A FINITE ELEMENT NUMERICAL ALGORITHM FOR MODELLING AND DATA FITTING IN COMPLEX SYSTEMS

F.J. NAVARRO-GONZÁLEZ & Y. VILLACAMPA
Department Applied Mathematics, Alicante University. Apartado 99, E-03080. Alicante. Spain

ABSTRACT

Numerical modelling methodologies are important by their application to engineering and scientific problems, because there are processes where analytical mathematical expressions cannot be obtained to model them. When the only available information is a set of experimental values for the variables that determine the state of the system, the modelling problem is equivalent to determining the hyper-surface that best fits the data.
This paper presents a methodology based on the Galerkin formulation of the finite elements method to obtain representations of relationships that are defined a priori, between a set of variables: $y = z(x^1, x^2,...., x^d)$. These representations are generated from the values of the variables in the experimental data. The approximation, piecewise, is an element of a Sobolev space and has derivatives defined in a general sense into this space. The using of this approach results in the need of inverting a linear system with a structure that allows a fast solver algorithm. The algorithm can be used in a variety of fields, being a multidisciplinary tool.
The validity of the methodology is studied considering two real applications: a problem in hydrodynamics and a problem of engineering related to fluids, heat and transport in an energy generation plant. Also a test of the predictive capacity of the methodology is performed using a cross-validation method.
Keywords: complex systems, fast algorithm, finite elements, galerkin, modelling.

1 INTRODUCTION

Given a system and a set of variables $\left(x^1,x^2,....,x^d,y\right)$ that determine its state, the modelling problem is usually to obtain, in anyway, the unknown relation given by $y = z(x)$. The form of this dependence must be fixed using a set of experimental data $\left\{\left(x^1_{[k]},x^2_{[k]},......,x^d_{[k]},y_{[k]}\right)\right\}_{k=1,2,....,p}$. This is why the problem is also called the data fitting problem.

Sometimes the forms of these relations are known or at least can be supposed. The linear regression is a well-studied case in which the variables obey the equation:

$$y = a + \beta_1 \cdot x^1 + ... + \beta_d \cdot x^d \tag{1}$$

However, the linear relations are a good tool for processes that are near the system equilibrium points, but usually they are not useful far from there. In natural problems, the number of non-linear and chaotic processes is greater that the linear ones.

If the relation is not linear, but it is known, the problem consists also in the calculation of some unknown parameters $(\lambda_1,..\lambda_Q)$ of the expression using the data.

$$y = z\left(x^1,...,x^d,\lambda_1,..\lambda_Q\right) \tag{2}$$

Determining the values of the parameters of the model can be done minimizing an error function defined over the experimental points. The usual method is to consider the mean square error (MSE) of the estimated values on the experimental points.

ISSN: 2046-0546 (paper format), ISSN: 2046-0554 (online), http://www.witpress.com/journals
DOI: 10.2495/CMEM-V4-N2-100-113

$$e\left(\lambda_1,..\lambda_Q\right)=\sum_{k=1}^{P}\left(y_{[k]}-z\left(x_{[k]}^1,\dots,x_{[k]}^d,\lambda_1,..\lambda_Q\right)\right)^2 \tag{3}$$

In the general case, when neither the relation nor the number of parameters is known, the methodologies can be oriented to obtain an analytical expression or a numerical approximation to the relation (2).

Data fitting is related to the problem of interpolating a set of points, but there is a key difference between them: in the data fitting problem, the error on the experimental points cannot be assumed as null, because the existence of an experimental error:

$$y_{[k]}=z\left(x_{[k]}\right)+\varepsilon_{[k]} \tag{4}$$

However, methods related to the multivariate interpolation problem (radial basis function, full and sparse grid interpolations, and multivariate splines) are sometimes used in the n-dimensional numerical data fitting problem. Also, other techniques, as neural networks, radial basis function nets, kernel regression, and so on are widely used. Other methodologies have been presented in [1] and [2] to model (2) thought mathematical equations.

The finite element method (FEM) is a well-known method with a wide amount of applications as solid mechanics, fluid flow, electricity and magnetism, heat transfer and other problems. In engineering, is used for the design of structures, the study of materials, and so on.

The applications mentioned above use the FEM to solve approximately the differential equations that govern the system behaviour. The method transforms the differential equation in a system of linear equations, whose unknown variables are the values of the function in a defined set of points called nodes.

The use of FEM in the methodology presented in this paper gives as a result of the data fitting problem an element of a Sobolev space, that is, a piecewise function with derivatives defined in a general way. It is an evolution of those presented in [3–7]. There is no need for additional conditions or constraints for the regression hyper-surface, resulting in a substantial improvement in the applicability and the algorithmic efficiency.

The definition of a slightly different error function, defined over the entire function domain, allows the use of the Galerkin weighted residual method, resulting in a Kronecker-product system matrix with an efficient inversion algorithm.

2 THE FINITE ELEMENT METHOD

The FEM is widely used in engineering problems. Given a differential equation defined by a differential operator *D*:

$$D\left(f\right)=v \tag{5}$$

Where $f,v\in V$ and V is a function space. The FEM replaces V by a finite dimensional subspace $V_h\subset V$ which is composed by continuous piecewise polynomial functions of degree K, associated with a division of the domain Ω where the problem is defined in N_e parts called elements.

$$\Omega=\bigcup_{i=1}^{Ne} \tag{6}$$

The problem to solve is now:

$$D\left(f_h\right) = v_h \text{ where } f_h, v_h \in V_h \tag{7}$$

Considering a basis for the functions of V_h and supposing $\dim V_h = N$:

$$V_h = \left\langle \varphi_1(x), \varphi_2(x), ..., \varphi_N(x) \right\rangle \tag{8}$$

The approximate solution takes the form:

$$f_h(x) = \sum_{i=1}^{N} u_i \cdot \varphi_i(x) \tag{9}$$

The values of u_i must be determined to get a good approximation for the solution to the differential equation.

The basis is selected considering a set of N points called nodes where:

$$\left(\varsigma_1, \varsigma_2, ..., \varsigma_N\right) / \varphi_i\left(\varsigma_j\right) = \delta_{ij}$$

The functions of the basis are called shape functions, and they are used to interpolate in points different from the nodes. The approximate function is part of a Sobolev space.

3 GALERKIN'S WEIGHTED RESIDUAL METHOD

Weighted Residual Methods (WRMs) are used in the solution of Partial Differential Equations, looking for a function that is the solution of the problem [8–10]. Given the eqn (3), an error or residual function can be defined as the difference between the solution and its approximation,

$$e(x) = f(x) - f_h(x) = f(x) - \sum_{i=1}^{N} u_i \varphi_i(x) \tag{10}$$

WRMs represent a set of methods where an integral of this error is minimized in a way that characterizes each method, depending on the selected weight function (collocation method, sub-domain method, Least Square Method, Galerkin method, method of moments):

$$\int_D e(x) \cdot W_j(x)\, dx = 0, \quad i = 1, 2, ..., N \tag{11}$$

Where the number of weight functions $W_j(x)$ is equal to the number of unknown u_i constants.

In the Galerkin's method, the N unknown parameters are obtained by selecting as weight functions the same functions used in the approximation of f, $W_j(x) = \varphi_j(x)$.

3.1 Use of finite elements in numerical data fitting. Direct and Lagrangian formulation

Given an unknown relation $y = z(x)$, defined on a domain Ω and a sample of points obtained from it: $\left\{\left(x^1_{[k]}, ..., x^d_{[k]}, y_{[k]}\right)\right\}_{k=1..P}$

The problem is to approximate the relation to be able to estimate its value in any point of the domain. When the form of the relation is supposed, the problem can be solved for determining the values of the parameters included in the expression (usually minimizing a sum of squared errors).

If no assumptions on the relation expression can be done, one can consider using general techniques as neural networks, radial basis functions, etc.

In [5–7], a methodology that applies the FEM to obtain numerical models for an unknown relation between the variables was presented by the authors, defining an error function over each possible solution $z_h(x) \in V_h$ (representation) for the desired relation. Using as variables, the values of the objective function in the nodes of a partition of the domain where the function is defined, the problem considered is:

$$e : \Upsilon \to \Re :$$
$$e\left(u_1, \ldots., u_N\right) = \sum_{k=1}^{P} \left[z_h\left(x_{[k]}; u_1, \ldots., u_N \right) - z_{[k]} \right]^2 \qquad (12)$$

This function can be minimized, considering the derivative:

$$\frac{\partial e}{\partial u_r} = -2 \cdot \sum_{k=1}^{P} \left[z_h\left(x_{[k]}; u_1, \ldots., u_N\right) - z_{[k]} \right] \cdot \frac{\partial z_h}{\partial u_r} = -2 \cdot \sum_{k=1}^{P} \left[\sum_{j=1}^{N} u_j \cdot \varphi_j\left(x_{[k]}\right) - z_{[k]} \right] \cdot \varphi_r\left(x_{[k]}\right) \qquad (13)$$

Reordering terms, and imposing the character of extreme:

$$\sum_{j=1}^{N} \left\{ \sum_{k=1}^{P} \varphi_r\left(x_{[k]}\right) \cdot \varphi_j\left(x_{[k]}\right) \right\} \cdot u_j = \sum_{k=1}^{P} z_{[k]} \cdot \varphi_r\left(x_{[k]}\right) \qquad (14)$$

This is a linear system, and can be solved for the variables $\left(u_1, u_2, \ldots., u_N\right)$. But in most cases, the system is under-determined, so it is necessary to add some extra equations to obtain a unique solution.

There are different options to obtain an invertible matrix, but they have one point in common. The original matrix is transformed using a regularization term (or a rigidization term in the terminology used by the authors in previous papers by physical considerations). In the paper [6] and [7], the function to minimize is:

$$E : \Upsilon \to \Re :$$
$$E\left(u_1, \ldots., u_N\right) = \sum_{k=1}^{P} \left[z_h\left(x_{[k]}; u_1, \ldots., u_N\right) - z_{[k]} \right]^2 + \lambda \cdot \left| D^2 f_h \right|^2 \qquad (15)$$

And the system takes the form:

$$\sum_{j=1}^{N} \left\{ \sum_{k=1}^{P} \varphi_r\left(x_{[k]}\right) \cdot \varphi_j\left(x_{[k]}\right) + \lambda \cdot R_{rj} \right\} \cdot u_j = \sum_{k=1}^{P} z_{[k]} \cdot \varphi_r\left(x_{[k]}\right) \qquad (16)$$

If the data are normalized, the problem can be studied on the domain $\Omega = [0,1]^d$. Using linear squared elements with volume $h^d = (1/c)^d$, where the parameter c is called the complexity, the system has a symmetric matrix with size $(c+1)^d \times (c+1)^d$.

Different algorithms can be used to obtain the solution of the system, from LU decomposition to techniques based on Krylov subspaces, with limits for the computational cost $t(c,d)$ given for the mathematical operations on the matrix elements of:

$$o\left[(c+1)^{2\cdot d}\right] \le t(c,d) \le o\left[(c+1)^{3\cdot d}\right] \tag{17}$$

These values depend on the characteristics of the matrix (symmetry, sparseness, positive definiteness).

4 DATA FITTING METHOD USING THE GALERKIN METHOD

Let $y = z(x)$ an unknown relation defined on a domain Ω and a sample of points obtained from it: $\left\{\left(x^1_{[k]},\ldots,x^d_{[k]},y_{[k]}\right)\right\}_{k=1..P}$

Given a discretization of the domain with size $h = 1/c$, the approximate solution has the form

$$z_h(x) = \sum_{i=1}^{N} u_i \cdot \varphi_i(x) \tag{18}$$

An error function can be defined as:

$$e(x) = z(x) - z_h(x) = z(x) - \sum_{i=1}^{N} u_i \cdot \varphi_i(x) \tag{19}$$

Using the weighted residual method, optimum values of u can be calculated:

$$\int_{[0,1]^d}\left(z(x) - \sum_{i=1}^{N} u_i \cdot \varphi_i(x)\right)\cdot W_j(x)\,d^d x = 0 \qquad j = 0..N \tag{20}$$

At this point, an approximation for the $z(x)$ function is needed to calculate the integrals (11). But this is the goal of the process, so a shortcut would be useful. The function $z(x)$ can be roughly estimated using a radial base function-like approach, to obtain a piecewise non-continuous set of constant values for each element of the discretization. The Galerkin's conditions will account of this discontinuity obtaining a smooth hyper-surface for $z_h(x)$:

$$z(x) \approx z_{\{E\}}(x) = \left\{z_{\{E\}} \quad where \quad x \in \Omega_E\right\} \tag{21}$$

A radial basis function is a function that only depends on a distance to a point $\psi\left(\left|x-\eta_E\right|\right)$. Sums of radial basis functions are used to approximate unknown functions from a set of values. They can be seen as a simple neural network approximation. They are used frequently as kernels in support vector machines and probability density estimation algorithms.

Where the estimated values are calculated as:

$$z_{\{E\}} = \sum_{r=1}^{P} y_{[r]} \cdot \psi\left(\left|x_{[r]} - \eta_E\right|\right) \tag{22}$$

Being the centre of the e-th element.

Introducing the multi-indexes

$$\left(I \to \left(i_1,\dots,i_d\right), J \to \left(j_1,\dots,j_d\right) \quad j_r \in \{0,1\dots,c\} \quad E \to \left(e_1,\dots,e_d\right) \quad e_r \in \{0,1\dots,c-1\}\right)$$

For the elements and the nodes of the discretization:

$$z(x) \approx z_{\{E\}}(x) = \left\{ z_{\{e_1,\dots,e_d\}} \quad where \quad x \in \Omega_{e_1,\dots,e_d} \right\} \tag{23}$$

These multi-indexes are just a convenient form of numeration for the different nodes and elements. For example, let consider the nodes. In dimension one, the nodes are indexed by their global index from 0 to c. In the two-dimensional case, the nodes have indexes from 0 to $(c+1)^2 - 1$, but they are distributed on a squared net, so considering each position separately, it is possible to give a pair of coordinates (i_1, i_2) to determinate each one, where $i_k \in [0,c]$. This pair (i_1, i_2) are the multi-indexes corresponding to the considered node. A similar reasoning can be applied to define the element multi-index numeration.

So, given that the finite elements, we are considering squared and homogeneous, the structure can be viewed as the product of unidimensional equally spaced intervals, as shown in Fig. 1

The combination of the Galerkin's WRM and squared elements gives a specially appropriated form to the resulting system:

$$\int_{[0,1]^d} \left(z_{\{E(x)\}} - \sum_{i_1,\dots,i_d} u_{i_1,\dots,i_d} \cdot \varphi_{i_1,\dots,i_d}(x) \right) \cdot \varphi_{j_1,\dots,j_d}(x) d^d x = 0 \tag{24}$$

The shape functions in the case of linear elements can be decomposed in the product of one-dimensional shape functions as:

$$\varphi_{j_1,\dots,j_d}\left(x^1,\dots,x^d\right) = \varphi_{j_1}^{[1]}\left(x^1\right) \cdot \varphi_{j_2}^{[1]}\left(x^2\right) \cdots \varphi_{j_d}^{[1]}\left(x^d\right) \tag{25}$$

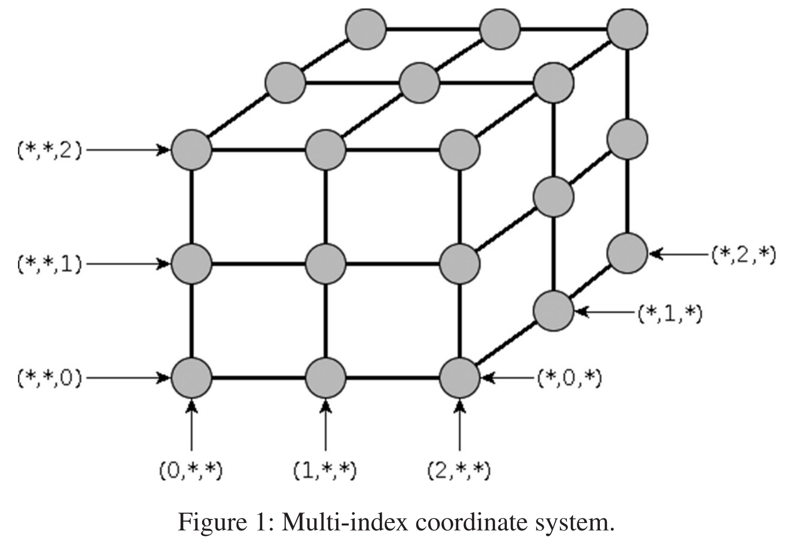

Figure 1: Multi-index coordinate system.

The expression of each one is:

$$\varphi_{i_r}^{[1]}\left(x^r\right)=\begin{cases}\dfrac{x^r-x_{i-1}^r}{h}=1+\dfrac{x^r-x_i^r}{h} & x_{i-1}^r\le x^r<x_i^r\\ \dfrac{x_{i+1}^r-x^r}{h}=1-\dfrac{x^r-x_i^r}{h} & x_i^r\le x^r<x_{i+1}^r\\ 0 \quad \text{otherwise,} & \left|x^r-x_i^r\right|\ge h\end{cases} \tag{26}$$

So, the weighted errors are:

$$\int_{[0,1]^d} z_{\{E(x)\}}\cdot\varphi_{j_1,\dots,j_d}(x)d^dx=\int_{[0,1]^d}\sum_{i_1,\dots,i_d}u_{i_1,\dots,i_d}\cdot\varphi_{i_1,\dots,i_d}(x)\cdot\varphi_{j_1,\dots,j_d}(x)d^dx \tag{27}$$

For the left term:

$$\int_{[0,1]^d} z_{\{E(x)\}}\cdot\varphi_{j_1,\dots,j_d}(x)d^dx=\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)}\int_{\Omega_{e_1,\dots e_d}} z_{[e_1,\dots e_d]}\cdot\varphi_{j_1}^{[1]}\left(x^1\right)\cdot\varphi_{j_2}^{[1]}\left(x^2\right)\cdots\varphi_{j_d}^{[1]}\left(x^d\right)d^dx=$$
$$\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)}\int_{\Omega_{e_1,\dots e_d}} z_{[e_1,\dots e_d]}\cdot\varphi_{j_1}^{[1]}\left(x^1\right)\cdot\varphi_{j_2}^{[1]}\left(x^2\right)\cdots\varphi_{j_d}^{[1]}\left(x^d\right)d^dx=$$
$$\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)} z_{[e_1,\dots e_d]}\cdot\int_{\Omega_{e_1,\dots e_d}}\varphi_{j_1}^{[1]}\left(x^1\right)\cdot\varphi_{j_2}^{[1]}\left(x^2\right)\cdots\varphi_{j_d}^{[1]}\left(x^d\right)d^dx=$$
$$\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)} z_{[e_1,\dots e_d]}\cdot\int_{\left[x_{j_1-1}^1,x_{j_1}^1\right]}\varphi_{j_1}^{[1]}\left(x^1\right)dx^1\cdots\cdot\int_{\left[x_{j_d-1}^d,x_{j_d}^d\right]}\varphi_{j_d}^{[d]}\left(x^d\right)dx^d \tag{28}$$

Where $\Omega_{e_1,\dots,e_d}$ are the elements adjacent to the node $\left(j_1,\dots,j_d\right)$.
Calculating the integrals the left term is:

$$\int_{[0,1]^d} z(x)\cdot\varphi_{j_1,\dots,j_d}(x)d^dx=\left(\frac{h}{2}\right)^d\cdot\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)} z_{[e_1,\dots e_d]}\cdot\varepsilon_{j_1}\cdots\cdot\varepsilon_{j_d} \tag{29}$$

Where $\varepsilon_{j_i}^k=\delta_{j_i}^k$

The other side of the equation is:

$$\int_{[0,1]^d}\sum_{i_1,\dots,i_d}u_{i_1,\dots,i_d}\cdot\varphi_{i_1,\dots,i_d}(x)\cdot\varphi_{j_1,\dots,j_d}(x)d^dx$$
$$=\int_{[0,1]^d}\sum_{i_1,\dots,i_d}u_{i_1,\dots,i_d}\cdot\varphi_{i_1}^{[1]}\left(x^1\right)\cdot\varphi_{i_2}^{[1]}\left(x^2\right)\cdots\varphi_{i_d}^{[1]}\left(x^d\right)\cdot\varphi_{j_1}^{[1]}\left(x^1\right)\cdot\varphi_{j_2}^{[1]}\left(x^2\right)\cdots\varphi_{j_d}^{[1]}\left(x^d\right)d^dx=$$
$$\sum_{i_1,\dots,i_d}u_{i_1,\dots,i_d}\cdot\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)}\int_{\Omega_{e_1,\dots e_d}}\varphi_{i_1}^{[1]}\left(x^1\right)\cdot\varphi_{i_2}^{[1]}\left(x^2\right)\cdots\varphi_{i_d}^{[1]}\left(x^d\right)\cdot\varphi_{j_1}^{[1]}\left(x^1\right)\cdot\varphi_{j_2}^{[1]}\left(x^2\right)\cdots\varphi_{j_d}^{[1]}\left(x^d\right)d^dx=$$
$$\sum_{i_1,\dots,i_d}u_{i_1,\dots,i_d}\cdot\sum_{\Omega_{e_1,\dots e_d}\in adj(j_1,\dots,j_d)}\int_{\left[x_{j_1-1}^1,x_{j_1}^1\right]}\varphi_{i_1}^{[1]}\left(x^1\right)\cdot\varphi_{j_1}^{[1]}\left(x^1\right)dx^1\cdots\cdot\int_{\left[x_{j_d-1}^d,x_{j_d}^d\right]}\varphi_{i_d}^{[1]}\left(x^d\right)\cdot\varphi_{j_d}^{[D]}\left(x^d\right)dx^d=$$
$$\left(\frac{h}{6}\right)^d\cdot\sum_{i_1,\dots,i_d}u_{i_1,\dots,i_d}\cdot M_{j_1}^{i_1}\cdot M_{j_2}^{i_2}\cdots\cdot M_{j_d}^{i_d} \tag{30}$$

Writing the system using matrix:

$$M \otimes ...... \otimes M \cdot u = 3^d \cdot \sum_{E \in adj} Z_{\{E\}} \cdot \left(\varepsilon \otimes ...... \otimes \varepsilon\right)^{\{E\}} \quad \rightarrow$$

$$u = 3^d \cdot \sum_{E \in adj} Z_{\{E\}} \cdot \left(\left[M^{-1} \cdot \varepsilon\right] \otimes ...... \otimes \left[M^{-1} \cdot \varepsilon\right]\right)^{\{E\}}$$

where M is the tridiagonal matrix:

$$M = \begin{pmatrix} 2 & 1 & 0 & ... & 0 & 0 & 0 \\ 1 & 4 & 1 & ... & 0 & 0 & 0 \\ 0 & 1 & 4 & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 4 & 1 & 0 \\ 0 & 0 & 0 & ... & 1 & 4 & 1 \\ 0 & 0 & 0 & ... & 0 & 1 & 2 \end{pmatrix}$$

Higher order polynomials could be used to improve the approximation of the method, but they would imply a lost in the characteristics of the algorithm by two causes. First, the linear form allows to construct the global system matrix as the Kronecker product of identical matrices that has a specially convenient form. Also the ‘one-dimensional-related’ tridiagonal matrix has a simple expression that would vary in the case of using higher order approximation functions.

5 RESULTS

Three examples are used to test the behaviour of the methodology. The first involves a six-dimensionalproblem from fluid dynamics, while the second is a test of performance for a four-dimensional problem with a great number of experimental points. In the last example, a random subset of training and test points are selected from the second four-dimensional problem dataset to test the power of prediction of the algorithm.

5.1 Yacht hydrodynamics data

The main goal of the methodology is to extend the applicability to the problems of dimensionality greater than 4 that was a practical limit for the previous methodologies [3,5–7] given the high time of computing.

For the first example, data from the UCI Machine Learning Repository have been selected [1] (Yacht Hydrodynamics data set, which have dimension six). The objective variable is the residuary resistance of sailing yachts considering as inputs the basic hull dimensions and the boat velocity. The Delft data set comprises 308 full-scale experiments, which were performed at the Delft Ship Hydromechanics Laboratory for that purpose. Previous papers have studied this datasets [11,12].

These experiments include 22 different hull forms, derived from a parent form closely related to the Standfast 43 designed by Frans Maas.

The considered variables are:

- Longitudinal position of the centre of buoyancy, adimensional
- Prismatic coefficient, adimensional
- Length-displacement ratio, adimensional
- Beam-draught ratio, adimensional
- Length-beam ratio, adimensional
- Froude number, adimensional

And the measured variable is the residuary resistance per unit weight of displacement, and the residuary resistance per unit weight of displacement is adimensional.

The complexity used for the calculus is 25. This gives a number of nodes for the problem near of 300 millions. The program has been run on a computer with 2 GHz dual core. The elapsed time has been of 15 hours approximately (Fig. 2).

The coefficient of determination of the model is R2 = 0.992574.

This can be seen again in other form, using Fig. 3.

The REC curve of the model is represented in Fig. 4.

The order of the algorithm can be seen in Fig. 5, where the variable log(time) is represented over log(complexity). A linear regression gives a relation $time \approx complexity^{5.81}$

5.2 Combined cycle power plant data set

To test the methodology with a high number of experimental points, the 'Combined Cycle Power Plant' (CCPP) data from the UCI Machine Learning Repository is studied. The dataset is defined in [13]:

"The dataset contains 9568 data points collected from a CCPP over 6 years (2006–2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant. A CCPP is

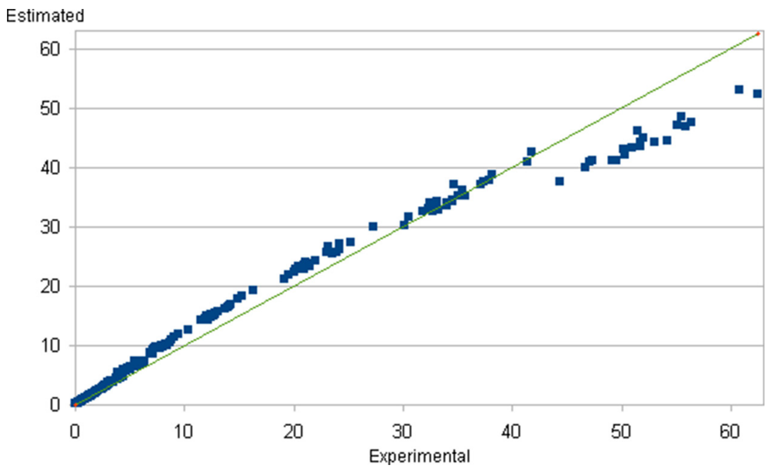

Figure 2: Estimated versus experimental values.

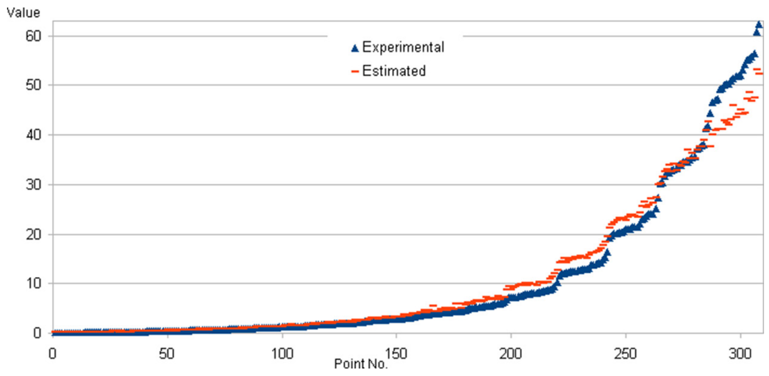

Figure 3: Comparison of values between points (value ordered).

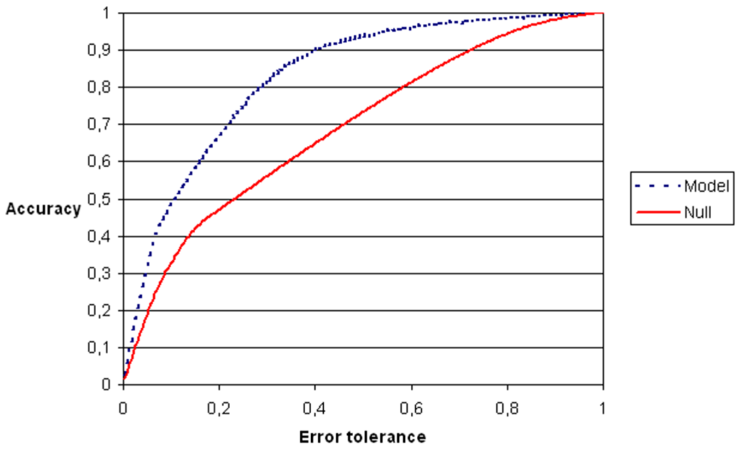

Figure 4: REC curve for the yacht hydrodynamics’ model.

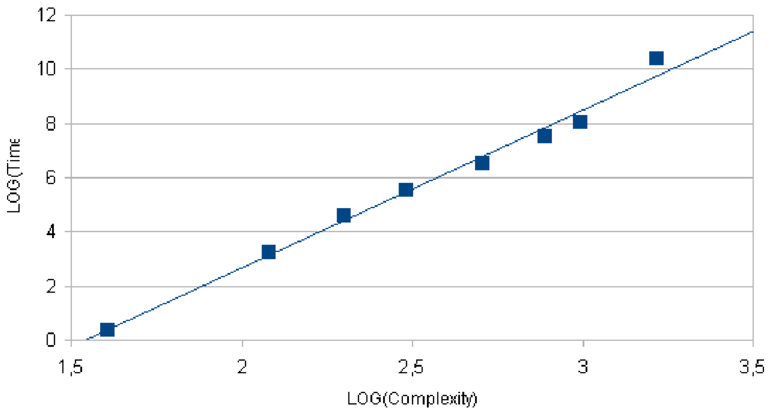

Figure 5: Time for computation.

composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity is generated by gas and ST, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum is collected from and has effect on the Steam Turbine, the other three of the ambient variables collected the GT performance".

The model has been calculated using a complexity of 30. The coefficient R2 of the model is R2 = 0.974456, and the computing time has been 18 min 44 s.

This can be seen again in other form, using Fig. 7.

The REC curve of the model is represented in Fig. 8.

These results can be compared with those obtained in previous studies [13].

The time for the computation follows the equation $time \approx complexity^{3.6}$, as can be seen from Fig. 9.

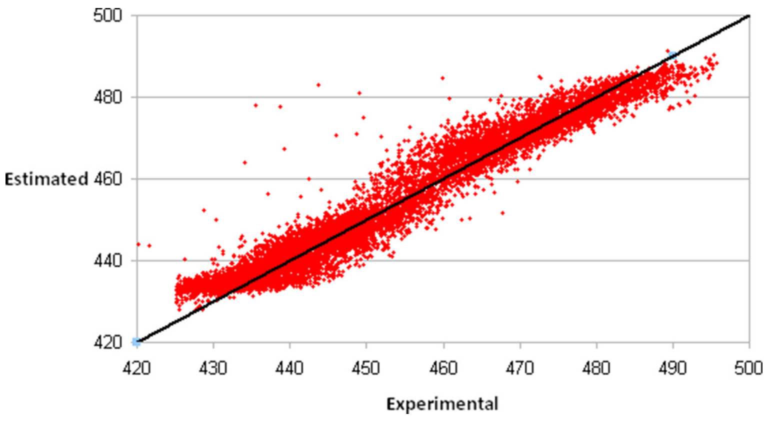

Figure 6: Estimated versus experimental values.

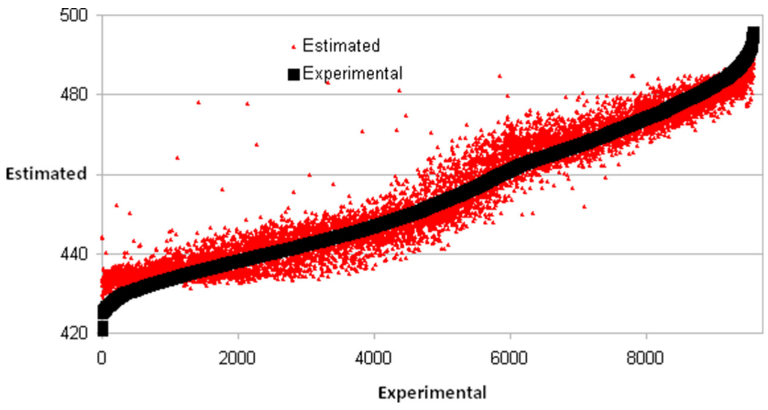

Figure 7: Comparison of values between points (value ordered).

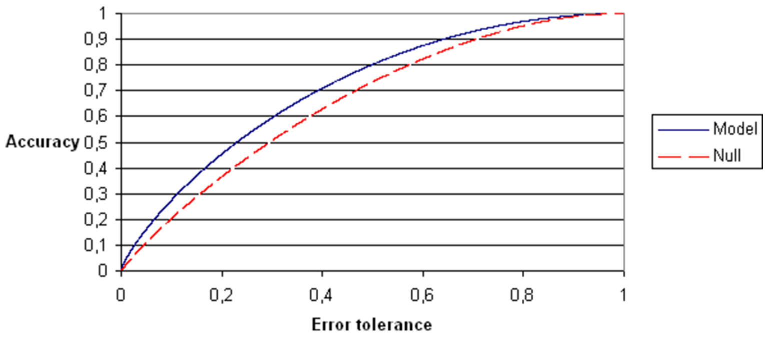

Figure 8: REC curve for the Combined Cycle Power Plant data.

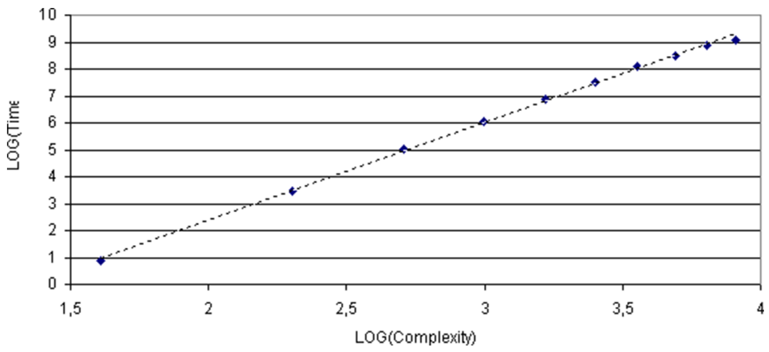

Figure 9: Time for computation.

5.3 Testing the results: yacht hydrodynamics

A random set of 270 points are extracted for the ‘Yacht Hydrodynamics’ dataset. The other 38 points are used as test set. The results obtained by the methodology can be shown in Figs. 10 and 11.

6 CONCLUSIONS

The problem of modelling a complex system using analytical or numerical methodologies is widely present in natural and social sciences, and engineering.

In some cases, linear and non-linear regression techniques are used. However, doing this is equivalent to the knowledge of the kind of the relation between the variables. In the general case, this relation is unknown and no assumptions can be done. For problems with several variables, other techniques as neural networks are frequently used. Neural networks results are dependent on the network parameters: number of hidden layers and neurons, epoch, learning rate, and so on, and the results have not always an easy interpretation.

The methodology presented in this paper continues the previous research line of the authors, directed to study numerical modelling techniques and improves the computational efficiency of the available algorithms.

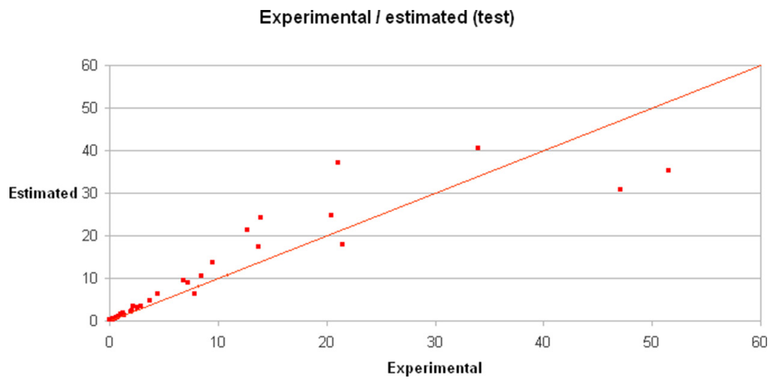

Figure 10: Experimental- Estimates for test points.

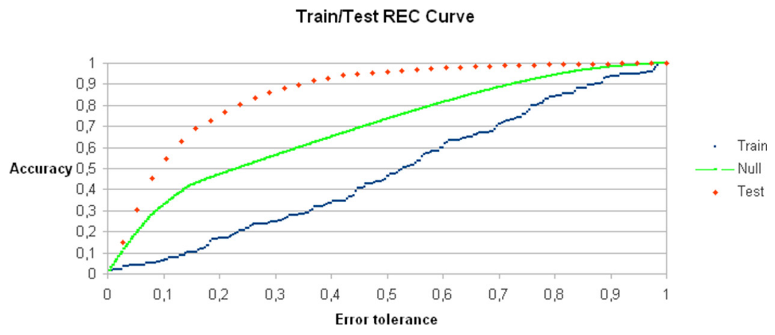

Figure 11: Train and test REC curve.

The use of the FEM allows obtaining numerical models with a clear mathematical meaning. The existence of a definition of derivatives in a general (weak) form can also be considered as a positive factor. The present formulation of the problem using a Galerkin approach improves the computational complexity ($O\left([c+1]^{d}\right)$, compared with the previous methodologies $O\left([n+1]^{3\cdot d}\right)$).

Also, the new approach allows an easy implementation of a parallelized version of the algorithm to obtain faster and further results in complexity and dimensionality. The development of these parallel versions of the algorithm is the main investigation line in future studies.

Other investigation lines would be the study of the approximation error behaviour depending on c and d, and the introduction of cross validation techniques to determine the most convenient complexity.

REFERENCES

[1] Bache, K. & Lichman, M., UCI machine learning repository, 2013, http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science. 2007.

[2] Verdú, F. & Villacampa, Y., A computational algorithm for the multiple generation of non-linear mathematical models and stability study. *Advances in Engineering Software*, **39**(5), pp. 430–437, 2008.
http://dx.doi.org/10.1016/j.advengsoft.2007.03.004

[3] Villacampa, Y., Navarro-González, F.J. & Llorens, J., A geometric model for the generation of models defined in Complex Systems. *Ecosystems and Sustainable Development VII*, eds C.A. Brebbia & E. Tiezzi, WIT Press: Southampton, 2009.

[4] Perez-Carrió, A., Villacampa, Y., Llorens, J. & García-Alonso, F., A computational algorithm for system modelling based on bi-dimensional finite element techniques. *Advances in Engineering Software,* **40**(1), pp. 30–40, 2009.
http://dx.doi.org/10.1016/j.advengsoft.2008.03.010

[5] Navarro-González, F.J., Modelos de Representación por Elementos Finitos n-dimensionales para Sistemas Complejos. *Tesis Doctoral*. Universidad de Alicante, 2011.

[6] Navarro-Gonzalez, F. & Villacampa, Y., A new methodology for complex systems using n-dimensional finite elements. *Advances in Engineering Software*, **48**, pp. 52–57, 2012.
http://dx.doi.org/10.1016/j.advengsoft.2012.02.001

[7] Navarro-Gonzalez, F. & Villacampa, Y., Generation of representation models for complex systems using Lagrangian functions. *Advances in Engineering Software*, **64**, pp. 33–37, 2013.
http://dx.doi.org/10.1016/j.advengsoft.2013.04.015

[8] Brenner, S.C. & Scott, L.R., *The Mathematical Theory of Finite element Methods*, Springer: Printed in the United States, New York. Inc, 2002.

[9] Gallagher, R.H., *Finite Element Analysis*. Prentice Hall Inc: New Jersey, 1978. Printed in the United States, New York. Inc.

[10] Hughes, T.J.R., Franca, L.P. & Hulbert, G.M., A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, **73**(2), pp. 173–189, 1989.
http://dx.doi.org/10.1016/0045-7825(89)90111-4

[11] Gerritsma, J., Onnink, R. & Versluis, A., Geometry, resistance and stability of the delft systematic yacht hull series. *In International Shipbuilding Progress*, **28**, pp. 276–297, 1981.

[12] Ortigosa, I., Lopez, R. & Garcia, J., A neural networks approach to residuary resistance of sailing yachts prediction. *Proceedings of the II International Conference on Computational Methods in Marine Engineering* (MARINE 2007), pp. 223–226, 2007.

[13] Kaya, H., Tüfekci, P. & Gürgen, S.F., Local and global learning methods for predicting power of a combined gas & steam turbine. *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE*, Dubai, pp. 13–18, 2012.