Neurointerfaces

Bernard Widrow, *Life Fellow, IEEE*, and Marcelo Malini Lamego, *Member, IEEE*

***Abstract*—A neurointerface is a nonlinear filtering system based on neural networks (NNs) 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 and the weights of an adaptive plant disturbance canceller. Application of these ideas to backing a truck with two trailers under human direction is described. The "truck backer" has been successfully demonstrated by computer simulation and by physical implementation with a small radio-controlled truck and trailers.**

***Index Terms*—Adaptive control, man–machine interfaces, neural networks (NNs).**

I. 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) [1]–[4], 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 intends to show 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. Neurointerfaces can be regarded as circuitries, algorithms, and devices implementing NNs to facilitate the human operation of complex systems.

The design of neurointerfaces involves the training of NNs, which are incorporated in nonlinear adaptive filters. This work applies the inverse modeling technique to designing neurointerfaces. In the past, many works have described training procedures and design techniques for inverse modeling using NNs (see [5], [6], and references therein). This study will apply the inverse modeling technique to the design of man–machine interfaces for complex dynamic systems.

Manuscript received August 29, 2000; revised July 14, 2001. Manuscript received in final form August 24, 2001. Recommended by Associate Editor D. W. Repperger.

B. Widrow is with the Electrical Engineering Department, Stanford University, Stanford CA 94305 USA (e-mail: widrow@stanford.edu).

M. M. Lamego is with The Boston Consulting Group Brazil, Ltd., CEP 01451–0000 Sao Paulo-SP, Brazil (e-mail: mmlamego@ieee.org).

Publisher Item Identifier S 1063-6536(02)00335-4.

In the work of Narendra and Parthasarathy [7], NNs were proposed for the identification and control of nonlinear dynamic systems. As a result, a considerable number of papers have appeared on the general subject of control with NNs. Training schemes based on the backpropagation algorithm [8] and its variations were proposed and applied to NN control.

This work applies dynamic optimization [9], [10] to the training of a multilayer NN with a tapped delay line (neurointerface) cascaded with the plant model. The neurointerface training is formulated as a constrained optimization problem and is solved through dynamic optimization. The technique assumes the nonlinear plant is continuous, minimum phase, and controllable with bounded states (Lagrangian stability). If the plant is unstable, it must first be stabilized by feedback. The backpropagation algorithm is used in one of the steps of the neurointerface training method. For the interested reader, the application of dynamic optimization to state-feedback control with NNs can be found in the works of Lamego [11], Shen [12], and Plumer [13].

The neurointerface training is done off-line. This is mainly because the dynamic optimization algorithm is computationally very expensive and it also needs the plant model. In addition, depending on the system (plant model plus neurointerface) under study, the time of convergence may be too lengthy for real-time adaptive schemes. Nonetheless, the neurointerface design can be often extended to integrated off-line schemes, wherein a system identification algorithm obtains a continuous representation of the plant model (probably, using an NN) from on-line acquired data. Then, the dynamic optimization algorithm uses the plant model for training the neurointerface, which in turn controls the plant. The processes of plant identification and neurointerface training can be repeated periodically using real-time acquired data.

In order to keep the neurointerface training description simple, this paper focus solely on pure off-line training. Once the basic concepts related to plant inversion and dynamic optimization applied to NNs are understood, extensions to real-time learning schemes can be readily made.

A neurointerface is used to facilitate 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.

II. What is a Neurointerface?

A neurointerface may be thought of as a form of inverse of the plant to be controlled. The 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

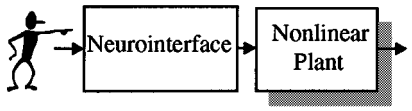


Fig. 1. A cascade of the trained neurointerface and the plant.

operator. Thus, an unskilled operator using a neurointerface can reproduce the actions 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 a truck and multiple 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.

It should be noted that the driver is not eliminated in this work. Nguyen and Widrow [14] proposed an NN that provided full automation in backing a trailer truck to a loading dock and indeed, eliminating the presence of the driver. In the present work, the human action is essential. In fact, the driver is concerned with providing the desired spatial trajectory, free of obstacles and normally the shortest one.

The truck-backing-up exercise is a kinematic inverse modeling problem. Kinematic, in this sense, 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 human-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 about achieving movement free of oscillations when shifting the load from one point to another. 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.

The neurointerface is designed to operate in real time. The training procedure is 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 relationship 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.

The basic topology of a neurointerface is shown in Fig. 2. It is a feedforward nonlinear adaptive transversal filter consisting of a tapped delay line connected to a multilayer NN. The weights $w_0, w_1, \ldots, w_n$ of each neuron are adjusted automatically during the training process. The filter input is r_k, the filter output is u_k, the discrete-time index is k, and a unit delay is represented by the symbol $\triangle$.

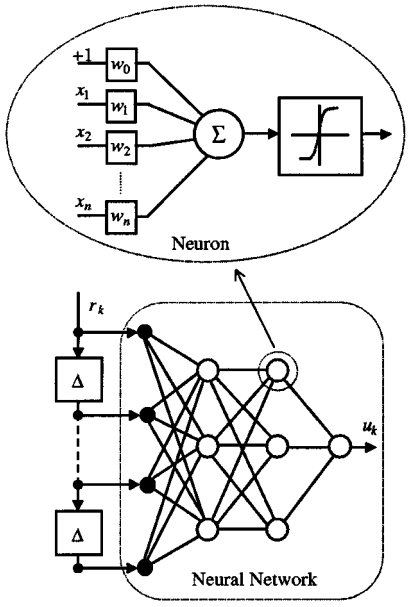


Fig. 2. A feedforward nonlinear adaptive filter incorporating a three-layer NN.

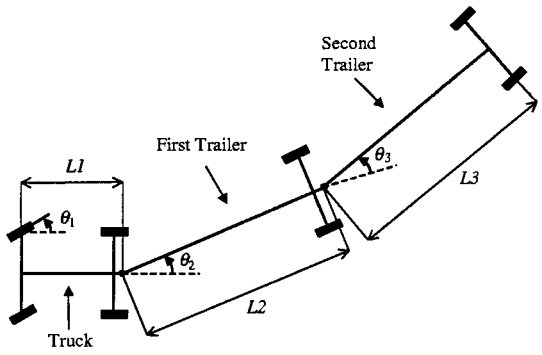


Fig. 3. Schematic diagram of a truck and two trailers.

III. The Truck Backer: A Case Study

The kinematic equations for the motion of the truck and double trailers are easily derived from geometric considerations [11]. Regarding the schematic diagram of the truck and trailers shown in Fig. 3, these equations are

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

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. It is assumed in this work that the plant to be controlled is known. Such is the case with the truck backer.

The first step is to stabilize the unstable plant about an equilibrium point. This can be done in many cases by making use of negative feedback with fixed gains. The idea is illustrated in Fig. 4.

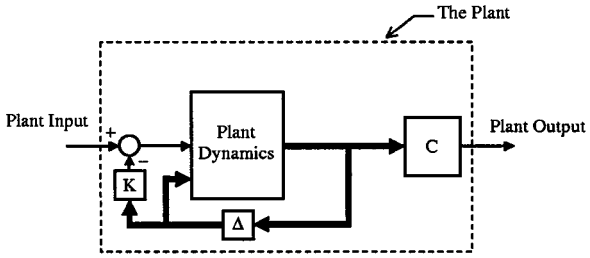


Fig. 4. The plant state-space representation, including stabilization feedback.

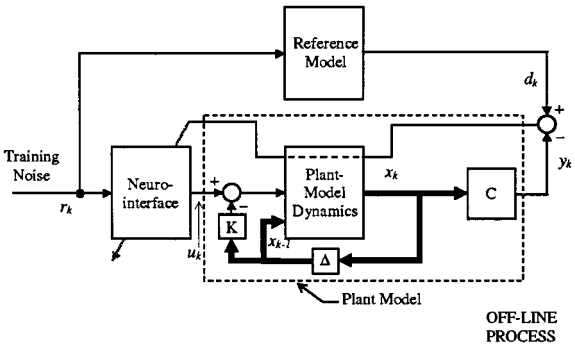


Fig. 5. Off-line learning process for training the neurointerface.

In Fig. 4, the plant is represented in state-space form. The plant is a single-input–single-output (SISO) 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 input is the steering angle θ_1 of the truck's front wheels. The Plant output variable to be controlled is the state variable θ_3. For example, $\theta_3 = 0$, the truck will be backing straight back.

The gain values of the linear combiner K are calculated through some state-feedback technique (plant linearization around the equilibrium [15], Lyapunov energy functions [16], etc.). In the truck-backer example, the gain values were obtained by simple plant model linearization around the equilibrium ($\theta_2 = \theta_3 = 0$). This approach has shown to be very effective and it has been verified through computer simulation and physical implementation that the Lagrangian stability is achieved in the operating region in which the truck and multiple trailers are supposed to work. Alternatively, the linear combiner may be replaced with a nonlinear feedback topology obtained through state-feedback linearization [17]. The advantage of such a technique is that the domains of attraction around the equilibrium point are usually increased, providing better and smoother transient responses.

The input command to the neurointerface controls the trajectory of the truck and trailers. A constant input command causes the truck and trailers in steady state to back along a circle of fixed radius. A sudden step change of the input command causes the truck and trailers to back along a circle of a different fixed radius after a transient takes place and dies out. A zero command input causes the truck and trailers to back along a straight line after the transient dies 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 an SISO system. This would be true even if there were more than two trailers.

IV. 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. Assuming that the plant is continuous, minimum phase, and controllable[1] with bounded states (Lagrangian stability), ideally the neurointerface would develop into an inverse of the plant if the reference model were a unit gain, and there were no limits on the absolute maximum values of the neurointerface output signals applied to the plant input. However, the neurointerface can only generate bounded output signals (continuous or not), and the corresponding plant output signals are always continuous and with certain time-response features. Thus, the cascade of the trained neurointerface and plant can not behave like a unit gain block. For this reason, in Fig. 5, the reference model block is regarded as a general dynamic system (to be defined by the designer) incorporating the essential (desired) time-response features of the trained neurointerface plus plant. An example of an essential time-response feature would be a time delay in the plant response. Because the neurointerface control system is causal, it cannot eliminate plant delays, and indeed, the reference model would incorporate the same delay or more in its time-response specifications. In Fig. 5, the reference model block is usually regarded as a linear dynamic system. With the truck backer, for instance, a reference model with a double pole has been used, giving exponential transients in response to step changes in the reference model input. The idea of reference models was introduced by Draper and Li [18] and has been widely used since then.

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. All that 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

[1]Here, the plant is assumed to be controllable in the operating region in which it is supposed to work.

plant model. The specific details of how this is done and how the neurointerface is trained using dynamic optimization [10] 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:

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

The variable k is the time index, $\mathcal{T}$ is the initial time, and $\mathcal{K}+\mathcal{T}$, the final time. Vector $x_k \in \mathbb{R}^{n_x}$ represents the state variables, and $x_{\mathcal{T}}$ (initial values for the state variables) is assumed to be known. $u_k \in \mathbb{R}$ is the plant input, and $y_k \in \mathbb{R}$ 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 controllable and Lagrangian stable [15], [17]. If not so, it is assumed that the feedback gain $K \in \mathbb{R}^{n_x}$ makes the plant Lagrangian stable (bounded states) in an open bounded region containing the origins of the state space and plant input. In this case, all the state variables used for stabilization must be accessible.

The neurointerface is described by

$$u_k = g(R_k, w)$$

$$\text{where } R_k \triangleq [r_k \quad r_{k-1} \quad \cdots \quad r_{k-n_R+1}]^T \in \mathbb{R}^{n_R}. \tag{3}$$

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.

Refer to Fig. 5. 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

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

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:

$$\begin{aligned} &\text{minimize } \hat{J} \\ &\text{subject to (2) and (3)} \\ &\text{for } k = \mathcal{T}+1, \ldots, \mathcal{K}+\mathcal{T}, \text{ and } x_{\mathcal{T}} \text{ specified.} \end{aligned} \tag{5}$$

Using Lagrangian multipliers, (5) can be represented as an unconstrained optimization problem in the form

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

and the objective is to calculate the gradient $\partial J/\partial w$ so w can be adjusted using a small step dw in the direction of $-(\partial J/\partial w)$. This will reduce the value of the mean-square error defined in (4). The optimization variables are now the Lagrangian multipliers β_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 $\partial J/\partial w$ 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 $\partial g(R_k, w)/\partial w$ and β_k, for $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$. Since $\partial g(R_k, w)/\partial w$ is the partial derivative of the feedforward NN output with respect to the weight vector w, it can be computed for each k using the backpropagation algorithm [8].

The computation of β_k however follows a different approach based on dynamic optimization [10]. The values of β_k, for $k = \mathcal{T}+1, \ldots, \mathcal{T}+\mathcal{K}$, 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 (6). As a result, the plant model equations need to be computed for $\mathcal{K}$ samples of the time window. They are

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

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:

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

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

$$\begin{aligned} &\lambda_k = \left(\frac{\partial f\left(x_k, u_{k+1} - K^T x_k\right)}{\partial x_k}\right)^T \lambda_{k+1} + \delta_k C, \\ &\text{for } k = \mathcal{K}+\mathcal{T}-1, \ldots, \mathcal{T}+1 \text{ and } \lambda_{\mathcal{K}+\mathcal{T}} = \delta_{\mathcal{K}+\mathcal{T}} C. \end{aligned} \tag{11}$$

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

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

With these values, it is possible to compute the gradient $\partial J/\partial w$ using (7). The Lagrangian multiplier β_k is the "error" signal referred to the output of the neurointerface, needed to adapt it.

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

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

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

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

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

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 $\partial J / \partial w$

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

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

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

where α is a small positive number.

The algorithm converges to local minima. A local minimum may be eventually a global one. However, there is no guarantee this will be the case since by forcing optimality in (6) with respect to the Lagrangian multipliers does not ensure J will be a contraction mapping in w. Therefore, depending on the initial value of w, its iterative sequence generated by (13) may or may not reach a local minimum with an acceptable value for J (a small one). It should be noted, however, that the algorithm always converges to a local minimum provided the nonlinear plant is continuous, minimum phase, and controllable with bounded states, and that the parameter α in equation (13) is small enough. As a result, the designer is supposed to evaluate the value of J and check if its local-optimum value is small enough. If not so, he or she should repeat the algorithm, using a different initial value for w, the NN weight vector.

Once the neurointerface is trained, it can be used to control the plant. Fig. 6 shows a trained neurointerface connected to the plant. In the neurointerface block, the term "copy" is used to emphasize that the neurointerface is trained off-line using the plant model, and afterwards, a digital copy of the trained neurointerface is used to drive the real plant. The human command input to the neurointerface causes the plant's output to respond as if the cascade of neurointerface and plant were equivalent to the reference model.

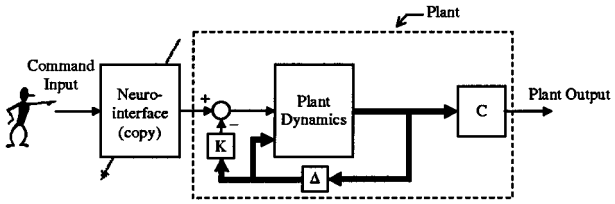


Fig. 6. The trained neurointerface connected to the plant.

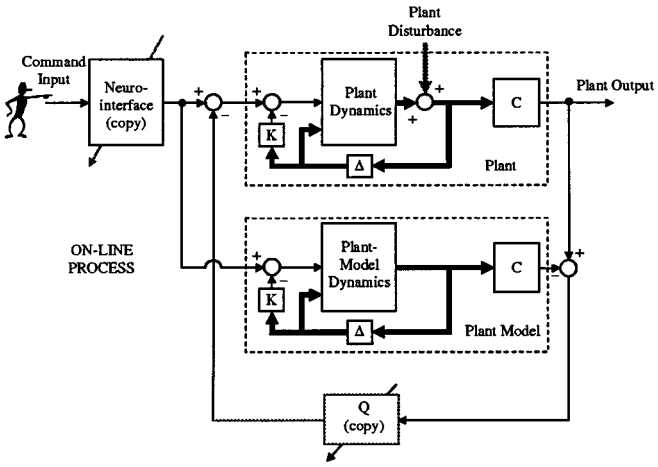


Fig. 7. A neurointerface connected to a plant with a disturbance canceller.

V. 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 disturbances 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 [6, Ch. 8]. The method can be applied to neurointerface control. The idea is illustrated by the block diagram of 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.

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 an SISO nonlinear adaptive filter that has been trained by an 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 [6] 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

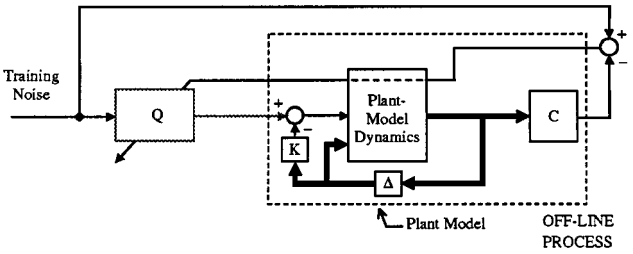


Fig. 8. Training the filter Q for use in the plant disturbance canceller.

been proven yet for nonlinear systems, but simulation experiments have shown the adaptive canceller to be highly effective. 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 affect a learning process that makes the cascade of Q and the plant model behave as the best possible in the least squares 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.

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

VI. Experimental Results With the Truck Backer

Application of neurointerface control has been made to the truck backer-upper. this was done two ways: by computer simulation and with a physical toy truck and trailer that is approximately 1.5 m long. Backing the truck and trailers is not a "toy" problem, however. The authors have had many discussions with professional truck drivers, and have discovered that most professional drivers would have a difficult time backing up a tandem—a truck and two trailers. In normal use, the trailers are uncoupled before backing one at a time.

The trailer-truck system is a continuous plant as displayed in (1). During the neurointerface training, the plant is regarded as a discrete-time system. Thus, to apply the off-line training procedure described in Section IV [Algorithm 1 and (13)] to both neurointerface and disturbance canceller circuits, it is necessary to discretize the continuous plant. Two discretization methods were used. In step 1 (one) of Algorithm 1, the plant simulation was accomplished using a fourth-order Runge–Kutta method, so that the simulation results could reflect as closely as possible the dynamical behavior of the trailer-truck continuous model. In steps 2 (two) and 3 (three), the partial derivatives $\partial f/\partial x_k$ and $\partial f/\partial u_k$ were computed from a discrete version of the continuous plant obtained from Euler's method. This was done to make the computation of the partial derivatives simpler. This introduced some small numerical errors during the computation of the Lagrangian multipliers, β_k and λ_k, which in turn might have slightly changed the final solution for the weight vector w.

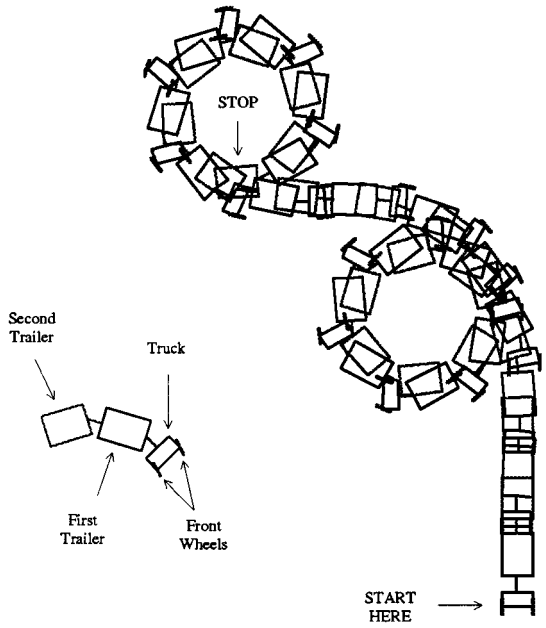
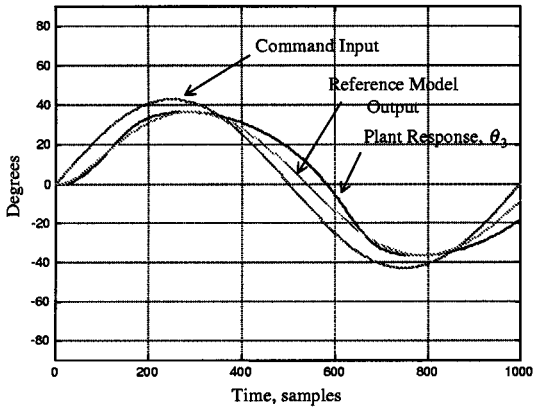
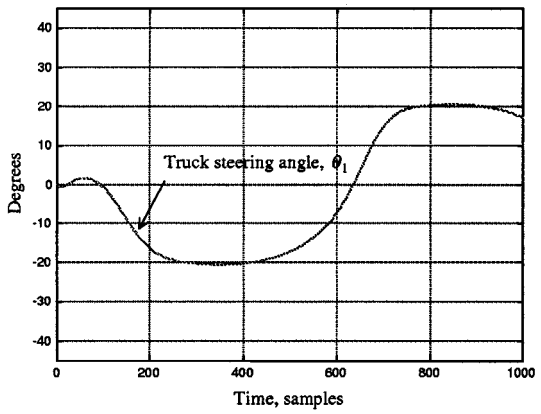


Fig. 9. Trajectory of truck and trailers.

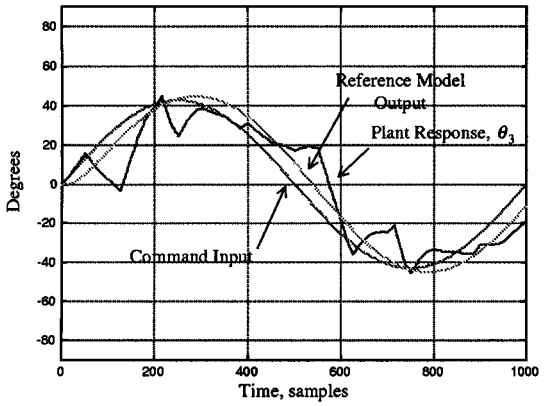


(a)

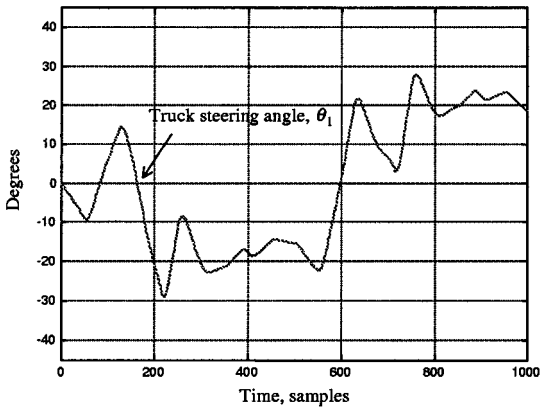


(b)

Fig. 10. Time plots of truck backer. (a) Command input, reference model output, and angle θ_3. (b) Steering angle θ_1.



(a)



(b)

Fig. 11. Time plots of truck backer with disturbance. (a) Command input, reference model output, and angle θ_3. (b) Steering angle θ_1.

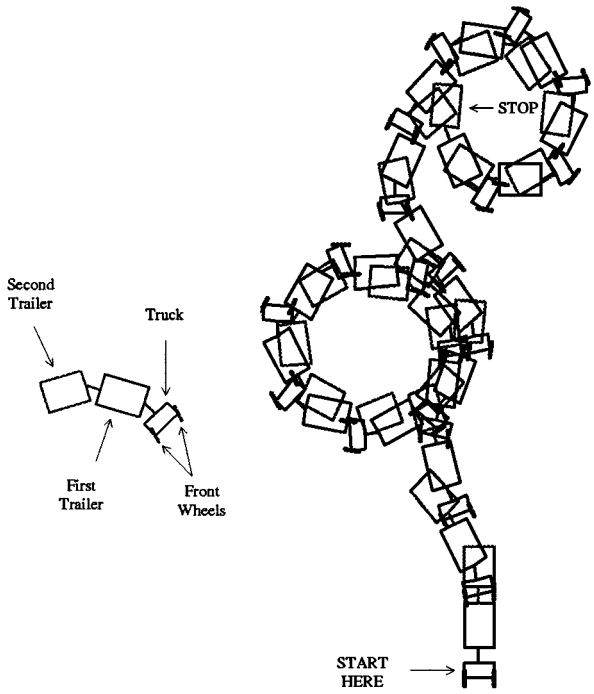


Fig. 12. Trajectory of truck and trailers with disturbance.

Fig. 13. Truck with double trailer.

However, once the training algorithm converged (to a local optimum), the discrete plant simulation response in step 1 closely matched that of the continuous plant model.

The results of a typical simulation experiment are shown in Figs. 9 and 10. In Fig. 9, the backing trajectory of the truck and trailers is shown. This trajectory results from application of a sinusoidal command input, plotted in Fig. 10(a). The command input exercises control over the plant variable θ_3. This is the plant output, and it is also plotted in Fig. 10(a). The motion of θ_3 versus time should match the response of the reference model driven by the command input. This response has been computed, and it is also plotted in Fig. 10(a). These plots are quite similar. This indicates that the trained neurointerface, when cascaded with the nonlinear plant model of Fig. 5, has a response that fairly closely matches that of the linear reference model.

The truck steering angle is plotted versus time in Fig. 10(b). This strange steering function causes the backing trajectory of Fig. 9 and the angle θ_3 response plotted in Fig. 10(a). The θ_1 variable to produce the trajectory in Fig. 9 is somewhat counterintuitive to the operator.

In this simulation, the total length of the truck and trailers was 1.5 m, and the truck was backing at a constant speed of 1 m/s. The sampling rate for the simulation was 50 samples/s. For the off-line computations to obtain the neurointerface and the filter Q, the moving average window in each case contained $\mathcal{K} = 10$ samples. The training algorithm converged after 935 iterations, corresponding to a final value for J smaller than 1% of its initial value.

In the above described experiment, there was no plant disturbance. The same experiment was repeated with a fairly violent plant disturbance to test the disturbance canceller of Fig. 7. The command input, the plant response, and the computed response of the reference model are plotted in Fig. 11(a). The truck steering angle is plotted versus time in Fig. 11(b). The jitter in steering angle which was needed to compensate for the plant disturbance is very evident. In spite of the disturbance, the system remains stable and does not jackknife. The backing trajectory is shown in Fig. 12. The disturbance has caused the truck to take a different course. A human controlling the truck could have kept the truck on course, if he wished, without needing to worry about jackknifing.

Experiments with the toy truck under human steering command have been done. Instead of a computer generated sinusoidal command input, manual steering commands have been inputted to the neurointerface by means of a small steering wheel connected to a radio transmitter. The received command input was fed to a neurointerface implemented by an Intel 486 battery-operated computer. The QNX real-time operating system [19] was used by the computer. All programming was done in the C language. The output of the neurointerface drove a servo that controlled the steering angle θ_1 of the truck.

A photograph of the truck and double trailer is shown in Fig. 13. At the time of this writing, the truck has been seen dashing down the halls of Stanford University. The next phase

of the research has a goal of including an obstacle avoidance capability in the neurointerface.

VII. Conclusion

Man–machine interfaces have been implemented in the past to facilitate manual control of plants that are difficult or impossible for human operators to control. Typically, plants that are difficult to control are unstable and nonlinear. Interfaces for such systems generally provide stabilization feedback. Examples are unstable aircraft, for instance helicopters [20], that are only flyable because of stabilization provided by their autopilots.

In addition to stabilization, this paper proposes the use of a series filter between the human operator and the stabilized plant to further ease the task of human control. This filter is designed to be an approximate inverse of the stabilized plant.

This paper proposes that the nonlinear series filter be implemented in the form of a transversal filter consisting of a tapped delay line with the signals at the taps providing inputs to a multilayered neural network. The filter output is the neural network output. The filter is called a neurointerface. It can be trained by a learning algorithm which is a form of dynamic optimization wherein the derivatives of the neurointerface output with respect to the weights are calculated by backpropagation.

The idea of an inverse comes from linear system theory. The inverse has a transfer function that is the reciprocal of the transfer function of the plant. Nonlinear plants do not have transfer functions. Nevertheless, approximate inverses of nonlinear plants can be made using adaptive algorithms that are similar to those used for making inverses of linear plants. The existence of approximate inverses of nonlinear plants is at present based more on experiment and experience rather than on analytical proof.

Plant disturbance makes human control more difficult, and reducing disturbance to the lowest possible level is certainly desirable. This paper proposes a method that feeds the disturbance signal back to the plant input in such a way that the plant dynamics are unaffected. A related method is shown by Widrow and Walach [6] to be optimal for cancellation of disturbance in linear plants. Optimality for nonlinear plants is plausible but not yet proven.

The method proposed in this paper has been applied to the problem of backing a trailer-truck with two trailers under human control. A backing trailer-truck is an unstable nonlinear plant. Without a neurointerface, backing causes immediate jackknifing. With a neurointerface, it is almost as easy for a person to back a truck with two trailers as it is to drive a truck forward. This has been tested and analyzed by computer simulation, and has been demonstrated with a scale-model truck and trailers steered by human control through a neurointerface. The NN was implemented by a battery-operated computer mounted in one of the trailers. Use of an adaptive disturbance canceller and use of a plant inverse as an interface have greatly facilitated the human control task.

Acknowledgment

The authors wish to thank all the referees for their valuable comments and suggestions, and Prof. E. P. Ferreira for helping to build the radio-controlled truck and trailers shown in Fig. 13.

References

[1] S. S. Haykin, *Neural Networks*. Upper Saddle River, NJ: Prentice-Hall, 1998.
[2] B. Widrow and M. Lehr, "Thirty years of adaptive neural networks: Perceptron, Madeline, and backpropagation," *Proc. IEEE*, pp. 1415–1442, 1990.
[3] M. A. Lehr, B. Widrow, and D. E. Rumelhart, "Neural networks: Applications in industry, business, and science," *Commun. Assoc. Comput. Machinery*, pp. 93–105, 1994.
[4] R. Rojas, *Neural Networks: A Systematic Introduction*. New York: Springer-Verlag, 1996.
[5] J. B. D. Cabrera and K. S. Narendra, "Issues in the application of neural networks for tracking based on inverse control," *IEEE Trans. Automat. Contr.*, pp. 2007–2027, 1999.
[6] B. Widrow and E. Walach, *Adaptive Inverse Control*. Upper Saddle River, NJ: Prentice-Hall, 1996.
[7] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, pp. 4–27, 1990.
[8] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Boston, MA, 1974.
[9] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. Washington, DC: Hemisphere, 1975.
[10] J. A. E. Bryson, *Dynamic Optimization*. Reading, MA: Addison-Wesley, 1999.
[11] M. M. Lamego, "Neurointerfaces," Ph.D. dissertation, Elect. Eng. Dept., Stanford Univ., Stanford, CA, Apr. 2000.
[12] R. T. Shen, "Optimal control of nonlinear systems with neural networks," Ph.D. dissertation, Elect. Eng. Dept., Stanford Univ., Stanford, CA, 1998.
[13] E. S. Plumer, "Optimal terminal control using feedforward neural networks," Ph.D. dissertation, Elect. Eng. Dept., Stanford Univ., Stanford, CA, 1993.
[14] D. Nguyen and B. Widrow, "Neural networks for self-learning control systems," *IEEE Contr. Syst. Mag.*, pp. 18–23, 1990.
[15] J. C. Willems, *Stability Theory of Dynamical Systems*. New York: Wiley, 1970.
[16] A. M. Lyapunov, *General Problem of the Stability of Motion*. London, U.K.: Taylor and Francis, 1992. English translation of the Russian original published in 1892.
[17] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. New York: Springer-Verlag, 1999.
[18] C. S. Draper and Y. T. Li, *Principles of Optimalizing Control Systems and an Application to the Internal Combustion Engine*. New York: ASME, 1951.
[19] QNX Software Systems Ltd. *QNX System Architecture*, Ontario, Canada, 1996.
[20] G. D. Padfield, *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*. Washington, DC: AIAA Education Series, 1996.

Bernard Widrow (M'58–SM'75–F'76–LF'95) received the Bachelor's, Master's, and Doctor of science degrees from the Massachusetts Institute of Technology (MIT), Cambridge.

He served on the MIT faculty before coming to Stanford in 1959. He has published two textbooks and numerous journal articles. He is a pioneer in the field of adaptive filters and neural networks and a long-time Professor of Electrical Engineering at Stanford who concentrates his research on adaptive signal processing, adaptive control systems, and adaptive neural networks.

He was inducted into the National Academy of Engineering in 1995.

Marcelo Malini Lamego (S'98–M'00) received the Bachelor's and Master of science degrees in electrical engineering from the Federal University of Espírito Santo, Brazil, and Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in April 2000.

His current research interests include advanced adaptive algorithms and dynamic models for business valuation and risk analysis in investment decisions.