Chapter 5

The Moment-Matching Procedure: A Marine Application

The following papers, by Liut *et al.*, are based on the contents of the this Chapter:

- **"Control of Rolling in Ships by Means of Active Fins," (reference [52])**
- **"Control of Rolling in Ships by Means of Active Fins Governed by a Neural-Network Controller," (reference [53])**
- **"Neural-Network and Fuzzy-Logic Control to Suppress the Rolling Motion in Ships by Means of Active Fins," (reference [55])**
- **"Roll Reduction in Ships by Means of Active Fins Controlled by a Neural Network," (reference [60])**

5.1 Introduction

The reduction of the rolling motion of ships constitutes a very important issue for a broad variety of applications. A sustained large roll can drastically reduce the performance of personnel on board and generate an adverse environment for sensitive equipment, interfering with their normal operation and reducing their accuracy below practical requirements. For some specialized vessels, such as those used for scientific or military purposes, large sustained roll motion may drastically reduce their expected capabilities.

To develop a roll-suppression system, we model a ship with movable fins attached to its hull. We use a neural-network-based controller to command the deflections of the fins relative to the hull. To predict the hydrodynamic loads, we combine two different hydrodynamic models. We use the same hydrodynamic approach we introduced in Chapter 2. Therefore we have a model and a program called LAMP (Large Amplitude Motion Program) which simulates the motion of a ship without fins in a seaway. As we discussed in Chapter 2, the hydrodynamics of LAMP is computed by means of a distribution of source panels on the wetted portion of the hull, and a time-dependent distribution of free-surface Green functions to model the sea.

The second model and program called FINS (also introduced in Chapter 2) simulates the flow around the fins attached to the hull by means of a general, unsteady, vortex-sheet method. The fins, nearby regions of the hull, and the wakes emanating from the trailing edges and tips of the fins are modeled.

Both programs operate interactively: at every time step LAMP computes the motion of the ship and passes this information to FINS; next FINS computes the hydrodynamic forces and moments acting on the fins and passes this information back to LAMP. Then LAMP uses this information to compute the ship motion. In Chapter 2 we showed some results when hull-fixed fins are used to reduce roll motion. Those results indicated a good passive performance.

In the present chapter we seek to improve the performance of the fins by means of a neural-network controller, which commands the rotations of the fins about a span-wise axis on each fin in order to reduce further the rolling motion. Both the neural-network controller and its training routine were implemented as part of the FINS code. To generate a suitable control law we developed a new training strategy, based on a moment-matching procedure. It can generate an adequate control law with relatively short computational efforts. The same training technique could also be applied on line on a real ship. This technique does not require using two extra neural networks to emulate LAMP and the FINS in order to generate a suitable control law, as a classical system-identification approach would demand (Narendra and Parthasarathy [71] and Nguyen

and Widrow [76]). A system-identification strategy is usually very time consuming, and the extra neural-network emulators required by it may convey the approximation of being *models of models*. In our problem this would imply generating neural-network models of LAMP and FINS, which are already models. A procedure of this sort that could be used for the application discussed in this chapter is summarized in Figure 5-1.

In Chapter 6 we will use the same principles of the moment-matching procedure to train a neural-network controller to reduce the seismic response of buildings.

5.2 The Dynamics of the Problem

The ship is assumed to be a rigid body. Two coordinate systems are used to describe its motion: one fixed to solid ground and considered inertial, and the other fixed to the ship. They are referred to as the *G*- and *B*-frames, respectively. Both coordinate systems are represented in Figure 5-2. In this picture *O* is any point in the ship, and *G* is its mass center. The description is set up this way because it may not always be as convenient to calculate the hydrodynamic moments around the mass center as it is to calculate them around some other point, such as the geometric center of the body. The position of *O* is given by

$$\mathbf{R}_O = X_O(t)\hat{\mathbf{I}} + Y_O(t)\hat{\mathbf{J}} + Z_O(t)\hat{\mathbf{K}} \tag{5.1}$$

where $(\hat{\mathbf{I}}, \hat{\mathbf{J}}, \hat{\mathbf{K}})$ are the base vectors aligned with the *G*-frame. The position of *G* with respect to *O* is given by

$$\mathbf{r}_G = x_G\,\hat{\mathbf{i}} + y_G\,\hat{\mathbf{j}} + z_G\,\hat{\mathbf{k}} \tag{5.2}$$

where $(\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}})$ are the base vectors aligned with the body-fixed axes. The absolute velocity (*i.e.*, the velocity with respect to the ground) of *O* can be expressed as

$$\mathbf{v}_O = v_x\,\hat{\mathbf{i}} + v_y\,\hat{\mathbf{j}} + v_z\,\hat{\mathbf{k}} \tag{5.3}$$

The angular velocity of the ship, that is the angular velocity of the *B*-frame with respect to the *G*-frame, is given by

$$\boldsymbol{\omega} = \omega_x\,\hat{\mathbf{i}} + \omega_y\,\hat{\mathbf{j}} + \omega_z\,\hat{\mathbf{k}} \tag{5.4}$$

The orientation of the ship with respect to the ground is given by a set of Euler angles. We start with the *G*- and *B*-frames aligned and follow Fossen [22]: first the ship is given a yaw-like rotation around the *z*-axis through the angle ψ (to produce a new heading); second, it is given a pitch-like rotation around the *y*-axis in the new position of the *B*-frame through the angle θ; third, it is given a roll-like rotation around the *x*-axis in the new position of the *B*-frame through the angle ϕ.

The position of point *O* in the ground-fixed system and the Euler angles are related to the components of the velocities expressed in the body-fixed system by the following differential equations:

$$\begin{bmatrix} \dot{X}_0 \\ \dot{Y}_0 \\ \dot{Z}_0 \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} c\psi\ c\theta & -s\psi\ c\phi + c\psi\ s\theta\ s\phi & s\psi\ s\phi + c\psi\ c\phi\ s\theta & 0 & 0 & 0 \\ s\psi\ c\theta & c\psi\ c\phi + s\phi\ s\theta\ s\psi & -c\psi\ s\phi + s\theta\ s\psi\ c\phi & 0 & 0 & 0 \\ -s\theta & c\theta\ s\phi & c\theta\ c\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & s\phi\ t\theta & c\phi\ t\theta \\ 0 & 0 & 0 & 0 & c\phi & -s\phi \\ 0 & 0 & 0 & 0 & \dfrac{s\phi}{c\theta} & \dfrac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{5.5}$$

where *c*, *s*, and *t* represent cosine, sine, and tangent functions, respectively. The equations of motion for the ship are given by (see, *e.g.*, Fossen [22]):

$$m\left[\dot{v}_x - v_y\omega_z + v_z\omega_y - x_G\left(\omega_y^2 + \omega_z^2\right) + y_G\left(\omega_x\omega_y - \dot{\omega}_z\right) + z_G\left(\omega_x\omega_z + \dot{\omega}_y\right)\right] = l_x + f_x \tag{5.6}$$

$$m\left[\dot{v}_y - v_z\omega_x + v_x\omega_z - y_G\left(\omega_z^2 + \omega_x^2\right) + z_G\left(\omega_y\omega_z - \dot{\omega}_x\right) + x_G\left(\omega_y\omega_x + \dot{\omega}_z\right)\right] = l_y + f_y \tag{5.7}$$

$$m\left[\dot{v}_z - v_x\omega_y + v_y\omega_x - z_G\left(\omega_x^2 + \omega_y^2\right) + x_G\left(\omega_z\omega_x - \dot{\omega}_y\right) + y_G\left(\omega_z\omega_y + \dot{\omega}_x\right)\right] = l_z + f_z \tag{5.8}$$

$$\begin{aligned} & I_{xx}\dot{\omega}_x + \left(I_{zz} - I_{yy}\right)\omega_y\omega_z - I_{xz}\left(\dot{\omega}_z + \omega_z\omega_y\right) + I_{yz}\left(\omega_z^2 - \omega_y^2\right) - I_{xy}\left(\dot{\omega}_y - \omega_x\omega_z\right) \\ & + \; m\left[y_G\left(\dot{v}_z - v_x\omega_y + v_y\omega_x\right) - z_G\left(\dot{v}_y - v_z\omega_x + v_x\omega_z\right)\right] = \eta_x + \mu_x \end{aligned} \tag{5.9}$$

$$\begin{aligned} & I_{yy}\dot{\omega}_y + \left(I_{xx} - I_{zz}\right)\omega_z\omega_x - I_{xy}\left(\dot{\omega}_x + \omega_y\omega_z\right) + I_{zx}\left(\omega_x^2 - \omega_z^2\right) - I_{yz}\left(\dot{\omega}_z - \omega_y\omega_x\right) \\ & + \; m\left[z_G\left(\dot{v}_x - v_y\omega_z + v_z\omega_y\right) - x_G\left(\dot{v}_z - v_x\omega_y + v_y\omega_x\right)\right] = \eta_y + \mu_y \end{aligned} \tag{5.10}$$

$$\begin{aligned} & I_{zz}\dot{\omega}_z + \left(I_{yy} - I_{xx}\right)\omega_x\omega_y - I_{yz}\left(\dot{\omega}_y + \omega_z\omega_x\right) + I_{xy}\left(\omega_y^2 - \omega_x^2\right) - I_{zx}\left(\dot{\omega}_x - \omega_z\omega_y\right) \\ & + \; m\left[x_G\left(\dot{v}_y - v_z\omega_x + v_x\omega_z\right) - y_G\left(\dot{v}_x - v_y\omega_z + v_z\omega_y\right)\right] = \eta_z + \mu_z \end{aligned} \tag{5.11}$$

where $\mathbf{l} = l_x\,\hat{\mathbf{i}} + l_y\,\hat{\mathbf{j}} + l_z\,\hat{\mathbf{k}}$ and $\boldsymbol{\eta} = \eta_x\,\hat{\mathbf{i}} + \eta_y\,\hat{\mathbf{j}} + \eta_z\,\hat{\mathbf{k}}$ denote the loads predicted by LAMP, and $\mathbf{f} = f_x\,\hat{\mathbf{i}} + f_y\,\hat{\mathbf{j}} + f_z\,\hat{\mathbf{k}}$ and $\boldsymbol{\mu} = \mu_x\,\hat{\mathbf{i}} + \mu_y\,\hat{\mathbf{j}} + \mu_z\,\hat{\mathbf{k}}$ denote the loads predicted by FINS. Here $\mathbf{l}$ and $\mathbf{f}$ are forces, whereas $\boldsymbol{\eta}$ and $\boldsymbol{\mu}$ are moments about *O*.

The loads predicted by LAMP include those generated by the wave motion at the surface of the sea, roll damping based on an empirical model to imitate the effects of viscosity, as well as all of the hydrodynamic and hydrostatic loads due to the motion of the ship. These include restoring loads (loads proportional to the displacement of *O* and the Euler angles), the so-called added-inertia effects (loads proportional to the acceleration), and radiation (or wave) damping (loads proportional to the velocity). These loads are functions of the wave motion, the acceleration, the velocity, and the position and orientation of the ship, as well as the recent history of its motion.

The loads predicted by FINS are functions of the motion of the ship, the motion of the fins relative to the ship, and the history of both motions. Because of the disparity in size, the hull in the neighborhood of the fins is re-gridded with a smaller size when the loads on the fins are calculated. In Figure 5-3, part *a*, half of the hull, one of the fins, and its wake are represented; in part *b* a side view of the same fin, its wake and the re-gridded neighborhood of the hull are also shown; in parts *c* and *d* the two grids are shown together for comparison. The no-penetration condition is imposed on the surfaces of the hull and the fins simultaneously and interactively. Smaller grid sizes require smaller time steps so LAMP and FINS use different time steps, as we described in Chapter 2. These numerical models have accurately predicted the motion of actual ships as well as the unsteady loads on the fins.

Based on Hamming's method (see Carnaham *et al.* [9]), a predictor-corrector algorithm was developed to integrate numerically the six equations of motion and the six kinematic relationships while solving simultaneously for all of the hydrodynamic loads on the hull and fins. The procedure yields the position of *O*, the Euler angles and the velocity of the ship, as well as all of the hydrodynamic forces, as functions of time. This approach was chosen for two reasons: 1) the general unsteady vortex-lattice method

performs better when the loads are evaluated at integral time steps; 2) the loads depend implicitly on the acceleration of the ship so that the acceleration appears on both sides of the governing equations. Because the procedure is iterative it can, with some modifications, compute the added masses and inertias at each time step as part of the solution. For both reasons, Runge-Kutta methods are not suitable. Detailed descriptions of the procedures for numerically integrating the equations and for calculating the flowfield around the fins are given in Chapter 2.

The motion of the fins relative to the ship is governed by a neural-network control system, which is described in the next section.

5.3 The Control System

The control law to govern the fins is a neural network. The input vector to the neural network for the present problem is noted by **p**. It consists of all of the state variables for the ship except X_0, the deflection of the rudder and a bias. The output $\delta = \delta_c$ is the commanded deflection of the fins.

For the present model, we built a controller based on the neural network we have described in Chapter 4. The equations we used for this particular neural network can be expressed as

$$\begin{bmatrix} v_1^I \\ v_2^I \\ v_3^I \\ v_4^I \end{bmatrix} = \begin{bmatrix} w_{1\ 1} & w_{1\ 2} & \cdot\cdot & \cdot\cdot & w_{1\ 12} & w_{1\ 13} \\ w_{2\ 1} & w_{2\ 2} & \cdot\cdot & \cdot\cdot & w_{2\ 12} & w_{2\ 13} \\ w_{3\ 1} & w_{3\ 2} & \cdot\cdot & \cdot\cdot & w_{3\ 12} & w_{3\ 13} \\ w_{4\ 1} & w_{4\ 2} & \cdot\cdot & \cdot\cdot & w_{4\ 12} & w_{4\ 13} \end{bmatrix} \begin{bmatrix} Y_0 \\ Z_0 \\ v_x \\ v_y \\ v_z \\ \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z \\ \delta_r \\ b \end{bmatrix} \quad \text{or} \quad \mathbf{v}^I = \mathbf{W}^I\ \mathbf{p} \tag{5.12}$$

and

$$\begin{bmatrix} y_1^I \\ y_2^I \\ y_3^I \\ y_4^I \end{bmatrix} = \begin{bmatrix} A^I\ \tanh\left(\beta^I\ v_1^I\right) \\ A^I\ \tanh\left(\beta^I\ v_2^I\right) \\ A^I\ \tanh\left(\beta^I\ v_3^I\right) \\ A^I\ \tanh\left(\beta^I\ v_4^I\right) \end{bmatrix} \qquad \text{or} \qquad \mathbf{y}^I = A^I\ \mathbf{Tanh}\left(\beta^I\ \mathbf{v}^I\right) \tag{5.13}$$

From Equation (5.13) we see that the squashing functions selected for our neural network are hyperbolic tangents. The vector **Tanh** is a vector of hyperbolic tangents, A and β are the amplitude and "frequency" of the squashing functions, $\mathbf{v}^I = \left[v_1^I, v_2^I, v_3^I, v_4^I\right]^T$ and $\mathbf{y}^I = \left[y_1^I, y_2^I, y_3^I, y_4^I\right]^T$. The outputs y_j^I from every neuron are restricted by the amplitudes A according to $-A^I \le y_j^I \le A^I$. For our two-layer, single-output architecture, the second layer (composed of one neuron, in the present example) yields the output of the neural network, δ_c:

$$v^{II} = \mathbf{W}^{II}\ \mathbf{y}^I \quad \text{and} \quad \delta_c = A^{II}\ \tanh\left(\beta^{II}\ v^{II}\right) \tag{5.14}$$

where $\mathbf{W}^{II} = \left[w_1^{II}, w_2^{II}, w_3^{II}, w_4^{II} \right]$ is the vector of weights for the second layer.

The deflection of the port fin is δ_p and that of the starboard fin is $\delta_s = -\delta_c$. In this work we use a five-neuron, two-layer arrangement, which is represented schematically in Figure 5-4.

The amplitudes and frequencies of the squashing functions are determined heuristically. However, some information about the magnitude of the input data and expected output is very helpful for the tuning process.

5.4 The Training Procedure

The weight matrices define the control law. The technique used to select them constitutes the neural-network training procedure. There are several methods to provide training for a neural network. In this chapter we introduce a novel backpropagation-based approach. We will refer to this method as the moment-matching technique. To describe it we will apply the backpropagation-technique concepts discussed in Chapter 4 to a multiple-input-single-output, two-layer neural network controller, to govern the motion of the fins. The corresponding roll-reduction performance is discussed in section 5.6.

Let's suppose there is a certain desired deflection of the fins $\delta_d(t)$, which yields an optimal rolling suppression for all time. An error $e(t)$ can be defined as the difference between this deflection and the commanded deflection $\delta_c(t)$, [equation (5.14)]:

$$e(t) = \delta_d(t) - \delta_c(t) \tag{5.15}$$

To abbreviate the notation, we omit the time argument from all variables in the following description.

A measure of the performance Θ is defined as half of the square of the error e:

$$\Theta = \frac{1}{2} e^2 = \frac{1}{2} (\delta_d - \delta_c)^2 \tag{5.16}$$

The backpropagation technique implemented for this work makes a time-step-by-time-step adjustment of all of the weights of the neural network. This adjustment is proportional to the error *e* and seeks to cancel it. Following the same procedure described in Chapter 4, we compute the derivatives of the performance index, Θ, with respect to the neural-network weights as:

$$\frac{\partial \Theta}{\partial w_j^{II}} = \frac{\partial \Theta}{\partial e} \frac{\partial e}{\partial \delta_c} \frac{\partial \delta_c}{\partial v^{II}} \frac{\partial v^{II}}{\partial w_j^{II}} = - \frac{\beta^{II} A^{II} (\delta_d - \delta_c) y_j^{II}}{\cosh^2 (\beta^{II} v^{II})} \tag{5.17}$$

where $\frac{\partial \Theta}{\partial e} = \delta_d - \delta_c$, $\frac{\partial e}{\partial \delta_c} = -1$, $\frac{\partial \delta_c}{\partial v^{II}} = \frac{\beta^{II} A^{II}}{\cosh^2 (\beta^{II} v^{II})}$ and $\frac{\partial v^{II}}{\partial w_j^{II}} = y_j$.

In the same manner, for the first layer, the derivatives of Θ with respect to the corresponding weights are given by

$$\frac{\partial \Theta}{\partial w_{ji}^{I}} = \left[\frac{\partial \Theta}{\partial e} \frac{\partial e}{\partial \delta_c} \frac{\partial \delta_c}{\partial v^{II}} \right] \frac{\partial v^{II}}{\partial y_j^{I}} \frac{\partial y_j^{I}}{\partial v_j^{I}} \frac{\partial v_j^{I}}{\partial w_{ji}^{I}} = - \frac{\beta^{I} \beta^{II} A^{I} A^{II} (\delta_d - \delta_c) w_j^{II} p_i}{\cosh^2 (\beta^{I} v_j^{I}) \cosh^2 (\beta^{II} v^{II})} \tag{5.18}$$

where from the last layer $\left[\frac{\partial \Theta}{\partial e} \frac{\partial e}{\partial \delta_c} \frac{\partial \delta_c}{\partial v^{II}} \right] = - \frac{\beta^{II} A^{II} (\delta_d - \delta_c)}{\cosh^2 (\beta^{II} v^{II})}$ and where $\frac{\partial v^{II}}{\partial y_j^{II}} = w_j^{II}$, $\frac{\partial y_j}{\partial v_j^{I}} = \frac{\beta^{I} A^{I}}{\cosh^2 (\beta^{I} v_j^{I})}$, and $\frac{\partial v_j^{I}}{\partial w_{ji}^{I}} = p_i$. Following the Δ-rule, the adjustments in the weights are made proportional to both $\frac{\partial \Theta}{\partial w_j^{II}}$ and $\frac{\partial \Theta}{\partial w_{ji}^{I}}$. Applying this at every time step, *k*, we continuously update the weights according to

$$w_j^{II}(k+1) = w_j^{II}(k) + \lambda^{II} \frac{\partial \Theta}{\partial w_j^{II}}(k) = w_j^{II}(k) + \lambda^{II} \left[- \frac{\beta^{II} A^{II} (\delta_d(k) - \delta_c(k)) y_j^{II}(k)}{\cosh^2 \left(\beta^{II} v^{II}(k) \right)} \right] \quad (5.19)$$

$$w_{ji}^{I}(k+1) = w_{ji}^{I}(k) + \lambda^{I} \frac{\partial \Theta}{\partial w_{ji}^{I}}(k) = w_{ji}^{I}(k) + \lambda^{I} \left[\frac{\beta^{I} \beta^{II} A^{I} A^{II} (\delta_d(k) - \delta_c(k)) \; u_j(k) \; p_i(k)}{\cosh^2 \left(\beta^{I} v_j^{I}(k) \right) \cosh^2 \left(\beta^{II} v^{II}(k) \right)} \right] \quad (5.20)$$

The constants of proportionality λ^I and λ^{II} are the learning parameters for the first and second layers, respectively. They are determined by trial and error.

We already have the basic building blocks to apply the backpropagation technique. However, since δ_d is not known, we are not ready to compute the error *e*, yet. It will be the moment-matching approach what will give us an effective procedure to find an appropriate estimator of this error.

If we consider the *x*-component of the angular acceleration, $\dot{\omega}_x$ [recall equation (5.9)], we obtain

$$\begin{aligned} I_{xx} \dot{\omega}_x \;=\; & -(I_{zz} - I_{yy}) \omega_y \omega_z + I_{xz} (\dot{\omega}_z + \omega_z \omega_y) - I_{yz} (\omega_z^2 - \omega_y^2) + I_{xy} (\dot{\omega}_y - \omega_x \omega_z) \\ & - m \left[y_G (\dot{v}_z - v_x \omega_y + v_y \omega_x) - z_G (\dot{v}_y - v_z \omega_x + v_x \omega_z) \right] + \eta_x + \mu_x \end{aligned} \quad (5.21)$$

The goal of the moment-matching procedure is to minimize $\dot{\omega}_x$. The only quantity that can be controlled directly is δ_c, which has a strong effect on the *x*-component of the moment generated by the fins, μ_x. If the fins have the desired deflection δ_d (*i.e.*, the deflection that minimizes $|\dot{\omega}_x|$), the moment μ_x has the desired value and no further adjustment to the control law is needed; the commanded deflection δ_c is the desired

deflection. However, if $|\dot{\omega}_x|$ has not reached a minimum, an adjustment to the moments currently produced by the fins should be applied. This adjustment in μ_x is a function of the difference between the commanded and desired deflections. Presumably at any given instant it can be expressed as a Taylor series in powers of this difference. We assume that we can use only the first term in this expansion to obtain a reasonably accurate approximation to the difference between the commanded and desired deflections (the procedure is similar to the familiar Newton-Raphson procedure for solving nonlinear equations). Thus we put,

$$\Delta\mu_x = -\tilde{\lambda}\, I_{xx}\, \dot{\omega}_x \cong \gamma\,(\delta_d - \delta_c) \tag{5.22}$$

where γ and $\tilde{\lambda}$ are constants. Actually they should be adjusted from one time-step to the next. This would lead to a second-order gradient-search algorithm, which is not studied in this work. The adjustment $\Delta\mu_x$ accounts for the hydrodynamic moments due to the motion of the ship through the sea, the so-called inertial moments, and the moments currently being generated by the fins. But, to avoid instability, we limit the size $\Delta\mu_x$, taking only small steps in the right direction.

Combining equation (5.22) with equations (5.19) and (5.20), we update the weights according to

$$w_j^{II}(k) = w_j^{II}(k-1) + \tilde{\lambda}^{II}\left[\frac{\beta^{II} A^{II} \dot{\omega}_x\, y_j^{II}}{\cosh^2\left(\beta^{II} v^{II}\right)}\right] \tag{5.23}$$

$$w_{ji}^{I}(k) = w_{ji}^{I}(k-1) + \tilde{\lambda}^{I}\left[-\frac{\beta^{I}\beta^{II} A^{I} A^{II} \dot{\omega}_x\; u_j\; p_i}{\cosh^2\left(\beta^{I} v_j^{I}\right)\cosh^2\left(\beta^{II} v^{II}\right)}\right] \tag{5.24}$$

where the modified learning parameters, $\tilde{\lambda}^{I}$ and $\tilde{\lambda}^{II}$, account for the moment of inertia and the constant in equation (5.22). Modifying the learning parameters is not an added complication since they are determined heuristically in any case.

To avoid local minima we also add numerical momentum to the algorithm. To this end, at every time step, the following corrections are added to the weights:

$$w_j^{II}(k+1) = w_j^{II}(k) + \kappa^{II}\left[w_j^{II}(k) - w_j^{II}(k-1)\right] \tag{5.25}$$

$$w_{ji}^{I}(k+1) = w_{ji}^{I}(k) + \kappa^{I}\left[w_{ji}^{I}(k) - w_{ji}^{I}(k-1)\right] \tag{5.26}$$

where κ^{I} and κ^{II} are the numerical moment coefficients for layers *I* and *II*, respectively. They are lesser-than-one constants, which are tuned heuristically.

If at the beginning of the next time step, the updated set of weights constituted an improvement of the control law, $\dot{\omega}_x$ would then be closer to the desired minimum value. It seems likely that the situation at the next time step will be similar to the one at the current time step and so we use the new values for the next time step as an improvement over the current values. In this way, the weights are continuously changing while the ship is in motion.

In the numerical simulation the same seastate and the same initial conditions are run several times for the same time window. Each of these runs is called an epoch or cycle. In the first epoch the weights are initialized randomly with small values. We have found that the overall tuning procedure is simplified if the input data are normalized according to some criterion. The criterion applied here consists of determining the average absolute value of each input datum over the first epoch. To this end, no training is performed during the first cycle. Then for every subsequent cycle, each input p_i is divided by its average as determined during the first epoch. In this way every input to the network is adjusted to have approximately the same magnitude.

If the size of the time window is sufficiently large, the training process could be completed in just one epoch. This would be analogous to a real-life situation where the training procedure described here could be implemented in an actual ship. When using mathematical models (such as LAMP and FINS) computational limitations will determine whether to use one large time window or several smaller windows in successive cycles.

The training will proceed until a minimum in the rolling response is achieved. The optimal sets of weights for different conditions are different, but the same training procedure can be applied to find them all. Different sets of weights could then be stored for different seastates, speeds and headings to be used as initial guesses to enhance the convergence when conditions change and the training procedure begins to search for a new optimal set.

5.5 Convergence

In this section we analyze the convergence of the training routine. First we study an algorithm which determines when the optimal weights were reached for the current sea characteristics, and load distribution. Then we propose a procedure to update the weights when sea, structural and/or load changes occur.

5.5.1 Convergence Algorithm

The error signal used by the present algorithm is proportional to the rolling acceleration. If there exists a control law that can suppress any roll acceleration, the training process will eventually find it and the roll will be suppressed. Then the backpropagation error will be reduced to zero for all time. But if such an ideal control law does not exist, then the rolling acceleration will never be totally suppressed. In this case a residual backpropagation error would always be present and that may have the

effect of growing the weights indefinitely, which could eventually cause the system to diverge.

In this section we introduce an algorithm which can estimate when a near optimal control law has been reached in order to stop updating the weights. This algorithm does not require the use of epochs. However if epochs are still used for other reasons (such as limited computational memory, for example), the algorithm can be adjusted to *see* a single cycle of successive epochs.

First a time window, TW, is defined. At the beginning of the TW, the current weights are stored. For a two-layer neural network, such as the one used here, this can be represented by

$$\mathbf{w}_0^I = \mathbf{w}^I \qquad \text{and} \qquad \mathbf{w}_0^{II} = \mathbf{w}^{II} \tag{5.27}$$

where $\mathbf{w}^{II}$ is the matrix of weights of the second layer (which is actually a vector) and $\mathbf{w}^I$ is the matrix of weights of the first layer. Matrices $\mathbf{w}_0^I$ and $\mathbf{w}_0^{II}$ are referred to as the reference-weight matrices. During the TW an average of the wave heights of the surrounding sea is calculated. For our problem we use four *virtual* pressure sensors, one on either side of the ship close to its geometrical center, one at the bow, and one at the stern (see Figure 5-7). The pressures, $h_i(k)$, are measured at these stations at every time step, and added according to

$$\sigma(k+1) = \sigma(k) + \sum_{i=1}^{N_w} h_i(k) \tag{5.28}$$

where N_w is the number of wave stations ($N_w = 4$ for our example). Also during a TW the absolute values of the rolling speed, $\dot{\phi}$, are added according to

$$\overline{\zeta}(k+1) = \overline{\zeta}(k) + \left|\dot{\phi}(k)\right| \tag{5.29}$$

Whenever a TW starts both σ and $\bar{\zeta}$ are initialized to zero. At the end of each TW, $\bar{\zeta}$ is normalized with σ as follows:

$$\zeta(K) = \frac{\bar{\zeta}(K)}{\sigma(K)} \tag{5.30}$$

where *K* is the total number of time steps in the TW. This gives a measure of the ratio of the rolling motion to the wave excitation that the ship encountered during the TW. Then ζ is compared with ζ' from the previous TW. If $\zeta < \zeta'$, $\mathbf{w}_0^I$ and $\mathbf{w}_0^{II}$ are updated with the current weights according to Equation (5.27), ζ' is replaced with ζ, and a new TW is started. If $\zeta > \zeta'$ the current weights are modified so they take the values they had at the beginning of the current TW, and we do not replace ζ' with the new ζ; the current ζ' will be used as a ζ-reference until a TW is found where $\zeta < \zeta'$. Thus we ignore the weight updates done over any TW in which ζ was not reduced.

The size of the window we used was of the order of magnitude of two average wave periods. The algorithm performs better if an additional time window, larger than TW, is defined. We refer to this window as the large time window (LTW). Any LTW is initialized whenever $\zeta < \zeta'$ at the end of each TW, and the reference weights $\mathbf{w}_0^I$ and $\mathbf{w}_0^{II}$ are updated with the current weights according to Equation (5.27). But if at the end of any TW $\zeta > \zeta'$ then the current ζ' will be used as a ζ reference until a new $\zeta < \zeta'$ is encountered, as before, but no reset is done to the weights nor to the LTW. The process of resetting the weights to the reference values ($\mathbf{w}_0^I$ and $\mathbf{w}_0^{II}$) is now performed whenever a LTW is completed. This only occurs when, for successive TW's, the roll performance was not improved ($\zeta > \zeta'$). We experimented with a LTW ten times larger than the current TW. To avoid abrupt changes of the weights when the end of a LTW is

reached, a gradual linear resetting of the weights is performed towards the corresponding reference weights ($\mathbf{w}_0^I$ and $\mathbf{w}_0^{II}$) over a time window of 20% the size of the LTW.

5.5.2 Adapting to Changes in Sea, Structure and Ship Loads

Still some additional steps are needed to update the control law when significant changes in the sea, structure and/or loads occur. This will give us a neural-network controller that is constantly adapting to the environment and changes in the ship.

Here we discuss an algorithm that may meet these demands. However this is part of our future work.

The algorithm is based on a gradual and slow relaxation of the reference weights. For our two-layer neural network this would be done according to

$$\mathbf{w}_0^I(k+1) = r\,\mathbf{w}_0^I(k) \qquad \text{and} \qquad \mathbf{w}_0^{II}(k+1) = r\,\mathbf{w}_0^{II}(k) \tag{5.31}$$

where $r < 1$ is a factor very close to one. Then at the end of each TW, estimates of the weight-vector size and the reference-weight vector size are computed, according to

$$\overline{w} = \sum_i^P \sum_j^{N^I} \left(w_{ji}^I\right)^2 + \sum_j^{N^I} \left(w_j^{II}\right)^2 \tag{5.32}$$

$$\overline{w}_0 = \sum_i^P \sum_j^{N^I} \left(w_{0_{ji}}^I\right)^2 + \sum_j^{N^I} \left(w_{0\ j}^{II}\right)^2 \tag{5.33}$$

respectively, where P is the number of input data and N^I is the number of neurons in the first layer. Now instead of using the criterion $\zeta < \zeta'$ we use

$$V = \left(\zeta < \zeta' \quad \text{or} \quad \overline{w}_0 < c\,\overline{w}\right) \tag{5.34}$$

where c is a constant such that $0 < c < 1$. If $V = True$ then $\mathbf{w}_0^I$ and $\mathbf{w}_0^{II}$ are updated with the current weights according to Equation (5.27) and a new TW and LTW are started as before. If $V = False$ we execute the same procedure described previously, when the criterion $\zeta > \zeta'$ was applied. In this way we prevent the weights from being *frozen* at some values that could have been optimal for a certain sea and/or ship characteristics but which may not constitute the best weight configuration when these characteristics have changed. Thus the decaying factor r works like a memory relaxation or, in contrast to the learning parameters, like a *forgetting parameter* of previous control knowledge expressed in terms of current weight matrices. In this manner, we would have a dynamic learning process, which would keep reinforcing optimal-control patterns extracted from recent environmental or structural data while discarding control patterns previously learned from environmental or structural conditions that have changed over time.

5.6 Results

The hull geometry and the inertia components used in the present examples are typical of a CG47 cruiser of the US Navy. The ship is moving at 20 knots. A fin is attached to each side of the hull amidships close to the bilge area. The fins are tapered and have an aspect ratio of 2.2. The ratio of the tip-to-root chords is 0.8. Only the leading edge is tapered. The length of the root chord of the fins is 2% of the length of the ship. The fins have a dihedral angle of 15 degrees towards the bottom of the ship.

First a random beam sea is considered. In Figure 5-5.a, the Euler angle ϕ is plotted as a function of time for two cases. The light line represents the motion when there are no fins. The dark line represents the motion when passive fins are present. Figure 5-5.b is the counterpart of Figure 5-5.a, but for ω_x. A ϕ-performance, ϕ_p, is defined as

$$\phi_p = \frac{\sum_{k=1}^{K} |\phi(k)|_e}{\sum_{k=1}^{K} |\phi(k)|_0} \tag{5.35}$$

where 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-performance. For the examples shown in Figure 5-5, the ϕ-performance was 62% and the ω_x-performance 67%.

In Figure 5-6, the results for active fins for a ship in a random beam sea are presented. For both ϕ and ω_x, the light line represents the motion when there are no fins and the dark line represents the motion when active fins are present. They are controlled by the neural network shown in Figure 5-4. The network has been trained over a window from 200 to 3000 time steps, over 7 training sessions or epochs. The ϕ-performance for this case was 32% and the ω_x -performance 33%.

The input data can be modified to include some information about the position of the surface of the sea at the four points indicated in Figure 5-7. Such measurements would be easy to obtain from transducers mounted on the hull. When this information is added to the input vector, the heave and pitch are eliminated. Then the training was repeated. In Figure 5-8 the results obtained for this case, also for a random beam sea, are displayed. For this case we obtained both a ϕ-performance and ω_x-performance of 29%.

It is clear that the neural-network-commanded fins are very effective in further reducing the roll motion when compared with passive fins and with no fins at all. From the results in Figures 5-5 and 5-7, some improvement in the active fin performance is detected when the input data contain information about the sea surface.

The case considered in Figure 5-9 is for a random sea that has a strong component at 45 degrees relative to the ship. Now the ϕ-performance is 41% and the ω_x-performance is 29%. Some small degradation in the performance is detected.

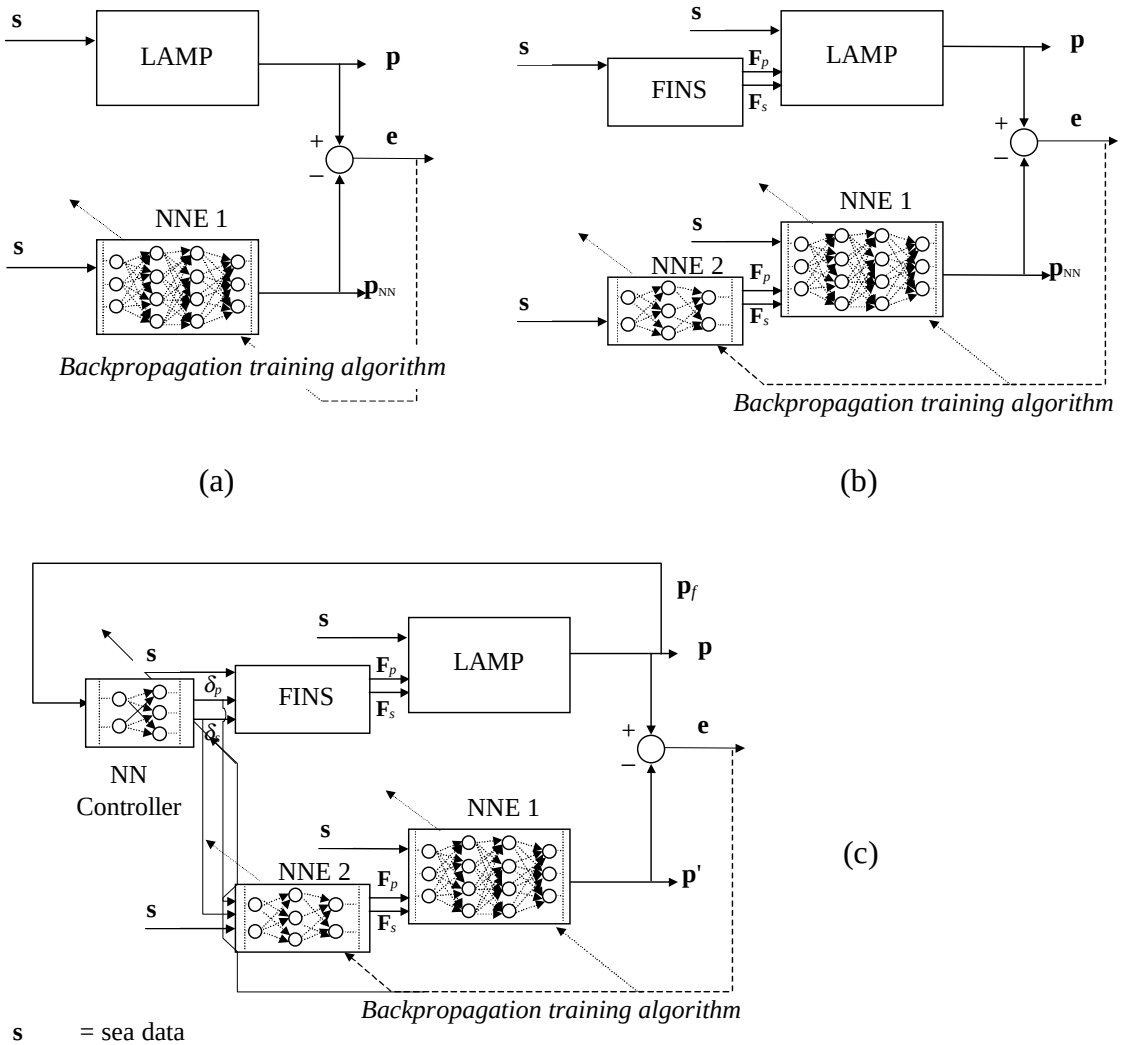
Figure 5-10 shows ϕ, ω_x, and $\dot{\omega}_x$ as functions of time when the ship is in a regular beam sea. The dark lines (which correspond to the active fins in all cases) show how fast the neural network can *learn* a very efficient control law for this particular sea. In the case of a real-life situation this would correspond to less than half an hour of control training. In random seas it takes more time to train the controller, on the order of ten times longer, which could still be considered relatively fast training in real-life applications. However, about three times more training effort with respect to a regular sea case (about two hours for a real ship) is sufficient to obtain an effective control (70-80% of the optimal performance). The performance of the fins in regular seas was the best of all the cases studied, with a ϕ-performance of 10% and ω_x-performance of 11%. In Figure 5-10.c the plots of $\dot{\omega}_x$ were included, which show a $\dot{\omega}_x$-performance of 18%. These last performances were computed for the last 1,000 time steps, where the training process had reached a steady state.

For the results shown so far we allowed the ship to move in all degrees of freedom, except in the *x*-direction. Part of our future work will be the study of the fins-propeller interaction, which would lead to a fully integrated propulsion-fin control system.

5.7 Concluding Remarks

In this chapter, we described a method to predict the performance and behavior of a ship with active fins attached to its hull. A novel training technique based on a moment-matching approach was developed to train a neural-network controller in order to minimize the rolling response of a ship in various types of random and regular seas. This approach does not require an additional neural-network to emulate the ship as is commonly the case with other backpropagation-based schemes, and could be applied on line on a real ship. The training process proved to be efficient and expeditious. The

performance of the controller is improved when information about the surrounding sea is available.



s = sea data
$\mathbf{p}_f$ = feedback vector
$\mathbf{F}_p$, $\mathbf{F}_s$ = port and starboard loads produced by the fins
δ_p, δ_s = port and starboard fin deflections

Figure 5-1: System identification training process. In part *a*, NNE 1 is the LAMP neural-network emulator. By comparing the output vectors **p** and **p**', an error is determined, which is backpropagated through NNE 1. When $\mathbf{p} \cong \mathbf{p}'$ for any input data, LAMP is considered identified by NNE 1, and the training process is stopped. Then the fin emulator NNE 2 is added (part *b*). The process is repeated; the error **e** is backpropagated through NNE 1 and NNE 2 until $\mathbf{p} \cong \mathbf{p}'$, for any input data. Only the weights of NNE 2 are adjusted. Finally the neural-network controller is added (part *c*). A new error **e**' is defined as a subset of **p**, which is minimized. When **e**' is backpropagated through NNE 1 and NNE 2, only the weights of the neural-network controller are adjusted until **e**' meets the desired average minimization values.

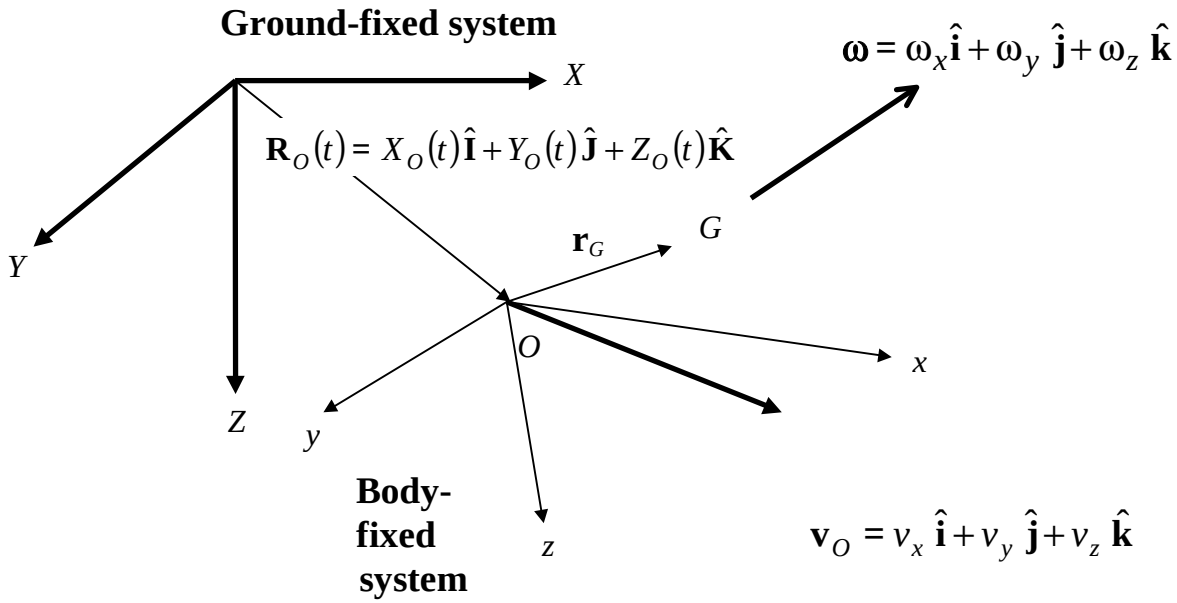


Figure 5-2: The two coordinate systems used to describe the motion of a ship.

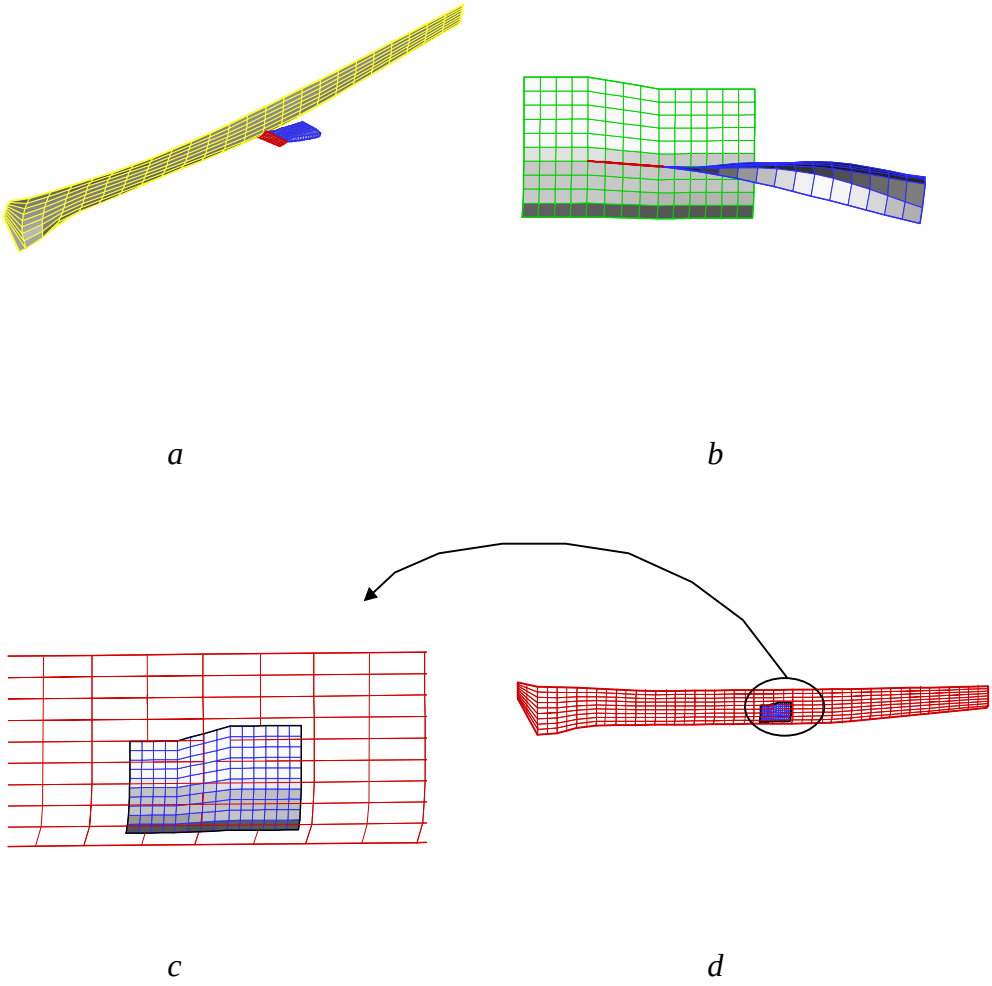


Figure 5-3: Grids for hull, fins and wake.

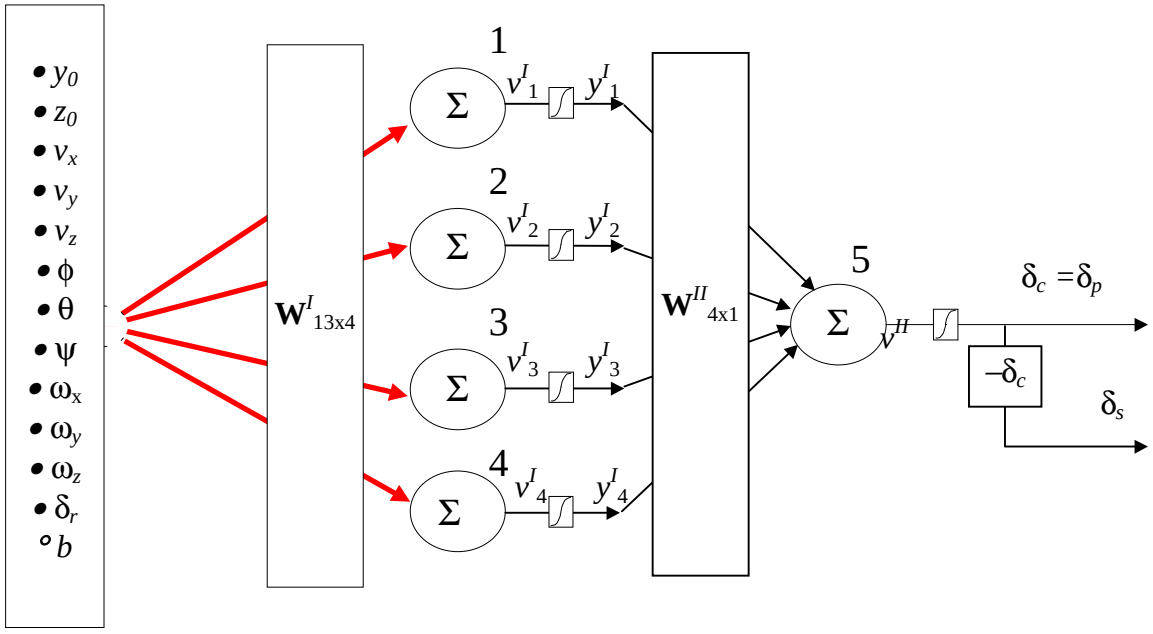
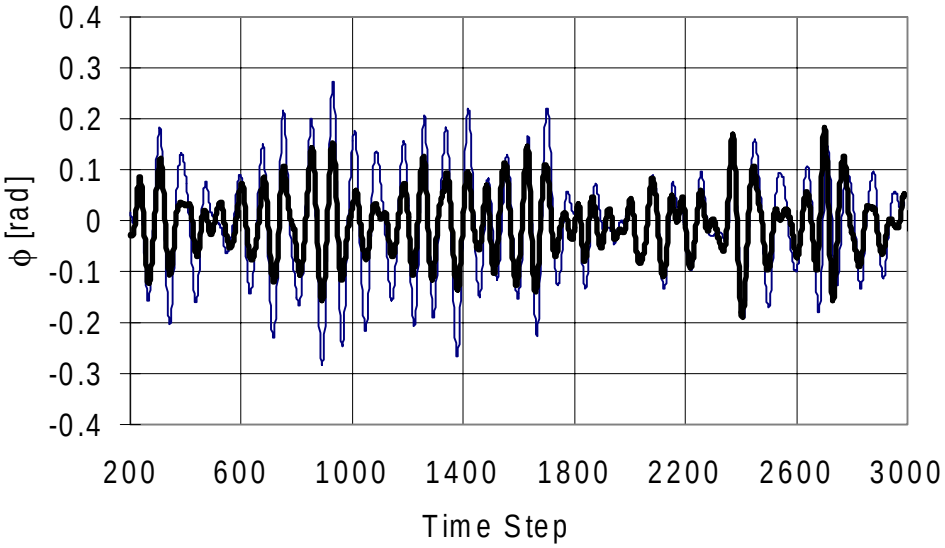
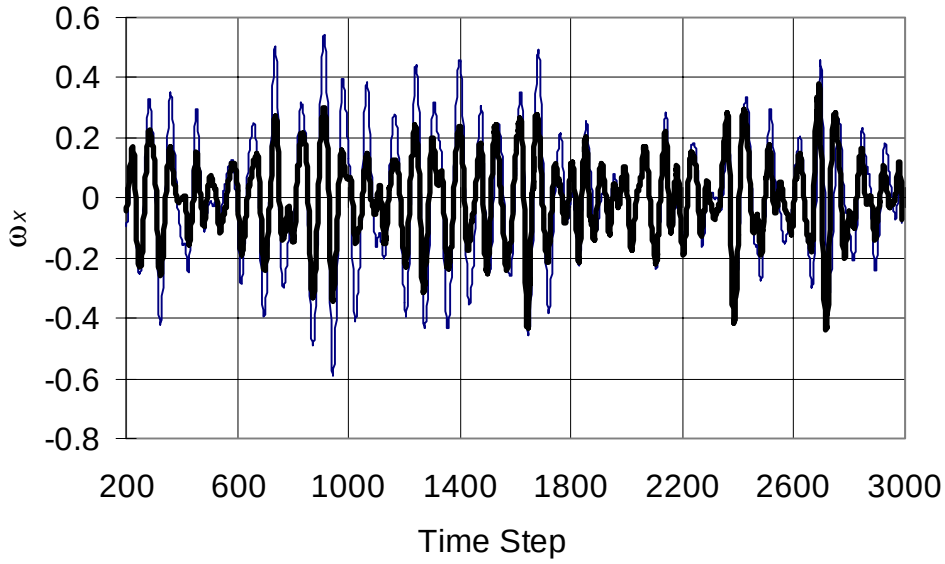


Figure 5-4: Neural-network controller used in the present example. Thick arrows account for a set of axons with origin at each input data. Squashing functions are represented by $\int$.

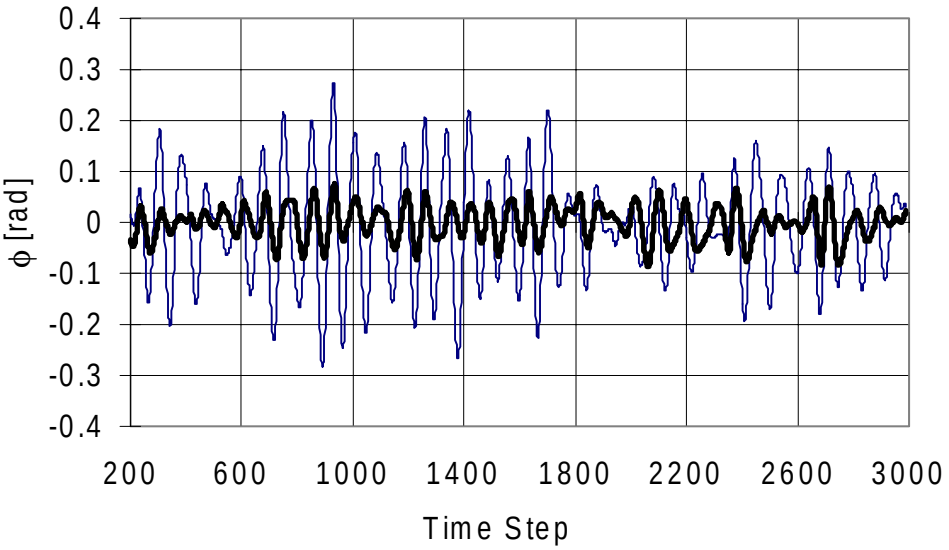


(a)

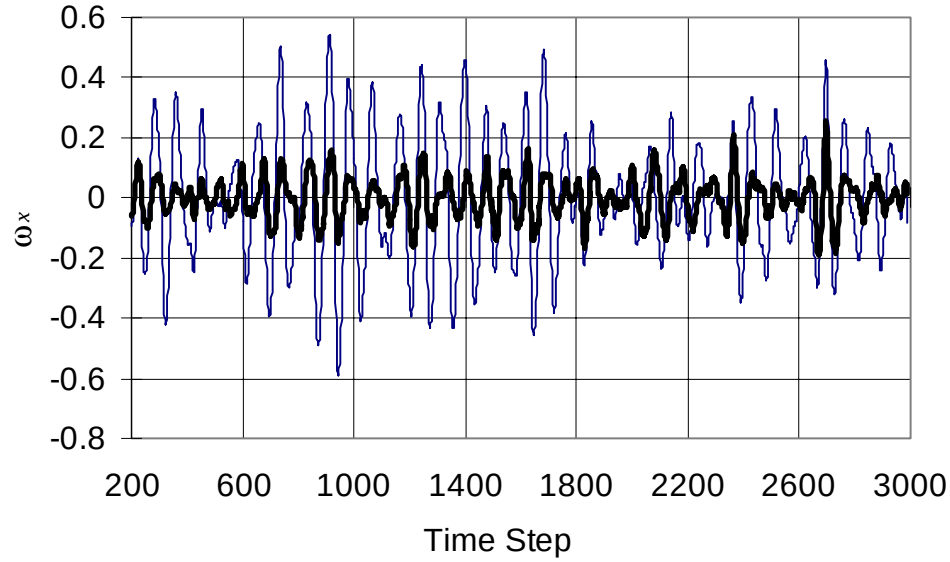


(b)

Figure 5-5: (a) Roll angle as a function of time. (b) Roll rate as a function of time. Thin curve: no fins. Thick curve: passive fins. Random beam sea.



(a)



(b)

Figure 5-6: (a) Roll angle as a function of time. (b) Roll rate as a function of time. Thin curves: no fins. Thick curves: active fins. Random beam sea.

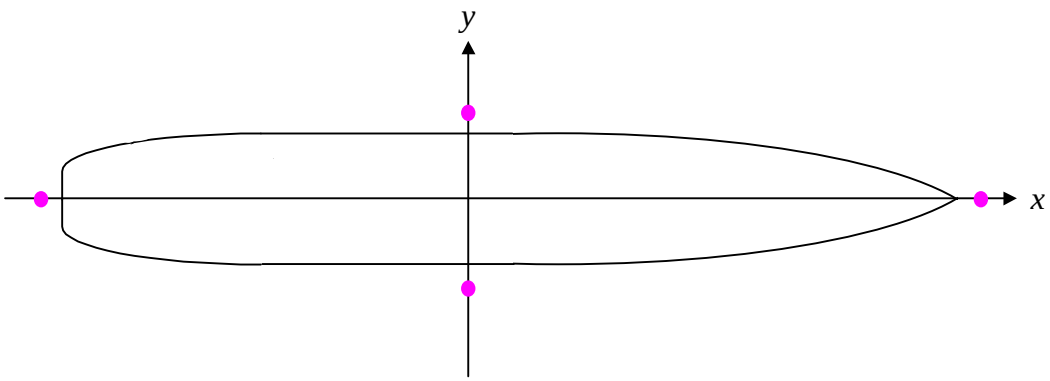
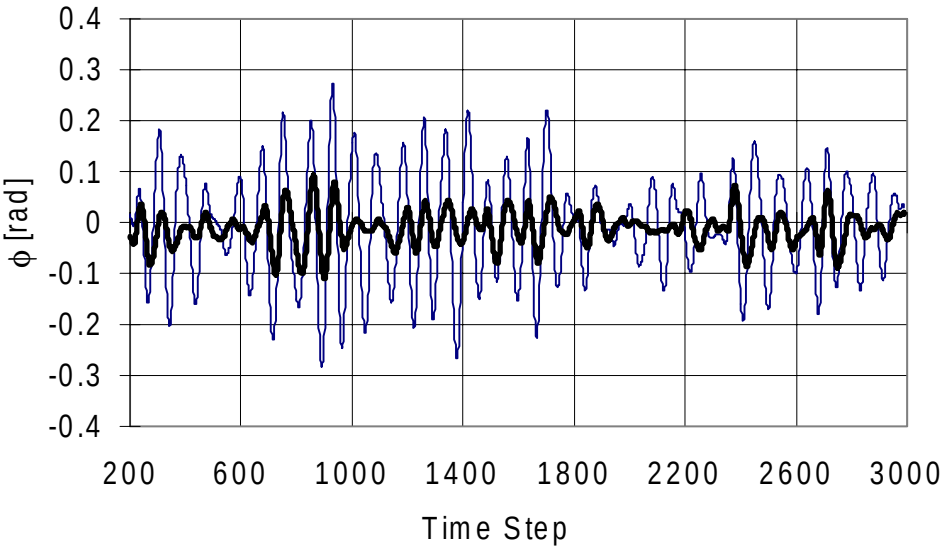
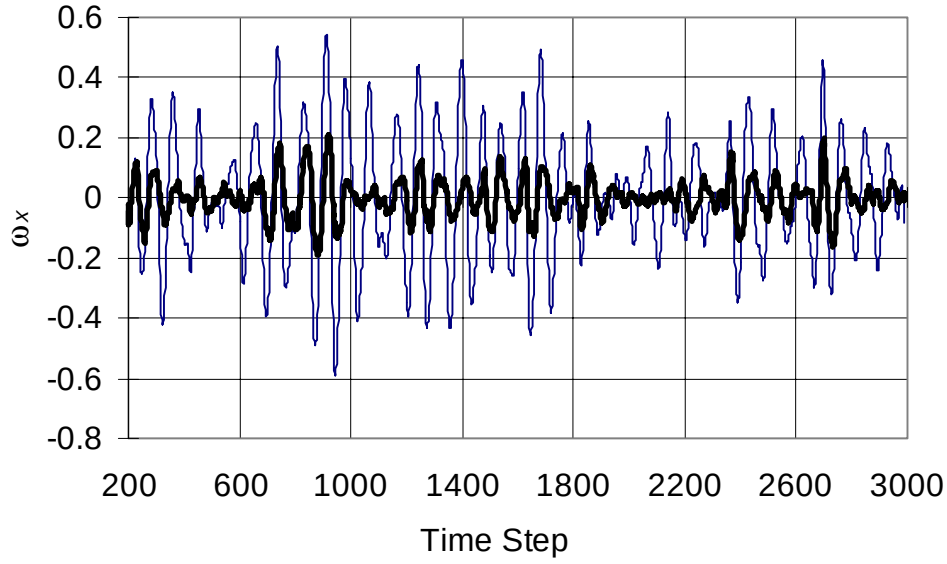


Figure 5-7: Points where sea-surface heights close to the hull are measured (•).

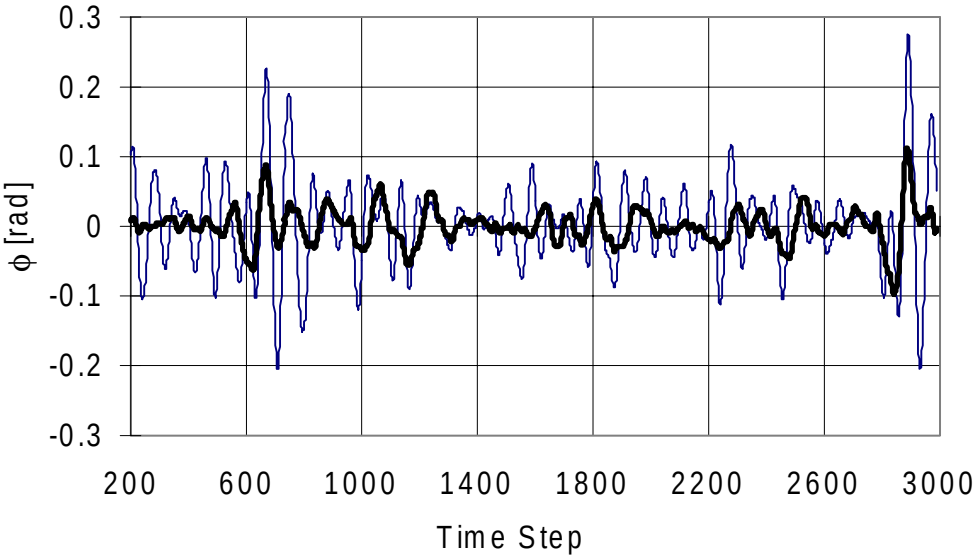


(a)

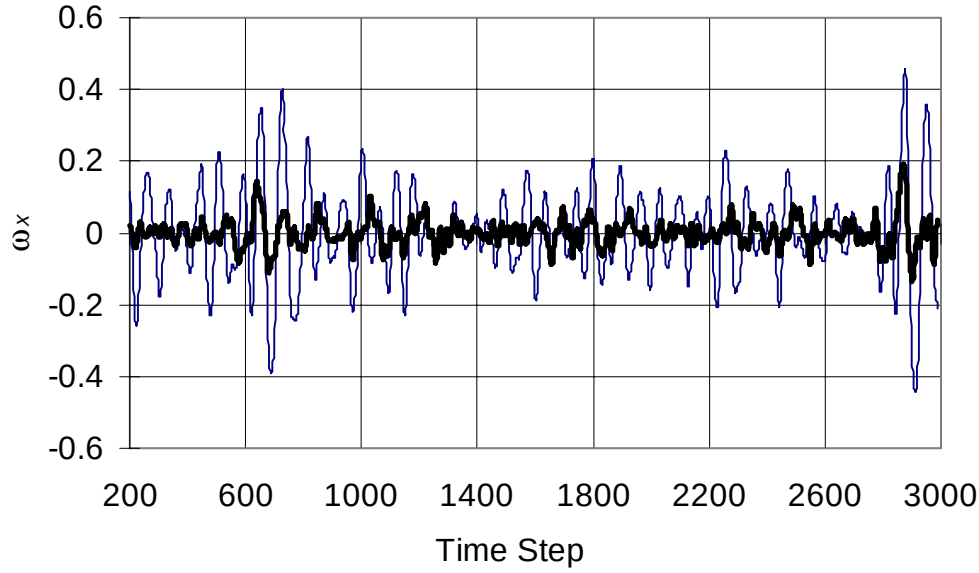


(b)

Figure 5-8: (a) Roll angle as a function of time. (b) Roll rate as a function of time. Thin curves: no control. Thick curves: active fins. Sea-surface information available for the controller. Random beam sea.

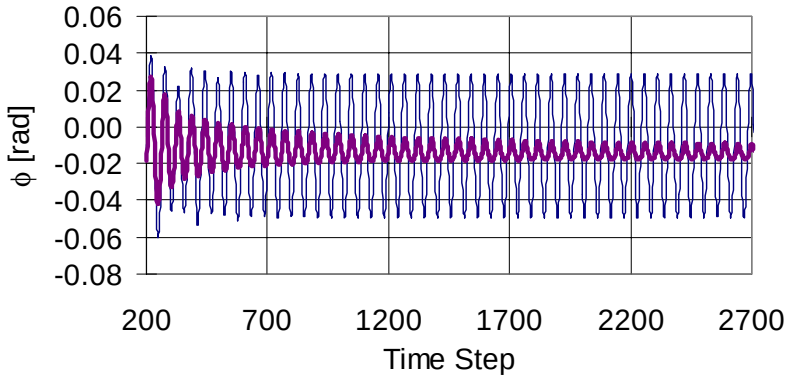


(a)

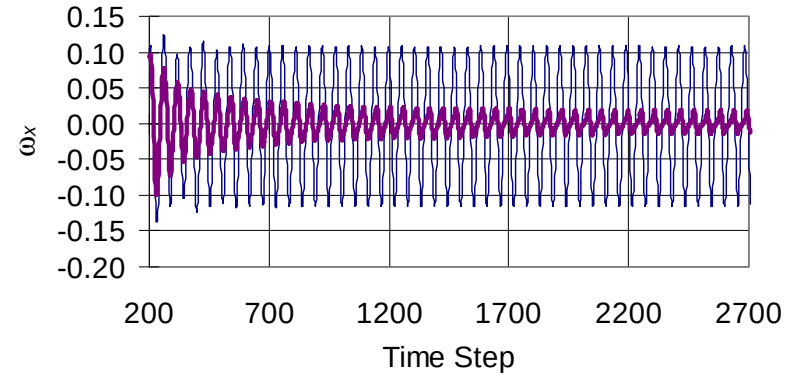


(b)

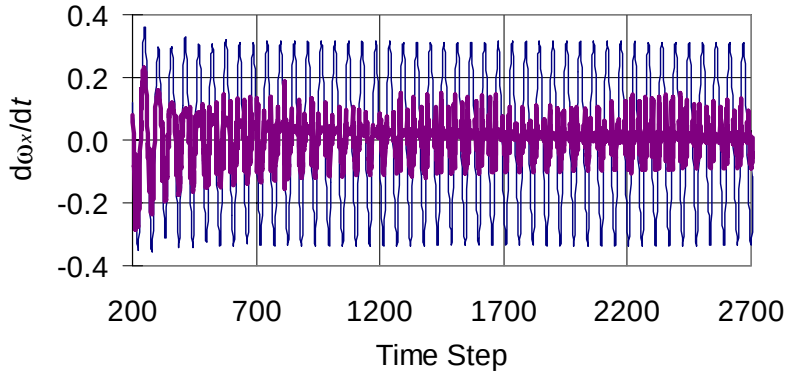
Figure 5-9: (a) Roll angle as a function of time. (b) Roll rate as a function of time. Thin curves: no control. Thick curves: active fins. Sea-surface information available for the controller; 45 degrees incoming random sea.



(a)



(b)



(c)

Figure 5-10: (a) Roll angle as a function of time. (b) Roll rate as a function of time. (c) Roll acceleration as a function of time. Thin curves: no fins. Thick curves: active fins. Sea-surface information available for the controller; 45 degrees incoming regular sea.