Scalable Weak Constraint Gaussian Processes

Rossella Arcucci[1], Douglas McIlwraith[1], and Yi-Ke Guo[1]

Data Science Institute, Imperial College London, UK

Abstract. A Weak Constraint Gaussian Process (WCGP) model is presented to integrate noisy inputs into the classical Gaussian Process predictive distribution. This follows a Data Assimilation approach i.e. by considering information provided by observed values of a noisy input in a time window. Due to the increased number of states processed from real applications and the time complexity of GP algorithms, the problem mandates a solution in a high performance computing environment. In this paper, parallelism is explored by defining the parallel WCGP model based on domain decomposition. Both a mathematical formulation of the model and a parallel algorithm are provided. We prove that the parallel implementation preserves the accuracy of the sequential one. The algorithm's scalability is further proved to be $\mathcal{O}(p^2)$ where p is the number of processors.

Keywords: Gaussian Processes · Data Assimilation · Domain Decomposition · Parallel Algorithms · Big Data

1 Introduction and Motivations

Gaussian processes have been widely used since the 1970's in the fields of geostatistics and meteorology. In geostatistics, prediction with Gaussian processes is termed Kriging, named after the South African mining engineer D. G. Krige by Matheron [10]. Naturally in spatial statistics the inputs to the process are the two or three space dimensions, however, Over the past decade or so, there has been much work on Gaussian processes in the machine learning community, typically over higher dimensionality spaces.

GPs have had substantial impact in technologies including geostatistics [3] and feature reduction [12]. Current applications are in diverse fields such as geophysics, medical imaging, multi-sensor fusion [13, 19, 21] and sensor placement [8]. The latter of these is well suited application for the GP model we propose in this paper. For example, this could allow the study of optimal sensor placement for collecting air pollution data in big cities.

The main limitations of traditional GP regression are its sensitivity to noisy input and the computational complexity. The contribution of this paper is a new GP algorithm that overcomes these limitations simultaneously. By considering the inputs to the GP as noisy observations and reformulating the GP, we create a new model in which inputs are assimilated to estimate the true values of inputs. We further demonstrate that our method is highly suitable for parallel processing, both formally and with experimental results.

2 Background and contribution

In [7], the authors expand the Gaussian process around the input mean (delta method), assuming the random input is normally distributed and they derive a new process whose covariance function accounts for the randomness of the input. In [11] the input noise variances are inferred from the data as extra hyperparameters. Instead, we develop a Weak Constraint Gaussian Process (WCGP) model to be used to improve the accuracy of the classical Gaussian process predictive distribution [15]. Noisy inputs are integrated into the Gaussian process through a data assimilation approach - i.e. by considering information provided by observed values of the noisy input. As the "assimilated observations" are not verified exactly [1], we may consider this a weak constraint over the inputs. The resulting model (which we call WCGP) is still a GP model with modified mean and variance - as we will demonstrate in section 3. The number of processed data points for this model is increased with respect the classical GP and we show the GP time complexity is $\mathcal{O}(N^3)$.

An approach to reduce the time complexity is to introduce approximation methods. In [18] the covariance matrix is approximated by the Nystrom extension of a smaller covariance matrix evaluated on M training observations ($M << N$). Approximation methods help to reduce the computation cost from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ and make makes running less expensive, but parameters must still be selected a-priori, and, consequently, important sensitivities may be missed [2].

Due to the large number of states required in real applications plus the time complexity of GP algorithms, the problem mandates the solution in a high performance computing environment. In [22] a scalable Sparse Gaussian process (GP) regression [16] and Bayesian Gaussian process latent variable model (GPLVM) [17] are presented. This work represents the the first distributed inference algorithm which is able to process datasets with millions of points. However, even with sparse approximations it is inconceivable to apply GPs to training set sizes of data sets of size more than $\mathcal{O}(10^7)$. In [4] a distributed Gaussian processes model is introduced. Their key idea is to recursively distribute computations to independent computational units and, subsequently, recombine them to form an overall result. Local predictions are recombined by a parent node, which subsequently may play the role of an expert at the next level of the model architecture. Even if this approach allows us to face problems with large data sets, the interaction between the local and the parent node introduces a bottleneck which affects the efficiency of processing for datasets which may be considered "big data".

As claimed in [6], the partitioning problem (i.e, decomposability: to break the problem into small enough independent less complex subproblems) is a universal source of scalable parallelism. In [14] a domain decomposition approach for the classical GP or [20] applied to urban flows simulation are presented based on the definition of boundary conditions for the subproblems. While the introduced approach allows "big data" problems to be tackled, there is a subsequent loss to solution accuracy.

In this paper, we formally address the parallelism problem by defining the parallel Weak Constraint GP (WCGP) model based on the previously introduced domain decomposition approach. The accuracy of the proposed approach is proved by showing that the solution obtained by the parallel algorithm is the same as obtained by the sequential one. In particular, a parallel algorithm to be implemented on a distributed computing architecture is presented. Also, the algorithm's scalability is studied taking into account both the execution time (i.e. the time complexity) and the communication overhead given by the implementation of the algorithm on a parallel and distributed computing architecture. Finally, an upper bound on the achievable performance gain is provided, which turns out to be independent of the computing architecture utilised.

This paper is structured as follows. In Section 3 the Weak Constraint Gaussain Process model is described. Then, the Domain Decomposition based Weak Constraint Gaussian Process is introduced in Section 4. In this section we investigate the accuracy of the introduced method and a theorem demonstrating the conservation of accuracy is presented. In Section 5 the scalability of the resulting algorithm is discussed and experimental results are provided in Section 6. Conclusion and future work is summarised in Section 7.

3 Weak Contraint Gaussian Process

A spatial noisy input GP regression is formulated as follows: given a training data set $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, N\}$ of n pairs of noisy inputs x_i and noisy observations y_i , obtain the predictive distribution for the realization of a latent function at a test point x_* , denoted by $f_* = f(x_*)$. We assume that the latent function comes from a zero-mean Gaussian random field with a covariance function $k(\cdot, \cdot)$ on a domain $\Omega \subset \Re^{N \times N}$ and the noisy input x_i and observations y_i are given by

$$y_i = f(x_i + e_{xi}) + e_{yi} \tag{1}$$

where $e_{xi} = \mathcal{N}(0, \sigma_x^2)$ and $e_{yi} = \mathcal{N}(0, \sigma_y^2)$.

Gaussian Process: Denote

$$x = [x_1, x_2, \ldots, x_N]^T$$

and

$$y = [y_1, y_2, \ldots, y_N]^T.$$

The joint distribution of (f_*, y) is

$$(f_*, y) = \mathcal{N}\left(0, \begin{bmatrix} k_{x_*x_*} & k_{xx_*}^T \\ k_{xx_*} & \sigma_y^2 I + K_{xx} \end{bmatrix}\right) \tag{2}$$

where $k_{xx_*} = (k(x_1, x_*), \ldots, k(x_N, x_*))^T$ and K_{xx} is an $N \times N$ matrix $K_{xx} = \{k(x_i, x_j)\}_{i=1,\ldots,N; j=1,\ldots,N}$. By the conditional distribution for Gaussian variables, the predictive distribution of f_* given y is

$$P(f_*|y) = \mathcal{N}\left(k_{xx_*}^T A^{-1} y, k_{x_*x_*} - k_{xx_*}^T A^{-1} k_{xx_*}\right) \tag{3}$$

where

$$A = \sigma_y^2 I + K_{xx} \tag{4}$$

Data Assimilation: At each step i, $i = 1, \dots, N$ (see Figure 1), let $o = H(x)$ be the observations vector $o = \{o_i\}_{i=1,\dots,N}$ where H is a non-linear operator collecting the assimilated observations at each step. The aim of DA problem is to find an optimal tradeoff between the current estimate of the system state (background) defined in (3) and the available assimilated observations. Let R be a covariance matrix whose elements provide the estimate of the errors[1] on o, the assumption of input noise and the assimilation of it by a data assimilation approach (see Chapter 5 of [9]) introduce a corrective term

$$O = H^T R^{-1} H \tag{5}$$

to the output noise. Then the resulting Gaussian process is

$$P(f_*|y, o) = \mathcal{N}\left(k_{xx_*}^T \hat{A}^{-1} y, k_{x_*x_*} - k_{xx_*}^T \hat{A}^{-1} k_{xx_*}\right) \tag{6}$$

where

$$\hat{A} = A + O \tag{7}$$

with A and O defined in (4) and (5) respectively, and where we assume that each input dimension is independently corrupted by noise, thus R is diagonal:

$$R = \sigma_x^2 I \tag{8}$$

The predictive mean $k_{xx_*}^T \hat{A}^{-1} y$ gives the point prediction of $f(x)$ at location x_*, whose uncertainty is measured by the predictive variance $k_{x_*x_*} - k_{xx_*}^T \hat{A}^{-1} k_{xx_*}$. The point prediction given above is the best linear unbiased predictor in the following sense [14]. Consider all linear predictors

$$\mu(x_*) = u(x_*)^t y, \tag{9}$$

satisfying the unbiasedness requirement $E[\mu(x_*)] = 0$. We want to find the vector $u(x_*)$ which minimizes the mean squared prediction error $E[\mu(x_*) - f(x_*)]^2$. Since $E[\mu(x_*)] = 0$ and $E[f(x_*)] = 0$, the mean squared prediction error equals the error variance $var[\mu(x_*) - f(x_*)]$ and can be expressed as

$$\begin{aligned} \sigma(x_*) &= u(x_*)^t E(yy^t) u(x_*) - 2u(x_*)^t E(yf_*) + E(f_*^2) \\ &= u(x_*)^t (\sigma_y^2 I + K_{xx} + H^T R^{-1} H) u(x_*) - 2u(x_*)^t k_{xx_*} + k_{x_*x_*} \end{aligned} \tag{10}$$

Equation (10) is a quadratic form in $u(x_*)$. It is easy to see $\sigma(x_*)$ is minimized if and only if $u(x_*)$ is chosen to be $(\sigma_y^2 I + K_{xx} + H^T R^{-1} H)^{-1}$. Based on the above discussion, the mean of the predictive distribution in (6) or the best

[1] i.e., the assimilated observations are not verified exactly, it is a weak constraint over the inputs to the Gaussian Process

linear unbiased predictor can be obtained by solving the following minimization problem: for $x_* \in \Omega$, compute

$$\bar{u}(x_*) = argmin_{u(x_*)\in\Re^N} J(u(x_*)) \tag{11}$$

with

$$J(u(x_*), \sigma_y, K_{xx}, H, R, \Omega) = u(x_*)^t(\sigma_y^2 I + K_{xx} + H^T R^{-1} H)u(x_*) - 2u(x_*)^t k_{xx_*} \tag{12}$$

To compute the minimum of J in (12), the Jacobian of J has to satisfy the condition $\nabla J = 0$ which implies the solution of a linear system of the matrix $\hat{A}$. Due the ill conditioning of the matrix K_{xx}, the matrix $\hat{A}$ is ill conditioned as well, then is mandatory to introduce a preconditioning [15]. Since K_{xx} is symmetric and positive definite, it is possible to compute its Cholesky factorization $K_{xx} = VV^t$. Let be $\mu(\cdot)$ denote the condition number, by this way it is:

$$\mu(K_{xx}) = \mu(V)^{-\frac{1}{2}}$$

i.e., the Cholesky factorization of the covariance matrix K_{xx} mitigates the ill conditioning of the minimization problem. Let be $v = V^+ u$, where $+$ denotes the generalised inverse of the matrix V. After the preconditioning the minimization problem in (11) and (12) is written as:

$$\bar{v}(x_*) = argmin_{v(x_*)\in\Re^N} J(v(x_*)) \tag{13}$$

with

$$J(v(x_*), \sigma_y, V, H, R, \Omega) = v^t V^t (I + \sigma_y^2 I + H^T R^{-1} H) V v - 2v^t V^t k_{xx_*} \tag{14}$$

The exact treatment of this function would require the consideration of a distribution over Taylor expansions. Due the high computational load, several approximations are usually introduced in order to face this issue [11, 7]. Here we face the problem concerning the high computational cost by introducing a domain decomposition approach into the mathematical model. In next sections we provide a proof that the solution obtained by the decomposed problem we are introducing is the same solution of the sequential problem. Also, it is proved that the time complexity is reduced as well.

4 Domain Decomposition based Weak Constraint Gaussian Process

In this section a synthetic mathematical formalization of the weak constraint GP model based on a domain decomposition approach is presented. Local functions are introduced. These functions are defined on subdomains which constitute a partitioning $DD(\Omega)$ of the domain $\Omega \subset \Re^N$ with overlapping as described in Figure 2 and such that:

$$DD(\Omega) = \{\Omega_i\}_{i=1,\dots,p} \tag{15}$$

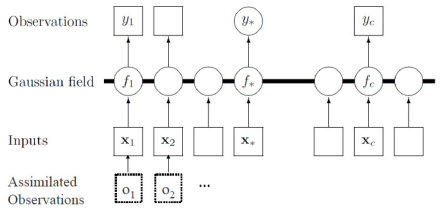

Fig. 1. *Graphical model (chain graph) for a WCGP for regression. Squares represent observed variables and circles represent unknowns. Dotted squares represent innovation with respect a classical GP [4].*

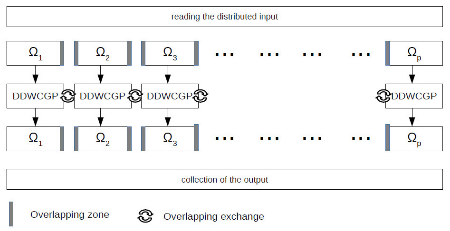

Fig. 2. *Domain Decomposition based Weak Constraint Gaussian Process*

with $\Omega_i \subset \Re^{r_i}$, $r_i \leq N$ and for $i = 1, \ldots, p$, it is such that $\Omega = \bigcup_{i=1}^{p} \Omega_i$ with $\Omega_i \cap \Omega_j = \Omega_{ij} \neq \emptyset$
The *local* WCGP function describes the local WCGP problem on a sub-domain Ω_i of the domain decomposition. It is obtained restricting the WCGP function J to the sobdomain Ω_i and by adding a *local* constraint onto the overlap region between adjacent domains Ω_i and Ω_j such that:

$$J_{\Omega_i}(v_i) = v_i^t V_i^t (I_i + \sigma_y^2 I_i + H_i^T R_i^{-1} H_i) V_i v_i - 2 v_i^t V_i k_{xx_*} + |v_{ij} - v_{ji}| \tag{16}$$

where v_i, V_i, I_i, R_i and H_i are restrictions on Ω_i of vectors and matrices in (12), and $v_{ij} = v_i/\Omega_{ij}$, $v_{ji} = v_j/\Omega_{ij}$, are restriction on $\Omega_{ij} = \Omega_i \cap \Omega_j$ of v_i and v_j respectively, $\forall j$ such that $\Omega_i \cap \Omega_j \neq \emptyset$.

4.1 Accuracy of the decomposition approach

In this section the accuracy of the introduced DDWCGP model is studied. An important point is to ensure that the introduction of a decomposition among the dataset does not introduce errors which affects the accuracy of the GP solution. Let $\bar{v}_i = argmin_{v_i} J_{\Omega_i}(v_i)$ denote the minimum of J_{Ω_i} and let $\widetilde{v}$ denote *extension vector* defined as the sum of the extensions of $\bar{v}_i$ $\forall i$, to the domain Ω:

$$\widetilde{v}_i = \begin{cases} \bar{v}_i & in\ \Omega_i \\ 0 & in\ \Omega - \Omega_i \end{cases} \quad and \quad \widetilde{v} = \sum_{i=1}^{p} \widetilde{v}_i \tag{17}$$

A theorem which ensures that the solution obtained by the parallel algorithm $\widetilde{v}$ is the same as obtained by the sequential one $\bar{v}$ is proved here. First, some preliminary observations are introduced in order to help the mathematical description of the restriction and the extension of the function among the subdomains and the global domain respectively. Let $\widetilde{J}_i$ denote the *extension function* of J_{Ω_i} to Ω:

$$\widetilde{J}_i = \begin{cases} J_{\Omega_i} & in\ \Omega_i \\ 0 & in\ \Omega - \Omega_i \end{cases} \tag{18}$$

Even if the functions $\widetilde{J}_i$ are defined on Ω, we consider here the subscript i to denote the starting subdomain. We can observe that

$$\sum_{i=1}^{p} \widetilde{J}_i = J. \tag{19}$$

Theorem 1. *Let $DD(\Omega)$ be a decomposition of the domain Ω. Also, let $\widetilde{v}$ defined in (17) and $\bar{v} = \bar{v}(x_*)$ defined in (13), it follows that:*

$$\widetilde{v} = \bar{v}.$$

Proof: *Let $\bar{v}_i$ be the minimum of J_{Ω_i} on Ω_i, it is*

$$\nabla J_{\Omega_i}[\bar{v}_i] = 0, \quad \forall i : \Omega_i \in DD(\Omega). \tag{20}$$

From (20) it follows that $\sum_i \nabla J_{\Omega_i}[\bar{v}_i] = 0$ which gives from the property of the gradient:

$$\nabla \sum_i J_{\Omega_i}[\bar{v}_i] = 0 \tag{21}$$

Let consider the extension vector and the extension function as defined in (17) and (18) respectively, from the (21) follows

$$\nabla \sum_i \widetilde{J}_i(\widetilde{v}_i) = 0 \tag{22}$$

which gives from the (19) and the second in (17):

$$\nabla J(\widetilde{v}) = 0 \tag{23}$$

then, from (23) follows that $\widetilde{v}$ is a stationary point for J. As $\bar{v}$ as defined in (13) is the global minimum, it follows that $J(\bar{v}) \leq J(\widetilde{v})$. We prove that $\bar{v} = \widetilde{v}$, then $J(\bar{v}) = J(\widetilde{v})$, by reduction to the absurd. Infact, by assuming

$$J(\bar{v}) < J(\widetilde{v}), \tag{24}$$

on Ω the (24) gives $J(\bar{v}(x_j)) < J(\widetilde{v}(x_j))$ for all $x_j \in \Omega$. In particular, let Ω_i be a subset of Ω, then $J(\bar{v}(x_j)) < J(\widetilde{v}(x_j))$ for all $x_j \in \Omega_i$. Hence, considering the restriction to Ω_i, $\forall i$ and by assuming the (24) we have

$$J_{\Omega_i}(\bar{v}/\Omega_i) < J_{\Omega_i}(\widetilde{v}/\Omega_i), \quad \forall i : \Omega_i \subset \Omega \tag{25}$$

which gives from (17):

$$J_{\Omega_i}(\bar{v}/\Omega_i) < J_{\Omega_i}(\bar{v}_i). \tag{26}$$

Equation (26) is an absurd. In fact, if $\bar{v}/\Omega_i \neq \bar{v}_i$, the (26) says that there exists a point such that the value of the function in that point is smaller than the values of the function in the point of global minimum. Then the theorem is proved.

This result ensures that $\bar{v}$ (the global minimum of J) can be obtained by patching together all the vectors $\bar{v}_i$ (global minimums of the operators J_{Ω_i}), i.e. by using the domain decomposition, the global minimum of the operator J can be obtained by patching together the minimums of the *local* functionals J_{Ω_i}. This result has important implications from the computational viewpoint as it will be explained in the next section.

5 The DD based WCGP algorithm (DDWCGP)

An algorithm to implement the minimization problem in (13) and (14) is presented as Algorithm 1. Hereafter it is assumed that the number of computing processors equals the number of subdomains p which constitutes the decomposition as described in (15).

Algorithm 1 $\mathcal{A}(\Omega_i, p)$: The DD based WCGP algorithm for each subdomain Ω_i of a partition of Ω in p subdomains

```
1: Input: ${(x_i, y_i)}_{i=0,...,N}$, ${o_i}_{i=0,...,N}$ and $x_*$
2: Input: $L$   ▷ define the number of points which constitutes the overlapping region
3: Define $\sigma_y$ , $R_i$ and $K_{xx}$   ▷ define the covariances
4: Compute $V \leftarrow Cholesky(K_{xx})$   ▷ compute the Cholesky factorization of the covariance matrix $K_{xx}$
5: Compute $D_i \leftarrow I_i + \sigma_y^2 I_i$
6: Define $H_i$   ▷ define the interpolation function
7: Compute $\hat{D}_i \leftarrow A_i + H_i^T R_i^{-1} H_i$
8: Compute $\hat{A}_i \leftarrow V_i^T \hat{D}_i V_i$   ▷ compute $\hat{A}$ as defined in (7)
9: Define the initial value of $u_i$
10: Compute $v_i = V_i^+ u_i$
11: While (convergence on $v_i$ is obtained)   ▷ start of the minimization steps
12:    Send and Receive $L$ overlapping values from the adjacent domains
13:    Compute $J_i \leftarrow J_i(v_i)$   ▷ Defined in (16)
14:    Compute new values for $v_i$
15: Compute $u_i = V_i v_i$
```

5.1 Complexity and Scalability

In this section we provide an estimate of the scalability of our approach through a study of time complexity. We also estimate the performance gain achievable by implementing the algorithm on a parallel and distributed computing architecture. First we provide the following definition of *Scaling factor*, which measures the performance gain in terms of time complexity reduction, as the ratio

$$S_p = \frac{\tau\,(\mathcal{A}(\Omega, 1))}{p\ \tau(\mathcal{A}(\Omega_i, p))}. \tag{27}$$

where $\mathcal{A}$ denotes the parallel algorithm, p is the size of the partition which constitutes the decomposition of Ω and $\tau(\mathcal{A})$ denotes the time complexity of the algorithm.

Theorem 2. **(Scaling factor estimation)** *The scaling factor of Algorithm 1 is such that*

$$S_p = p^2, \quad \forall p \tag{28}$$

Proof. The time complexity of a classical GP (i.e. when no decompositions are introduced) is $\mathcal{O}(N^3)$. *Then it is*

$$\tau(\mathcal{A}(\Omega, 1)) = N^3 \tag{29}$$

By introducing the domain decomposition of Ω as defined in Definition **??**, *and by considering a partition of size p, we are going to solve, by $\mathcal{A}$, p subproblems of size about N/p, i.e. the time complexity is related to the subproblems and it is*

$$\tau(\mathcal{A}(\Omega_i, p)) = \left(\frac{1}{p}N\right)^3 \tag{30}$$

From (27), (29), and (30) we have that

$$S_p = \frac{N^3}{p\left(\frac{1}{p}N\right)^3} = \frac{1}{\left(\frac{1}{p}\right)^2} = p^2$$

then (28) follows.

The parameter p in (27) denotes the rank of the partition which constitutes the decomposition. If we consider the DDWCGP model implemented in an algorithm on a parallel computing architecture of p processors (i.e. such that p denotes the rank of the partition as well as the number of processors involved, one subset Ω_i of the partition for each processor), we may evaluate the execution time of the algorithm running on the computing architecture and then estimate the *Measured Scaling factor*:

$$Measured\ S_p = \frac{T^1_{\mathcal{A}}(N)}{p\, T^p_{\mathcal{A}}(N)}. \tag{31}$$

where p is here the number of computing processors and $T^p_{\mathcal{A}}(N)$ denotes the execution time for solving a problem of size N with p processors.
The execution time $T^p_{\mathcal{A}}(N)$ can be mainly expressed as the sum of the time necessary for computation and the time necessary for communication. Then it is

$$T^p_{\mathcal{A}} = T^p_{flop}(N) + T^p_{comm}(N) \tag{32}$$

Regard $T^p_{flop}(N)$, by denoting with t_{flop} the time required by one floating point operation, it is

$$T^p_{flop}(n) = \tau(\mathcal{A}(\Omega, p))\ t_{flop} \tag{33}$$

In an ideal case, in absence of communication among the processors, by using (32) and (33) in (31) we clearly obtain $Measured\ S_p = S_p$. In reality, the estimation provided by (28) represents an upper bound of the performance gain we may obtain by implementing $\mathcal{A}$ on a parallel computing architecture due an overhead produced by the communication. The *Communication overhead* is the proportion of time the processors spend communicating with each other instead of computing:

$$T^p_{over}(N) = \frac{T^p_{comm}(N)}{T^p_{flop}(N)} \tag{34}$$

where N denotes the size of the problem. The following results hold:

Theorem 3 (Estimation of the Measured Scaling factor).

$$Measured\ S_p = C^p_{over}\ S_p \tag{35}$$

where C^p_{over} is a quantity which depends on the communication overhead defined in (34). Moreover, $C^p_{over} < 1$, which gives

$$Measured\ S_p < S_p. \tag{36}$$

Proof. *From (31) and (32), it is*

$$Measured\ S_p = \frac{T^1_{flop}(N)}{pT^p_{flop}(\frac{N}{p}) + pT^p_{comm}(\frac{N}{p})} \tag{37}$$

dividing by $pT^p_{flop}(\frac{N}{p})$, we have

$$Measured\ S_p = \frac{\frac{T^1_{flop}(N)}{pT^p_{flop}(\frac{N}{p})}}{1 + \frac{pT^p_{comm}(\frac{N}{p})}{pT^p_{flop}(\frac{N}{p})}}$$

also, by using (33), it can be written as

$$Measured\ S_p = \frac{1}{1 + \frac{T^p_{comm}(\frac{N}{p})}{T^p_{flop}(\frac{N}{p})}}\ \frac{\tau(\mathcal{A}(\Omega, p))\ t_{flop}}{p\ \tau(\mathcal{A}(\Omega, p))\ t_{flop}}$$

which, from (27), gives

$$Measured\ S_p = \frac{1}{1 + \frac{T^p_{comm}(\frac{N}{p})}{T^p_{flop}(\frac{N}{p})}}\ S_p \tag{38}$$

then, from (34), we have that

$$Measured\ S_p = \frac{1}{1 + T_{over}(\frac{N}{p})}\ S_p \tag{39}$$

and for

$$C^p_{over} = \frac{1}{1 + T_{over}(\frac{N}{p})} \tag{40}$$

then (35) follows. Moreover, as in (40) $T_{over}(\frac{N}{p}) > 0$, it follows that $C^p_{over} < 1$, which gives (36).

The communication overhead $T_{over}(\frac{N}{p})$ in (39) can be estimated by using the so called surface-to-volume ratio [5] (denoted as S/V) which is a measure of the amount of data exchange (proportional to surface area of domain) per unit operation (proportional to volume of domain). The goal is to minimize this ratio.

Lemma 1. *Concerning Algorithm 1, the surface-to-volume ratio equals to*

$$\frac{S}{V} = \frac{2Lp^3}{(N+2Lp)^3} \tag{41}$$

Proof. In Algorithm 1 the amount of data exchange (see Step 12) at each iteration is $S = 2L$. The amount of operations for each processor corresponds to the time complexity of the sub problem solved by the processor. Then it is $V = \left(\frac{1}{p}N + 2L\right)^3$. From these estimations of S and V, we have that (41) holds.

6 Experimental results

In this section we provide experimental results that demonstrate the applicability of our approach. In particular, we show experimentally that DDWCGP provides accurate and scalable results as proved in Theorem 1 and Theorem 2.

In order to point out the expected performance of the parallel Algorithm 1 in terms of scalability and execution time reduction, test cases for $N = 100$, $N = 1000$ and $N = 10000$ are been studied which correspond to problems of complexity $O(10^6)$, $O(10^9)$ and $O(10^{12})$. We show experimentally that DDWCGP provides accurate and scalable results over a dataset such that the kernel function is the Radial Basis Function kernel $k(x, x') = exp\left(-\frac{1}{2}\left(\frac{x-x'}{\lambda}\right)^2\right)$ with $\lambda = 0.1$, the y values (unbeknownst to our model) from our x are such that $y = sin(x)$ and the input covariance matrix is defined in (8) for $\sigma_x^2 = 0.5$: $R = 0.5\ I$ where I is the identity matrix.

Figure 3 shows results obtained for a number of computing processors (i.e. a number of subdomains which constitutes the decomposition described in (15)) equal to $p = 1$, $p = 2$ and $p = 4$. As shown in Figure 3, the result obtained by collecting results on subdomains matches perfectly the result obtained by the code for the test case without any decomposition. This constitutes an experimental validation of the accuracy result proved in Theorem 1.

Results in Table 1 show how the performance improves as the size of the problem increasing. This is explicitly confirmed in Figure 4 shows values of the ratio

$$R_p(N) = \frac{S_p}{Measured\ S_p} \tag{42}$$

this is due to the surface to volume ratio in (41) which is smaller for bigger domains. Namely, the value of S/V is close to zero when the volume is very big with respect to the surface (i.e. the computation is very heavy with respect to the communication). For values of S/V close to zero, the quantity C^p_{over} is close to 1. In fact, by the definition of S/V it holds that $C^p_{over} = \frac{1}{1+\frac{S}{V}}$. This is the reason why, in Table 1, by increasing the problem size, the performance of the parallel algorithm improves. As confirmed by results in Table 1, the values of p, up to whom the $Measured\ S_p$ is very close to the value of S_p, depend on the problem size.

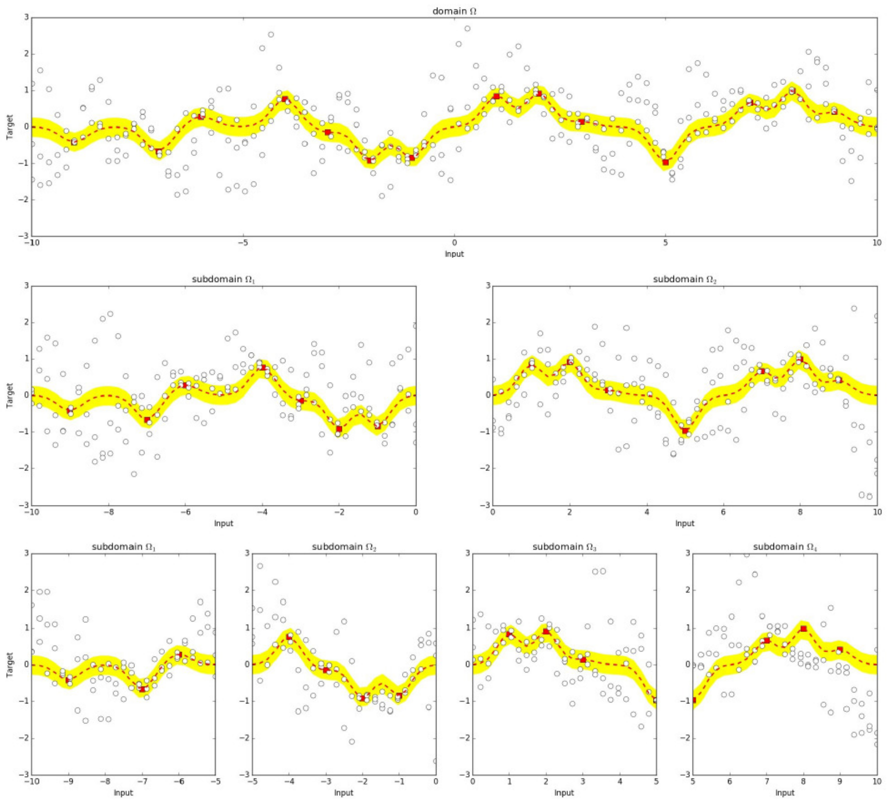

Fig. 3. *Results of the test case with $L = 2$ for $p = 1$, $p = 2$, $p = 4$ computing processors (i.e. for $p = 1$, $p = 2$, $p = 4$ subdomains which constitute the decomposition described in (15))*

Problem size:		$N = 100$	$N = 1000$	$N = 10000$
p	S_p	*Measured* S_p	*Measured* S_p	*Measured* S_p
2	4	3.9	3.9	4.0
4	16	15.9	15.9	15.9
8	64	63.9	63.9	63.9
16	256	254.1	255.9	255.9
32	1024	994.4	1023.8	1023.9
64	4096	3763.0	4093.0	4095.9
128	16384	13815.3	16314.9	16383.8
256	65536	50694.2	64287.8	65532.2

Table 1. *values of Measured Scaling factor (Measured S_p) for $N = 100$, $N = 1000$ and $N = 10000$ compared with the Scaling Factor (S_p) for a number of processors* $2 \leq p \leq 256$

Figure 5 shows the time reduction T^p_{red} which is the execution time $T^P_{\mathcal{A}}$ in (32) normalized to 1 *second*. Also in this case, the obtained results underline how the performance improvement in terms of execution time reduction increases when

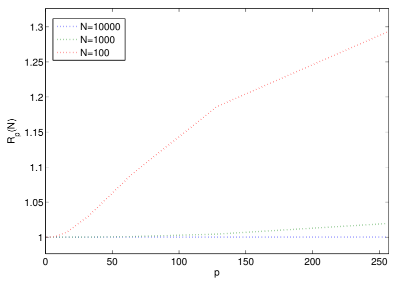

Fig. 4. *values of the ratio* $R_p(N)$ *in (42) for* $N = 100$*,* $N = 1000$*,* $N = 10000$ *and for a number of processors* $1 \leq p \leq 256$

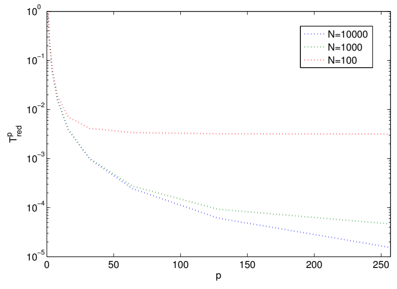

Fig. 5. *values of time reduction* T^p_{red} *for* $N = 100$*,* $N = 1000$*,* $N = 10000$ *and for a number of processors* $1 \leq p \leq 256$

the volume of the data increases. This is still due to the communication overhead which dominates datasets from smaller domains.

7 Conclusions

In this paper a Weak Constraint Gaussian Process (WCGP) algorithm is presented to integrate noisy inputs into the classical Gaussian Process predictive distribution and we have demonstrate the scalability and the accuracy of our approach. The algorithm developed starts by integrating noisy input into a GP model before to provide a parallel implementation i.e. by decomposing the domain. A mathematical formulation of the model is provided and the mathematical validity of this formulation is proved. Furthermore, the algorithmic scalability has been proved to be $\mathcal{O}(p^2)$ where p is the number of processors. In order to evaluate the performance of the parallel algorithm, a scaling factor (which measures the performance gain in terms of time complexity reduction) is introduced and compared with a measured scaling factor (which measures the performance gain in terms of execution time reduction). The algorithm has been evaluated on data sets for $100 \leq N \leq 10000$. For a fixed number of processors p, results show how the performance improves as the size of the problem size increases due to the communication overhead which decreases.

References

1. Arcucci, R., D'Amore, L., Pistoia, J., Toumi, R., Murli, A.: On the variational data assimilation problem solving and sensitivity analysis. Journal of Computational Physics **335**, 311–326 (2017)
2. Cacuci, D., Navon, I., Ionescu-Bujor, M.: Computational methods for data evaluation and assimilation. CRC Press (2013)
3. Cressie, A.: Statistics for spatial data. WileyInterscience (1993)
4. Deisenroth, M.P., Ng, J.W.: Distributed gaussian processes. Proceedings of the International Conference on Machine Learning (ICML), Lille, France (2015)

5. Foster, I.T.: Designing and building parallel programs - concepts and tools for parallel software engineering. Addison-Wesley (1995)
6. Fox, G., Williams, R., Messina, P.: Parallel. computing works! Morgan Kaufmann Publishers Inc., Los Altos, CA (1994)
7. Girard, A., Murray-Smith, R.: Learning a gaussian process model with uncertain inputs. Technical Report TR-2003-144 (2003), Department of Computing Science, University of Glasgow
8. Guestrin, C., Krause, A., Singh, A.: Near-optimal sensor placements in gaussian processes. 22nd International Conference on Machine Learning (ICML) pp. 265–272 (2005)
9. Kalnay, E.: Atmospheric modeling, data assimilation and predictability. Cambridge University Press, Cambridge, MA (2003)
10. Matheron, G.: The intrinsic random functions and their applications. Advances in Applied Probability **5**, 439–468 (1973)
11. McHutchon, A., Rasmussen, C.E.: Gaussian process training with input noise. NIPS'11 Proceedings of the 24th International Conference on Neural Information Processing Systems pp. 1341–1349 (2011)
12. Neil, L.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. Machine Learning Research Vol.6 pp. 1783–1816 (2005)
13. Osborne, M., Rogers, A., Ramchurn, A., Roberts, S., Jennings, N.R.: Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. IPSN 2008: International Conference on Information Processing in Sensor Networks (2008)
14. Park, C., Huang, J.Z., Ding, Y.: Domain decomposition approach for fast gaussian process regression of large spatial data sets. Journal of Machine Learning Research **12**, 1697–1728 (2011)
15. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. the MIT Press (2006). https://doi.org/ISBN 026218253X
16. Titsias, M., Lawrence, N.: Variational learning of inducing variables in sparse gaussian processes. 12th International Conference on Artificial Intelligence and Statistics (AISTATS) (2009)
17. Titsias, M., Lawrence, N.: Bayesian gaussian process latent variable model. 13th International Conference on Artificial Intelligence and Statistics (AISTATS) (2010)
18. Williams, C.K.I., Seeger, M.: Using the nystrom method to speed up kernel machines. In Advances in Neural Information Processing Systems 12 pp. 682–688 (2000)
19. Xiao, D., Fang, F., Zheng, J., Pain, C., Navon, I.: Machine learning-based rapid response tools for regional air pollution modelling. Atmospheric Environment **199**, 463–473 (2019)
20. Xiao, D., Heaney, C., Fang, F., Mottet, L., Hu, R., Bistrian, D., Aristodemou, E., Navon, I., Pain, C.: A domain decomposition non-intrusive reduced order model for turbulent flows. Computers & Fluids (2019)
21. Xiao, D., Heaney, C., Mottet, L., Fang, F., Lin, W., Navon, I., Guo, Y., Matar, O., Robins, A., Pain, C.: A reduced order model for turbulent flows in the urban environment using machine learning. Building and Environment **148**, 323–337 (2019)
22. Yarin, G., van der Wilk Mark, E., R.C.: Distributed variational inference in sparse gaussian process regression and latent variable models. In Advances in Neural Information Processing Systems (2014)