SIMULATION OF A NON-AXISYMMETRIC UNDERSEA VEHICLE USING A RECURSIVE NEURAL NETWORK

David E. Hess*, William E. Faller**, Jonghyuk Lee* and James W. Broncheau†
*Naval Surface Warfare Center, **Applied Simulation Technologies,
†Northrop Grumman Shipbuilding

Keywords: *real-time simulation, neural network, non-body-of-revolution vehicle*

Abstract

Turning maneuvers of a non-axisymmetric undersea vehicle have been conducted by Northrop Grumman Shipbuilding, and experimental data has been acquired from their free-running Nnemo1 model. This paper will focus on the application of a nonlinear time domain technique, based on a fast recursive neural network (RNN) approach, to simulate the six degree-of-freedom (6-dof) motion of the vehicle. Graphs are presented comparing the predicted motions with the experimental measurements, and error measures are used to quantify the results. The predictions clearly capture the details of the maneuvers and demonstrate a faster-than-real-time simulation capability. The simulation will be coupled with a controller to allow advanced, predictive control strategies to be explored.

1 Introduction

There has been recent interest in the development of an undersea vehicle configuration exploiting an advanced hull that is not based on a body of revolution, but on a non-axisymmetric design. Newport News Shipbuilding constructed a non-axisymmetric, free-running model, as shown in Fig. 1 and denoted Newport News Experimental Model 1 (Nnemo), and conducted turning maneuvers with the vehicle to generate a data base for future concept vehicle designs [1]. These open water trials have shown that the vessel can develop large roll angles under certain operating conditions. There is a need to understand the causes of the roll, to model it via a simulation and to then develop effective control strategies.

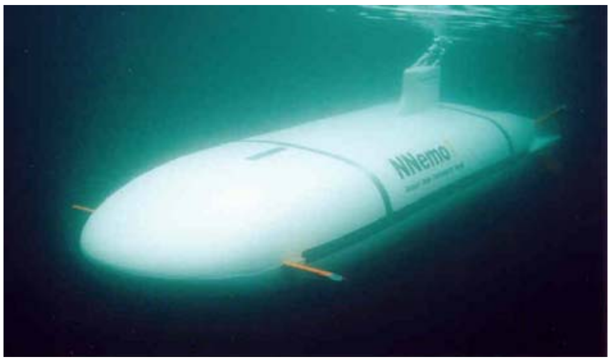


Fig. 1a. Forward view of Nnemo.

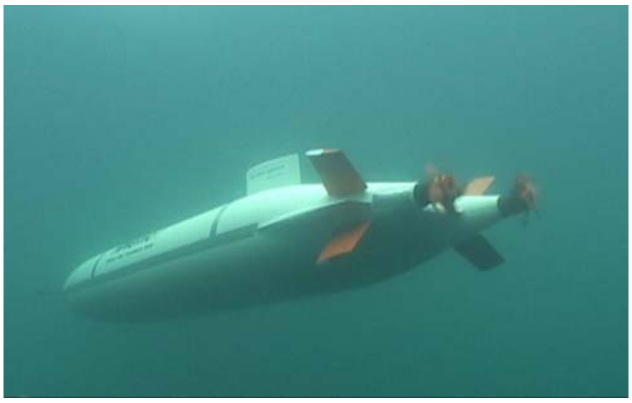


Fig. 1b. Aft view of Nnemo.

The Maneuvering and Control Division (MCD) at the Naval Surface Warfare Center, Carderock Division along with Applied Simulation Technologies have developed a faster-than-real-time simulation capability for predicting the six degree of freedom motions of marine vehicles. The simulation employs a recursive neural network as a computational technique for developing time-dependent nonlinear equation systems that relate input control variables to output state variables. The RNN is used to predict the time histories of maneuvering variables of NNemo conducting turning maneuvers. These maneuvers have been used to train and validate the neural network.

Upon completion of training, data from additional validation maneuvers (not included in the set of training maneuvers) are input into the simulation, and predictions of the motion of the vehicle are obtained and compared with measurements. The simulations are implemented in FORTRAN using code developed by the authors.

This work builds upon previous successful efforts to simulate underwater vehicles [2-3]. In contrast to these previous efforts, the current vehicle employs an aft sail configuration, four adjustable sternplanes in an X configuration, twin propellers, adjustable bowplanes and a non-axisymmetric hull design.

This simulation approach has also been directed towards surface ships operating in calm water, regular waves and random seas. Specifically, predictions for calm water operation and maneuvers in extreme regular waves were developed for a 46th scale model of a pre-contract DDG51 Arleigh Burke class destroyer [4-5]. The method has been used to predict the motion of a full scale vessel operating in sea states 4 & 5; namely, the Office of Naval Research (ONR) high speed catamaran experimental vessel, Sea Fighter [6]. Work is also in progress to develop a 6-dof maneuvering simulation for the Landing Craft, Air Cushion (LCAC) amphibious landing craft operating in calm water and in waves.

The goals of the current work are: to develop a nonlinear simulation of a non-body-of-revolution (NBOR) undersea vehicle; to require that the simulation correctly capture the details of the roll motion; to successfully demonstrate new computational capabilities for NBOR vehicles; and to couple a controller to the simulation in order to compensate for undesired vehicle motions. We turn now to a description of the vehicle, and the data used to train the simulation.

2 Description of Measured Data

The Advanced Concepts Group at Northrop Grumman Shipbuilding, Newport News Operations funded the construction and operation of the free-running Nnemo undersea vehicle. The vehicle has a length of 14.5 ft and a width of 3 ft. with a non-axisymmetric hull and appendage arrangements as shown in Fig. 1. The motion of the submerged vehicle is controlled by: two five-bladed propellers, four adjustable aft planes in an X-configuration, and two adjustable bow planes. Sensors recorded the rotational speed of the propellers and the deflection angles of the planes during the maneuvers. The vehicle is ballasted to be neutrally buoyant with no static pitch or roll angles. The sail can be mounted in a forward or aft position on the vehicle; the aft position was used for all of the maneuvers in this study. Other quantities measured during maneuvers included: time, speed over bottom, course over bottom (heading), depth, roll pitch & yaw, and roll, pitch & yaw rates.

Three types of turning maneuvers were conducted. The first type is a *planes only turn* initiated by using the aft planes. They are deflected in a manner to give an equivalent rudder deflection of 10°, 20° or 30°. Both port and starboard turns were performed at two speeds: 1.8 kn and 3 kn. Once the planes have been deflected, they are maintained at that position throughout the maneuver until it is terminated. Bowplanes are used by the controller to maintain depth. The second type of turn is denoted a *differential thrust turn*. This maneuver is carried out by rotating the propulsors at different forward speeds, or by rotating one at a forward speed and the other at a reverse speed. The third type of turn is referred to as a *combined turn*, because it uses differential thrust as well as planes to execute the turn.

The measured time series data described above were collected at 100 Hz and packaged in ASCII data files for each maneuver. Prior to using the data for developing the simulation, a series of data preparation steps were performed. These included: decimating the data to 10 Hz, repairing dropouts in some of the channels, correcting 2π transitions in the heading variable, and transforming the data to the coordinate system shown in Fig. 2. Then, needed variables were derived from the measured quantities. These included trajectory components x and y; linear velocity components

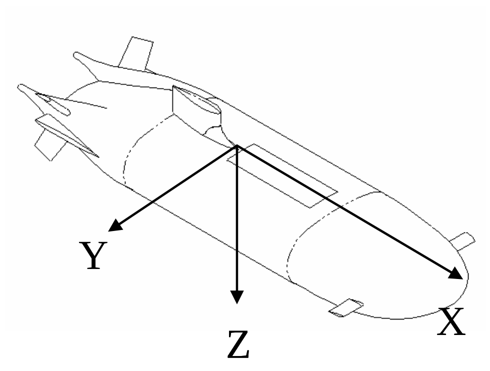
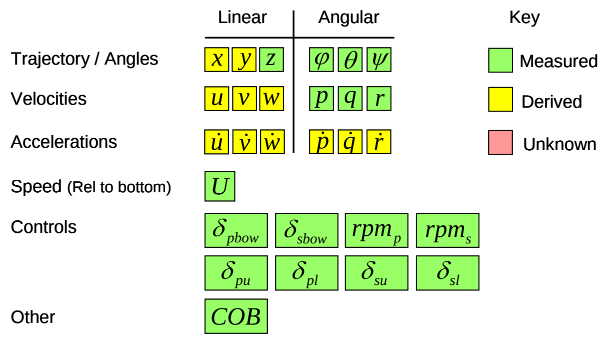


Fig. 2. Coordinate system.

u, *v* and *w*; angular velocity components *p*, *q* and *r*; and the six accelerations $\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}$ *and* $\dot{r}$. Note that the derived angular velocity components were checked against their measured counterparts; the derivation was performed to ensure that velocities and accelerations were mathematically consistent with the measured trajectory and attitude variables. All of the data were then digitally lowpass filtered at 0.5 Hz. A summary of the measured and derived variables is given in Fig. 3.



	Linear	Angular	Key
Trajectory / Angles	🟨 x 🟨 y 🟨 z	🟩 φ 🟩 θ 🟩 ψ	🟩 Measured
Velocities	🟨 u 🟨 v 🟨 w	🟨 p 🟨 q 🟨 r	🟨 Derived
Accelerations	🟨 $\dot{u}$ 🟨 $\dot{v}$ 🟨 $\dot{w}$	🟨 $\dot{p}$ 🟨 $\dot{q}$ 🟨 $\dot{r}$	🟥 Unknown
Speed (Rel to bottom)	🟩 U		
Controls	🟩 δ_{pbow} 🟩 δ_{sbow} 🟩 rpm_p 🟩 rpm_s 🟩 δ_{pu} 🟩 δ_{pl} 🟩 δ_{su} 🟩 δ_{sl}		
Other	🟩 COB		

Fig. 3. Measured and derived data.

Forces components directed along the three coordinate axes depicted in Fig. 2 are denoted by *X*, *Y* and *Z*, although we will use convenient designations such as lift and thrust as well. Moments about these axes are: roll moment, *K*, pitch moment, *M*, and yaw moment, *N*.

3 Description of Simulation

A schematic diagram of the simulation is given in Fig. 4. The figure shows a recursive neural network at the heart of the simulation. A recursive network is one that employs feedback; namely, the information stream issuing from the outputs is redirected to form additional inputs to

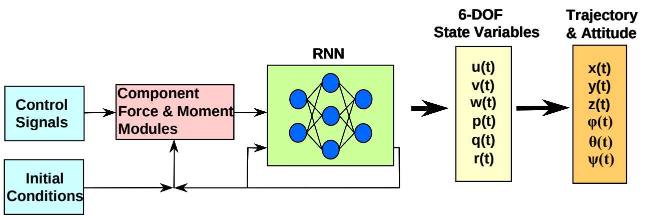


Fig. 4. Simulation schematic

the network. The input data consist of the initial conditions of the vehicle (first time step of each variable) and time histories of the control variables: propeller rotation speeds and appendage deflection angles. As the simulation proceeds, these inputs are combined with past predicted values of the state variables (outputs) to estimate the forces and moments that are acting on the vehicle. The resulting outputs are predictions of the time histories of the state variables: linear and angular velocity components which can then be used to recover the remaining hydrodynamic variables required to describe the motion of the vehicle. The equations used to describe the forces and moments are designed to *pose* the problem well to the simulation, and the network is trained to learn how the forces and moments lead to vehicle motion.

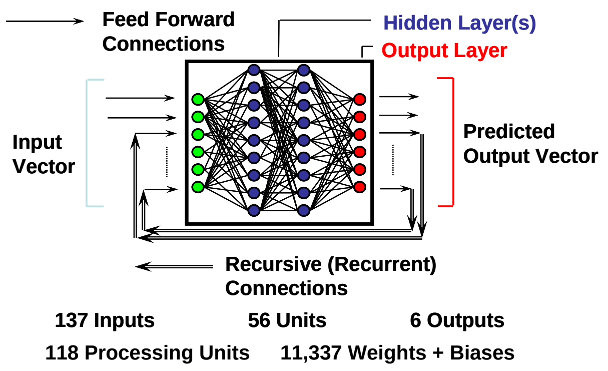


Fig. 5. RNN architecture.

The architecture of the neural network is illustrated in Fig. 5. The network consists of four layers (groupings of nodes): an input layer, two hidden (internal) layers and an output layer. Within each layer are nodes, which contain a nonlinear transfer function that operates on the inputs to the node and produces a smoothly varying output. The binary sigmoid function was used for this work; for input x ranging from $-\infty$ to ∞, it produces the output y that varies from 0 to 1 and is defined by

$$y(x) = \frac{1}{1+e^{-x}} \quad . \qquad (1)$$

The nodes in the input layer simply serve as a means to couple the inputs to the network; no computations are performed within these nodes. The nodes in each layer are fully connected to those in the next layer by weighted links. As data travels along a link to a node in the next layer it is multiplied by the weight associated with that link. The weighted data on all links terminating at a given node is then summed and forms the input to the transfer function within that node. The output of the transfer function then travels along multiple links to all the nodes in the next layer, and so on. So, as shown in Fig. 5, an input vector at a given time step travels from left to right through the network where it is operated on many times before it finally produces an output vector on the output side of the network. Most nodes have a bias; this is implemented in the form of an extra weighted link to the node. The input to the bias link is the constant 1, which is multiplied by the weight associated with the link and then summed along with the other inputs to the node. This process is illustrated in Fig. 6.

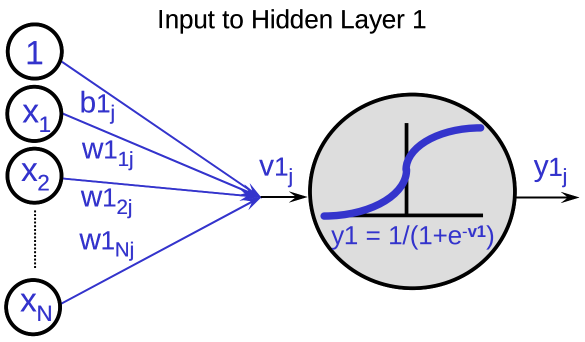


Fig. 6. RNN architecture.

The equations describing the transformation of the input vector into the output vector are given in Fig. 7. The notation used considers $i = 1,\ldots,N$ input nodes, $j = 1,\ldots,N1$ and $k = 1,\ldots,N2$ nodes in the two hidden layers and $l = 1,\ldots,No$ output nodes. The notation is similar to that found in [7].

A recursive neural network has feedback; the output vector is used as additional inputs to the network at the next time step. For the first time step, when no outputs are available, these inputs are filled with initial conditions. The

Input to Hidden Layer 1
$$\left\{\begin{aligned} v1_j &= b1_j + \sum_{i=1}^{N} w1_{ij}\; x_i \qquad (j = 1,\ldots,N1) \\ y1_j &= \frac{1}{1+e^{-v1_j}} \end{aligned}\right.$$

Hidden 1 to Hidden 2
$$\left\{\begin{aligned} v2_k &= b2_k + \sum_{j=1}^{N1} w2_{jk}\; y1_j \qquad (k = 1,\ldots,N2) \\ y2_k &= \frac{1}{1+e^{-v2_k}} \end{aligned}\right.$$

Hidden 2 to Output
$$\left\{\begin{aligned} v_l &= b_l + \sum_{k=1}^{N2} w_{kl}\; y2_k \qquad (l = 1,\ldots,No) \\ y_l &= \frac{1}{1+e^{-v_l}} \end{aligned}\right.$$

$$y_l = y_l\left(x_1, x_2, \ldots x_N\right)$$

Fig. 7. Equations relating outputs to inputs.

time step at each iteration represents a step in dimensionless time, $\Delta t'$. Time is rendered dimensionless using the vehicle's length and its speed computed from the preceding iteration; thus, the dimensionless time step represents a fraction of the time required for the flow to travel the length of the hull. The neural network is stepped at a constant rate in dimensionless time through each maneuver. An input vector at the dimensionless time, t', produces the output vector at $t' + \Delta t'$, where

$$t' + \Delta t' = t' + \frac{\Delta t\, U(t')}{L} \quad \text{and} \quad \Delta t' = 0.05 \quad .(2)$$

The network has 137 inputs. Each hidden layer contains 56 nodes, and each of these nodes uses a bias. The output layer consists of 6 nodes, and does not use bias units. The network contains 118 computational nodes and a total of 11,337 weights and biases.

The network predicts, at each time step, dimensionless forms of the six state variables: three linear velocity components, *u*, *v*, and *w*, and three angular velocity components, *p*, *q* and *r*. Specifically, the outputs are defined as

$$\begin{aligned} u'(t'+\Delta t') &= \frac{u(t'+\Delta t')}{U(t')}, \; v' \text{ and } w' \text{ similar} \\ p'(t'+\Delta t') &= \frac{p(t'+\Delta t')\, L}{U(t')}, \; q' \text{ and } r' \text{ similar} \end{aligned} \quad .$$

$$(3)$$

These velocity predictions are then used to compute at each time step the remaining kinematic variables described in Fig. 3, trajectory components, Euler angles and accelerations.

The 137 contributions that form the input vector are described as follows. Fifteen basic force and moment terms describe the influence of the control inputs and of time-dependent flow field effects; the first two are the thrust from the two propellers, T_{port} and T_{stbd}. When these two thrust terms are unequal, the resulting differential thrust contributes to a yaw moment on the vehicle, $N_{\Delta T}$. The deflections of the X-planes are converted into equivalent rudder and sternplane deflections, then the lift on these equivalent planes is computed: upper rudder, L_{rud-u}; lower rudder, L_{rud-l}; port sternplane, L_{stpl-p} and starboard sternplane, L_{stpl-s}. Lift from port and starboard bowplanes are the inputs, L_{bow-p} and L_{bow-s}. Unequal deflections of corresponding pairs of the aft X-planes can contribute to roll moments that act on the vehicle; these are summarized in two contributions, K_{rud} and K_{stpl}. Two restoring moments resulting from disturbances in roll and pitch, K_r and M_r, and two Munk moments acting on the hull, M_{Munk} and N_{Munk} make up the balance of the input vector. These terms are developed from knowledge of the controls (propeller rotation speed, equivalent sternplane and rudder deflection angles), geometry of the vehicle, and from output variables which are recursed and made available to the inputs.

Additional inputs are obtained by retaining past values of the fifteen basic inputs. This gives the network memory of the force and moment history acting on the vehicle and permits the network to learn of any delay that can occur between the application of the force or moment and the response of the vehicle. The number of past inputs that is used is displayed in the summary given in Table 1. The number of past values to keep is chosen empirically and appears to be a function of the frequency response of the vehicle. In this case the network is given information about past events for a period of time required for the flow about the vehicle to travel a distance of 0.5L.

Table 1 Summary of network inputs.

Input Description	#
$T_{port}(t'), T_{port}(t'-\Delta t'), \ldots, T_{port}(t'-4\Delta t')$	5
$T_{stbd}(t'), T_{stbd}(t'-\Delta t'), \ldots, T_{stbd}(t'-4\Delta t')$	5
$N_{\Delta T}(t'), N_{\Delta T}(t'-\Delta t'), \ldots, N_{\Delta T}(t'-4\Delta t')$	5
$L_{rud-u}(t'), L_{rud-u}(t'-\Delta t'), \ldots, L_{rud-u}(t'-9\Delta t')$	10
$L_{rud-l}(t'), L_{rud-l}(t'-\Delta t'), \ldots, L_{rud-l}(t'-9\Delta t')$	10
$L_{stpl-p}(t'), L_{stpl-p}(t'-\Delta t'), \ldots, L_{stpl-p}(t'-9\Delta t')$	10
$L_{stpl-s}(t'), L_{stpl-s}(t'-\Delta t'), \ldots, L_{stpl-s}(t'-9\Delta t')$	10
$L_{bow-p}(t'), L_{bow-p}(t'-\Delta t'), \ldots, L_{bow-p}(t'-9\Delta t')$	10
$L_{bow-s}(t'), L_{bow-s}(t'-\Delta t'), \ldots, L_{bow-s}(t'-9\Delta t')$	10
$K_{rud}(t'), K_{rud}(t'-\Delta t'), \ldots, K_{rud}(t'-9\Delta t')$	10
$K_{stpl}(t'), K_{stpl}(t'-\Delta t'), \ldots, K_{stpl}(t'-9\Delta t')$	10
$K_r(t'), K_r(t'-\Delta t'), \ldots, K_r(t'-4\Delta t')$	5
$M_r(t'), M_r(t'-\Delta t'), \ldots, M_r(t'-4\Delta t')$	5
$M_{Munk}(t'), M_{Munk}(t'-\Delta t'), \ldots, M_{Munk}(t'-9\Delta t')$	10
$N_{Munk}(t'), N_{Munk}(t'-\Delta t'), \ldots, N_{Munk}(t'-9\Delta t')$	10
$u'(t'), v'(t'), w'(t'), p'(t'), q'(t'), r'(t')$	6
$u'(t'-\Delta t'), v'(t'-\Delta t'), w'(t'-\Delta t'),$ $p(t'-\Delta t'), q'(t'-\Delta t'), r'(t'-\Delta t')$	6
Total	137

Recursed outputs from the prior time step are used as six additional contributions to the input vector. Furthermore, the output vector from one previous time step is also retained and made available as six additional inputs. Knowledge of the output velocities for two successive time steps permits the network to implicitly learn about the accelerations of the vehicle. We turn now to a discussion of the results.

4 Results

As mentioned previously, information presented to the inputs of the simulation is modified as it flows through the network by the presence of the weights and by the nonlinear outputs of each of the nodes until it arrives at the output layer of the network. Thus, at each time step, an input vector produces a predicted output vector; this is then compared to the actual (target) output vector determined from the data. The difference between the target and predicted output vectors is a measure of the error of the prediction. The process by which the network is iteratively presented with an input vector in order to produce outputs that are then compared with a desired output vector is known as training. The purpose of training is to gradually modify the weights between the nodes in order to reduce the error on subsequent iterations. In other words, the neural network *learns* how to reproduce the correct answers. When the error has been minimized, training is halted, and the resultant collection of weights that have been established among the many connections in the network represent the knowledge stored in the trained neural net. Therefore, a training algorithm is required to determine the errors between the predicted outputs and the desired target values and to act on this information to modify the weights until the error is reduced to a minimum. The training algorithm employed here is called backpropagation, which is a gradient descent algorithm. The collection of input and corresponding target output vectors comprise a training set, and these data are required to prepare the network for further use. Data files containing time histories of all of the variables described in Fig. 3 for the three turning maneuvers formed the training sets.

After the neural network has been successfully trained, the weights are no longer modified and remain fixed. At this point the network may be presented with an input vector similar to the input vectors in the training set (that is, drawn from the same parameter space), and it will then produce a predicted output vector. This ability to generalize, that is, to produce reasonable outputs for inputs not encountered in training is what allows neural networks to be used as simulation tools. To test the ability of the network to generalize, a subset of the available data files must be set aside and not used for training. These validation data files then demonstrate the predictive capabilities of the network.

The simulation was trained to predict the time histories of maneuvering variables of an undersea vehicle executing submerged turning maneuvers of the three types described previously. A total of 24 maneuvers were used. Of these, 17 were used for training and 7 *validation runs* were set aside to test how well the simulation was able to predict maneuvers similar to, but different from, maneuvers in the training set. The network was trained for 100,000 epochs, where an epoch is defined as one presentation of the time series for all inputs and outputs for all files in the training set.

Periodically, training is paused and both sets of training and validation files are presented to the network, and the outputs are compared to the measured data. Two error measures are used to quantify the comparison: an Average Angle Measure, *AAM*, and a correlation coefficient, *R*. The *AAM* was developed by the Maneuvering Certification Action Team at NSWCCD in 1993-1994 and some details may be found in the papers [8-9]. Both *AAM* and *R* quantify (with a single number) the accuracy of a predicted time series when compared with the actual measured time series. A value of 1 corresponds to perfect magnitude and phase correlation, -1 implies perfect magnitude correlation but 180° out of phase and zero indicates no magnitude or phase correlation.

The epoch, at which a maximum in the average value of the *AAM* and *R* (computed over all 24 files) was found, was noted and designated the stopping epoch. The simulation was then restarted and trained to the stopping epoch of 29000. The resulting values are given in Table 2.

Table 2 Quantitative error measures.

	AAM	R	AVG
Trn	0.893	0.961	0.927
Val	0.860	0.932	0.896
All	**0.876**	**0.946**	**0.911**

The columns of the table give the *AAM*, *R* and the average (of *AAM* and *R*) values, and the rows give the file sets over which the results were determined. So, row 1 gives the values of *AAM* and *R* computed over the 17 training files, row 2 gives the answers for the 7 validation files and the last row computes the numbers over all 24 files in both sets. The first indication of success is the high values achieved for the training files. This is a sign that the problem is *posed* well, and the input vector contains sufficient information for the simulation to capture the details of the maneuver. The excellent numbers for the validation files demonstrate that the network has *learned* how the vehicle maneuvers during these three types of turns, as these files were not seen during the training process. Another indication of the quality of the simulation is the closeness in the relative magnitudes of the error measures for the validation files and the training files; the simulation is producing stable results of uniform quality.

The averaged error measures given in Table 2 are computed for four critical variables: speed, depth, roll and pitch. These variables are sensitive measures of the performance of an undersea vehicle. The speed magnitude U is computed from velocity components as

$$U = \sqrt{u^2 + v^2 + w^2} \quad . \qquad (4)$$

Besides speed, the other critical variables are obtained by integrating the outputs of the simulation. Small errors in the outputs could easily grow during the integration to produce large errors in these variables. The point is that choosing these critical variables is appropriate for this type of vehicle, and represents a more stringent test than just looking at simulation outputs alone. The results in Table 2 have been decomposed into contributions from the four critical variables computed over all 24 files, and are presented in Table 3.

Table 3 Critical variable performance.

	Speed	Depth	Roll	Pitch
AAM	0.991	0.957	0.808	0.761
R	0.998	0.957	0.955	0.884

Speed and depth are extremely well predicted; whereas, the simulation is having relatively more difficulty with roll and pitch. The graphs which follow bear these relative indications out; however, as will be seen, the graphical comparison between the predictions and results is very accurate.

Figure 8 depicts measured and predicted time series for 12 variables for training run 301, which is a 10° planes only port turn at 3 kn. The right hand column (Fig. 8b) shows the six velocity outputs of the simulation with the predictions in red and the measured data in black. The typical behavior: speed loss in the turn, lateral velocity away from the center, rapidly changing yaw rate that settles down to a near constant value in the turn, and fluctuating vertical roll and pitch rates are all captured very accurately. The left hand column (Fig. 8a) gives speed, depth, roll, pitch, heading and trajectory (x vs. y). The drop in speed is predicted nearly perfectly. Large oscillations in roll, to ±30°, are evident, and the simulation is following the measurements. Some small discrepancies are apparent in pitch and depth, on the order of 2° in pitch and perhaps 2 ft in depth. The overall shape and character of the trajectory is picked up nicely. Apparently a small error, perhaps in yaw rate, early in the maneuver, biased the prediction slightly and led to the shift in the curves. This makes clear just how *precise* the velocity predictions must be in order for accurate integrated variables to be achieved. The conclusion to draw from this training run is that the simulation has successfully *understood* the 6-dof motion of the vehicle and can replicate it.

Figure 9 shows another typical training run; this one is a combined turn using differential thrust and an equivalent 10° deflection to port at a speed of 1.8 kn. For this and later figures, only the integrated variables and speed are shown. Roll excursions of ±12° are apparent, and the simulation picks this up nicely with errors of 2-3° or less. Errors in pitch and depth are at most 2° and 1 ft, respectively. The tactical diameter of the turn is correct and the overall shape of the trajectory is predicted nicely. We now turn to some examples of validation maneuvers, which are the true test of the simulation.

Figures 10 & 11 show two typical validation turning maneuvers: a 20°, 1.8 kn planes only turn to starboard and a 30°, 1.8 kn planes only turn to port. Roll variations of ±20° can be found, and the simulation matches

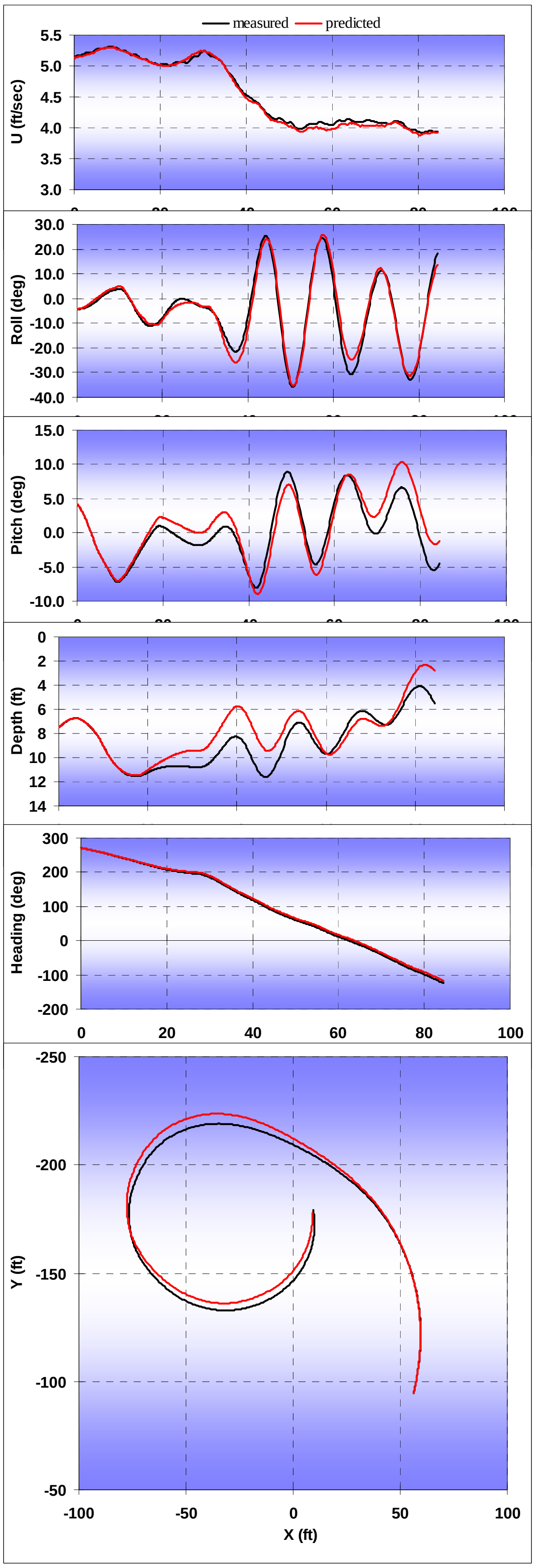


Fig. 8a. Trn. 301: 10° Port Turn, 3 kn.

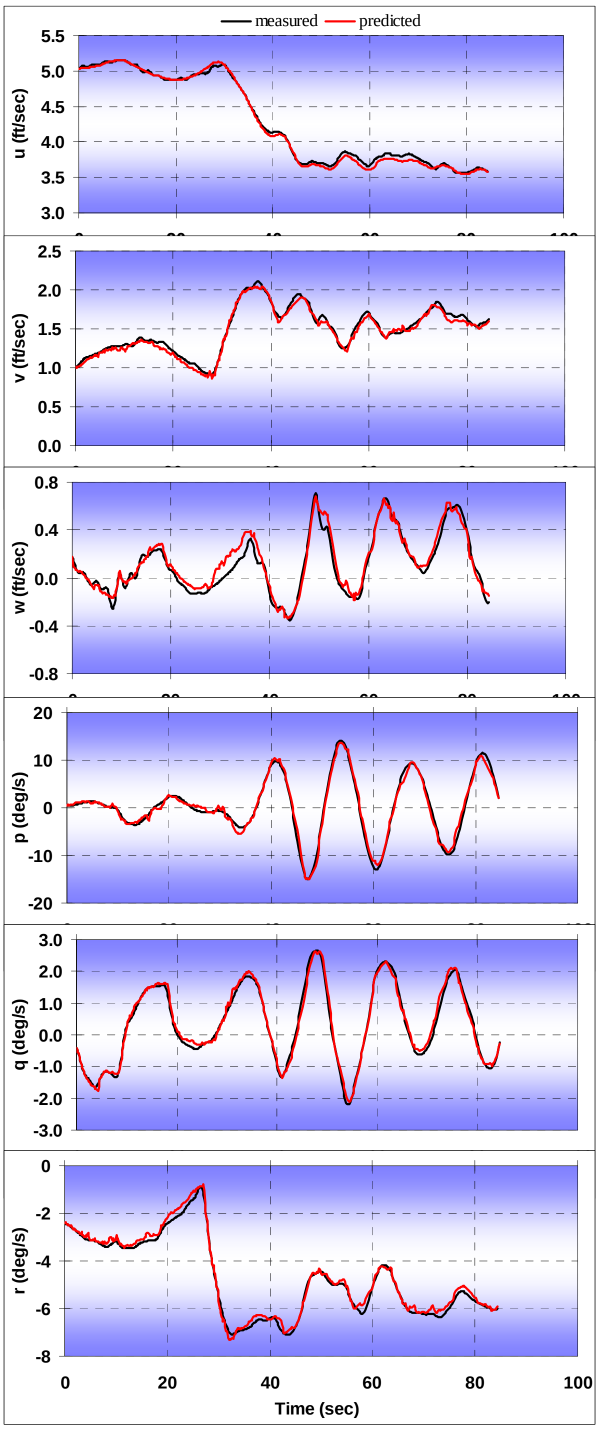


Fig. 8b. Trn. 301: 10° Port Turn, 3 kn, Vels.

.

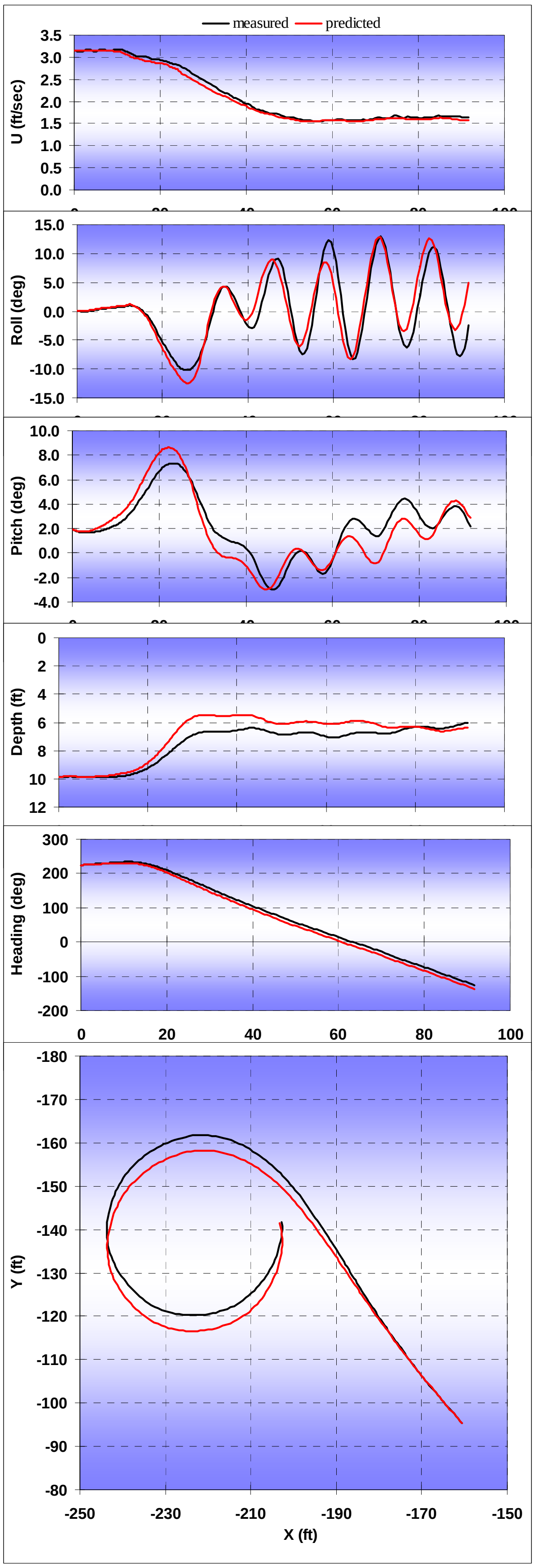


Fig. 9. Trn. 328: 10° Port. Comb. Turn, 1.8 kn.

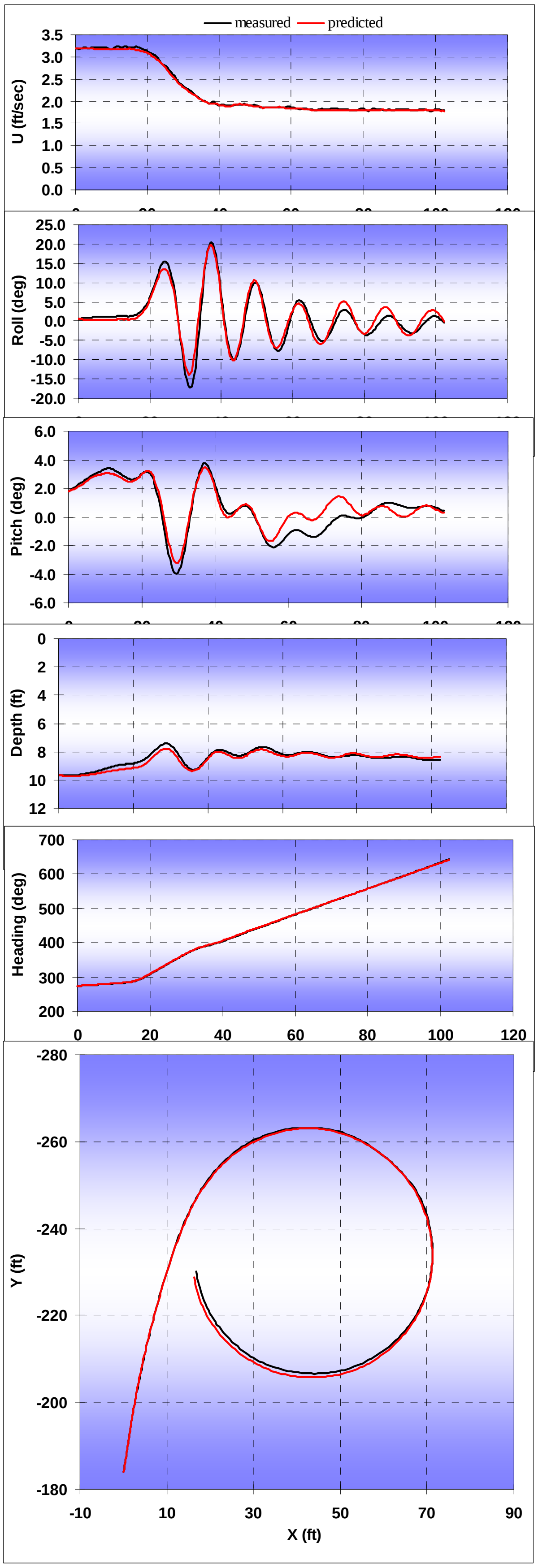


Fig. 10. Val. 314: 20° Stbd. Turn, 1.8 kn.

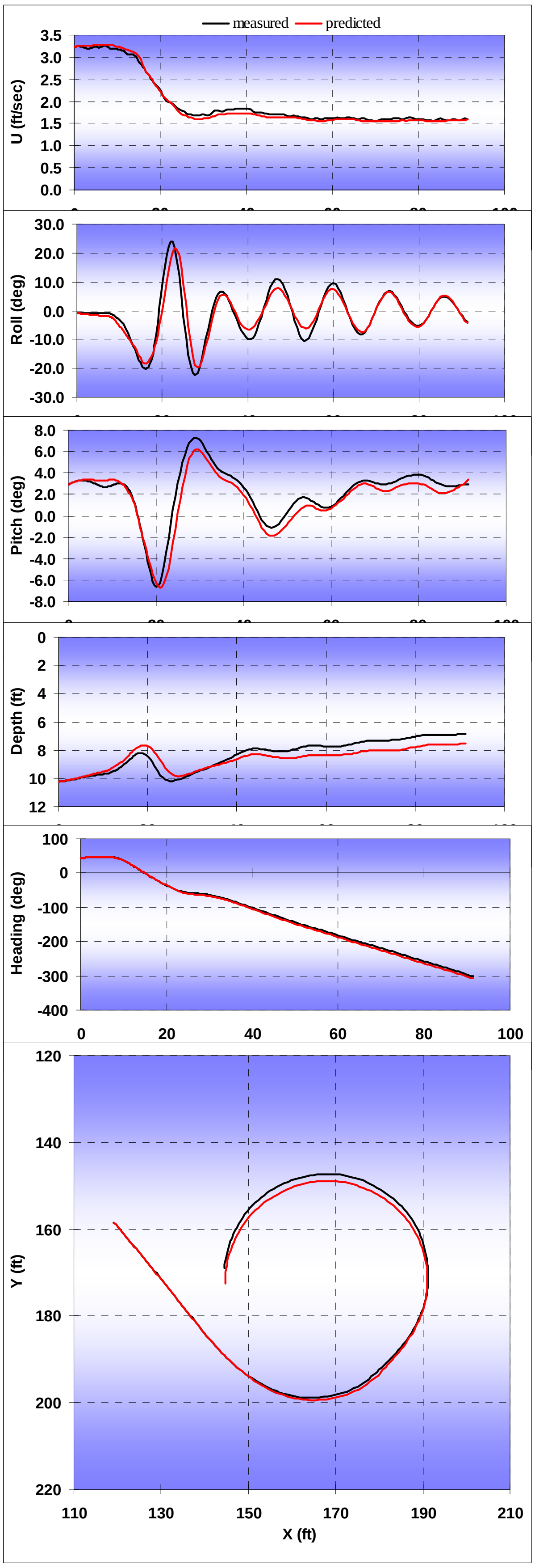


Fig. 11. Val. 319: 30° Port. Turn, 1.8 kn.

this behavior within a couple of degrees. For both maneuvers, depth and trajectory are captured extremely accurately. Indeed, the worst apparent predictions are for pitch, but the expanded scale on these plots show the error to be at most a degree or two.

5 Conclusions

When considering the nonlinear simulation results, they demonstrate the clear potential for the development of models of undersea vehicle behavior. These models can be used not only for time-domain simulation of vehicle motions but also for predictive control applications. The results show that the open loop simulation can predict into the future without *a priori* guidance from real-time measured sensor feedback, and therefore without real-time feedback from an on-board control system driving the simulation to a known solution. The information predicted about the future ship motion history can be used to determine ordered control inputs to the ship that are needed to either realize a desired ship motion history or to counter an undesired ship motion history in the future.

Such work is currently in progress. An existing X-plane controller for an undersea vehicle is being coupled to the faster-than-real-time simulation. The goal is to demonstrate that the roll excursions can be reduced for all three types of turning motions.

Not least among the outcomes discussed here is the fact that a simulation capability has been demonstrated for undersea vehicles that have a non-axisymmetric shape. A new generation of simulation capabilities permits greater flexibility in the design of such vehicles for future missions.

Acknowledgments

The U.S. Office of Naval Research sponsored this work, and the program monitor is Dr. Ronald D. Joslin, Code 331. The authors would like to thank Mr. Todd Sedler of Northrop Grumman Shipbuilding, Newport News Operations, Advanced Concepts Group, Dept. E33, Submarine Technology for providing the

Nnemo captive model experimental data and for his many supportive conversations.

References

[1] Sedler T. Nnemo 1 and 2 Public Information Release Index. Prepared for OFOISR Release 07-S-2037, Northrop Grumman Shipbuilding, Newport News Operations, Advanced Concepts Group, Dept. E33, Submarine Technology, Aug 23, 2007.

[2] Faller W, Smith, W and Huang T. Applied dynamic system modeling: six degree-of-freedom simulation of forced unsteady maneuvers using recursive neural networks. 35th AIAA Aerospace Sciences Meeting, 97-0336, 1997, pp. 1-46.

[3] Hess D and Faller W. Using recursive neural networks for blind predictions of submarine maneuvers. 24th Symposium on Naval Hydrodynamics, Fukuoka, Japan, July 8-13, 2002.

[4] Hess D, Faller W, Lee J, Fu T and Ammeen E. Ship maneuvering simulation in wind and waves: A nonlinear time-domain approach using recursive neural networks. 26th Symposium on Naval Hydrodynamics, Rome, Italy, September 17-22, 2006.

[5] Faller W, Hess D, Fu T and Ammeen E. Fast time-domain nonlinear simulations of ship motions in extreme waves using a new formulation for the wave elevation. To be presented at the 27th Symposium on Naval Hydrodynamics, Seoul, Korea, October 5-10, 2008.

[6] Hess D, Faller W, Minnick L and Fu T. Maneuvering simulation of Sea Fighter using a fast nonlinear time domain technique. 9th International Conference on Numerical Ship Hydrodynamics, Ann Arbor, Michigan, August 5-8, 2007.

[7] Haykin S. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[8] Ammeen E. Evaluation of correlation measures. Naval Surface Warfare Center Report, CRDKNSWC-HD-0406-01, March 1994, pp. 1-65.

[9] Roddy R., Hess D. and Faller W. Utilizing neural networks to predict forces and moments on a submarine propeller. 2008-0888, 46th AIAA Aerospace Sciences Mtg, Reno, NV, Jan. 7-10, 2008.

Copyright Statement