CHAPTER 1
INTRODUCTION

1.1 The Need for Nonlinear Control in Industrial Processes

The objective of control is to influence the behaviour of systems. Two important control problems are regulation and tracking. Regulation involves keeping system's variables at desired constant values, while tracking involves forcing them to follow prescribed trajectories. The control problem involves determining the values of the manipulated input using all available information to achieve the control objective.

Most physical systems are nonlinear and multivariable. By nature, they have inherent interconnected nonlinearities in their dynamics where the relationship between the input and output variables varies depending on the operating conditions. For example, for a step change on one of the inputs of the system, parameters such as steady-state gains, time constants and time delays for the outputs depend not only on the amplitude of the step but also on the operating values of the rest of the variables.

Many common control problems involve dynamic systems that exhibit nonlinear behaviour. If the nonlinearities are mild or the operating conditions do not change much, then the effect of nonlinearities may not be severe, and linear control techniques are applicable. However, many industrial systems exhibit strong nonlinear behaviour and they may be required to operate over a wide range of operating conditions. When conventional linear controllers are used to control highly nonlinear systems, the controllers must be tuned in a conservative manner in order to avoid unstable behaviour. However, this can result in a serious deterioration of control performance.

1.2 Nonlinear Control Strategies

An exhaustive review of the nonlinear control theory is very ambitious and is beyond the scope of this book. The area is extremely diverse and undergoing continuous development. This section narrows the focus to the more relevant techniques for the purposes of this book, namely gain scheduling and feedback linearisation.

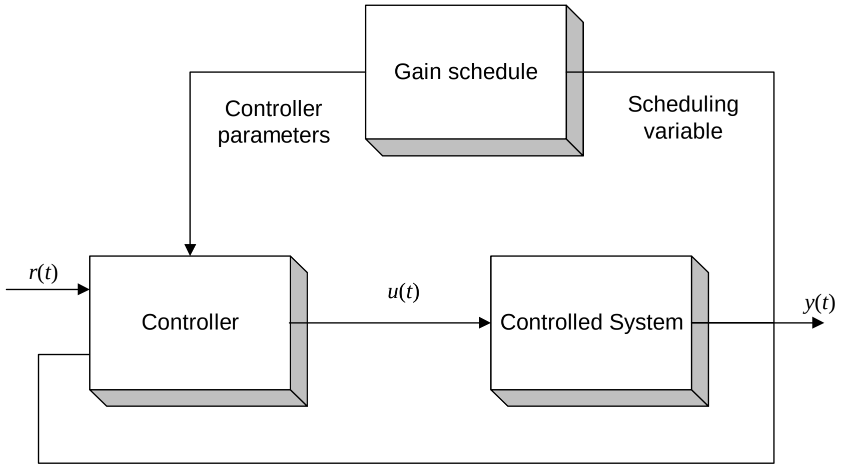

Fig. 1.1. Illustration of gain scheduling

1.2.1 Gain scheduling

Gain scheduling is an engineering approach that has been widely used to compensate for nonlinear process characteristics in single-input single-output (SISO) systems [1, 2, 3]. In gain scheduling, the controller parameters are changed or scheduled following known static nonlinearities of the plant, such that the loop gain is approximately constant. Gain scheduling may be implemented using look-up tables or nonlinear transformations. This technique has been widely applied in different fields, such as flight control and process control. One of the disadvantages of gain scheduling is the lack of a systematic procedure for selecting the scheduling variables. The approach is illustrated in Figure 1.1. The use of nonlinear transformations in gain scheduling is related to the feedback linearization approach [4], which is introduced below.

1.2.2 Feedback linearisation

Jacobian linearisation involves approximating a nonlinear system by a linear one in the vicinity of a reference equilibrium point. In particular, a system described by:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t)) \end{aligned} \tag{1.1}$$

where $f(\cdot,\cdot)$ and $h(\cdot)$ are differentiable nonlinear vector functions, $x \in \Re^n$ is the state, $y \in \Re^p$ is the output and $u \in \Re^m$ is the input, can be locally approximated around an equilibrium point given by (x_s, u_s, y_s) as follows:

$$\begin{aligned} \frac{d\bar{x}(t)}{dt} &= \left[\left.\frac{\partial f}{\partial x}\right|_{x_s,u_s} \right] \bar{x} + \left[\left.\frac{\partial f}{\partial u}\right|_{x_s,u_s} \right] \bar{u} \\ \bar{y}(t) &= \left[\left.\frac{\partial h}{\partial x}\right|_{x_s} \right] \bar{x} \end{aligned} \tag{1.2}$$

where $\bar{x} = x - x_s$, $\bar{u} = u - u_s$, and $\bar{y} = y - y_s$ are deviations from the equilibrium state, input and output, respectively.

In nonlinear control theory, the term *feedback linearisation* has a very different meaning from Jacobian linearisation. Feedback linearisation is perhaps the most important nonlinear control design strategy developed during the last few decades [5]. It has attracted a great deal of research interest resulting in a rigorously formalised field. The main objective of the approach is to algebraically transform nonlinear system dynamics into linear ones by using state feedback and a nonlinear coordinate transformation based on a differential geometric analysis of the system. By eliminating nonlinearities in the system, conventional linear control techniques can be applied. The linearisation is carried out by model-based state transformations and feedback rather than by linear approximations of the dynamics, as used in Jacobian linearisation where the resulting linear model is only locally valid. Differential geometry has proved to be a successful means of analysing and designing nonlinear control systems, equivalently to that of linear algebra and Laplace transform in relation to linear systems. Feedback linearisation is a strong research field with rigorous mathematical formulations [6-10].

To illustrate the basics of feedback linearisation, consider a second order nonlinear SISO system described by the following state equations:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= f_2(x(t)) + g_2(x(t))u(t) \\ y(t) &= x_1(t) \end{aligned} \tag{1.3}$$

where $f_2(\cdot)$ and $g_2(\cdot)$ are known nonlinear functions and $x(t) = [\, x_1(t)\ x_2(t)\,]^T$. Assume that $g_2(x(t)) \neq 0$ and let:

$$u(t) = \frac{v(t) - f_2(x(t))}{g_2(x(t))} \tag{1.4}$$

where $v(t)$ is an artificial input variable. Replacing 1.4 in 1.3:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= f_2(x(t)) + g_2(x(t)) \times \left[\frac{v(t) - f_2(x(t))}{g_2(x(t))} \right] \\ y(t) &= x_1(t) \end{aligned} \tag{1.5}$$

In this way, the original nonlinear system 1.3 has been transformed into the following linear system that uses the artificial input variable $v(t)$:

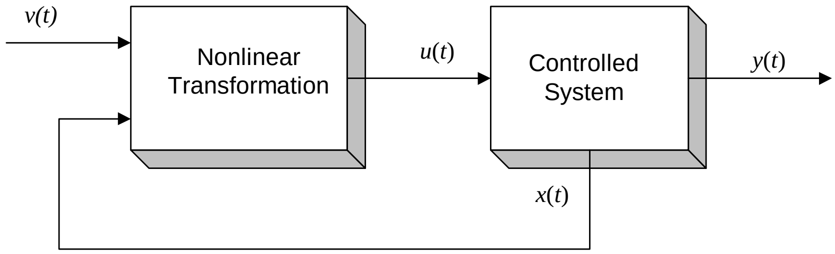

Fig. 1.2. Illustration of feedback linearisation

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= v(t) \\ y(t) &= x_1(t) \end{aligned} \tag{1.6}$$

Note that in order to compute the input $u(t)$ as given by Equation 1.4, information about the state $x(t)$ is required, so that state feedback is being employed as is illustrated in Figure 1.2. This is why the approach is known as *feedback linearisation.*

1.2.3 Feedback linearisation-decoupling

For multivariable nonlinear systems, feedback linearisation may be used not only to eliminate nonlinearities in the input–output relations but also to cancel the interactions between variables. A state feedback linearising law is designed to compensate for these interconnections in order to decompose the multivariable nonlinear system into several single-input single-output linear systems [6, 11, 12].

1.3 Nonlinear System Models: a Key Issue

The information about the nonlinear dynamic behaviour of a system is encapsulated in a dynamic model that often takes the form of a set of nonlinear differential equations plus a measurement model:

$$\begin{aligned} \dot{x} &= f(x(t), u(t)) \\ y &= h(x(t)) \end{aligned} \tag{1.7}$$

where $f : \Re^n \times \Re^m \rightarrow \Re^n$ is a nonlinear mapping, $x \in \Re^n$ is a state vector, $u \in \Re^m$ is a vector of input variables, $y \in \Re^p$ is a vector of measured outputs and $h : \Re^n \rightarrow \Re^p$ is a state to output mapping, and t is a continuous time variable. Many nonlinear control techniques are model-based in that they require the use of this type of dynamic model. A way of obtaining a nonlinear model of the plant is to derive it from physical principles. These models provide a rich insight into the process and are applicable over a wide range of operating conditions. However, physical models are often not available due to the high engineering effort and cost associated with their development and maintenance. Also, due to the complexity of industrial processes, physical models are unsuitable for control purposes. An alternative way for obtaining the required model is to identify it using measured input–output data. The development of techniques of system identification has made possible the synthesis of empirical dynamic models using data measured from the system [13]. In recent years, there has been considerable interest in developing nonlinear dynamic models from input–output data and a variety of model structures and techniques are available [10]. The predominant family of structures for obtaining nonlinear empirical models is known as artificial neural networks, which are inspired by the connectionism of biological neurons [14].

1.4 Neural Networks

Neural networks are distributed, adaptive and generally nonlinear learning machines built from many processing elements, which are often called neurons. Each processing element receives connections from other processing elements. The interconnectivity and number of processing elements define the network architecture. Neural networks are inspired by the connectionism of biological neurons and are capable of approximating arbitrary nonlinear input–output maps.

The field of neural computations has evolved from its neurological roots when the first artificial neural models were proposed [15], to its formalised mathematical foundations [16, 17, 14]. Neural networks can be divided into static and dynamic networks. A static neural model is described by an algebraic equation while a dynamic neural network is represented by a difference or differential equation depending on whether it is based on a discrete or continuous domain, respectively. The most common architectures for static neural networks are the single-layer feedforward networks [14], multilayer feedforward networks or multilayer perceptrons (MLPs) [16], [18], radial basis functions (RBF) [19, 20, 21, 22] and cerebellar model articulation controller (CMAC) networks [23, 24]. When feedback was introduced other relevant architectures were suggested, such as Hopfield networks [25, 26], Bolzmann machines [27, 28], Kohonen self-organising networks [29] and adaptive resonance networks [30].

The best-known type of neural network is the multilayer perceptron. The mathematics of the multilayer perceptron are briefly described below. Figure 1.3 shows a static processing element and associated signals.

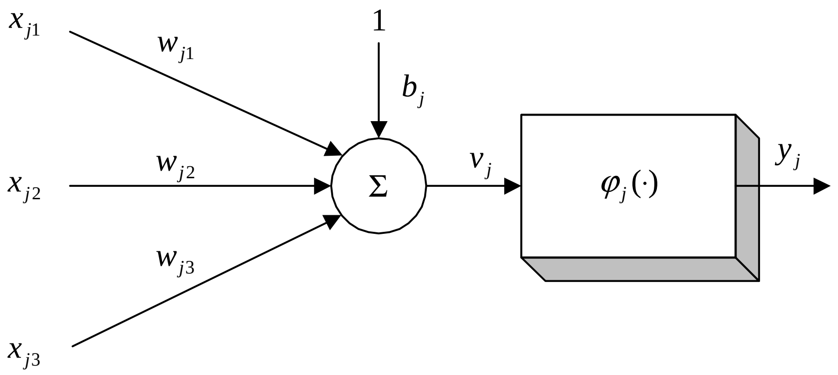

Fig. 1.3. Illustration of a static processing element or neuron

Define the inputs to a processing element j for a given sample n as u_{ji}, $i = 1, \cdots, m_j$. Then the input d_j to the activation of this processing element is given by:

$$d_j(n) = \sum_{i=1}^{m_j} w_{ji} u_{ji}(n) + b_j \tag{1.8}$$

where w_{ji} the weight at the connection between input i and processing element j and b_j is the bias applied to processing element j.

The output of the processing element is the result of passing the scalar value $d_j(n)$ through its activation function $\sigma_j(\cdot)$:

$$z_j(n) = \sigma_j(d_j(n)) \tag{1.9}$$

The actual shape of the activation function σ_j varies between applications. Figure 1.4 shows two different activation functions: the linear activation function and the hyperbolic tangent activation function.

Figure 1.5 shows a typical three-layer perceptron, with three input signals in the input layer, one hidden layer with two nodes and a single output signal.

For the multilayer perceptron shown in Figure 1.5 it is possible to write the output variable as follows:

$$z = \sigma_2(W_2\ \sigma_1(W_1 u + b_1) + b_2) \tag{1.10}$$

where $u \in \Re^3$ is an input vector, $\sigma_1 : \Re^2 \rightarrow \Re^2$ is the activation function of the hidden layer, $\sigma_2 : \Re \rightarrow \Re$ is the activation function of the output layer, $b_1 \in \Re^2$ is a vector associated with the hidden layer, b_2 is the scalar bias associated with the output layer, $W_1 \in \Re^{2\times 3}$ is a weight matrix associated with the hidden layer, $W_2 \in \Re^{1\times 2}$ is a weight matrix associated with the output layer.

A training algorithm is used to adjust the weights of the interconnections according to the training data, which consists of input and output values that

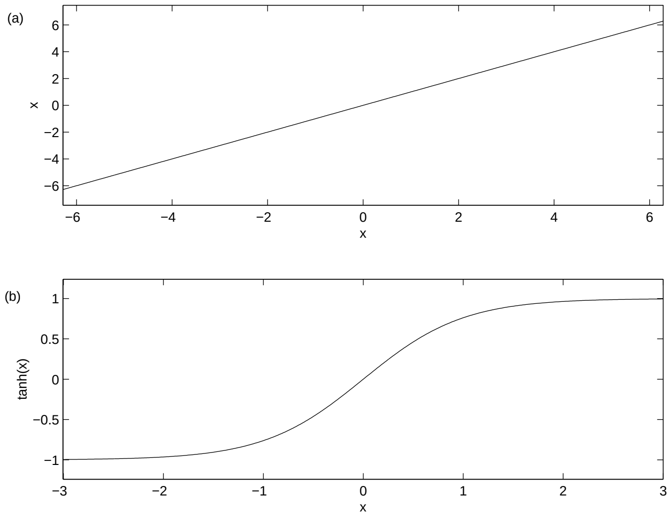

Fig. 1.4. Two common activation functions. (a) Linear $\sigma(x) = x$; (b) hyperbolic tangent $\sigma(x) = \tanh(x)$

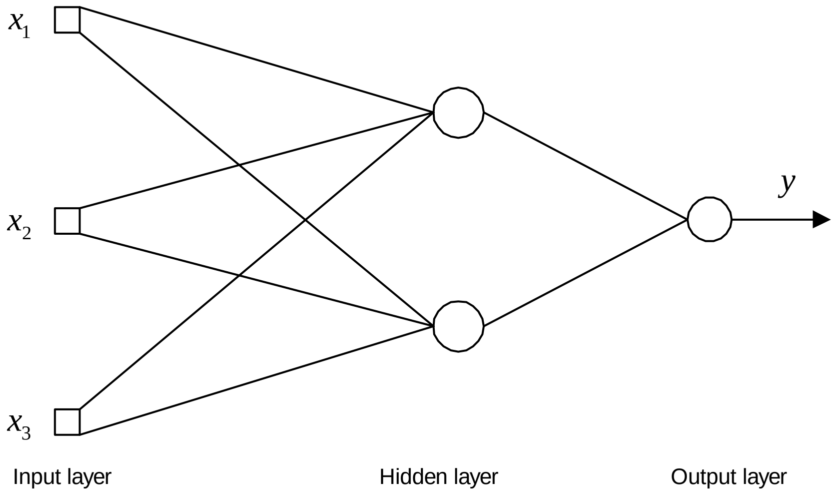

Fig. 1.5. A typical multilayer perceptron

the network is required to learn. The most widely used training methods to determine the weights of a multilayer perceptron is known as the *backpropagation algorithm* [14]. There are several variants of the backpropagation algorithm.

1.5 System Identification

System identification is the experimental approach to modelling dynamical systems. Using system identification techniques, it is possible to find dynamic models of systems based on measured input–output data. System identification involves the following steps [31, 32]:

- **Experiment**. The data needed to identify dynamic models is collected from the system. This involves changing the inputs in an appropriate way to excite the system, and measuring the output history over a period of time, as illustrated in Figure 1.6. The result of an identification experiment is a set of M discrete data points, $Z_M = \{\ [y(t_k),\ u(t_k)\,],\ k = 1, \cdots, M\ \}$, where y and u are vectors of measured and input variables, respectively. The resulting data set may need some pre-processing (e.g. filtering) before it can be used for parameter estimation.
- **Model structure selection**. A model structure is a set of possible models with a number of free parameters. The choice of model structure is dependent on the purpose of the model, but the designer often has to choose between linear and nonlinear modes, input–output or state space descriptions, continuous or discrete-time models, etc.
- **Parameter estimation**. The free parameters of a model are estimated using an optimisation procedure that is typically aimed at minimising the differences between the measured outputs and the model outputs. Depending on the model structure used, simple non-iterative techniques like the least squares method may be used, although for nonlinear model structures iterative procedures are required.
- **Model validation**. This step involves evaluating the model to see if it satisfies the requirements for acceptance, which are in turn connected to the purpose of the model. A data set different from the one used for parameter estimation is typically employed for validating the model. If the model is not acceptable, then it may be necessary to repeat some of the above steps.

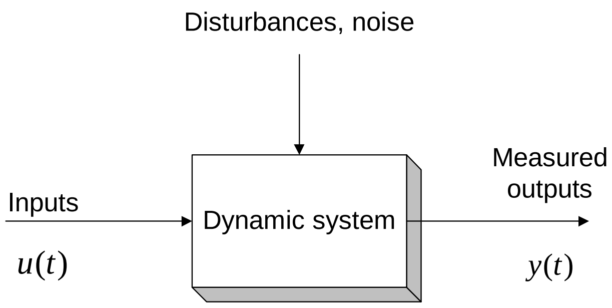

Fig. 1.6. The system identification experiment

1.6 Static Neural Networks for Identification and Control

Given the ability of multilayer perceptrons to approximate arbitrary continuous functions, neural networks have found wide applications both for function approximation and pattern recognition. Since the seminal work by Narendra and Parthasarathy [33], in which the use of neural networks was proposed for identification and control, a great deal of progress has been made in this field. The purpose of this section is to introduce a number of basic ideas on the use of static neural networks for identification and control.

General nonlinear discrete-time input–output models can be described as follows[1]:

$$y(t) = \hat{y}(t|\theta) + e(t) = g(\sigma(t,\theta),\theta) + e(t) \tag{1.11}$$

where $y \in \Re^p$ is the output of the system, $\hat{y} \in \Re^p$ is the output of the model, $g : \Re^{n_\sigma} \times \Re^{n_\theta} \rightarrow \Re^p$ is a nonlinear mapping, $\sigma \in \Re^{n_\sigma}$ is known as the regression vector (which depends on past input–output information), $\theta \in \Re^{n_\theta}$ is the parameter vector, and $e \in \Re^p$ is known as the model residual.

A common nonlinear discrete-time structure, which is illustrated in Figure 1.7, is known as nonlinear auto-regressive with exogenous input (NARX). For this structure, the regression vector is given by:

$$\sigma(t,\theta) = [y(t-1), \cdots, y(t-n_a), u(t-d), \cdots, u(t-d-n_b+1)]^T \tag{1.12}$$

where n_a is the number of past outputs, n_b is the number of past inputs and d an integer representing pure delay. The values of n_a, n_b, and d determine the size of the regression vector and the external structure of the model.

A multilayer perceptron is often used with the NARX structure to provide the static mapping g between the model input (the regression vector $\sigma(t,\theta)$) and the model output $\hat{y}(t|\theta)$. In that case, there is an internal structural choice in terms of the number of hidden layers and the number of neurons in each hidden layer. Once the structure is fixed, the vector of free parameters θ contains all the weights and biases associated with the multilayer perceptron. In this way, a static neural network can be used for modelling a discrete-time dynamical system.

Many strategies have been proposed to use neural networks for control. A recent review was provided by Agarwal [34]. Neural network based control may be classified as being *direct* or *indirect*, depending on the role played by neural networks in the control strategy. Direct neural network based control implies the use of a neural network as the controller. Indirect neural network based control involves the use of a neural network as an aid for modelling, control action or supervisory action.

[1] Notice that in the case of discrete-time models like Equation 1.11, variable t represents an integer time index.

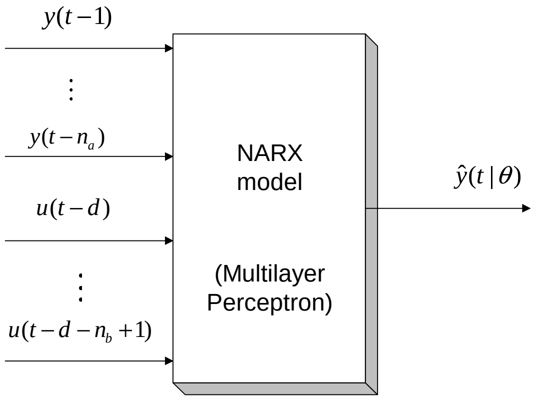

Fig. 1.7. The NARX structure

Direct inverse control. One of the first and simplest strategies proposed for direct neural network based control was the use of a neural network to approximate the inverse mapping of a system. Assume for simplicity that the system is SISO. If the controlled system can be described by:

$$y(t+1) = g(y(t), \cdots, y(t-n_a+1), u(t), \cdots, u(t-n_b+1)) \tag{1.13}$$

then, the inverse model of the system computes the required input $u(t)$ to achieve a desired output $y^{(d)}(t+1)$ at the next time step:

$$u(t) = \hat{g}^{-1}(y^{(d)}(t+1), y(t), \cdots, y(t-n_a+1), u(t), \cdots, u(t-n_b+1)) \tag{1.14}$$

where the mapping $\hat{g}^{-1}$ may be obtained by training a static neural network, such as a multilayer perceptron [35]. This approach is illustrated in Figure 1.8. Direct inverse control is simple and intuitive, but problems occur when the inverse models are not well damped or unstable. Also, this approach is very sensitive to noise and disturbances.

Internal model control (IMC). In this direct strategy, the difference e between the output of a neural network forward model $\hat{y}$ and the output of the plant y is used as feedback signal, while a neural network approximating the inverse of the system is placed in the forward path [36]. If the forward model is perfect, the error signal will be zero and the control system will operate as if it was under direct inverse control. This approach is illustrated in Figure 1.9. The forward model can be a multilayer perceptron based NARX model.

1.7 Dynamic Neural Networks for Nonlinear Identification

The introduction of feedback into a feedforward neural network architecture produces a state space dynamic model. A dynamic recurrent neural network

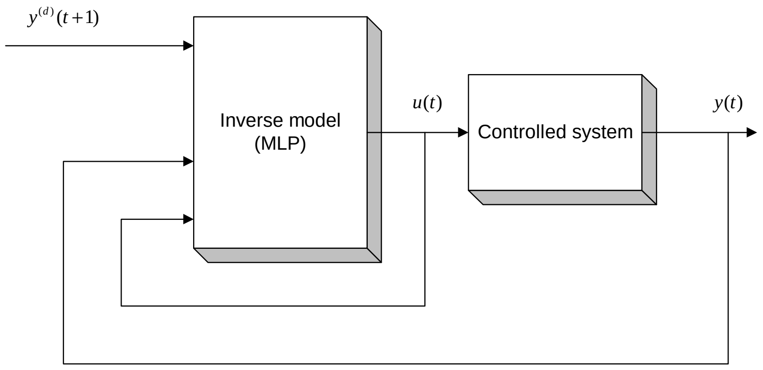

Fig. 1.8. Direct inverse control

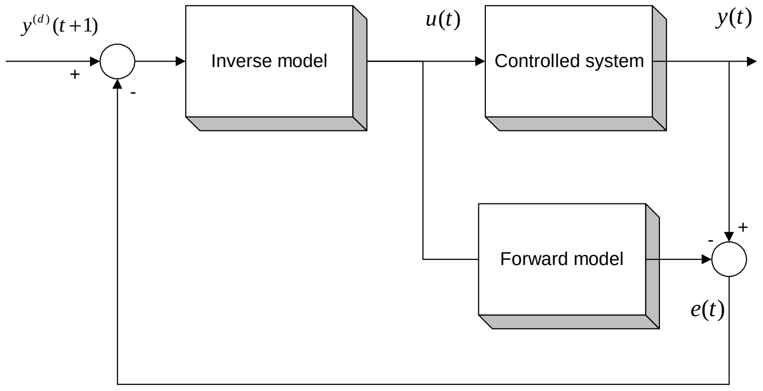

Fig. 1.9. Internal model control

(DRNN), or simply a dynamic neural network (DNN), is a collection of dynamic neurons partially interconnected to a function of their own output. Such networks can be represented by a state space neural model of the form

$$\begin{aligned} \dot{x} &= -\beta x + \omega\sigma(x) + \gamma u \\ y_n &= C_n x \end{aligned} \tag{1.15}$$

where x are coordinates on $\Re^N, \omega \in \Re^{N\times N}$ $, \sigma(x) = [\sigma(x_1), \ldots, \sigma(x_N)]^T$, $\gamma \in \Re^{N\times p}$, $u \in \Re^p, C_n = [I_{p\times p}\emptyset_{p\times(N-p)}]$ and $\beta \in \Re^{N\times N}$ is a diagonal matrix with diagonal elements $\{\beta_1, \ldots, \beta_N\}$.

The use of a dynamic neural network model for system identification purposes seems now a straightforward task. Introducing inherent dynamic capabilities in the neurons has made possible that an array of these elements be used as generic model for system identification.

1.8 Input–Output Linearisation-Decoupling and the Use of Dynamic Neural Networks

Many industrial processes are multivariable in the sense that they have several inputs and outputs. In addition to nonlinear behaviour, such processes often exhibit interactions, in the sense that a change in an input variable may originate changes in several output variables. In a multivariable system, in addition to linearising the behaviour from inputs to outputs, it is possible to use feedback in order to eliminate or weaken the interactions and reduce the system, at least from an input–output perspective, to several independent single-input single-output systems. This synthesis method is known as the input–output linearisation-decoupling scheme and it requires a model of the system. Instead of using a process model derived from physical considerations, the control strategies proposed in this book are based on dynamic neural network models as given in Equation 1.15. These models approximate the behaviour of the system and are trained using input–output data. A feedback linearising-decoupling law is synthesised for the neural model and applied on the MIMO plant. Once the dynamics are linearised and decoupled to a certain extent, the system is immersed in an outer loop with a standard multivariable Proportional+Integral (PI) scheme. The use of neural networks for feedback linearisation started recently and similar general motivations appear to underlie recent approaches where the existence of a neural network model encouraged the use of feedback linearising techniques. Feedback linearisation of SISO systems has been carried out by means of feedforward networks [37, 38], while other work has used dynamic networks from a robust control perspective [39]. Input–output linearisation of multivariable processes within an inverse model control framework has also been carried out [40, 41]. Also for SISO systems, input–output linearisation techniques have been proposed using dynamic neural networks [42] and CMAC neural networks [43]. Later work suggested the use of discrete time neural models for feedback linearisation of chemical processes [44]. In recent work, an adaptive linearising feedback technique has been developed for induction motor control based on dynamic neural networks [45].

1.9 Potential applications

Model predictive control (MPC) is a technique that periodically uses a model of the controlled system to calculate the control action based on optimal input–output predictions over a time horizon. This technique has enjoyed remarkable industrial success since the first reported industrial implementations in the late 1970s [46]. Key features contributing to its success are that multivariable systems and constraints can be accommodated effectively in the control problem, and the use of empirical models which can be built from measured input–output data. The traditional application areas were refining and petrochemicals, but significant growth in areas such as chemicals, pulp and paper, food processing, aerospace and automotive industries has been noticed in the last few years [47]. To date, most commercial predictive controllers employ linear empirical models. However, nonlinear empirical models are starting to be used in some commercial predictive control packages [48]. The resulting approach is known as nonlinear predictive control. The control problem is analogous to the linear version of MPC except that a nonlinear dynamic model is used. It can be expressed as determining the control actions by solving a nonlinear programming problem at each sampling interval in order to minimise the error between the outputs and their setpoints over the optimisation horizon [49, 50]. One of the drawbacks of this approach is the great computational burden associated with the on–line solution of the nonlinear programming problem [51]. Feedback linearisation techniques based on empirical nonlinear models can potentially be used to enable MPC techniques to consider nonlinearities, while at the same time have the computational advantage of a linear model, as preliminary research has shown [52, 53, 54, 55].