Chapter 8

Modal Neural Networks: A Naval Application

8.1 Introduction

The purpose of this chapter is to introduce a new neural-network arrangement suitable for multiple-input, multiple-output (MIMO) control applications. In Figure 5-8 we implemented a multiple-input-single-output (MISO) neural network controller to command the deflections of two stabilizers, δ_p and δ_s, in order to reduce the rolling motion of a particular ship. Although we used two fins for this purpose, the controller was single output because the motion of the fins was antisymmetrical. However, for ships with structural asymmetries or asymmetric load distributions, or for ships navigating in seas coming at an angle other than zero, independent motions of the fins may yield better roll reduction. This would lead to a MIMO system.

If independent motions of the pair of fins were to be used to reduce the rolling motion of the ship, the moment-matching procedure, explained in section 5-8, would require a more involved analysis to be implemented. When we tried to implement, for this problem, the gradient-search training procedure discussed in Chapter 7 to train a two-output neural network controller such as the one shown in Figure 8-1, we found some difficulties. The main inconvenience was related to the weight initialization. To have fast convergence and to avoid local minima required the weight initialization, at least, to generate an approximately antisymmetric initial motion of both stabilizers. That demand presupposed some foreknowledge of the control law for the controller to be designed. This does not imply that the training algorithm would not converge to an

effective control law, but that it could take large computational efforts, and local minima could be encountered during the training process. On the other hand, the choice for a particular weight initialization is associated with some intuitive foreknowledge of the control law. Such insight may not be available in many instances.

In the next section, a novel neural-network architecture is discussed, which can naturally be implemented for MIMO systems, and which does not require any foresight of the control-system behavior.

8.2 The Modal Neuron and Modal Neural Networks

In this section, we introduce an effective neural network architecture suitable for MIMO systems. We describe this in the context of control in naval applications, but it can be applied to other types of MIMO systems.

The basis of this approach is to take into account a complete set of modes of the control output of the system. By a complete set of modes we mean the minimum number of simple control configurations that can be linearly combined to form any complex or final control configuration. For example, in Figure 8-2 we show a set of four modes associated with controlling the roll of ships utilizing a pair of fins and a rudder. In the first mode, the rudder is deflected to port and the angles of attack of the fins are increased by the same amount. In the second mode, the rudder is deflected as in the first mode, but the fins are deflected antisymmetrically the same amount. In the third mode, the rudder is deflected to starboard by the same amount it had been deflected to port in the first mode, and the fins are deflected as they were in the first mode. In the fourth mode the rudder is deflected as in the third mode and the fins are deflected antisymmetrically as in the second mode.

The basic component of a standard neural network (SNN) is the artificial neuron. The basic component of a *modal neural network* (MNN) is a set of ν neurons called a *modal neuron*, where ν is the total number of output control variables. The neurons of each modal neuron are called *sub-neurons.* The output of each sub-neuron *n* of any modal neuron is the contribution of this modal neuron to the corresponding component of the

output of the system, ${}^{n}\boldsymbol{\delta}$. Different modal neurons can be set in parallel to define the first layer of a MNN. The input to the first layer is the input vector **a**, of dimension η. The modes of the system are described by μ vectors of dimension ν, which are denoted by ${}^{m}\mathbf{D}$, where $m = 1 \ldots \mu$. Each set of vectors ${}^{m}\mathbf{D}$ can be arranged in a $\nu \times \mu$ modal matrix $\mathbf{D} = {}^{nm}d$, where $n = 1 \ldots \nu$, which defines the shape of the output modes. The elements of **D** are arbitrarily given by plus and minus ones, according to the basic shapes of the modes. In Figure 8-2 a four-mode, three-output system is represented schematically. The corresponding modal matrix is given by

$$ {}^{nm}d = \left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{8.1} $$

In Figure 8-3, the two-mode, double-output control system for a single pair of fins, is represented. The corresponding modal matrix is given by

$$ {}^{nm}d = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{8.2} $$

Next we introduce the architecture of a modal neuron in the first layer. First-layer modal neurons capture the modal nature of the entire system. In a SNN, a set of η weights is assigned to each neuron in such a way that each weight of the set multiplies a different element of the input vector **a**. In a MNN, μ sets of η weights per set are assigned to each modal neuron of the first layer, where μ is the number of modes of the system. We refer to these sets as the modal weight sets. Taking into account all modal neurons of the first layer, we can define μ modal weight matrices. We express them as ${}^{m}\mathbf{W}^{I} = {}^{m}w_{ji}^{I}$, where *m* denotes the mode number, *j* is the modal neuron number for each modal neuron of the first layer, and *i* accounts for each component of the input vector **a**.

As in previous chapters, roman superscripts account for layer number. The first layer contains σ modal neurons. For the sake of clarity the ranges of *i*, *j*, *m* and *n* are summarized as follows:

$$\begin{array}{rcll} i & = & 1\ldots\eta & \text{where} \quad \eta = \text{input-vector dimension} \\ j & = & 1\ldots\sigma & \text{where} \quad \sigma = \text{number of modal neurons of layer } I \\ m & = & 1\ldots\mu & \text{where} \quad \mu = \text{number of modes} \\ n & = & 1\ldots\nu & \text{where} \quad \nu = \text{number of sub-neurons in each modal neuron} \end{array} \tag{8.3}$$

The modal weight matrices, ${}^{m}\mathbf{W}^{I}$, are multiplied by the modal matrix **D** according to:

$${}^{n}\mathbf{\Omega} = {}^{n}\hat{\omega}^{I}_{ji} = \sum_{m=1}^{\mu} {}^{nm}d \;\; {}^{m}w^{I}_{ji} \qquad \text{for} \quad \begin{array}{l} i = 1\ldots\eta \\ j = 1\ldots\mu \\ n = 1\ldots\nu \end{array} \tag{8.4}$$

This yields the *active modal weight matrices*, ${}^{n}\mathbf{\Omega}$**.** Though these matrices are only defined for the first layer, they capture the modal nature of the entire MNN.

Next we multiply the input vector **a** by the active modal weight matrices according to:

$${}^{n}v^{I}_{j} = \sum_{i=1}^{\eta} {}^{n}\hat{\omega}^{I}_{ji} \; a_i \qquad \text{for} \quad \begin{array}{l} j = 1\ldots\mu \\ \\ n = 1\ldots\nu \end{array} \tag{8.5}$$

For each modal neuron *j*, the outputs from the summation given by equation (8.5) become the inputs for the squashing functions of each sub-neuron of each modal neuron. We express this as:

$$ {}^{n}y_j^I = \mathbf{\Phi}^I\left({}^{n}v_j^I\right) \quad \text{for} \quad \begin{matrix} j = 1\ldots\sigma \\ n = 1\ldots\nu \end{matrix} \tag{8.6} $$

where $\mathbf{\Phi}^I = {}^{n}\varphi_j^I$ are the squashing functions for the first layer. Each modal neuron j has ν squashing functions, one for each subneuron. In Figure 8-4 we illustrate a first-layer modal neuron for a four-mode, three-output system.

A new set of modal neurons can be connected to the first layer of neurons to generate a second layer. In a MNN the output from every modal neuron j is a vector of ν sub-outputs, $\mathbf{b}_j$. We connect each component of every output vector $\mathbf{b}_j$ from the modal neurons of the first layer with the corresponding inputs of the sub-neurons of each modal neuron of the second layer. In Figure 8-5 we show a two-layer MNN with two modes and two outputs. There we can see the connection between the outputs from the subneurons from the first layer with the corresponding inputs of the subneurons of the single modal neuron of the second layer.

For a second layer of modal neurons we define ν modal weight matrices, ${}^{n}\mathbf{W}^{II}$, for $n = 1\ldots\nu$. Each matrix n has a dimension $\kappa\times\sigma$, where κ is the number of modal neurons of the second layer. The outputs from the first layer are multiplied by the corresponding modal weight matrix n according to

$$ {}^{n}v_k^{II} = \sum_{j=1}^{\sigma} {}^{n}w_{kj}^{II}\ {}^{n}y_j^I \quad \text{for} \quad \begin{matrix} k = 1\ldots\kappa \\ n = 1\ldots\nu \end{matrix} \tag{8.7} $$

For each mode n, the summation given by equation (8.7) becomes the input for the n squashing functions of each modal neuron of the second layer. We express this as:

$$ {}^{n}y_{l}^{II} = \mathbf{\Phi}^{II}\left({}^{n}v_{l}^{II}\right) \qquad \text{for} \quad \begin{matrix} l = 1\ldots\kappa \\ n = 1\ldots\nu \end{matrix} \tag{8.8} $$

where $\mathbf{\Phi}^{II} = {}^{n}\varphi_{l}^{II}$ are the squashing functions for the second layer. For a two-layer modal neural network, $\kappa = 1$ and equation (8.8) can be simplified as

$$ {}^{n}y^{II} = \mathbf{\Phi}^{II}\left({}^{n}v^{II}\right) \quad \text{for} \quad n = 1\ldots\nu \tag{8.9} $$

where each component *n* of ${}^{n}y^{II}$ yields the (multiple) output of the system.

We can follow the same procedure for any number of layers we choose for the architecture of our MNN. The last layer must always contain only one modal neuron (see Figure 8-5). From what we have said so far, we see that the main difference between an ordinary neural network and a modal neural network resides in the basic building block, the *modal neuron*, and the particular care that must be taken in connecting the modal neurons from subsequent layers.

8.3 Results

In this section we present an example of an application of a MNN. The purpose is to control, at the same time, the rolling motion and pitch motion of a ship by means of a single pair of fins. We use basically the same model of a ship discussed in Chapters 2 and 5. Again the hull geometry and the inertia components are typical of a CG47 cruiser of the US Navy. The ship is moving at 20 knots and the seas are random. To have pitch controllability we place the fins 10% aft the bow. The fins we used are somewhat bigger than the fins we used for rolling reduction in Chapters 2 and 5. Their root chord is 1.7% of the length of the ship and their aspect ratio is 3.2. As in Chapters 5, the ratio of the tip-to-root chords is 0.8, and only the leading edge is tapered. Two reasons led us to

place the fins forward instead of aft: 1) the fins seemed to be less effective aft than forward; 2) the curvature of the bottom of the ship we use for our example (CG47) is significantly upwards in the aft portions of the ship. Therefore, the further aft we place the fins the higher the risk they will come out of the water due to the pitching motion. This could lead to instabilities, undesired vibrations and, in a real life situation, the fins could be damaged when slamming back into the water (see *e.g.* Benford [3]).

To produce the rolling and pitching control we used a two-layer, two-mode, two-output MNN, such as the one described in Figure 8-5. The two modes of the controller are depicted in Figure 8-3. Mode 1 accounts for the symmetrical motion of the fins, and is associated with pitching control. Mode 2 accounts for the antisymmetrical motion of the fins, and is associated with rolling control. To train the neural network we used the moment-matching procedure introduced in Chapters 5. The error signal for mode 1 is the pitching acceleration, $\dot{\omega}_y$, whereas the error signal for mode 2 is the rolling acceleration, $\dot{\omega}_x$. We want to emphasize that the main goal of this chapter is to introduce MNN's, not how to train them. Such as any SNN, a MNN could be trained with a wide variety of different methods (genetic algorithms (*e.g.*, Goldberg [24]), simulated annealing (*e.g.*, Jang [35]), *etc*).

We trained the MNN over 10 epochs of 3,000 time steps each. To show our results we first define a rolling ϕ-performance, ϕ_p; a pitching θ-performance, θ_p; and a heaving z-performance, z_p; as:

$$\phi_P = \frac{\sum_{k=1}^{K} |\phi(k)|_e}{\sum_{k=1}^{K} |\phi(k)|_0} \tag{8.10}$$

$$\theta_P = \frac{\sum_{k=1}^{K} |\theta(k)|_e}{\sum_{k=1}^{K} |\theta(k)|_0} \tag{8.11}$$

$$z_P = \frac{\sum_{k=1}^{K} |z(k)|_e}{\sum_{k=1}^{K} |z(k)|_0} \tag{8.12}$$

where, following the same conventions of Chapters 5, ϕ is the first Euler angle (roll-like motion), θ is the second Euler angle (pitch-like motion), z is the vertical displacement of the ship in a ground-fixed frame of reference, k is the time step, K is the total number of time steps of each epoch, the subscript e denotes the current epoch and the subscript 0 denotes the initial epoch with no fins. Similarly we can define the ω_{x-}, $\dot{\omega}_{x-}$, ω_{y-}, $\dot{\omega}_{y-}$, $\dot{z}_-$ and $\ddot{z}_-$ performances.

In Figure 8-6 we show the roll reduction we obtained with this configuration. The ϕ-performance for this case was 62%, the ω_{x-} performance 58% and the $\dot{\omega}_{x-}$ performance 76%. In the same figure we can see a good reduction with respect to the passive case (fixed fins).

Analogously for the pitch motion we obtained a θ-performance of 88%, a ω_{y-} performance of 83%, and a $\dot{\omega}_{y-}$ performance of 83%. This is shown in Figure 8-7. From this figure we can see that the reduction with respect to the passive case is not very large. However, by reducing pitching we also reduced heaving. We show this in Figure 8-8 where the z-performance is of 90%, the $\dot{z}_-$ performance of %88 and the $\ddot{z}_-$ performance of 81.5%. Unlike the pitching reduction, we see a significant reduction of heaving between active and passive fins.

Though the training process seemed to provide adequate training, the reductions observed were not very large for the pitching motion. We also explored pitching reduction by using a single-pitching-mode SNN; the corresponding pitching and heaving reductions were very similar to the two-mode case. It seems that controlling pitching is not an easy task. Most likely, a single pair of fins is not the most efficient arrangement for an efficient pitch-roll control system. However the fact that the average wave-length is in general larger than the ship length, or at least of the same order of magnitude, have

the effect of forcing the ship to follow the shape of the waves thus leaving little margin for pitching reduction.

From a technical point of view, the aspect ratio of the fins is rather large compared to the average aspect ratios of fins installed on real ships. For a real-life application it should also be considered the feasibility and efficiency in terms of the materials to build such fins and the power demand to be provided to the control actuators. However, the main purpose of this example was not to show an efficient or a feasible control system, but to give an example of application of a MNN.

8.4 Concluding Remarks

In this chapter we have introduced a new neural network approach very suitable for MIMO control, and in general, for MIMO systems. Training a modal neural network can speed up the learning process and reduce the chances to encounter local minima. We implemented a two-mode, two-output MNN for a roll-pitching control system for a pair of fins. We obtained a good rolling reduction. Though the pitching reduction was less important a significant heaving reduction was generated. For the particular problem we discussed, the training speed was comparable to that for an analogous MISO system (see Chapters 5).

8.5 Complementary Discussion: The Backpropagation Technique in Modal Neural Networks.

Here we discuss the training of a two-layer MNN by means of the backpropagation technique.

For a two-layer modal neural network the modal weights are updated at every time-step according to:

Layer *I*:

$$ {}^{m}w_{ji}^{I}(k) = {}^{m}w_{ji}^{I}(k-1) + \Delta {}^{m}w_{ji}^{I}(k) \tag{8.13} $$

Layer *II*:

$$ {}^{n}w_{j}^{II}(k) = {}^{n}w_{j}^{II}(k-1) + \Delta {}^{n}w_{j}^{II}(k) \tag{8.14} $$

where

$$ \begin{array}{llllll} i & = & 1\ldots\eta & \text{where} & \eta & = \text{input-vector dimension} \\ j & = & 1\ldots\sigma & \text{where} & \sigma & = \text{number of modal neurons of layer } I \\ m & = & 1\ldots\mu & \text{where} & \mu & = \text{number of modes} \\ n & = & 1\ldots\nu & \text{where} & \nu & = \text{number of sub-neurons of each modal neuron} \end{array} \tag{8.15} $$

In the following, we describe how to estimate $\Delta {}^{m}w_{ji}^{I}(k)$ and $\Delta {}^{n}w_{j}^{II}(k)$.

We start by defining an error ${}^{mn}e = {}^{mn}e(k)$ as the difference between the desired output from the MNN, ${}^{mn}o_d = {}^{mn}o_d(k)$, and the current output from the MNN, ${}^{mn}o_c = {}^{mn}o_c(k)$, for each mode *m*. We express this as:

$$ {}^{mn}e(k) = {}^{mn}o_d(k) - {}^{mn}o_c(k) \qquad \text{for} \qquad m = 1 \ldots \mu \tag{8.16} $$

For the moment-matching procedure, μ errors, ${}^{m}\varepsilon(k)$, are given proportional to μ different accelerations. The corresponding matrix ${}^{mn}e(k)$ can be obtained as

$$ {}^{mn}e(k) = {}^{m}\varepsilon(k) \otimes {}^{nm}d \qquad \text{for} \qquad n = 1 \ldots \nu \tag{8.17} $$

where the operator ⊗ means that each element *m* of ${}^{m}\varepsilon$ is multiplied by the corresponding elements *m* of ${}^{mn}d$ $\left(e.g., \text{for } m = 2, \text{ then } {}^{2n}e(k) = {}^{2}\varepsilon(k) \;\; {}^{n2}d\right)$. In what follows we usually omit the argument *k*.

For each mode *m*, a measure of the performance ${}^{m}\Theta$ is defined as:

$$ {}^{m}\Theta = \frac{1}{2} \sum_{n=1}^{\nu} {}^{nm}e^2 \tag{8.18} $$

The procedure is analogous to that described in Chapter 4. The backpropagation algorithm estimates the gradient in the weight space of the function ${}^{m}\Theta$, and seeks to minimize it. For a two-layer modal neural network, ${}^{mn}y^{II} = {}^{mn}o_c$ represents the contribution to the total output of the system ${}^{n}y^{II} = {}^{n}o_c$ from each mode *m*. Thus,

$$ {}^{n}y^{II} = \sum_{m=1}^{\mu} {}^{mn}y^{II} \tag{8.19} $$

For the output layer, we define the *sensitivity factors* as

$$\frac{\partial\ ^{m}\Theta}{\partial\ ^{n}w_{j}^{II}} = \frac{\partial\ ^{m}\Theta}{\partial\ ^{mn}e}\frac{\partial\ ^{mn}e}{\partial\ ^{mn}y^{II}}\frac{\partial\ ^{mn}y^{II}}{\partial\ ^{n}y^{II}}\frac{\partial\ ^{n}y^{II}}{\partial\ ^{n}v^{II}}\frac{\partial\ ^{n}v^{II}}{\partial\ ^{n}w_{j}^{II}} \tag{8.20}$$

From equation (8.7) $^{n}y_{j}^{I} = \frac{\partial\ ^{n}v^{II}}{\partial\ ^{n}w_{j}^{II}}$. Replacing this expression into equation (8.20) yields

$$\frac{\partial\ ^{m}\Theta}{\partial\ ^{mn}w_{j}^{II}} = \frac{\partial\ ^{m}\Theta}{\partial\ ^{mn}e}\frac{\partial\ ^{mn}e}{\partial\ ^{mn}y^{II}}\frac{\partial\ ^{mn}y^{II}}{\partial\ ^{n}y^{II}}\frac{\partial\ ^{n}y^{II}}{\partial\ ^{n}v^{II}}\ ^{n}y_{j}^{I} \tag{8.21}$$

Let us define the *local gradient* of layer *II* as

$$^{mn}\delta^{II} = \frac{\partial\ ^{m}\Theta}{\partial\ ^{mn}e}\frac{\partial\ ^{mn}e}{\partial\ ^{mn}y^{II}}\frac{\partial\ ^{mn}y^{II}}{\partial\ ^{n}y^{II}}\frac{\partial\ ^{n}y^{II}}{\partial\ ^{n}v^{II}} \tag{8.22}$$

Then equation (8.21) can be written as:

$$\frac{\partial\ ^{m}\Theta}{\partial\ ^{mn}w_{j}^{II}} = \ ^{mn}\delta^{II}\ \ ^{n}y_{j}^{I} \tag{8.23}$$

If we differentiate equations (8.16) and (8.18) with respect to $^{mn}y^{II}$ and ^{mn}e, respectively, we obtain:

$$\frac{\partial\ ^{mn}e}{\partial\ ^{mn}y^{II}} = -1 \tag{8.24}$$

$$\frac{\partial^{m}\Theta}{\partial^{mn}e} = {}^{mn}e \tag{8.25}$$

From equation (8.19) we can also compute:

$$\frac{\partial^{mn}y^{II}}{\partial^{n}y^{II}} = 1 \tag{8.26}$$

Replacing equations (8.24)-(8.26) into equation (8.22), gives

$${}^{mn}\delta^{II} = -{}^{mn}e \quad {}^{n}\varphi'^{II}\left({}^{n}v^{II}\right) \tag{8.27}$$

where ${}^{n}\varphi'^{II}\left({}^{n}v^{II}\right) = \frac{\partial^{n}y^{II}}{\partial^{n}v^{II}}$. If we apply the Δ-*Rule* to $\frac{\partial^{m}\Theta}{\partial^{mn}w_{j}^{II}}$, at every time step we get:

$$\Delta^{n}w_{j}^{II}(k) = \sum_{m=1}^{\mu}{}^{mn}\lambda^{II}\ \frac{\partial^{m}\Theta}{\partial^{mn}w_{j}^{II}}(k) \tag{8.28}$$

or

$$\Delta^{n}w_{j}^{II}(k) = \sum_{m=1}^{\mu}{}^{mn}\lambda^{II}\ {}^{mn}\delta^{II}(k)\ {}^{n}y_{j}^{I}(k) \tag{8.29}$$

where ${}^{mn}\lambda^{II}$ are the learning parameters of layer *II*, for each mode *m* and output *n*.

For the case of the first layer, we define the corresponding *sensitivity function* as

$$\frac{\partial^{m}\Theta}{\partial^{m}w_{ji}^{I}} = \frac{\partial^{m}\Theta}{\partial^{n}y_{j}^{I}}\frac{\partial^{n}y_{j}^{I}}{\partial^{n}v_{j}^{I}}\frac{\partial^{n}v_{j}^{I}}{\partial^{n}\hat{\omega}_{ji}^{I}}\frac{\partial^{n}\hat{\omega}_{ji}^{I}}{\partial^{m}w_{ji}^{I}} \tag{8.30}$$

From equations (8.5) and (8.4) we have $a_i = \frac{\partial\ {}^{n}v_j^I}{\partial\ {}^{n}\hat{\omega}_{ji}^I}$ and ${}^{nm}d = \frac{\partial\ {}^{n}\hat{\omega}_{ji}^I}{\partial\ {}^{m}w_{ji}^I}$.

Replacing these equations into equation (8.30) we have,

$$\frac{\partial\ {}^{m}\Theta}{\partial\ {}^{m}w_{ji}^I} = \frac{\partial\ {}^{m}\Theta}{\partial\ {}^{n}y_j^I}\frac{\partial\ {}^{n}y_j^I}{\partial\ {}^{n}v_j^I} \quad {}^{nm}d \quad a_i \tag{8.31}$$

Let's define the *local gradients* for layer *I* as:

$${}^{mn}\delta_j^I = \frac{\partial\ {}^{m}\Theta}{\partial\ {}^{n}y_j^I}\frac{\partial\ {}^{n}y_j^I}{\partial\ {}^{n}v_j^I} \tag{8.32}$$

Then equation (8.31) can be written as:

$$\frac{\partial\ {}^{m}\Theta}{\partial\ {}^{m}w_{ji}^I} = {}^{mn}\delta_j^I \quad {}^{nm}d \quad a_i \tag{8.33}$$

If we apply the Δ-Rule to the first layer, at every time step we get:

$$\Delta\ {}^{m}w_{ji}^I(k) = {}^{mn}\lambda^I \frac{\partial\ {}^{m}\Theta}{\partial\ {}^{m}w_{ji}^I}(k) \tag{8.34}$$

or

$$\Delta\ {}^{m}w_{ji}^I(k) = {}^{mn}\lambda^I \quad {}^{mn}\delta_j^I(k) \quad {}^{nm}d \quad a_i(k) \tag{8.35}$$

which is analogous to equation (8.29). Here ${}^{mn}\lambda^{I}$ are the learning parameters used in layer *I*, for each mode *m* and output contribution *n*, and ${}^{mn}\delta_{j}^{I}$ are the corresponding local gradients; they are given by

$$ {}^{mn}\delta_{j}^{I} = \frac{\partial\, {}^{m}\Theta}{\partial\, {}^{n}y_{j}^{I}} \frac{\partial\, {}^{n}y_{j}^{I}}{\partial\, {}^{n}v_{j}^{I}} = \left[{}^{mn}\delta^{II} \;\; {}^{n}w_{j}^{II} \right] \left[{}^{n}\varphi'^{I}\left({}^{n}v_{j}^{I}\right) \right] \tag{8.36} $$

where $\frac{\partial\, {}^{n}y_{j}^{I}}{\partial\, {}^{n}v_{j}^{I}} = {}^{n}\varphi'^{I}\left({}^{n}v_{j}^{I}\right)$. The term $\frac{\partial\, {}^{m}\Theta}{\partial\, {}^{n}y_{j}^{I}} = {}^{mn}\delta^{II} \;\; {}^{n}w_{j}^{II}$ comes from

$$ \frac{\partial\, {}^{m}\Theta}{\partial\, {}^{n}y_{j}^{I}} = \frac{\partial\, {}^{m}\Theta}{\partial\, {}^{mn}e} \frac{\partial\, {}^{mn}e}{\partial\, {}^{mn}y^{II}} \frac{\partial\, {}^{mn}y^{II}}{\partial\, {}^{n}y^{II}} \frac{\partial\, {}^{n}y^{II}}{\partial\, {}^{n}v^{II}} \frac{\partial\, {}^{n}v^{II}}{\partial\, {}^{n}y_{j}^{I}} = {}^{mn}\delta^{II} \;\; {}^{n}w_{j}^{II} \tag{8.37} $$

where ${}^{mn}\delta^{II} = \frac{\partial\, {}^{m}\Theta}{\partial\, {}^{mn}e} \frac{\partial\, {}^{mn}e}{\partial\, {}^{mn}y^{II}} \frac{\partial\, {}^{mn}y^{II}}{\partial\, {}^{n}y^{II}} \frac{\partial\, {}^{n}y^{II}}{\partial\, {}^{n}v^{II}}$ and ${}^{n}w_{j}^{II} = \frac{\partial\, {}^{n}v^{II}}{\partial\, {}^{n}y_{j}^{I}}$. The learning parameters of each layer (${}^{mn}\lambda^{I}$ and ${}^{mn}\lambda^{II}$) are determined heuristically. In actual practice we obtained good results by using two generic learning parameters for each layer (λ^{I} and λ^{II}).

To avoid local minima the concept of numerical momentum may also be applied. To that end a momentum coefficient μ is defined such that $0 < \mu < 1$. Then a certain memory of the evolution of every weight can be considered by computing each new weight according to

$$ {}^{n}w_{ji}^{I}(k+1) = {}^{n}w_{ji}^{I}(k) + {}^{n}\mu^{I}\left[{}^{n}w_{ji}^{I}(k) - {}^{n}w_{ji}^{I}(k-1) \right] \tag{8.38} $$

$$ {}^{n}w_{j}^{II}(k+1) = {}^{n}w_{j}^{II}(k) + {}^{n}\mu^{II}\left[{}^{n}w_{j}^{II}(k) - {}^{n}w_{j}^{II}(k-1)\right] \tag{8.39} $$

where k indicates the current time step.

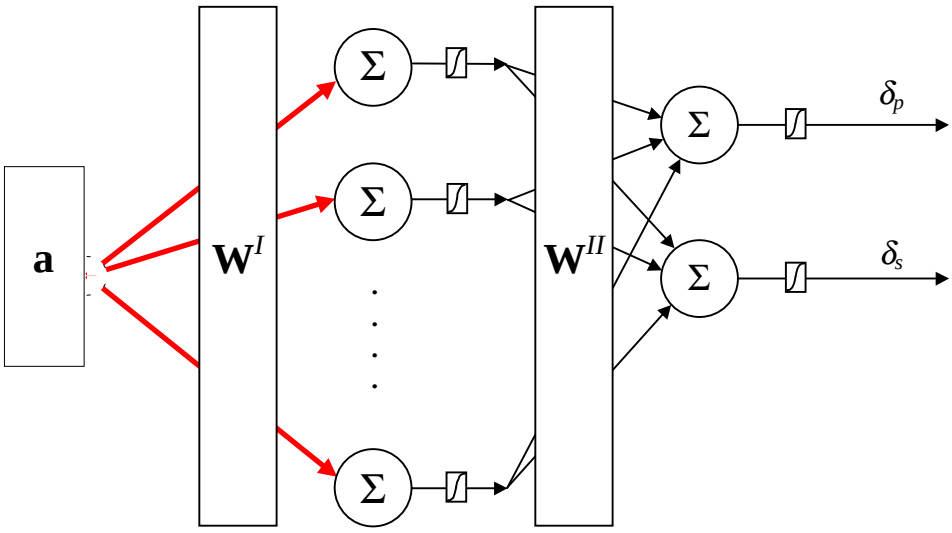

Figure 8-1: Neural-network controller for a MIMO system. The input represents the data fed to the controller. The two outputs δ_p and δ_s represent the port and starboard fin deflections for a fin-stabilizer system for ships. Thick arrows account for a set of axons with origin at each input data. Squashing functions are represented by $\int$.

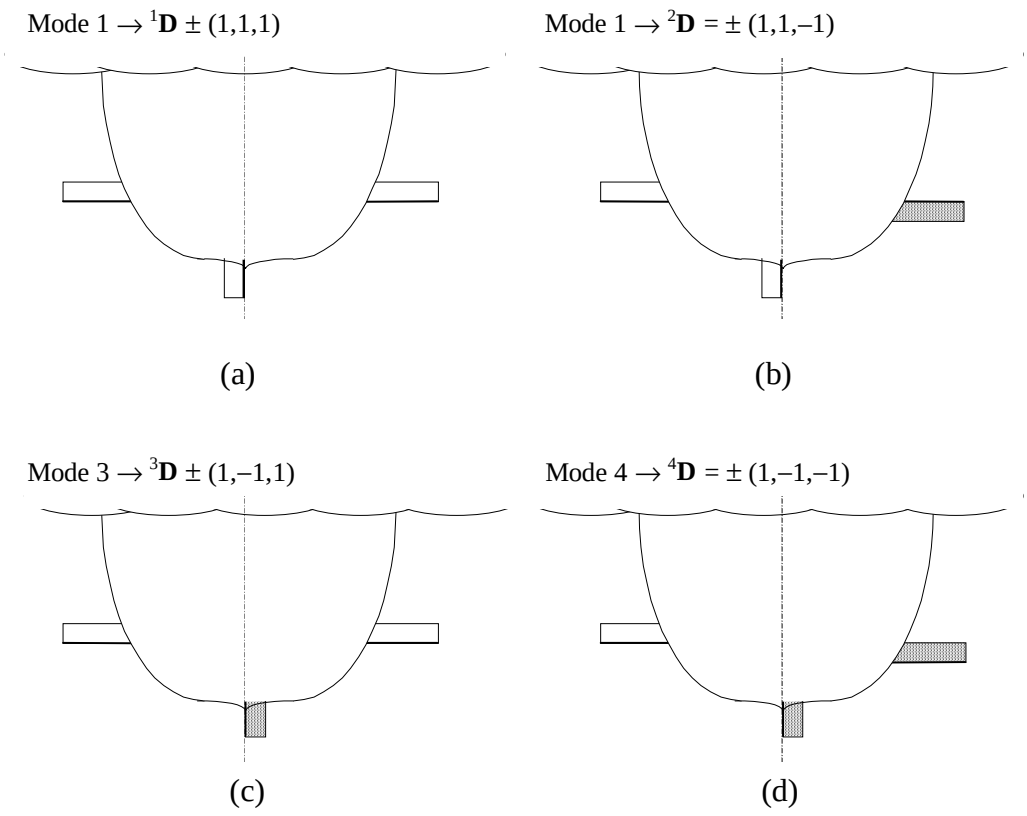

Figure 8-2: Four-mode rolling control by using a pair of fins and a rudder.

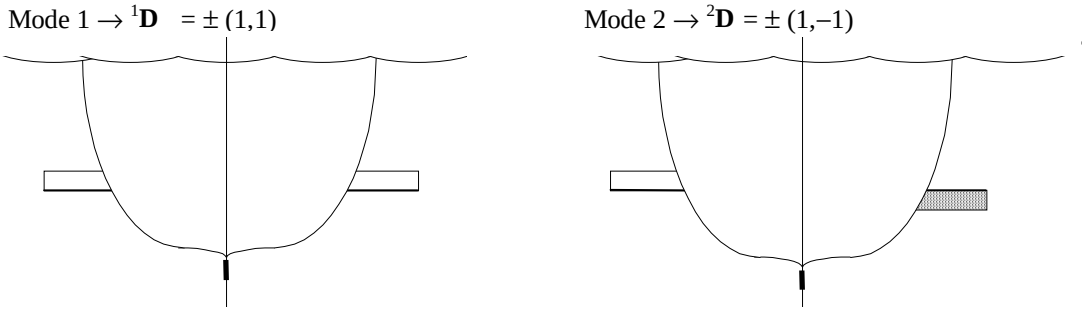

Figure 8-3: Two-mode rolling control by using a pair of fins.

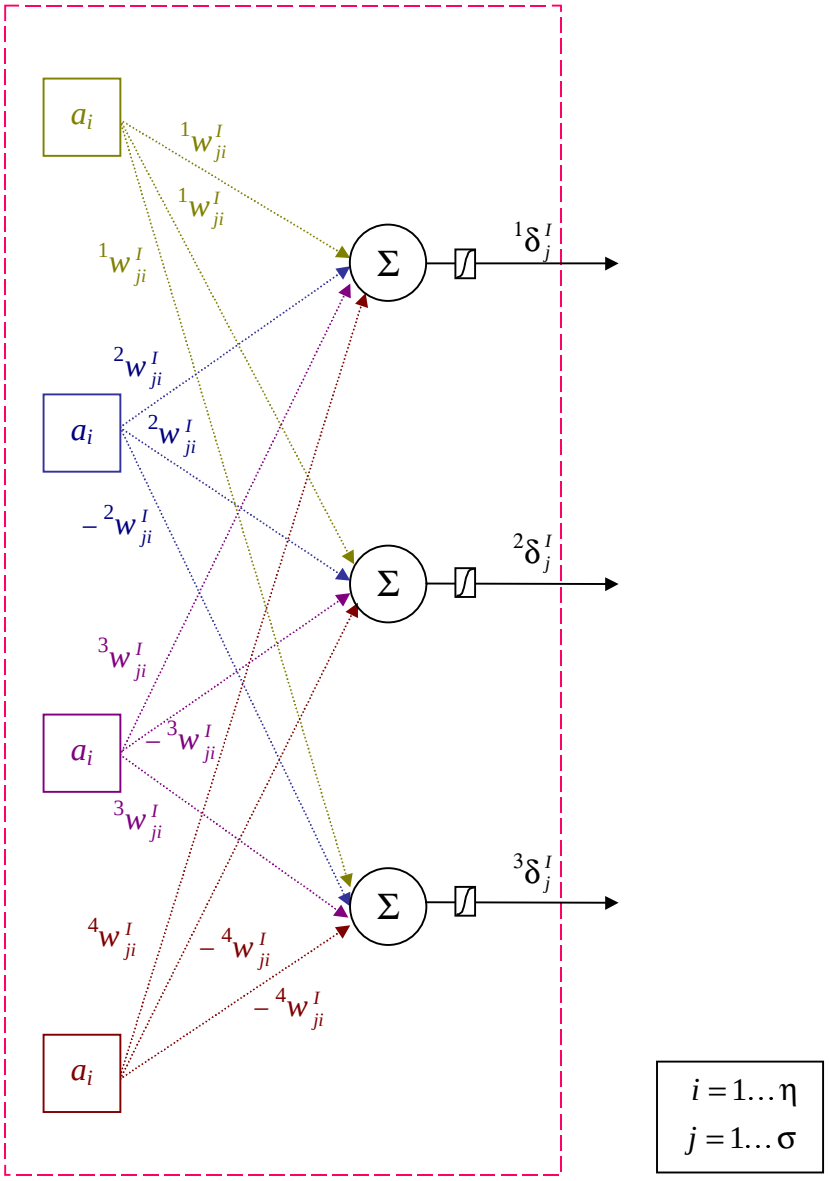

Figure 8-4: First-layer, four-mode, three-output modal neuron, *j*. Vector **a** is the input vector to the system of dimension η. Squashing functions are represented by ⎡∫⎤.

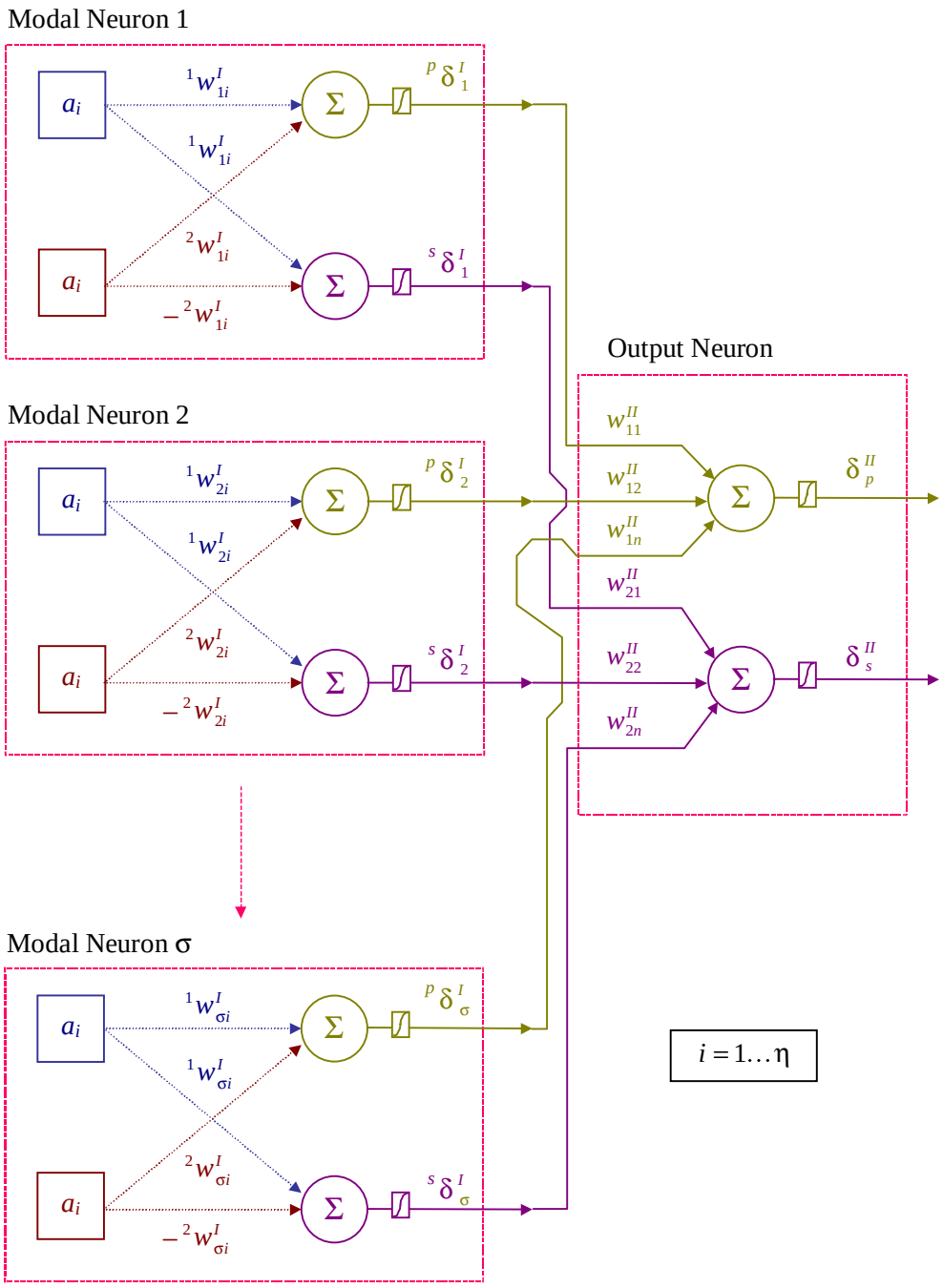

Figure 8-5: Two-layer, two-mode, two-output MNN controller. The outputs are the port and starboard fin deflections (δ_p and δ_s) used to reduce the rolling motion of a system. The first layer has σ two-mode modal neurons. Subscripts and superscripts *p* and *s* stand for port and starboard, respectively. Vector **a** is the input vector of dimension η. Squashing functions are represented by $\boxed{\int}$.

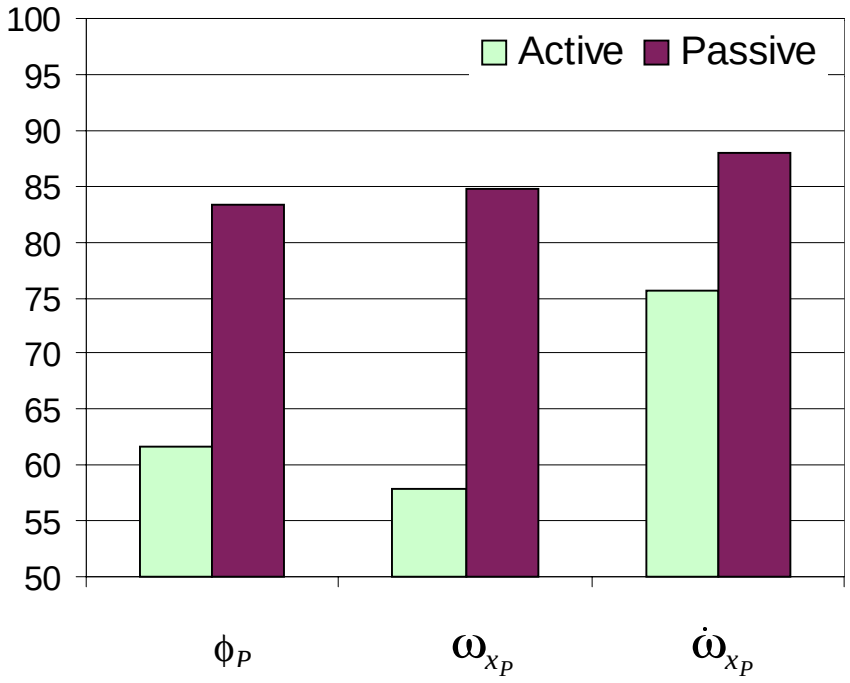

Figure 8-6: Roll reduction using a pair of fins controlled by a two-mode neural network.

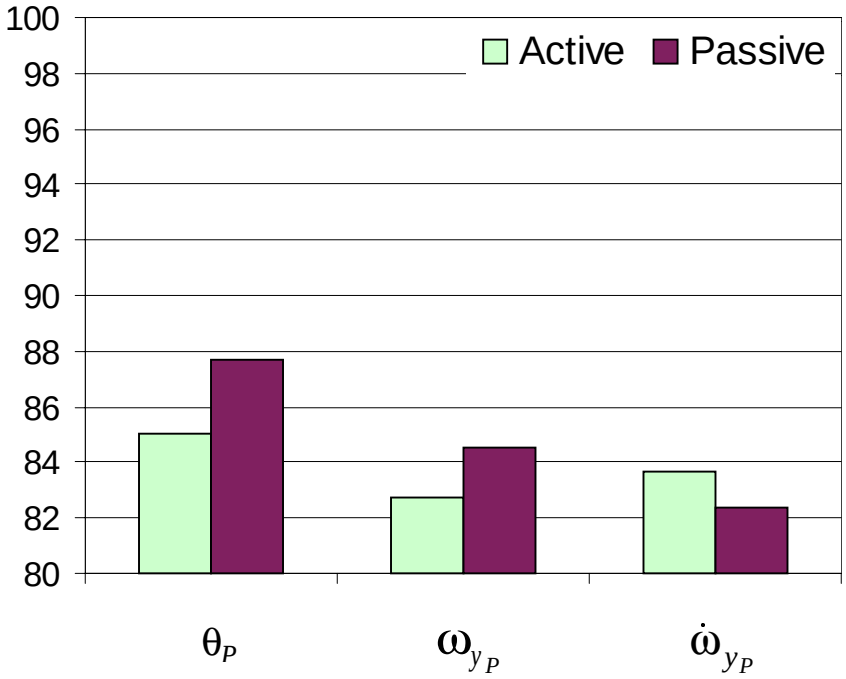

Figure 8-7: Pitch reduction using a pair of fins controlled by a two-mode neural network.

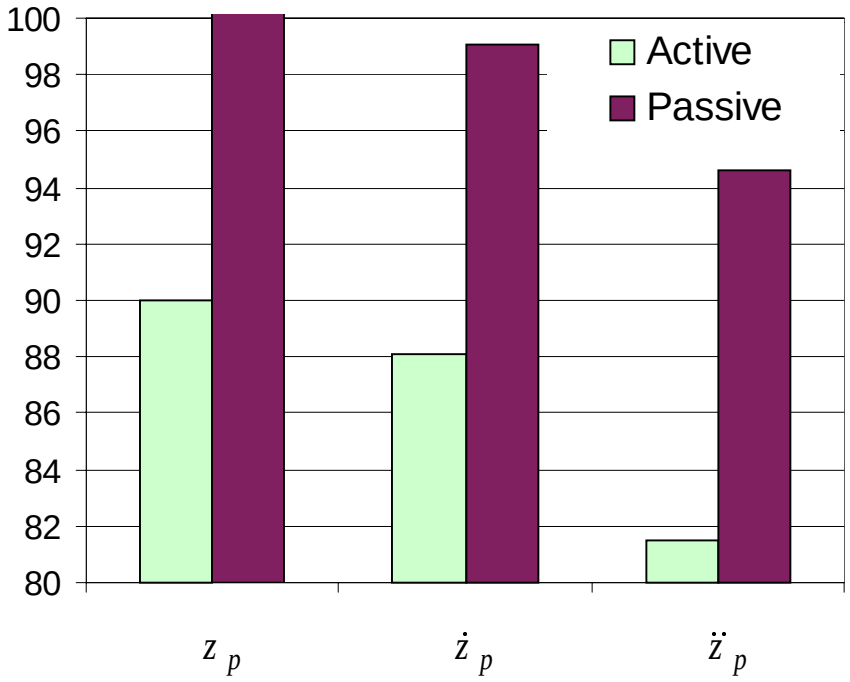

Figure 8-8: Heave reduction using a pair of fins controlled by a two-mode neural network.