Machine intelligence
Neural Networks in Control Systems

Dr. Ahmad Al-Mahasneh

Introduction

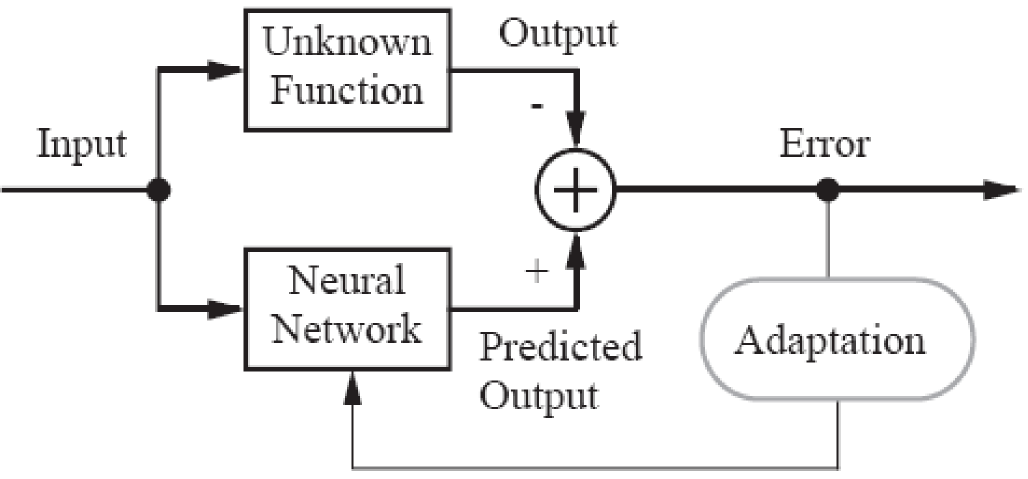


Figure 1 Neural Network as Function Approximator

NNs are universal approximator: they can approximate any nonlinear function under certain conditions.

So what that implies? We can use them for mapping nonlinear systems in system identification and control.

Dynamic systems representation

Differential equation
Continuous-time

$$\dot{x}(t) = \Phi\left[x(t), u(t)\right]$$

$$y(t) = \Psi\left[x(t)\right]$$

Difference equation
Discrete-time

$$x(k+1) = \Phi\left[x(k), u(k)\right]$$

$$y(k) = \Psi\left[x(k)\right]$$

Linear difference equation

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k)$$

Multilayer Perceptron Architecture

The most popular types of NNs are the MLPs.

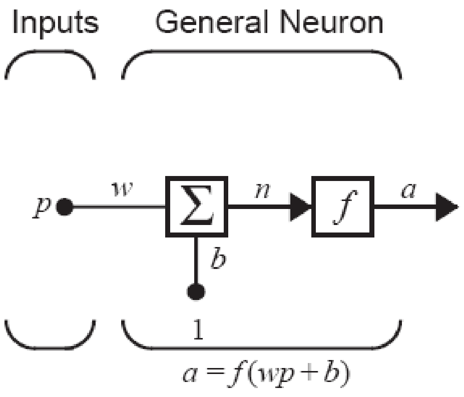


Figure 2 Single-Input Neuron

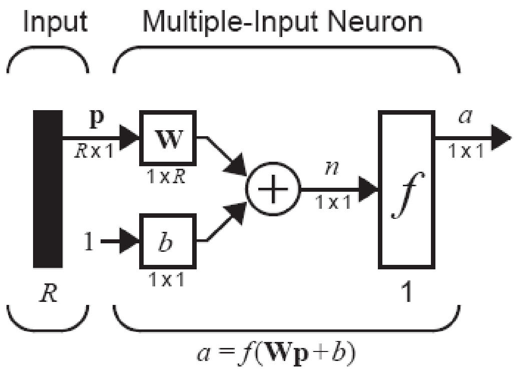


Figure 5 Neuron with R Inputs, Matrix Notation

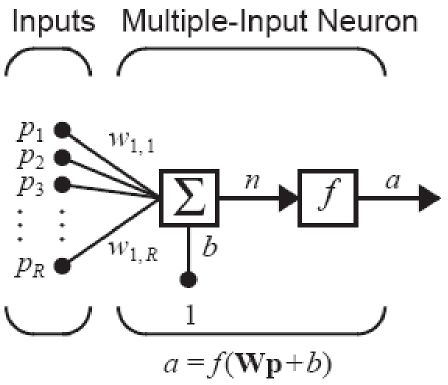


Figure 4 Multiple-Input Neuron

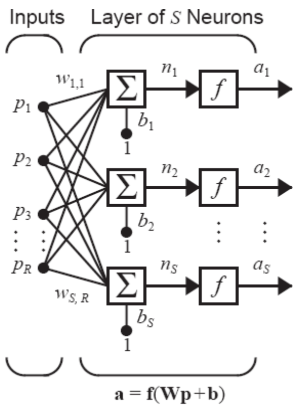
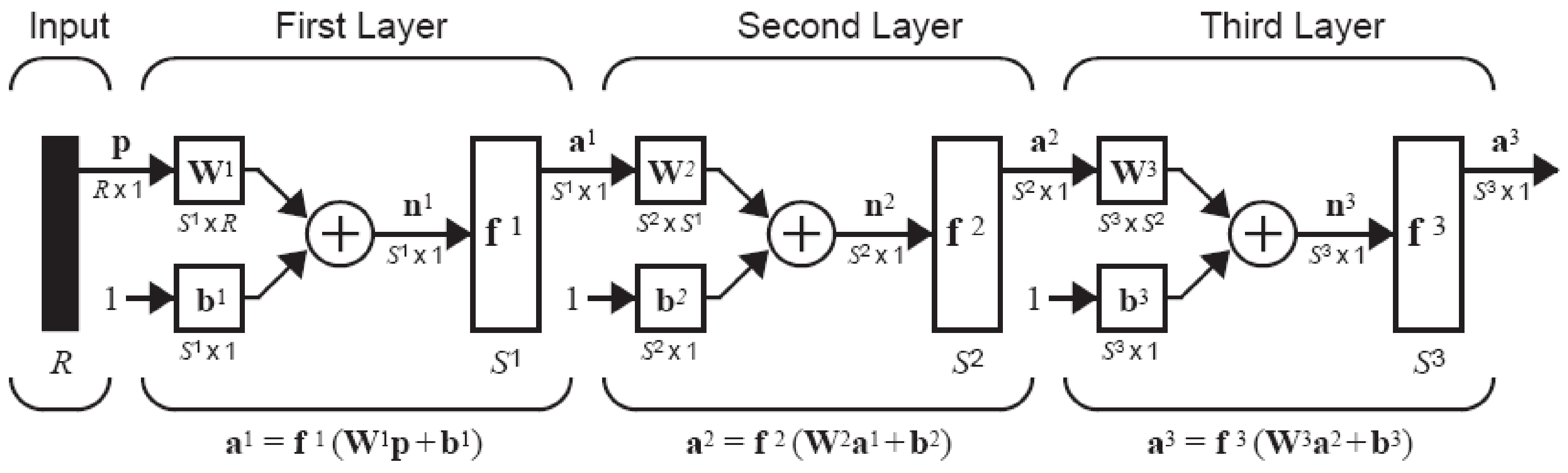


Figure 6 Layer of S Neurons

Multilayer Perceptron Architecture



$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)$ $\mathbf{a}^2 = \mathbf{f}^2(\mathbf{W}^2\mathbf{a}^1+\mathbf{b}^2)$ $\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{a}^2+\mathbf{b}^3)$

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

Figure 8 Three-Layer Network

Control System Applications

- Two steps involved in neural network control:
  - System Identification
  - Control Design
- Types of NNs controllers:
  1. Direct/indirect NN control
  2. Backpropagation-Through-Time Control
  3. Direct inverse control
  4. Model predictive control
  5. Model reference adaptive control (MRAC)
  6. Self-tuning control (STC)
  7. NARMA-L2 Control.
  8. Reinforcement learning.

System Identification

NNs will be used to approximate the plant dynamics.

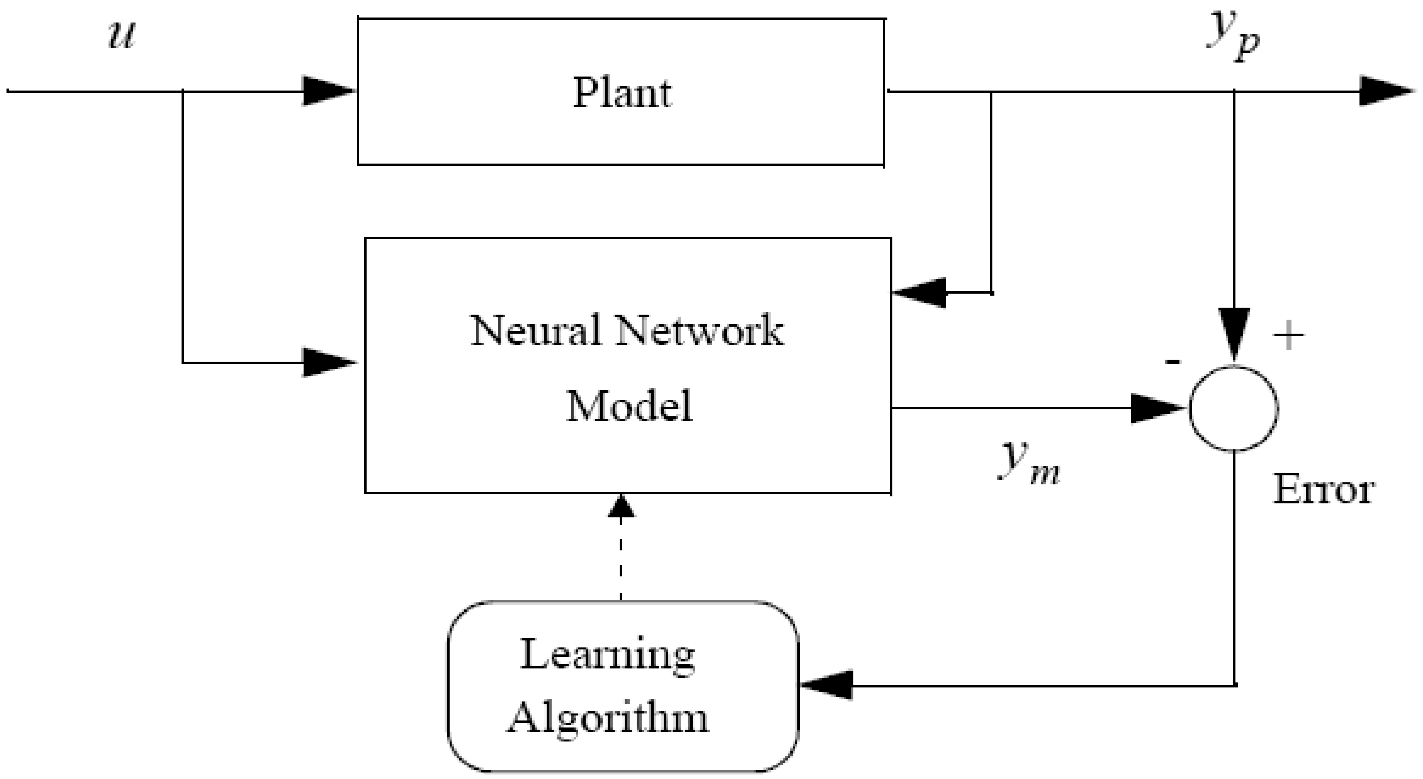


Figure 13 Plant Identification

System Identification

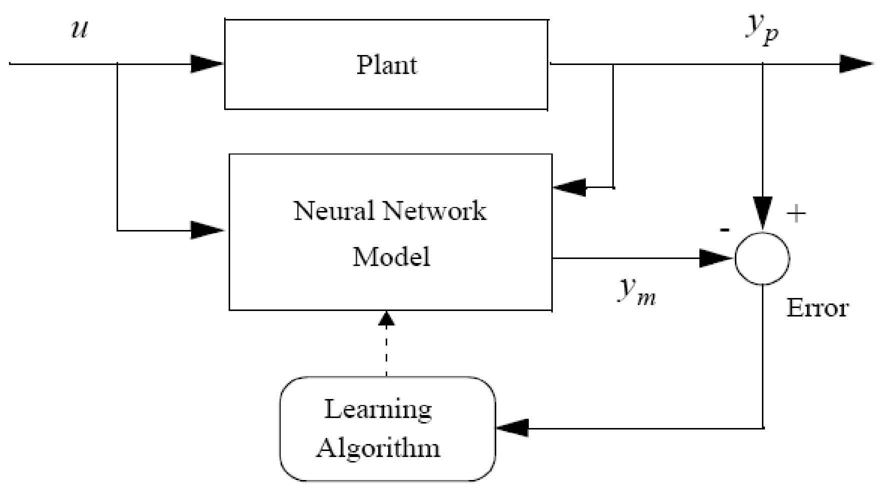


Figure 13 Plant Identification

$$y(k+d) = h[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)]$$

$$y_m(k+1) = \hat{h}[y_p(k), \ldots, y_p(k-n+1), u(k), \ldots, u(k-m+1); \mathbf{x}]$$

$\mathbf{x}$ is the vector containing all network weights and biases

System Identification

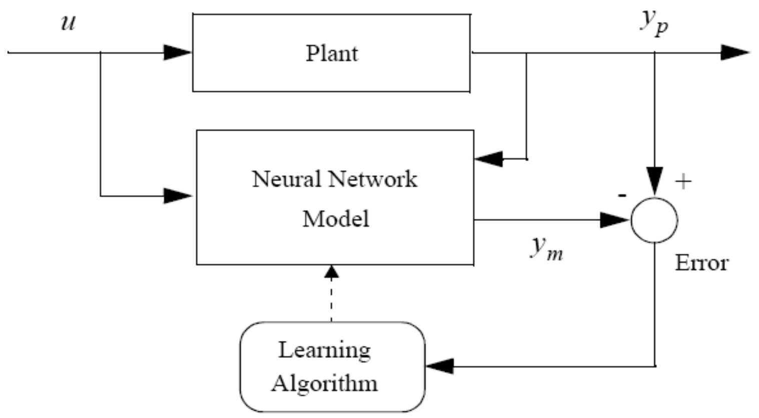


Figure 13 Plant Identification

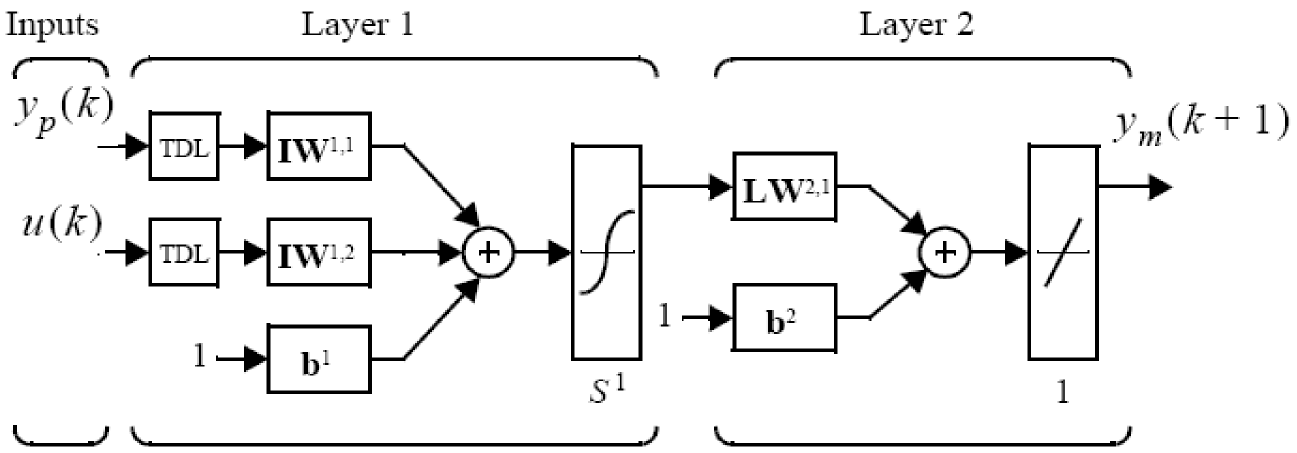
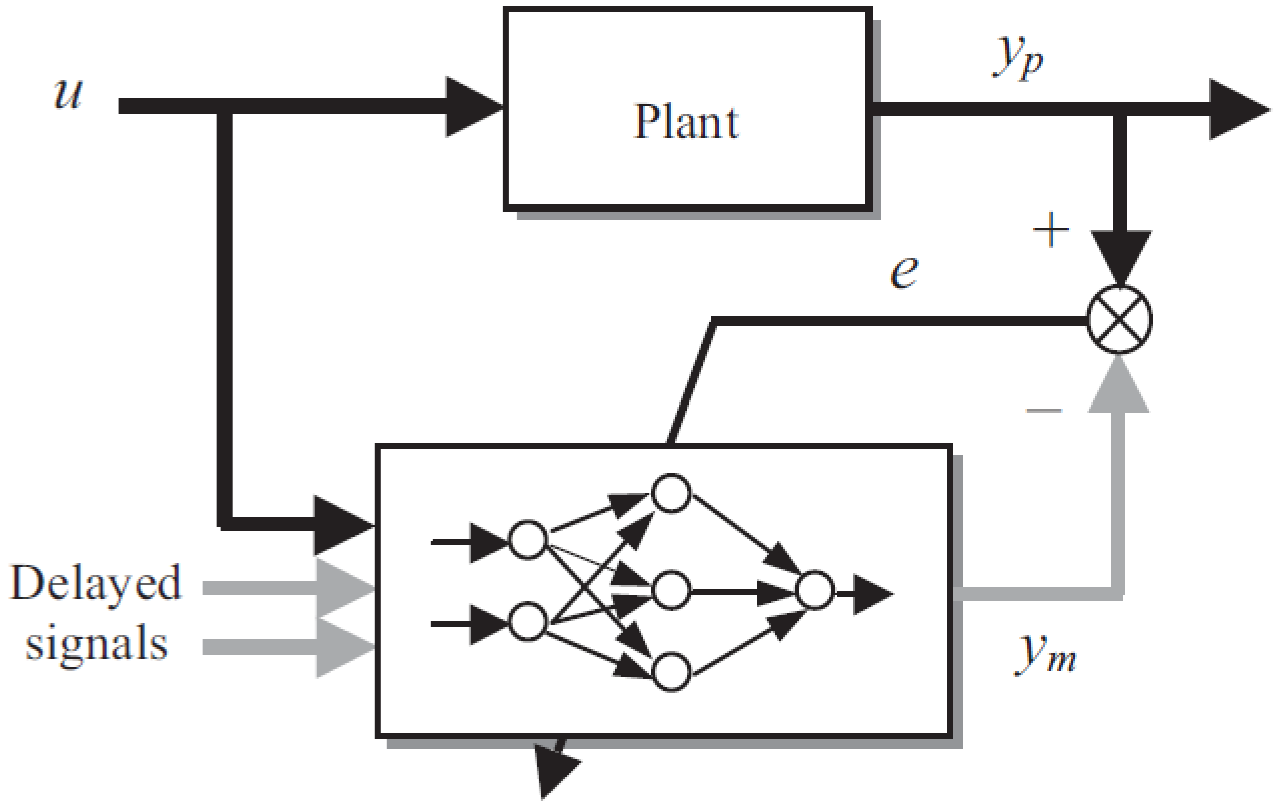
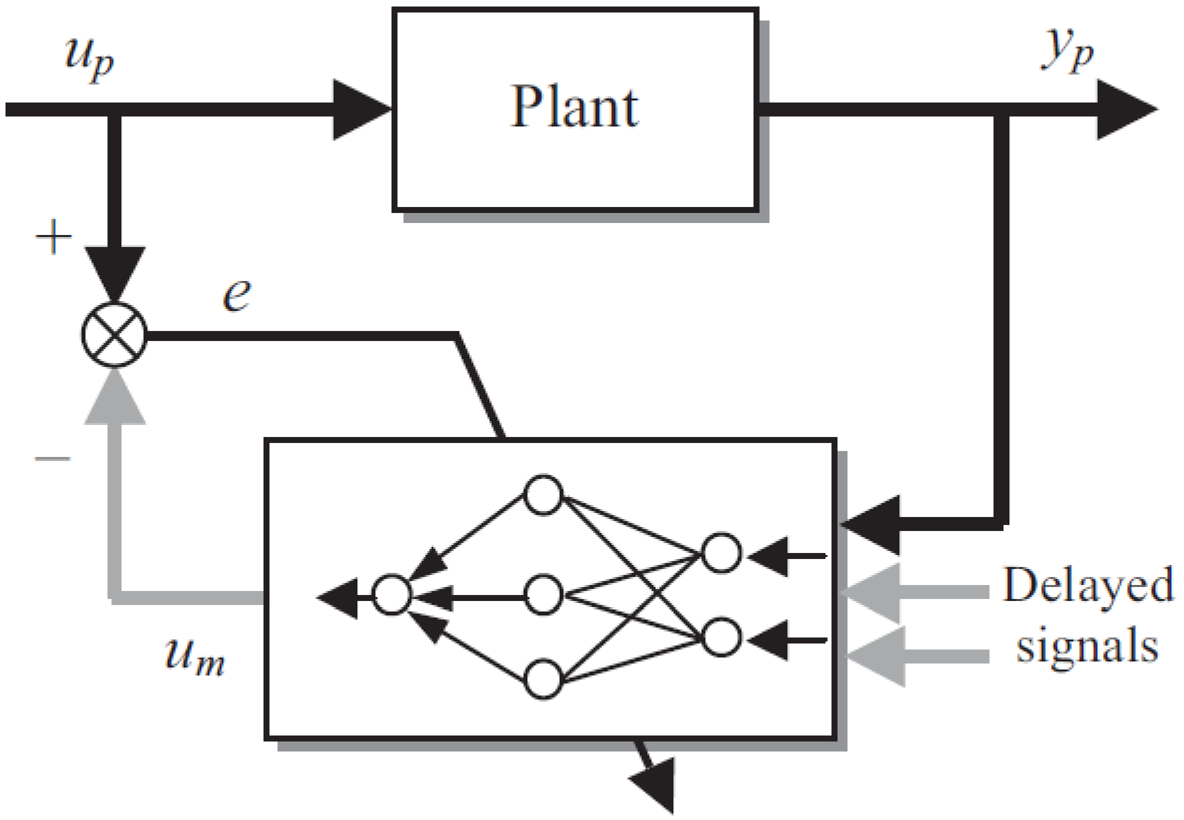


Figure 14 Neural Network Plant Model

Forward plant identification



Direct inverse identification



Feedback control system

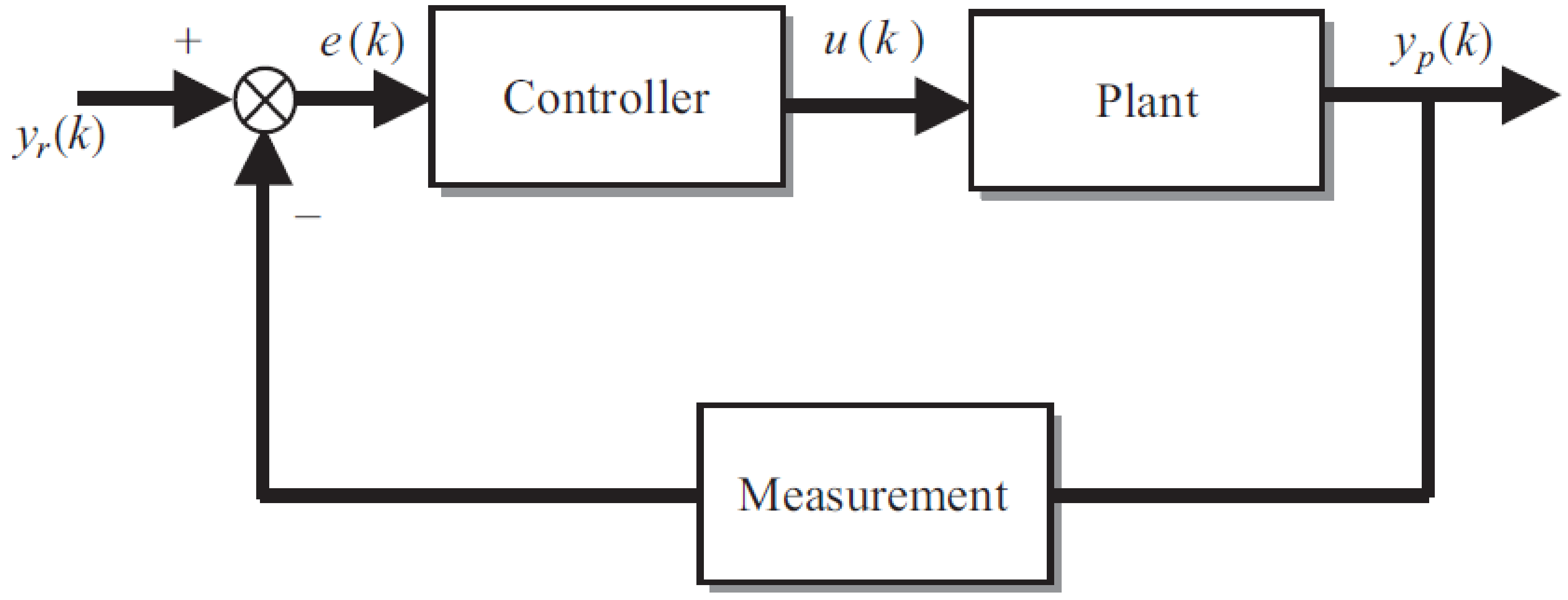


Figure 5.2 Block diagram of feedback control system

NNs in control

Traditional control methodologies are mainly based on linear systems theory, while real systems are nonlinear in nature.

NNs are good at control because:

- The inherent massive parallelism.
- Fast adaptability of neural network implementations are additional advantages.
- Powerful learning algorithms and Variety of architectures
- The ability to train the NNs from input/output functions and/or experiential data.
- NNs provide simpler solutions to complex control problems.
- The success of the backpropagation algorithm to train multilayered NNs.

NNs in control

NNs are used in control applications such as:

- Process control
- Robotics
- Manufacturing
- Aerospace

The basic objective of neuro-control is to provide the appropriate input signal to a given physical system (process or plant) to yield its desired response (desired performance).

There are typically two steps involved when using NNs for control:

- System (plant or process) identification and
- Control design.

Direct adaptive control

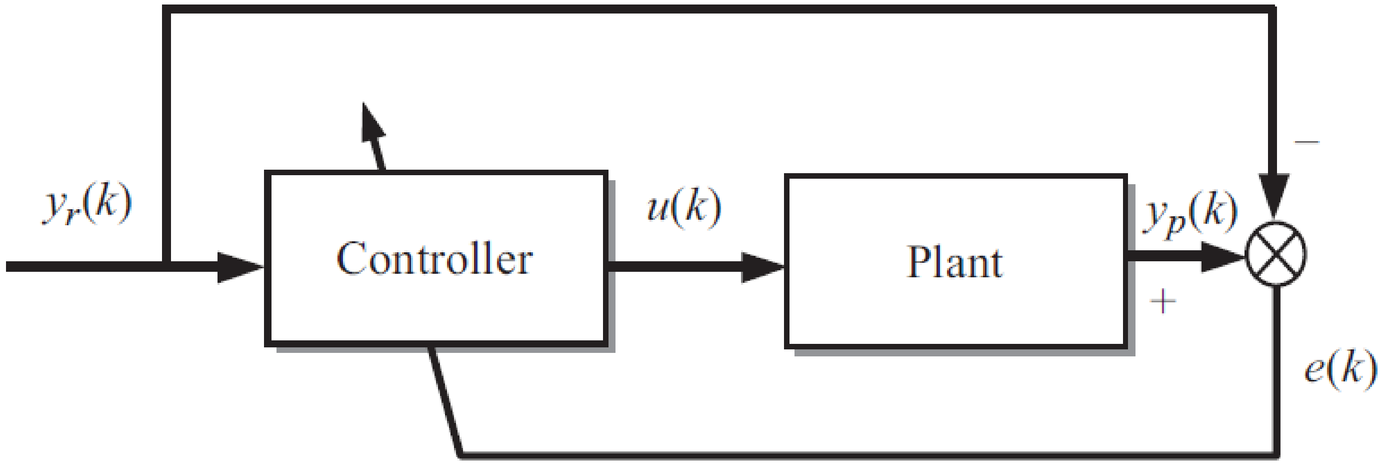


Figure 5.3 Block diagram of direct adaptive control system

- One advantage is that no identification is involved in this method.
- The disadvantage of the direct NN control is that the plant's initial stability is not guaranteed for this method of control.
- The Jacobian of the plant is required which is difficult to obtain if the plant dynamics is not known *a priori*

Direct NNs control

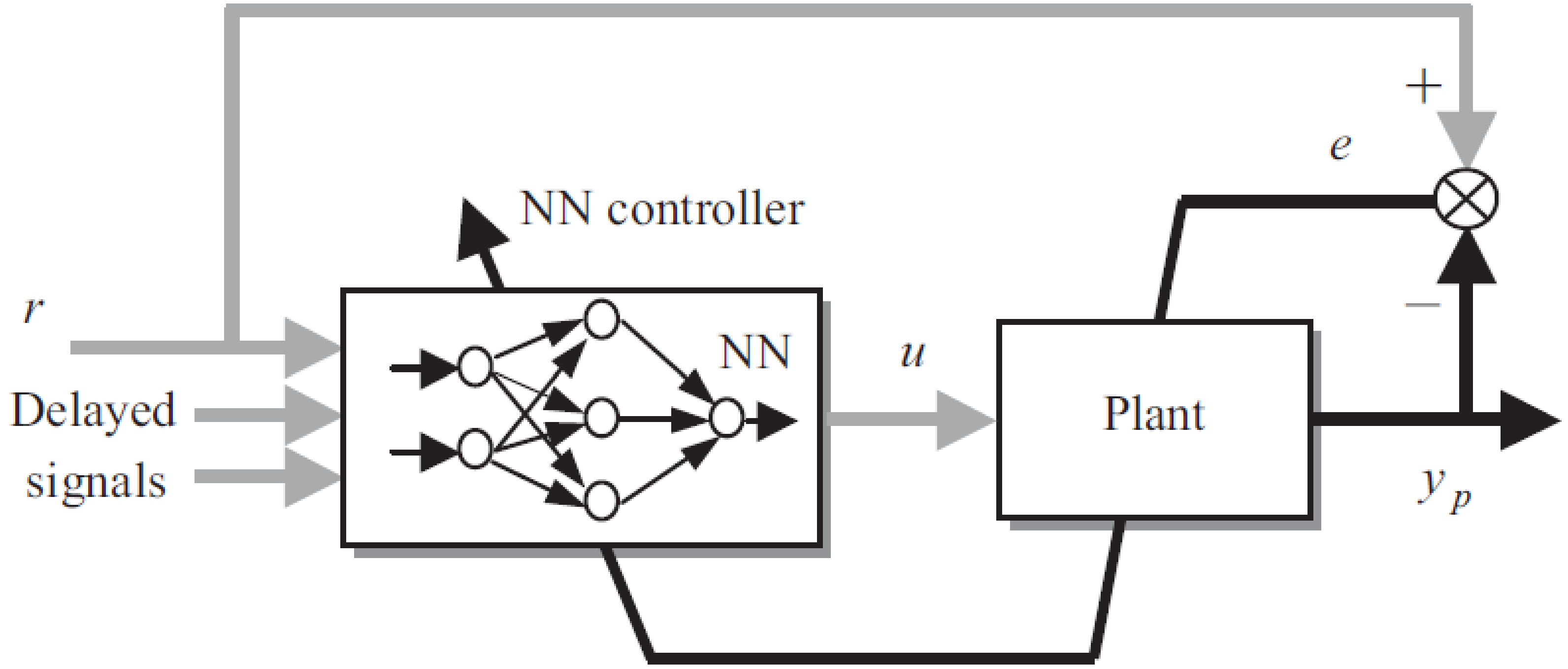
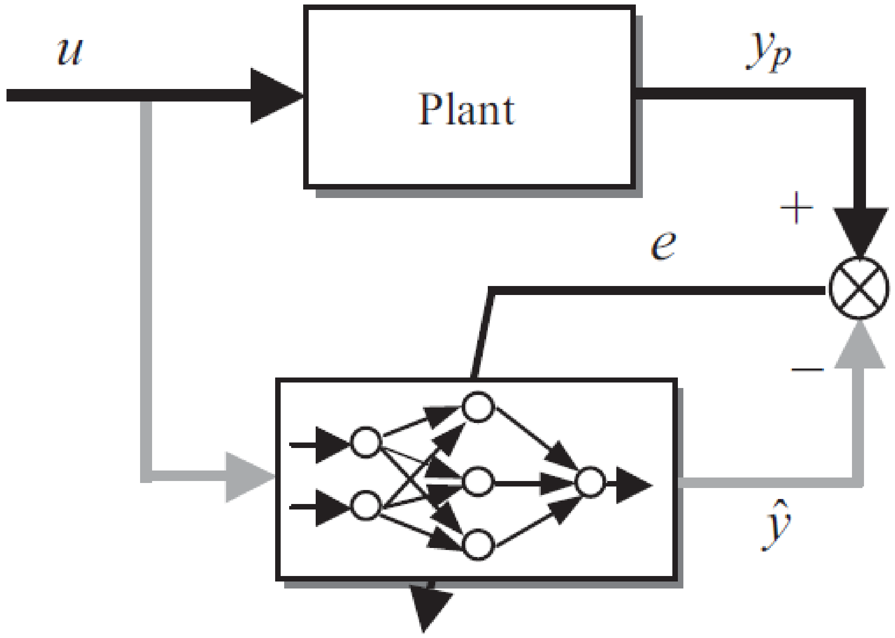


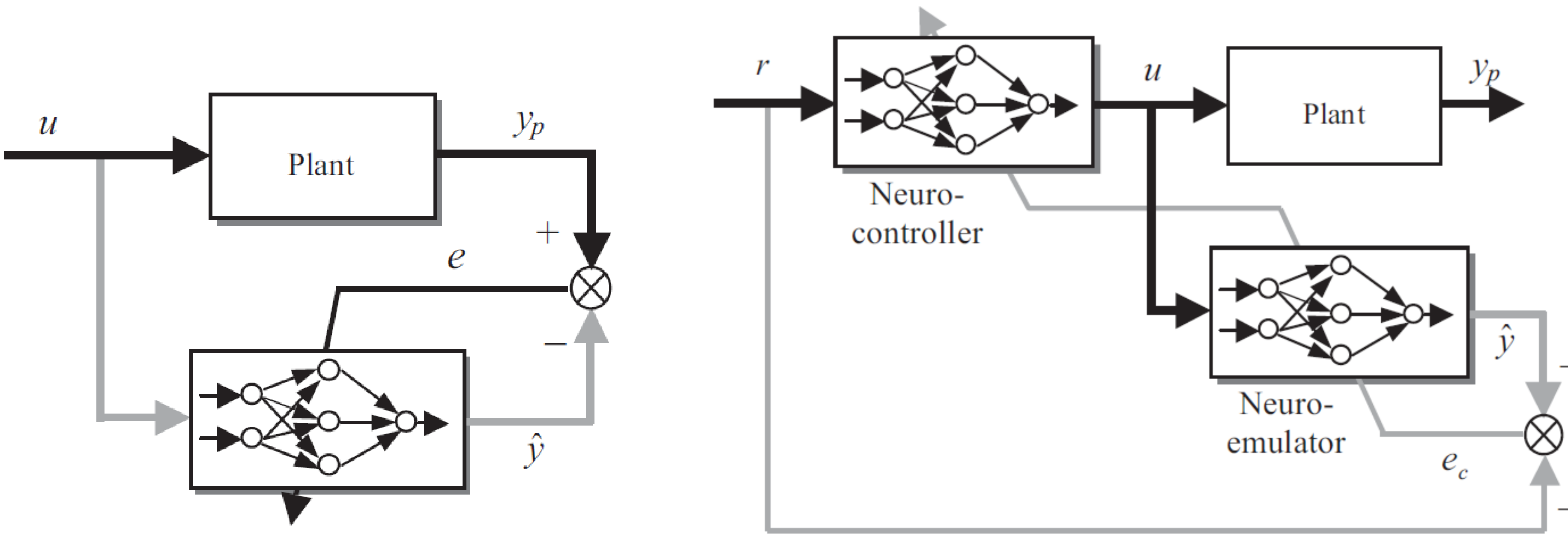
Figure 5.6 Direct (or specialized learning) control architecture

Indirect NNs control



- The added advantage of indirect control is that the parameters of the neuro-emulator can be readjusted online during operation of the controller if the neuro-emulator appears not to be accurate.
- Among the disadvantages of the indirect control scheme appears to be the robustness of the controller, as there is no feedback loop used in the control strategy.

Indirect NNs control



(a) Training of neuro-emulator; (b) Training of neuro-controller

Backpropagation-Through-Time Control

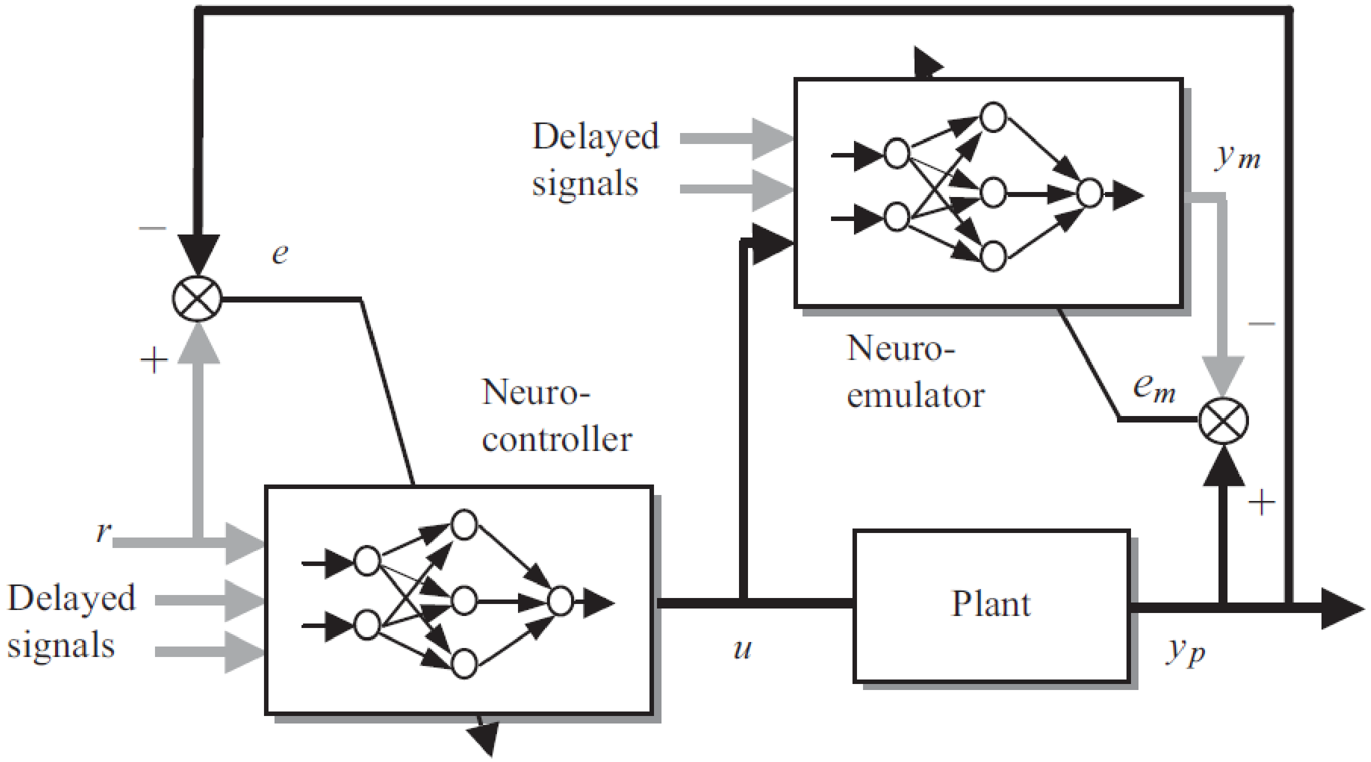
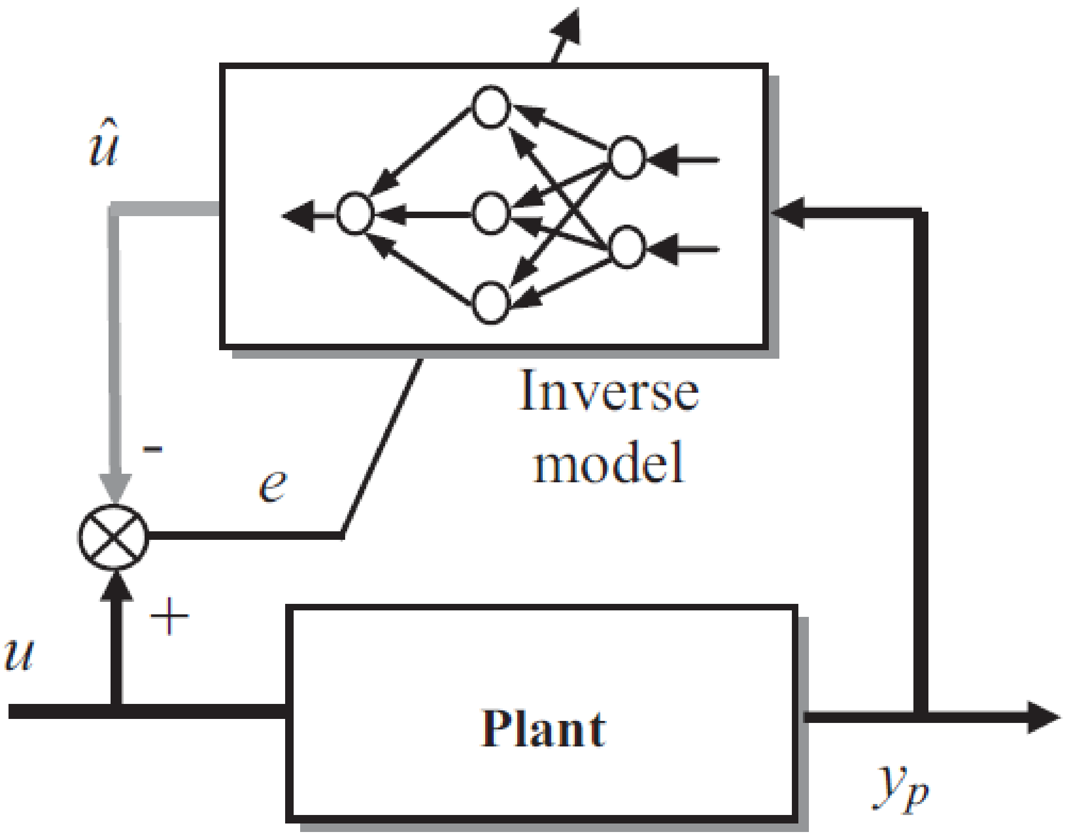
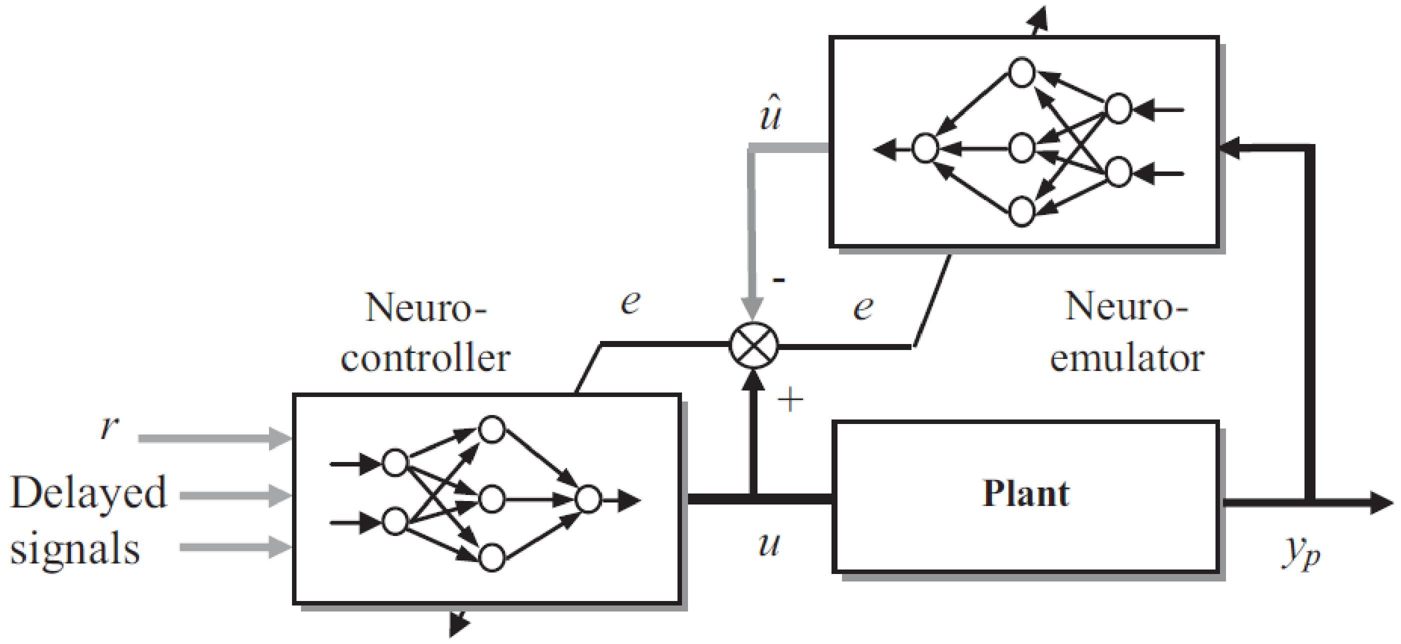


Figure 5.8 Backpropagation-through-time architecture

NNs direct inverse control



NNs direct inverse control



Disadvantages of direct inverse control include:

- Lack of robustness, resulting from the fact that no direct feedback error is used in direct inverse control.
- Inefficient learning, caused by improper operational range of data.
- Actual operational data may be hard to define *a priori*.

NN-based MPC

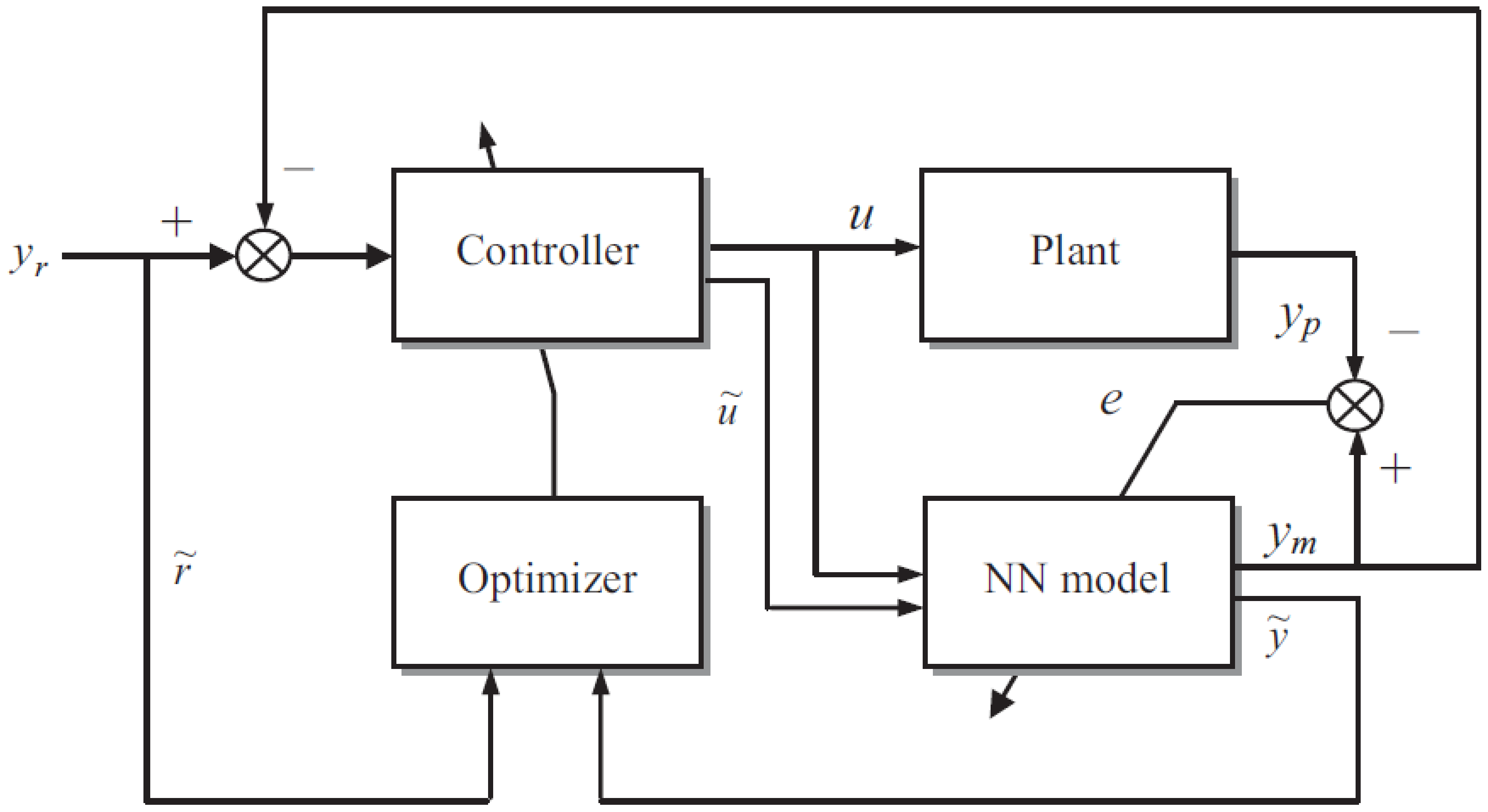


Figure 5.10 NN-based model predictive control

NN-based MPC

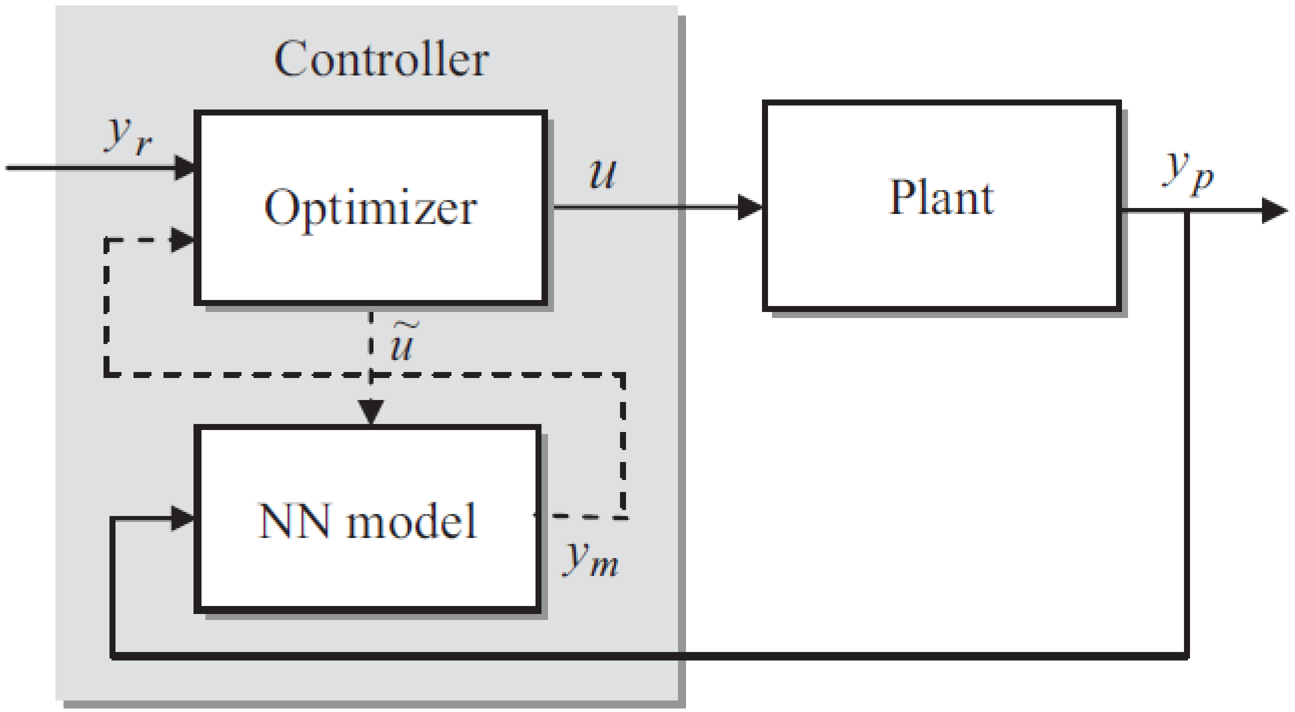


Figure 5.11 Block diagram of model predictive controller

$$J = \sum_{j=N_1}^{N_2} (y_r(t+j) - y_m(t+j))^2 + p \sum_{j=1}^{N_u} (\tilde{u}(t+j-1) - \tilde{u}(t+j-2))^2$$

NN-based MPC

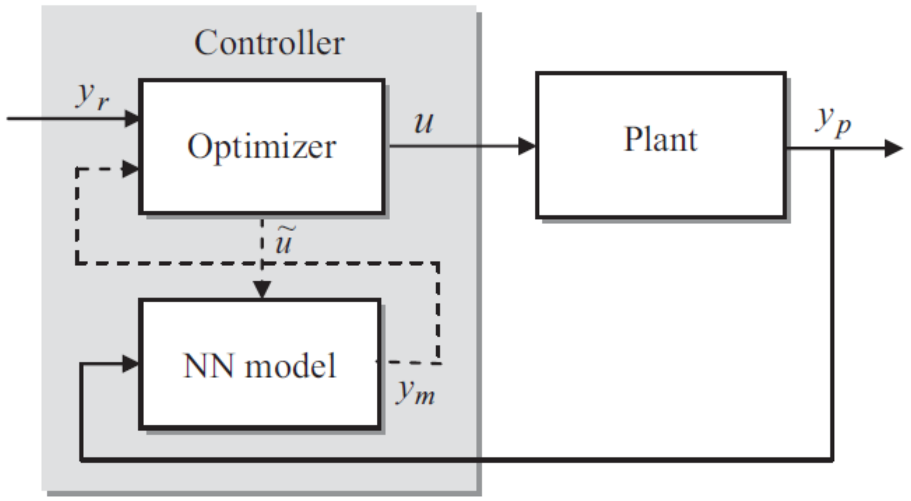
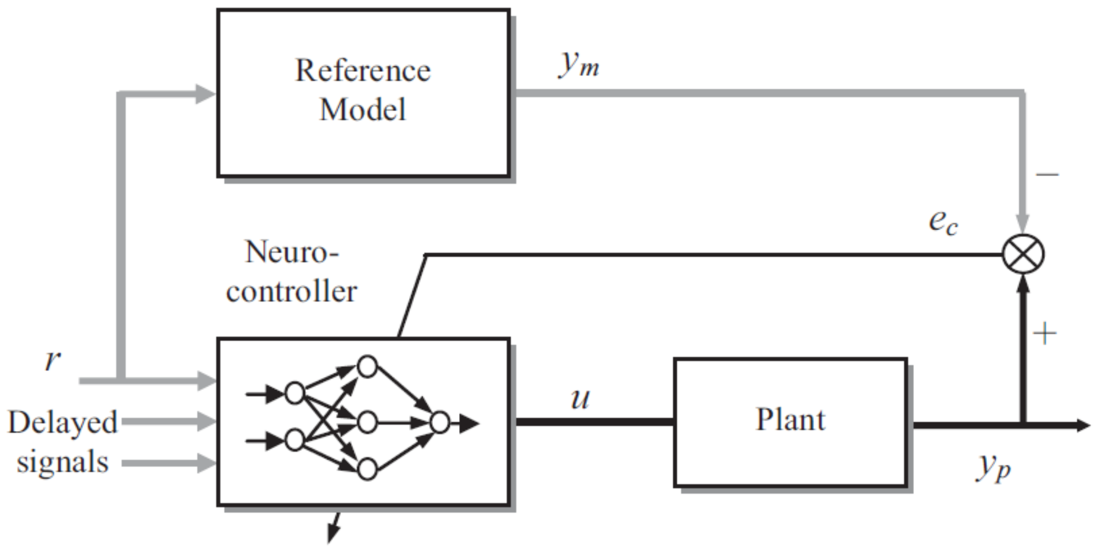


Figure 5.11 Block diagram of model predictive controller

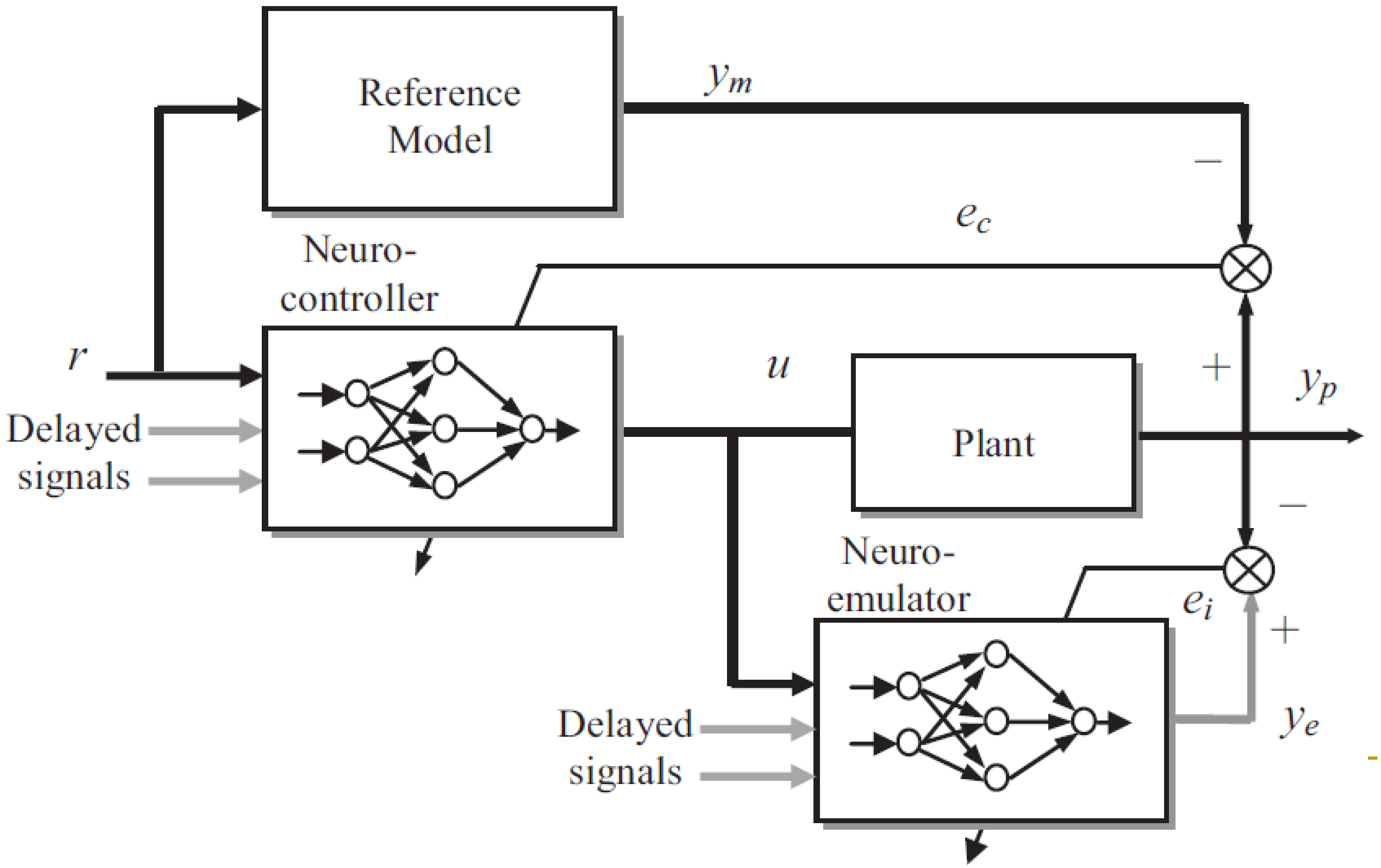
The main disadvantage of MPC is that the controller requires a significant amount of online computation, because an optimization algorithm is performed at each sample time to compute the optimal control input.

Direct Model-reference adaptive control



NN-based MRAC poses stability problems for nonlinear plants.

Indirect Model-reference adaptive control



NN-based self-tunning control

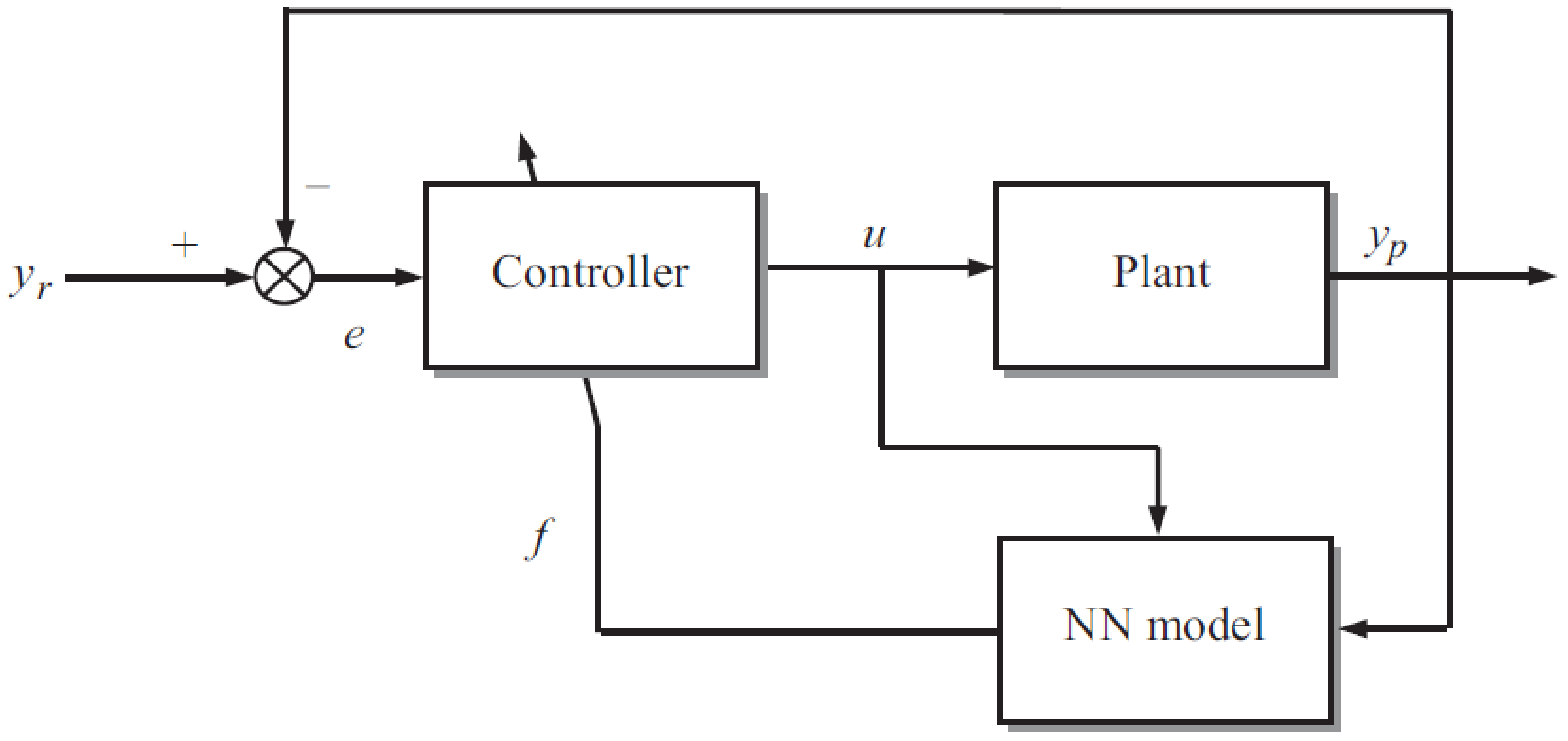


Figure 5.13 NN-based self-tuning control

NN-based self-tunning PID control

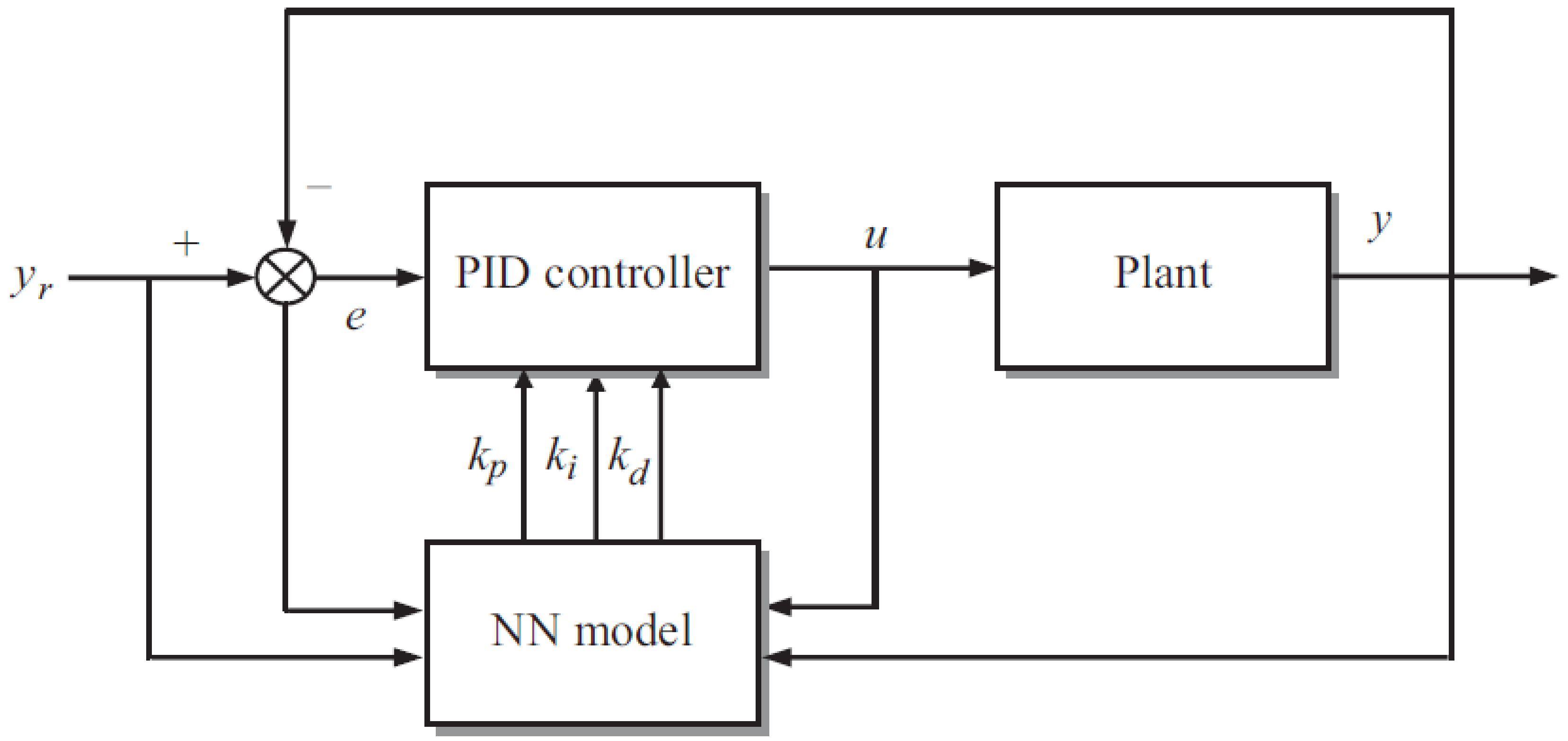


Figure 5.14 NN-based self-tuning PID control

NN-based self-tunning PID control

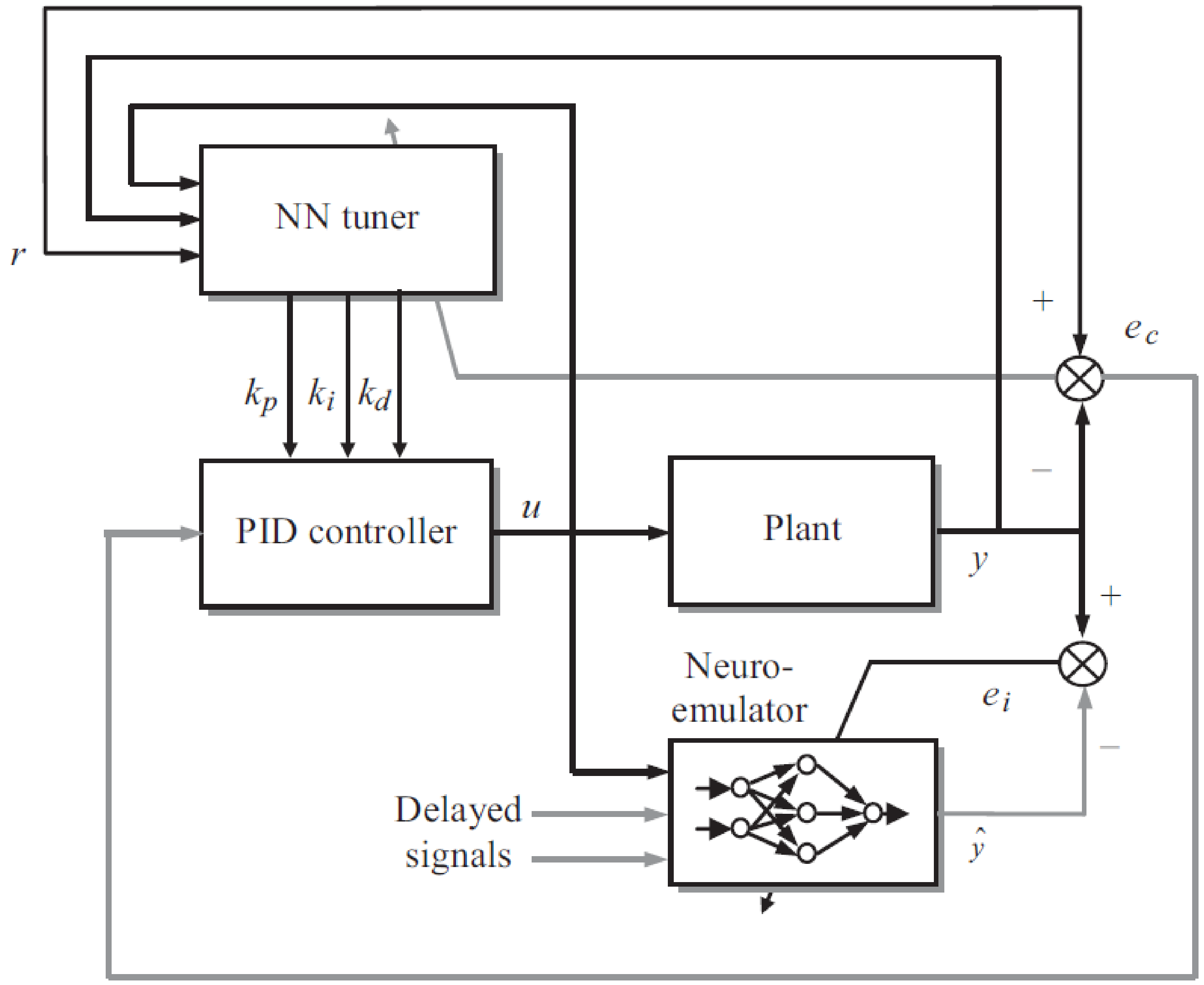


Figure 5.15 NN-based indirect STC for PID controller

NARMA-L2 Control (feedback linearization)

There are two steps involved in the NARMA-L2 control design: systems identification and control design.

$$y(k+1) = \Phi\,[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-n+1)]$$

$$u(k) = \Gamma[y(k), y(k-1), \ldots, y(k-n+1), y_r(k+d), u(k-1), \ldots, u(k-m+1)]$$

$$u(k) = \frac{y_r(k+d) - f\,[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1)]}{g[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1)]}$$

The feedback linearization is a systematic approach for nonlinear control system design. The central idea is to transform nonlinear system dynamics into linear dynamics by cancelling the nonlinearities so that known linear control techniques can be used to design controllers.

NARMA-L2 Control (feedback linearization)

The NARMA-L2 approximate model is parameterized using two neural networks denoted $\hat{f}$ and $\hat{g}$, to be used to identify the system:

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1)] \\ + \hat{g}[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1)] \cdot u(k)$$

NARMA-L2 Control (feedback linearization)

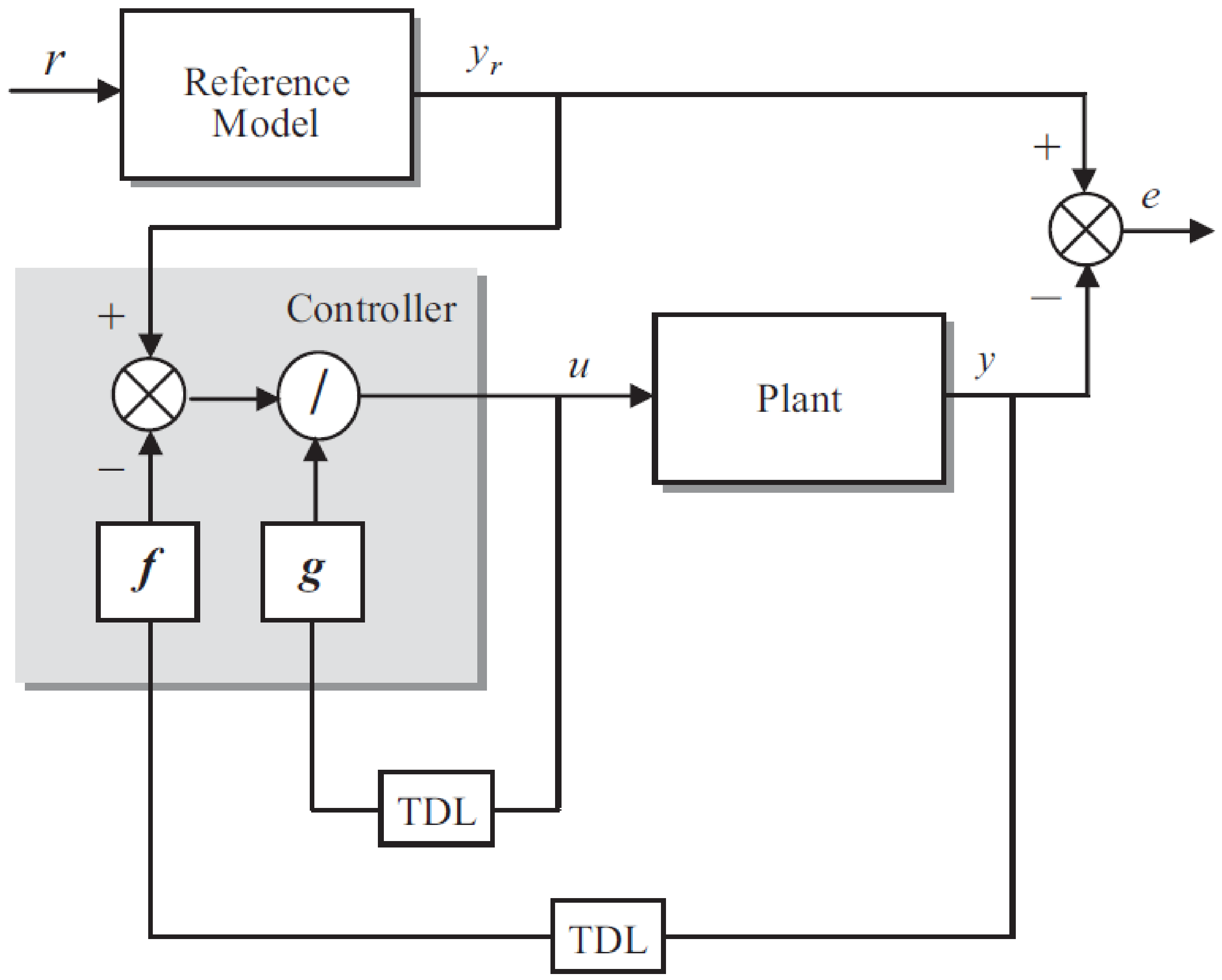


Figure 5.16 Block diagram of NARMA-L2 controller

Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control

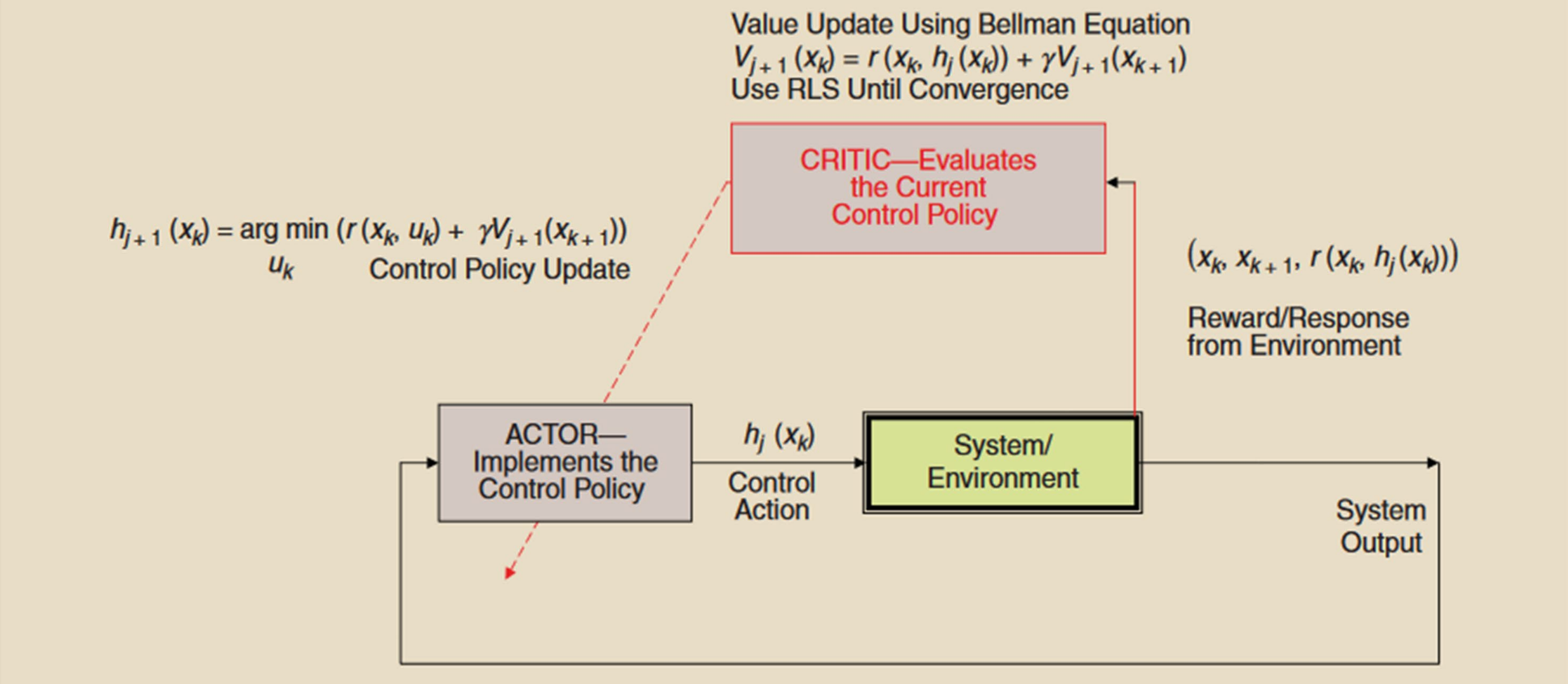

Value Update Using Bellman Equation
$V_{j+1}(x_k) = r(x_k, h_j(x_k)) + \gamma V_{j+1}(x_{k+1})$
Use RLS Until Convergence
CRITIC—Evaluates the Current Control Policy
$h_{j+1}(x_k) = \arg\min_{u_k} (r(x_k, u_k) + \gamma V_{j+1}(x_{k+1}))$
Control Policy Update
$(x_k, x_{k+1}, r(x_k, h_j(x_k)))$
Reward/Response from Environment
ACTOR—Implements the Control Policy
$h_j(x_k)$
Control Action
System/ Environment
System Output

Figure 3. Reinforcement learning using policy iteration. At each time one observes the current state, the next state, and the cost incurred. This is used to update the value estimate. Based on the new value, the action is updated.

http://www.derongliu.org/adp/adp-cdrom/Lewis-CAS-ADP-proof.pdf

Thanks

References

- Chapter 5, Computational intelligence synergies of fuzzy logic, neural networks and evolutionary computing.

- An introduction to the use of neural networks in control systems by m. Hagan, h. Demuth and o. de Jesus.