{\partial a_{1\,i+1}}{\partial a_{1\,i}} & \cdots & \frac{\partial a_{1\,i+1}}{\partial a_{M\,i}} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_{M\,i+1}}{\partial a_{1\,i}} & \cdots & \frac{\partial a_{M\,i+1}}{\partial a_{M\,i}} \end{bmatrix} \tag{12}$$

$$\nabla_{\mathbf{a}_n} \hat{y}_n = \begin{bmatrix} \frac{\partial \hat{y}_n}{\partial a_{1\,n}} \cdots \frac{\partial \hat{y}_n}{\partial a_{M\,n}} \end{bmatrix} \tag{13}$$

$$= \begin{bmatrix} \frac{\partial f_0(\mathbf{a}_n, u_n, \mathbf{w})}{\partial a_{1\,n}} \cdots \frac{\partial f_0(\mathbf{a}_n, u_n, \mathbf{w})}{\partial a_{M\,n}} \end{bmatrix}$$

Based upon these equations, we present below an algorithm by which $\nabla_{\mathbf{w}} h(\mathbf{w}, \mathbf{u})$ can be determined. We begin with two all-zeros matrices, $\mathbf{H} = \mathbf{0}^{N \times V}$ and $\mathbf{J} = \mathbf{0}^{N \times M}$. Note that the notation $\mathbf{H}(i, :)$ refers to all the columns at row $i$ in $\mathbf{H}$, $\mathbf{H}(:, i)$ refers to all the rows in column $i$ of $\mathbf{H}$, and $\mathbf{H}(i : n, :)$ refers to the matrix block defined by all the columns and the rows from $i$ to $n$.

1. $\mathbf{J}(N, :) = \nabla_{\mathbf{a}_N} \hat{y}_N$
2. $\mathbf{H}(N, :) = \nabla_{\mathbf{a}_N} \hat{y}_N \nabla_{\mathbf{w}} \mathbf{a}_N$
3. for i = N-1:1 step -1
4. $\quad \mathbf{J}(i+1 : N, :) \leftarrow \mathbf{J}(i+1 : N, :) \nabla_{\mathbf{a}_i} \mathbf{a}_{i+1}$
5. $\quad \mathbf{J}(i : N, :) \leftarrow \nabla_{\mathbf{a}_i} \hat{y}_i$
6. $\quad \mathbf{H}(i : N, :) \leftarrow \mathbf{H}(i : N, :) + \mathbf{J}(i : N, :) \nabla_{\mathbf{w}} \mathbf{a}_i$
7. end
8. $\nabla_{\mathbf{w}} h(\mathbf{w}, \mathbf{u}) \leftarrow \mathbf{H}$

## 5. EXAMPLES OF TAYLOR-MADE ADAPTIVE DYNAMIC NONLINEAR FILTERS

In this section, we describe two sample applications of algorithm (6) and illustrate how one may Taylor-make a temporal network to perform a specific nonlinear dynamic operation, such network being efficiently trained with the new technique. Consider first the identification and inversion of a simple Wiener-Type Nonlinear System. The Wiener architecture illustrated in fig. 3 models an audio amplifier which exhibits crossover distortion. The memoryless nonlinearity

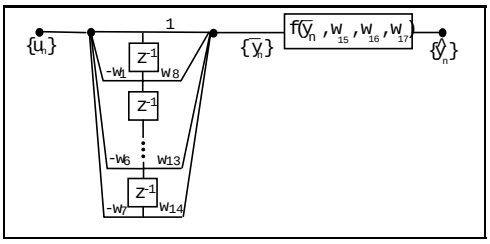


Figure 3: Wiener model of a nonlinear amplifier

at the amplifier's output is emulated with a parameterized function:

$$f(\bar{y_n}, w_{15}, w_{16}, w_{17}) = w_{15}\frac{1+e^{w_{16}\bar{y}_n}}{1-e^{w_{16}\bar{y}_n}} + w_{17}\bar{y}_n \quad (14)$$

The linear dynamics of the amplifier are emulated over the audible band with a $7^{th}$ order IIR digital filter.

The data for training the adjustable parameter set $\{w\}$ is gathered according to the method of fig. 4. The amplifier is excited with a chirp signal ranging in frequency from $19.2 - 0kHz$, summed with zero-mean Gaussian noise. The amplifier outputs are sampled to obtain the desired network output sequence. The network input sequence is obtained by sampling the input signal to the amplifier. Algorithm (6) is then used to identify the parameters of the filter in fig. 3.

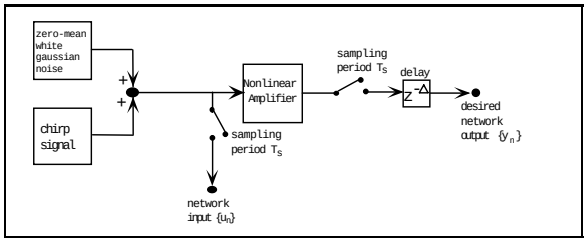


Figure 4: Obtaining an input-output truth model

In an IIR system, a non-minimum phase zero cannot be precisely dynamically compensated, since it will cause an unstable pole in the compensator. Consequently, if the identification yields a non-minimum-phase zero for the linear filter, the output sequence is delayed relative to the input sequence and the identification is repeated. For input sequence $\{u_n\}$, the filter is trained to estimate outputs $\{y_{n-1}\}$. Successive delays may be added until the zeros of the identified linear filter are minimum-phase. Once the filter parameters are identified, the linear and nonlinear blocks are analytically inverted using well-known techniques [21].

A lowpass digital filter with cutoff at roughly $20kHz$ is cascaded with the IIR filter inverse to limit high-frequency gain. A signal is pre-warped by the inverse nonlinearity and then the inverse linear dynamics before being input to the amplifier. The A-D and D-A conversions in this case are performed with a 16 bit AD1847 Codec, and the pre-warping of the signal is achieved with an ADSP2181 microprocessor. Fig. 5 shows the spectral response of the nonlinear amplifier when excited with a dual tone test signal. Fig. 6 displays the spectral response of the amplifier to a pre-warped dual tone. Note that the amplitudes of nonlinear distortion harmonics have been reduced in the pre-warped signal by more than 20dB.

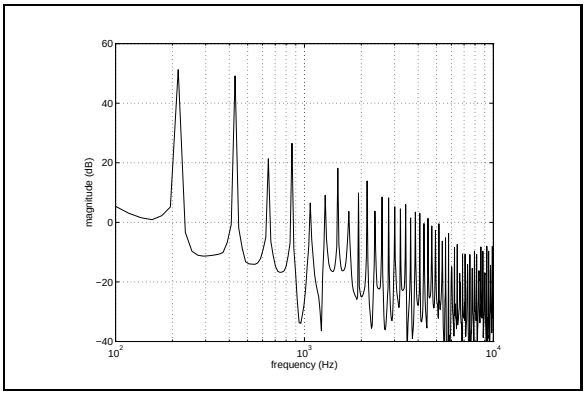


Figure 5: Spectral response of the amplifier to a dual tone signal

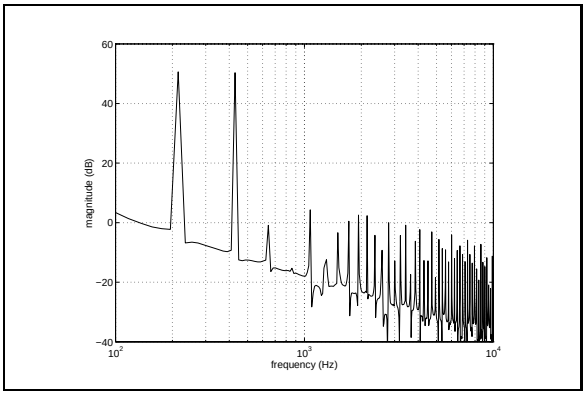


Figure 6: Spectral response of the amplifier to a dual tone signal

It should be noted that in the case of a simple Wiener architecture, the linear and nonlinear blocks can be separately identified using error prediction algorithms [22, 23]. These techniques cannot be applied, however, if the system is more complex. Consider our second example, the generic tracking system displayed in fig. 7. This system can be considered as some nonlinear sensor, feeding electronics with nonlinear components and linear dynamics, which output to some plant. The dynamics on the feedback path could represent parasitic capacitance in the system. For this example,

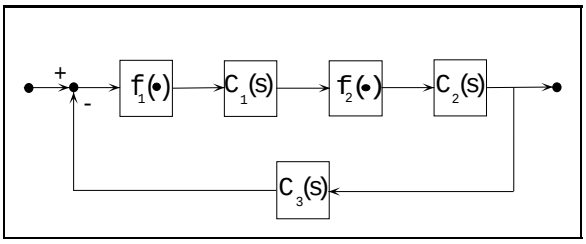


Figure 7: Block Diagram of a tracking system with static nonlinearities, a control law and parasitic filtering on the feedback path

the dynamic transfer functions and nonlinearities can be described:

$$
\begin{aligned}
C_1(s) &= \frac{1e5s + 3e7}{s^2 + 1.007e5s + 7e7} \qquad (15)\\
C_2(s) &= \frac{1e7}{s^2 + 1e5s}\\
C_3(s) &= \frac{1e8}{s^2 + 1.01e5s + 1e8}\\
f_1(u) &= u + \frac{1}{2}u^2 + \frac{1}{3}u^3\\
f_2(u) &= 1.2551u^7 - 1.4337u^5 + 0.7038u^3 + 0.9540u
\end{aligned}
$$

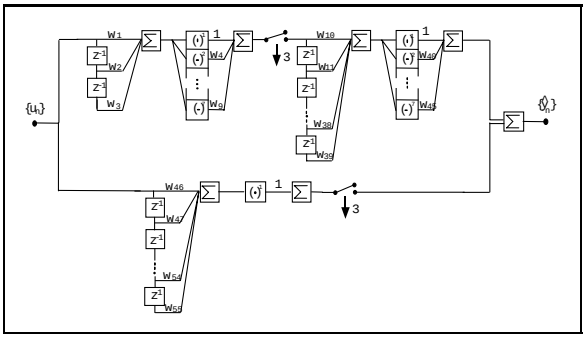

Figure 8: The network employed to invert the tracking system described by equ. (15)

We would like the system outputs to exactly track system inputs, however, with nonlinearities, dynamics and feedback, the outputs are a severely distorted version of the original system inputs. We seek an adaptive compensator which can be trained to map the system outputs to the original undistorted inputs. Based on an estimate of the time constants of the linear dynamics to be emulated, and an estimate of the order of the nonlinear functions to be emulated, we can Taylor-make an architecture for inverting the system of fig. 7 as shown in fig. 8. This filter uses limited degrees of computational freedom, and has good generalization characteristics[3]. The training and performance of this filter architecture are more extensively discussed in [16]. We will focus here only on the relative performance of canonical training algorithms applied to the architecture. The training input signal to the system of fig. 7 consisted of a chirp with added Gaussian noise. Figure (9) shows the RMS network error for $100$ parameter updates using algorithm (6), and $200$ updates respectively using other suitable optimization algorithms namely the Kalman Filter, the Gauss-Newton technique, the BFGS algorithm with a line search [24], and BPTT. The line search for BFGS was conducted using the method of false position [25]. Notice that all Newton-like techniques outperform BPTT. The superior convergence rate and cost minimization achieved with BFGS and (6) are clearly evident. Note that in contrast to BFGS, (6) does not require a line search so each epoch involves substantially less computation than is required for a BFGS update. By applying the trained filter to post-linearizing the nonlinear tracking system (15), the distortion peaks in response to multi-tone signals can be reduced by roughly 35dB [16].

[3]A rigorous treatment of generalization can be found in [17, 18]

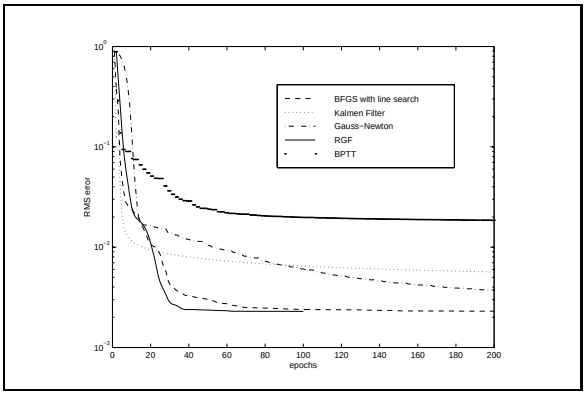


Figure 9: The RMS error of the network over 100 epochs using (6), and over 200 with an assortment of canonical techniques

## 6. CONCLUSION

We have described an adaptive Newton-like algorithm for training dynamic nonlinear filters which contain embedded memory or which use recursion. These architectures are applicable to problems involving the emulation, control, pre- or post- processing of nonlinear dynamic systems. Such problems are best addressed by taylor-made filters which are structured so as to restrict their degrees of computational freedom and which may be trained by the novel algorithm. The approach was illustrated in application to the pre- and post-linearization respectively of a nonlinear audio amplifier and a nonlinear feedback tracking system. For a wide range of problems using input/output truth models, the al-

gorithm converges more quickly and more accurately than do BPTT, the Gauss-Newton method, Kalman Filtering and the Broyden-Fletcher-Goldfarb-Shanno technique.

## Acknowledgments

This work was supported by the Stanford Gravity Probe-B project under NASA contract NAS 8-36125.

### 7. REFERENCES

[1] T. K. B. Hassibi, "$h^\infty$ optimal training algorithms and their relation to backpropagation," *Proceedings of the NIPS94 - Neural Information Processing Systems: Natural and Synthetic*, pp. 191–198, Nov-Dec 1994.

[2] P.Eykhoff, *System Identification*. John Wiley and Sons, 1st ed., 1979.

[3] R. S.P.Boyd and M.K.Lau, "Set-membership identification of systems with parametric uncertainty," *IEEE Transactions on Automatic Control*, vol. 37, pp. 929–941, July 1992.

[4] N.Wiener, *Nonlinear Problems in Random Theory*. Wiley, New-York, 1958.

[5] V.Volterra, *Theory of Functions*. Blackie and Sons, Glascow, 1930.

[6] S. Boyd, *Volterra Series: Engineering Fundamentals*. PhD thesis, University of California at Berkeley, 1985.

[7] S. Nam, *Application of Higher-order Spectral Analyses to Nonlinear System Identification*. PhD thesis, University of Texas at Austin, 1990.

[8] K.I.Kim and E.J.Powers, "Orthogonalised frequency domain volterra model for non-gaussian inputs," *IEE Proceedings-F*, vol. 140, no. 6, pp. 403–409, 1993.

[9] J. Brendat, *Nonlinear System Analyses and Identification from Random Data*. New York, Wiley, 1990.

[10] S. S. B. Widrow, *Adaptive Signal Processing*. Prentice Hall, 1985.

[11] D. Hyland, "U.s. patent: Multiprocessor system and method for identification and adaptive control of dynamic systems," Tech. Rep. 5,796,920, US Patent and Trademark Office, August 1998.

[12] S. W. et al., "U.s. patent: Nonlinear adaptive filter," Tech. Rep. 4,843,583, US Patent and Trademark Office, June 1989.

[13] D. Nguyen, *Applications of Neural Networks in Adaptive Control*. PhD thesis, Stanford University, June 1991.

[14] D. Hanks, "U.s. patent: Adaptive feedback system for controlling head/arm position in a disk drive," Tech. Rep. 5,548,192, US Patent and Trademark Office, August 1996.

[15] M. Rabinowitz, G. Franklin, and G.M.Gutt, "Adaptive post linearization of dynamic nonlinear systems with artificial neural networks," *ASME Journal of Dynamic Systems, Measurement and Control*, unknown 1999.

[16] M. Rabinowitz, G. Franklin, and G.M.Gutt, "An adaptive gauss-newton algorithm for training multi-layer nonlinear filters which have embedded memory," *Cicuits, Systems and Signal Processing*, vol. 18, no. 4, 1999.

[17] D.H.Wolpert, "A mathematical theory of generalization: Part 1," *Complex Systems*, vol. 4, pp. pp 151–200, 1990.

[18] D.H.Wolpert, "A mathematical theory of generalization: Part 2," *Complex Systems*, vol. 4, pp. pp 201–249, 1990.

[19] M.Vidyasagar, *Nonlinear System Analyses*. Prentise-Hall, 2nd ed., 1993.

[20] M.Rabinowitz, "Patent application: A method and system for training dynamic nonlinear adaptive filters which have embedded memory," Tech. Rep. 09/201,927, Application: US Patent and Trademark Office, December 1999.

[21] A. Oppenheim and R.W.Schafer, *Discrete-Time Signal Processing*. Prentice-Hall, Inc., 1989.

[22] T.Wigren, "Recursive predicition identification using the nonlinear wiener model," *Automatica*, vol. 29, pp. pp 1011–1025, 1993.

[23] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Addison-Wesley, 2nd ed., 1991.

[24] D. Bertsekas, *Nonlinear Programming*, vol. 1. Athena Scientific, 2nd ed., 1995.

[25] D. G. Luenberger, *Linear and Nonlinear Programming*. Addison-Wesley, 2nd ed., 1984.