Neurointerfaces : Principles

Bernard Widrow & Marcelo M. Lamego
Electrical Engineering Dept., Stanford University, CA 94305
mmlamego@ieee.org

Abstract

A Neurointerface is a trainable filter based on neural networks that serves as a coupler between a human operator and a nonlinear system or plant that is to be controlled or directed. The purpose of the coupler is to ease the task of the human controller. The equations of the plant are assumed to be known. If the plant is unstable, it must first be stabilized by feedback. Using the plant equations, off-line automatic learning algorithms are developed for training the weights of the Neurointerface. If the plant is subject to disturbance, an adaptive disturbance canceller is used to minimize the effect. The Neurointerface can be adapted to be an inverse of the plant, so that when it is cascaded with the plant, the overall plant response closely approximates the human command input.

1 Introduction

For many tasks, productivity, safety, and liability conditions require a considerable degree of skill from human operators. In order to overcome lack of skill, special man-machine interfaces may be adopted. The basic idea is to change the operational space through a Neural Network (NN), allowing the human operator to interact with the process through less-specialized commands. Hence, the operator devotes his attention to solving a less complex problem, directly at the task level. The objective is to improve the productivity and safety levels of such tasks even in the case of unskilled operators.

This paper aims at showing how NNs can be applied to the design of man-machine interfaces for practical real-time problems. The term "Neurointerface" is chosen to emphasize the use of NNs for the solution of man-machine interface problems.

The design of Neurointerfaces involves the training of NNs, which are incorporated in nonlinear adaptive filters. One searches for the set of parameters (NN weights) that are best solutions for a predefined cost function (mean-square error). The Backpropagation (BP) algorithm [1] is used in this work for the design of Neurointerfaces.

A real man-machine interface is used here as a case study. The problem is to design a Neurointerface that facilitates the backing of a truck connected to a double-trailer configuration under human steering control. The steering commands of the human driver are fed to the Neurointerface whose output controls the steering angle of the front wheels of the truck.

2 What is a Neurointerface?

A Neurointerface may be thought of as a form of inverse of the plant to be controlled. A desired plant response can be realized by driving the plant with an inverse controller whose input consists of simple command signals applied by a human operator. Thus, an unskilled operator using a Neurointerface can produce actions like those of an experienced operator.

While cases might exist in which the Neurointerface provides only an approximation to the actions taken by an expert operator, the change of operational space made by the Neurointerface allows the human operator to interact with the process through easier less-specialized actions. This is the case, for instance, in backing the truck and trailers. The Neurointerface may be considered as a black box that takes commands from the driver (desired direction of the trailer back part) and provides the necessary actions (steer the wheels) in order to achieve such a goal. The angle between cab and first trailer and the angle between first and second trailer contain sufficient information to obtain an approximate inverse modeling of the system.

It should be noted that the driver is not eliminated in this work. Nguyen and Widrow (1990) [2] described a NN that provided full automation in backing a trailer truck to a loading dock and indeed, eliminating the need for the driver. In the present work, human action is essential. In fact, the driver is concerned with providing the desired spatial trajectory, free of obstacles and normally the shortest one.

0-7803-5800-7/00/$10.00 ©2000 IEEE

The truck-backing-up exercise is a kinematics inverse modeling problem. It means that the dynamic effects that may occur during the operation are not significant. The Neurointerface can also be applied to dynamic inverse modeling problems. A good example of a dynamic system that could be controlled by a Neurointerface is a humanly-operated construction crane. A flexible cable does the coupling between the trolley and the load. Normally, movements in the trolley generate oscillations in the load. Thus, the crane operator is concerned when shifting the load from one point to another about achieving movement free of oscillations. Here, the Neurointerface may be regarded as a black box that takes commands from the crane operator (desired trajectory of the load) and provides the necessary actions (actuation on each degree of freedom of the crane) in order to provide a smooth load movement.

Human control of robotic systems is another potential application area for the Neurointerface idea.

The Neurointerface is designed to operate in real time. The training procedure is generally performed off-line, before the trained Neurointerface is used in the real system. Fig. 1 shows a cascade of a trained Neurointerface driving the actual process (nonlinear plant). This is the basic configuration in which the Neurointerface is supposed to work. Feedback is provided by the human operator sensing and observing the plant output and changing the control input as required to get the desired plant response. The relation between the plant output and the Neurointerface input must be simple however in order for the human operator to be able to control the plant.

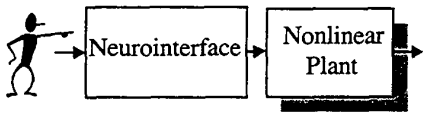

Figure 1: A cascade of the trained Neurointerface and the plant.

The basic topology of a Neurointerface is shown in Fig. 2. It is a feedforward nonlinear adaptive filter consisting of a tapped delay line connected to a multi-layer NN. The weights $w_0, w_1, \ldots, w_n$ are adjusted automatically by a learning process.

3 The Truck Backer

The kinematic equations for the motion of the truck and double trailers are easily derived from geometric

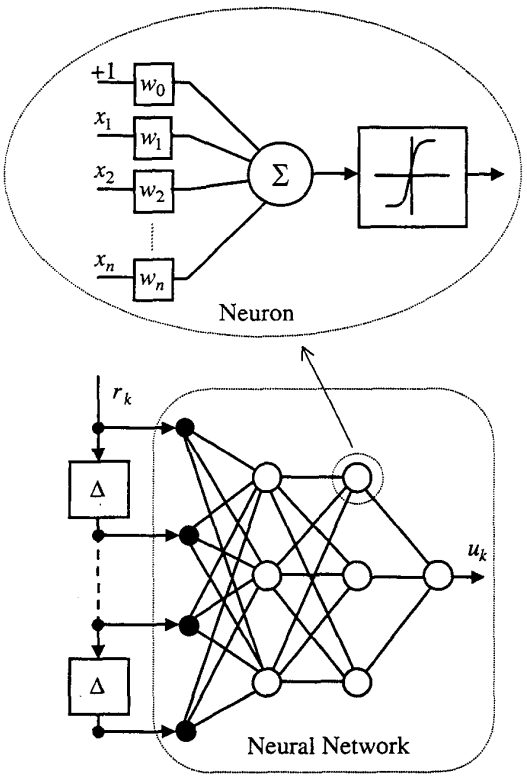

Figure 2: A feedforward nonlinear adaptive filter incorporating a three-layer NN.

considerations. Regarding the schematic diagram of the truck and trailers shown in Fig. 3, these equations are

$$\frac{\partial \theta_2}{\partial t} = v\left(\frac{\sin\theta_2}{L2} + \frac{\tan\theta_1}{L1}\right) \qquad (1)$$
$$\frac{\partial \theta_3}{\partial t} = v\left(-\frac{\sin\theta_2}{L2} + \frac{\cos\theta_2 \sin\theta_3}{L3}\right),$$

where v is the backing speed of the truck and $L1$, $L2$ and $L3$ are, respectively, the effective lengths of the truck, the first and second trailers.

The truck backer is an interesting example. It represents the control of a nonlinear unstable system. Our goal is to control nonlinear unstable systems under human direction. We assume in this work that the plant to be controlled is known. The backing truck and trailers is a good example.

The first step is to stabilize the unstable plant about an equilibrium point, and this can be done in many cases by making use of negative feedback with fixed gains.

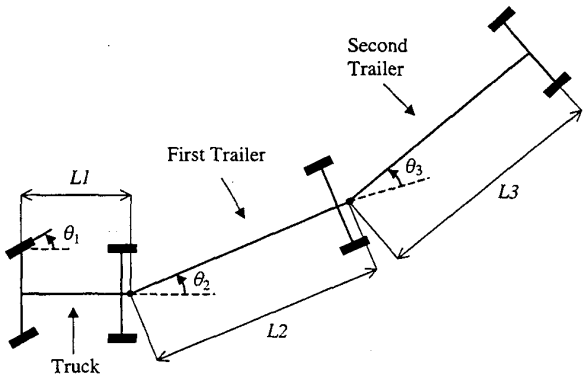

Figure 3: Schematic diagram of a truck and two trailers.

The idea is illustrated in Fig. 4.

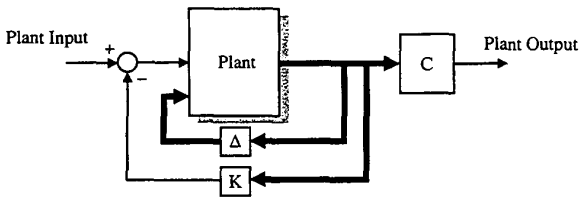

Figure 4: The plant state space representation, including stabilization feedback.

In Fig. 4, the plant is represented in state-space form. The plant is a single input, single output system. The thin lines carry scalar signals, and the heavy lines carry vector signals. The box C is a linear combiner with fixed weights that converts the plant state variables into the plant output. The box K is another linear combiner with fixed weights that converts the state variables into a scalar stabilizing signal. For the truck backer example, the state variables are θ_2 and θ_3. The plant output variable to be controlled is simply θ_3. The plant input is the steering angle θ_1 of the front wheels of the truck.

The input command to the Neurointerface controls the trajectory of the truck and trailers. A constant input causes backing along a circle of fixed radius. A sudden step change of the input command causes backing along a circle of a different fixed radius, after a transient takes place and dies out. A zero command input causes backing along a straight line, after transients die out.

Steering the truck system through the Neurointerface is a lot like steering a conventional automobile while driving forward. The instantaneous angle of the steering wheel determines the radius of curvature of the circle that the car follows. Changing the car's steering angle causes an instantaneous change in curvature of the trajectory, but without transients.

For the truck backer, controlling angle θ_3, the angle between the two trailers, would be sufficient to control the trajectory. If the angle θ_1 of the truck front wheels is controlled to achieve and maintain the correct fixed value of θ_3, the desired motion along a circle of fixed radius would occur, after transients die out. Thus, the truck backer is a Single-Input Single-Output (SISO) system. This would be true even if there were more than two trailers.

4 Training a Neurointerface

A block diagram illustrating the training of the Neurointerface is shown in Fig. 5. The Neurointerface is adapted so that the cascade of it and an exact model of the plant would have the same response as a chosen reference model. The Neurointerface would develop into an inverse of the plant if the reference model were a unit gain. If there is a response delay in the plant, the reference model would need the same delay or more. The reference model could be a linear system having a simple two-pole response. A reference model with a double pole has been used with the truck backer, giving exponential transients with step changes in the input command signals.

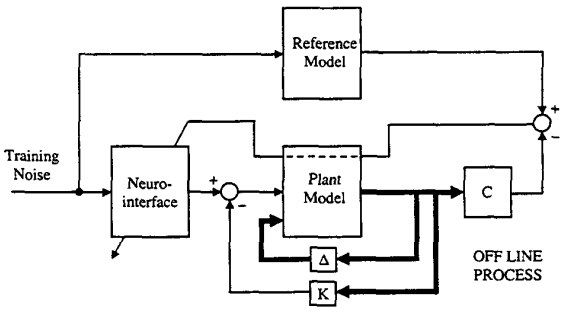

Figure 5: Off-line learning process for training the Neurointerface.

Training the Neurointerface is done off line. A noise input to the Neurointerface is used in the training process. This noise signal is also used to drive the input of

the reference model. The output of the reference model is compared with the plant output, and the difference is an error signal that is to be minimized by adjusting the weights of the NN in the Neurointerface. The structure of the Neurointerface is shown in Fig. 2.

In order to adapt the weights of the Neurointerface, an error signal at the Neurointerface output is needed. What is available however is the error signal at the output of the plant model. In order to get the appropriate error signal for adapting the Neurointerface, it is necessary to "backpropagate" the available error signal through the known equations of the plant model. The basic ideas are explained in the Prentice-Hall book "Adaptive Inverse Control" by Widrow and Walach [3], pages 480–484. The specific details of how this is done are given next.

The SISO nonlinear plant of Fig. 5 which is to be controlled by the Neurointerface is described by the following discrete-time state space equations

$$\begin{aligned} x_k &= f(x_{k-1}, u_k - K^T x_{k-1}), \\ y_k &= C^T x_k. \end{aligned} \tag{2}$$

Vector $x_k \in \mathbb{R}^{n_x}$ represents the state variables, $u_k \in \mathbb{R}$ is the plant input, and y_k is the plant output. Function $f : \mathbb{R}^{n_x} \times \mathbb{R} \longrightarrow \mathbb{R}^{n_x}$ is assumed to be analytic and $f(0,0) = 0$. The plant is considered to be Lagrangian stable (bounded states). If this is not the case, it is assumed that the feedback gain $K \in \mathbb{R}^{n_K}$ makes the plant Lagrangian stable in an open bounded region containing the origins of the state space and plant input.

The Neurointerface is described by the equation

$$u_k = g(R_k, w), \tag{3}$$

$$\text{where } R_k \triangleq \begin{bmatrix} r_k & r_{k-1} & \cdots & r_{k-n_R+1} \end{bmatrix}^T \in \mathbb{R}^{n_R}.$$

Signal $r_k \in \mathbb{R}$ is the Neurointerface command input, and signal $u_k \in \mathbb{R}$, the Neurointerface output. Vector $w \in \mathbb{R}^{n_w}$ represents the weights of the feedforward NN. The components of the vector R_k represent the signals generated by the Neurointerface's tapped delay line. They are connected to the feedforward NN inputs as shown in Fig. 2.

During the training phase, the Neurointerface output, u_k, is connected directly to the plant model input (also denoted by u_k), and the goal is to adapt the weight vector w step-by-step so the mean-square error,

$$\widehat{J} = \frac{1}{\mathcal{K}} \sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} e_k^2, \qquad e_k \triangleq d_k - y_k, \tag{4}$$

defined in a time window of $\mathcal{K}$ samples, is reduced. The signal d_k is the reference model output, and is the desired signal that the plant output y_k is suppose to follow at each k. The following constrained optimization problem reflects this idea:

$$\begin{gathered} \text{minimize } \widehat{J} \\ \text{subject to equations 2 and 3} \\ \text{for } k = \mathcal{T}+1, \ldots, \mathcal{K}+\mathcal{T}, \text{ and } x_{\mathcal{T}} \text{ specified.} \end{gathered} \tag{5}$$

Using Lagrangian techniques [4], equation 5 can be represented as an unconstrained optimization problem in the form,

$$\begin{aligned} J = \frac{1}{\mathcal{K}} \sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} e_k^2 + \sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} \beta_k \left(u_k - g(R_k, w)\right) + \\ \sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} \lambda_k^T \left(x_k - f(x_{k-1}, u_k - K^T x_{k-1})\right) + \\ + \sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} \delta_k \left(y_k - C^T x_k\right) \end{aligned} \tag{6}$$

and the objective is to calculate the gradient $\frac{\partial J}{\partial w}$ so w can be adjusted using a small step $\triangle w$ in the direction of $-\frac{\partial J}{\partial w}$. This will reduce the value of the mean-square error defined in equation 4. The optimization variables are now the Lagrangian multipliers $\beta_k, \delta_k \in \mathbb{R}$ and $\lambda_k \in \mathbb{R}^{n_x}$, the state variables x_k, the plant input u_k, the plant output y_k, and the weight vector w.

The gradient $\frac{\partial J}{\partial w}$ is given by

$$\frac{\partial J}{\partial w} = -\sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} \beta_k \frac{\partial g(R_k, w)}{\partial w}. \tag{7}$$

In order to compute it, one must calculate the values of β_k, $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$. They are obtained by applying the optimality conditions,

$$\frac{\partial J}{\partial \beta_k} = \frac{\partial J}{\partial \delta_k} = \frac{\partial J}{\partial \lambda_k} = \frac{\partial J}{\partial x_k} = \frac{\partial J}{\partial u_k} = \frac{\partial J}{\partial y_k} = 0, \tag{8}$$

to equation 6. As a result, the plant model equations need to be computed for $\mathcal{K}$ samples of the time window. They are:

$$\begin{aligned} u_k &= g(R_k, w), \\ x_k &= f(x_{k-1}, u_k - K^T x_{k-1}), \\ y_k &= C^T x_k \\ k &= \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}, \text{ and } x_{\mathcal{T}} \text{ specified.} \end{aligned} \tag{9}$$

Likewise, the Lagrangian variables are also computed in the same time window. First, δ_k is computed using the error signal e_k and the following equation:

$$\delta_k = \frac{2}{\mathcal{K}} e_k, \quad k = \mathcal{T}+1, \ldots, \mathcal{K}+\mathcal{T}. \tag{10}$$

Second, λ_k is computed through a recursive equation running backwards in time:

$$\lambda_k = \left(\frac{\partial f(x_k, u_{k+1} - K^T x_k)}{\partial x_k} \right)^T \lambda_{k+1} + \delta_k C, \tag{11}$$

for $k = \mathcal{K}+\mathcal{T}-1, \ldots, \mathcal{T}+1$ and $\lambda_{\mathcal{K}+\mathcal{T}} = \delta_{\mathcal{K}+\mathcal{T}} C$

Finally, the values of β_k, $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$ are computed through the following equation:

$$\beta_k = \lambda_k^T \left(\frac{\partial f(x_{k-1}, u_k - K^T x_{k-1})}{\partial u_k} \right), \tag{12}$$

With these values, it is possible to compute the gradient $\frac{\partial J}{\partial w}$ using equation 7. The Lagrangian multiplier β_k is the "error" signals referred to the output of the Neurointerface, needed to adapt it.

The following algorithm summarizes the steps necessary to compute the gradient $\frac{\partial J}{\partial w}$.

Algorithm 1: Given R_k and d_k for $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$; given $x_{\mathcal{T}}$ and w;

1. for $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$, compute:

$$u_k = g(R_k, w)$$
$$x_k = f(x_{k-1}, u_k - K^T x_{k-1})$$
$$y_k = C^T x_k$$
$$\delta_k = \frac{2}{\mathcal{K}} e_k$$

2. for $k = \mathcal{K}+\mathcal{T}-1, \ldots, \mathcal{T}+1$ and $\lambda_{\mathcal{K}+\mathcal{T}} = \delta_{\mathcal{K}+\mathcal{T}} C$, compute:

$$\lambda_k = \left(\frac{\partial f(x_k, u_{k+1} - K^T x_k)}{\partial x_k} \right)^T \lambda_{k+1} + \delta_k C$$

3. for $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$, compute:

$$\beta_k = \lambda_k^T \left(\frac{\partial f(x_{k-1}, u_k - K^T x_{k-1})}{\partial u_k} \right)$$

4. Compute the gradient $\frac{\partial J}{\partial w}$:

$$\frac{\partial J}{\partial w} = -\sum_{k=\mathcal{T}+1}^{\mathcal{T}+\mathcal{K}} \beta_k \frac{\partial g(R_k, w)}{\partial w}$$

The gradient $\frac{\partial J}{\partial w}$ is a moving average of the $\mathcal{K}$ samples in the window. With its value, the weight vector w can be updated using the equation:

$$\triangle w = -\alpha \frac{\partial J}{\partial w}^T, \quad \alpha > 0, \tag{13}$$

where α is a small positive number.

Once the Neurointerface is trained, it can be used to control the plant. Referring to Fig. 6, the human command input to the Neurointerface causes the plant output to respond as if the cascade of Neurointerface and plant were equivalent to the reference model.

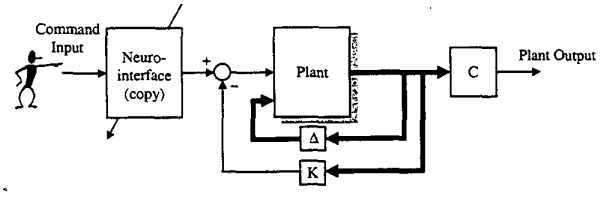

Figure 6: The trained Neurointerface connected to the plant.

5 Plant Disturbance

An important subject is that of plant disturbance. The configuration of Neurointerface and plant of Fig. 1 does not show this. In fact, if plant disturbance were present, it would be apparent to the human operator who in some cases might be able to modify the command input in order to counteract the disturbance. Generally, this would not be easy for the operator to do because of the effects of inherent delay in the plant dynamic response. Some other means for dealing with plant disturbance without requiring action on the part of the human operator would be desirable.

A method for cancelling plant disturbance without affecting the plant dynamics is taught in Chapter 8 of the 1998 Prentice-Hall book by Widrow and Walach [3]. The method can be applied to Neurointerface control. The idea is illustrated by the block diagram of Fig. 7. The following is a brief explanation. A full description is given in the reference.

Refer now to Fig. 7. It can be seen that once again a copy of the Neurointerface is used to drive the plant. This diagram is more complicated than that of Fig. 6 however because it includes an adaptive disturbance canceller.

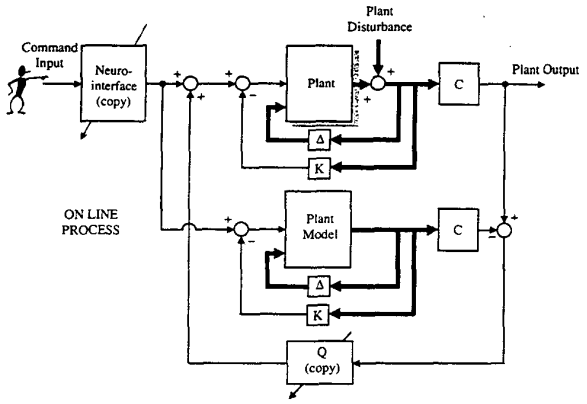

Figure 7: A Neurointerface connected to a plant with a disturbance canceller.

In Fig. 7, both the plant and an exact model of the plant are driven by the Neurointerface output. The output of the plant model, which is disturbance free, is subtracted from the plant output. The difference is pure plant disturbance, referred to the plant output. The plant disturbance is fed to the box labeled Q (copy). This box is a SISO nonlinear adaptive filter that has been trained by a off line process (see Fig. 8) to be a best least squares inverse of the plant. The output of Q is subtracted from the plant input, but not subtracted from the plant model input. It is shown in the Widrow, Walach reference that if the plant is linear, this feedback noise canceller is optimal, and that it reduces the plant disturbance observed at the plant output to the lowest level physically possible in the least squares sense. This optimality has not been proven yet for nonlinear systems, but simulation experiments have shown the adaptive canceller to be highly effective. In any event, because the driven response of the plant and the plant model are identical, subtracting their outputs to obtain the disturbance signal to drive Q and to obtain feedback results in a feedback loop with zero gain around it. Thus, the disturbance canceller does not affect the dynamic response of the plant, whether the plant is linear or nonlinear. The training of the box Q, shown in Fig. 8, uses training noise to effect a learning process that makes the cascade of Q and the plant model behave as best possible in the least square sense like a piece of wire, i.e. a unit gain. The training process is identical to that used in Fig 5 to train the Neurointerface. Both the filter Q and the Neurointerface are configured like the nonlinear adaptive filter shown in Fig. 2.

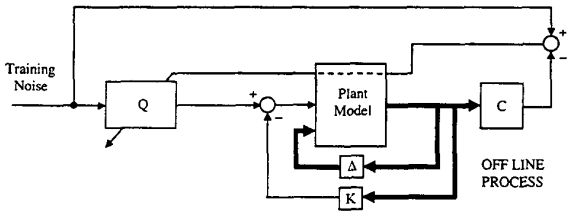

Figure 8: Training the filter Q for use in the plant disturbance canceller.

If the filter Q is an exact inverse of the plant, then the plant disturbance will be perfectly cancelled. This will never happen perfectly however, because there must always be at least one sample time of delay around the loop. Also, any delay in the plant will prevent Q from being a perfect inverse of the plant. In the linear case, if the plant is nonminimum phase, Q can not be a perfect inverse, but the adaptive disturbance canceller is nevertheless optimal. In the nonlinear case, optimality is plausible but yet unproven.

6 Conclusion

A Neurointerface is an inverse or model reference inverse of a nonlinear plant to be controlled. It serves as a trainable interface between the plant and a human operator. In a strict sense, nonlinear plants do not have inverses. Yet, an adaptive filter based on Neural Networks can be trained to be a best least squares inverse of a nonlinear plant. Once converged, the inverse or Neurointerface makes a good approximate inverse, even for input signals other than the training signals. Use of a Neurointerface makes it easy for the human operator to control or direct a nonlinear plant.

References

[1] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences.* PhD thesis, Harvard University, Boston, MA, 1974.

[2] D. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Control Systems Magazine*, pp. 18–23, 1990.

[3] B. Widrow and E. Walach, *Adaptive Inverse Control.* Prentice-Hall , Inc., Upper Saddle River, NJ., 1996.

[4] A. E. Bryson, *Dynamic Optimization.* Addison Wesley Longman, 1999.