

# Automated Hydrodynamic Shape Optimization Using Neural Networks

A. Schmitz<sup>\*</sup>, E. Besnard<sup>†</sup> & H. Hefazi<sup>‡</sup>

<sup>\*</sup> Research Associate

<sup>†</sup> Associate Professor

<sup>‡</sup> Professor and Department Chair

Mechanical and Aerospace Engineering Department, California State University Long Beach

## ABSTRACT

*This paper presents an optimization method integrating automated CAD-based Computational Fluid Dynamics (CFD) and Neural Networks for hydrodynamic shape optimization, and its applications to underwater hull configurations. Unlike in classical optimization methods where direct CFD computations are used in the optimization loop, the present method uses trained Neural Networks to evaluate the hydrodynamic performance of the configuration throughout the optimization process. Here, the CFD tool is used to generate the data sets used for training the Network. Both the classical optimization and Neural Network approaches are presented and applied to the design/optimization of one of Pacific Marine's advanced lifting bodies. Results are compared and show that the NN approach can produce better designs at substantially lower computational costs than the classical approach. For the example treated with 28 design variables where total CPU time was the limiting factor, a 34% improvement with the NN approach is obtained when the classical approach only yields a 26% improvement using five times more CPU time. Areas of future research are outlined.*

## NOMENCLATURE

CAD: Computer Aided Design  
CasCor: Cascade Correlation  
CCDoTT: Center for Commercial Deployment of Transportation Technologies  
CFD: Computational Fluid Dynamics  
CSULB: California State University, Long Beach  
DV: Design Variable  
GA: Genetic Algorithm  
HU: Hidden Unit  
LOD and L/D: Lift to Drag Ratio  
LT: Long Tons  
NN: Neural Networks  
TS: Training Set

VS: Validation Set

## INTRODUCTION

Synthesis tools used in the concept design phase have been around for many years and are used widely by industry to develop point solutions of ship designs to populate and study the trade space based on the designers' experiences. This process has the potential to be greatly enhanced by the application of optimization methods to the design problem.

Optimization will allow for a more complete analysis of the trade space in two ways. First, the automation of the synthesis tool required for optimization will allow for more ship designs within the

trade space to be examined, or a greater population of data in the trade space in a shorter period of time. Secondly, advanced procedures integrated into the optimization process, such as Neural Networks, will be able to better define the ‘hot spots’, or areas of the trade space worth examining in greater detail. The development of design/optimization and analysis tools will allow alternative hull forms to be explored with models of higher level of fidelity thus allowing for better comparisons between various hulls for a given ship capability, and faster generation of concept designs.

Since 1998, CSULB, under programs funded by the Office of Naval Research (ONR) [Besnard 1998] and the Center for Commercial Deployment of Transportation Technologies (CCDoTT) [Hefazi *et al.*, 2002, 2003(a), (b), (c), (d), (e)] has been developing advanced automated optimization methods for applications to fast ship design. The focus of these programs has been the optimization of lifting bodies, such as the Pacific Marine’s Blended Wing Body (BWB) for which the lift to drag ratio has been maximized [Hefazi *et al.*, 2002 and 2003(e)]. Technologies developed at CSULB under these programs include:

- CAD-based parametric shape definition which allow for the automation of the shape optimization process and automatic coupling with the CFD mesh generator and solver;
- Integration of Neural Networks in the optimization method and its application to the design of various lifting bodies.

This paper summarizes these developments. While the focus of this paper is only on hydrodynamic shape optimization of lifting bodies, the methodology can be extended to multi-disciplinary optimization of other ships.

## DESIGN/OPTIMIZATION METHOD

### Overview of the classical and NN-based methods

A general optimization process is illustrated in Fig. 1. An initial set of  $s$  design variables,  $\mathbf{x} = (x_i)_{1 \leq i \leq s}$ , which might represent the configuration designed by experienced engineers, is supplied to the optimizer. Then, for this design, the objective function,  $f$ , is evaluated and the constraints,  $g_i$ , are analyzed to check whether they are violated or not. If the optimum is not reached, these values are fed back to the optimizer that modifies the design vector  $\mathbf{x}$ . The process is repeated until convergence.

For the application to aerodynamic or hydrodynamic optimization, the three main components of the numerical method are, (1) the representation of a configuration by a set of design variables (D.V.),  $\mathbf{x}$ , (2) the optimization method, and (3) the evaluation of the aerodynamic performance, i.e.  $f$ , for a given configuration. The constraints,  $g_i$ , are analyzed at the stage appropriate for the problem considered.

In the classical case discussed below, the objective function calculation requires the use of a CFD code that solves the equations of fluid dynamics and evaluates the aero- or hydrodynamic performance needed to perform the optimization. This calculation is usually very computer intensive – and drives the cost of the optimization process – with one flow solution taking from several minutes to several hours or days in the most complex problems, even using massive parallel computations. The integration of this CFD analysis in the optimization process requires further an automated shape and mesh generation which can also be very computer intensive.

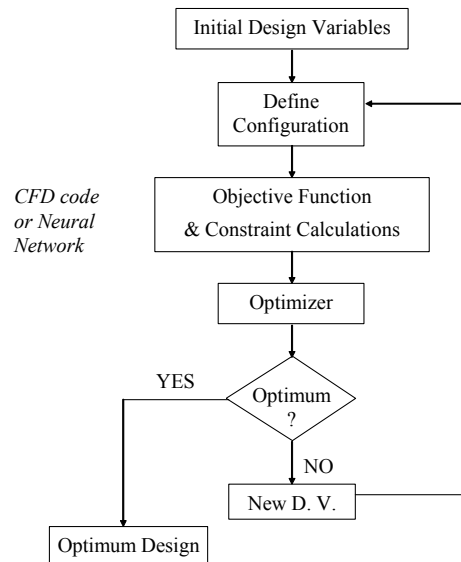


Fig. 1 Flowchart of the numerical optimization.

In the approach using neural networks (NN), the process is divided in two parts: (1) generation of a training set and training of the neural network, and (2) optimization of the configuration, like in Fig. 1, with the NN instead of the CFD code. The Network is trained with a relatively small set of data, which can be a mix of experimental data and/or data generated with the CFD code. Then, this network is coupled with a global optimization method without the drastic increase in CPU time since the NN allows for a very fast evaluation of the hydrodynamic performance. All other elements of the optimization approach, such as configuration representation and selection of the

numerical optimization algorithm are currently independent of the use of Neural Networks to predict the hydrodynamic properties.

Instead of having the cost buried in the function evaluations during the optimization, and therefore not easily known *a priori*, the approach using Neural Networks moves the cost to the generation of the training set (TS). By controlling the size of the training set, not only the cost can be known *a priori*, but also reduced. Knowing the cost of the optimization offers advantages from a programmatic stand point because it allows managers to know exactly when they will get a solution to the problem at hand and also what the cost will be. In addition, the algorithm allows for external data, such as previous suitable computations or experimental data, to be included in the training set if desired.

The more specific elements of classical and neural-network-based optimization methods are discussed in the next two subsections.

### Automated “Classical” Optimization Approach

The classical optimization approach is described in detail in [Hefazi *et al.*, 2002]. It is summarized here for comparison with the neural network-based approach. Also some elements of the two approaches are common.

Nowadays, all design engineers are accustomed to designing their vehicles using some Computer Aided Design (CAD) or solid modeling package, such as Pro-Engineer, CATIA, UniGraphics, IDEAS. In addition, the designer usually represents the configuration by a set of parameters, or design variables, which can be varied to improve the design by linking the CAD software with an appropriate analysis module. This approach is routinely used for structural design, for example, by linking the CAD software with a Finite Element (FE) method. Such approach is not yet routinely used, however, in the case of hydrodynamic shape optimization, because additional challenges face the designer. Among these issues are

- The cost and accuracies associated with flow analysis using CFD
- The grid requirements for CFD methods

In order to address the issue of CFD shape optimization, one must be able to *automatically* vary the shape of various elements of the configuration in the CAD method, generate a mesh of sufficient quality for the CFD method, and use an efficient and accurate CFD method to obtain the hydrodynamic performance of the configuration being analyzed. The driver in the selection of the components of the CFD optimization method is the ability to link these tools together in an

automated fashion, without user intervention, while ensuring that the flow analysis is both efficient and accurate for the problem at hand. For this reason, several options for each tool were considered and the following set of tools was selected:

- CAD software: *Pro-Engineer* [Parametric Technology Corporation, <http://www.ptc.com/>]
- Grid generation software: *ICEM-CFD* [ICEMCFD Engineering, <http://www.icemcfd.com/>]
- CFD software: CSULB-developed Interactive Boundary Layer (IBL) approach with free surface modeled by negative images [Besnard 1998, and Hefazi *et al.*, 2002]
- Optimization software: *iSIGHT* [Engineous Software, <http://www.engineous.com/>]

*Pro-Engineer* and *ICEM-CFD* were selected because of the existence of a module which does allow for automatic data transfer between the CAD and grid generation package. The IBL approach was chosen because at high Reynolds numbers and low angles of attack, it is a very accurate and efficient approach. In addition, because of the large Froude number for the case at hand, the free surface can be modeled with negative images. Finally, the numerical optimizer *iSIGHT* offers an easy to use platform for the optimization and/or design of experiments. The method, controlled by *iSIGHT*, integrates the different software packages with several scripts, which, once setup, performs all tasks automatically, without user intervention. This automatic setup is critical in the optimization process.

Using these elements, the classical optimization approach follows the process shown in Fig. 1 to search for an optimum.

### Neural-Network-based Optimization Approach

As discussed earlier, the use of the Neural Network (NN) approach encompasses several steps:

1. Generation of the Training Set (TS) & Validation Set (VS) (to be defined below)
2. NN training to obtain a NN “evaluator(s)”
3. Optimization as in Fig. 1 with the NN evaluator(s)

The first two steps are explained in detail in the next subsections. The third step is essentially the same as the classical optimization with the CFD code replaced by the Neural Network, and thus is not repeated here.

### GENERATION OF THE TRAINING AND VALIDATION SETS

A training set (TS) corresponds to a set of known data points (design variables and their associated dependent variables, such as objective function(s) and constraints) used to train the NN, i.e. the network attempts to achieve an output which matches the input (training set). A validation set (VS) is a set which, unlike the TS, is not used for training per say, but rather is used for stopping the training. The purpose of the VS is to avoid over-fitting which can occur with the NN topology used here and described in the next section, Cascade Correlation. These are described in detail along with the theoretical foundation and NN methodology in Ref. [Hefazi *et al.*, 2003(b)].

The generation of the VS and TS is accomplished using a similar approach to the classical optimization approach but with the “optimizer” tool within iSIGHT replaced by a Latin Hypercube sampling program. The outputs are 2 datasets which contain a series of design variables covering the design space of interest and their corresponding dependent variable. The TS and VS are then used for training the network.

#### NN TRAINING WITH CASCADE CORRELATION ALGORITHM

Artificial Neural Networks can be used for various tasks such as pattern recognition, function approximation, control systems, etc. The number of neurons, their activation function and the topology of the connections enable the NN to perform its task. Cascade-Correlation (CasCor), first introduced by Fahlman and Lebiere, [Fahlman 1990] is a supervised learning algorithm for NN. Instead of just adjusting the weights in a network of fixed topology, CasCor begins with a minimal network, then automatically trains and adds new hidden units one-by-one in a cascading manner. This architecture has several advantages over other algorithms: it learns very quickly; the network determines its own size and topology; it retains the structure it has built even if the training set changes; and it requires no back-propagation of error signals through the connections of the network. For a large number of inputs, the most widely used learning algorithm, back-propagation, is known to be very slow. CasCor does not exhibit this limitation. [Fahlman 1990]

Also, and foremost, this method was chosen because the size of the network does not need to be determined in advance. Methods which use a fixed network topology involve evaluating in advance (before training) the type of network that would best suit the application (how many neurons, how many hidden-layers...) to match the complexity of the NN to the complexity of the function. For most engineering applications like CFD shape optimization, one does not know exactly how “complicated” the function is.

There are essentially two approaches for training multi-layer feedforward networks for function approximation which can lead to variable networks. The Pruning Algorithms start with a large network, train the network weight until an acceptable solution is found, and then use a pruning method to remove unnecessary units or weights. On the other hand, Constructive Algorithms start with minimal network, and then grow additional hidden units as needed.

The advantages of Constructive Algorithms (CA’s) versus Pruning Algorithms (PA’s) are that the NN size is automatically determined. CA’s are computationally economical compared to PA’s which spend most time training on networks larger than necessary. Also, they are likely to find smaller network solutions, thus requiring less training data for good generalization. And they require a small amount of memory because they usually use a greedy approach where only part of the weights is trained at once, whereas the other part is kept constant.

The constructive CasCor algorithm begins with a minimal network consisting of the input and output layers and no hidden unit (neurons), then automatically trains and adds one hidden unit at a time until the error between the targets and the outputs from the network reaches a desired minimal value. Thus, it self determines the number of neurons needed as well as their connectivity or weights.

The CasCor algorithm has been extensively studied for the past three years at CSULB. The original algorithm of Fahlman and Lebiere was geared towards pattern recognition; it has been greatly improved to make it a robust and accurate method for function approximation. [Schmitz 2002, Hefazi *et al.*, 2003(a) & (b)]

A synopsis of the algorithm follows.

1. Start with the required input (independent variables/design variables) and output (dependent variable) units; both layers are fully connected. The number of inputs and outputs is dictated by the problem.
2. Train all connections ending at an output unit with a usual learning algorithm until the squared error  $E$  (Eq. 1) of the NN no longer decreases.  $E$  is defined by

$$E = \frac{1}{2} \sum_{p=1}^{P_{\max}} \|\mathbf{y}_p - \mathbf{t}_p\|^2 = \frac{1}{2} \sum_{p=1}^{P_{\max}} \sum_{i=1}^m [y_{i,p} - t_{i,p}]^2 \quad (1)$$

where  $m$  is the number of outputs,  $P_{\max}$  the size of the training set,  $y_{i,p}$  the  $i^{\text{th}}$  output from the NN, and  $t_{i,p}$  is the corresponding target.

3. Generate a pool of candidate units that receive trainable input connections from all of the network’s external inputs and from all pre-existing hidden units.

The output of each candidate unit is not yet connected to the active network (output). Train each unit (its weights) to maximize the correlation formula  $C$  (Eq. 2). Training takes place with an ordinary learning algorithm and is stopped when  $C$  no longer improves.

$$C = \sum_{i=1}^m \left| \sum_{p=1}^{p_{\max}} (z_{o,p} - \langle z_o \rangle) (E_{i,p} - \langle E_i \rangle) \right| \quad (2)$$

where  $z_o$  is the output of the candidate hidden unit and  $E_{i,p}$  is the residual error of the outputs calculated at Step 2,  $E_{i,p} = y_{i,p} - t_{i,p}$ . The quantities between brackets  $\langle \rangle$  are averaged quantities over the training set.

Each candidate unit has a different set of random initial weights. All receive the same input signals and see the same residual error for each training pattern. Because they do not interact with one another or affect the active network, they can be trained simultaneously. A pool of 7 candidates was used in this study.

4. Only the candidate whose correlation score is the best is installed. The other units in the pool are discarded. The best unit is then connected to the outputs and its input weights are frozen. The candidate unit acts now as an additional input unit. Train again the input-outputs connections by minimizing the squared error  $E$  as defined in Step 2. The use of several candidates greatly reduces the chances that a useless unit will be permanently installed because an individual candidate was trapped in a local maximum during training.
5. Repeat until the stopping criterion is verified. This stopping criterion makes use of a Validation Set which is smaller than the TS. [Hefazi *et al.*, 2003(a)]

Also a sigmoid  $\sigma(x) = 1/(1 + e^{-x})$  was chosen as activation function for the hidden units.

Steps 2 and 4 require the use of an optimization routine. Commercially available software, DOT, developed by Vanderplaats [Vanderplaats 1994] was chosen. This software contains a choice of the latest state-of-the-art optimization methods. TY. Kwok and DY. Yeung demonstrated in reference [Kwok 1993] that the correlation algorithm can always reach an  $E < \epsilon$  for a given  $\epsilon > 0$  for  $L^2$  functions, even when using a local optimizer. Therefore, the Broydon-Fletcher-Goldfarb-Shanno (BGFS) method from the DOT software was chosen for its proven efficiency. [Vanderplaats 1994]

The training program is a C++ software in which the Cascade Correlation algorithm is implemented. The outcome of the training is a NN in the form of an

executable in which the proper number of hidden units and corresponding weights – found during training – have been implemented. This approach of generating such executable for each network was selected for efficiency during the optimization process. The other approach would consist of printing to a file a list of weights and hidden unit information. This information would have to be read at each function call during the optimization process, thus requiring substantial amount of computer input/output (I/O) time when compared with the short CPU time needed for the actual computation. The error information w.r.t. TS and VS is also given to give an estimate of magnitude of the accuracy to be expected during optimization. Hefazi *et al.*, 2003(b) describes the Neural Network and its characteristics, as well as how the software is used. Hefazi *et al.*, 2003(c) shows its application to a model problem.

## APPLICATIONS OF THE METHODS TO LIFTING BODIES.

### Problem description

Both classical and NN optimization approach are applied to the shape optimization of one of Pacific Marine's advanced lifting bodies. This patented underwater hull is made of two displacement bodies, called H-bodies, linked with a thin foil, referred to as cross-foil, and attached to the parent hull by two struts (Fig. 2). The entire arrangement is referred to as the twin H-body configuration and can be fitted to catamaran or pentamaran hull forms. The displacement bodies are designed to provide good sea-keeping properties at lower speeds when the hulls of the catamaran or pentamaran are partially submerged, while the cross-foil is designed to provide additional lift at higher speed when the multiple hulls are lifted out of the water in order to reduce drag.

A very similar configuration was optimized under a previous work reported by [Hefazi *et al.*, 2002], using the classical optimization process. Sea trials of a half-scale of this configuration on a 44 ft test platform were conducted to provide data for the validation and demonstrate the application of lifting body technology on a real test platform [Hefazi *et al.*, 2003]. This configuration is used in the so-called HDV-100 Technology Demonstrator. HDV-100 has a "Deep-V" hull form and is currently being constructed in Westerly Marine in Santa Ana, California. It is expected to go to sea in 2004.

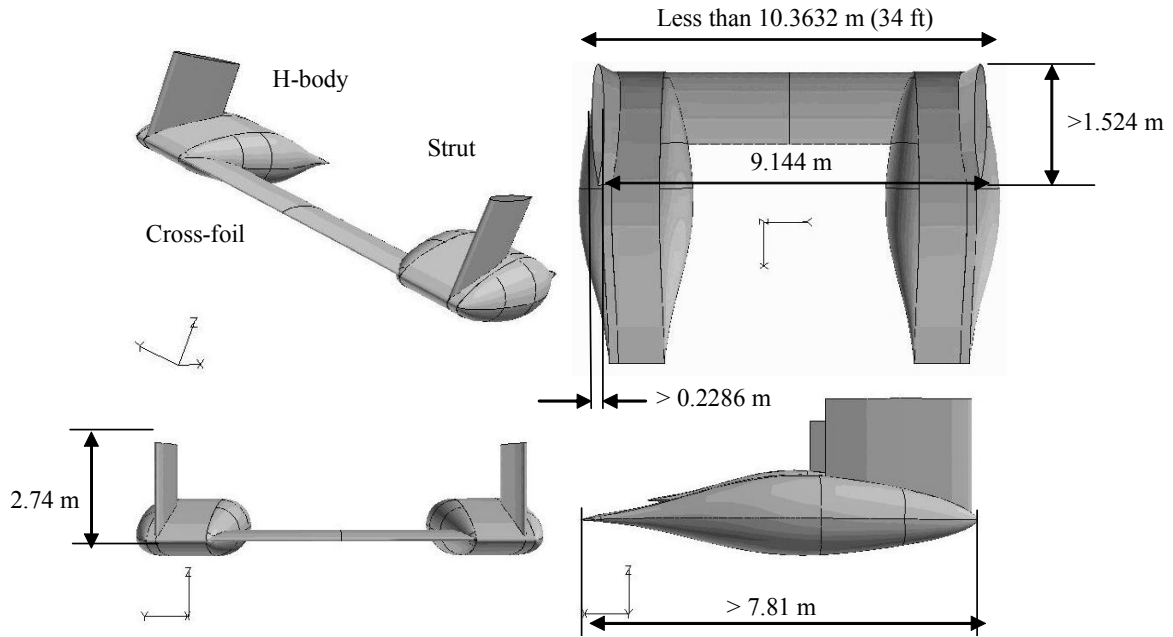


Fig. 2 Twin H-body configuration, Baseline

The optimization is performed for a boat speed of 47 knots (24.2 m/s). At this velocity, the parent hull will run dry. The whole configuration must be able to generate a total lift of 80 LT, including a minimum of 16 LT of displacement.

The objective is to maximize range, i.e. lift-to-drag ratio, which corresponds to minimizing drag at constant lift. In addition, the configuration is to be designed such that it can operate cavitation free at 52 knots. Additionally, the operating draft is to be 2.74 m (waterline to lowest point - 9 ft) and a structural constraint is to be imposed to prevent yield of the cross-foil. Also, the maximum width of configuration shall not exceed 10.36 m (34 ft) and the strut to strut distance is fixed by requirements for mating to upper hull.

From an optimization problem point of view, the objective function and design constraints can be summarized as:

- Objective: Maximize L/D
- Constraints:
- Operational speed of 47 knots (i.e. Reynolds number is  $211.63 \times 10^6$  based on a reference chord of 8.69 m in 75 F HI waters).
- Cavitation free at 52 knots, i.e. (HI water at 75 F).
- Total lift equal to 80 LT.
- Displacement greater than 16 LT.
- 2.74 m (9 ft) operating draft.

- Strut centerline to strut centerline distance is fixed at 9.14 m (30 ft) for mating with upper hull.
- Overall beam length should not exceed 10.36 m (34 ft).
- Minimum strut thickness 0.2286 m (0.75 ft).
- Minimum strut chord of 1.524 m (5 ft).
- The cross foil should have the structural integrity not to yield using a material of yield strength of 344.7379 Mpa (50 ksi), assuming solid section.
- Displacement pod length should not exceed by more than 20% the parent body (defined as optimized configuration of Hefazi *et al.*, 2002), i.e. 7.81 m.

### Classical optimization

#### OPTIMIZATION SETUP (AND TRAINING SET GENERATION)

As described above, the method integrates several commercially available software packages with several scripts, which, once setup, perform all tasks automatically, without user intervention. This automatic setup is critical in the optimization process. *iSIGHT* controls the process and calls the various scripts, which, **from the design variables**,

1. Define and generate the new geometry (Pro-Engineer);
2. Check for any constraint violation (from output of Pro-Engineer);
3. Generates a mesh suitable for the CFD method;
4. Executes the CFD method; and
5. Extracts the data needed by *iSIGHT* (**objective function and constraint values**) and calculate the **constrained objective function**,  $f_c$ , for the next iteration.

The constrained objective function is such that, when at least one constraint is strongly violated,  $f_c$  is set close to  $f_{max}$ .  $f_{max}$  is typically on the order of 1 and corresponds to a normalized value of the maximum objective function. The normalization value is given by the user and is typically chosen at the higher values of expected  $f$  over the search space. For example, with an optimization where  $L/D$  is the objective function on the order of 10-15, a normalization value of 10 to 20 can be chosen. Choosing 10 would mean that  $f_c$  might reach 1.5.

Two positive parameters  $\varepsilon_1$  and  $\varepsilon_2$  are now defined. If  $\forall i, g_i \leq -\varepsilon_1$ , then the constrained objective function is  $f$ . If  $\exists i \mid g_i \geq \varepsilon_2$ , then the penalized cost function gets close to  $f_{max}$  depending on how the constraints are violated. In other words,  $\varepsilon_1$  decides when constraints become active, and  $\varepsilon_2$  when they become prohibitive.

The generalized constraint  $G$  is defined as

$$G(x) = \frac{1}{n_{con}} \left( \sum_{g_i \geq -\varepsilon_1} g_i(x) + \varepsilon_1 \right)$$

and the constrained objective function becomes

$$f_c(x) = [1 - \varphi(y)]f(x) + \varphi(y)f_{max} \left( 1 - \frac{e^{-y}}{2} \right)$$

where

$$\varphi(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ e^{-1/x^2} & \text{if } x > 0 \end{cases}$$

and

$$y = \frac{10G(x)}{(\varepsilon_1 + \varepsilon_2)}$$

The generation of the training and validation sets is performed using the same approach, except that instead of using *iSIGHT* setup to run an optimization (Genetic Algorithms in the present case), it is designed to run Latin Hypercube samplings of the desired TS and VS sizes.

The process implemented for the Twin-H body optimization is shown in Fig. 3. The first step involves generating a geometrically feasible configuration from the selected set of design variables (28 here). This process is problem dependent in its implementation and, in general, involves two parts:

1. Airfoil shape definitions (“shape.in” files)
2. Configuration parameterization (“.ptr” files)

Each individual component of the configuration is represented by a set of parameters and, once assembled, these components describe the entire configuration. Simple geometrical parameters, such as width, length, chord, etc., are used to characterize the elements. These correspond to the “Configuration parameterization”. Foil cross-sections are defined by their mean camber line and thickness distributions. Camber is parameterized by classical NACA 2-parameter camber. [Abbott 1959] The thickness distribution,  $y_{th}$ , is represented by a 6-deg. polynomial, with an additional term,  $a_0$ , for controlling the leading edge radius:

$$y_{th}(x) = a_0\sqrt{x} + \sum_{i=1}^6 a_i x^i, \quad 0 \leq x \leq 1$$

Depending on the problem, one or several independent scripts or programs are used to regenerate updated Pro-Engineer files (“.ptr” files). Then, Pro-Engineer automatically updates the geometry based on the revised data.

In the second step, constraints which may be determined from the newly generated solid model, such as volume, structural constraint, etc., are evaluated. In third step, a suitable mesh is automatically generated using ICEM-CFD. This mesh is then used along with the CFD input data file to execute the CFD code. When using an Interactive Boundary Layer approach [Cebeci 1999] for the CFD, a surface mesh is sufficient. Note that this surface mesh includes, in general, trailing wakes behind lifting surfaces as well as free-surface definition so that the inviscid method which assumes infinite Froude number (solution by negative image), could be replaced by a non-linear free surface method. See Hefazi *et al.*, 2002 for more details. The use of a Reynolds-Averaged Navier-Stokes method would require the use of a volume mesh which could be implemented with the tools used here since (ICEMCFD).

The last step involves the use of a constrained objective function which integrates both the objective function and constraints. It is used when a global optimization method (such as Genetic Algorithms) is used within *iSIGHT*. If a gradient-based optimization method is used, objective function and constraints are used separately from one another. Also, it is *not* used in the NN-based approach; instead of representing the single constrained objective function with the NN,

NN's for the objective function and each of the constraints will be generated.

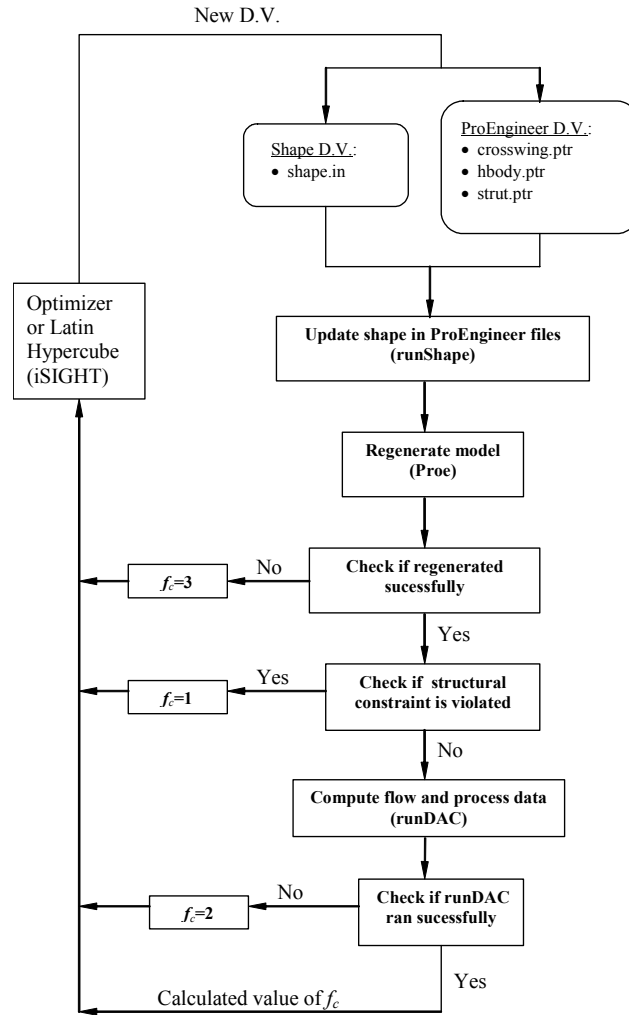


Fig. 3 Optimization with classical approach or training/validation set generation process

## RESULTS OF THE CLASSICAL OPTIMIZATION

A Genetic Algorithm optimization was run over the 28-dimension design space for 5000 iterations. The outcome was a configuration with an L/D of 12.92, which represents a 26% improvement in L/D. A comparison of the performance of the baseline and optimum configurations is shown in Table 1.

Table 1. Performance of optimized vs. baseline configuration (28 design variables)

	Baseline	Optimum	Objective
Displacement	18.4 LT	<b>16.1 LT</b>	> 16 LT
Total lift	80.2 LT	<b>81.6 LT</b>	= 80 LT
$C_{p,min}$	-0.263	<b>-0.250</b>	> - 0.269
L/D	10.23	<b>12.92</b>	Maximum

## NN Optimization

The optimization approach specific to the twin H-body optimization problem calls for the use of 5 single output neural networks as shown in Fig. 4, one for the objective function and the others for the constraints: lift-to-drag ratio (LOD), minimum pressure (Cpmin), dynamic lift (DL), buoyant lift (BL) and maximum stress value (struc). Alternatively, a single NN with 5 outputs could have been used, but typically, the resulting network is much more complex than 5 individual networks and, hence, takes more time to generate, i.e. train, than 5 networks.

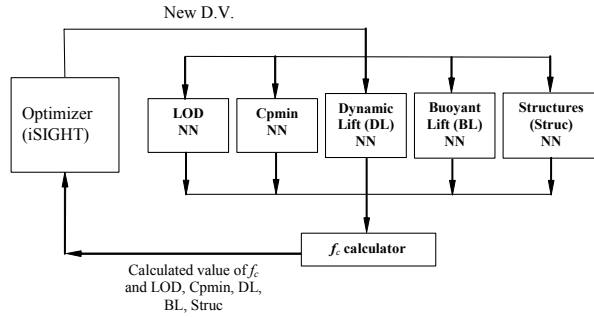


Fig. 4 Optimization process with NN approach

Also, as noted in the previous section, although the optimization is performed on the constrained objective function,  $f_c$ , it was not represented directly by a single NN because it would have required a very good training set definition in the –small– regions of the design space where the design was feasible. Instead, using 5 networks allows for accurate representation of each function over the design space and thus improved predictions for  $f_c$ .

## TRAINING AND VALIDATION SETS

For the present analysis, a validation set (VS) of 300 points was generated with the iSIGHT setup of Fig. 3 and using the same 28 design variables as above, but using a Latin Hypercube instead of an optimization algorithm. Three training sets (TS) were also generated using the same process, one with 1000 points, the second with 2000 points, and the third with 5000 points. Although a practical application may not involve the generation of multiple TS, this was done here to analyze the effect of TS size on result quality. Each data point takes approximately 6 to 7 minutes of computer time on an SGI Origin 3200 server.

For each size of Latin Hypercube sampling in the design space of Table 2, approximately 20 to 25% of the points were geometrically unfeasible (i.e. the model could not be successfully reconstructed) and had to be removed from the training sets. Nevertheless, the VS and TS will be referred to as 300-VS, 1000-TS, 2000-TS and 5000-TS subsequently.

## NEURAL NETWORK TRAINING

Each neural network (NN) was trained with 28 inputs (the design variables) and one output, i.e. one for each function: Buoyant Lift (BL), minimum pressure coefficient ( $C_{pmin}$ ), Dynamic Lift (DL), L/D (LOD), Structural constraint (Struc) and Total Lift (TL). In all cases, 7 candidate units were used along with a stopping criterion UP3 which consists in stopping training when the error on the VS has increased in three

consecutive strips of length  $k$ , where  $k = 5$ . After every  $k$  hidden units (HU) have been added to the network, the error on the VS is compared with the previous one and training is stopped if this error has increased three consecutive times. Then, the “best” number of HU (less than when the training stopped) is chosen for the NN which leads to the minimum validation error. This stopping criterion was determined by Hefazi *et al.*, 2003 (c) to be very accurate as it was shown in the study that the absolute minimum validation error was found each time on the two-dimensional model problem. For each function, 10 NN’s were built and the one that had the best validation error was chosen. Typically, training each network takes from about an hour (for 1000-TS) to a day for the more complete data sets (5000 TS). Unlike CFD methods which are demanding in computing power (I/O, memory), training demands relatively little other than CPU time and thus, all training can be done in parallel on the same server without interfering with other on-going computations. This feature is yet another advantage compared to training a single multiple-output network.

Table 2 presents the results of the training. Total lift (TL) has been included although it is not used. Instead, in the optimization method, it is calculated by adding buoyant and dynamics lifts as determined by their respective network. When the number of TS points increases, the squared Error, E/Pmax, decreases on both VS and TS, as would be expected. Also when the size of the TS increases, the number of HU found usually increases. This also makes sense since the number of points is relatively small compared to the complexity of the function over the design space. At some point, when there are enough points to describe accurately the function, one should expect the number of HUs to stop increasing. It should be noted, however, that the purpose of the current approach is not to reach this “best” number of HUs since this would require a very large number of points in the TS thus rendering useless the use of NN. Instead, the objective is to assess whether with a limited number of points in the TS, one is able to generate an improved design when compared to the classical CFD approach.

As for the maximum error,  $E_{max}$ , it is the maximum deviation between the targets and the outputs from the network over the entire TS or VS. Since the targets and outputs have been non-dimensionalized by the average target over the TS, this value gives an idea of the percentage of *maximum* error which can be expected with the NN replacing the CFD. This maximum error, although quite large for certain variables such as  $C_{pmin}$ , will be in general much smaller than that, and in particular, is *likely* to be much better at or near the optimum point.

The largest max errors are found on Cpmin (up to 34% for 1000-TS) and the structural constraint (15%). This is partly due to the fact that Struc and Cpmin are both  $C^0$  functions but not  $C^1$  and that the network is built to create  $C^\infty$  functions. One can also notice that in

general there is a significant improvement in maximum error on the TS and VS between 1000 and 2000, but that for 5000, there is no improvement especially on the TS. The reason for this behavior is unclear and is being investigated.

**Table 2.** Output from NN training. Note that errors are normalized

Function	TS	#HU	Error on TS		Error on VS	
			(E/Pmax)tr	(EMax)tr	(E/p)val	(EMax)val
BL	1000	6	1.044E-05	1.876E-02	2.025E-05	2.831E-02
	2000	15	5.959E-06	2.290E-02	1.351E-05	1.590E-02
	5000	17	4.011E-06	2.538E-02	4.548E-06	1.089E-02
Cpmin	1000	3	5.334E-04	3.369E-01	6.584E-04	2.023E-01
	2000	26	1.138E-04	8.772E-02	5.123E-04	1.604E-01
	5000	32	1.663E-04	2.754E-01	4.230E-04	1.497E-01
DL	1000	3	3.174E-05	9.733E-02	2.611E-05	2.215E-02
	2000	2	2.155E-05	7.087E-02	2.082E-05	2.059E-02
	5000	7	2.185E-05	1.115E-01	1.933E-05	2.306E-02
LOD	1000	2	2.758E-04	9.457E-02	3.672E-04	8.725E-02
	2000	5	2.154E-04	7.931E-02	2.957E-04	6.631E-02
	5000	13	1.605E-04	1.053E-01	2.340E-04	6.232E-02
Struc	1000	5	2.415E-04	1.398E-01	4.370E-04	1.521E-01
	2000	10	2.124E-04	1.369E-01	3.579E-04	9.857E-02
	5000	54	1.853E-05	2.446E-02	1.350E-04	1.234E-01
TL (not used)	1000	2	2.471E-05	8.064E-02	1.879E-05	2.349E-02
	2000	2	1.566E-05	5.904E-02	1.574E-05	2.126E-02
	5000	9	1.421E-05	9.617E-02	1.403E-05	1.939E-02

The neural networks generated in the form of 5 executables for Cpmin, LOD, DL, BL and Struc using the different sizes of training sets (1000, 2000 and 5000) were integrated in iSIGHT as shown in Fig. 4. A Genetic Algorithm (GA) optimization was used with an initial population of 50 and 35000 iterations were run based on the constrained objective function.

The best 100 runs resulting from the optimization with the five NNs (best  $f_c$ ) were then run using the Pro-Engineer model and the CFD code to compute the objective function and constraints and compare them with those of the NN near the optimum. Also, a few results from the optimization (i.e. as determined by the NN) with a slightly higher LOD than that of the best  $f_c$  but with constraints closer to their limit (therefore resulting in a lower  $f_c$ ) were chosen and also run through the CFD package. A summary of the results is shown in Table 3. For each size of training set, the table shows:

- The best  $f_c$  as determined by the optimizer (i.e. after 35000 GA iterations using the NN)

- The best LOD based on the NN with minimal constraint violations: corresponds to some hand-picked results from the optimization showing a slightly better LOD but with constraints close to the acceptable limits resulting in a higher  $f_c$
- The best  $f_c$  based on CFD results from the 100 best NN points (as determined by the GA)
- The best LOD based on CFD results from the 100 best NN points (as determined by the GA)

In each case, buoyant lift (BL), total lift (TL), lift-to-drag ratio (LOD) and minimum pressure (Cpmin) values computed by the NN and the CFD method are shown. The stress value (Struc) is not shown because it exhibited little variations between points and because the constraint was not violated. Also, the dynamic lift (DL) can be directly calculated from total and buoyant lift values (the optimization actually uses DL and BL to determine TL, but the latter is presented because it

corresponds to a primitive twin H-body requirement). It should also be noted that the constraints are not implemented as step functions but rather very steep functions which do vary near the constraint border. For example, the primitive requirement calls for a displacement or buoyant lift greater than 16 LT, but as

implemented here, a value close to 15 is acceptable. For this reason,  $f_c$  may vary in a counter-intuitive fashion rendering the analysis of its values difficult. It is therefore not shown.

**Table 3.** Results from NN optimization and comparison with CFD

TS	Results	Output from NN				Same DV but with CFD			
		BL	TL	LOD	Cpmin	BL	TL	LOD	CpMin
1000	Best $f_c$ optimized from NN	15.23	80.59	14.39	-0.265	15.24	81.59	13.76	-0.266
	Best LOD optimized from NN	14.75	79.94	14.44	-0.267	14.78	81.04	13.85	-0.268
	Best $f_c$ (using CFD) out of 100 best runs	15.49	80.81	14.30	-0.264	15.51	81.67	13.70	-0.266
	Best LOD (using CFD) out of 100 best	15.25	80.60	14.38	-0.265	15.25	81.64	13.78	-0.265
2000	Best $f_c$ optimized from NN	15.20	80.67	14.49	-0.265	15.13	80.86	13.70	-0.271
	Best LOD Optimized from NN	15.07	80.66	14.51	-0.265	15.01	80.78	13.81	-0.269
	Best $f_c$ (using CFD) out of 100 best runs	15.30	80.74	14.42	-0.263	15.23	80.96	13.71	-0.265
	Best LOD (using CFD) out of 100 best	15.17	80.59	14.47	-0.265	15.11	80.75	13.79	-0.270
5000	Best $f_c$ optimized from NN	15.17	80.64	14.43	-0.265	15.08	80.71	13.65	-0.275
	Best LOD Optimized from NN	15.05	80.90	14.48	-0.264	14.98	80.97	13.69	-0.274
	Best $f_c$ (using CFD) out of 100 best runs	15.18	80.19	14.09	-0.265	15.18	80.14	13.59	-0.274
	Best LOD (using CFD) out of 100 best	15.17	80.64	14.43	-0.266	15.08	80.71	13.68	-0.276

The differences between the NN and the CFD are within acceptable limits; they do not exceed the  $E_{max}$  shown in the previous section and are in general much less. In particular, they are very close in many instances. Also, differences between the values resulting from different selection processes for a particular TS are rather small. Finally, while one observes that the LOD is over-predicted by the NN, the differences between points are about the same for a given TS.

Unfortunately for 5000-TS, the Cpmin constraint is violated ( $<-0.269$ ) for the 100 “best” runs, resulting in only near-feasible designs. Also, the optimum LOD determined by this TS (around 13.65 based on CFD) is lower than found with the 1000 and 2000 TS (higher than 13.7 for both). Regardless of which TS is used, however, LOD is greatly improved from the 12.9 result obtained with the direct CFD method.

### Comparison between classical and NN-based method

All results for all TS show LOD ranging from 13.59 to 13.85, which is a definite improvement from the classical method which lead to 12.92. This improvement is due to the ability to increase the

exploration of the design space within the time available in a given design project. Because CPU time when using direct CFD is going to limit the design space exploration (5000 GA iterations require approximately one month of constant calculations here), a true optimum cannot be reached. **With as low as 1000 points generated to approximate the various functions over a design space with 28 design variables, improvements of about 34% in LOD compared with the original baseline twin H-body can be reached at one fifth the cost needed to get a 26% improvement!**

These results also point to a few improvements which would need to be implemented related to function differentiability (to improve Cpmin predictions, for example), to which constrained objective function to use, to the “best” size of the training set, and to the choice of optimization method. The latter comment is particularly pertinent for problems in which, like it might be the case for the results obtained with the 5000 TS, the optimizer (GA here) has focused its attention in a region of the design space where one (or more) function (objective or constraint) is not well approximated (Cpmin in this case). One could benefit from having an optimization

method with explores several regions of the design space, as illustrated in Lin 2002.

Finally, more sample test cases would need to be investigated to confirm the findings presented here. Results do show, however, that the method could provide the design engineer with a valuable tool for improving designs within a limited time frame and possibly at a lower cost than using conventional analysis tools.

## CONCLUSION

A novel approach combining automated CAD-based CFD and Neural Networks hydrodynamic shape optimization is presented. In this method, the CFD tool is “extracted” from the optimization loop and, instead, used to generate the (smaller) data sets needed for the NN training. The NN method is based on constructive Neural Networks trained using an improved Cascade Correlation algorithm. This topology allows for efficient Neural Network determination when dealing with function representation over large design spaces. One of Pacific Marine’s lifting bodies known as the twin H-body, (previously optimized under different conditions) is optimized using both the conventional optimization approach and that using the NN. The classical approach yields a design improvement of 26% while the NN-based method allows reaching a 34% improvement at one fifth of the cost of the former. Based on the analysis of the results, areas for future improvements and research are outlined. The results, in general, demonstrate the potential of the method in saving valuable development cycle time and increasing the performance of future ship designs.

## ACKNOWLEDGEMENTS

This work was conducted in collaboration with Pacific Marine of Hawaii. The authors would like to express appreciation to Mr. Steven Loui and his group at Pacific Marine for their various contributions. The authors also would like to acknowledge Mr. David Egan for providing all of the CAD and grid generation models and interfaces. Finally this work was sponsored by the Center for Commercial Deployment of Transportation Technologies. The authors thank Stan Wheatley, Steven Hinds, Dr. Ken James and Carrie Scoville for their support.

## REFERENCES

Abbott, I.H. and Von Doenhoff, A.E. *Theory of Wing Sections*, Dover Publications, Inc., NY, 1959.

Besnard E., Schmitz A., Kaups K., Tzong G., Hefazi H., Chen H.H., Kural O., and Cebeci T., "Hydrofoil Design and Optimization for Fast Ships," *Proceedings of the 1998 ASME International Congress and Exhibition*, Anaheim, CA, November 1998.

Cebeci T., *An Engineering Approach to the Calculation of Aerodynamic Flows*, Horizon Pub., July 1999.

Engineous Software, Inc., iSIGHT software, <http://www.engineous.com/>

Fahlman S.E. and Lebiere C. “The Cascade-Correlation Learning Architecture,” Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1990

Hefazi H. *et al.* “CFD Design Tool Development and Validation, CCDoTT FY 00 Task 2.8 report.” Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Feb. 2002. Available on-line at [www.ccdot.org](http://www.ccdot.org)

Hefazi H. *et al.* “CFD Design Tool Development and Validation, Part II, At Sea Trials, CCDoTT FY 01 Task 2.14.1 report.” Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Jan. 2003. Available on-line at [www.ccdot.org](http://www.ccdot.org)

Hefazi H. *et al.* “CFD Design Tool Development and Validation, Part I, CCDoTT FY 01 Task 2.14 report.” Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, January, 2003 (a) Available on-line at [www.ccdot.org](http://www.ccdot.org)

Hefazi H. *et al.* “Computer Software Product End Item, Deliverable 2, Optimization Tool Development Based on Neural Networks Computer DI-MCCR-80700 report.” Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003(b). Available on-line at [www.ccdot.org](http://www.ccdot.org)

Hefazi H. *et al.* “Software Test Report, Deliverable 2, Optimization Tool Development Based on Neural Networks DI-IPSC-81440A report.” Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003(c). Available on-line at [www.ccdot.org](http://www.ccdot.org)

Hefazi H. *et al.* "Computer Software Product End Item, Deliverable 3, Optimization Tool Development Based on Neural Networks Computer DI-MCCR-80700 report." Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003(d). Available on-line at [www.ccdot.org](http://www.ccdot.org)

Hefazi H. *et al.* "Software Test Report, Deliverable 3, Optimization Tool Development Based on Neural Networks DI-IPSC-81440A report." Center for the Commercial Deployment of Transportation Technologies, Long Beach, CA, Sept. 2003(e). Available on-line at [www.ccdot.org](http://www.ccdot.org)

ICEMCFD Engineering, Inc., ICEMCFD software, <http://www.icemcfd.com/>

Kwok TY. and Yeung DY. "Theoretical Analysis of constructive Neural Networks," Technical Report HKUST-CS-93-12, Hong Kong University of Science and Technology, 1993.

Lin C.Y. and Wu W.H., "Niche Identification Techniques In Multimodal Genetic Search With Sharing Scheme," *Advances in Engineering Software* 22, pp. 779-791, 2002.

Parametric Technology Corporation, Pro/Engineer software, <http://www.ptc.com/>

Schmitz A., Besnard E., Vives E. "Reducing the Cost of Computational Fluid Dynamics Optimization Using Multi Layer Perceptrons." ICJNN2002\_1391, International Joint Conference on Neural Networks, Hawaii 2002.

Vanderplaats, Miura & Associates, Inc., "DOT Users Manual, Version 4.10," VMA Engineering, 1994.