CFD & OPTIMIZATION 2011 - 000
An ECCOMAS Thematic Conference
23-25 May 2011, Antalya TURKEY

AERODYNAMIC OPTIMIZATION OF HIGH-SPEED TRAINS NOSE USING A GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORKS

J. Muñoz-Paniagua*, J. García* and A. Crespo*

*Grupo de Investigación de Mecánica de Fluidos apicada a la Ingeniería Industrial
Escuela Técnica Superior de Ingenieros Industriales, Universidad Politécnica de Madrid
C/ José Gutiérrez Abascal 2, 28006, Madrid, Spain
e-mail: le.munoz@upm.es

Key words: Shape Optimization, High-speed train, Genetic Algorithm, Metamodel, Artificial Neural Network, Bèzier Curves

Abstract. *An aerodynamic optimization of the train aerodynamic characteristics in term of front wind action sensitivity is carried out in this paper. In particular, a genetic algorithm (GA) is used to perform a shape optimization study of a high-speed train nose. The nose is parametrically defined via Bèzier Curves, including a wider range of geometries in the design space as possible optimal solutions. Using a GA, the main disadvantage to deal with is the large number of evaluations need before finding such optimal. Here it is proposed the use of metamodels to replace Navier-Stokes solver. Among all the posibilities Response Surface Models and Artificial Neural Networks (ANN) are considered. Best results of prediction and generalization are obtained with ANN and those are applied in GA code. The paper shows the feasibility of using GA in combination with ANN for this problem, and solutions achieved are included.*

1 INTRODUCTION

The increasing importance of trains as an alternative means of transport, which also complements reducing the traffic of an already overwhelmed communications network, justifies the need to carry out studies related to improve its performance. Even when the train makes the most economical use of energy form of public transport, reducing energy consumption is pretended. Consumption is directly related to its aerodynamic. Then, it is necessary to perform aerodynamic studies of the train in order to improve its efficiency. The aim of the research is the definition of optimal nose shapes which involve a minimal drag coefficient. This is traditionally done by a trial-and-error procedure, but it is very expensive in terms of machine and designer time, and rely heavily on previous analyses. Instead, an automatic method of optimization of aerodynamic shapes is proposed here. This method involves the use of genetic algorithms (GA) as the optimization tool[1].

GA are a technique that mimic the mechanics of natural evolution. Once a population of potential solutions is defined, it combines survival-of-the-fittest concept to eliminate unfit characteristics and utilizes random information exchange, with exploitation of knowledge contained in old solutions, to effect a search mechanism with power and speed[2]. Iteratively, better results are obtained until a solution closer to globally optimal solution is reached. However, the main drawback when using GA is their need of a large number of evaluations of the objective function. Furthermore, this problem is considerably more important when evaluations are computational cost-effective.

To remedy this inconvenience, the use of metamodels is proposed here. The basic idea of metamodels is to construct approximations of the analysis codes or numerical solvers that are more efficient to run, enabling a faster evaluation and optimization process. Metamodeling involves choosing an experimental design for generating data, choosing a model to represent the data, and then fitting the model to the observed data. Response Surface Models[3] (RSM) and Kriging models[4] have been already applied as metamodels in high-speed train optimization. In this paper, a comparative study of RSM and Artificial Neural Networks is performed.

2 OPTIMIZATION METHODS

As it has been indicated, the objective is to geometrically optimize the nose of a high-speed train by minimizing its drag coefficient when it is exposed to a frontal wind. This single-objective optimization problem can be defined by[5]

$$\begin{array}{lll} \text{Minimize} & f(\vec{\mathbf{x}}) & \\ \text{subject to} & g_j(\vec{\mathbf{x}}) \leq 0 & j = 1...m \\ & h_l(\vec{\mathbf{x}}) = 0 & l = 1...n \\ & x_i^l \leq x_i \leq x_i^u & i = 1...k \end{array} \tag{1}$$

being $\vec{\mathbf{x}}$ the vector of design variables and $f(\vec{\mathbf{x}})$ the objective function. The optimal design minimize this function. The inequality and equality constraints represent respec-

tively constraints to be satisfied by the optimal candidate and relations between its design variables. The different optimization methods existing are classified depending on the order of derivatives of the objective function used. Zero-order methods, such as random search, simulated annealing and evolutionary algorithms (among which GA are included) use only the function values in their search for the minimum, while first and second order methods use respectively the first and second derivatives, commonly known as gradient methods and Newton method. Although the latter are more precise, they require some gradient information of the objective function, which can be a numerically intensive task, especially if the number of design variables is large and if one single evaluation is numerically expensive[5]. Moreover, the reliability and success of gradient methods generally requires a smooth design space and the existence of only a single global extremum, or an initial guess close enough to the global extremum that will ensure proper convergence[6]. Since in the field of aerodynamics, objective functions often have multi-peaks[7], it is expected that non-gradient methods will work more efficiently. Then, for a multimodal and high-dimensional design space GA are proposed as optimization method.

2.1 Genetic Algorithm

A GA is a stochastic optimization method based on darwinian natural evolution. It repeatedly modifies a set of individuals (population) considered as optimal candidates by means of three operators, selection, crossover and mutation. At each iteration the algorithm selects individuals at random from the current population to be parents, and use them to produce the children for the next generation. Although such operation works randomly it is driven in such a way that benefits the selection of those individuals who result in the fittest candidates. Children are produced either by making random changes to a single parent (mutation operator) or by combining a couple of parents (crossover). There are a large number of different definitions of each operator[1]. Both operations are performed with an specific probability, mutation probability P_m and crossover probability P_c respectively. Once new individuals are obtained, the algorithm replaces the current population with the children to form the next generation, and the population size remains constant. The optimal values is always searched for within a group of possible solutions, which is an important difference from other one-by-one basis search methods. Figure 1 shows schematically how the GA works. In order to stop the iterative process, a convergence test according to a prefixed stopping criteria is done after every new population is evaluated. If this is satisfied, it ends the cycle. Otherwise, it continues until convergence is observed.

Each individual is defined as a codified structure. The most common way to represent each one is by binary code, so an individual is a bit string. This string is created by concatenating a number of genes, being each one the codification of each design variable. Therefore, it is necessary to represent each possible optimal design (*i.e.* a high-speed train nose shape) as a design vector. The design vector consists out of parameters that define the shape of the geometry. These parameters, and their respective range, must be chosen

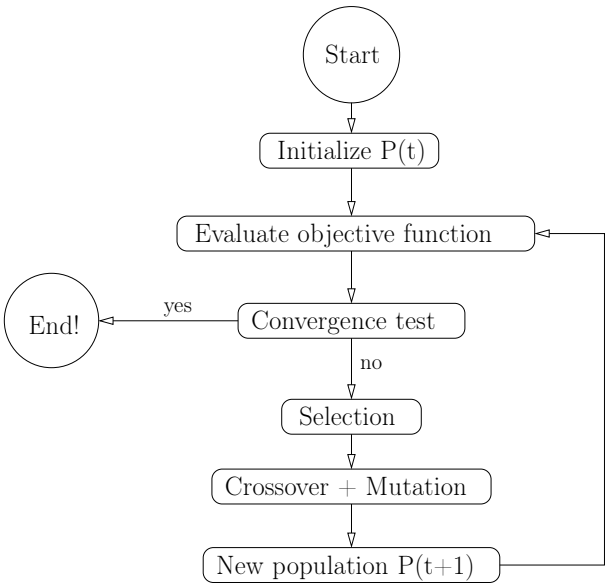


Figure 1: How the genetic algorithm works

carefully so that any geometry candidate to optimal is represented by them, while keeping its number as low as possible in order to reduce the design space and help the optimal search.

The objective function evaluation is the most crucial task in a GA performance. The time it takes to carry out all the evaluations the algorithm needs to reach an optimal design will determine the efficiency of it. The total number of evaluations is directly related to the population size and the number of generations. It is obvious that the population size should be large enough to guarantee a satisfactory genetic diversity, which is essential to the GA because it allows the algorithm to search a larger region of the design space[10]. Thus, to cut down the time cost without affecting significatively the population size, surrogate models (metamodels) are constructed from and then used in place of the actual simulation models.

A schematical representation of the whole optimization scheme applied in this paper is shown in figure 2. A more detailed analysis of it is presented in the results section. Within the GA flow chart from figure 1, these two tasks (individual codification and metamodel working) are included. In the following sections the parametric design of each individual and a introduction about the metamodels utilized in this study are introduced and developed deeply.

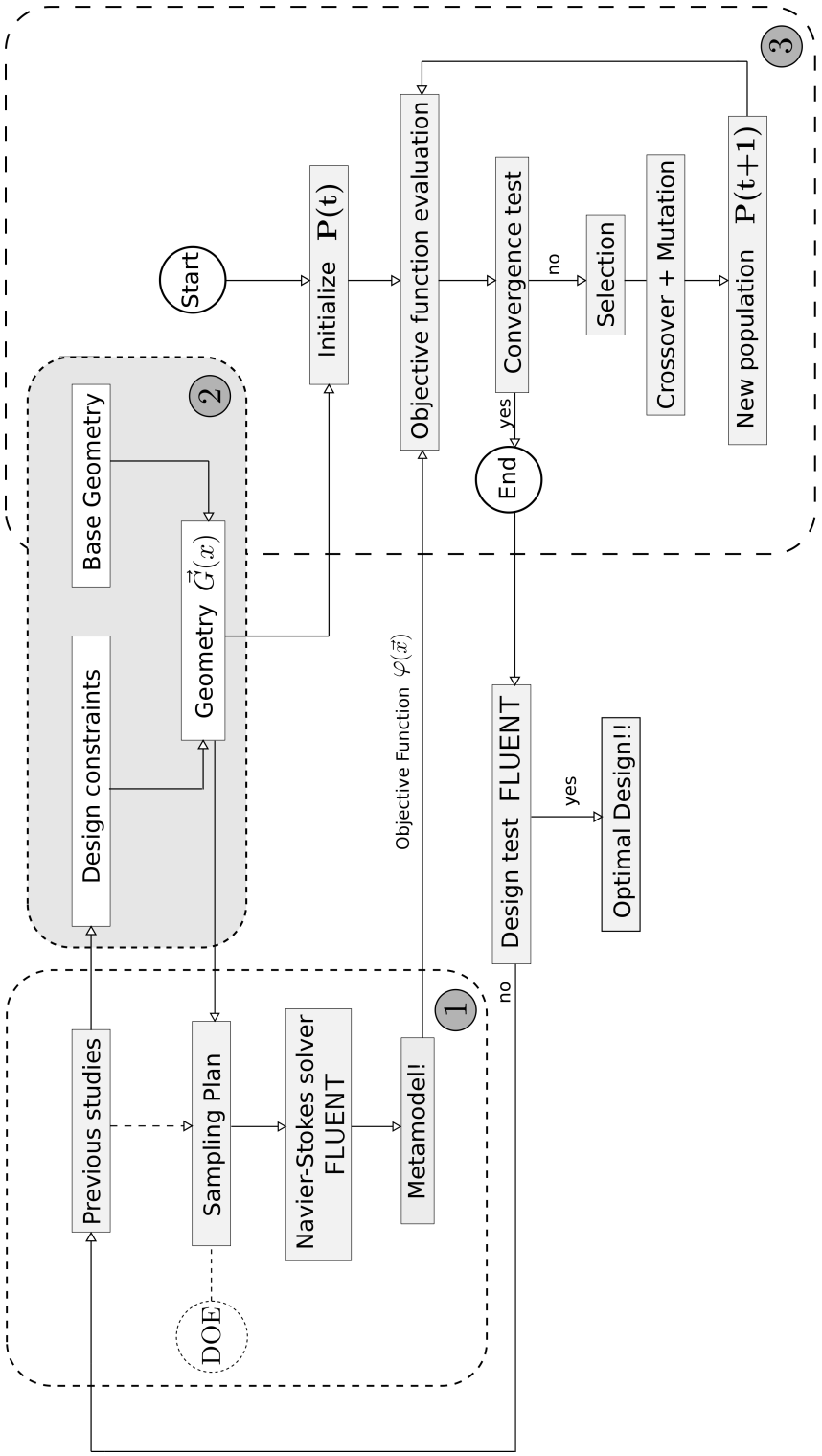


Figure 2: Schematical representation of the whole optimization scheme

3 OPTIMIZATION APPROACH

3.1 Parametric design of a high-speed train nose

In this section we present the parametrization of the nose shape of a high-speed train. The aim is to represent any possible geometry as a design vector. Some publications have introduced different alternatives. Guilmineau[12] proposes the use of potential flow equations combined with classical airfoil parametrization to obtain the geometry of a car model. Chiu and Squire[13], and Krajnovic[3], make use of simple ellipical and parabolic equations to generate a 3-D nose shape, while Vytla[4] uses five control variables to define a simple 2-D geometry. Kwon[10] and Lee[11] give a more complete and sofisticated alternative by means of the Hicks-Henne function for defining the geometry. This function is widely used in train aerodynamic designs and airfoil shape design. It is defined as

$$G(x) = G_{base} + \sum_{i=1}^{n} w_i f_i \tag{2}$$

where G_{base} is referred to the baseline shape (typically a parabolic function) and w_i and f_i denote the weighting factor and the shape function respectively. f_i usually is a sinusoidal function. Instead, here we propose the application of Bézier curves[14] for defining the geometry of a 2-D high-speed train nose. Bézier curves are highly suited for the parametrization of a design. Comparing them with previous Hicks-Henne function, they have a simpler formulation by means of polynomial functions. Moreover, the characteristics of the curve are strongly coupled with the underlying polygon of control points, simplifying the link between parameters and real design variables.

The Bézier curve of degree n equation is given as

$$\mathbf{C}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \, \mathbf{P}_i \tag{3}$$

where $0 \leq t \leq 1$ is a parameter control, and $\mathbf{P}_i$ are the control points to be weighted. We have considered two quadratic and two cubic curves to obtain our base geometry. As result the geometry is somehow divided into four different parts in order to study the effect of the slope of both the front part and window, the relative length of the roof versus the characteristic nose length L, and the nose underbody. In total eleven control points are used (from $\mathbf{P}_0$ to $\mathbf{P}_{10}$). It means twenty-two design variables. However, it is possible to relate some of the control points by geometric relationships, and this number is decreased up to ten. Table 1 presents all the design variables and their range. Figure 3 shows the parametric design.

Values of L and H have been obtained from some references about aerodynamic of high-speed trains[15,16,17,18]. Raghunathan include an interesting study about the drag coefficient of different nose lengths. This lets us to define the range of variable l_1. Both angles are

Design variable	Range
l_1	[2.56 , 6.40]
l_9	[13.60 , 12.80]
α_1	[0.00, $\pi/2$]
α_6	[0.00, $\pi/2$]
k_2	[0.20, 0.80]
k_3	[0.20, 0.80]
k_4	[0.20, 0.80]
k_6	[0.20, 1.00]
k_7	[0.20, 0.80]
k_8	[0.00, 0.80]

Table 1: Example of the construction of one table

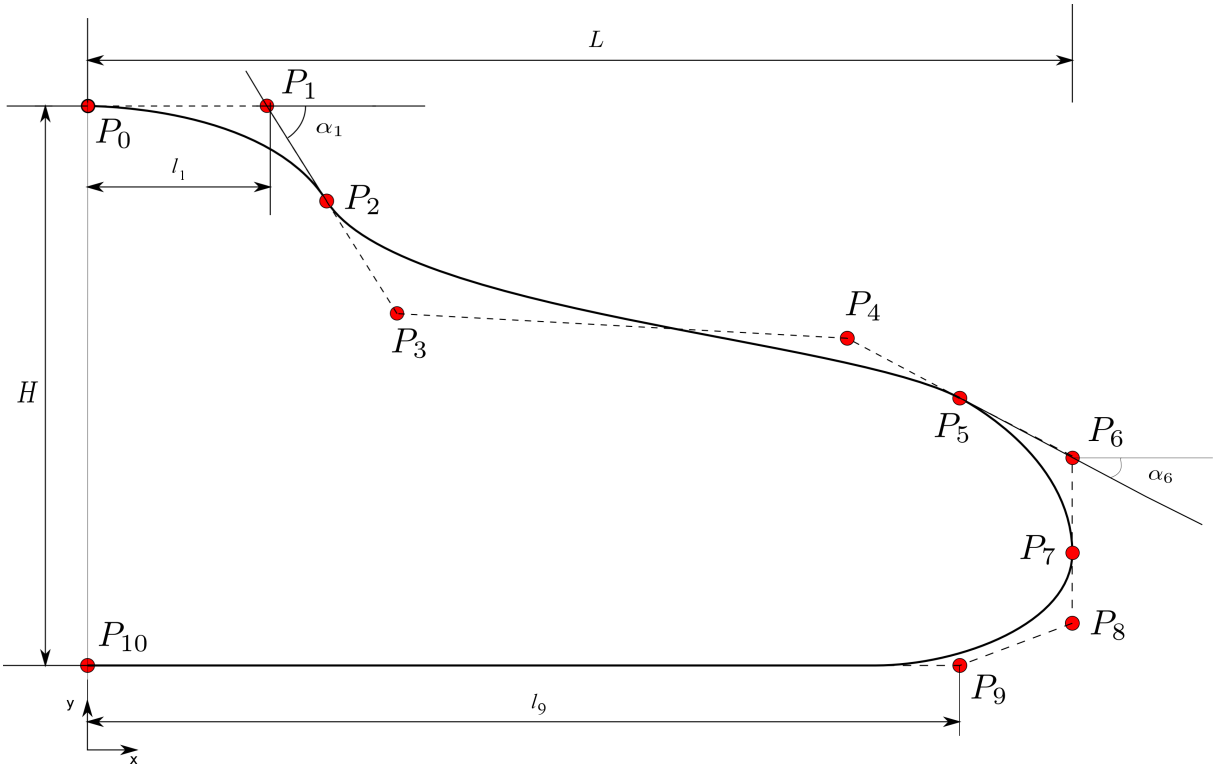


Figure 3: Base geometry parametrization

limited from 0 to $\pi/2$ to allow a wide variety of designs, and the geometric parameters k_i are defined in order to avoid superposition of the relative control points.

3.2 Metamodels. Response Surface Model and Artificial Neural Network

The expensive cost of running complex engineering simulations makes it impractical to rely exclusively on numerical codes for the purpose of aerodynamic optimization. Although it is possible to perform all the evaluations on a processors cluster[18] for a simplified geometry, it still results in a too long process. By using approximation models, we replace the expensive simulation model and speed up the genetic algorithm performance. A variety of metamodelling techniques exist, and an excelent comparison and review of these methods can be found in some references[19,20]. Jin[20] points out that there is no best one, although some works better than others for a specific application. To compare them accuracy, efficiency, robustness, model transparency and simplicity must be taken into account. Therefore, the simplest one, Response Surface Model, is compared in this paper with Artificial Neural Networks, which is a well-considered technique for large-scale problems (ten or more design variables) and low-order nonlinearity (square regression around 0.99 when using first or second-order polynomial model).

The classical polynomial response surface model (RSM) is still probably the most widely used form of surrogate model in engineering design. A second-order polynomial model can be expressed as

$$\hat{y}(\vec{x}) = \varphi(\vec{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_i \sum_j \beta_{ij} x_i x_j \tag{4}$$

The coefficients β are determined by least squares regression analysis by fitting the response surface approximations $\varphi(\vec{x})$ to already evaluated designs $y(\vec{x})$. A more complete discussion of RSM is found in Myers[22]. The main drawback is that for a second-order polynomial, the number of unknowns (coefficients) is proportional to $(k + 1)(k + 2)/2$, so it is clear that it is not the best method to apply when a high-dimensional design space is considered.

An artificial neural network (ANN) is composed of many very simple processing units (neurons) connected to form a network. Each connection is characterized by the corresponding weight, which specifies the effect of each unit on the overall model. Among all different types of neural networks, multi-layer percepton (MLP) is considered in this paper. In an MLP, the network is arranged in layers of processing units: an input layer, one or more hidden layers, and an output one. In our ANN the input layer, composed of as many input units as dimensions of the design space, is connected to the unique hidden layer with n_h hidden neurons, which finally is connected to the output layer of just one output unit. Then, the approximation function is given as

$$\varphi(\vec{x}) = \sum_{j=1}^{n_h} w_j^o z_j(\vec{x}) + b_j^o \tag{5}$$

where w_j^o is the weight given to the connection of the j-th hidden neuron and the

output unit, b_j^o the error or bias associated to the j-th hidden neuron and $z_j(\vec{x})$ is the base function. While in the case of polynomial models this function is prefixed, in ANN $z_j(\vec{x})$ is defined as

$$z_j(\vec{x}) = g(\sum_{i=0}^{k} w_{ij}^h x_i) \tag{6}$$

being w_{ij}^h the intensity of the connection between the i-th unit of the input layer and the j-th one in the hidden layer. Function g$(\cdot)$ is known as the activation function. A wide variety of activation functions exists. Here sigmoidal function is applied. It is expressed as $g(a) = \frac{1}{1+e^{-a}}$. The composition of both functions gives the relation between inputs and outputs for a ANN.

$$\varphi(\vec{x}) = \sum_{j=0}^{n_h} w_j^o g\left(\sum_{i=0}^{k} w_{ij}^h x_i\right) \tag{7}$$

Consequently, the unknowns to be adjusted are the connection weights w_j^o and w_{ij}^h, and the number of hidden neurons n_h. The determination of the unknown parameters is called the training of the ANN. Back-propagation is the most commonly used method for training of multilayer networks. It is a form of supervised learning method. The desired output for a given set of input (N_{trn}) is known during training. The error at each neuron is calculated as the difference btween the approximated solution and the desired output. This error is then asigned to each of the hidden neurons according to the output values of each hidden neuron and its relative connecting weight. In order to minimize such error, the connecting weights are modified and corrected. Training process continues until training error is minimal. However, sometimes the network is able to learn the training data, and laterly it is unable to predict new unobserved data. To avoid it, another set of data, N_{val} is considered to compute the validation error and to indicate if overfitting is observed while training process is running. Finally, a third set of data is used to check the prediction capability of the metamodel once the training has finished. In this way, the existing or evaluated designs to fit the metamodel have to be divided into three different subsets.

4 RESULTS

Metamodel building involves choosing an 'experimental' design for generating a database and fitting the model to the observed data. The accuracy of the approximation predictions depends on the information contained in the database. To maximize this information and fit the model by a representative sample of the design space Design of Experiments (DOE) method is applied. Traditionally, experimental designs like factorial or fractional factorial designs, developed for effective physical experiments, have been used. However, because of the deterministic nature of the response, no random error exists in a computer

experiment. So, it is correct to admit that classical DOE are not the best proposal when dealing with computer analysis codes. As an alternative here it is proposed to use space-filling designs. Among all the space-filling techniques available, we use Latin Hypercube Sampling (LHS)[21]. LHS can be configured with any number of samples, and is not restricted to sample sizes that are multiples of k. This allow us to test the effectiveness of both metamodels (RSM and ANN) for different database sizes, listed in table 2. Three different sizes are considered, minimal, small and large data sets.

Database	Theoretical size	Actual size
minimal set	n_p	66
small set	$10k$	100
large set	$3n_p$	198

Table 2: Sample data set size

n_p is the number of coefficients in a second-order polynomial model, given as $n_p = (k + 1)(k + 2)/2$, where k is the number of design variables.

First, RSM effectiveness is studied. According to Myers[22], statistical methods like t-test or F-test are inadequated when deterministic responses are analyzed. Then, to measure the accuracy of the model three different metrics are used: R-square (R^2) and R-square adjusted (R^2_a), and Relative Maximum Absolute Error (RMAE). The equations for theses three metrics are given as

$$R^2 = 1 - \frac{SS_E}{SS_T}$$

where SS_E is the sum of squared errors, and SS_T is the total sum of squares, defined as $SS_T = \sum_{i=1}^{N} y_i^2 - \frac{1}{N}(\sum_{i=1}^{N} y_i)^2$, being y_i the response for sample design i included in the initial database of size N.

$$R^2_a = 1 - \frac{SS_E/(N - n_p)}{SS_T/(N - 1)}$$

$$RMAE = \frac{\max(|y_1 - \varphi_1|, |y_2 - \varphi_2|, \ldots, |y_N - \varphi_N|)}{\sigma}$$

where σ stands for standard deviation. According to it, the larger the value of R-square is, the more accurate the metamodel results. The same behaviour presents R-square adjusted, while small RMAE is preferred. To provide a more complete picture of metamodel performance, checking their capacity of prediction as well, two more metrics are defined, Prediction Error Sum of Squares (PRESS) and an equivalent prediction R-squared (R^2_p). Their equations are given below.

$$PRESS = \sum_{i=1}^{N} (y_i - \varphi_{(i)})^2$$

$$R_p^2 = 1 - \frac{PRESS}{SS_T} \tag{8}$$

Each data set is obtained by running the number of simulations indicated in table 2. The simulations are performed using the commercial Navier−Stokes solver FLUENT, considering the flow as incompressible. A second order upwind method was chosen for the discretization of the Navier−Stokes equations while the velocity− pressure coupling was done by a SIMPLEC scheme. The realizable $k - \epsilon$ turbulence model implemented in FLUENT was considered to solve the equations. Table 3 summarizes the set size effect comparison using a second-order polynomial model.

Database	R^2	R_a^2	$RMAE$	$PRESS$	R_p^2
minimal set	1	1	0	-	-
small set	0.961	0.886	0.411	3.269	0.66
large set	0.951	0.927	0.667	2.095	0.889

Table 3: Summary of accuracy and prediction capabilities for different data set sizes.

Since using PRESS metric implies fitting the model with $N-1$ data points, no results are obtained for the minimal set. It is observed that the larger the data set is, the smaller R^2 becomes, and also larger RMAE is. It is obvious that more data points leads to a poorer fitting, but this is still high enough to accept such accuracy. Checking prediction capability, R_p^2 increases if a larger size value is considered. Therefore, the metamodel becomes more predictive as a more representative sample of the design space is used.

Now these results are compared with ANN performance. However, when using an ANN as a metamodel, some questions have to be answered previously. To start, which is the best architecture for the network in this particular problem, *i.e.* how many hidden layers and how many neurons in this layer have to be used. The Kolmogorov Theorem[5] specifies that any continuous function, from $\mathbb{R}^n$ to $\mathbb{R}^m$, can be represented exactly by a one-hidden-layer network, with n and m elements in the input and output layer respectively. Hence, this architecture is utilized in this paper. Different authors indicate that there is no yet an analytical method to determine which is the exact number of hidden neurons n_h to be taken. So, an estimating test must be carried out, trying different networks and choosing the 'best' one. Nevertheless, it is possible to start from a reference value n_{h_m}. If N is the number of data points utilized to fit or train the network, and it is divided into the three subsets already explained, the maximum number of hidden neurons n_{h_m} is given then by

$$n_h << n_{h_m} = \frac{N_{trn} \cdot n_{out} - n_{out}}{n_{in} + n_{out} + 1}$$

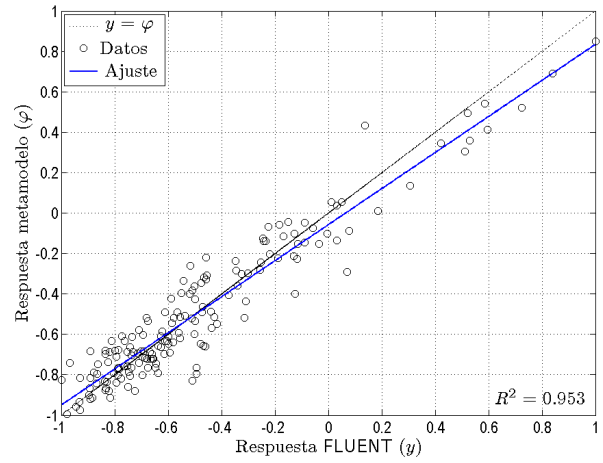
If $n_{in} = k$, and fixing just one output, this equation is translated to $n_{h_m} = \frac{N_{trn}-1}{k+2}$. We have concluded that RSM performs best with larger data sets. Thus, here N is taken as 198, to which 20 extra points are added. Having $N = 218$, and considering a proportional distribution of {75,10,15} for training, validation and testing respectively, $n_{h_m} = 13$ neurons. From this value a parametric study is carried out, decreasing the number of hidden neurons and analyzing the performance of the neural network. Ten different networks are taken into account with each n_h, choosing the one with the best R^2_{trn}. Table 4 listes the information about each n_h characteristic case.

n_h	R^2_{trn}	R^2_{val}	R^2_{tst}
13	0.997	0.958	0.987
12	0.989	0.978	0.971
11	0.993	0.971	0.981
10	0.972	0.944	0.979
9	0.968	0.924	0.944
8	0.953	0.928	0.936

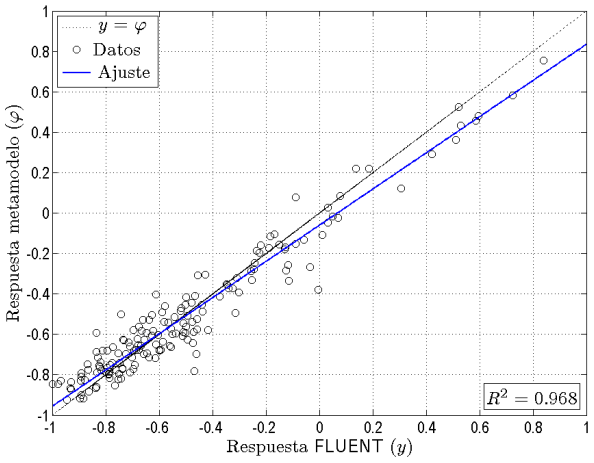
Table 4: Fitting information for ANN with different n_h n_h.

The lower n_h is, the worse fitting becomes, although it still remains extraordinarily high, even for $n_h = 8$. It is concluded that the optimal value of n_h is included in the interval [11 - 13]. Figures complete this information. The axes represent the training and testing data. Real response, obtained with FLUENT, is plotted versus approximated response (output) from the metamodel. The response (C_d) is represented normalized in the interval [-1, 1]. The dashed line represents the ideal equality *true response* = *output*. The solid line represents the best fit linear regression line between them. Also R^2 values is indicated. The scatter plot helps to see that fitting is increased when more hidden neurons are utilized. From table 4 it is observed that validation fitting is worse when n_h is arised up to 13, what could mean existence of overfitting. Regarding at the testing figures, a considerable data dispersion is presented with 8 and 9 hidden neurons. With 10 neurons, a better fit is observed graphically than 11-neurons does although its R^2 value is lower. But when training fitting is considered as well, it seems to be more correct considering n_h = 11 as the optimal value. Nevertheless, because of such small differences both designs are used with the GA.

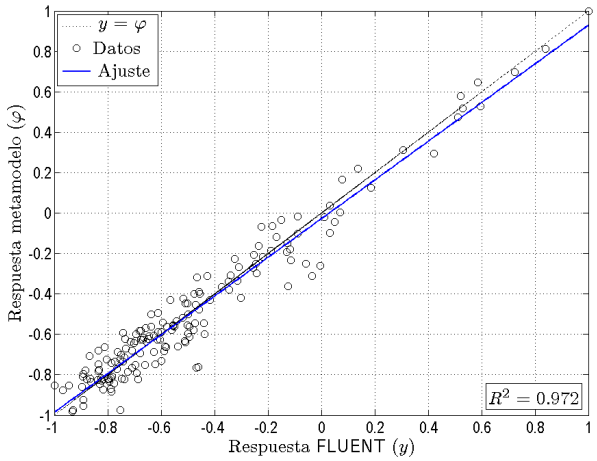
Comparing the response surface model and neural network fitting results, it is concluded that in this case ANN behaves better as an approximation model than RSM. Therefore, ANN are used instead of RSM in the GA running. Here the MATLAB code implementation, included in the Optimization Toolbox, is used for running the genetic algorithm. According to the theory previously introduced, some operation parameters have to be fixed. Selection, crossover and mutation operators, respective crossover and mutation probabilities and population size are parameters that can strongly affect on



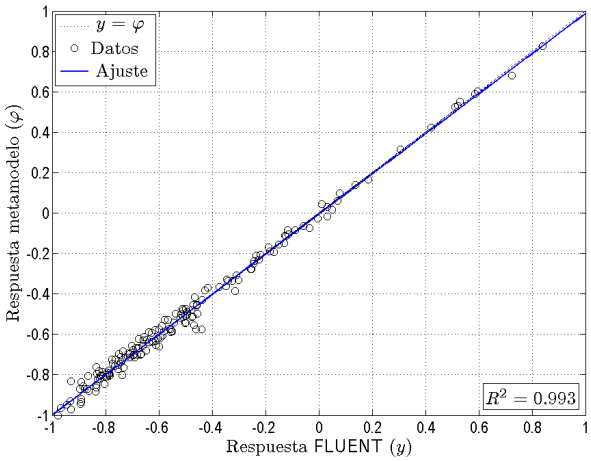
(a) $n_h = 8$



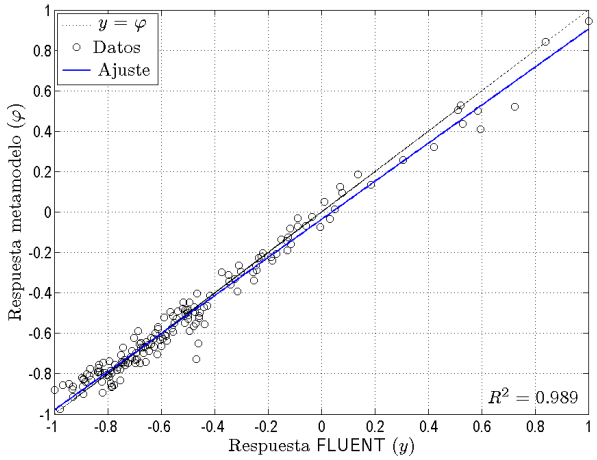
(b) $n_h = 9$



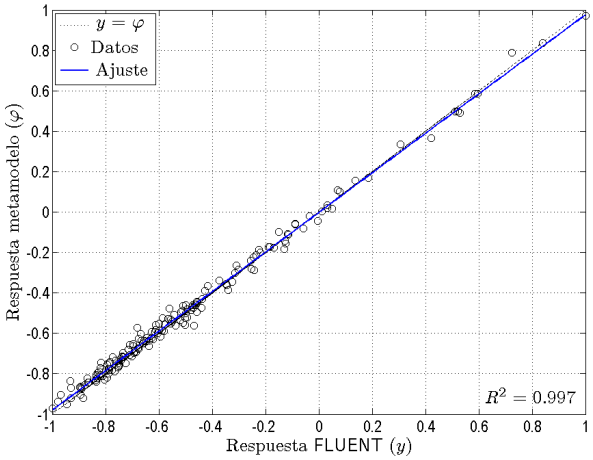
(c) $n_h = 10$



(d) $n_h = 11$

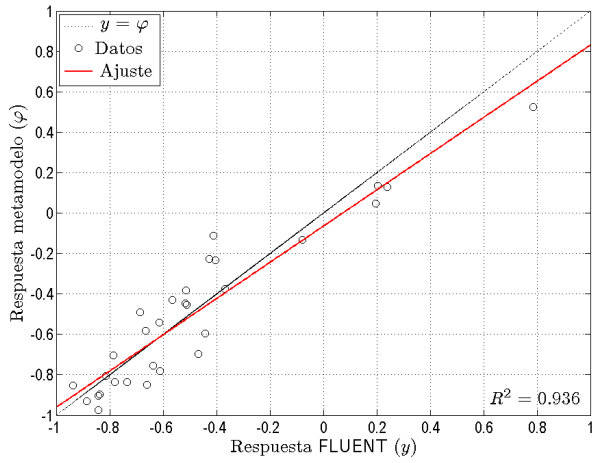


(e) $n_h = 12$

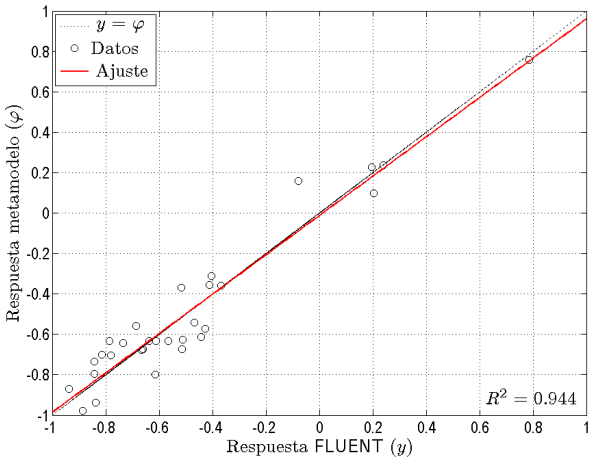


(f) $n_h = 13$

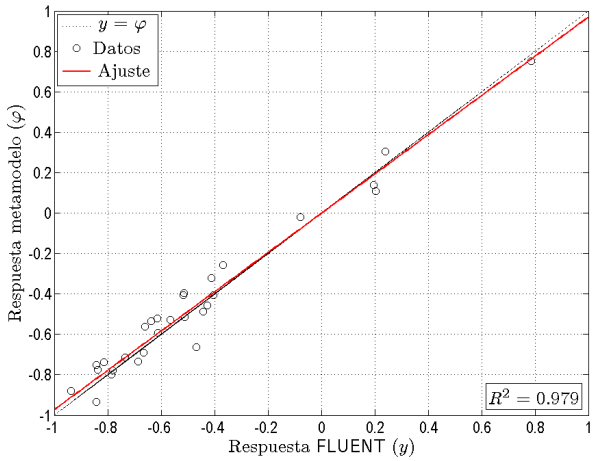
Figure 4: Training regression plots for neural networks considered in table 4.



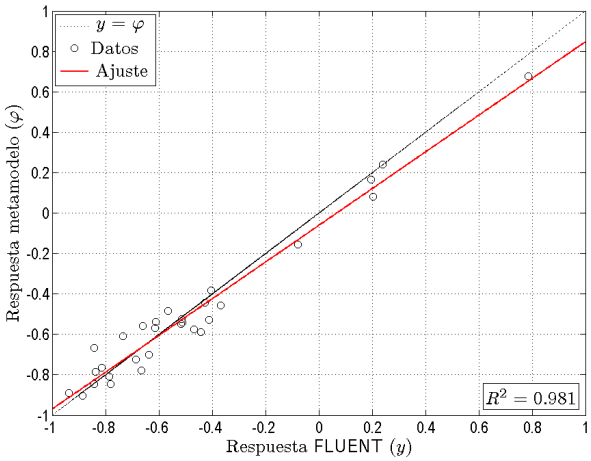
(a) $n_h = 8$



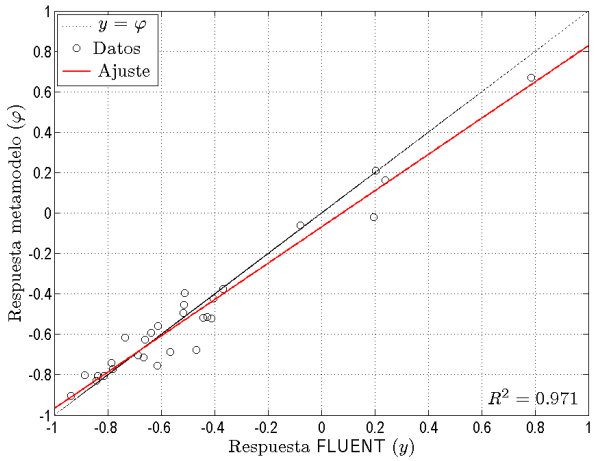
(b) $n_h = 9$



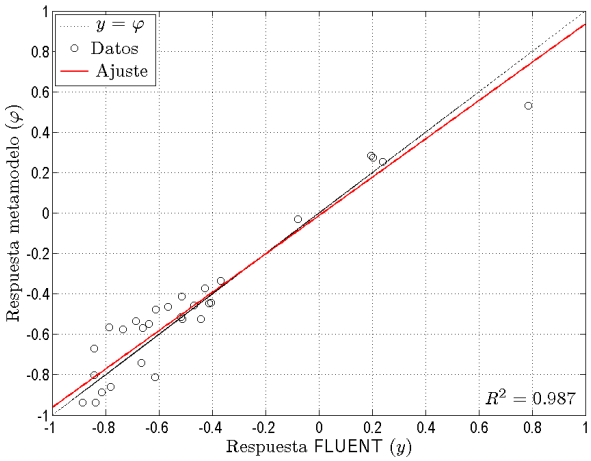
(c) $n_h = 10$



(d) $n_h = 11$



(e) $n_h = 12$



(f) $n_h = 13$

Figure 5: Testing regression plots for neural networks considered in table 4.

the GA efficiency. Therefore, a parametric study to determine which values allow a best performance is necessary. Ten different population sizes (from 20 to 200 individuals per population) are tested. Crossover probability is also studied, varying it from $P_c = 0.25$ up to 1. Mutation probability is fixed to 0.01. Table 5 summarizes the most important information from the algorithm applied.

Population size	20 - 200
Elitism	Yes
# elite individuals	2
Selection function	Tournament (4)
Scaling function	Ranking
Crossover function	Two-points
P_c	0.25 - 1
Mutation function	Uniform
P_m	0.01
# max generations	300

Table 5: GA parameters values.

In this way, 160 tests are performed. The best combination of these parameters is selected regarding at the optimal found by the algorithm. It is obtained when $P_c = 0.45$ and the population size is 160. The optimal design results in the following design variables values

l_1	α_1	k_2	k_3	k_4	α_6	k_6	k_7	k_8	l_9	C_d
-0.9993	0.0466	0.9901	-0.7751	0.0611	-0.9989	-0.9981	0.1822	-0.9657	0.8025	1.0013
2.5614	0.8229	0.7970	0.2675	0.5183	0.0009	0.2047	0.5547	0.0137	12.6159	1.0013

Table 6: Design variables and C_d for the optimal design.

Both normalized and absolute values of the design parameters are indicated. Figure 6 represents the optimal geometry, and figure 7 shows the pressure field around the nose. As it was expected, the optimal geometry is close to a strongly sharp nose with a small cone angle (α_6). Since angles α_1 and α_6 are almost complementary no inflection point is observed and a linear profile for the window and front part is obtained. Since k_6 is minimal the nose height is also very small.

The aim of using the ANN for optimization was to avoid using a Navier - Stokes solver for each individual evaluation in the GA run. As it has been shown, the metamodel, because of its intrinsic nature of approximation model, has an estimation or prediction error (in the best case around 9%). The implication of this error is that the optimization process could lead to the incorrect optimal solution being found. However, the solution should still be within 9% of the true optimal, assuming that 9% is the maximal ANN error. Therefore, the optimal geometry obtained might not be the global optimal, although very close to it.

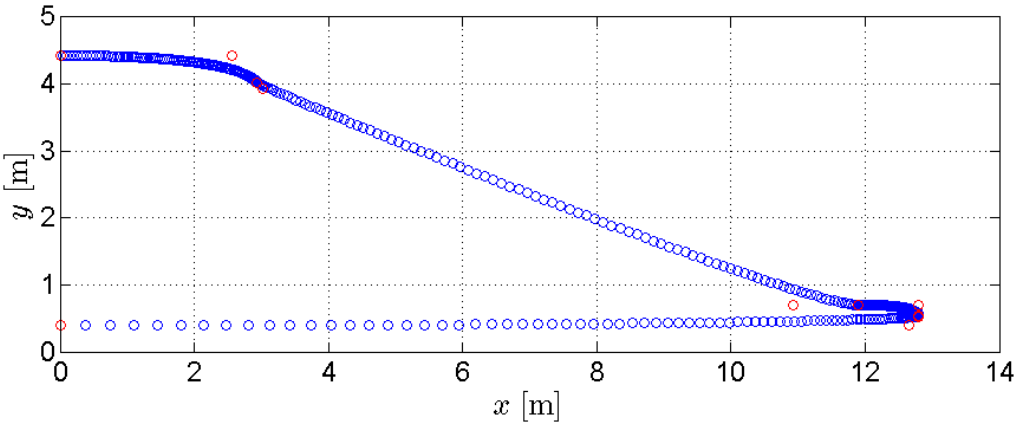


Figure 6: Optimal geometry obtained with the GA.

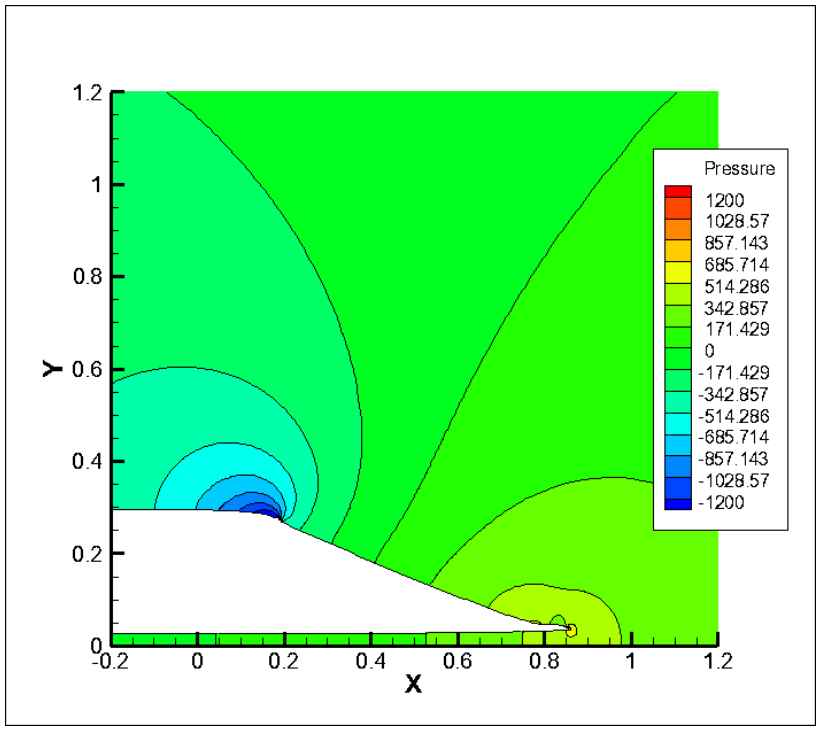


Figure 7: Pressure field around the nose for the optimal geometry.

5 CONCLUSSIONS

In this paper a new optimization approach is proposed. Using a genetic algorithm as an alternative optimization tool has been tested and good results have been obtained. To speed it up a metamodel has been used satisfactorily, avoiding in this way to use a time consuming Navier−Stokes solver. Among all the different techniques available, a comparative study has been performed, considering the simplest response surface model, and a more precise neural network method. Both techniques have been developed, and interesting results have been achieved. However, there are still many other posibilities to analyze. In particular an extensive study in order to optimize the ANN performance is proposed, including testing two-hidden-layer architectures. A more complete study about reducing the database size is also proposed. Other metamodels, like Kriging models, could also be tested. Only one objective has been considered in this paper, the reduction of drag coefficient. This explains that the geometry obtained is so simple. Thanks to the geometry parametrization proposed here a more complete study could be carried out, and a multiobjective optimization study becomes of maximum interest. It is expected that quite different optimal geometries to the one here reached could appear when minimizing pressure waves inside a tunnel or optimizing crosswind stability are also pursued. Finally, a transformation from 2-D to a 3-D model is the final objective of this research project.

6 ACKNOWLEDGMENTS

This work is financed by Ministerio de Ciencia e Investigación (Eng. Ministry of Science and Technology) under contract TRA-2010-20582, included in the VI Plan Nacional de I+D+i 2008-2011. It is also part of the research project included in Subprograma INNPACTO, from Ministerio de Ciencia e Innovación.

REFERENCES

[1] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, *Addison Wesley* (1989)

[2] K. KrishnaKumar, Genetic Algorithms, an Introduction and Overview of their Capabilities, In proceedings of the *AIAA Guidance, Navigation and Control Conference*, South Carolina, USA, AIAA-92-4462-CP, 728–738 (1992)

[3] S. Krajnovic, Shape Optimization of High-Speed Trains for Improved Aerodynamic Performance, *Proceedings of the Institution of Mechanical Engineers. Part F: Journal of Rail and Rapid Transit*, **223**, 439–452 (2009)

[4] V. V. Vytla, P. G. Huang and R. C. Penmetsa, Response Surface Based Aerodynamic Shape Optimization of High Speed Train Nose, In proceedings of the *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Florida, USA, Paper n° AIAA 2010-1509 (2010)

[5] T. Verstraete, Multidisciplinary Turbomachinery Component Optimization Considering Performance, Stress and Internal Heat Transfer, *Ph.D Thesis*, Universiteit Gent, Belgium (2008)

[6] T. L. Holst and T. H. Pulliam, Aerodynamic Shape Optimization using a Real-number-encoded Genetic Algorithm, In proceedings of the *19th AIAA Applied Aerodynamics Conference*, California, USA, Paper n° A01-31030 (2001)

[7] M. Suzuki, M. Ikeda and K. Yoshida, Study on Numerical Optimization of Cross-sectional Panhead Shape for High-Speed Train, *Journal of Mechanical Systems for Transportation and Logistics*, **1** 100–110 (2008)

[8] J. J. Grefenstette, Optimization of Control Parameters for Genetic Algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, **16**, 122–128 (1986)

[9] Genetic Algorithm and Direct Search Toolbox^{TM} 2, User's Guide, MATLAB (2008)

[10] H.-B. Kwon, K.-H. Jang, Y.-S. Kim, K.-Y. Yee and D.-H. Lee, Nose Shape Optimization of High-Speed Train for Minimization of Tunnel Sonic Boom, *JSME International Journal Series C*, **44**, 890–899 (2001)

[11] J. Lee and J. Kim, Approximate Optimization of High-Speed Train Nose Shape for Reducing Micropressure Wave, *Structural Multidiscipline Optimization*, **35**, 79–87 (2008)

[12] E. Guilmineau and F. Chometon, Effect of Side Wind on a Simplified Car Model. Experimental and Numerical Analysis, *Journal of Fluids Engineering*, **131** (2009)

[13] T. W. Chiu and L. C. Squire, An Experimental Study of the Flow over a Train in a Crosswind at large Yaw Angles up to 90°, *Journal of Wind Engineering and Industrial Aerodynamics*, **45**, 47–74 (1992)

[14] G. Farin, Curves and Surfaces for Computer-Aided Geometric Design. A Practical Guide, *Academic Press Professional*, 3rd ed. (1993)

[15] F. Cheli, F. Ripamonti, D. Rocchi and G. Tomasini, Aerodynamic behaviour investigation of the new EMUV250 train to cross wind, *Journal of Wind Engineering and Industrial Aerodynamics*, **98**, 189–201 (2010)

[16] R. S. Raghunathan, H.-D. Kim and T. Setoguchi, Aerodynamics of High-Speed Railway Train, *Progress in Aerospace Sciences*, **38**, 469–514 (2002)

[17] J. A. Schetz, Aerodynamics of High-Speed Trains, *Annual Review of Fluid Mechanics*, **33**, 371–414 (2001)

[18] J. B. Doyle, R. J. Hartfield and C. Roy, Aerodynamic optimization for freight trucks using a genetic algorithm and CFD, In proceedings of the *48th AIAA Aerospace Sciences Meeting and Exhibit*, Nevada, USA, Paper n° AIAA 2008-323 (2008)

[19] A. A. Giunta, S. F. Wojtkiewicz Jr. and M. S. Eldred, Overview of modern design of experiments methods for computational simulations, In proceedings of the *41st AIAA Meeting*, Nevada, USA, Paper n° AIAA 2003-649 (2003)

[20] R. Jin, W. Chen and T. W. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, *Journal of Structure and Multidisciplinary Optimization*, **23**, 1–13 (2001)

[21] T. W. Simpson and J. D. Peplinski, Metamodels for computer-based engineering design. Survey and recommendations, *Engineering with Computers*,**17**, 129–150 (2001)

[22] R. H. Myers and A. I. Khuri and W. H. C. Jr., Response Surface Methodology. 1966−1988, *Technometrics* **31** 137–157 (1989)