Lecture Series on

Intelligent Control

Lecture 16

Generalized Backpropagation Through Time

Kwang Y. Lee
Professor of Electrical & Computer Engineering
Baylor University
Waco, TX 76706, USA
Kwang_Y_Lee@baylor.edu

1

An Optimal Tracking Neuro-Controller for Nonlinear Dynamic Systems

Park, Y.-M., M.-S. Choi, and K. Y. Lee, IEEE Transactions on Neural Networks, Vol.7, No. 5, September 1996, pp. 1099-1110.

2

2

Introduction

- Multilayer neural networks are used to design an optimal tracking neuro-controller (OTNC) for discrete-time nonlinear dynamic systems with quadratic cost function.
- The OTNC is made of two controllers: Feedforward Neuro-Controller (FFNC) and Feedback Neuro-Controller (FBNC).
- The FFNC controls the steady-state output of the plant, while the FBNC controls the transient-state output of the plant.
- The FFNC is designed using a novel inverse mapping concept by using a neuro-identifier.
- A Generalized Backpropagation-Through-Time (GBTT) algorithm is developed to minimize the general quadratic cost function for the FBNC training.
- The proposed methodology is useful as an off-line control method where the plant is first identified and then a controller is designed for it.
- A case study for a typical plant with nonlinear dynamics shows good performance of the proposed OTNC.

3

3

Optimal Tracking Problem

A. Optimal Tracking Problem

We consider a system in the form of the general nonlinear auto-regressive moving average (NARMA) model: (1)

$$y(k+1) = f(y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-m+1))$$

Where y and u, respectively, represent output and input variables, *k* represents time index, and *n* and *m* represent the respective output and input delay orders.

When the target output of a plant holds up for some time and varies from time to time, the control objectives can be defined as follows:

(1) Minimize the summation of the squares of regulating output error and the squares of input error in transient.

(2) Reduce the steady-state error to zero.

4

4

Optimal Tracking Problem

The above control objectives can be achieved by minimizing the following well-known quadratic cost function:

$$J = \frac{1}{2}\sum_{k=1}^{N}\left(Q(y_{ref} - y(k+1))^2 + R(u_{ref} - u(k))^2\right)$$

y_{ref} is a reference output,

u_{ref} is the steady-state input corresponding to y_{ref}

Q and *R* are positive weighting factors.

This quadratic cost function or the performance index, not only forces the plant output to follow the reference, but also forces the plant input to be close to the steady-state value in maintaining the plant output to its reference value.

5

5

Optimal Tracking Problem

A linear counter part to the NARMA model (1) is the following linear time-invariant system:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \end{aligned}$$

In steady-state, then the above state equation becomes

$$\begin{aligned} x_{ref} &= Ax_{ref} + Bu_{ref} \\ y_{ref} &= Cx_{ref}, \end{aligned}$$

By subtracting this from the system equation, and shifting the vectors as

$$\begin{aligned} u'(k) &= u(k) - u_{ref} \\ x'(k) &= x(k) - x_{ref} \\ y'(k) &= y(k) - y_{ref}, \end{aligned}$$

6

6

Optimal Tracking Problem

A linear counter part to the NARMA model (1) is the following linear time-invariant system:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k), \end{aligned}$$

In steady-state, then the above state equation becomes

$$\begin{aligned} x_{ref} &= Ax_{ref} + Bu_{ref} \\ y_{ref} &= Cx_{ref}, \end{aligned}$$

By subtracting this from the system equation, and shifting the vectors as

$$u'(k) = u(k) - u_{ref}$$
$$x'(k) = x(k) - x_{ref}$$
$$y'(k) = y(k) - y_{ref},$$

7

7

Optimal Tracking Problem

Then, the optimal tracking problem is converted to the optimal regulator problem with zero output in steady-state:

$$\begin{aligned} x'(k+1) &= Ax'(k) + Bu'(k) \\ y'(k) &= Cx'(k), \end{aligned}$$

with the quadratic cost function

$$J = \frac{1}{2}\sum_{k=1}^{N} (Q(y'_{(k+1)})^2 + R(u'_{(k+1)})^2)$$

The control law for the optimal regulator problem for $N = \infty$ is

$$u'(k) = Fx'(k)$$

where *F* is the optimal feedback gain matrix obtained.

Thus, the optimal control for the original linear system is

$$u(k) = Fx'(k) + u_{ref}$$

This shows an important observation that the control input consists of two parts, feedforward and feedback:

$$u(k) = u_{fb}(x'(k)) + u_{ff}(y_{ref})$$

8

8

Optimal Tracking Neuro-Controller

B. Architecture for Optimal Tracking Neuro-Controller

Following the above observation, an optimal tracking neuro-controller (OTNC) is designed with two neuro-controllers in order to control a nonlinear plant that has a non-zero set point in steady-state.

- A *feedforward* neuro-controller (FFNC) is constructed to generate feedforward control input corresponding to the set point and trained by the well-known error Backpropagation algorithm.
- A *feedback* neuro-controller (FBNC) is constructed to generate feedback control input and trained by a Generalized BTT (GBTT) algorithm to minimize the quadratic performance index.
- An independent neural network named *neuro-identifier* (NI) is used when the above two neuro-controllers are in training mode. This network is trained to emulate a plant dynamics and to backpropagate an *equivalent error* or *generalized delta* to the controllers under training.

9

9

Optimal Tracking Neuro-Controller

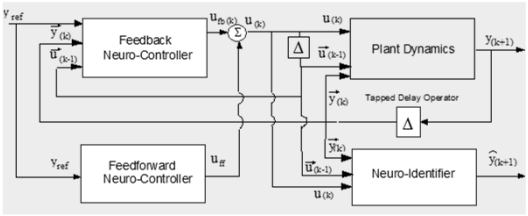


Fig. 1. Block diagram for the optimal tracking neuro-controller.

10

10

Neuro-Identifier

A. Neuro-Identifier

The function of the neuro-identifier is to identify plant dynamics. It is then used to backpropagate the equivalent error to the neuro-controllers. Training the neuro-identifier can be regarded as an approximation process of a nonlinear function using input-output data sets.

A NARMA model (1) can be viewed as a nonlinear mapping from (*n+m*)-dimensional input space to a one-dimensional output space:

$$y(k+1) = f(\vec{X}(k))$$

where $\vec{X}(k) = (y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-m+1))$ is regarded as input vector.

Therefore, the neuro-identifier for the plant can be represented as

$$\hat{y}(k+1) = F(\vec{X}(k), \vec{W})$$

$\hat{y}(k+1)$ is the output estimated, and

$\vec{W}$ is the weight parameter vector for the neuro-identifier.

11

11

Neuro-Identifier

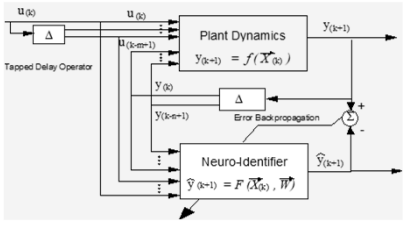


Fig. 2. Block diagram for training the neuro-identifier.

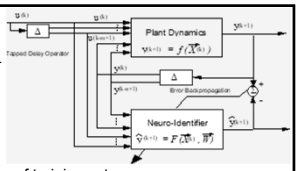
12

12

Neuro-Identifier

The objective of training the neuro-identifier is to reduce the average error defined by

$$J = \frac{1}{N}\sum_{l=1}^{N}\frac{1}{2}\left(y^l(k+1) - \hat{y}^l(k+1)\right)^2,$$



where N is the number of samples in a group of training set.

The equivalent error on the output node of the network for the *i*-th sampled data is defined as the negative of the gradient:

$$\delta_o^i \triangleq -\frac{\partial J}{\partial o^i} = -\frac{\partial J}{\partial \hat{y}^i(k+1)} = \frac{1}{N}\left(y^i(k+1) - \hat{y}^i(k+1)\right)$$

This error is then used backward to compute an equivalent error for a node in an arbitrary layer to update weight parameters in the Backpropagation Algorithm (BPA). Through the learning process, the plant characteristics is stored in the weight parameters of the neuro-identifier:

$$y(k+1) = f(\vec{X}(k)) \quad \approx \quad \hat{y}(k+1) = F(\vec{X}(k), \vec{W})$$

13

13

Feedforward Neuro-Controller

B. Feedforward Neuro-Controller

In designing a controller for a plant to follow an arbitrary reference output, it is necessary to keep the steady-state tracking error to zero. For this purpose, the feedforward neuro-controller (FFNC) is designed to generate a control input which will maintain the plant output to a given reference output in steady-state. The FFNC is then required to learn the inverse dynamics of the plant in steady-state.

A novel approach is now proposed to develop the inverse mapping with the aid of the neuro-identifier:

Note that the steady-state control input can be obtained by setting

$$y(k) \equiv y_{ref} \quad \text{and} \quad u(k) \equiv u_{ref}$$

for all *k* in the NARMA model (1):

$$y(k+1) = f(y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-m+1))$$

i.e.,

$$y_{ref} = f(y_{ref}, y_{ref}, \cdots, y_{ref}, u_{ref}, u_{ref}, \cdots, u_{ref})$$

or equivalently

$$u_{ref} = g(y_{ref})$$

which is the *inverse function* of the steady-state NARMA model.

14

14

Feedforward Neuro-Controller

The FFNC network G, as an inverse mapping of the plant in steady-state, can be developed by using the neuro-identifier F as shown in Fig. 3, i.e.,

$$\hat{y}_{ref} = F(y_{ref}, y_{ref}, \cdots, y_{ref}, u_{ff}, u_{ff}, \cdots, u_{ff}, \vec{W})$$

$$u_{ff} = G(y_{ref}, \vec{W})$$

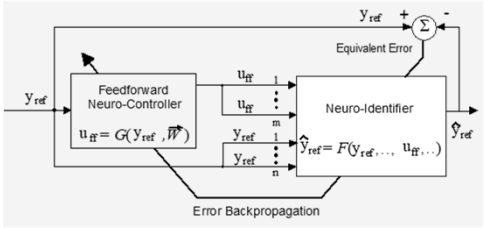


Fig. 3. Block diagram for training the feedforward controller.

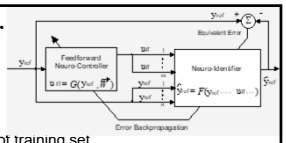
15

15

Feedforward Neuro-Controller

The objective of training the FFNC is to reduce the average error defined by

$$J = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^i_{ref} - \hat{y}^i_{ref}(u^i_{ff}))^2$$



where N is the number of samples in a group of training set.

To update the weight parameters in the FFNC the equivalent error is propagated backward through the neuro-identifier. The equivalent error on the output of the FFNC is defined as the negative sensitivity of the above performance index with respect to u_{ff}, which can be calculated from the equivalent error on the neuro-identifier input nodes:

$$\delta^i_{u_{ff}} \triangleq -\frac{\partial J}{\partial u^i_{ff}} = -\frac{\partial J}{\partial \hat{y}^i_{ref}}\cdot\frac{\partial \hat{y}^i_{ref}}{\partial u^i_{ff}}$$

Since u_{ff} is applied to the first m input nodes of the neuro-identifier,

i.e., $(I_k)^i = u^i_{ff}$, $k = 1, 2, , m,$ then,

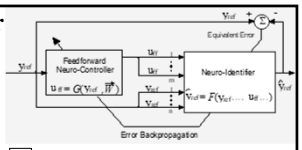
$$\delta^i_{u_{ff}} = \sum_{k=1}^{m} -\frac{\partial J}{\partial \hat{y}^i_{ref}}\cdot\frac{\partial \hat{y}^i_{ref}}{\partial (I_k)^i} = \sum_{k=1}^{m}\delta^i_{I_k},$$

16

16

Feedforward Neuro-Controller

$$\delta^i_{u_{ff}} = \sum_{k=1}^{m} -\frac{\partial J}{\partial \hat{y}^i_{ref}}\cdot\frac{\partial \hat{y}^i_{ref}}{\partial (I_k)^i} = \sum_{k=1}^{m}\delta^i_{I_k},$$



Where $\delta^i_{I_k}$ is the equivalent error of u_{ff}-input node in the neuro-identifier, which is computed by the BPA. Since u_{ff} is also the output of the FFNC, the equivalent error can directly be used as the equivalent error for the network G in the BPA.

Training begins with small random values of weight parameters in the FFNC. This allows the feedforward control input to grow from a small random value, and converge to the smallest solution of u_{ref}, which is preferred over all other possible solutions.

At the end of the training, the weight parameters in the FFNC are adjusted so that the output of the neuro-identifier follows a given reference output

$$y_{ref} = f(y_{ref}, y_{ref}, \cdots, y_{ref}, u_{ref}, u_{ref}, \cdots, u_{ref}) \approx \hat{y}_{ref} = F(y_{ref}, y_{ref}, \cdots, y_{ref}, u_{ff}, u_{ff}, \cdots, u_{ff}, \vec{W})$$

$$u_{ref} \approx u_{ff} = G(y_{ref}, \vec{W})$$

17

17

Feedback Neuro-Controller

C. Feedback Neuro-Controller

The feedback neuro-controller (FBNC) is to stabilize tracking error dynamics when the plant output is following an arbitrarily given reference output. This objective can be achieved by minimizing the modified quadratic performance index:

$$J = \sum_{k=1}^{N} J_k = \frac{1}{2}\sum_{k=1}^{N}(Q(y_{ref} - y_{(k+1)})^2 + R(u_{fb(k)})^2)$$

where $u_{fb(k)}$ is the feedback control input.

From the NARMA model, the feedback control input can be viewed as an inverse mapping

$$u_{fb(k)} = h(y_{ref}, y_{(k)}, y_{(k-1)}, \cdots, y_{(k-n+1)}, u_{(k-1)}, u_{(k-2)}, \cdots, u_{(k-m+1)})$$

The corresponding FBNC can be represented as a nonlinear network

$$u_{fb(k)} = H(y_{ref}, y_{(k)}, y_{(k-1)}, \cdots, y_{(k-n+1)}, u_{(k-1)}, u_{(k-2)}, \cdots, u_{(k-m+1)}, \vec{W})$$

Since the target value for the optimal feedback control $u_{fb(k)}$ is not available for training, traditional BPA method is not applicable here.

18

18

Generalized Backpropagation-Through-Time

D. Generalized Backpropagation-Through-Time Algorithm

The GBTT is to generate an equivalent error from a general quadratic cost function, and it is an extension of the Backpropagation-Through-Time (BTT) algorithm of Werbos [10]. The original BTT was for the cost function with output error only. On the other hand, the GBTT is for the general quadratic cost function which includes not only output errors, but also input variables.

The GBTT is based upon output and input sensitivities of the cost function defined by:

$$\delta_y^k \triangleq -\frac{\partial J}{\partial y(k)} \qquad \delta_u^k \triangleq -\frac{\partial J}{\partial u(k)}$$

Since, for a fixed feedforward control,

$$\delta_{ub}^k = -\frac{\partial J}{\partial u_{fb}(k)} = -\frac{\partial J}{\partial u(k)}\frac{\partial u(k)}{\partial u_{fb}(k)} = -\frac{\partial J}{\partial u(k)}\frac{\partial (u_{ff}+u_{fb}(k))}{\partial u_{fb}(k)} = -\frac{\partial J}{\partial u(k)} = \delta_u^k$$

the subscript *fb* will be dropped in the following development.

19

19

Generalized Backpropagation-Through-Time

Output Sensitivity Equation (OSE) :

An output $y(k)$ will influence the plant dynamics for the next n steps:

$$y(k+1) = f\big(y(k), y(k-1), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-m+1)\big)$$

Similarly, since the inverse dynamics also has n delayed output variables, an output $y(k)$ will influence the input for the next n steps:

$$u_{fb}(k) = h\big(y_{ref}, y(k), y(k-1), \cdots, y(k-n+1), u(k-1), u(k-2), \cdots, u(k-m+1)\big)$$

Recall that the performance index is defined on a finite interval, i.e.,

$$J = J\big(y(j+1), u(j)\ ;\ j = 1, 2, \cdots, N\big)$$

Thus, the gradient of J with respect to an output y(k) is

$$\frac{\partial J}{\partial y(k)} = \sum_{\substack{i=1 \\ k+i\le N+1}}^{n} \frac{\partial J}{\partial y(k+i)}\frac{\partial y(k+i)}{\partial y(k)} + \sum_{\substack{i=0 \\ k+i\le N}}^{n-1} \frac{\partial J}{\partial u(k+i)}\frac{\partial u(k+i)}{\partial y(k)} - Q(y_{ref} - y(k))$$

$$= \sum_{\substack{i=k+1 \\ i\le N+1}}^{k+n} \frac{\partial J}{\partial y(i)}\frac{\partial y(i)}{\partial y(k)} + \sum_{\substack{i=k \\ i\le N}}^{k+n-1} \frac{\partial J}{\partial u(i)}\frac{\partial u(i)}{\partial y(k)} - Q(y_{ref} - y(k)) \ .$$

20

20

Generalized Backpropagation-Through-Time

$$J = \sum_{k=1}^{N} J_k = \frac{1}{2}\sum_{k=1}^{N} \big(Q(y_{ref} - y(k+1))^2 + R(u_{fb}(k))^2\big)$$

Thus, the gradient of J with respect to an output $y(k)$ is

$$\frac{\partial J}{\partial y(k)} = \sum_{\substack{i=1 \\ k+i\le N+1}}^{n} \frac{\partial J}{\partial y(k+i)}\frac{\partial y(k+i)}{\partial y(k)} + \sum_{\substack{i=0 \\ k+i\le N}}^{n-1} \frac{\partial J}{\partial u(k+i)}\frac{\partial u(k+i)}{\partial y(k)} - Q(y_{ref} - y(k))$$

$$= \sum_{\substack{i=k+1 \\ i\le N+1}}^{k+n} \frac{\partial J}{\partial y(i)}\frac{\partial y(i)}{\partial y(k)} + \sum_{\substack{i=k \\ i\le N}}^{k+n-1} \frac{\partial J}{\partial u(i)}\frac{\partial u(i)}{\partial y(k)} - Q(y_{ref} - y(k)) \ .$$

By using the definition of sensitivities,

$$\delta_y^k = \sum_{\substack{i=k+1, \\ i\le N+1}}^{k+n} \delta_y^i \frac{\partial y(i)}{\partial y(k)} + \sum_{\substack{i=k, \\ i\le N}}^{k+n-1} \delta_u^i \frac{\partial u(i)}{\partial y(k)} + Q\big(y_{ref} - y(k)\big)$$

Note that this output sensitivity equation (OSE) is depending on the input sensitivities as well.

21

21

Generalized Backpropagation-Through-Time

Thus, the gradient of J with respect to an input

$$\frac{\partial J}{\partial u(k)} = \sum_{\substack{i=1 \\ k+i \le N+1}}^{m} \frac{\partial J}{\partial y(k+i)} \frac{\partial y(k+i)}{\partial u(k)} + \sum_{\substack{i=1 \\ k+i \le N}}^{m-1} \frac{\partial J}{\partial u(k+i)} \frac{\partial u(k+i)}{\partial u(k)} + Ru_{fb}(k)$$

$$= \sum_{\substack{j=k+1 \\ j \le N+1}}^{k+m} \frac{\partial J}{\partial y(j)} \frac{\partial y(j)}{\partial u(k)} + \sum_{\substack{i=k+1 \\ i \le N}}^{k+m-1} \frac{\partial J}{\partial u(j)} \frac{\partial u(j)}{\partial u(k)} + Ru_{fb}(k).$$

By using the definition of sensitivities,

$$\delta_u^k = \sum_{\substack{i=k+1, \\ i \le N+1}}^{k+m} \delta_y^i \frac{\partial y(i)}{\partial u(k)} + \sum_{\substack{i=k+1, \\ i \le N}}^{k+m-1} \delta_u^i \frac{\partial u(i)}{\partial u(k)} - Ru_{fb}(k)$$

This ISE is also depending on the output sensitivities, and both are coupled to one another.

22

22

Generalized Backpropagation-Through-Time

Since the plant dynamics and the inverse dynamics are not known, they are approximated by the corresponding networks, the neuro-identifier *F* and the feedback neuro-controller *H*, to yield

$$\delta_y^k = \sum_{\substack{i=k+1, \\ i \le N+1}}^{k+m} \delta_y^i \frac{\partial F_i}{\partial \hat{y}(k)} + \sum_{\substack{i=k, \\ i \le N}}^{k+m-1} \delta_u^i \frac{\partial H_i}{\partial \hat{y}(k)} + Q(y_{ref} - \hat{y}(k))$$

$$\delta_u^k = \sum_{\substack{i=k+1, \\ i \le N+1}}^{k+m} \delta_y^i \frac{\partial F_i}{\partial u(k)} + \sum_{\substack{i=k+1, \\ i \le N}}^{k+m-1} \delta_u^i \frac{\partial H_i}{\partial u(k)} - Ru_{fb}(k)$$

It should be noted that the last terms in OSE and ISE are, respectively, the error terms for the output and input variables, and the terms under summation operations are the error (or delta) terms backpropagated through the networks *F* and *H*.

The objective of the GBTT is to compute the sensitivity δ_u^k, which will be used as the equivalent error for training of FBNC. This can be achieved by solving the OSE and ISE backward starting from $j = N+1$:

23

23

Generalized Backpropagation-Through-Time

j=*N*+1:

$$\delta_u^{N+1} = 0$$
$$\delta_y^{N+1} = Q(y_{ref} - \hat{y}(N+1)),$$

j=*N*:

$$\delta_u^N = \delta_y^{N+1} \frac{\partial F_{N+1}}{\partial u(N)} - Ru_{fb}(N)$$
$$\delta_y^N = \delta_y^{N+1} \frac{\partial F_{N+1}}{\partial \hat{y}(N)} + \delta_y^N \frac{\partial H_N}{\partial \hat{y}(N)} + Q(y_{ref} - \hat{y}(N)),$$

j=*N*-1:

$$\delta_u^{N-1} = \delta_y^{N+1} \frac{\partial F_{N+1}}{\partial u(N-1)} + \delta_y^N \frac{\partial F_N}{\partial u(N-1)} + \delta_u^N \frac{\partial H_N}{\partial u(N-1)} - Ru_{fb}(N-1)$$
$$\delta_y^{N-1} = \delta_y^{N+1} \frac{\partial F_{N+1}}{\partial \hat{y}(N-1)} + \delta_y^N \frac{\partial F_N}{\partial \hat{y}(N-1)} + \delta_u^N \frac{\partial H_N}{\partial \hat{y}(N-1)} + \delta_u^{N-1} \frac{\partial H_{N-1}}{\partial \hat{y}(N-1)} + Q(y_{ref} - \hat{y}(N-1)),$$

j=*N*-2:

$$\delta_u^{N-2} = \delta_y^{N+1} \frac{\partial F_{N+1}}{\partial u(N-2)} + \delta_y^N \frac{\partial F_N}{\partial u(N-2)} + \delta_y^{N-1} \frac{\partial F_{N-1}}{\partial u(N-2)} + \delta_u^N \frac{\partial H_N}{\partial u(N-2)} + \delta_u^{N-1} \frac{\partial H_{N-1}}{\partial u(N-2)} - Ru_{fb}(N-2)$$
$$\delta_y^{N-2} = \delta_y^{N+1} \frac{\partial F_{N+1}}{\partial \hat{y}(N-2)} + \delta_y^N \frac{\partial F_N}{\partial \hat{y}(N-2)} + \delta_y^{N-1} \frac{\partial F_{N-1}}{\partial \hat{y}(N-2)} + \delta_u^N \frac{\partial H_N}{\partial \hat{y}(N-2)} + \delta_u^{N-1} \frac{\partial H_{N-1}}{\partial \hat{y}(N-2)} + \delta_u^{N-2} \frac{\partial H_{N-2}}{\partial \hat{y}(N-2)} + Q(y_{ref} - \hat{y}(N-2)),$$

24

Generalized Backpropagation-Through-Time

Forward Simulator:

Before solving the OSE and ISE, the error terms need to be generated. This can be done by driving the neuro-identifier with the controllers FFNC and FBNC for *N* step forward. This process is illustrated by the *forward simulator* shown in Fig. 4.

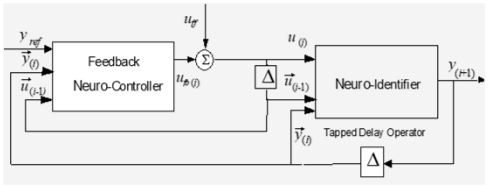


Fig. 4. Forward simulator for GBTT.

25

25

Generalized Backpropagation-Through-Time

Backward Simulator:

The OSE and ISE are to be simulated backward in order to backpropagate the error terms. This can be performed by the *backward simulator* shown in Fig. 5, where *the summed advance operator* ∇ : $X(i) = \sum_{k=1}^{n} x(i+k, k)$

The backward simulator can be shown to be the dual of the forward simulator, which is then constructed by using the *duality principle*, i.e., reversing the direction of arrows, interchanging the summers and nodes, and replacing the tapped delay operators with the summed advance operators.

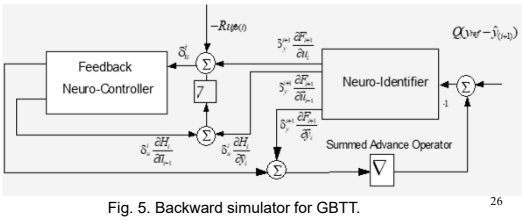


Fig. 5. Backward simulator for GBTT.

26

26

Generalized Backpropagation-Through-Time

The process of the GBTT training algorithm is summarized as follows:

1) Set the weight parameters of the FBNC with small random numbers.
2) Set the reference output and initial state with random numbers in the operation region of the plant.
3) Run the forward simulator for *N* step forward from *i* = 1.
4) Using the operation result in step 3), run the backward simulator backward from *i* = *N* to evaluate the equivalent error δ_u^i and the weight adjustment vector $\Delta \vec{W}^i$
5) Update the weight parameters in the FBNC by using the average of the weight adjustment vectors found in step 4).
6) Go to 2).

27

27

Case Study

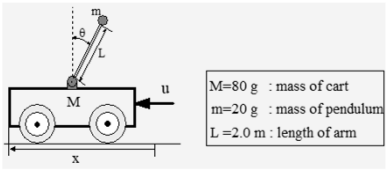


Fig. 6. Inverted pendulum.

The optimal tracking neuro-controller (OTNC) is constructed and trained by the proposed method to meet the following control objectives:

(1) Set the pendulum to an arbitrarily given reference angle.

(2) Minimize the quadratic cost function while tracking the reference angle.

28

28

Case Study

A. Training of the Neuro-Identifier

The nonlinear differential equation of the plant dynamics is as follows:

$$(M+m)\ddot{x} + mL\cos(\theta)\ddot{\theta} - mL\sin(\theta)\dot{\theta}^2 = u$$
$$m\ddot{x}\cos(\theta) + mL\ddot{\theta} = mg\sin(\theta),$$

A paradigm for the neuro-identifier is chosen by trial and error. It consists of two hidden layers with 40 nodes each, an input layer with 6 input nodes and an output layer with one node. Three of the six input nodes are for output history, $y(k),\ y(k-1),\ y(k-2)$ and two for input history, $u(k),\ u(k-1)$ and one for bias input, 1.0.

Training patterns of the neuro-identifier are generated from the mathematical model with random initial value and random input within the operation region of 1.1 [rad]. Discrete-time training patterns are obtained by applying the modified Euler method with time step-size of 0.13 [sec] in simulation.

29

29

Case Study

To avoid oscillation during training stage, weight parameters are corrected from the average of corrections calculated for every ten patterns. After training the neuro-identifier for 1 hour in a SUN-SPARC2 workstation, it is tested with arbitrary initial conditions and sinusoidal inputs of different amplitude, which is presented in Fig. 7. The neuro-identifier approximates the plant very closely and is sufficient for training the neuro-controllers.

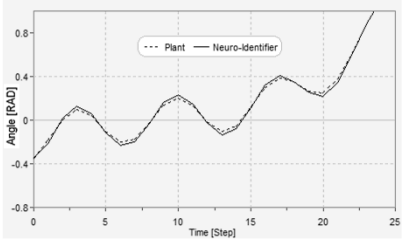


Fig. 7. Training results of the neuro-identifier:
(a) initial condition $\theta = -0.3$ $u = -0.9\cos(k)$

30

30

Case Study

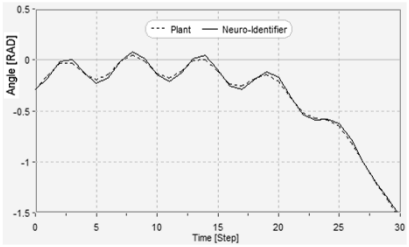


Fig. 7. Training results of the neuro-identifier:
(b) initial condition $\theta = -0.3$ $u = -0.8\cos(k)$

31

31

Case Study

B. Training of the Feedforward Controller

The feedforward neuro-controller (FFNC) has two hidden layers with 30 nodes each. The input layer has two nodes:
one for reference output y_{ref}
and one for the bias input,
and the output layer has one node for the control u_{ff}

The reference output is given randomly to be within the operation region of 0.9 [rad] to train the FFNC, which is coupled with the neuro-identifier and the plant.

After training the FFNC for 1 hour, it is tested with a reference output varying within the operation region. Although the plant tracks the time-varying reference output, error remains small as shown in Fig. 8.

32

32

Case Study

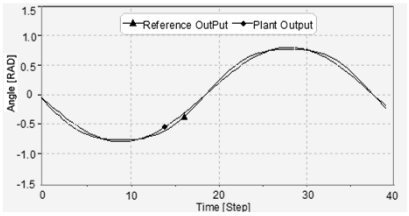


Fig. 8. Training result of the feedforward neuro-controller.

33

33

Case Study

C. Training of the Feedback Neuro-Controller

The feedback neuro-controller (FBNC) has two hidden layers with 30 nodes each. The input layer has five nodes: three for output history, $y(k), y(k-1), y(k-2)$ one for previous input, $u(k-1)$ and one for the bias input.

The cost function for the *N*-step ahead optimal controller is set as

$$J = \frac{1}{2}\sum_{k=1}^{N}(1.0(y_{ref} - y_{(k+1)})^2 + 0.3(u_{fb(k)})^2)$$

The FBNC is trained once for an initial condition and a reference output which are randomly selected while driving the plant for *N* steps. This training is repeated for other initial conditions and reference outputs. Each training is performed in two phases. First, the training is done with small *N* (=3) since the controller in the beginning has little knowledge of control. This also prevents the pendulum from falling down. Then, the step is increased gradually to *N* =15. The second phase training is carried on with *N* fixed at 15.

34

34

Case Study

After training the FBNC with the GBTT algorithm for 2 hours, it is tested with several non-zero set-points as presented in Fig. 9. It shows a larger overshoot for a larger set-point. A larger overshoot corresponds to an operating condition with severe nonlinearity.

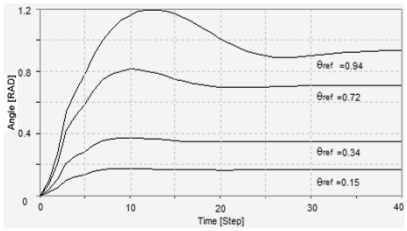


Fig. 9. Angle trajectories of the inverted pendulum for different reference set-points.

35

35

Case Study

Fig. 10 shows a case for a set-point changing at each 40 time-steps. It is seen that the OTNC for a nonlinear system behaves in a way similar to the usual optimal tracking controller for a linear quadratic problem; the shapes of output trajectories are typical fast responses with reasonable overshoots.

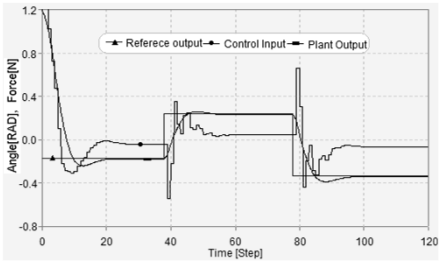


Fig. 10. Control result with the OTNC for changing reference set-point.

36

36

Case Study

Fig. 11 shows the corresponding feedforward and the feedback control inputs for the changing set-point. The FFNC generates the control input corresponding only to the reference output in steady-state. On the other hand, the FBNC generates the control input corresponding to the regulating error between the reference and the plant outputs during transient.

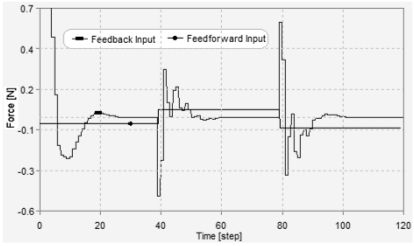


Fig. 11. Feedforward and the feedback control inputs for a changing reference set-point.

37

37

Conclusions

- For an optimal tracking control problem for nonlinear dynamic plants, a new architecture, the optimal tracking neuro-controller (OTNC), is developed using feedback and feedforward controls.
- First, the feedforward neuro-controller (FFNC) is introduced to solve the tracking problem with a non-zero set-point.
- A novel training method for the FFNC is developed by using the concept of an inverse mapping to generate the feedforward control input corresponding to the output set-point.
- Second, the feedback neuro-controller (FBNC) is designed to solve an optimal regulator problem with general quadratic cost function.
- A Generalized Backpropagation-Through-Time (GBTT) training algorithm is developed to train the FBNC.
- Simulation results show good performance over a wide range of nonlinear operation and the possibility of using the OTNC for the optimal tracking control of other nonlinear systems.

38

38