Control of the Archimedes Wave Swing using Neural Networks

Pedro Beirão[1], Mário J. G. C. Mendes[2], Duarte Valério[3] and José Sá da Costa[3]

[1] Instituto Superior de Engenharia de Coimbra
Dept. of Mechanical Engineering
Rua Pedro Nunes, 3030-199 Coimbra, Portugal
E-mail: pbeirao@isec.pt

[2] IDMEC / ISEL – Instituto Superior de Engenharia de Lisboa
Polytechnic Institute of Lisbon, Dept. of Mechanical Engineering
Rua Conselheiro Emídio Navarro 1, 1959-007 Lisboa, Portugal
E-mail: mmendes@dem.isel.ipl.pt

[3] IDMEC / IST, Technical University of Lisbon
Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal
E-mail: {dvalerio,sadacosta}@dem.ist.utl.pt

Abstract

This paper addresses the control of the Archimedes Wave Swing, a fully-submerged Wave Energy Converter (WEC), of which a prototype has already been built and tested. Simulation results are presented in which Internal Model Control (IMC) is used, both with linear models and with non-linear neural network (NN) models. To the best of our knowledge this is the first time NN-based control is being applied to design a controller for a WEC. NNs are a mathematical tool suitable to model the behaviour of dynamic systems, both linear and non-linear (as in our case). Significant absorbed wave energy increases were found, both using linear models and NNs. Results were better when IMC with NNs was employed (with a nearly sixfold increase against a fivefold increase), except for the May—September period, when IMC with linear models performs better.

Keywords: Neural Networks, Wave Energy, Internal Model Control, Archimedes Wave Swing, Control.

Nomenclature

Latin characters

A, B	=	Pierson-Moskowitz's spectrum parameters
b, b_{wj}, b_{vi}	=	neuron bias
C	=	controller
c	=	context unit
d	=	disturbance
e	=	error
F	=	filter
f	=	activation function
f_{exc}	=	wave excitation force
f_{lg}	=	force exerted by the ELG
f_u	=	control force
G	=	plant
G^*	=	inverse of model G'
G'	=	model of plant G
H_s	=	significant wave height
i	=	$\sqrt{-1}$
M	=	number of NN outputs
m	=	mass
n	=	number of NN inputs
p	=	number of points
R	=	resistance
S	=	stiffness
S_w	=	wave energy spectrum
s	=	Laplace transform variable
t	=	time
T_e	=	wave energy period
v_{ij}, w_i, w_{jk}	=	weight
W_u	=	absorbed wave energy
X	=	reactance
x	=	NN input
y	=	NN output
$\mathbf{y}$	=	vector of expected outputs y
z^{-1}	=	delay operator

Greek characters

α	=	optimum condition coefficient
ζ	=	neuron output
ξ	=	floater's vertical position
σ^2	=	variance
ω	=	frequency

Acronyms

AWS	=	Archimedes Wave Swing

© Proceedings of the 7th European Wave and Tidal Energy Conference, Porto, Portugal, 2007

ELG	=	Electrical Linear Generator
IMC	=	Internal Model Control
LRN	=	Layered-Recurrent (neural) Network
NN	=	Neural Network
NNARX	=	NN Auto-Regressive eXogenous (model)
RMS	=	Root Mean Square
TDM	=	AWS Time Domain Model simulator (for Matlab)
VAF	=	Variance Accounted For
WEC	=	Wave Energy Converter

Superscripts

$\tilde{x}$	=	NN estimate of variable x
$\hat{x}$	=	phasor of variable x
x^*	=	complex conjugate of variable x

Introduction

Several different sources of energy related to the sea can be distinguished, such as tidal energy, offshore wind energy, or wave energy [1–4]. In this last case, it is aimed to convert the energy present in sea waves into electricity. Sea waves are in fact a concentrated form of wind energy, which in turn is a concentrated form of solar energy, since it is the sun that causes the variations of atmospherical temperature that originate the wind.

Figure 1: The AWS before submersion

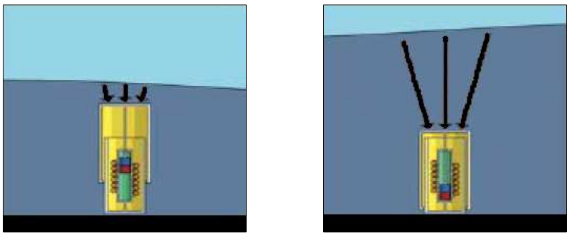

Figure 2: AWS working principle

Harnessing this energy, however, requires solving several engineering problems. The power of sea waves changes with time, and this means three things. Firstly, the energy extracted will have seasonal variations. Secondly, since short-term variations of wave energy are important, there must be some means of ensuring that electricity produced will be synchronous and phase-locked. Thirdly, peak values of wave energy may be so high as to pose a danger to the device. Because of this, and because the sea is by its very nature an aggressive environment, devices for producing electricity from sea waves (known as wave energy converters, WECs) must have a very resistent design.

There are many possible configurations for WECs. Some are installed at the shoreline, some are deployed near the shore, others are deployed offshore; they may float or be submerged; and there are several possible working principles. To this day, several promising prototypes have been built. Control engineering plays an important role in on-going research, to maximise the energy extracted and hence the rentability of the WEC.

In this paper we concentrate upon a particular WEC, the Archimedes Wave Swing (AWS). Our purpose is to present simulation results when Internal Model Control (IMC) is used, both with a linear model and with a non-linear neural network (NN) model. The paper is structured as follows: section 1 gives information about the AWS; section 2 briefly describes the principles of IMC; section 3 introduces neural networks; AWS modelling with neural networks is the subject of section 4; the implementation of control algorithms is given in section 5; then section 6 presents simulation results; finally, conclusions are drawn in section 7.

1 The AWS

The AWS is one of the several WECs currently under development (see Fig. 1). It is an offshore, underwater (43 m) device. It is a point absorber, that is to say, its size is neglectable compared to a typical wave length. It consists mainly in a cylindrical, bottom-fixed chamber (the silo), covered by a movable, cylindrical lid (the floater). Air is enclosed between the silo and the floater. As the wave crests and troughs pass, the height of the water column above the floater varies, and so does the resulting pressure. Thus, and since the air enclosed within is compressible, the floater will heave (see Fig. 2 for an illustration of this vertical movement). An electrical linear generator (ELG) converts this reciprocating movement into electricity [5].

From this description, it may be supposed that a mass–spring–damper system will provide a simple but reasonable dynamic model for the AWS. In reality, things are not so simple, since there are many non-linearities. A very complete and detailed non-linear simulator, the AWS Time Domain Model (AWS TDM), has been developed for Matlab [6–8]. But it is convenient, for control design purposes, at least in an initial stage, to have a linear model, even if less accurate. Fig. 3 shows that the non-linearities, though not neglectable, are not so important as to make such a linear model useless. This linear model has been obtained from data provided by the AWS TDM, used as an emulator of the AWS protoptype. This option was taken because, due to operational problems, very few experimental data is available from the 2 MW AWS prototype, which was tested at the Portuguese northern coast during 2004 (and then decommissioned). Additionally, due to industrial secrecy reasons, several parameters of the AWS TDM have been modified to provide the data upon which the linear model given below is based, as well as in all simulations in this paper.

To obtain the data, the AWS TDM was fed with several sinusoidal waves with an amplitude of 1.0 m. Since the AWS is a non-linear dynamic system, changing the amplitude of the waves will change (even if only slightly) the result obtained. This particular wave amplitude was chosen since it is the most frequent one in the location where the AWS prototype was deployed; data on wave climate was obtained from the ONDATLAS software [9][1]. The periods of the waves range from 4 s to

[1] The AWS was tested 5 km offshore Leixões. The location for

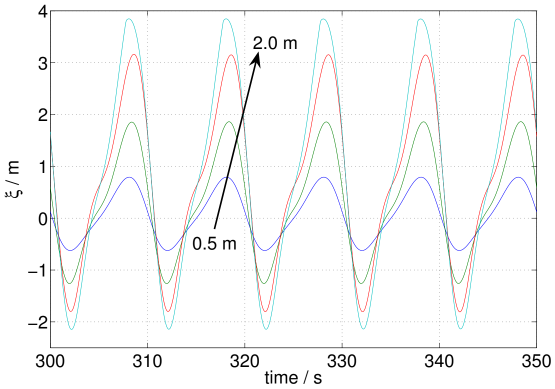

Figure 3: Output of the AWS TDM for regular waves with 10 s of period and amplitudes of 0.5 m, 1.0 m, 1.5 m and 2.0 m

14 s. These are the wave periods expected to occur at the AWS test site. The input of the desired linear model is the sum of the wave excitation force f_{exc} (that is to say, the force that the waves would exert in the AWS if it were stopped) and of the force exerted by the ELG f_{lg}. The output is the vertical position of the floater ξ.

Levy's identification method was applied to this data [5]. The identified linear model is

$$\frac{\Xi(s)}{F_{exc}(s) + F_{lg}(s)} = \frac{2.259 \times 10^{-6}}{0.6324s^2 + 0.1733s + 1} \qquad (1)$$

(Notice that lower-case letters are used for variables in the time domain, and capital letters for their Laplace transforms. Hence $F_{exc}(s) = \mathcal{L}[f_{exc}(t)]$, and so on.) For more details on this subject, see [10].

2 Internal model control

The IMC methodology [11] makes use of the control scheme of Fig. 4. In that control loop, G is the plant to control, G' is a model of G, G^* is an inverse of G' (or at least a plant as close as possible to the inverse of G'), and F is some judiciously chosen filter. If G' were exact, the error e would be equal to disturbance d. If, additionally, G^* were the exact inverse of G' (and hence also of G) and F were unity, control would be perfect. Since no models are perfect, the error will not be exactly the disturbance. That is also exactly why F exists and is usually a low-pass filter: to reduce the influence of high-frequency modelling errors. It also helps ensuring that product FG^* is realisable.

The interconnections of Fig. 4 are equivalent to those of Fig. 5 if the controller C is given by

$$C = \frac{FG^*}{1 - FG^*G'} \qquad (2)$$

In sections 4 to 6, IMC will be applied to the AWS, both using the linear model (1) and using neural network models [12].

3 Neural networks

In the 1940s and 50s attempts were made to find effective mathematical models of how the human nervous system works. This led to the development of (artificial) NNs. In the end these proved to be too simple to model the human brain, since they embed too many simplifying assumptions to accurately mirror what actually happens in our brain cells. But NNs proved to be an efficient means of modelling a system, and models obtained can be both linear or non-linear, static or dynamic. They can be better thought of as a biologically inspired means of building an approximation to some desired function. Their major usefulness comes from algorithms that optimise their parameters to mimic as accurately as possible any input-output data we may want. Hence the type of model we end up with is a black-box model: a model that (hopefully) does what we want, without any analytical tuning of the function parameters involved [13–15].

The main unit of a NN is (because of the biological analogy) called neuron. A neuron is in reality a real single-valued function depending of several inputs. Several functions, termed activation functions, are used in practice in the neurons; the most usual are a linear function, the Heaviside function (or a variation thereof), and the hyperbolic tangent function (or some other S-shaped function). Refer to the inset of Fig. 6 for the architecture of a neuron. In that figure, the two inputs are variables x_1 and x_2 (in the general case there might be any number of inputs), the activation function is f, ζ is the output, the w_i are weights, and b is a bias term. For the three activation functions mentioned above, and for n inputs, the output will be given by

$$\text{Linear:} \qquad \zeta = b + \sum_{k=1}^{n} w_k x_k \qquad (3)$$

$$\text{Heaviside function:} \qquad \zeta = \begin{cases} 1, \text{ if } b + \sum_{k=1}^{n} w_k x_k \geq 0 \\ 0, \text{ if } b + \sum_{k=1}^{n} w_k x_k < 0 \end{cases} \qquad (4)$$

$$\text{Hyperbolic tangent:} \qquad \zeta = \tanh\left(b + \sum_{k=1}^{n} w_k x_k\right) \qquad (5)$$

and similarly for other activation functions.

NNs consist of several neurons interconnected. In the most widespread architecture of NNs, neurons are disposed in several layers, each one feeding the next; the layers of neurons make up the NN. (Other network architectures are possible but unfrequent.) The inputs of the first layer and the outputs of the last layer are the inputs and outputs of the NN. Neuron layers before the last are called hidden layers. The several interconnections between neurons are called synapses, because of the biological analogy.

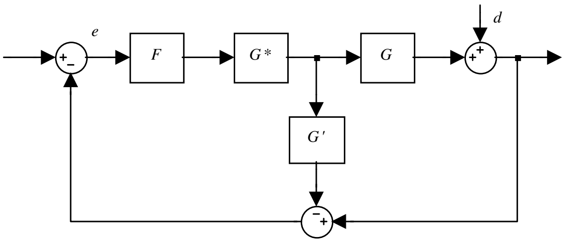

Figure 4: Block diagram for Internal Model Control

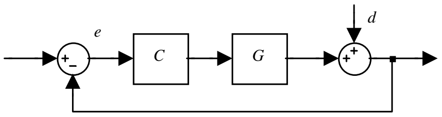

Figure 5: Block diagram equivalent to that of Fig. 4

which ONDATLAS supplies wave climate data is called Leixões-buoy (41°12.2′ N, 9°5.3′ W).

A possible layered NN is shown in Fig. 6 (where there happen to be two inputs and two outputs), or in some similar pattern. This is a feedforward layered NN, because layer outputs are never fed back in the network. The output y of such a NN, with only one hidden layer, is a vector with M components ($M = 2$ in the particular case of Fig. 6) that are determined in terms of the n components of the input vector x by the formula:

$$y_i = f_2 \left\{ \sum_{j=1}^{N_h} \left[b_{vi} + v_{ij} f_1 \left(b_{wj} + \sum_{k=1}^{n} w_{jk} x_k \right) \right] \right\}, \quad i = 1, \ldots, M \qquad (6)$$

where N_h is the number of hidden layer neurons ($N_h = 4$ in Fig. 6), w_{jk} are the first-layer interconnection weights (the first subscript index referring to the neuron, and the second to the input), v_{ij} are the second-layer interconnection weights, b_{wj} and b_{vi} are the threshold offsets (biases), and f_1 and f_2 are the activation functions of the two neuron layers. (It is supposed that all neurons in a layer use the same activation function.)

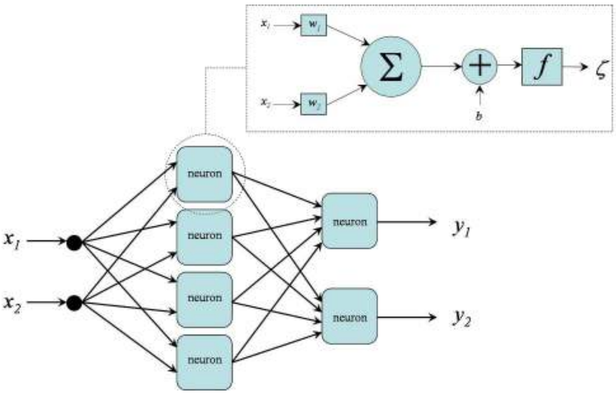

Figure 6: Scheme of a neural network

It is possible to include delays in the interconnections, thus obtaining dynamic models [16]. These will be discrete in time, with a constant sampling frequency. One of the possible architectures with delays is depicted in Fig. 7 (where there happen to be only one input and one output, with one delay in each). In this architecture, the network outputs are fed back as inputs. Models thus obtained are called neural network auto-regressive exogenous models (NNARX). Again, several variations are possible.

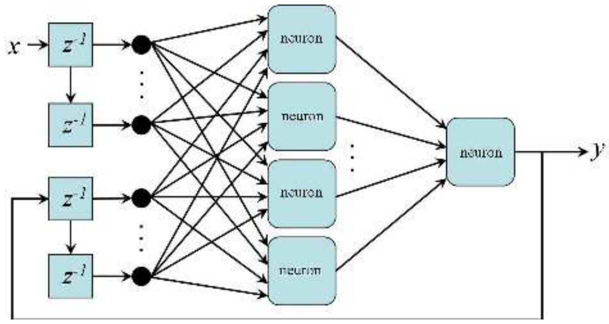

Figure 7: Scheme of a NNARX dynamic neural network

Another possible architecture for modelling dynamic systems, depicted in Fig. 8, feeds layer outputs back as inputs to that layer (save for the last layer, the outputs of which are not fed back). The delayed feedbacks are known as context units and are denoted by c in Fig. 8. Such NNs are called Elman neural networks if they have only two layers, the first using $\tanh$ activation functions and the last using linear activation functions [17]. Similar networks with different functions and/or more layers are called layered-recurrent (neural) networks (LRN). Clearly, Elman NNs are particular cases of LRN NNs.

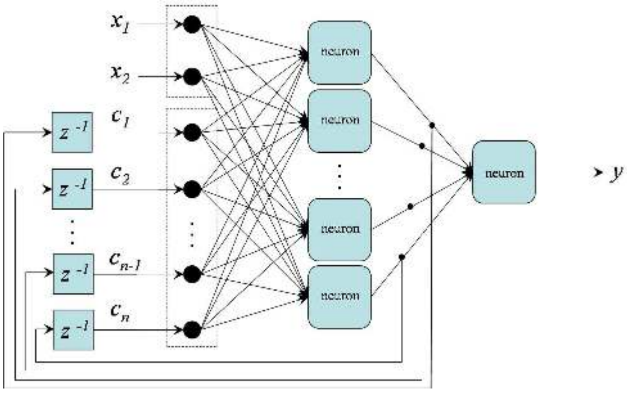

Figure 8: Scheme of an Elman dynamic neural network

There are some particular architectures known to provide good results for specific cases, and several rules of thumb for deciding before-hand how many layers, neurons and delays to use. These are given in the references (see for instance [14, 18]).

Finally, what is needed is an algorithm to find the parameters (weights and biases) that will make our NN have the behaviour we want. There are several possibilities; backpropagation is one of them. In few words, this algorithm is given some input-output data showing how the system behaves, and then varies the network's parameters trying to minimise the difference between its outputs for those inputs and the outputs is should be giving. The derivative (often numerically computed) of a suitable performance function is employed. We will not enter into further (mathematical) details, which may once more be easily found in the references. Let it suffice to say that this method heavily relies on the data the algorithm is fed with (the training data). Too few data may be less than enough to train the network properly, while too much data may overtrain it so that it will just reproduce the training set while being unable to give meaningful outputs for new situations. The data should also reflect all possible working conditions of the system. Additionally, there are several possible variations of backpropagation. In what follows, backpropagation together with the Levenberg-Marquardt algorithm was used [19].

The above is of course a very simplified depiction of NNs. For more details, see the references given.

4 Neural network models of the AWS

From the description above it is seen that NNs may provide a suitable tool for modelling the AWS, especially because of the ability of modelling non-linearities (both hard non-linearities, as is the case of all saturations in general, and in particular of the effect of the mechanical end-stops that limit the heaving of the floater; and soft non-linearities, as is the case of several continuous, differentiable non-linear relations between variables). To model the AWS using NNs, the wave excitation force f_{exc} and the force exerted by the ELG f_{lg} were selected as inputs, and the floater's vertical velocity $\dot{\xi}$ as output. For the inverse model, the inputs were f_{exc} and $\dot{\xi}$, and the output f_{lg}. f_{exc} is always considered an input since it is not created by the AWS; it is better

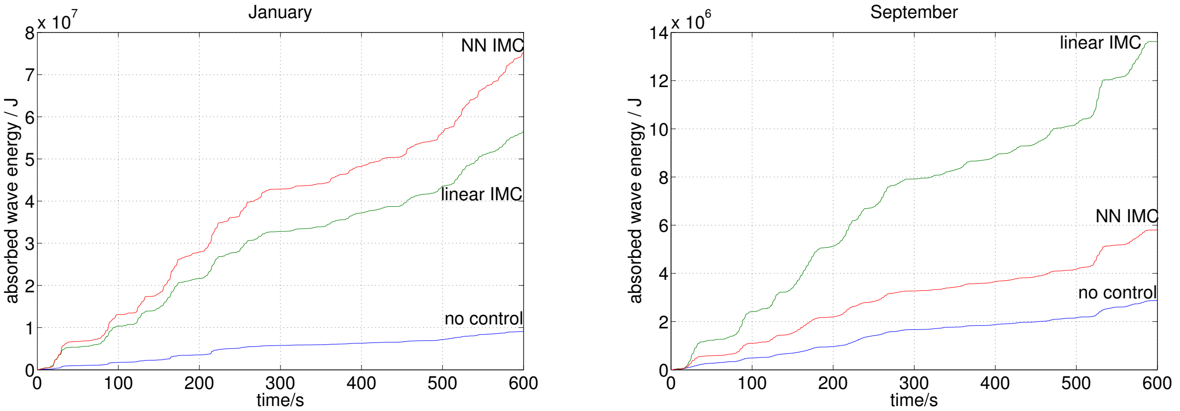

Figure 11: Wave energy absorption (figurative data)

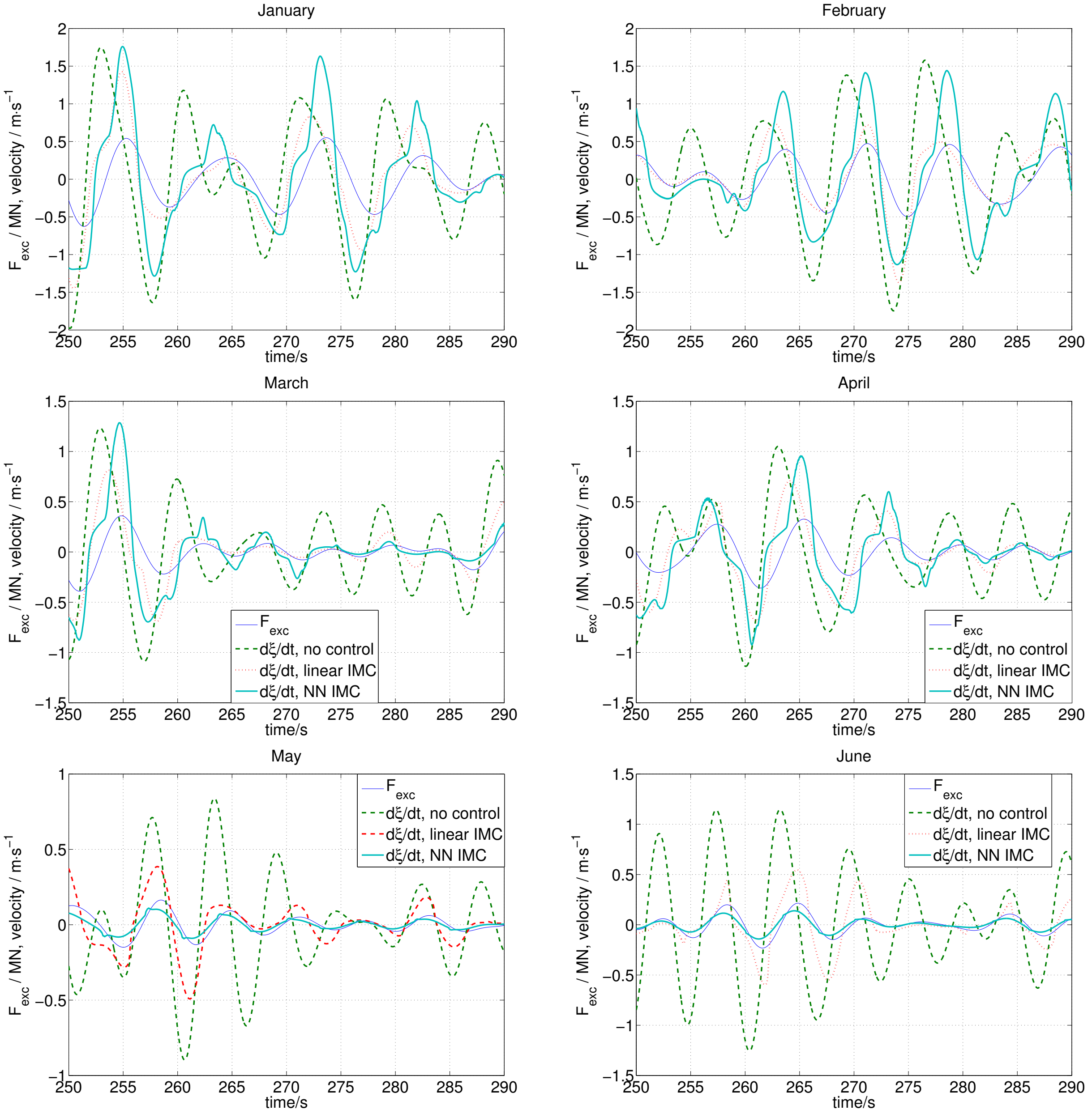

Figure 12: Wave excitation force and floater’s vertical velocity

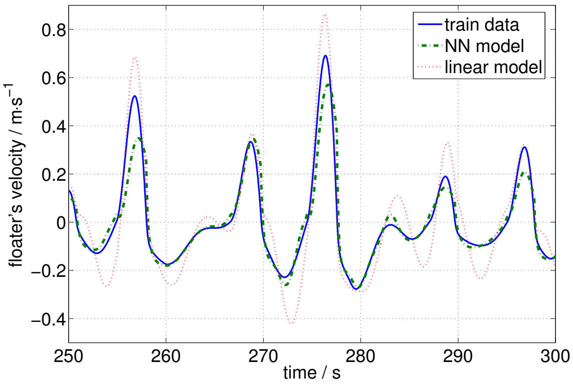

Figure 9: Performance of the direct (NN and linear) models

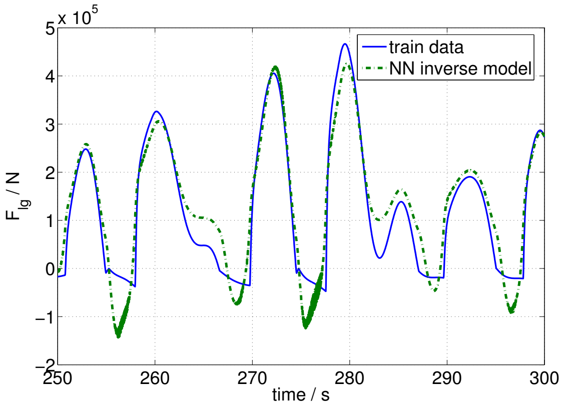

Figure 10: Performance of the inverse NN model

viewed as a perturbation. In each case two separate inputs had to be considered; models relating solely the floater's vertical velocity (or position) with only one force (similarly to the linear model (1)) were built, but were unable to provide any accurate results. On the other hand, two input variables sufficed to obtain good results, and hence it was not necessary to include further inputs, such as other forces applied in the floater.

To serve as input, 600 s long irregular (non-sinusoidal) waves were generated. All irregular waves mentioned in this paper follow Pierson-Moskowitz's spectrum, that accurately models the behaviour of real sea waves [20]. This spectrum is given by

$$S_w(\omega) = \frac{A}{\omega^5}\,\exp\left(-\frac{B}{\omega^4}\right) \tag{7}$$

where ω is the frequency and S_w is the wave energy spectrum, that is, a function such that

$$\int_0^{+\infty} S_w(\omega)d\omega \tag{8}$$

is the mean-square value of the wave elevation). Table 1 gives values for the significant wave height H_s (from trough to crest) and for the limits of the frequency range (corresponding to the maximum and minimum values of the wave energy period T_e) provided by ONDATLAS for the twelve months of the year and for the whole year. In (7), the numerical values $A = 0.780$ (SI) and $B = 3.11/H_s^2$ were used.

The AWS TDM uses a sampling time of 0.02 s. Hence a variable recorded for 600 s corresponds to a 30000 long vector. This proved to be too much data for training a network; hence, only data from 150 s to 400 s was used in training. This piece was found to be manageable, and yet to include sufficient data for the training. The whole 600 s long series were used for validating the identification.

The option that proved to be the best was to train the NN with an irregular wave corresponding to the wave parameters (H_s, T_e) for the whole year. (These are given in the last column of Table 1.) Since it is during winter months that more wave energy is available, it could have been convenient (to maximise wave energy absorption) to use models very well fitted for winter months, even if their performance would be poorer during the rest of the year. But in the end simulations proved that better results would be obtained with NNs trained with the wave parameters for the whole year.

NNARX, Elman and LRN architectures were used. Elman NNs proved to perform better than NNARX NNs; LRN NNs provided no additional benefit, and, since they are more complicated than Elman NNs, were not chosen. The number of neurons in the hidden layer was found to be an important factor. Elman NNs could usually model the AWS train data accurately, irrespective of the number of neurons. But NNs with too few or with too many neurons were less able to generalise the modelling to other data. (What too few or too many neurons were had to be found out by trial and error.) Closely related to this was the extent to which training was performed, since, as mentioned above, performing the train algorithm for too long might lead to overtraining, and hence to poor performance for inputs other than those used when training; while at the same time stopping the train algorithm too soon leads to NNs that are still not a good model.

After some trial and error (sometimes necessary in similar cases), two NNs were chosen, with the characteristics given in Table 2. In that table, VAF (variance accounted for) is a performance parameter computed as

$$VAF = 1 - \frac{\sigma^2(\tilde{\mathbf{y}} - \mathbf{y})}{\sigma^2(\tilde{\mathbf{y}})} \tag{9}$$

where $\mathbf{y}$ is the vector of expected outputs (collecting successive time-varying values of y) and $\tilde{\mathbf{y}}$ the vector of NN's outputs; and RMS (root mean square) is another performance parameter computed as

$$RMS = \sqrt{\frac{\sum_{k=1}^{p}(\tilde{y}_k - y_k)^2}{p}} \tag{10}$$

where p is the number of points (30000, since, as mentioned above, the whole 600 s long series was used for computing these indexes). The Table also gives those indexes for model (1).

Fig. 9 shows how the direct NN model performs and compares its output to that of linear model (1). Fig. 10 shows how the inverse NN model performs; no comparison can be presented with the output of the inverse of the linear model, since that inverse is not causal. In both cases, the data used is a 50 s slice of the test wave with which VAF and RMS were reckoned; after 250 s, the transient response at the beginning of the simulation has long passed away. It can be seen that models are fairly accurate, and that the NN direct model is able to follow the output reference more accurately than the linear model can. The NN inverse model sometimes originates chattering in its output. This deficiency, however, will prove to be unimportant for results (probably because the filter F chosen to implement IMC is a low-pass filter, as shall be seen below in the next section).

Table 1: Characteristics of several irregular waves according to ONDATLAS

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Whole year
H_s / m	3.2	3.0	2.6	2.5	1.8	1.7	1.5	1.6	1.9	2.3	2.8	3.1	2.3
$T_{e,min}$ / s	5.8	5.8	5.2	5.5	5.0	4.7	4.6	5.0	5.2	5.3	5.5	5.3	4.6
$T_{e,max}$ / s	16.1	14.5	13.7	14.8	12.2	9.7	11.1	10.5	12.0	12.6	13.3	14.2	16.1

Table 2: Neural network models of the AWS

	Linear direct model (1)	NN direct model	NN inverse model
Inputs	$f_{exc} + f_{lg}$	f_{exc}, f_{lg}	f_{exc}, $\dot{\xi}$
Outputs	$\dot{\xi}$	$\dot{\xi}$	f_{lg}
Type	Transfer function	Elman	Elman
Hidden layer	—	8 neurons	10 neurons
VAF	83.9778 %	90.9624 %	91.9242 %
RMS	0.0994	0.0645	0.0798
Training	—	7 epochs	5 epochs

5 Control implementation

5.1 Phase and amplitude control

If the dynamics of the AWS were given exactly by (1), maximising the power absorbed from the waves would require the floater's vertical velocity to be in phase with the wave excitation force acting thereupon. The remainder of this subsection consists in an outline of the proof, closely following [20]. Readers therein uninterested can skip directly to 5.2.

A general mass-spring-damper system corresponds to a dynamic behaviour given by

$$m\ddot{\xi}(t) + R\dot{\xi}(t) + S\xi(t) = f_{exc}(t) \qquad (11)$$

Parameters m (mass), R (resistance) and S (stiffness) are positive. Assuming that f_{exc} and $\dot{\xi}$ are sinusoidal with time, and defining complex-valued phasors $\hat{f}_{exc}$ and $\hat{\dot{\xi}}$,

$$\begin{aligned} f_{exc}(t) &= \frac{\hat{f}_{exc}}{2}e^{i\omega t} + \frac{\hat{f}^*_{exc}}{2}e^{-i\omega t} \qquad (12) \\ \dot{\xi}(t) &= \frac{\hat{\dot{\xi}}}{2}e^{i\omega t} + \frac{\hat{\dot{\xi}}^*}{2}e^{-i\omega t} \qquad (13) \end{aligned}$$

equation (11) becomes

$$\begin{aligned} & e^{i\omega t}\left[\hat{f}_{exc} - \left(R + i\omega m + \frac{S}{i\omega}\right)\hat{\dot{\xi}}\right] \\ & + e^{-i\omega t}\left[\hat{f}^*_{exc} - \left(R - i\omega m - \frac{S}{i\omega}\right)\hat{\dot{\xi}}^*\right] = 0 \qquad (14) \end{aligned}$$

Defining an impedance $Z = R + i\left(\omega m - \frac{S}{\omega}\right)$, expression (14) can be rewritten as

$$e^{i\omega t}\left(\hat{f}_{exc} - Z\hat{\dot{\xi}}\right) + e^{-i\omega t}\left(\hat{f}^*_{exc} - Z^*\hat{\dot{\xi}}^*\right) = 0 \qquad (15)$$

For (15) to be satisfied for all values of time t, condition

$$\hat{\dot{\xi}} = \frac{\hat{f}_{exc}}{Z} \Rightarrow \left|\hat{\dot{\xi}}\right| = \frac{\left|\hat{f}_{exc}\right|}{|Z|} \qquad (16)$$

must be verified.

The impedance can be rewritten as $Z = R + iX$. The real part $R = \text{Re}[Z]$ is called resistance and the imaginary part $X = \text{Im}[Z] = \omega m - \frac{S}{\omega}$ is called reactance. Suppose now that a control force f_u is applied to the AWS. This will be needed to ensure that the conditions leading to maximum wave energy absorption (or at least conditions as close as possible to those) are met. Then

$$Z(\omega)\dot{\Xi}(\omega) = F_{exc}(\omega) + F_u(\omega) \qquad (17)$$

The absorbed wave energy W_u can be given by

$$W_u = -\int_0^{+\infty} f_u(t)\dot{\xi}(t)dt \qquad (18)$$

Considering that f_u and $\dot{\xi}$ are real functions, i.e., $F^*_u(\omega) = F_u(-\omega)$ and $\Xi^*(\omega) = \Xi(-\omega)$, by applying Parseval's theorem, W_u can be given by

$$W_u = -\frac{1}{2\pi}\int_{-\infty}^{+\infty}\left[F_u(\omega)\dot{\Xi}^*(\omega)\right]d\omega \qquad (19)$$

Knowing that W_u is real,

$$\begin{aligned} F_u(\omega)\dot{\Xi}^*(\omega) &= \text{Re}\left[F_u(\omega)\dot{\Xi}^*(\omega)\right] = \\ &= \frac{1}{2}\left[F_u(\omega)\dot{\Xi}^*(\omega) + F^*_u(\omega)\dot{\Xi}(\omega)\right] \qquad (20) \end{aligned}$$

expression (19) can be rewritten as

$$W_u = \frac{1}{2\pi}\int_0^{+\infty}\left[-F_u(\omega)\dot{\Xi}^*(\omega) - F^*_u(\omega)\dot{\Xi}(\omega)\right]d\omega \qquad (21)$$

It will be convenient to add and subtract the term $\frac{F_{exc}(\omega)F^*_{exc}(\omega)}{2R}$ to the integrand of (21), and finally W_u is now given by (omitting the frequency argument)

$$\begin{aligned} W_u &= \frac{1}{2\pi}\int_0^{+\infty}\left[\frac{|F_{exc}|^2}{2R}\right. \\ &\qquad \left. -\frac{|F_{exc}|^2}{2R} - F_u\dot{\Xi}^* - F^*_u\dot{\Xi}\right]d\omega = \\ &= \frac{1}{2\pi}\int_0^{+\infty}\left[\frac{|F_{exc}|^2}{2R} - \frac{\alpha}{2R}\right]d\omega \qquad (22) \end{aligned}$$

where $\alpha(\omega)$, called optimum condition coefficient, is given by

$$\begin{aligned}\alpha(\omega) &= F_{exc}(\omega)F^*_{exc}(\omega) \\ &\quad + 2R\left[F_u(\omega)\dot{\Xi}^*(\omega) + F^*_u(\omega)\dot{\Xi}(\omega)\right] \quad (23)\end{aligned}$$

After some tedious manipulations, it turns out that $\alpha = \left|F_{exc}(\omega) - 2R\dot{\Xi}(\omega)\right|^2 \geq 0$. So, it is when $\alpha(\omega) = 0$ that W_u is maximal, and an optimum condition can be written as

$$F_{exc}(\omega) = 2R\dot{\Xi}(\omega) \quad (24)$$

Since $2R$ is constant, $\dot{\xi}$ must be in phase with f_{exc}.

5.2 IMC with linear models

The application of IMC with linear models to the AWS was already documented in [21]. The configuration of Fig. 4 was used together with a reference given by

$$\dot{\xi}_{\text{setpoint}} = \frac{2.2}{\max|f_{exc}|} f_{exc} \quad (25)$$

In other words, the floater's vertical velocity will be in phase with the wave excitation force. Constant 2.2 appears because the nominal value for the floater's vertical velocity that the AWS should work with is 2.2 m/s [22]. Foreknowledge of the incoming wave (and thus of f_{exc}) was assumed in all simulations, and thus the denominator of the fraction is a constant.

It should be noticed that this is not the constant in (24). Simulations with the AWS TDM have shown that (25) leads to a higher energy production. Since (24) is an optimum condition, it may seem odd that changing the constant can lead to better results. This is because (24) was derived for a linear model, such as (1). With a non-linear model, such as the AWS TDM, this condition is merely an approximation; it is no longer optimum, but the closer the linear model is from the non-linear one, the better will be the results obtained when the condition is satisfied. This is why it is possible that changes to condition (24) may improve the results.

Model G' was given by (1) multiplied by s (this additional zero at the origin serving to have the floater's vertical velocity—and not its position—as the output):

$$\frac{\dot{\Xi}(s)}{F_{exc}(s) + F_{lg}(s)} = \frac{2.259 \times 10^{-6}s}{0.6324s^2 + 0.1733s + 1} \quad (26)$$

The inverse model was $G^* = \frac{1}{G'}$. Since G^* is not causal, filter F had to have more poles than zeros. It was found by trial and error that a second order filter without zeros was the best option. The position of the poles was adjusted so as to maximise the absorbed wave energy for the simulation that uses an irregular wave with parameters corresponding to the month of March (deemed to be a significant month). The values found were

$$F = \frac{600}{(s+23)(s+20)} \quad (27)$$

This is reasonable since it corresponds to a low-pass filter that preserves the frequencies where waves are expected to appear, while cutting off higher ones.

Because of this, the product FG^* has an integral action. Since the signal it acts upon (labelled e in Fig. 4) has a residual non-null average, this lead to an ever-increasing (or ever-decreasing) control action, something that was not intended. To prevent this, the control action had to be corrected, by subtracting its average, computed from the beginning of the simulation and actualised on-line.

5.3 IMC with neural networks

Similar simulations were performed, replacing G' and G^* by the neural networks from section 4. The correction of the control signal was no longer required because the inverse model no longer has an integral action. Filter F was kept equal to (27) for two reasons: first, to make comparison easier (a different filter might be blamed for the differences in absorbed wave power); second, because it was found out that a different filter would not significantly improve results.

6 Simulation results

Table 3 gives data for the absorbed wave energy obtained when simulations are run with twelve irregular waves for the twelve months of the year. Fig. 11 shows how absorbed wave energy evolves with time for two significant months. Simulations with a duration of 600 s (10 min) were carried out, employing several incident waves, as mentioned above. In all cases, the absorbed wave energy is given by

$$W_u = \int_0^{600} -f_{lg}(t)\dot{\xi}(t)dt \quad (28)$$

Notice that f_{lg} is identified with f_u in (18). These Tables also include values for the situation when no control strategy is applied to the AWS, and the floater heaves freely. In this situation, f_{lg} has a residual value, and that is the source of the energy given. In the Tables, no comparison with the theoretical maximum obtainable by an ideal heaving WEC is given, since (as said above) the AWS model was modified, and so such a comparison is rendered pointless.

The first comment these results deserve is that absorbed wave energy increases are very significant. IMC with linear models performs best during the May—September period (a sort of extended summer); IMC with NNs performs best during the rest of the year. Actually, during summer IMC with NNs performs so poorly that in July absorbed wave energy values are *lower* than those obtained without any control at all. Nevertheless, considering the whole year, IMC with NNs is the most advisable control method.

Figures 12 and 13 give an insight into the reason why sometimes one control strategy works better than the other. They show 40 s of the data on wave excitation force and floater's velocity for all months of the year. It can be seen that, when no control strategy is used, the velocity is not in phase with the excitation force. On the other hand, IMC successfully puts these two variables in phase. This is why wave energy absorption increases. IMC with NNs is more efficient in putting the variables in phase than IMC with linear models. This accounts for the higher energy absorption, in most months, when the former control strategy is used. But there is another important parameter: the amplitude of the oscillations. It can be seen that, during those months in which IMC with NNs performs worst, the floater's velocity is always small (and hence the oscillations never have a large amplitude). That is why less energy can then be absorbed.

Results even better than these might be obtained switching between those two control strategies: IMC with linear models should be turned on during those months in which simulations show it will perform better. More refined switching strategies, based upon the frequency spectrum of incoming waves, could be conceived, but in this paper no attempt was made to implement them.

Table 3: Power in kW obtained under several irregular waves (figurative data)

Controller	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
None	14.3	11.4	9.1	7.7	4.5	4.5	3.5	3.4	4.5	6.7	9.9	13.6
IMC with linear model	87.5	68.1	50.6	48.3	18.1	12.5	9.6	10.2	21.0	33.5	57.4	78.3
% increase from no control	512	497	456	527	302	178	174	200	367	400	480	476
IMC with NN	115.7	92.6	63.0	55.4	6.9	4.7	3.2	3.9	8.7	42.5	76.2	103.1
% increase from no control	709	712	592	619	53	4	−9	15	93	534	670	658

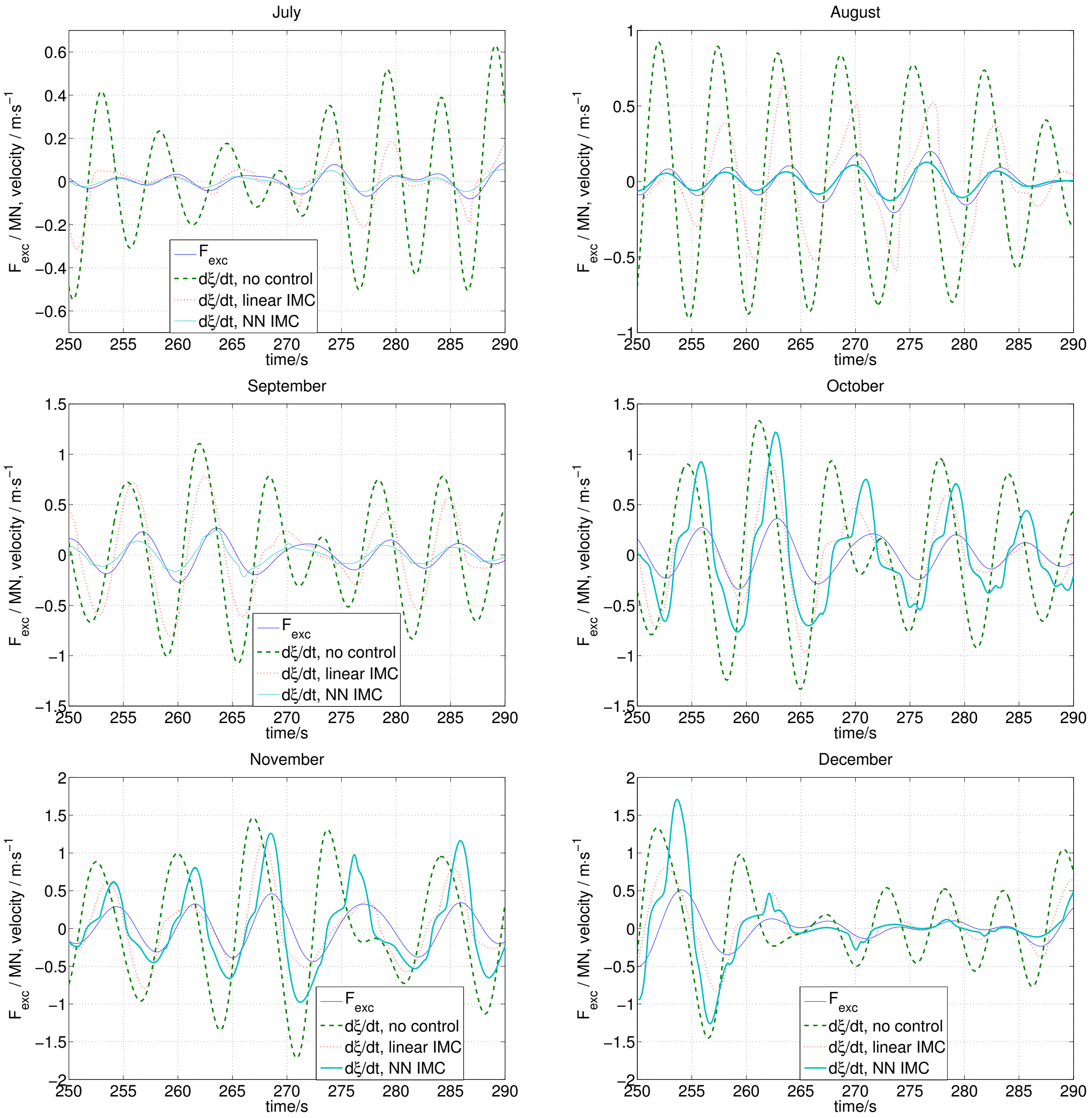

Figure 13: Fig. 12 continued

7 Conclusions

In this paper IMC was successfully applied to control the AWS in simulation. Significant absorbed wave energy increases were found, both using linear models and NNs. Results were better when IMC with NNs was employed (with a nearly sixfold increase of absorbed wave energy along the whole year, against a fivefold increase for IMC with linear models), save for the May—September period, when IMC with linear models performs better.

There is plenty of future work in this area. The identification of NN models for the AWS can still be improved. These control strategies must be compared with others, such as latching control. And some switching strategy, to commute between different controllers with different performances for waves with different characteristics, can be implemented. Finally, these control strategies should be implemented in the second-generation prototype of the AWS currently under development. These topics will be the subject of future research.

Acknowledgements

Both Pedro Beirão and Mário J. G. C. Mendes were partially supported by the "Programa do FSE-UE, PRODEP III, acção 5.3, III QCA". Duarte Valério was partially supported by grant SFRH/BPD/20636/2004 of FCT, funded by POCI 2010, POS C, FSE and MCTES; and also by the Portuguese Government and by FEDER under program "Programa de Financiamento Plurianual das Unidades de I&D da FCT" (POCTI-SFA-10-46-IDMEC) . Research for this paper was partially supported by grant PTDC/EME-CRO/70341/2006 of FCT, funded by POCI 2010, POS C, FSE and MCTES.

References

[1] A. Falcão. The history of and progress in wave energy conversion devices. In *9th World Renewable Energy Conference*, Firenze, August 2006.

[2] A. Clément, P. McCullen, A. Falcão, A. Fiorentino, F. Gardner, K. Hammarlund, G. Lemonis, T. Lewis, K. Nielsen, S. Petroncini, M. Pontes, P. Schild, B. Sjöström, H. Sørensen, and T. Thorpe. *Wave energy in Europe: current status and perspectives*, volume 6 of *Renewable and Sustainable Energy Reviews*, pages 405–431. Elsevier Science, 2002.

[3] R. Boud. Status and research and development priorities 2003: Wave and marine current energy. Technical Report DTI Report No. FES-R-132, AEAT Report No. AEAT/ENV/1054, UK Department of Trade and Industry, 2002.

[4] M. Pontes and A. Falcão. Ocean energies: Resources and utilisation. In *18th World Energy Conference*, Buenos Aires, 2001.

[5] D. Valério, P. Beirão, and J. Sá da Costa. Optimisation of wave energy extraction with the Archimedes Wave Swing. *Ocean Engineering*, 2007. Article in press.

[6] P. Pinto. Time domain simulation of the AWS. Master's thesis, Technical University of Lisbon, IST, Lisbon, 2004.

[7] J. Sá da Costa, P. Pinto, A. Sarmento, and F. Gardner. Modelling and simulation of AWS: a wave energy extractor. In *Proceedings of the 4th IMACS Symposium on Mathematical Modelling*, pages 161–170, Vienna, 2003. Agersin-Verlag.

[8] J. Sá da Costa, A. Sarmento, F. Gardner, P. Beirão, and A. Brito-Melo. Time domain model of the Archimedes Wave Swing wave energy converter. In *Proceedings of the 6th European Wave and Tidal Energy Conference*, pages 91–97, Glasgow, 2005.

[9] M. T. Pontes, R. Aguiar, and H. Oliveira Pires. A nearshore wave energy atlas for Portugal. *Journal of Offshore Mechanics and Arctic Engineering*, 127:249–255, August 2005.

[10] P. Beirão, D. Valério, and J. Sá da Costa. Linear model identification of the Archimedes Wave Swing. In *IEEE International Conference on Power Engineering, Energy and Electrical Drives*, Setúbal, 2007.

[11] T. Hägglund and K. Åström. Automatic tuning of PID controllers. In W. S. Levine, editor, *The control handbook*, pages 817–826. CRC Press, Boca Raton, 1996.

[12] I. Rivales and L. Personnaz. Nonlinear Internal Model Control using Neural Networks: application to processes with delay and design issues. *IEEE Transactions on Neural Networks*, 11(1):80–90, 2000.

[13] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-fuzzy and soft computing*. Prentice-Hall, Upper Saddle River, 1997. (chapters 8 to 11).

[14] M. Nørgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen. *Neural networks for modelling and control of dynamic systems: a practitioner's handbook*. Springer-Verlag, London, 2003.

[15] H. Demuth, M. Beale, and M. Hagan. *Neural Network Toolbox For Use with MATLAB®*. The MathWorks, Natick, 2006.

[16] D. Mandic and J. Chambers. *Recurrent Neural Networks for Prediction - learning algorithms, architectures and stability*. John Wiley & Sons, LTD, Chichester, England, 2001.

[17] J. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[18] S. Haykin. *Neural Networks—A Comprehensive Foundation*. Prentice Hall International, Inc, New Jersey, U.S.A., 2nd edition, 1999.

[19] D. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[20] J. Falnes. *Ocean waves and oscillating systems*. Cambridge University Press, Cambridge, 2002.

[21] J. Sá da Costa, P. Beirão, and D. Valério. Internal Model Control applied to the Archimedes Wave Swing. In *International Conference on Control Systems and Computer Science*, Bucharest, 2007.

[22] H. Polinder, M. Damen, and F. Gardner. Linear PM generator system for wave energy conversion in the AWS. *IEEE Transactions on Energy Conversion*, 19(3):583–589, September 2004.