Parallel approach of a Galerkin-based methodology for predicting the compressive strength of the lightweight aggregate concrete

Violeta Migallón[a,*], Francisco Navarro-González[b], Jose Penadés[a], Yolanda Villacampa[b]

[a] *Departamento de Ciencia de la Computación e Inteligencia Artificial, E-03071 Universidad de Alicante, Spain*
[b] *Departamento de Matemática Aplicada, E-03071 Universidad de Alicante, Spain*

Abstract

A methodology based on the Galerkin formulation of the finite element method has been analyzed for predicting the compressive strength of the lightweight aggregate concrete using ultrasonic pulse velocity. Due to both the memory requirements and the computational cost of this technique, its parallelization becomes necessary for solving this problem. For this purpose a mixed MPI/OpenMP parallel algorithm has been designed and different approaches and data distributions analyzed. On the other hand, this Galerkin methodology has been compared with multiple linear regression models, regression trees and artificial neural networks. Based on different measures of goodness of fit, the effectiveness of the Galerkin methodology, compared with these statistical techniques for data mining, is shown.

Keywords: Galerkin, modelling, parallel algorithms, compressive strength prediction, concrete

1. Introduction

The finite element method (FEM) is a technique developed to solve differential equations in two or three dimensional domains. As other discretization

*Corresponding author
Email address: violeta@ua.es (Violeta Migallón)

methods like finite difference [1, 2], boundary element [3], finite volume [4], discrete elements [5, 6] or multigrid [7], the main idea of the finite element method is to reduce the degrees of freedom of the equation solution to a finite dimensional functional space. The continuous solution can then be reconstructed from an interpolation over the set of discrete solutions.

Let us consider the original problem $D(z) = v$, where D is a differential operator defined on a domain Ω and $z,\ v$ belong to a functional space V. Then, V is transformed to a finite space V_h with $dim V_h = N$, determined by the discretization (or mesh). That is, the new problem is reformulated as $D(z_h) = v_h$, where $z_h,\ v_h \in V_h$. Therefore, if for this functional space it is possible to select a basis $B_h = \{\varphi_1(x), \varphi_2(x), \ldots, \varphi_N(x)\}$, the approximated solution will have the following form

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

The discretization implies that the basis must accomplish a set of N conditions related to its values at the nodes $(\varsigma_1, \varsigma_2, \ldots, \varsigma_N)$, that is, $\varphi_i(\varsigma_j) = \delta_{ij}$.

From the approximation in (1), an error function can be defined as follows

$$e(x) = z(x) - \sum_{i=1}^{N} u_i \cdot \varphi_i(x). \tag{2}$$

Then, this error can be used to define a variety of "global errors" for the approximation using weight functions $W_j(x)$

$$\int_{\Omega} e(x) \cdot W_j(x) dx = 0,\ j = 1, 2, \ldots, N. \tag{3}$$

Weighted residual methods are a set of methods designed to minimize the integral of the error over the domain following different strategies such as collocation method, sub-domain method, least square method, Galerkin method or method of moments [8]. In particular, the Galerkin method uses as weight functions those that form the functional space basis: $W_j(x) = \varphi_j(x), j = 1, 2, \ldots, N$. This method is widely applied in solving differential equations, providing a powerful numerical solution to engineering problems [9, 10, 11, 12], including studies

on concrete structures to predict the onset time of corrosion of reinforcements [13] and for simulating dynamic fracture in concrete [14].

On the other hand, the extension of the Galerkin method to the problem of regression of discrete experimental data has been proposed by Navarro-González and Villacampa in a previous work [15], improving the algorithmic complexity of the methodologies developed in [16, 17]. These Galerkin-based methodologies have been successfully used in several problems related to electrical and hydrodynamic engineering [15], and a variety of problems concerning biological and coastal engineering such as the modelling of the equilibrium beach profile [18], the modelling of the depth of closure of a beach [19], and the modelling of escherichia coli concentrations in coastal waters [20]. However, only small datasets with a few number of variables were analyzed in these problems.

For explaining this Galerkin methodology, let us consider a set of experimental points that have been normalized to belong to the hypercube $[0,1]^d$ and a discretization dividing the unitary interval in c pieces of size $h = \frac{1}{c}$. Then, the number of nodes is $N = (c+1)^d$.

Taking the expression (3), from (2) it follows

$$\int_{[0,1]^d} \left(z(x) - \sum_{i=1}^{N} u_i \cdot \varphi_i(x) \right) \cdot W_j(x) d^d x = 0, \ j = 1, 2, \ldots, N. \tag{4}$$

Then, given a sample of points obtained from the unknown function $y = z(x)$, $\{(x^1_{[k]}, \ldots, x^d_{[k]}, y_{[k]})\}_{k=1,\ldots,P}$, this function can be approximated by a constant function on each element calculated using a radial function ψ, that is, a function that only depends on the distance to a centre point, denoted as η_E, [15]

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

Moreover, the application of the Galerkin weight function in (4) gives the set of equations

$$\int_{[0,1]^d} \left(z_{\{E(x)\}} - \sum_{i_1,\ldots,i_d} u_{i_1,\ldots,i_d} \cdot \varphi_{i_1,\ldots,i_d}(x) \right) \cdot \varphi_{j_1,\ldots,j_d}(x) d^d x = 0. \tag{5}$$

Now, given that the discretization is composed of hypercubic elements of equal size, the form functions can be written as the product of d one-dimensional form functions

$$\varphi_{j_1,\ldots,j_d}(x^1,\ldots,x^d) = \varphi^{[1]}_{j_1}(x^1)\cdot\varphi^{[1]}_{j_2}(x^2)\cdots\varphi^{[1]}_{j_d}(x^d),$$

with

$$\varphi^{[1]}_{j_r}(x^r) = \begin{cases} \frac{x^r - x^r_{j-1}}{h} = 1 + \frac{x^r - x^r_j}{h} & x^r_{j-1} \le x^r < x^r_j \\ \frac{x^r_{j+1} - x^r}{h} = 1 - \frac{x^r - x^r_j}{h} & x^r_j \le x^r < x^r_{j+1} \\ 0 & \text{otherwise, } |x^r - x^r_j| \ge h \end{cases}$$

Following the reasoning done in [15], by means of the calculation of the integrals in (5) it obtains

$$\int_{[0,1]^d} z_{\{E(x)\}}\cdot\varphi_{j_1,\ldots,j_d}(x)\, d^dx = \left(\frac{h}{2}\right)^d \cdot \sum_{\Omega_{e_1,\ldots,e_d}\in adj(j_1,\ldots,j_d)} z_{[e_1,\ldots,e_d]}\cdot\varepsilon_{j_1}\cdot\ldots\cdot\varepsilon_{j_d}, \tag{6}$$

where $\Omega_{e_1,\ldots,e_d}$ are the adjacent elements to the node $(j_1,\ldots,j_d)$ and

$$\varepsilon_{j_i} = \int_{\left[x^i_{j_i-1}, x^i_{j_i}\right]} \varphi^{[1]}_{j_i}\left(x^d\right) dx^d,\ i = 1,2,\ldots,d,$$

and

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

$$\left(\frac{h}{6}\right)^d \cdot \sum_{i_1,\ldots,i_d} u_{i_1,\ldots,i_d}\cdot M^{i_1}_{j_1}\cdot M^{i_2}_{j_2}\cdot\ldots\cdot M^{i_d}_{j_d}. \tag{7}$$

Then, using the tensor product of matrices, from equations (6) and (7) the system can be written as follows

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

where M is the tridiagonal matrix

$$M = \begin{bmatrix} 2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 2 \end{bmatrix}.$$

In this way, the calculation of the integrals in (5) gives a linear system which solution is

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

In this paper, we propose the use of this Galerkin methodology in the field of building materials. In particular, we are interested in predicting the compressive strength of the lightweight aggregate concrete (LWAC).

LWAC is a valuable and versatile material used in modern construction. Its behaviour depends, among others, on the properties of the aggregates used for manufacturing it.

In the past, and before using artificial aggregates, LWAC was produced using volcanic aggregates. Nowadays, the technologies to produce lightweight aggregates (LWA), using minerals like clays, shales, and slates, as well as industrial by-products like fly ash, bed ash or blast furnace slag, are developed in factories [21].

LWAC offers a wide range of technical and environmental benefits due to its low density. Its low density reduces the dead load in structures and it provides high sound absorption and good heat insulation properties. However, LWAC is susceptible to segregation as a result of the differences between densities of its components. The segregation causes an anisotropy in the concrete reducing its durability and increasing the risk of cracking.

In order to quantify the segregation of concrete, following [22], we analyze the compressive strength of segregated LWAC using ultrasonic pulse velocity. The detailed experimental procedure and specimen preparation for obtaining the experimental data are explained in [22], where artificial neural networks were used to predict the compressive strength variation in segregated LWAC using, as lightweight aggregate, expanded clay of different characteristics.

On the other hand, as we mentioned above, the Galerkin methodology proposed in this work generates a problem that has a size of $(c+1)^d$ nodes. Therefore, there are some memory problems in its application to obtain precise results in problems further than low dimensionality (as an example, a discretization of complexity $c = 50$ in a problem of dimension 6 implies the storage of $17,596,287,801$ nodes). Taking into account both the memory requirements and the computational cost of this technique, as the number of input variables or the complexity increases, the execution of the algorithm becomes infeasible in a sequential mode. Therefore, it is necessary the parallelization of that methodology for solving realistic problems like the one treated here. For this purpose a mixed MPI/OpenMP parallel algorithm has been designed.

Furthermore, this methodology has been compared with other data mining techniques such that the multiple linear regression, regression trees and artificial neural networks. All these techniques have been extensively used for predicting different properties of concrete; see e.g., [22, 23, 24, 25, 26], and the references cited therein.

In Section 2 we introduce the problem to be solved. Furthermore, we briefly review the above mentioned data mining techniques and we introduce the parameters utilized for comparing the different models. The strategies of parallelization are treated in detail in Section 3. In Section 4 we explain and analyze the numerical results obtained with each one of these techniques, showing that the proposed Galerkin methodology performs better than the other techniques for predicting the compressive strength of the LWAC. Some conclusions are included in Section 5.

2. Materials and methods

Based on the experimental dataset obtained in [22], we are interested in predicting the compressive strength of the lightweight aggregate concrete (LWAC) using ultrasonic pulse velocity. For this purpose, four different LWAC (see Table 1) were intentionally segregated following the experimental procedure explained in [22]. The LWAC groups explain the different mixtures designed according to the Fanjul method [27]. Concretely, the LWAC type depends on both the target density with which the concrete was produced (1700 kg/m^3 or 1900 kg/m^3) and the type of lightweight aggregate (LWA) used to produce the concrete. The type of LWA of expanded clay has been explained, in Table 1, by its particle density. Two types of LWA have been considered, one of them with density 482 kg/m^3 and a granulometric fraction with sizes 6/10 and another with density 1019 kg/m^3 and fraction 4/10.

LWAC type	LWA particle density	LWAC fixed density
Group 1	482	1700
Group 2	482	1900
Group 3	1019	1700
Group 4	1019	1900

Table 1: Characteristics of the LWAC groups.

Table 2 defines the variables involved in this problem. This dataset, including 640 data points, was used in [22] to study the behaviour of artificial neural networks in this context. Firstly, we have considered models with six input variables (LWAC-6V problem) for explaining the compressive strength of the LWAC. The six variables considered for this purpose have been the theoretical density of the concrete according to the Fanjul method [27], the particle density of the LWA, the laying time of the concrete, the vibration time, the dry density of the specimen obtained experimentally after 28 days, and the segregation index. Note that, the dry density variable is indirectly related to the content of

lightweight aggregate since a low or high density of the concrete would indicate a high or low percentage of lightweight aggregate, respectively. On the other hand, the index considered in this work for describing the segregation phenomenon is based on the P-wave velocities and was proposed in [22], as an alternative to the segregation index proposed by Ke [28] when complete specimen is not considered but rather several cores from the concrete specimen.

These models, with six input variables, have been especially useful to study the performance of the parallelization of the Galerkin methodology treated herein; see Section 3. On the other hand, using the seven available variables of the dataset (including also the P-wave velocity) we study the powerful of this methodology for simulating the compressive strength in segregated LWAC (LWAC-7V problem).

Variable description	min	max	mean
LWAC fixed density (kg/m^3)	1700	1900	1800
LWA particle density (kg/m^3)	482	1019	750.5
Concrete laying time (min)	15	90	48.75
Vibration time (s)	0	80	30
Experimental dry density (kg/m^3)	1069.80	2486.84	1673.35
P-wave velocity (m/s)	3044.25	5253.73	3778.89
Segregation index	0.845	1.136	1
Compressive strength (MPa)	2.99	50.72	21.55

Table 2: Variables of the experimental dataset.

In order to evaluate the use of the Galerkin methodology, different models have been designed, varying the complexity from $c = 10$ to $c = 70$. Table 3 shows the number of nodes for each problem depending on the number of input variables considered for the estimation of the model and its complexity. Therefore, for evaluating the use of this methodology to predict the LWAC compressive strength, its parallelization has been necessary; see Section 3 for more detail.

In addition, as mentioned above, the Galerkin methodology has been compared with multiple linear regression, regression trees and artificial neural networks. For constructing these models, Matlab® and IBM SPSS Statistics 25.0 have been used. In the rest of this section we review these techniques and we introduce several parameters for comparing the usefulness of these models to predict the LWAC compressive strength.

Number of input variables = 6	
Complexity	Number of nodes
10	1,771,561
20	85,766,121
30	887,503,681
50	17,596,287,801
70	128,100,283,921
Number of input variables = 7	
Complexity	Number of nodes
10	19,487,171
20	1,801,088,541
30	27,512,614,111
50	897,410,677,851

Table 3: Size of problem versus complexity.

2.1. Multiple linear regression model

Multiple linear regression (MLR) is a statistical method that has been utilized for predicting some properties of concrete, such as slump and compressive strength; see e.g., [23, 24] and the references cited therein. Multiple linear regression is used to explain the linear relationship between one continuous dependent variable and two or more independent variables. The multiple linear regression model can be written as follows

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \epsilon, \tag{8}$$

where Y is the dependent variable, $X_1, X_2, \ldots, X_k$ are the independent variables, and $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ are the regression coefficients. Moreover, ϵ is the random error which is interpreted as the unpredictable part of the dependent variable Y. The model defined in (8) is obtained from n-tuples of observations as follows

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \epsilon_i, \ i = 1, 2, \ldots, n,$$

where the residuals $\epsilon_i, \ i = 1, 2 \ldots, n$, are independent and identically distributed normal random variables with $E(\epsilon_i) = 0$ and $Var(\epsilon_i) = \sigma^2$. In order to assure the validity of a linear regression model, besides the above assumptions, the relationships between the dependent variable and each of the independent variables must be linear. Furthermore, no multicollinearity between independent variables is assumed.

The independent variables of a multiple linear regression can be continuous or categorical. In the last case, dummy variables must be created to represent the categorical variables. Specifically, to represent a categorical variable, a number of dummy variables equal to the number of categories of that variable minus one are needed. A dummy variable assigns the numbers 0 and 1 to indicate membership into mutually exclusive categories.

2.2. Decision tree models

Decision tree models allow us to predict or classify future observations based on a set of decision rules [29]. Several works analyzing their use for modelling the compressive strength of concrete can be found e.g., in [25, 26, 30]. Furthermore, decision trees have also been applied for predicting the elastic modulus of recycled aggregate concrete [31] and other civil and engineering problems such as the modelling of damage in reinforced concrete buildings [32].

Decision trees can be used to predict both qualitative and quantitative variables. When the target (output) variable is qualitative (or categorical), these trees are called classification trees. In this case, based on the input variables and the training instance set, the tree can be used to classify new instances into

the set of classes predefined by the target variable. On the other hand, when the target variable is quantitative, the trees are called regression trees [33]. The interdependence between the input variables and the output variable in a decision tree, as its name suggests, is graphically represented by means of a tree structure. A recursive process is used for building the tree, in such a way that starting from the root node representing all the current dataset, this dataset is partitioned into smaller and smaller subsets based on the considered input variables, attempting to obtain a similar or homogeneous behaviour within nodes with respect to the target variable.

Taking into account that, in this work, the dependent variable "compressive strength" is quantitative, three growing algorithms are available in SPSS: CHAID (Chi-squared Automatic Interaction Detector) [34], exhaustive CHAID [35] and CRT [36]. These decision trees can be used for both classification and regression problems. Note that, in a regression tree, the leaves or nodes predict a real number and not a class. The regression trees treated here use the mean of the corresponding nodes.

Both CHAID and exhaustive CHAID algorithms create decision trees with multiple branches. Taking into account that, in both algorithms, only nominal or ordinal categorical predictors are allowed, continuous input variables are first transformed into ordinal categorical predictors. For classification problems, these algorithms use as splitting criterion the Chi-square test and for regression problems the F-test (including Bonferroni adjustment) [37]. The exhaustive CHAID algorithm is a modification of the CHAID algorithm that performs a more thorough heuristic for examining all possible splits of each predictor variable, choosing a partitioning that corresponds to the most significant split [38].

Whereas the CHAID and exhaustive CHAID algorithms build non-binary trees and their performance is determined by the corresponding significance test, the CRT algorithm builds binary trees in order to maximize within-node homogeneity. The extent to which a node does not represent a homogeneous subset of cases is an indication of impurity. For categorical dependent variables,

SPSS incorporates several impurity measures for building the tree, including the well-known Gini index [33]. For quantitative dependent variables, the impurity is measured by means of the least-squared deviation (LSD) [37]. It is computed as the within-node variance, adjusted for any frequency weights or influence values [39].

2.3. Artificial neural networks

Artificial neural networks (ANN) are becoming useful tools for modelling of civil engineering problems. These methods, based on biological networks, can be used to model complex relationships between inputs and outputs or to find patterns. Recently, several works have analyzed their use for modelling the compressive strength of the concrete, see e.g., [22, 23, 40]. Most research is based on back propagation neural networks. In order to compare the Galerkin methodology treated here with the use of ANN, we have considered the Levenberg-Marquardt back propagation algorithm analyzed in [22]. Generally, this algorithm is faster than the traditional back propagation algorithm, which use the gradient descent algorithm to obtain the weights of the neurons [41].

2.4. Criteria for model selection

Let us denote $y_i,\ i = 1, 2, \ldots, n$, the observed values, and $\hat{y}_i,\ i = 1, 2, \ldots, n$, the predicted values, and k the number of predictive variables in the model. Then, in order to measure the goodness of fit of the models we have considered the following parameters:

- Determination coefficient: $R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$. The determination coefficient is the square of the correlation coefficient R and takes values between 0 and 1. Particularly, this coefficient is useful to measure the degree of linear correlation between two variables, and it can be interpreted as the proportion of variability explained by the model.

- Adjusted determination coefficient or adjusted R^2: $R^2_{adj} = 1 - \frac{n-1}{n-k-1}(1 - R^2)$. This coefficient is better than the determination coefficient for measuring the goodness of fit of models with several independent variables.

- Mean squared error: MSE $= \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}$.
- Root mean square error: RMSE $= \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$.
- Mean absolute error: MAE $= \frac{\sum_{i=1}^{n}|y_i-\hat{y}_i|}{n}$.
- Mean absolute percentage error: MAPE $= \frac{1}{n}\sum_{i=1}^{n}|\frac{y_i-\hat{y}_i}{y_i}|$.

3. Parallel implementation of the Galerkin methodology

All parallel algorithms analyzed in this work have been implemented in C++ on an HPC cluster of 18 nodes HP Proliant SL390s G7 connected through a network of low-latency QDR Infiniband-based. Each node consists of two Intel XEON X5660 hexacore at up to 2.8 GHz and 12MB cache per processor, with 48 GB of RAM. The operating system is CentOS Linux 5.6 for x86 64 bit.

Taking into account the hierarchical hardware design of this high performance system, the parallel algorithms have been implemented combining distributed memory parallelization on the interconnected nodes with shared memory parallelization inside each node. For this purpose, MPI [42] and OpenMP [43] programming models were combined into a hybrid paradigm in which MPI is used for data distribution among nodes and OpenMP to exploit loop level parallelism within each node. In this way we have used a philosophy of distributed shared memory where various OpenMP threads are considered for each MPI process, that is, $p = s \times t$ indicates that s nodes of the parallel platform have been used for data distribution and for each one of these nodes, t OpenMP threads have been considered. Particularly, if $s = 1$, the algorithms are executed in shared memory using t threads on a single node. Conversely, if $t = 1$, we are working on distributed memory using s nodes.

In order to explain our parallel approaches for the proposed Galerkin methodology, let P be the number of FEM points to be estimated. In the first two approaches, the P FEM points to be estimated are divided among the s nodes in such a way that each node gets approximately the same amount of points. The computations of all points to be estimated inside a node are assigned to the

threads using a dynamic scheduling strategy, where groups of points of a user determined size (called chunk size) are assigned to the threads on a first-come, first-served basis. We have experimented with two ways of mapping the P FEM points into the nodes: using consecutive points assigned to each node and using a cyclic distribution. These approaches have been referred to as block mapping algorithm and cyclic mapping algorithm, respectively [44].

The time needed for the estimation of each point could be very different and hence the computational cost assigned to each MPI process can differ immensely when static mappings are considered. In order to balance the calculations of the FEM points, we have constructed a third approach in which a dynamic pool of $p = (s \times t) + 1$ workers is considered. This approach, labelled as dynamic algorithm, is also a hybrid MPI/OpenMP implementation in which the work is dynamically assigned to the processes. A master-worker paradigm is applied. More specifically, the first MPI process becomes the master process, which does not any calculation but hands out tasks to the workers. The rest of the $s \times t$ processes are workers, which receive a chunk of work, finish it, return the result to the master process, and then wait for more work. In our implementation a task is defined as the computations needed to estimate a FEM point. A priori, this approach is designed to get a better load balancing among processes. It needs that multiple OpenMP threads make MPI library calls. The MPI standard defines various classes of thread usage. We have experimented with two types of thread usage: MPI_THREAD_SERIALIZED, that is, multiple threads may make MPI calls, but only one at a time (all MPI calls are serialized), and MPI_THREAD_MULTIPLE, in which multiple threads may call MPI, with no restrictions. Similar conclusions were obtained in both cases.

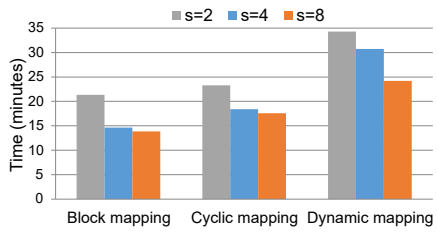
Note that the implementations of the proposed Galerkin methodology need to store a large number of intermediate computations in order to reduce the number of times these computations are performed. Each time that a new intermediate computation is needed, the algorithms check if it was already calculated. To store these large amount of data, hashed associative containers [45] were used. If the parallel algorithms are executed in shared memory ($s = 1$), a

unique hashed associative container shared by all processes (OpenMP threads) is needed; thus, all intermediate computations obtained by all processes can be reused by the other processes. On the other hand, when more than one node is used ($s > 1$) in the above explained approaches, s different hashed associative containers are needed, each one in the local memory of each MPI process. We point out that, in this case, the use of a global hashed associative container results in a non efficient algorithm due to the large amount of communications needed among processes running in different nodes.

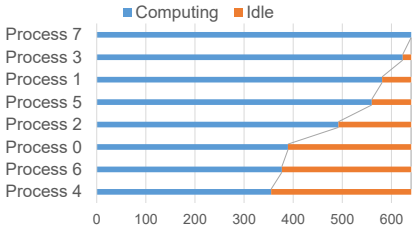
Figure 1 compares the three approaches for the LWAC-6V problem when 6 OpenMP threads per node are used. Figure 1(a) shows the time needed for the three algorithms when 2, 6 or 8 nodes are considered. It is observed that the block mapping algorithm reduces the execution time compared with the other two approaches (cyclic and dynamic mapping). The gain achieved with the block mapping algorithm in relation to the other approaches was up to 52.37%.

The block mapping algorithm uses a static distribution of the P points among the MPI nodes, as well as the cyclic mapping algorithm. This static mapping could cause an unbalanced workload that could have a significant impact on the performance of the parallel execution. In fact, in these cases, the MPI processes, each one working in a node, become idle as the computation proceeds. Figures 1(b) and 1(c) show this situation, in which, for example, process 0 of the block mapping algorithm becomes idle after 374 points (out of a total of 640) have been evaluated. In order to alleviate this situation, the dynamic mapping algorithm, based on a parallel dynamic load balancing strategy, was considered. Table 4 displays the number of points assigned to each MPI process for the static running compared with one of the dynamic runnings. Figure 1(d) shows the idle processes when this algorithm is executed using 8 nodes. We observe that all processes are working during almost all the execution time. Nevertheless, this behaviour does not have a positive impact when comparing the execution time of this algorithm with the block mapping algorithm. The thread options needed in the parallel dynamic implementation (MPI_THREAD_SERIALIZED or MPI_THREAD_MULTIPLE), have some disadvantages that can degrade the com-

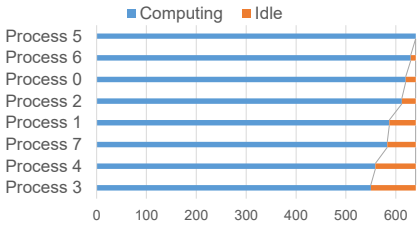
munication performance due to internal synchronization overheads, specially when the implementation has short length messages (as in the case of our algorithm). A big number of messages coming from different threads may cause additional latency overheads [46], obtaining that generally the block mapping algorithm also outperforms the dynamic one. On the other hand, in order to evaluate the use of hashed associative containers in each of our approaches, we obtained, for the computation of each point, the percentage of times that data stored in a container are reused. The averages of these percentages over all the evaluated points are 88.4%, 75.9% and 71.6% for the block, cyclic and dynamic mapping, respectively. Hence, the block mapping algorithm makes a better use of its containers that results in a better performance.



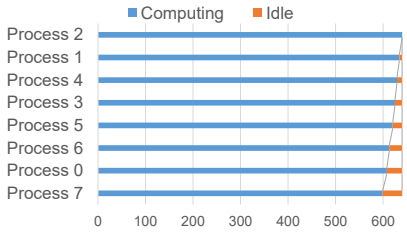
(a) Comparison of several mappings.



(b) Block mapping.



(c) Cyclic mapping.



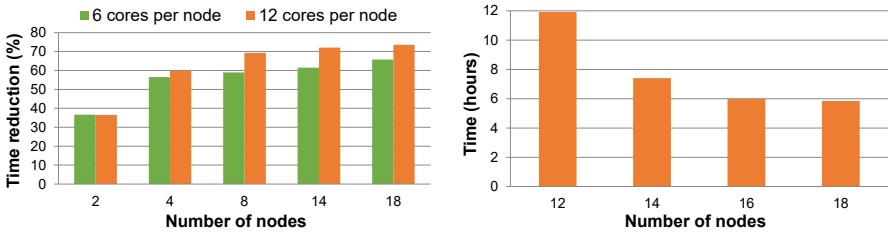
(d) Dynamic mapping.

Figure 1: Parallel Galerkin algorithm, complexity= 20, 6 cores per node, LWAC-6V problem.

Distribution	Static	Dynamic
Process 0	160	101
Process 1	160	76
Process 2	160	54
Process 3	160	68
Process 4	160	68
Process 5	160	130
Process 6	160	71
Process 7	160	72

Table 4: Number of points per MPI process.

Figure 2 shows the behaviour of the block mapping algorithm for several number of nodes and threads. We observe, in Figure 2(a), that a considerable time reduction is achieved when compared with the corresponding shared memory algorithm (only one node). On the other hand, Figure 2(b) shows the time reduction when the number of nodes is incremented.

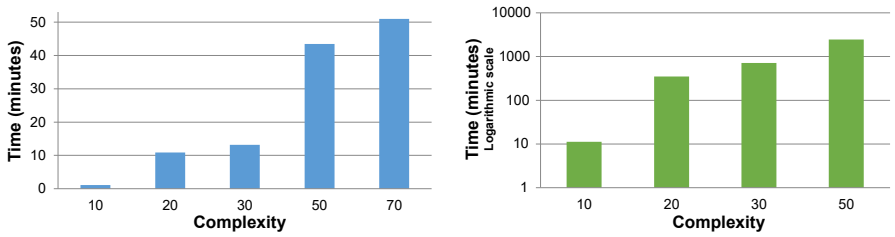


(a) LWAC-6V problem, time reduction in relation to one node, complexity=20.

(b) LWAC-7V problem, execution time, $p = s \times 12$, complexity=30.

Figure 2: Behaviour of parallel Galerkin algorithm.

Figure 3 displays how the execution time grows when the complexity is increased, for the problems treated here, justifying the need for the use of parallelism in our algorithms.



(a) LWAC-6V problem

(b) LWAC-7V problem

Figure 3: Parallel Galerkin algorithm. Execution time versus complexity, $p = 12 \times 12$.

4. Experimental modelling and numerical results

4.1. Statistical analysis

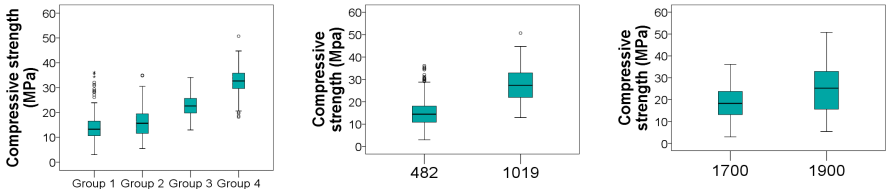
First, we have performed a univariate statistical analysis of the dataset. For testing the normality of the quantitative variables, both the Shapiro-Wilk test and the Kolmogorov-Smirnov test with the Lilliefors correction have been considered, obtaining that the assumption of normality is not meet for these variables. Therefore, the median and the lower and upper quartiles were obtained for each variable; see Table 5.

Variable description	median	Q_1	Q_3
LWAC fixed density (kg/m^3)	1800	1700	1900
LWA particle density (kg/m^3)	750.5	482	1019
Concrete laying time (min)	45.00	18.75	82.50
Vibration time (s)	20	10	40
Experimental Dry density (kg/m^3)	1677.15	1533.35	1810.84
P-wave velocity (m/s)	3718.49	3520.48	3945.65
Segregation index	0.999	0.978	1.021
Compressive strength (MPa)	20.25	14.37	28.76

Table 5: Descriptive statistics.

Furthermore, the nonparametric Kruskal-Wallis H test was used to analyze significant differences in the compressive strength depending on the type of LWAC, concrete laying time and vibration time. The obtained results showed statistically significant differences only for the type of LWAC. The order of

precedence of the LWAC type in relation to the compressive strength is obtained using the Mann Whitney U test to compare two independent samples, obtaining that the compressive strength of group 4 is higher than of group 3, the compressive strength of group 3 is higher than of group 2 and the compressive strength of group 2 is higher than of group 1; see Figure 4(a). Similar conclusions were obtained comparing the compressive strength depending on either the particle density of the LWA or the fixed density of the LWAC, Figures 4(b) and 4(c) display the box plots of the compressive strength for the two different groups of each one of these variables. As it can be expected, the more density of the LWAC or particle density of the LWA, the more compressive strength is obtained.



(a) LWAC type. (b) LWA particle density. (c) LWAC fixed density.

Figure 4: Boxplots for compressive strength of the LWAC.

4.2. Selection of the multiple linear regression model

Several multiple linear regression models for explaining the compressive strength of the concrete were analyzed.

The initial model assumes that the compressive strength of the concrete can be explained by means of linear relationships with the seven independent variables of the problem. The two binary qualitative variables, fixed density of the LWAC and particle density of the LWA, were incorporated to the analysis using the corresponding dummy variables. In both cases the value 0 was assigned to the smallest value of the original variable and the value 1 to the greatest one. The obtained linear model is explained in Table 6. Although the adjusted coefficient of determination was 0.765 and the F-test has obtained a

very small P-value (less than 10^{-198}), the t-tests show that several variables are non significant in the model. In fact, using backward elimination to identify the independent variables which have most impact on the outcome variable, the chosen MLR model has the same adjusted coefficient of determination (0.765). Table 7 includes both the unstandardized and standardized coefficients of this model.

	Unst. Coef.		St. Coef.		
Model	β_i	Std. Er.	β_i	t	Sig.
(Constant)	-33.964	4.912		-6.915	0.000
Concrete laying time	0.015	0.006	0.047	2.441	0.015
Vibration time	0.012	0.006	0.037	1.914	0.056
Experimental dry density	0.027	0.001	0.545	19.794	0.000
P-wave velocity	0.001	0.001	0.026	1.053	0.293
Segregation index	1.104	5.635	0.004	0.196	0.845
LWAC fixed density (dummy)	10.972	0.402	0.612	27.272	0.000
LWA particle density (dummy)	-0.297	0.458	-0.017	-0.649	0.517

Table 6: Full MLR model. Dependent variable: compressive strength.

	Unst. Coef.		St. Coef.		
Model	β_i	Std. Er.	β_i	t	Sig.
(Constant)	-30.623	1.647		-18.598	0.000
Concrete laying time	0.015	0.006	0.049	2.572	0.010
Vibration time	0.012	0.006	0.037	1.936	0.053
Experimental dry density	0.027	0.001	0.545	28.051	0.000
LWA particle density (dummy)	10.781	0.348	0.601	31.009	0.000

Table 7: Estimated MLR model. Backward elimination. Dependent variable: compressive strength.

In order to assess the validity of this model we have considered the following procedures: the linearity and homoscedasticity have been checked by means of the plot of standardized residuals versus standardized predicted values and the normality of the residual distribution has been studied using the Kolmogorov-Smirnov test, concluding that all necessary assumptions for the validity of the model are satisfied.

For the multicollinearity diagnostic, we have analyzed, among other aspects,

the variance inflation factor (VIF) and the tolerance of the predictors in the obtained estimated model. The tolerance is obtained as TOL $= 1 - R_i$, where R_i, is the squared multiple correlation of the i-th variable with the other predictor variables. A variable with a small tolerance close to 0, is almost a linear combination of the other input variables and it produces unstable regression coefficients. The variance inflation factor is the reciprocal of the tolerance. Values of VIF close to 1 are expected if there was no multicollinearity. As it can be seen in Table 8, no multicollinearity is identified in the estimated MLR model.

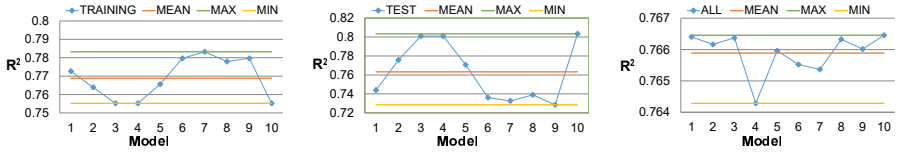
We want to point out that, before building the estimated MLR model, the degree of multicollinearity was also exhaustively evaluated for all variables of the full MLR model, identifying those involved in linear dependence relationships. We concluded that there was a strong collinearity between the segregation index and the experimental dry density. In fact, both variables are significant in their respective simple linear regressions, while when introducing only these two variables in a multiple linear regression model, the segregation index is not significant in the model. Therefore, the segregation index does not appear in the final estimated MLR model. A similar reasoning concludes that there also exists a strong collinearity between the LWAC fixed density and the LWA particle density and that only the LWA particle density should be included in the estimated MLR model.

Model	β_i	Std. Er.	Partial correlation	TOL	VIF
(Constant)	-30.623	1.647			
Concrete laying time	0.015	0.006	0.102	1.000	1.000
Vibration time	0.012	0.006	0.077	0.996	1.004
Experimental dry density	0.027	0.001	0.744	0.971	1.029
LWA particle density (dummy)	10.781	0.348	0.776	0.976	1.025

Table 8: Multicollinearity diagnostic of the estimated MLR model. Dependent variable: compressive strength.

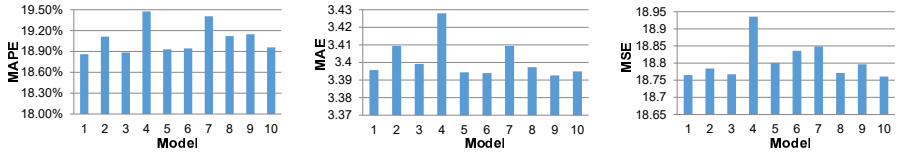
On the other hand, for the validation of the model, the sample has been partitioned into two datasets, the training set, which we have used to develop a new model using the same independent variables as in the above model, and

a test set, which is used to evaluate the predictive ability of the model. This process of validation has been done 10 times, by randomly partitioning the sample using a 75% of the sample for training and the rest for validation. As it can be observed in Figure 5, the determination coefficient was similar for both training and test datasets, obtaining a mean of 0.768 for the training samples and 0.763 for the test samples, while the determination coefficient mean of these models on all the sample was 0.766. Figures 6(a), 6(b) and 6(c) display the MAPE, MAE and MSE, respectively of these last models, and Table 9 compares several measures of goodness of fit for the model obtained without validation and the best model obtained after the validation process. As it can be seen, an adjusted R^2 of approximately 0.765 is obtained in both cases.



(a) Training dataset. (b) Test dataset. (c) All dataset.

Figure 5: Determination coefficient R^2 for the 10 MLR models.



(a) Mean absolute percentage error. (b) Mean absolute error. (c) Mean squared error.

Figure 6: MAPE, MAE and MSE for the 10 MLR models.

	Model without validation	Best model with validation
MAPE	19.04%	18.86%
MSE	18.72	18.76
MSE/Var	0.233	0.233
R^2	0.767	0.766
Adjusted R^2	0.765	0.765
MAE	3.394	3.396
RMSE	4.327	4.332

Table 9: Statistical measures for MLR models.

4.3. Selection of the regression tree

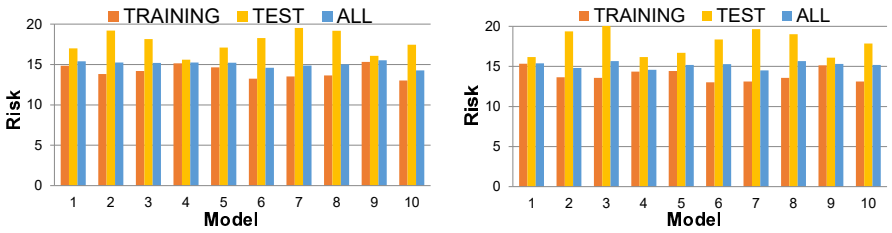
A performance comparison between the CHAID, exhaustive CHAID and CRT algorithms is conducted. The trees have been limited to five levels beneath the root node and the minimum number of cases considered for parent and child nodes, in the first trees analyzed here, has been 10 and 5, respectively. As significance value for splitting nodes and merging categories, the default significance level, 0.05 has been chosen. Moreover, for multiple comparisons, significance values have been adjusted using the Bonferroni method. With the CRT algorithm, the over-fitting of the model has been avoided by pruning the tree. Table 10 shows the risk estimate (MSE) and its standard deviation for the tree models. Although similar values were obtained, CHAID algorithm performs better than the other trees for predicting the compressive strength of concrete.

Problem	LWAC-6V		LWAC-7V	
Algorithm	Estimate	Standard deviation	Estimate	Standard deviation
CHAID	13.655	1.005	13.727	0.989
Exhaustive CHAID	14.720	1.083	14.346	1.039
CRT	14.156	0.938	13.895	0.950

Table 10: Tree algorithms comparison. Risk for LWAC-6V and LWAC-7V problems.

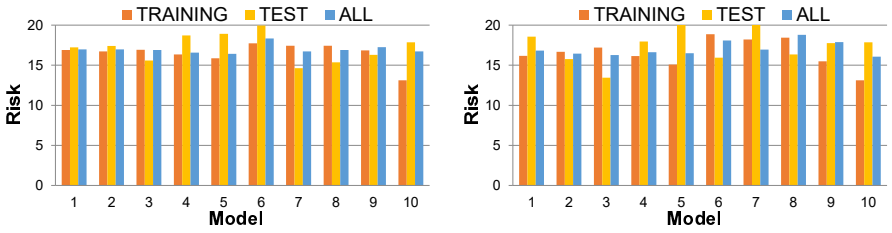
For the validation of the model, the sample has been partitioned into two datasets following the same process as in Section 4.2. The risk estimate (MSE) for the 10 models is explained in Figure 7. The variation of the risk was between 13.016 and 15.329 in the training samples, while for the test samples the risk

estimate varied between 15.585 and 19.523 using 6 variables. Using 7 variables, the variation was between 13.029 and 15.345 in the training samples and between 16.090 and 20.380 in the test samples. The risks of the test samples in Figure 7 are all larger than those of training samples. Therefore, it seems that these trees fit the training datasets better than the test datasets. Taking into account that regression trees are very susceptible to over-fitting, in order to avoid a presumable over-fit, the minimum number of cases needed for parent and child nodes has been augmented. Acceptable results have been obtained, with this pre-pruning method, using a minimum of 50 cases for parent nodes and 25 for child nodes. As it can be seen in Figure 8, the differences between risks of the training and test datasets are less than the differences obtained in the trees of Figure 7, at the expense of a greater risk in the training datasets.



(a) LWAC-6V problem. (b) LWAC-7V problem.

Figure 7: Risk variation for the CHAID tree, 10 models.



(a) LWAC-6V problem. (b) LWAC-7V problem.

Figure 8: Risk variation for the CHAID tree after pre-pruning process, 10 models.

Tables 11 and 12 display different measures of goodness of fit for the models

obtained with the CHAID algorithm using 6 and 7 input variables, respectively. By analyzing these tables, one can see that the trees built with a minimum of 10 cases for parent nodes and 5 for child nodes obtain better accuracy for both the model without validation and the best model with validation. However, the results obtained with a minimum of 50 cases for parent nodes and 25 for child nodes are comparable to those, besides that this model attempts to avoid a possible over-fitting.

	Model without validation		Best model with validation	
Measure	(a)	(b)	(a)	(b)
MAPE	15.43%	17.43%	16.25%	17.65%
MSE	13.655	16.102	14.279	16.433
MSE/Var	0.170	0.200	0.177	0.204
R^2	0.830	0.800	0.822	0.795
Adjusted R^2	0.828	0.798	0.821	0.793
MAE	2.808	3.101	2.922	3.175
RMSE	3.695	4.013	3.779	4.054

Table 11: Statistical measures, CHAID algorithm for LWAC-6V problem, (a) minimum of 10 cases for parent nodes and 5 for child nodes, (b) minimum of 50 cases for parent nodes and 25 for child nodes.

	Model without validation		Best model with validation	
Measure	(a)	(b)	(a)	(b)
MAPE	15.49%	17.20%	16.22 %	17.47%
MSE	13.727	15.686	14.498	16.062
MSE/Var	0.171	0.195	0.180	0.199
R^2	0.829	0.805	0.820	0.800
Adjusted R^2	0.827	0.803	0.818	0.798
MAE	2.829	3.056	2.928	3.075
RMSE	3.705	3.961	3.808	4.008

Table 12: Statistical measures, CHAID algorithm for LWAC-7V problem, (a) minimum of 10 cases for parent nodes and 5 for child nodes, (b) minimum of 50 cases for parent nodes and 25 for child nodes.

4.4. Selection of the artificial neural network architecture

For selecting the best ANN architecture, the same process followed in [22], for the problem with seven input variables (LWAC-7V problem), has also been

considered using only six input variables (LWAC-6V problem). Specifically, architectures of $\{2^n\}_{n=0}^{7}$ neurons in the hidden layer were obtained and for each one of these architectures, 50 models were randomly generated based on the corresponding dataset partition into training, testing and validation samples. These partitions were defined such that a 75% of the cases in the dataset was selected for training, a 15% for testing, and a 10% for validation. Taking into account the obtained results and the empirical criteria, recommended by different authors, for setting the number of hidden layer neurons [47], the architectures were refined considering models using from 1 to 10 neurons in the hidden layer. Based on the performed experiments, for the LWAC-7V problem, a six-neuron hidden layer architecture was finally chosen in [22]. Table 13 explains the variation of the determination coefficient R^2 for the 50 models generated using this architecture [22]. The choice of this architecture was based on the mean of R^2 which was one of the largest, and its most stability. After selecting this architecture, 10 different models were generated with this architecture to select the best model for predicting the compressive strength of the LWAC. Table 14 displays the different measures of goodness of fit obtained in [22] for the best of these 10 models.

Reasoning in a similar way as above, a nine-neuron hidden layer architecture has been chosen using 6 input variables (LWAC-6V problem). Table 15 summarizes the behaviour of R^2 for the 50 models generated with this architecture. Table 16 and Table 17 display different measures of goodness of fit for the best of the 10 after generated models, with nine and six neurons, respectively. As it can be seen, by comparing these tables, the architecture with nine neurons outperforms the architecture with six neurons in the LWAC-6V problem.

R^2	Training	Test	All
Mean	0.906	0.883	0.901
Maximum	0.923	0.930	0.915
Minimum	0.886	0.811	0.882
Min. error	0.016	0.047	0.014
Max. error	0.021	0.072	0.019
Dispersion	0.037	0.120	0.033

Table 13: Variation of the determination coefficient of the selected ANN (six-neuron hidden layer architecture), LWAC-7V problem [22].

Measure	Estimate
MAPE	15.85%
MSE	14.028
MSE/Var	0.174
R^2	0.825
MAE	2.897
RMSE	3.745

Table 14: Statistical measures, best six-neuron hidden layer architecture, LWAC-7V problem [22].

R^2	Training	Test	All
Mean	0.847	0.806	0.838
Maximum	0.898	0.881	0.868
Minimum	0.809	0.710	0.808
Min. error	0.038	0.076	0.030
Max. error	0.050	0.096	0.030
Dispersion	0.089	0.172	0.060

Table 15: Variation of the determination coefficient of the selected ANN (nine-neuron hidden layer architecture), LWAC-6V problem.

Measure	Estimate
MAPE	16.07%
MSE	14.789
MSE/Var	0.184
R^2	0.816
MAE	2.939
RMSE	3.846

Table 16: Statistical measures, best nine-neuron hidden layer architecture, LWAC-6V problem.

Measure	Estimate
MAPE	19.45%
MSE	20.483
MSE/Var	0.255
R^2	0.745
MAE	3.444
RMSE	4.526

Table 17: Statistical measures, best six-neuron hidden layer architecture, LWAC-6V problem.

4.5. Selection of the Galerkin-based methodology

In order to test the parallel Galerkin methodology for predicting the LWAC compressive strength, different models have been designed, based on both the number of input variables and the complexity selected in that methodology. Table 18 shows the measures of goodness of fit using 6 input variables, while the results with the 7 available input variables are displayed in Tabla 19.

Complexity	10	20	30	50	70
R^2	0.749	0.830	0.868	0.9035	0.9203
Adjusted R^2	0.747	0.828	0.866	0.9026	0.9196
MAE	3.492	2.793	2.404	2.006	1.822
MAPE	19.96%	15.22%	12.74%	10.16%	8.86%
MSE	20.144	13.678	10.628	7.752	6.401
RMSE	4.488	3.698	3.260	2.784	2.530
MSE/Var	0.251	0.170	0.132	0.097	0.080

Table 18: Galerkin-based methodology, LWAC-6V problem.

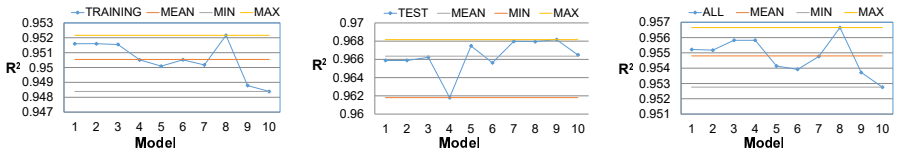
Complexity	10	20	30	50
R^2	0.813	0.893	0.924	0.9452
Adjusted R^2	0.811	0.892	0.923	0.9446
MAE	3.006	2.213	1.849	1.591
MAPE	16.68%	11.54%	9.15%	7.21%
MSE	15.050	8.589	6.101	4.404
RMSE	3.879	2.931	2.470	2.098
MSE/Var	0.187	0.107	0.076	0.055

Table 19: Galerkin-based methodology, LWAC-7V problem.

As it can be seen in these tables, using 6 input variables the determination coefficient of the model is $R^2 = 0.9203$ for a complexity of 70. However, with 7 input variables, a better R^2 is obtained with smaller complexities. Concretely,

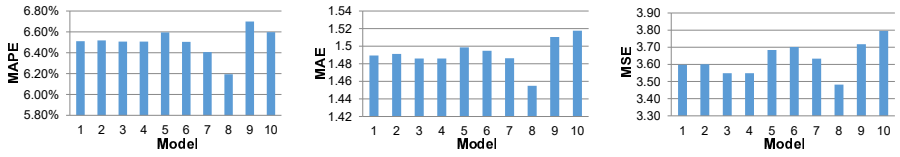
the determination coefficient of the model is $R^2 = 0.924$ setting a complexity of 30 and $R^2 = 0.9452$ for a complexity of 50. This last model obtains the best results with a MAPE of 7.21% and a MAE of 1.591. Note that the higher the model complexity, the higher the coefficient of determination and the lower the absolute percentage error and the mean absolute error.

For the validation of the Galerkin model, the same process as in previous sections has been considered. Figure 9 explains the variation of R^2 for the 10 models randomly generated using a 75% of the cases for training and the remainder for testing the model. As it can be observed in Figure 9, similar results were obtained in the 10 models, achieving a mean of 0.951 for the training samples, 0.966 for the test samples, and 0.955 for the entire sample. Figures 10(a), 10(b) and 10(c) display the MAPE, MAE and MSE, respectively for the 10 obtained models. The best model after the validation process gets a MAPE of 6.19%, a MAE of 1.455 and a MSE of 3.482. These results are close to those obtained in Table 19, for the complete dataset without validation.



(a) Training dataset. (b) Test dataset. (c) All dataset.

Figure 9: Determination coefficient R^2 for Galerkin-based methodology, LWAC-7V problem, complexity=50, 10 models.



(a) Mean absolute percentage error.

(b) Mean absolute error.

(c) Mean squared error.

Figure 10: MAPE, MAE and MSE for Galerkin-based methodology, LWAC-7V problem, complexity=50, 10 models.

4.6. Analysis and comparison of the results

According to the results analyzed in the previous sections, the best MLR model has obtained an adjusted R^2 of 0.765 and a MAPE of 18.86% (see Table 9), that comparing with the other models studied in this work, the MLR model is the worst for describing the segregation phenomenon in our experimental problem. On the other hand, the best regression trees and artificial neural networks have obtained similar results achieving an adjusted R^2 of approximately 0.82 and a MAPE around 16% (see Table 14).

Tabla 20 summarizes the measures of goodness of fit obtained with the Galerkin-based methodology for the 10 randomly generated models. This table includes the means, the medians and the confidence intervals (CI) for the means of these measures. These intervals have been obtained using a significance level of 0.05. Clearly, the Galerkin-based methodology is the best model, capable of predicting the compressive strength of the LWAC (using complexity 50) with a mean adjusted R^2 of approximately 0.95 and a mean MAPE around 6.5%.

On the other hand, taking into account that the best estimated MLR model only needs 4 input variables to predict the compressive strength of the LWAC, with an adjusted coefficient of determination of 0.765, it is interesting to analyze the behaviour of the proposed Galerkin methodology using these input variables for designing the model, thereby reducing computational time. Table 21 shows the measures of goodness of fit obtained in this case, where the input variables are concrete laying time, vibration time, experimental dry density, and LWA

Measure	Median	Mean	CI (95%)
R^2	0.955	0.955	[0.9539, 0.9556]
Adjusted R^2	0.954	0.954	[0.9535, 0.9551]
MAE	1.490	1.492	[1.479, 1.504]
MAPE	6.51%	6.50%	[6.41%, 6.60%]
MSE	3.617	3.631	[3.5635, 3.6983]

Table 20: Galerkin-based methodology, complexity=50, 10 models.

particle density. For a complexity of 70, the model achieves an adjusted R^2 of 0.904 and a MAPE around 11.19%, which is acceptable to some extent, outperforming not only the estimated MLR model but also the other techniques analyzed in this work.

The execution time of this methodology for this problem, using a complexity of 70 and 4 input variables, was around 2.75 minutes in a sequential mode, while the selection of the artificial neural network architecture needed around 7 minutes. The MLR models and the regression tree models were processed in a matter of a few seconds. By means of the parallelization of the Galerkin methodology, the execution time was reduced to 27.45 seconds using one node with eight cores. However, as the number of input variables or the complexity increases, the proposed Galerkin methodology has the highest computational cost compared with the above techniques and, hence, its parallelization becomes still more necessary.

Complexity	10	20	30	50	70
R^2	0.743	0.845	0.870	0.892	0.904
Adjusted R^2	0.741	0.844	0.869	0.891	0.903
MAE	3.588	2.755	2.507	2.263	2.113
MAPE	20.19%	15.28%	13.68%	12.15%	11.19%
MSE	20.642	12.460	10.425	8.663	7.698
RMSE	4.543	3.530	3.229	2.943	2.775
MSE/Var	0.257	0.155	0.130	0.108	0.096

Table 21: Galerkin-based methodology using the same input variables as the estimated MLR model.

5. Conclusions

In this work a Galerkin-based methodology has been studied in order to predict the compressive strength of the lightweight aggregate concrete using ultrasonic pulse velocity. For this purpose the following input variables have been considered: theoretical density of the concrete, particle density of the aggregate, laying time of the concrete, vibration time, dry density of the specimen obtained experimentally after 28 days, segregation index, and P-wave velocity. Due to memory requirements and the computational cost of the Galerkin methodology, a parallel algorithm has been designed, analyzing different approaches and data distributions. Concretely, MPI and OpenMP programming models were combined into a hybrid paradigm in which MPI is used for data distribution among nodes and OpenMP to exploit loop level parallelism within each node. For data distribution, block mapping, cyclic mapping and dynamic mapping were considered, obtaining generally the best results with the block mapping algorithm.

On the other hand, the Galerkin methodology has been compared with multiple linear regression models, regression trees and artificial neural networks, and the validation of the models was performed using split-sample validation.

As expected, MLR models were the worst to explain the compressive strength of the lightweight aggregate concrete. With regression trees and artificial neural networks, similar results were obtained with an acceptable MAPE around $16-17\%$. Nevertheless, based on all analyzed measures of goodness of fit, such as R^2, MAE, MSE and MAPE, the Galerkin methodology treated in this work significantly outperforms those other analyzed data mining techniques, obtaining a MAPE of 7.21% for the complete dataset, and a mean MAPE between 6.41% and 6.60% with a confidence of 95% in the validation process. On the other hand, as the number of input variables or the complexity increases, the proposed Galerkin methodology has the highest computational cost compared with the above techniques, being the price to improve the accuracy. Therefore, the parallelization of this methodology is needed for solving realistic problems.

Acknowledgements

The authors would like to thank Antonio José Tenza-Abril for providing the LWAC dataset analyzed in this work.

This research was supported by the Spanish Ministry of Science, Innovation and Universities Grant RTI2018-098156-B-C54, co-financed by the European Commission (FEDER funds).

References

[1] S. C. Chapra, R. P. Canale, Numerical Methods for Engineers, McGraw-Hill Higher Education, 2011.

[2] J. W. Thomas, Numerical partial differential equations: finite difference methods, Springer Science & Business Media, 2013.

[3] C. A. Brebbia, J. C. F. Telles, L. C. Wrobel, Boundary element techniques: theory and applications in engineering, Springer Science & Business Media, 2012.

[4] R. Eymard, T. Gallouët, R. Herbin, Finite volume methods, Handb. Numer. Anal. 7 (2000) 713–1018, https://doi.org/10.4249/scholarpedia.9835.

[5] A. A. Munjiza, The combined finite-discrete element method, John Wiley & Sons, 2004.

[6] N. Bićanić, Encyclopedia of Computational Mechanics Second Edition, John Wiley & Sons, 2017, Ch. Discrete element methods, pp. 1–38.

[7] W. Hackbusch, U. Trottenberg (Eds.), Multigrid methods: proceedings of the conference held at Köln-Porz, Springer, 1986.

[8] M. Hatami, Weighted Residual Methods: Principles, Modifications and Applications, Academic Press, 2018.

[9] R. R. Shamshiri, Implementation of Galerkin's method and modal analysis for unforced vibration response of a tractor suspension model., Res. J. Appl. Sci. Eng. Technol. 7 (1) (2014) 49–55, https://maxwellsci.com/msproof.php?doi=rjaset.7.219.

[10] J. Chan, R. J. Hewett, T. Warburton, Weight-adjusted discontinuous Galerkin methods: wave propagation in heterogeneous media, SIAM J. Sci. Comput. 39 (6) (2017) A2935–A2961, https://doi.org/10.1137/16M1089186.

[11] M. Balázsová, M. Feistauer, M. Hadrava, A. Kosík, On the stability of the space-time discontinuous Galerkin method for the numerical solution of nonstationary nonlinear convection-diffusion problems, J. Numer. Math. 23 (3) (2015) 211–233, https://doi.org/10.1515/jnma-2015-0014.

[12] J. Chan, Weight-adjusted discontinuous Galerkin methods: Matrix-valued weights and elastic wave propagation in heterogeneous media, Int. J. Numer. Methods Eng. 113 (12) (2018) 1779–1809, https://doi.org/10.1002/nme.5720.

[13] M. Bitaraf, S. Mohammadi, Analysis of chloride diffusion in concrete structures for prediction of initiation time of corrosion using a new meshless approach, Const. Build. Mater. 22 (4) (2008) 546–556, https://doi.org/10.1016/j.conbuildmat.2006.11.005.

[14] T. Belytschko, D. Organ, C. Gerlach, Element-free Galerkin methods for dynamic fracture in concrete, Methods Appl. Mech. Eng. 187 (3-4) (2000) 385–399, https://doi.org/10.1016/S0045-7825(00)80002-X.

[15] F. J. Navarro-González, Y. Villacampa, A finite element numerical algorithm for modelling and data fitting in complex systems, Int. J. Comput. Methods Exp. Meas. 4 (2) (2016) 100–113, https://doi.org/10.2495/CMEM-V4-N2-100-113.

[16] F. J. Navarro-González, Y. Villacampa, A new methodology for complex systems using n-dimensional finite elements, Adv. Eng. Softw. 48 (2012) 52–57, https://doi.org/10.1016/j.advengsoft.2012.02.001.

[17] F. J. Navarro-González, Y. Villacampa, Generation of representation models for complex systems using Lagrangian functions, Adv. Eng. Softw. 64 (2013) 33–37, https://doi.org/10.1016/j.advengsoft.2013.04.015.

[18] I. López, L. Aragonés, Y. Villacampa, F. J. Navarro-González, Gravel beaches nourishment: Modelling the equilibrium beach profile, Sci. Total Environ. 619–620 (2018) 772–783, https://doi.org/10.1016/j.scitotenv.2017.11.156.

[19] L. Aragonés, J. L. Pagán, I. López, F. J. Navarro-González, Y. Villacampa, Galerkin's formulation of the finite elements method to obtain the depth of closure, Sci. Total Environ. 660 (2019) 1256–1263, https://doi.org/10.1016/j.scitotenv.2019.01.017.

[20] A. Palazón, I. López, L. Aragonés, Y. Villacampa, F. J. Navarro-González, Modelling of Escherichia coli concentrations in bathing wa-

ter at microtidal coasts, Sci. Total Environ. 593–594 (2017) 173–181, https://doi.org/10.1016/j.scitotenv.2017.03.161.

[21] S. Chandra, L. Berntsson, Lightweight Aggregate Concrete, Elsevier, 2002.

[22] A. J. Tenza-Abril, Y. Villacampa, A. M. Solak, F. Baeza-Brotons, Prediction and sensitivity analysis of compressive strength in segregated lightweight concrete based on artificial neural network using ultrasonic pulse velocity, Const. Build. Mater. 189 (2018) 1173–1183, https://doi.org/10.1016/j.conbuildmat.2018.09.096.

[23] S. Charhate, M. Subhedar, N. Adsul, Prediction of concrete properties using multiple linear regression and artificial neural network, J. Soft Comput. Civ. Eng. 2–3 (2018) 27–38.

[24] S. Tavakkol, F. Alapour, A. Kazemian, A. Hasaninejad, A. Ghanbari, A. A. Ramezanianpour, Prediction of lightweight concrete strength by categorized regression, MLR and ANN, Comput. Concr. 12 (2) (2013) 151–167, https://doi.org/10.12989/cac.2013.12.2.151.

[25] J. S. Chou, A. D. Pham, Enhanced artificial intelligence for ensemble approach to predicting high performance concrete compressive strength, Const. Build. Mater. 49 (2) (2013) 554–563, https://doi.org/10.1016/j.conbuildmat.2013.08.078.

[26] H. I. Erdal, Two-level and hybrid ensembles of decision trees for high performance concrete compressive strength prediction, Eng. Appl. Artif. Intell. 26 (7) (2013) 1689–1697, https://doi.org/10.1016/j.engappai.2013.03.014.

[27] A. Fernández-Fanjul, A. J. Tenza-Abril, Méthode Fanjul: Dosage pondéral des bétons légers et lourds, Ann. Bâtim. Trav. Publics 5 (2012) 2–50.

[28] Y. Ke, Characterization of the mechanical behavior of lightweight aggregate concretes: experiment and modelling, Ph.D. thesis, Université de Cergy-Pontoise (2008).

[29] R. Bordawekar, B. Blainey, R. Puri, Analyzing Analytics, Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, 2015.

[30] N. Deshpande, S. Londhe, S. Kulkarni, Modeling compressive strength of recycled aggregate concrete by artificial neural network, model tree and non-linear regression, Int. J. Sustain. Built Environ. 3 (2014) 187–198, https://doi.org/10.1016/j.ijsbe.2014.12.002.

[31] A. Behnood, J. Olek, M. A. Glinicki, Predicting modulus elasticity of recycled aggregate concrete using M5′ model tree algorithm, Const. Build. Mater. 94 (2015) 137–147, https://doi.org/10.1016/j.conbuildmat.2015.06.055.

[32] A. Karbassi, B. Mohebi, S. Rezaee, P. Lestuzzi, Damage prediction for regular reinforced concrete buildings using the decision tree algorithm, Comput. Struct. 130 (2014) 46–56, https://doi.org/10.1016/j.compstruc.2013.10.006.

[33] L. Rokach, O. Maimon, Data mining with decision trees. Theory and applications, World Scientific Publishing, 2015.

[34] G. V. Kass, An exploratory technique for investigating large quantities of categorical data, Appl. Stat. 29 (2) (1980) 119–127, https://doi.org/10.2307/2986296.

[35] D. Biggs, B. D. Ville, E. Suen, A method of choosing multiway partitions for classification and decision trees, J. Appl. Stat. 18 (1) (1991) 49–62, https://doi.org/10.1080/02664769100000005.

[36] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and Regression Trees, The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis, 1984.

[37] T. Hill, P. Lewicki, Statistics: methods and applications. A comprehensive reference for science, industry, and data mining, StatSoft Inc., 2006.

[38] J. S. Chou, Y. C. Hsu, L. T. Lin, Smart meter monitoring and data mining techniques for predicting refrigeration system performance, Expert Syst. Appl. 41 (5) (2014) 2144–2156, https://doi.org/10.1016/j.eswa.2013.09.013.

[39] IBM, IBM SPSS decision trees 25, ftp://public.dhe.ibm.com/software/analytics/spss/documentation/statistics/25.0/en/client/Manuals/IBM_SPSS_Decision_Trees.pdf (2017).

[40] M. A. Kewalramani, R. Gupta, Concrete compressive strength prediction using ultrasonic pulse velocity through artificial neural networks, Autom. Constr. 15 (3) (2006) 374–379, https://doi.org/10.1016/j.autcon.2005.07.003.

[41] Saduf, M. A. Wani, Comparative study of back propagation learning algorithms for neural networks, Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3 (12) (2013) 1151–1156.

[42] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, MPI: The complete reference, 2nd Edition, The MIT Press, Cambridge, MA, 1998.

[43] OpenMP, OpenMP official site, http://www.openmp.org (2018).

[44] J. J. Dongarra, D. W. Walker, Software libraries for linear algebra computations on high performance computers, SIAM Rev. 37 (2) (1995) 151–180, https://doi.org/10.1137/1037042.

[45] The Google sparse hash map, http://goog-sparsehash.sourceforge.net/doc/sparse_hash_map.html (accessed December 6, 2018).

[46] INTERTWinE Consortium, Best Practice Guide to Hybrid MPI+OpenMP Programming, https://www.intertwine-project.eu/sites/default/files/images/INTERTWinE_Best_Practice_Guide_MPI%2BOpenMP_1.1.pdf (2017).

[47] T. K. Šipoš, I. Miličević, R. Siddique, Model for mix design of brick aggregate concrete based on neural network modelling, Const. Build. Mater. 148 (2017) 757–769, https://doi.org/10.1016/j.conbuildmat.2017.05.111.