Journal of Computational Science 63 (2022) 101817

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

A parallel methodology using radial basis functions versus machine learning approaches applied to environmental modelling

Violeta Migallón [a], Francisco J. Navarro-González [b], Héctor Penadés [a], José Penadés [a,*], Yolanda Villacampa [b]

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

ARTICLE INFO

Keywords:
Numerical modelling
Parallel programming
Radial basis function
Machine learning

ABSTRACT

Parallel nonlinear models using radial kernels on local mesh support have been designed and implemented for application to real-world problems. Although this recently developed approach reduces the memory requirements compared with other methodologies suggested over the last few years, its computational cost makes parallelisation necessary, especially for big datasets with many instances or attributes. In this work, several strategies for the parallelisation of this methodology are proposed and compared. The MPI communication protocol and the OpenMP application programming interface are used to implement the algorithm. The performance of this methodology is compared with various machine learning methods, with particular consideration of techniques using radial basis functions (RBF). Different methods are applied to model the daily maximum air temperature from real meteorological data collected from the Agroclimatic Station Network of the Phytosanitary Alert and Information Network of Andalusia, an autonomous community of southern Spain. The obtained goodness-of-fit measures illustrate the effectiveness of this nonlinear methodology, and its training process is shown to be simpler than those of other powerful machine learning methods.

1. Introduction

Mathematical models attempt to describe the human understanding of real-world objects or systems in mathematical terms. The process by which a simplified mathematical reality is obtained from a more complex physical reality is known as modelling [1]. Since there are conceptual models for which a formulation using analytical mathematical expressions is infeasible, various numerical approaches have been proposed for the problem of modelling, including numerical modelling methods and machine learning techniques [2].

Over the last few years, numerous methodologies for regression problems have been devised based on the finite element method (FEM). This method is used to compute an approximation of the real solution to a differential equation by means of discretisation [3]. That is, given a differential equation $D(f) = v$, where D is a differential operator defined on a domain Ω, and f, v belong to a functional space V, FEM is used to transform V to a finite space V_h with $\dim V_h = Q$, determined by discretisation (or a mesh), and the domain Ω is partitioned into N subdomains such that $\Omega = \cup_{i=1}^{N} \Omega_i$. The problem is then reformulated as $D(f_h) = v_h$, where $f_h, v_h \in V_h$. Hence, for a basis $B_h = \{\varphi_1(x), \varphi_2(x), \dots, \varphi_Q(x)\}$ of V_h, the approximate solution has the following form:

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

In this context, the basis functions are known as shape functions. Discretisation implies that each shape function must satisfy a set of conditions related to its values at a set of Q points called nodes $\{\zeta_j\}_{j=1}^{Q}$, that is, $\varphi_i(\zeta_j) = \delta_{ij}$. In addition, $\varphi_i(x) = 0$ when the node ζ_i is not part of the element containing point x.

In its most general form, FEM is used to determine the values u_i in (1) using weighted residual methods. Examples of well-known weighted residual methods include the collocation, sub-domain, least-squares, and Galerkin methods [4]. These methods have not only been widely applied in diverse fields of engineering [5–8] but also in other disciplines such as atmospheric modelling [9–11] or biological modelling [12–15].

In previous works, authors have presented different methodologies for regression problems based on FEM. For instance, the methodology described in [16] uses FEM to generate a geometric model in

* Corresponding author.
E-mail addresses: violeta@ua.es (V. Migallón), francisco.navarro@ua.es (F.J. Navarro-González), jhpm1@alu.ua.es (H. Penadés), jpenades@ua.es (J. Penadés), villacampa@ua.es (Y. Villacampa).

https://doi.org/10.1016/j.jocs.2022.101817
Received 11 February 2022; Received in revised form 27 June 2022; Accepted 2 August 2022
Available online 10 August 2022

an n-dimensional hypercube, which provides the basis for defining a numerical methodology for studying and modelling complex systems. The regression algorithm proposed in [16] used a least-squares approach and was applied to environmental problems. This numerical methodology was improved in [17] through the use of the Lagrangian FEM.

A Galerkin-based methodology was proposed in [18] that reduced the algorithmic complexity of the methods developed in [16,17]. This approach has been successfully applied in several problems related to electrical and hydrodynamic engineering [18], and biological and coastal engineering [19–21]. However, although the complexity of the algorithm was reduced in relation to previous approaches, as the number of input variables or the mesh size was increased, the computational cost made it impracticable for sequential processing. Hence, only small datasets with few variables were considered in these works. To overcome these drawbacks, a parallel version of this method was proposed in [22]. This parallel Galerkin-based methodology was used to predict the compressive strength of lightweight aggregate concrete [23], and outperformed a variety of machine learning techniques, including artificial neural networks.

Recently, in [24], a new methodology based on radial basis functions [25] was presented, in which the algorithmic complexity did not depend on the mesh size but on the number of points used for computation and the number of input variables (see Section 2 for a comprehensive description of this methodology). The performance of this scheme was analysed in [24] using two well-known datasets, the Airfoil Self-Noise dataset and the Combined Cycle Power dataset, both of which are available from the UCI Machine Learning Repository [26]. The Airfoil Self-Noise dataset contains 1,503 instances, while the Combined Cycle Power dataset contains 9,568 instances, and the numbers of input variables were five and four, respectively. Although this methodology can offer an alternative to the above techniques for the study of complex systems, from a computational point of view the complexity order of the algorithm still strongly limits its use. Parallelisation is therefore necessary to solve realistic problems such as the one addressed in this work. In particular, the problem studied in this work involves modelling the daily maximum air temperatures based on meteorological data provided by the Agroclimatic Station Network of the Phytosanitary Alert and Information Network of Andalusia, an autonomous community of Spain. Section 3 explains in detail the characteristics of this area and the problem to be solved.

In the last decade of the twentieth century, machine learning methods began to be applied in the field of environmental sciences to extract information from data. Nowadays, these techniques expect to have a great impact on analysis and modelling in this field [2,27]. Hence, to study the performance of the parallel nonlinear modelling algorithm proposed in this work, our approach is compared with multiple linear regression (MLR), artificial neural networks (ANN), K-nearest neighbours (KNN) and support vector regression (SVR). Section 4 briefly introduces these methods, and Section 5 explains the goodness-of-fit measures used to compare these techniques. Section 6 describes and analyses the parallel approaches put forward in this work. Section 7 discusses the numerical results obtained in this study, and shows that the proposed methodology outperforms other techniques for the studied problem. Lastly, some conclusions are presented in Section 8.

2. The nonlinear modelling method

In order to describe the method developed in [24], let us consider a dataset of n d-dimensional real points, $x^{(i)} \in \mathbb{R}^d$, $i = 1, 2, \dots, n$. Consider that m of these points ($m < n$), which form the set L, have an observed dependent real value $y_j \in \mathbb{R}$. For the sake of simplicity, these m points are considered to be the first points in the dataset, that is, we have the set:

$$\{(x^{(1)}, y_1), (x^{(2)}, y_2), \dots, (x^{(m)}, y_m)\} \subset L \times \mathbb{R}. \quad (2)$$

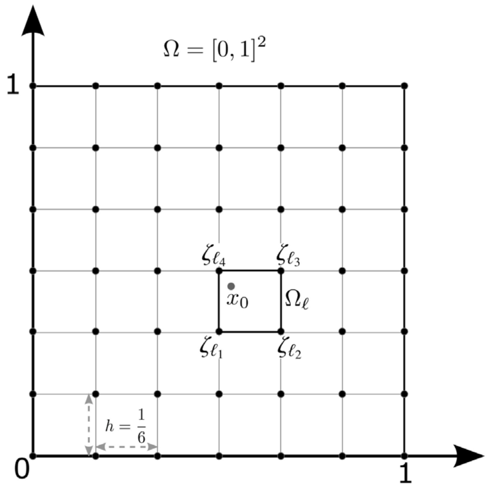


Fig. 1. Decomposition of the domain $\Omega = [0,1]^2$ with a mesh formed by regular elements with edge length $h = \frac{1}{6}$.

This set is used as a training sample for the cross-validation process. The remaining points are used as a test set:

$$U = \{x^{(m+1)}, x^{(m+2)}, \dots, x^{(n)}\}, \quad (3)$$

and are considered without the associated observed dependent values. However, in the numerical experiments reported in Section 7, the dependent values for the test set, $y_j \in \mathbb{R}$, $m+1 \le j \le n$, are known and will only be used to measure the goodness of fit of the model (see Section 5).

The problem we are interested in involves determining a real function f defined on the set $V = L \cup U$, which will be used to estimate the observed values $y_j \in \mathbb{R}$, $m+1 \le j \le n$, under the condition that $f(x^{(i)}) = y_i$, $i = 1, 2, \dots, m$. The estimated value assigned to each point $x^{(i)} \in \mathbb{R}^d$ is denoted by $\hat{y}(x^{(i)}) \in \mathbb{R}$, $i = 1, 2, \dots, n$.

The dataset is assumed to be normalised to the domain $\Omega = [0,1]^d$. For FEM, we used a mesh formed by regular hypercubic elements with edge length h. We set the complexity parameter c to the number of elements in each dimension, such that $h = 1/c$. The total number of hypercubic elements is $N = c^d$, and the total number of nodes is $Q = (c+1)^d$. These N elements and Q nodes are denoted as $\Omega = \cup_{i=1}^N \Omega_i$ and $\{\zeta_j\}_{j=1}^Q$, respectively.

Following [24], for interpolation over a mesh of finite elements, let us consider the decomposition of the domain $\left(\{\Omega_i\}_{i=1}^N, \{\zeta_j\}_{j=1}^Q\right)$ with the associated shape functions $\{\varphi_j(x)\}_{j=1}^Q$, that satisfy

$$\left.\begin{array}{rcl} \sum_{j=1}^{Q} \varphi_j(x) & = & 1 \\ \varphi_j(\zeta_i) & = & \delta_{ji} \end{array}\right\}.$$

Let Φ also be a parameterised radial function, that is, a function $\Phi : \mathbb{R}^+ \times \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ characterised by a parameter ω that meets the conditions

$$\left.\begin{array}{rcl} \forall \omega \neq 0,\ \lim\limits_{r\to\infty} \Phi(r,w) & = & 0 \\ \forall r \neq 0,\ \lim\limits_{w\to 0} \Phi(r,w) & = & 0 \end{array}\right\}.$$

A natural choice for the parameter ω of the radial function is the value corresponding to the element width h. A well-known example of a parameterised radial function is the Gaussian function $\Phi(r,h) = e^{-\left(\frac{r}{h}\right)^2}$.

Hence, given a point x_0 in the hypercube $\Omega_\ell \subset \Omega$ with 2^dnodes $\{\zeta_{\ell_k}\}_{k=1}^{2^d}$ (see Fig. 1), the regression estimation $\hat{y}(x_0)$ associated with the radial kernel Φ and the decomposition of the domain $\left(\{\Omega_i\}_{i=1}^N,\right.$

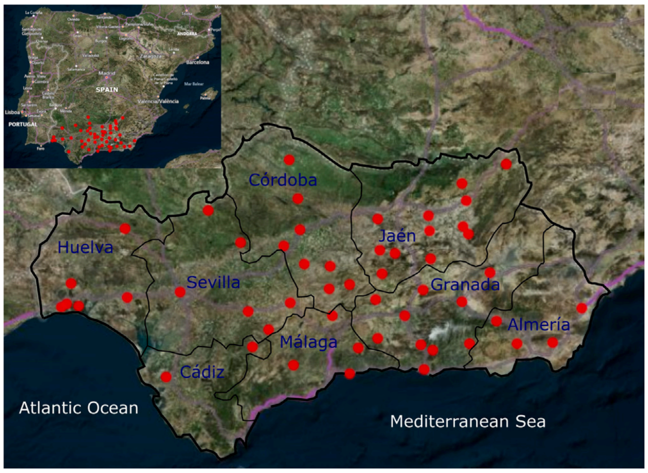


Fig. 2. Study area and spatial distribution of the automated weather stations.

$\{\zeta_j\}_{j=1}^{Q}\big)$ is defined as the interpolation of the weighted averages of the observed values on the nodes of Ω_ℓ,

$$\hat{y}(x_0) = \sum_{k=1}^{2^d} \varphi_{\ell_k}(x_0) \cdot c_*(\zeta_{\ell_k}), \tag{4}$$

where $c_*(\zeta_{\ell_k})$ can be computed as

$$\sum_{j=1}^{m} \left[c_*(\zeta_{\ell_k}) - y_j \right] \cdot \Phi(\|x^{(j)} - \zeta_{\ell_k}\|, h) = 0, \quad k = 1, 2, \dots, 2^d. \tag{5}$$

3. Study area and dataset

The area analysed in this work is the autonomous community of Andalusia. This region of Spain, with an area of approximately 87,597 km^2 and over 8.5 million inhabitants, is located in the south of the Iberian Peninsula (see Fig. 2). It is made up of eight provinces: Almería, Cádiz, Córdoba, Granada, Huelva, Jaén, Málaga and Sevilla. Its surface area represents 16.7% of the Spanish territory, and approximately 2.3% of the entire area of the European Union.

The geography of the western provinces is dominated by the Guadalquivir river valley. The valley depression separates the Sierra Morena in the north from the Bética mountain range in the southeast. The western limit is determined by the Guadiana river, which separates the province of Huelva from Portugal. Half of Andalusia is mountainous: a third of its surface is at an altitude of more than 600 m, and it has 46 peaks exceeding 1,000 m. The highest mountains in the Iberian Peninsula are found in the Sierra Morena (Mulhacen and Veleta), and exceed 3,400 m.

Andalusia generally has a temperate Mediterranean climate characterised by warm, dry summers and mild winters. However, the orographic heterogeneity of this territory, bathed by both the Mediterranean Sea and the Atlantic Ocean, generates a great variety of climates, including coastal, inland, mid-mountain and high-mountain climates, and semi-arid and arid climates (see [28] for more details of the classification of the Andalusian climatic zones).

We are interested in modelling the daily maximum air temperature in this region. Table 1 defines the input meteorological variables involved in this problem, which are related to the relative air humidity, the global radiation, the total rainfall, and the wind speed and direction. The elevation and month variables are used as auxiliary variables.

The climatic data used to analyse the behaviour of the algorithms were obtained from the Agroclimatic Station Network of the Andalusian Phytosanitary Alert and Information Network. These stations, equipped with electronic sensors, provide useful daily climatological information for the control of pests and diseases. The data were downloaded from https://juntadeandalucia.es/datosabiertos/portal.html. Daily maximum air temperatures from March to August were extracted from the data collected by 52 weather stations in Andalusia over three years (2017–2019). After a cleaning process had been applied to correct any inconsistencies in the data and remove instances with missing values, our dataset consisted of 27,988 data points.

The geographical distribution of these stations can be seen in Fig. 2, while Tables 2 and 3 display the summary statistics for the daily maximum air temperatures recorded by each station over the studied period. Table 2 shows the mean, maximum and minimum values of this variable for the stations sited in Almería, Cádiz, Córdoba and Granada, while Table 3 shows these values for the stations in Huelva, Jaén, Málaga and Sevilla.

Table 1
Meteorological predictor variables.

Variable description	Mean	Min	Max
Maximum relative humidity (%)	75.89	13.21	100.00
Minimum relative humidity (%)	31.10	4.59	100.00
Mean relative humidity (%)	52.61	10.60	100.00
Mean radiation (Wh/m^2)	1.03	0.00	2.96
Maximum wind speed (Km/h)	15.33	0.00	83.90
Minimum wind speed (Km/h)	0.01	0.00	13.12
Mean wind speed (Km/h)	3.74	0.00	33.58
Total rainfall (mm)	0.98	0.00	78.40
Dominant wind direction	2.45	1.00	4.00

Table 2
Weather stations sited in Almería, Cádiz, Córdoba and Granada; descriptive statistics of the daily maximum air temperature (°C) from March to August (2017–2019).

Province	Station	Mean	Min	Max
Almería	Nijar	27.14	13.90	41.39
	Antas	26.70	12.12	38.36
	Alhama de Almería	27.32	11.09	39.57
	Abla	25.79	8.13	40.43
Cádiz	Olvera	26.48	9.51	43.42
	Jerez de la Frontera	27.22	14.21	42.42
Córdoba	Nueva Carteya	27.72	9.90	43.47
	La Rambla	29.10	10.96	44.67
	Lucena	26.55	9.86	43.49
	Carcabuey	26.89	8.88	43.88
	Córdoba	28.48	11.83	45.93
	Guadalcazar	29.63	12.40	44.54
	Obejo	26.40	7.76	40.97
	Dos Torres	26.95	7.20	43.49
Granada	Iznalloz	25.76	5.82	41.70
	Guadix	27.22	8.34	41.10
	Motril	25.99	13.16	36.68
	Alhama de Granada	23.69	5.89	39.86
	Lecrín	25.18	9.52	39.00
	Montefrío	25.12	6.86	41.25
	Órjiva	27.42	12.11	40.33
	Santa Fe	28.62	8.33	44.40
	Zújar	25.84	6.68	41.74
	Ugijar	25.96	11.79	41.20

4. Machine learning approaches

4.1. Multiple linear regression

A multiple linear regression (MLR) model can be expressed as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_d X_d + \epsilon, \tag{6}$$

where Y is the dependent variable, $X_1, X_2, \dots, X_d$ are the independent or predictor variables, $\beta_0, \beta_1, \beta_2, \dots, \beta_d$ are the regression coefficients and ϵ is the random component of the model, that is, the unpredictable part of the dependent variable. When a dataset of n instances

Table 3
Weather stations sited in Huelva, Jaén, Málaga and Sevilla; descriptive statistics of the daily maximum air temperature (°C) from March to August (2017–2019).

Province	Station	Mean	Min	Max
Huelva	Lepe	26.81	13.93	40.73
	Bollullos	28.10	13.23	42.94
	Aracena	27.67	9.04	45.93
	Gibraleón	27.95	13.43	42.71
	San Bartolomé de la Torre	27.23	8.88	42.39
	Cartaya	27.51	14.31	42.54
Jaén	Bedmar-Garcíez	26.21	7.85	42.74
	Los Villares	24.93	5.82	40.27
	Torres de Albánchez	26.71	5.63	42.63
	Villacarrillo	26.39	5.63	42.83
	Castillo de Locubín	26.71	7.62	42.40
	Huelma	26.35	5.88	41.89
	Peal del Becerro	28.47	6.94	44.30
	Castellar de Santisteban	25.76	5.12	41.46
	Baeza	26.38	7.14	42.13
	Martos	28.36	9.03	45.16
	Higuera de Arjona	29.28	7.89	44.46
	Quesada	27.97	7.23	43.52
Málaga	Casarabonela	27.67	10.45	40.43
	Villanueva de Algaida	27.56	9.15	43.37
	Rincón de la Victoria	25.86	14.86	41.00
	Periana	25.73	9.44	40.34
Sevilla	Sevilla	30.03	13.90	44.31
	Alanis	25.75	8.63	41.85
	El Saucejo	25.29	8.97	41.04
	Puebla de los Infantes	28.73	11.36	44.70
	Estepa	27.36	10.77	44.60
	Puebla de Cazalla	26.71	10.59	43.48

is considered, the model defined in (6) can be written as follows:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_d x_{id} + \epsilon_i, \;\; i = 1, 2, \ldots, n.$$

To ensure the validity of a linear regression model, the relationship between the dependent variable and each of the independent variables must be linear, and the residuals ϵ_i, $i = 1, 2, \ldots, n$, must be normally distributed with $\mathrm{E}(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = \sigma^2$. It is also assumed that there is no multicollinearity between predictor variables.

4.2. *K*-Nearest neighbours

The K-nearest neighbour (KNN) algorithms are supervised machine learning algorithms that can be used for both classification and regression problems. In the case at hand, which involves predicting the values of a quantitative variable Y from the values obtained for the predictor variables $X_1, X_2, \ldots, X_d$, this algorithm considers a training set, say $\{(x^{(1)}, y_1), (x^{(2)}, y_2), \ldots, (x^{(m)}, y_m)\}$, with $x^{(i)} = (x_{i1}, \ldots, x_{id})$, $i = 1, 2, \ldots, m$, and seeks the K shortest distances between the data to be evaluated and the training data. More specifically, given a test instance $x \in \mathbb{R}^d$, the distances $d(x^{(i)}, x)$ for all i, $i = 1, 2, \ldots, m$, are computed and sorted from lowest to highest. Then, the predicted value for the test instance x, denoted as $\hat{y}(x)$, is obtained as the average of the outputs of its K nearest neighbours (denoted as $y_i(x)$, $i = 1, 2, \ldots, K$). That is, $\hat{y}(x) = \frac{1}{K} \sum_{i=1}^{K} y_i(x)$. Different types of distance may be used in the algorithm; in particular, for predicting values of continuous variables, the Euclidean and Manhattan distances are the most well-known. However, finding the optimal value of K is not an easy task, and this problem has received considerable attention in the literature (see e.g., [29] and the bibliography cited therein).

4.3. *Artificial neural networks*

Artificial neural networks (ANN) are a set of useful techniques in the field of machine learning that are based on biological neural networks, and are used to model complex relationships between inputs and outputs or to find patterns. Among the characteristics that differentiate them from other machine learning methods are their ability to learn dynamically and their fault tolerance. The simplest processing element in a neural network is the neuron. Each neuron i may have multiple inputs, $x_1, x_2, \ldots, x_d$, and synaptic weights, $w_{i1}, w_{i2}, \ldots, w_{id}$. An input or propagation function combines these inputs with their synaptic weights, and an activation function is subsequently applied to the obtained result to generate the corresponding output. One of the most well-known input functions is the weighted sum of the inputs based on the corresponding synaptic weights, such that for a neuron i, the input function is defined as $g_i(x_1, x_2, \ldots, x_d, w_{i1}, w_{i2}, \ldots, w_{id}) = \sum_{j=1}^{d} w_{ij} x_j + \theta_i$, where θ_i is a bias value. This value is treated as a weight whose input is equal to 1. The output value of neuron i can therefore be expressed as $y_i = f_i(\sum_{j=1}^{d} w_{ij} x_j + \theta_i)$, where f_i is some activation function. Different activation functions can be used depending on the characteristics of the problem. Some of the most widely used are the sigmoid function, the hyperbolic tangent function (tanh), the softmax function and the ReLU (Rectified Linear Unit) function (see e.g., [30]).

Neural networks are usually organised into layers, each consisting of a set of neurons such that the neurons in one layer may connect with those in the anterior or posterior layer. There are three types of layers in a neural network: the input layer, the hidden layers, and the output layer. The neurons in the input layer receive the inputs and propagate them to the next layer. The hidden layers form the intermediate layers, while the output layer transfers the information from the network to the outside, and gives the response for each input pattern.

To create the predictive model, a supervised and iterative learning process is followed, starting with an initial configuration of the synaptic weights. These weights are adjusted iteratively based on the inputs of the training set. A loss function compares the output with the data to be predicted. The learning algorithm seeks to find the weights of the neural network that minimise this function; gradient descent is an optimisation algorithm that is commonly used for this purpose. If the output of the neural network is continuous, a widely used loss function is the mean squared error (see e.g., [31]).

Since the proposed nonlinear modelling algorithm uses radial kernels, the alternative ANNs analysed in this work are the radial basis function (RBF) networks. In this way, the performance of both methods when using RBFs can be compared. An RBF network is a type of feedforward ANN with three layers that uses RBFs as activation functions. These functions are commonly used in function approximation and regression problems. RBF networks are usually implemented using the Gaussian function as an RBF (see e.g., [32]).

4.4. *Support vector regression*

Support vector machine (SVM) techniques are widely used for both classification and regression problems. Based on an independent and identically distributed training instance set, SVM classification techniques aim to find a discriminant function that can correctly predict new instances into a set of classes predefined for the output variable. From a geometric point of view, this is equivalent to finding the equation for the multidimensional surface that best separates the different classes in the feature space (see e.g., [33]). This discriminant technique, which uses predefined kernel functions, maps the data onto a higher-dimensional space before solving the machine learning task. From an algorithmic point of view, the learning process of SVM is a convex optimisation problem in which the optimal solution is found analytically rather than heuristically, such that for a predetermined kernel and a given training dataset, the training phase returns a uniquely defined model.

The support vector regression (SVR) technique is an extension of the SVM classification algorithm that is used for regression problems, in which the model outputs are continuous values [34]. In order to obtain the estimated continuous-valued multivariate function, an ϵ-insensitive region around the function, called the ϵ-tube, is considered; the optimisation problem then involves finding the tube that best approximates it. In other words, SVR can be formulated as an optimisation problem in which a multiobjective function is constructed from a convex ϵ-insensitive loss function and the geometrical properties of the tube. The objective is to find the narrowest tube centred around the surface, while minimising the prediction error (see e.g., [33]). In SVR, the estimation function is obtained by mapping the input samples onto a higher-dimensional feature space by means of a nonlinear mapping ϕ and learning a linear regressor in the feature space [35]. More specifically, given the training vectors $x^{(i)} \in \mathbb{R}^d$, $i = 1, 2, \dots, m$, and a vector $y \in \mathbb{R}^m$, we assume that the regression estimating function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ is defined by $f(x) = \omega^t \phi(x) + b$, where ω is a vector in the feature space and b is a scalar threshold. Then, to determine the values of ω and b, SVR solves the following primal problem [33,36,37]:

$$\begin{aligned} &\min_{\omega,b,\xi,\xi^*} \frac{1}{2}\omega^t\omega + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \\ &\text{subjet to } y_i - \omega^t\phi(x^{(i)}) - b \le \epsilon + \xi_i, \\ &\qquad \omega^t\phi(x^{(i)}) + b - y_i \le \epsilon + \xi_i^*, \\ &\qquad \xi_i, \xi_i^* \ge 0,\ i = 1, 2, \dots, m, \end{aligned}$$

where $C, \epsilon \in \mathbb{R}^+$ are input parameters and ξ, ξ^* are nonnegative vectors of slack variables. This optimisation problem can be solved in its dual form using the Lagrange multipliers method, as follows:

$$\begin{aligned} &\min_{\alpha,\alpha^*} \frac{1}{2}(\alpha - \alpha^*)^t P(\alpha - \alpha^*) + \epsilon e^t(\alpha + \alpha^*) - y^t(\alpha - \alpha^*) \\ &\qquad \text{subject to } e^t(\alpha - \alpha^*) = 0, \\ &\qquad 0 \le \alpha_i, \alpha_i^* \le C,\ i = 1, 2, \dots, m, \end{aligned}$$

where $\alpha, \alpha^* \in \mathbb{R}^m$ are the Lagrange multipliers, e is a vector of all ones and P is an $m \times m$ positive semidefinite matrix, with $P_{ij} = \phi(x^{(i)})^t\phi(x^{(j)})$.

Then, by applying the kernel trick, that is, taking $K(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^t\phi(x^{(j)})$, where K is a kernel function, the prediction of an input vector $x \in \mathbb{R}^d$ is obtained as $f(x) = \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)K(x^{(i)}, x) + b$.

To rewrite and solve the dual problem, several kernel functions can be used depending on the characteristics of the dataset, such as linear, polynomial or sigmoid functions or RBFs. Since the proposed methodology uses Gaussian radial basis kernels on local mesh support, its performance is compared with the SVR technique using Gaussian RBF kernels. This kernel is defined as $K(x^{(i)}, x^{(j)}) = e^{-\gamma\|x^{(i)}-x^{(j)}\|^2}$, where the parameter $\gamma > 0$ must be determined in the training phase.

5. Criteria for model selection

In order to measure the goodness of fit of the models treated here, the following parameters are considered, where y_i, $i = 1, 2, \dots, n$, denote the observed values, $\bar{y}$ is the mean of the observed values, $\hat{y}_i$, $i = 1, 2, \dots, n$, are the predicted values, and d is the number of predictive variables in the model.

- Relative error, $\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$. This goodness-of-fit measure is often used to determine the performance of ANN.
- Determination coefficient, $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$. This goodness-of-fit coefficient is one of the most frequently used in linear regression to measure the degree of linear correlation between two variables, and can be interpreted as the proportion of variability explained by the model. Note that the determination coefficient in a linear least-squares regression with an intercept is equivalent to the square of the correlation coefficient, and therefore takes values between zero and one. However, outside of this context, the determination coefficient may be negative.
- Adjusted determination coefficient or adjusted R^2, $R^2_{adj} = 1 - \frac{n-1}{n-d-1}(1 - R^2)$. This coefficient is better than the determination coefficient for measuring the goodness of fit of models with several independent variables.
- Mean squared error, MSE $= \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}$. This parameter reflects the systematic deviation of both predicted and observed values, as well as the precision of the prediction (see e.g., [38]).
- Root mean square error, RMSE $= \sqrt{\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n}}$. Although the RMSE is an appropriate indicator to represent the performance of the model, its sensitivity to outliers is the most common concern (see e.g., [39]).
- Mean absolute error, MAE $= \frac{\sum_{i=1}^{n}|y_i-\hat{y}_i|}{n}$. In some studies, the MAE is considered a better measure of goodness of fit than the RMSE (see e.g., [40]). Nevertheless, there are some circumstances where using the RMSE is more beneficial (for example, when the errors are normally distributed). Hence, it would be hard to argue that one is better than the other. Instead, as discussed in [39], a combination of metrics is often used to analyse the model performance.
- Mean absolute percentage error, MAPE $= \frac{100}{n}\sum_{i=1}^{n}|\frac{y_i-\hat{y}_i}{y_i}|$. This measure provides the error in terms of percentages. Both the MAPE and the MAE are robust to the effects of outliers, due to the use of absolute values in the formulae.

6. Parallel algorithms: Description and analysis

Due to the computational requirements of the methodology described in Section 2, we are interested in the development of parallel algorithms that can yield a considerable reduction in the computing time. In essence, a parallel algorithm running on a parallel computer can execute several operations simultaneously on different processing units, and the partial results can eventually be communicated between them. The outputs of all processing units are combined to obtain the final result.

The parallel algorithms developed in this work are implemented and analysed on a high performance computing (HPC) cluster with 20 nodes, HP Proliant SL390s G7, connected through a QDR Infiniband low-latency network. Each computing node has two Intel XEON X5660 hexacores, with up to 2.8 GHz and 12 MB cache per processor, with 48 GB of RAM; that is, each node consists of 12 core processors. Hence, the total number of available processing units is 240. The operating system is CentOS Linux 5.6 for x86 (64 bit), and the algorithms are implemented in the C++ language using the GNU C++ compiler known as g++.

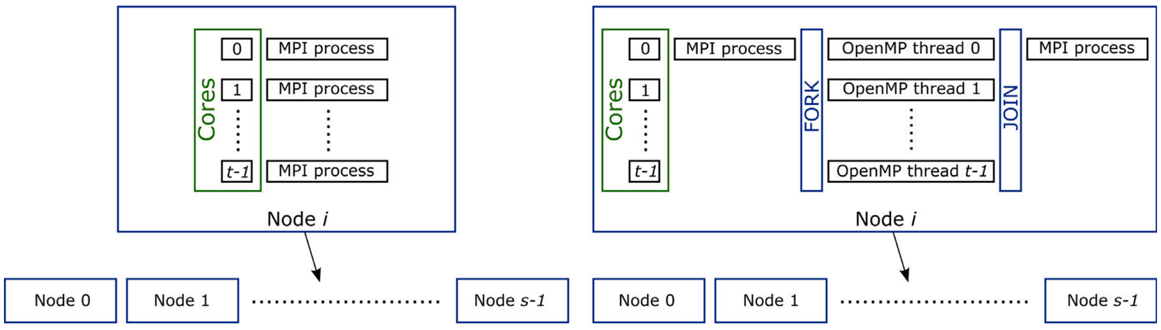
Following (4) and (5), the proposed methodology involves a loop over all points to be estimated. Since there are no data dependencies between the iterations of this loop, the computation of each estimated value can be assigned to a different processing unit. There are typically far fewer processing units than points to be computed, and a good strategy for distributing the computation of all estimated values among the available processing units is required.

The choice of strategy depends on the programming model used in the parallel implementation. In this work, two programming models are considered. The first is based on the use of the MPI (Message Passing Interface [41]) communication protocol for parallel programming; in this case, the MPI interface is used both for distributed memory parallelisation over the interconnected nodes and for shared memory parallelisation within each node. In the second programming model, in addition to the use of MPI for communications between nodes, OpenMP (Open Multi-Processing [42]) is used within nodes. OpenMP is a multithreading implementation that supports multi-platform shared-memory multiprocessing. This last programming model, in which MPI is used for data distribution among the nodes and OpenMP is applied to exploit loop-level parallelism within each node, is called a hybrid paradigm. Our parallel implementation of this hybrid model has no MPI calls inside the OpenMP parallel regions, and follows a hybrid master-only model (see e.g., [43,44]).

In the first parallel programming model, all processes are MPI processes running on different cores of several nodes. We use the notation

Algorithm 1 Parallel NLM-RK algorithm

```
1: Given a set of points defined in (2) and (3) normalised in the domain $\Omega = [0,1]^d$.
2: Given a complexity $c$ that determines a regular hypercubic decomposition of the domain $\left(\{\Omega_i\}_{i=1}^N, \{\zeta_j\}_{j=1}^Q\right)$ with shape functions $\{\varphi_j(x)\}_{j=1}^Q$.
3: Given a radial kernel $\Phi$.
4: Let $p$ be the number of MPI processes.
5: Divide the $n$ points across the $p$ processes, each one with $n_s$, $1 \le s \le p$ points.
6: for $s = 1 \to p$ in parallel do
7:     if dynamic then
8:         #pragma omp parallel for schedule(dynamic, chunk-size)
9:     else
10:        #pragma omp parallel for schedule(static)
11:    end if
12:    for $i = \sum_{l=1}^{s-1} n_l + 1 \to \sum_{l=1}^{s} n_l$ do
13:        Initialise the estimated value $\hat{y}(x^{(i)}) = 0$.
14:        Find the element $\Omega_\ell$ of the mesh containing $x^{(i)}$.
15:        Find the $2^d$ nodes of $\Omega_\ell$, namely $\zeta_{\ell_k}$, $1 \le k \le 2^d$.
16:        for $k = 1 \to 2^d$ do
17:            Initialise $nodeEstim = 0$.
18:            Initialise $dist = 0$.
19:            for $j = 1 \to m$ do
20:                Increment $dist$ with $\Phi(||x^{(j)} - \zeta_{\ell_k}||, h)$.
21:                Increment $nodeEstim$ with $y_j \cdot \Phi(||x^{(j)} - \zeta_{\ell_k}||, h)$.
22:            end for
23:            Update the node estimate as $nodeEstim = nodeEstim/dist$.
24:            Increment $\hat{y}(x^{(i)})$ with $nodeEstim \cdot \varphi_{\ell_k}(x^{(i)})$.
25:        end for
26:    end for
27:    Communicate the $n_s$ estimated values to a central process.
28: end for
```



(a) Pure MPI version. (b) Hybrid MPI+OpenMP version.

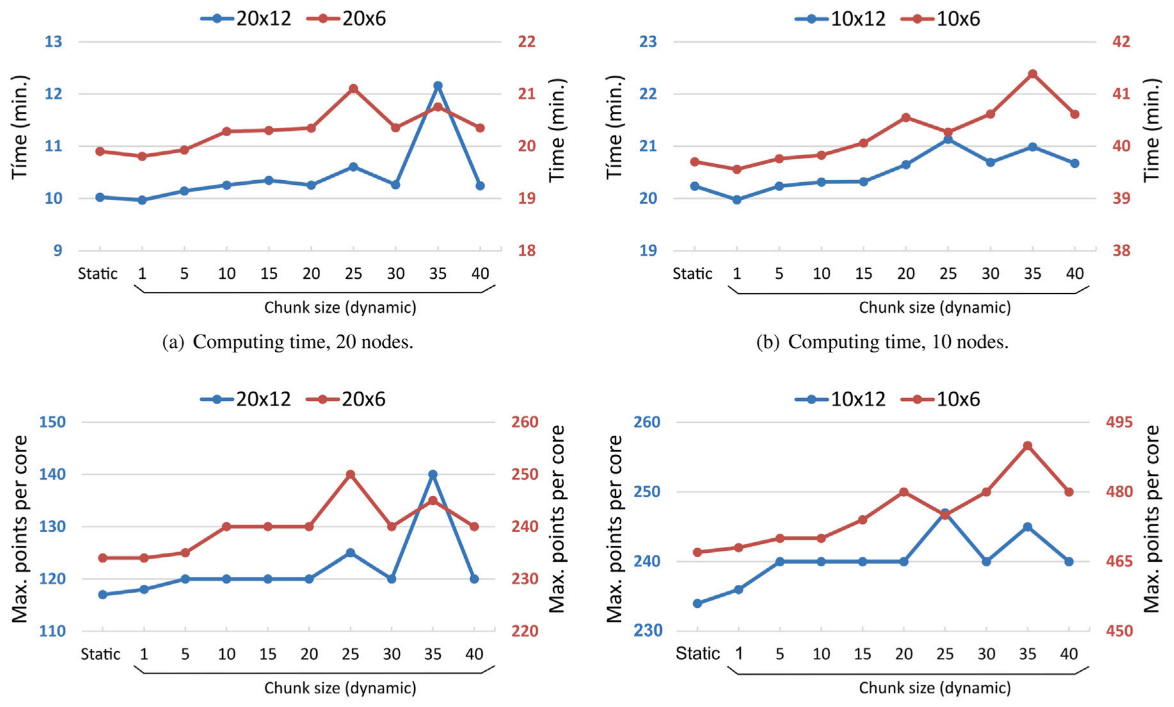
Fig. 3. Pure MPI and hybrid MPI+OpenMP parallel programming models.

$p = s \times t$ to state that p processes are running on s nodes of the parallel platform, such that t MPI processes are assigned to t different cores of each node. In the second parallel programming model, the $p = s \times t$ processes are assigned as follows. First, s MPI processes are executed on s nodes of the parallel platform. Then, for each MPI process, t OpenMP threads are started on t different cores of the node. Fig. 3 illustrates these models. Due to the characteristics of the HPC cluster, the values of s and t are less than or equal to 20 and 12, respectively.

The computational resources required to compute each estimated value are similar. Then, assuming that the processing units are homogeneous, the points to be estimated are distributed among the MPI processes in such a way that each one gets approximately the same amount of points. More specifically, the maximum difference in the number of estimated values assigned to two MPI processes is one. This data distribution is considered in both parallel programming models. In addition, two scheduling strategies (dynamic and static) are implemented when OpenMP is used within each node. In the dynamic scheduling strategy, the loop over the points to be estimated on each node is divided into groups of a user-determined size (called the chunk size) that are assigned to the threads on a 'first come, first served' basis. In the static scheduling strategy, without specifying a chunk size, OpenMP divides the loop into chunks of approximately equal size. Algorithm 1 describes these parallel programming models and scheduling strategies. Note that lines 7 to 11 of Algorithm 1 only apply when OpenMP is used within nodes.

Algorithm 1 not only describes the computation of the test points but also provides the estimation of the training points. In other words, this algorithm computes all the points needed to accomplish the cross-validation procedure. Note that the computational cost of each estimated value in Algorithm 1 is on the order of $O(m \cdot d \cdot 2^d)$; this is lower than for the previous methodologies presented in [17,18], which have a computational cost per estimated value on the order of $O(d \cdot (c+1)^{3d})$ and $O(d \cdot (c+1)^d)$, respectively. Unlike these approaches, the computational cost of the method presented in Algorithm 1 does not depend on the selected complexity c (see [24]).

A value of m = 22,391 was selected for the reported numerical experiments, meaning that an 80:20 ratio was used for the training and



(a) Computing time, 20 nodes.

(b) Computing time, 10 nodes.

(c) Maximum number of points per core, 20 nodes.

(d) Maximum number of points per core, 20 nodes.

Fig. 4. Behaviour of the static and dynamic scheduling strategies for the hybrid MPI+OpenMP model.

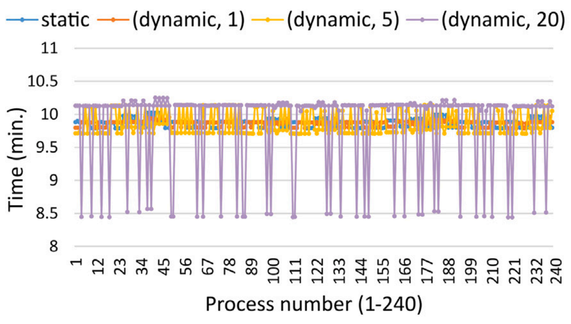
test datasets in Algorithm 1. This partition was done by means of simple random sampling.

Fig. 4 compares the static and dynamic scheduling strategies of the hybrid MPI+OpenMP model, for several values of the chunk size. Figs. 4(a) and 4(b) show the computing time (on two different vertical axes) when 20 nodes (blue axis on the left) and 10 nodes (red axis on the right) are used, respectively. The best dynamic scheduling strategy was obtained by setting the chunk size to one. Moreover, the computing time of the static scheduling strategy is comparable to that of the best dynamic specification. These conclusions are consistent with the fact that the dynamic scheduling strategy (with a chunk size greater than one) may yield a difference in the load balancing when all iterations have the same computational cost and the processing units are homogeneous. To illustrate this, Figs. 4(c) and 4(d) show the maximum number of points to be computed inside each core for the static and dynamic scheduling strategies. It is clear that a good load balance is achieved when this maximum is as small as possible. In this case, each process is assigned approximately the same amount of points to be estimated. It can be appreciated that these figures follow the same pattern as in Figs. 4(a) and 4(b), which show the computation time in each case; clearly, this time is closely related to the load distribution. The effect of the load distribution on the execution time for several scheduling strategies is illustrated in Fig. 5, which shows the execution time taken to compute all points assigned to each processing unit. A configuration of $240 = 20 \times 12$ processes is used in Fig. 5(a), while Fig. 5(b) shows the results for a configuration of $60 = 10 \times 6$ processes. The computing time for each processing unit remains within a small interval regardless of whether the dynamic scheduling strategy with a chunk size of one or the static specification is selected, that is, when a load-balanced distribution is considered.

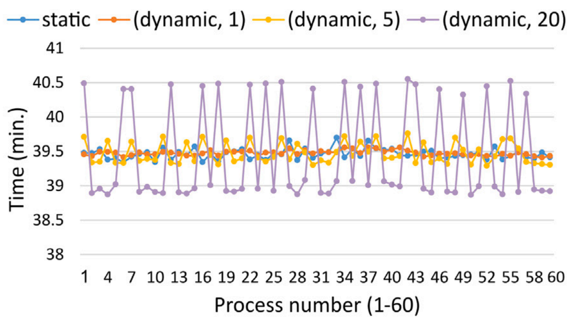
Fig. 6 compares the pure MPI model with the two best scheduling strategies for the hybrid MPI+OpenMP model: the dynamic strategy with a chunk size equal to one and the static scheduling strategy. This figure shows the behaviour of these models when varying numbers of nodes and cores are used. Fig. 6(a) shows the parallel execution times when 12 cores are used in each node, and Fig. 6(b) presents the times when six cores per node are used. It can be observed that a considerable reduction in the computation time is achieved as the number of processing units increases. Moreover, Fig. 6 shows similar performance for the pure MPI and hybrid MPI+OpenMP parallel programming models. However, in all cases, the best performance is achieved by the hybrid MPI+OpenMP model with a dynamic scheduling strategy and a chunk size of one. The differences between the pure MPI model and the hybrid MPI+OpenMP model are less significant when six cores per node are used. This is typical behaviour for pure MPI and hybrid MPI+OpenMP models: the performance of the pure MPI version is better for low core counts, but the hybrid version performs much better when the number of cores is increased (see e.g., [45,46]).

The abovementioned findings from a performance comparison between both models and scheduling strategies can also be seen in Fig. 7, which shows the speed-up and the parallel efficiency. The speed-up when computations are performed with p processing units is computed as $S_p = T_s/T_p$, where T_s is the elapsed time when executing a program on a single processor and T_p is the execution time when p processors are used. The efficiency can be considered as the average utilisation of the p allocated processing units, and is computed as S_p/p. Figs. 7(a) and 7(b) display the speed-up for the MPI model and several scheduling strategies for the MPI+OpenMP model, for a range of nodes (10 to 20 nodes), using 12 and 6 cores per node, respectively. These speed-ups are close to the perfect linear speed-up ($S_p = p$). Figs. 7(c) and 7(d) show the efficiency of those models for the full range of nodes (1 to 20 nodes). All efficiencies remain close to one as the number of processing units increases, thus indicating the good utilisation of the parallel resources. For example, the average time needed to calculate one point in the sequential mode is about 4.85 s, while this is reduced to 0.0214 s when 240 processing units are used.

Finally, it is interesting to explore the performance of the pure MPI model when all processing units appear as a distributed memory computing resource, and hence there is no possibility of using a hybrid model. In this case, a maximum of 20 MPI processes can be assigned, each to a node of the parallel resource. Fig. 8 displays, on two different vertical axes, the speed-up (blue axis on the left) and the efficiency (red axis on the right) of the MPI model for between 1 and 20 MPI processes. These speed-ups are very close to the perfect linear speed-up, giving an average efficiency of close to one (0.9975).

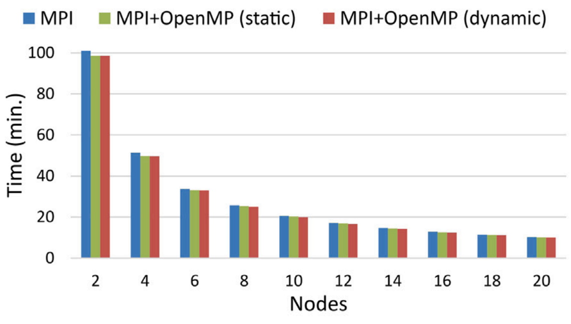


(a) 20 nodes and 12 cores per node.

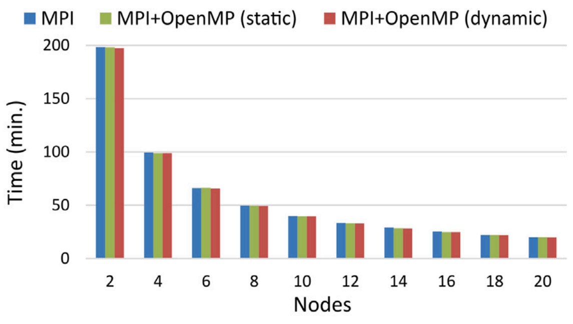


(b) 10 nodes and 6 cores per node.

Fig. 5. Computing time taken for each process of the hybrid MPI+OpenMP model, for several scheduling strategies.

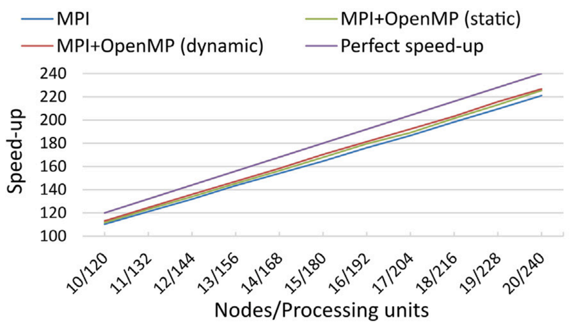


(a) 12 cores per node.

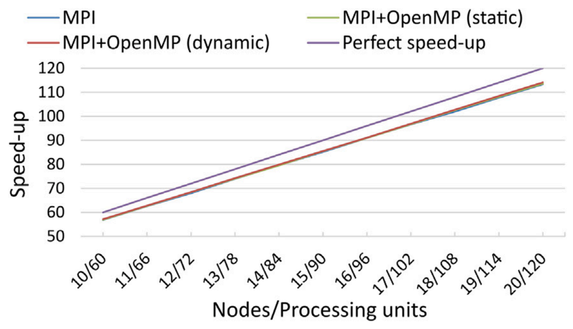


(b) 6 cores per node.

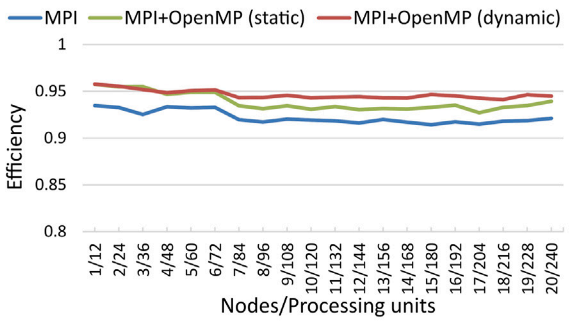
Fig. 6. Comparison of parallel programming models for varying numbers of nodes and cores.



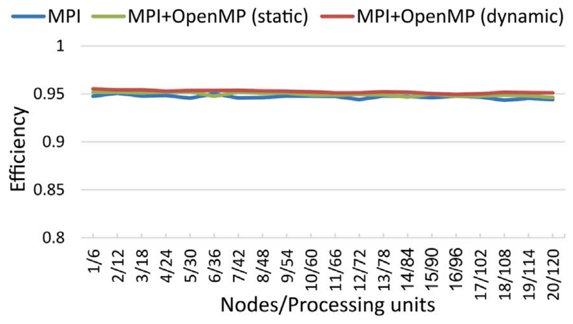
(a) Speed-up, 12 cores per node.



(b) Speed-up, 6 cores per node.



(c) Efficiency, 12 cores per node.



(d) Efficiency, 6 cores per node.

Fig. 7. Speed-up and efficiency of parallel programming models for varying numbers of nodes and cores.

7. Numerical experiments

To analyse the prediction performance of the proposed parallel NLM-RK algorithm, four machine learning techniques are adopted as benchmarks for comparison, using the real dataset described in Section 3. MLR, KNN, ANN and SVR are used. Regression trees were also explored, but these models failed to improve the performance of the above machine learning models. Note that the high variance of regression trees can produce poor predictive accuracy. Moreover, tree

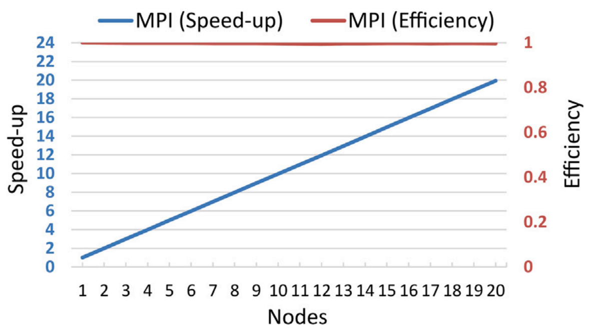


Fig. 8. Speed-up and efficiency of the pure MPI model.

Table 4
Descriptive statistics.

Variable description	Median	Q_1	Q_3
Maximum air temperature (°C)	27.54	21.49	33.09
Maximum relative humidity (%)	79.52	63.89	90.40
Minimum relative humidity (%)	27.27	18.6	40.35
Mean relative humidity (%)	51.67	39.08	65.67
Mean radiation (Wh/m^2)	1.07	0.84	1.23
Maximum wind speed (Km/h)	15.34	10.64	20.29
Minimum wind speed (Km/h)	0.00	0.00	0.00
Mean wind speed (Km/h)	3.46	2.23	4.95
Total rainfall (mm)	0.00	0.00	0.00
Dominant wind direction	2.54	1.00	3.33
Elevation (m)	463.00	234.00	698.00

models are neither smooth nor continuous, meaning that they may not be good at extrapolation. Hence, for the sake of simplicity, these results are omitted.

To validate each model, the sample was split into two datasets: the training set, which was used to estimate the new model, and a test set, which was used to evaluate the predictive ability of the model. This validation process was repeated 10 times, by randomly partitioning the sample into 80% for training and a 20% for testing. For the implementation and construction of the machine learning models, we used Python and version 1.0.1 of the Scikit-learn API [47] for predictive data analysis. In addition, IBM SPSS Statistics 26.0 [48] was used for several statistical analyses.

A preliminary statistical analysis to test the normality of the quantitative variables was carried out using the Kolmogorov–Smirnov test with the Lilliefors correction, and we concluded that there was evidence to reject the assumption of normality. Hence, additional statistical measures other than the mean were analysed. Table 4 displays the median and the lower and upper quartiles for the input and target variables.

The non-parametric Kruskal–Wallis H test was used to analyse the significant differences in the daily maximum air temperature in relation to the considered months or years, and the Mann Whitney U test was applied to analyse the differences between each pair of independent groups. As expected, the obtained results showed statistically significant differences in the distribution of the daily maximum air temperature, both between months and between years, with the differences being more noticeable between months (see Fig. 9). In addition to the input topographic and meteorological variables shown in Table 4, the month was therefore also used as an input variable for predicting the maximum air temperature.

7.1. Machine learning models

In this section, we explore the performance of the MLR, KNN, RBF networks and SVR in terms of obtaining the best models for predicting the daily maximum air temperature in the climatological problem considered here.

First, we analysed the MLR models. To validate these models, the linearity and homoscedasticity were checked by means of a plot of the standardised residuals versus the standardised predicted values, and the normality of the residual distribution was checked with the Kolmogorov–Smirnov test.

Table 5 summarises the first estimated MLR model, in which all input variables were used to predict the daily maximum air temperature. Although the t-tests showed that all input variables were significant in the model and its adjusted R^2 was 0.84, we detected multicollinearity based on the variance inflation factor (VIF) and the tolerance of the predictors (TOL). Note that in this table, the maximum VIF value and the minimum TOL value are 30.664 and 0.033, respectively. Furthermore, this model contains five factors with a VIF value greater than four (i.e., a TOL value less than 0.25).

To reduce multicollinearity, linking correlated features, several variables were iteratively removed. The estimated final MLR model is explained in Table 6, where the maximum VIF value and minimum TOL value are 1.907 and 0.524, respectively. Hence, with an adjusted $R^2 = 0.837$, similar to that obtained in the full MLR model, this new model does not have potential multicollinearity problems.

Fig. 10 shows the adjusted coefficient of determination and the RMSE for the 10 random runs of the validation process. It can be seen that these measures of the goodness of fit were similar for both the training and test datasets, with a mean of approximately 0.84 for the adjusted R^2 and 3.04 for the RMSE, in addition to correcting the multicollinearity detected previously. Table 7 summarises the different measures of the goodness of fit obtained in the cross-validation procedure using the eight input variables in Table 6. The mean MAPE obtained for the estimated MLR model was approximately 10.29%.

The methods analysed next were the KNN algorithms. More specifically, a comparison of the prediction accuracy of the KNN algorithm depending on the value of the parameter K was conducted. The distances used in the KNN algorithm were the Euclidean and Manhattan distances. Fig. 11 illustrates the behaviour of the algorithm for varying values of K from 1 to 50. Fig. 11(a) illustrates the variation in the MAE for different values of K, and Fig. 11(b) displays the corresponding RMSE.

Based on these measures, we can see that the KNN regression algorithm with the Manhattan distance outperforms the algorithm using the Euclidean distance. Moreover, the optimal value of K in this problem was $K = 9$, with the cross-validation procedure giving a mean of approximately 0.87 for the adjusted R^2, 2.6965 for the RMSE and 2.0829 for the MAE (see Table 8, in which the different measures of goodness of fit are summarised for the KNN models for $K = 3, 5, 9$). All of the models in this table outperform the best MLR model. In fact, the mean values of MAPE obtained for the KNN models of this table are less than the mean MAPE of the best MLR model, with a mean MAPE of approximately 8.87% for the best KNN model.

Following our analysis of the machine learning methods, RBF networks were also used to predict the maximum air temperature from the 11 predictive variables. To determine the number of neurons in the hidden layer, architectures with between 1 and 2^{11} neurons in the hidden layer were built. Fig. 12 illustrates the behaviour of the relative error $(1 - R^2)$ as the number of neurons increases. It can be seen that the shape of the network affects the results. The best neural network architectures were obtained for a number of neurons greater than or equal to 260. To avoid the risk of overfitting, and taking into account the empirical criteria recommended by different authors [49], the smallest of the optimal values was considered, choosing an architecture with 260 neurons in the hidden layer.

Fig. 13 explains the variation in the RMSE and MAPE for the 10 models generated with this architecture. It can be observed that no significant differences were obtained for the training and test datasets. Table 9 summarises several goodness-of-fit measures for this RBF network, giving a mean of approximately 0.83 for the adjusted R^2, 3.0918 for the RMSE, 2.4154 for the MAE, and 10.18% for the MAPE. Clearly,

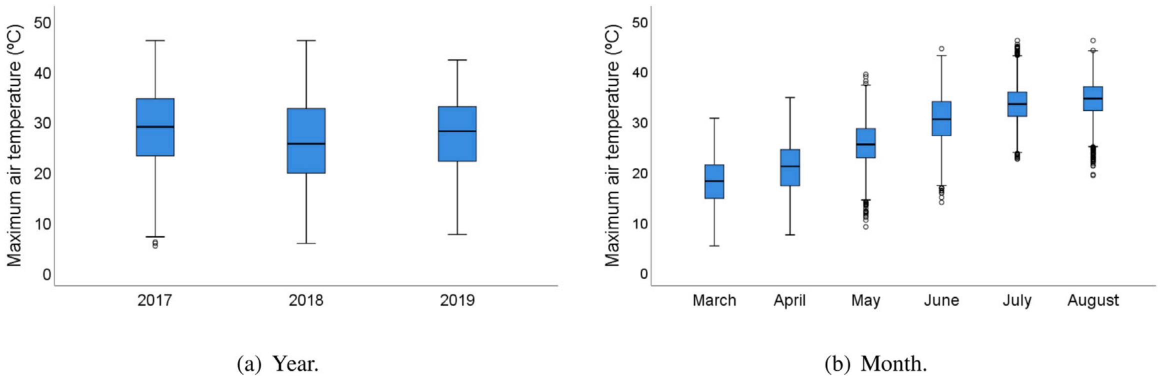


Fig. 9. Box plots of the daily maximum air temperatures.

Table 5
Full MLR model. Dependent variable: Maximum air temperature. Adjusted $R^2 = 0.840$.

	Unstandardised coefficients		Standardised coefficients			Collinearity statistics	
	β_i	Std. Error	β_i	t	Sig.	TOL	VIF
(Constant)	20.870	0.162		128.883	0.000		
Maximum relative humidity	0.025	0.003	0.058	7.750	0.000	0.102	9.840
Mean relative humidity	−0.097	0.006	−0.225	−16.990	0.000	0.033	30.664
Minimum relative humidity	−0.129	0.004	−0.278	−34.267	0.000	0.086	11.561
Mean radiation	1.496	0.070	0.068	21.486	0.000	0.567	1.763
Maximum wind speed	0.048	0.004	0.059	11.661	0.000	0.226	4.417
Minimum wind speed	0.495	0.097	0.013	5.112	0.000	0.898	1.113
Mean wind speed	−0.316	0.014	−0.115	−23.090	0.000	0.228	4.380
Total rainfall	0.021	0.005	0.012	4.260	0.000	0.744	1.345
Dominant wind direction	0.269	0.018	0.039	14.620	0.000	0.783	1.276
Altitude	−0.005	0.000	−0.177	−68.595	0.000	0.852	1.173
Month	2.506	0.012	0.568	201.726	0.000	0.719	1.391

Table 6
Estimated MLR model. Dependent variable: Maximum air temperature. Adjusted $R^2 = 0.837$.

	Unstandardised coefficients		Standardised coefficients			Collinearity statistics	
	β_i	Std. Error	β_i	t	Sig.	TOL	VIF
(Constant)	19.596	0.141		138.819	0.000		
Minimum relative humidity	−0.203	0.002	−0.437	−131.000	0.000	0.524	1.907
Mean radiation	1.465	0.069	0.067	21.317	0.000	0.596	1.677
Minimum wind speed	0.363	0.097	0.009	3.742	0.000	0.917	1.090
Mean wind speed	−0.168	0.007	−0.061	− 22.614	0.000	0.797	1.254
Total rainfall	0.026	0.005	0.014	5.117	0.000	0.748	1.336
Dominant wind direction	0.301	0.018	0.044	16.732	0.000	0.840	1.190
Altitude	−0.004	0.000	−0.167	− 64.838	0.000	0.883	1.133
Month	2.568	0.012	0.582	208.952	0.000	0.752	1.329

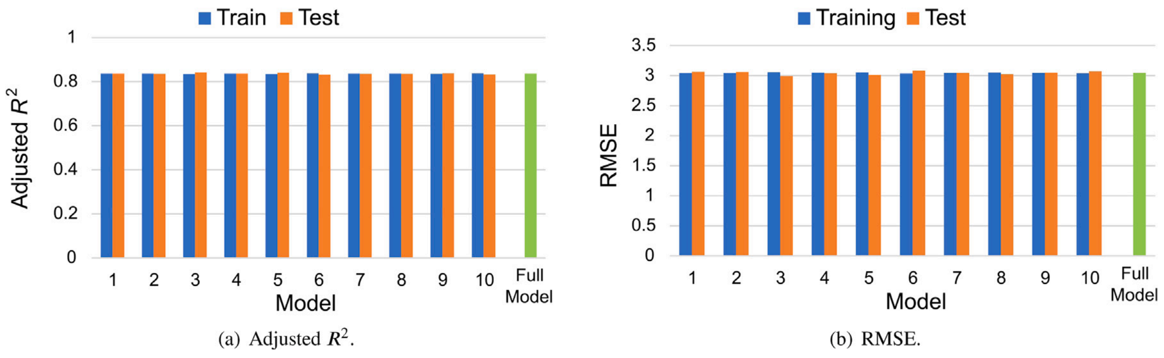
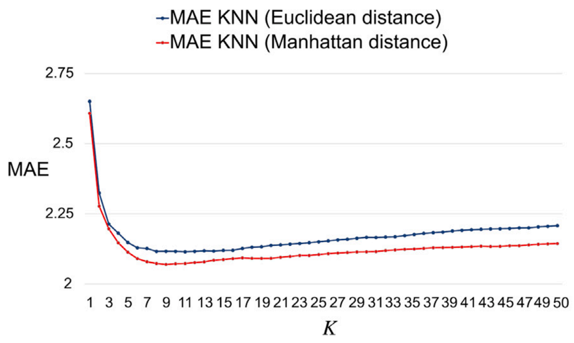


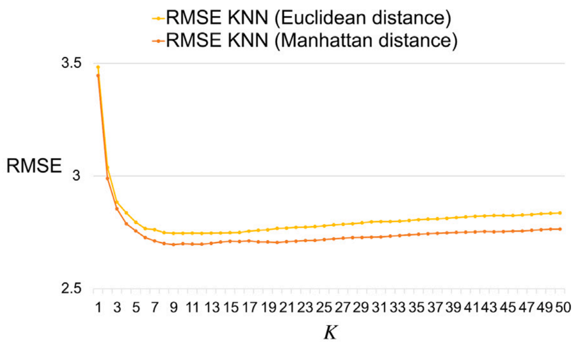
Fig. 10. Estimated MLR model, 10 models randomly generated with an 80:20 ratio by means of simple random sampling.

RBF networks fail to improve the performance of the KNN methods (see also Table 8).

Lastly, in the context of machine learning, SVR methods were studied. To obtain the best SVR model using RBF kernels, different



(a) MAE.



(b) RMSE.

Fig. 11. Comparison of KNN models, $1 \leq K \leq 50$.

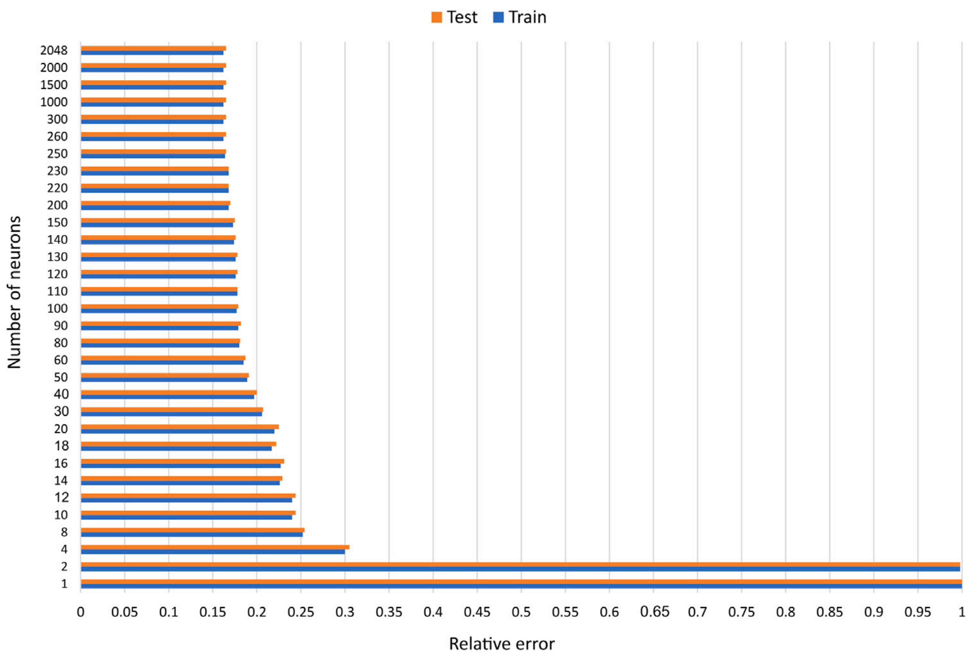
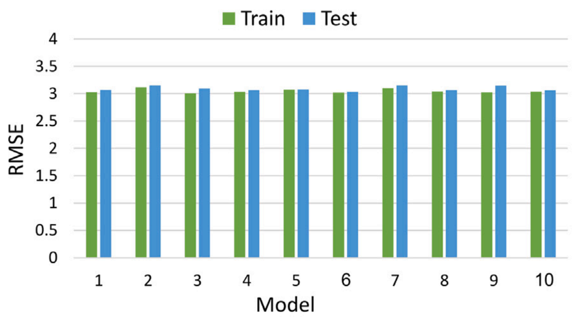
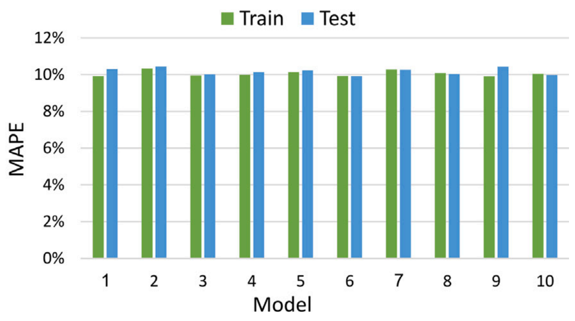


Fig. 12. Relative error of the RBF networks for varying number of neurons in the hidden layer.



(a) RMSE.



(b) MAPE.

Fig. 13. Estimated RBF networks, 10 models randomly generated with an 80:20 ratio by means of simple random sampling.

values of the parameters C, γ and ϵ were analysed (see Section 4.4). The parameter γ defines the influence of a single training example; the larger this parameter, the closer other examples must be to be affected. Hence, if γ is too large, there is a risk of overfitting, while a value that is too small may cause underfitting [50]. The penalty parameter C in the SVR model allows for a trade-off between the misclassification of training examples and the simplicity of the decision surface. A low value of C makes the decision surface smooth, while a high value aims at classifying all training examples correctly. Fig. 14(a) illustrates the behaviour of the RMSE with varying values of γ, where "auto" means that $\gamma = \frac{1}{d}$, and "scale" means $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$, where d is the number of input variables and $\mathrm{Var}(X)$ is the variance of the input dataset. It can be seen that regardless of the value of C, setting $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$ produces a competitive model.

Table 7
Estimated MLR model, 10 models randomly generated with an 80:20 ratio by means of simple random sampling. Statistical measures for test data.

Statistical measures	Median	Mean	CI (95%)
R^2	0.8366	0.8368	[0.8345, 0.8392]
Adj. R^2	0.8363	0.8365	[0.8342, 0.8389]
MAE	2.4104	2.4069	[2.3923, 2.4215]
MAPE	10.2962	10.2914	[10.2097, 10.3730]
RMSE	3.0475	3.0444	[3.0243, 3.0645]

Table 8
KNN model, 10 models randomly generated with an 80:20 ratio by means of simple random sampling. Statistical measures for test data.

Model	St. measures	Median	Mean	CI (95%)
$K = 3$ Euclidean distance	R^2	0.8508	0.8511	[0.8488, 0.8533]
	Adj. R^2	0.8505	0.8508	[0.8485, 0.8531]
	MAE	2.2314	2.2294	[2.2155, 2.2433]
	MAPE	9.4519	9.4298	[9.3700, 9.4897]
	RMSE	2.9064	2.9085	[2.8905, 2.9265]
$K = 3$ Manhattan distance	R^2	0.8551	0.8546	[0.8523, 0.8569]
	Adj. R^2	0.8548	0.8543	[0.8520, 0.8566]
	MAE	2.2045	2.2037	[2.1900, 2.2175]
	MAPE	9.3560	9.3542	[9.3027, 9.4057]
	RMSE	2.8722	2.8741	[2.8572, 2.8911]
$K = 5$ Euclidean distance	R^2	0.8619	0.8616	[0.8602, 0.8631]
	Adj. R^2	0.8616	0.8614	[0.8660, 0.8628]
	MAE	2.1623	2.1628	[2.1515, 2.1741]
	MAPE	9.1597	9.1633	[9.1269, 9.1997]
	RMSE	2.8002	2.8036	[2.7930, 2.8142]
$K = 5$ Manhattan distance	R^2	0.8662	0.8662	[0.8642, 0.8682]
	Adj. R^2	0.8659	0.8659	[0.8640, 0.8679]
	MAE	2.1257	2.1239	[2.1092, 2.1385]
	MAPE	9.0410	9.0251	[8.9705, 9.0796]
	RMSE	2.7581	2.7568	[2.7396, 2.7740]
$K = 9$ Euclidean distance	R^2	0.8669	0.8670	[0.8649, 0.8691]
	Adj. R^2	0.8666	0.8667	[0.8647, 0.8688]
	MAE	2.1296	2.1287	[2.1143, 2.1431]
	MAPE	9.0217	9.0283	[8.9716, 9.0850]
	RMSE	2.7495	2.7486	[2.7314, 2.7658]
$K = 9$ Manhattan distance	R^2	0.8721	0.8720	[0.8702, 0.8738]
	Adj. R^2	0.8718	0.8717	[0.8699, 0.8735]
	MAE	2.0797	2.0829	[2.0693, 2.0964]
	MAPE	8.8745	8.8668	[8.8065, 8.9271]
	RMSE	2.6951	2.6965	[2.6810, 2.7121]

Table 9
RBF network, 10 models randomly generated with an 80:20 ratio by means of simple random sampling. Statistical measures for test data.

Statistical measures	Median	Mean	CI (95%)
R^2	0.8327	0.8310	[0.8269, 0.8351]
Adj. R^2	0.8324	0.8306	[0.8266, 0.8347]
MAE	2.4032	2.4154	[2.3864, 2.4444]
MAPE	10.1888	10.1774	[10.0424, 10.3125]
RMSE	3.0733	3.0918	[3.0602, 3.1234]

The parameter ϵ specifies the epsilon tube, within which no penalty is allocated in the training loss function to points predicted within a distance of ϵ from the actual value. Fig. 14(b) shows the obtained RMSE for both the training and test datasets, for $\epsilon = 0.1$ and varying C. Fig. 14(c) displays the behaviour of the RMSE for varying ϵ and the optimum values of C and γ.

An exhaustive analysis of these parameters shows that an optimal model is achieved with $C = 20$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$ and $\epsilon = 0.01$, where the cross-validation procedure gives a mean of approximately 0.88 for the adjusted R^2, 2.6146 for the RMSE, 2.1687 for the MAE and 8.59% for the MAPE (see Table 10). For the problem studied here, this SVR model outperformed all of the machine learning models previously considered.

Table 10
RBF SVR model, 10 models randomly generated with an 80:20 ratio by means of simple random sampling, $C = 20$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$.

	Statistical measures	Median	Mean	CI (95%)
$\epsilon = 0.01$	R^2	0.8789	0.8797	[0.8782, 0.8811]
	Adj. R^2	0.8786	0.8794	[0.8780, 0.8809]
	MAE	2.1672	2.1687	[2.1576, 2.1798]
	MAPE	8.5798	8.5923	[8.5163, 8.6683]
	RMSE	2.6164	2.6146	[2.5975, 2.6317]
$\epsilon = 0.1$	R^2	0.8734	0.8733	[0.8720, 0.8747]
	Adj. R^2	0.8732	0.8732	[0.8718, 0.8744]
	MAE	2.1331	2.1314	[2.1225, 2.1403]
	MAPE	8.9587	8.9610	[8.9011, 9.0209]
	RMSE	2.6815	2.6824	[2.6697, 2.6952]

Table 11
NLM-RK model, 10 models randomly generated with an 80:20 ratio by means of simple random sampling. Statistical measures for the test data.

Statistical measures	Median	Mean	CI (95%)
R^2	0.8683	0.8687	[0.8664, 0.8710]
Adj. R^2	0.8680	0.8684	[0.8661, 0.8707]
MAE	2.0768	2.0700	[2.0560, 2.0840]
MAPE	8.6950	8.7238	[8.6206, 8.8270]
RMSE	2.6728	2.6640	[2.6465, 2.6815]

7.2. Analysis of the behaviour of NLM-RK models and comparison with machine learning models

Parallel nonlinear models using radial kernels on local mesh support were tested based on the complexity chosen in the methodology (see Algorithm 1). Fig. 15 shows the performance of these models depending on the selected complexity.

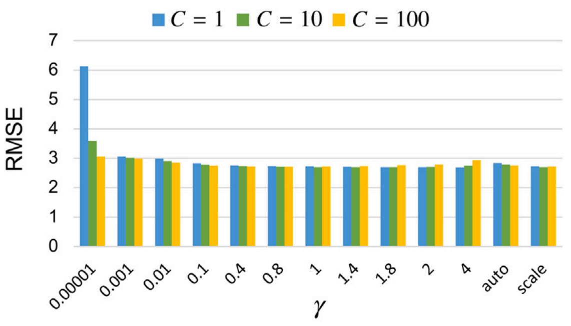
The behaviour is consistent with that reported in [24], with the training errors being smaller than the testing errors. Specifically, for our climatological problem, the smallest differences between the training and test datasets were obtained by setting the complexity to about 20.

To avoid overfitting and underfitting, a complexity of between 20 and 30 can be selected in the model; in the testing phase, the best results were obtained with a complexity of 30. Table 11 summarises different goodness-of-fit measures for this complexity, obtained from the same cross-validation procedure as in the previous section. The adjusted R^2, MAE, MAPE, and RMSE values were obtained for the 10 models that were randomly generated using 80% of the sample for training and the remainder for testing. Similar results were obtained for the 10 models, which gave the following mean values after the validation procedure: an adjusted R^2 of 0.8684, a MAPE of 8.72%, a MAE of 2.07 and an RMSE of 2.664.

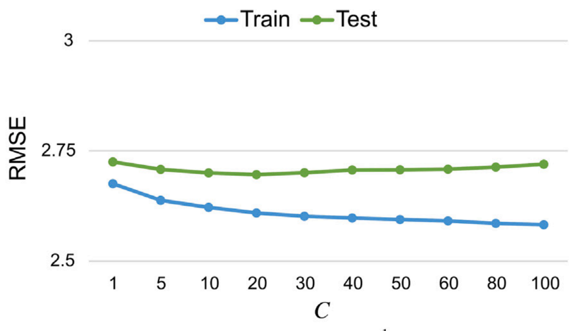
Fig. 16 compares the performance of this model with that obtained when each province is considered separately. To create this last model, we split the dataset into eight smaller sets containing the instances relative to the corresponding province. For each of these datasets, the cross-validation process was carried out and the goodness-of-fit measures were computed. No significant differences were obtained, and the same mean values for the goodness-of-fit measures were obtained as in the previous model (the global model in Fig. 16).

In accordance with the results presented in Section 7.1, the worst models for predicting the maximum air temperature were the MLR models and the RBF networks, with mean values for the MAPE of 10.29% and 10.18%, respectively (see Tables 7 and 9 in Section 7.1).

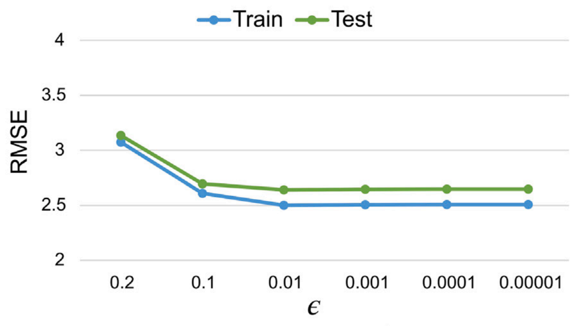
The NLM-RK model gave similar performance to the best SVR model (with $C = 20$, $\epsilon = 0.01$ and $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$), which achieved values for the RMSE of 2.66 and 2.61, respectively. In addition, the NLM-RK model outperformed the KNN models and the rest of the machine learning models analysed in Section 7.1. In particular, as can be seen from Fig. 17, the RBF SVR model achieved the smallest RMSE and MAPE, while the NLM-RK model achieved the smallest MAE with a mean



(a) RMSE, $\epsilon = 0.1$, $C = 1, 10, 100$.

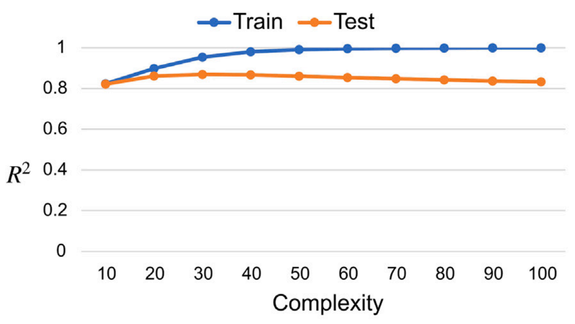


(b) RMSE, $\epsilon = 0.1$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$.

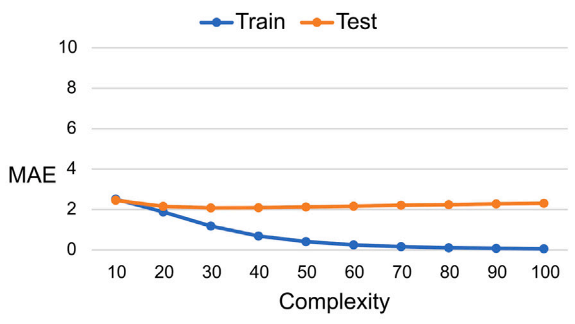


(c) RMSE, $C = 20$, $\gamma = \frac{1}{d\,\mathrm{Var}(X)}$.

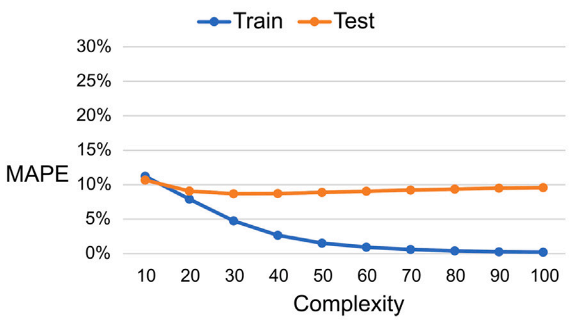
Fig. 14. RMSE for the RBF SVR models.



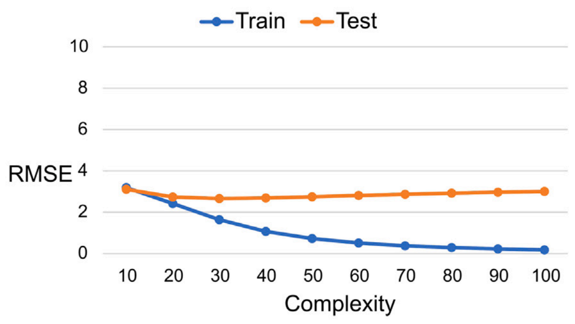
(a) R^2.



(b) MAE.



(c) MAPE.

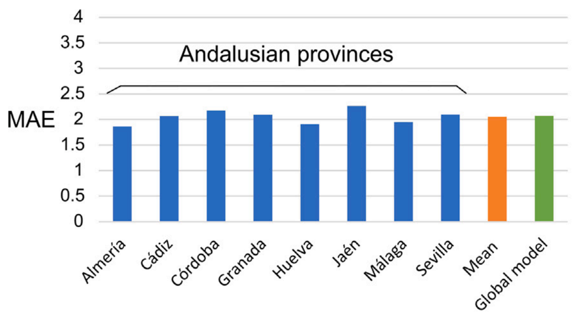


(d) RMSE.

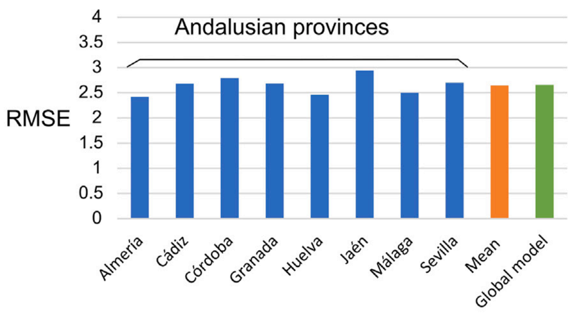
Fig. 15. NLM-RK model generated with an 80:20 ratio by means of simple random sampling. Goodness-of-fit measures for the training and test datasets.

of 2.07. Unlike neural networks, these techniques are deterministic algorithms, in the sense that for a specific selection of parameters or hyperparameters, the estimated model depends only on the experimental dataset; that is, there are no differences between two runs of the method.

Regarding the computational cost, the best parallel NLM-RK algorithm required approximately 9.97 min to estimate all points of the dataset. The SVR model took approximately 6.24 min, using the best set of parameters. However, the determination of the optimal parameters for SVR models is a more complex and arduous task than finding the optimal value for the complexity of the NLM-RK algorithm, meaning that a wrong choice of C, γ or ϵ can greatly reduce the performance of the SVR model (see Fig. 14(b) and Table 10 in Section 7.1), whereas a competitive value of the complexity in the NLM-RK algorithm can be



(a) MAE.



(b) RMSE.

Fig. 16. NLM-RK model, 10 models randomly generated with an 80:20 ratio by means of simple random sampling. Statistical measures for the test data for provinces.

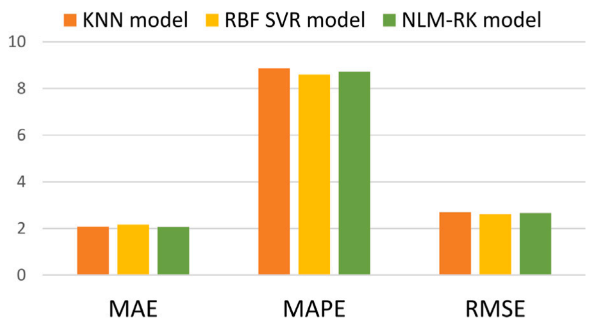


Fig. 17. Comparison of the best models.

easily obtained by performing a few initial iterations of the algorithm. In fact, this preprocessing time is practically negligible: for example, performing 240 initial iterations with the above-selected 80:20 ratio takes only about 0.18 s in parallel mode. However, the number of experiments needed to obtain an optimal SVR model gives rise to a high computational cost, and a total running time that is greater than for the NLM-RK algorithm.

8. Conclusions

In this work, a parallel RBF-based methodology (NLM-RK model) has been proposed to overcome the usage limitations imposed by the computational cost of some current FEM-based methodologies [16–18, 22,23]. The parallel NLM-RK methodology is applicable to the modelling of real-world objects or systems by means of a numerical method that is not subject to the problem at hand. Its computational cost does not depend on the complexity, that is, on the number of elements in each dimension of the domain. Nevertheless, the model may obtain different levels of prediction accuracy depending on the complexity and the problem dataset used. For each experimental dataset, we can obtain a competitive value for the complexity by performing a few initial iterations of the algorithm. In future work, we will investigate the development of a heuristic framework to derive suitable values for the complexity parameter in real time.

This parallel approach, based on the methodology proposed in [24], not only reduces the computational complexity compared to previous methodologies but also allows the modelling of problems that are larger than those considered in the abovementioned papers. The proposed parallel algorithms use two different strategies. The first uses MPI for parallel programming, while the second uses a hybrid paradigm in which MPI and OpenMP are combined. The best results were obtained when a hybrid MPI+OpenMP model was selected, using a dynamic scheduling strategy with a chunk size equal to one or a static strategy. The performance of the pure MPI model was comparable to that of the best hybrid model. Both parallel programming models achieved good utilisation of the parallel resources, with an efficiency close to one. The experimental problem treated here involves an attempt to model the daily maximum air temperature in Andalusia, an autonomous community of Spain. Data were obtained from the Agroclimatic Station Network of the Andalusian Phytosanitary Alert and Information Network, and the final dataset had 27,988 instances and 11 input attributes.

The modelling performance of the proposed parallel NLM-RK algorithm was compared with the performance of four well-known machine learning techniques. MLR, KNN, SVR algorithms and RBF networks were implemented and evaluated to select the best machine learning models. The cross-validation procedure showed that the MLR models and RBF networks were the worst techniques in terms of modelling the maximum air temperature, as they achieved a MAPE slightly greater than 10%. However, the NLM-RK model, with a MAPE of approximately 8.72%, a MAE of 2.07, and an RMSE of 2.664, obtained similar performance to the best SVR model, and outperformed both KNN and the other SVR models. Since determining the optimal complexity for the NLM-RK methodology is much less complex and costly than searching for the optimal parameters of the SVR model, our parallel NLM-RK model is shown to be a competitive method, in which the computing time is drastically reduced as the number of processing units increases. An extension that could be explored in the near future is the use of more powerful multicomputers and graphics processing units (GPUs). The design of new parallel approaches for the NLM-RK model will be studied using these architectures.

CRediT authorship contribution statement

Violeta Migallón: Conceptualization, Methodology, Data curation, Software, Analysis, Validation, Writing – original draft, Writing – review & editing, Supervision. **Francisco J. Navarro-González:** Conceptualization, Methodology, Software, Validation. **Héctor Penadés:** Data curation, Software, Analysis, Writing – original draft. **José Penadés:** Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Supervision. **Yolanda Villacampa:** Conceptualization, Methodology, Software, Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

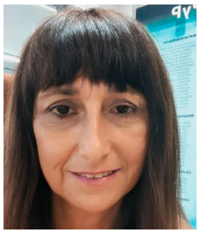
Acknowledgements

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), and by the University of Alicante.

References

[1] S.L. Barbour, J. Krahn, Numerical modelling - prediction or process? in: Geotechnical News, 2004, pp. 44–52.
[2] W.W. Hsieh, Machine Learning Methods in the Environmental Sciences: Neural Networks and Kernels, Cambridge University Press, 2009.
[3] R.H. Gallagher, Finite Element Analysis: Fundamentals, Prentice-Hall, 1975.
[4] M. Hatami, Weighted Residual Methods: Principles, Modifications and Applications, Academic Press, 2017.
[5] R. Shamshiri, W.I.W. Ismail, Implementation of Galerkin's method and modal analysis for unforced vibration response of a tractor suspension model, Res. J. Appl. Sci. Eng. Technol. 7 (2014) 49–55, http://dx.doi.org/10.19026/rjaset.7.219.
[6] 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 (2015) 211–233, http://dx.doi.org/10.1515/jnma-2015-0014.
[7] J. Chan, R.J. Hewett, T. Warburton, Weight-adjusted discontinuous Galerkin methods: Wave propagation in heterogeneous media, SIAM J. Sci. Comput. 36 (2017) A2935–A2961, http://dx.doi.org/10.1137/16M1089186.
[8] J. Chan, Weight-adjusted discontinuous Galerkin methods: Matrix-valued weights and elastic wave propagation in heterogeneous media, Internat. J. Numer. Methods Engrg. 113 (2018) 1779–1809, http://dx.doi.org/10.1002/nme.5720.
[9] S. Marras, J.F. Kelly, M. Moragues, A. Müller, M.A. Kopera, M. Vázquez, F.X. Giraldo, G. Houzeaux, O. Jorba, A review of element-based Galerkin methods for numerical weather prediction: Finite elements, spectral elements, and discontinuous Galerkin, Arch. Comput. Methods Eng. 23 (2016) 673–722, http://dx.doi.org/10.1007/s11831-015-9152-1.
[10] N. Beisiegel, S. Vater, J. Behrens, F. Dias, An adaptive discontinuous Galerkin method for the simulation of hurricane storm surge, Ocean Dyn. 70 (2020) 641–666, http://dx.doi.org/10.1007/s10236-020-01352-w.
[11] A. Constantin, R.S. Johnson, On the modelling of large-scale atmospheric flow, J. Differ. Equ. 285 (2021) 751–798, http://dx.doi.org/10.1016/j.jde.2021.03.019.
[12] C. Qin, J. Tian, X. Yang, K. Liu, G. Yan, J. Feng, Y. Lv, M. Xu, Galerkin-based meshless methods for photon transport in the biological tissue, Opt. Express 16 (2008) 20317–20333, http://dx.doi.org/10.1364/OE.16.020317.
[13] J. Mathault, H. Landari, F. Tessier, P. Fortier, A. Miled, Biological modeling challenges in a multiphysics approach, in: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems, MWSCAS, 2017, pp. 88–91, http://dx.doi.org/10.1109/MWSCAS.2017.8052867.
[14] M. Dehghan, N. Narimani, An element-free Galerkin meshless method for simulating the behavior of cancer cell invasion of surrounding tissue, Appl. Math. Model. 59 (2018) 500–513, http://dx.doi.org/10.1016/j.apm.2018.01.034.
[15] G. Ruocco, P. Caccavale, M.V. De Bonis, Chapter 18 - a predictive oncology framework-modeling tumor proliferation using a FEM platform, in: S.C. Kundu, R.L. Reis (Eds.), Biomaterials for 3D Tumor Modeling, Materials Today, Elsevier, 2020, pp. 427–450, http://dx.doi.org/10.1016/B978-0-12-818128-7.00018-6.
[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, http://dx.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, http://dx.doi.org/10.1016/j.advengsoft.2013.04.015.
[18] 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 (2016) 100–113, http://dx.doi.org/10.2495/CMEM-V4-N2-100-113.
[19] 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, http://dx.doi.org/10.1016/j.scitotenv.2017.11.156.
[20] L. Aragonés, J.I. 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, http://dx.doi.org/10.1016/j.scitotenv.2019.01.017.
[21] A. Palazón, I. López, L. Aragonés, Y. Villacampa, F.J. Navarro-González, Modelling of Escherichia coli concentrations in bathing water at microtidal coasts, Sci. Total Environ. 593-594 (2017) 173–181, http://dx.doi.org/10.1016/j.scitotenv.2017.03.161.
[22] V. Migallón, F.J. Navarro-González, J. Penadés, Y. Villacampa, Parallel approach of a Galerkin-based methodology for predicting the compressive strength of the lightweight aggregate concrete, Constr Build Mater. 219 (2019) 56–68, http://dx.doi.org/10.1016/j.conbuildmat.2019.05.160.
[23] 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, Constr. Build Mater. 189 (2018) 1173–1183, http://dx.doi.org/10.1016/j.conbuildmat.2018.09.096.
[24] F.J. Navarro-González, Y. Villacampa, M. Cortés-Molina, S. Ivorra, Numerical non-linear modelling algorithm using radial kernels on local mesh support, Mathematics 8 (2020) http://dx.doi.org/10.3390/math8091600.
[25] M.D. Buhmann, Radial basis functions, Acta Numer. 9 (2000) 1–38, http://dx.doi.org/10.1017/S0962492900000015.
[26] D. Dua, C. Graff, UCI machine learning repository, 2017, http://archive.ics.uci.edu/ml.
[27] S. Zhong, K. Zhang, M. Bagheri, J.G. Burken, A. Gu, B. Li, X. Ma, B.L. Marrone, Z.J. Ren, J. Schrier, W. Shi, H. Tan, T. Wang, X. Wang, B.M. Wong, X. Xiao, X. Yu, J.-J. Zhu, H. Zhang, Machine learning: New ideas and tools in environmental science and engineering, Environ. Sci. Technol. 55 (2021) 12741–12754, http://dx.doi.org/10.1021/acs.est.1c01339.
[28] J. Gómez-Zotano, J. Alcántara-Manzanares, E. Martínez-Ibarra, J.A. Olmedo-Cobo, Applying the technique of image classification to climate science: the case of Andalusia (Spain), Geogr. Res. 54 (2016) 461–470, http://dx.doi.org/10.1111/1745-5871.12180.
[29] M. Azadkia, Optimal choice of k for k-nearest neighbor regression, 2020, arXiv:1909.05495.
[30] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, 2018, arXiv:1811.03378.
[31] C.C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.
[32] H. Faris, I. Aljarah, S. Mirjalili, Chapter 28 - evolving radial basis function networks using moth-flame optimizer, in: P. Samui, S. Sekhar, V.E. Balas (Eds.), HAndbook of Neural Computation, Academic Press, 2017, pp. 537–550, http://dx.doi.org/10.1016/B978-0-12-811318-9.00028-4.
[33] M. Awad, R. Khanna, Efficient learning machines: Theories, concepts, and applications for engineers and system designers, apress media, 2015, http://dx.doi.org/10.1007/978-1-4302-5990-9.
[34] F. Zhang, L.J. O'Donnell, Chapter 7 - support vector regression, in: A. Mechelli, S. Vieira (Eds.), Machine Learning, Academic Press, 2020, pp. 123–140, http://dx.doi.org/10.1016/B978-0-12-815739-8.00007-9.
[35] S. Balasundaram, M. Tanveer, On Lagrangian twin support vector regression, Neural Comput. Appl. 22 (2013) 257–267, http://dx.doi.org/10.1007/s00521-012-0971-9.
[36] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (2004) 199–222, http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.
[37] W. Gani, H. Taleb, M. Limam, Support vector regression based residual control charts, J. Appl. Stat. 37 (2010) 309–324, http://dx.doi.org/10.1080/02664760903002667.
[38] M.D. Schluchter, Mean Square Error, American Cancer Society, 2005, http://dx.doi.org/10.1002/0470011815.b2a15087.
[39] T. Chai, R.R. Draxler, Root mean square error (RMSE) or mean absolute error (MAE)? – arguments against avoiding RMSE in the literature, Geosci. Model Dev. 7 (2014) 1247–1250, http://dx.doi.org/10.5194/gmd-7-1247-2014.
[40] C.J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Clim. Res. 30 (2005) 79–82, http://www.jstor.org/stable/24869236.
[41] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, MPI: The Complete Reference, second ed., The MIT Press, Cambridge, MA, 1998.
[42] OpenMP, OpenMP official site, 2012, http://www.openmp.org. (Accessed on October, 2021).
[43] G. Hager, G. Jost, R. Rabenseifner, Communication characteristics and hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes, in: Proceedings of Cray User Group, 2009.
[44] A. Pal, A. Agarwala, S. Raha, B. Bhattacharya, Performance metrics in a hybrid MPI–OpenMP based molecular dynamics simulation with short-range interactions, J. Parallel Distrib. Comput. 74 (2014) 2203–2214, http://dx.doi.org/10.1016/j.jpdc.2013.12.008.
[45] M.J. Chorley, D.W. Walker, Performance analysis of a hybrid MPI/OpenMP application on multi-core clusters, J. Comput. Sci. 1 (2010) 168–174, http://dx.doi.org/10.1016/j.jocs.2010.05.001.
[46] INTERTWinE-Consortium, Best practice guide to hybrid MPI OpenMP programming, 2017, http://www.intertwine-project.eu/sites/default/files/images/INTERTWinE_Best_Practice_Guide_MPI%2BOpenMP_1.2.pdf.
[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[48] IBM Corp, IBM SPSS statistics for windows, 2019, https://www.ibm.com/uk-en/analytics/spss-statistics-software.
[49] K.G. Sheela, S.N. Deepa, Review on methods to fix number of hidden neurons in neural networks, Math. Probl. Eng. 2013 (2013) 425740, http://dx.doi.org/10.1155/2013/425740.

[50] M. Yoshida, T. Imanishi, H. Nishi, Feature extraction and background information detection method using power demand, in: 2017 IEEE 26th International Symposium on Industrial Electronics, ISIE, 2017, pp. 1336–1341, http://dx.doi.org/10.1109/ISIE.2017.8001439.



Violeta Migallón is a Full Professor of the Computing Science and Artificial Intelligence Department and a research member of the University Institute for Computing Research, both at the University of Alicante (Spain). She received her Ph.D. degree in Computer Science in 1993. She is a member of the "High-Performance Computing and Parallelism" research group at the University of Alicante. Her main research interests are parallel computing for solving linear and nonlinear systems, and predictive modelling in engineering including artificial intelligence techniques.

Francisco J. Navarro-González obtained his degree in Physics from the University of Valencia. He received his Ph.D. degree in Mathematics in 2011, with a doctoral thesis on the generation of models from experimental data using techniques based on the finite element method. He obtained a master's degree in Economic Research in 2017. His research fields concern the development of algorithms for system modelling and numerical solution of algebraic problems and differential equations. Adjunct Professor at the University of Alicante (Spain) since 2008 in the Department of Applied Mathematics and Associate Professor since 2021.

Héctor Penadés obtained a bachelor's degree in Industrial Electronics and Automation Engineering in 2020 at the University Carlos III of Madrid, Spain, and a master's degree in Automation and Robotics in 2021 at the University of Alicante, Spain. He is a member of the research team of QuixMind, a technology-based company at the University of Alicante. His research interests focus on computer vision, machine learning, and sensor technology and applications.

José Penadés is a Full Professor of the Computing Science and Artificial Intelligence Department and a research member of the University Institute for Computing Research, both at the University of Alicante (Spain). He received his Ph.D. degree in Computer Science in 1993. His main research interests are parallel algorithms for heterogeneous platforms for solving linear and nonlinear systems, high-level interface design, which simplifies programming in parallel environments, and predictive analysis methodologies in engineering including artificial intelligence techniques.

Yolanda Villacampa studied at the University of Valencia, where she received her Ph.D. in Mathematics. She is a Full Professor of Applied Mathematics at the University of Alicante (Spain). She heads the research group "Mathematical Modelling of Systems" and her main research interest is the development of computational algorithms for the modelling and representation of complex systems and differential equations. She has participated in numerous research projects, being the main researcher in ten of them, and has collaborated with the Wessex Institute of Technology (UK) and the University of Ancona (Italy).