Journal of Computational Science 42 (2020) 101110

Contents lists available at ScienceDirect

Journal of Computational Science

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

Weak Constraint Gaussian Processes for optimal sensor placement

Tolga Hasan Dur[a], Rossella Arcucci[a,*], Laetitia Mottet[b], Miguel Molina Solana[c,a], Christopher Pain[b], Yi-Ke Guo[a]

[a] Data Science Institute, Department of Computing, Imperial College London, UK
[b] Department of Earth Science & Engineering, Imperial College London, UK
[c] Department of Computer Science and AI, Universidad de Granada, Spain

ARTICLE INFO

Article history:
Received 11 December 2019
Received in revised form 4 February 2020
Accepted 11 March 2020
Available online 14 March 2020

Keywords:
Gaussian Processes
Sensor placement
Data assimilation
Parallel algorithms
Big data

ABSTRACT

We present a Weak Constraint Gaussian Process (WCGP) model to integrate noisy inputs into the classical Gaussian Process (GP) predictive distribution. This model 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 use the algorithm for an optimal sensor placement problem. Experimental results are provided for pollutant dispersion within a real urban environment.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

New computational tools and the maturity of Machine Learning as well as Big Data techniques, have enabled the management and processing of large amount of data. With applications in several domains, those from the natural sciences, such as meteorology and pollution, remain amongst the most relevant for researchers and with a highest societal impact. Traditional and state-of-the-art forecasting and prediction techniques are greatly benefiting from availability of real and simulated data. Cheaper sensing devices are also contributing to having more accurate and redundant measures over larger areas.

Gaussian Processes (GPs) are one of the most widely used techniques in forecasting since the 1970s in the fields of geostatistics and meteorology. They have had a substantial impact in technologies including geostatistics [1] and feature reduction [2], but its current applications are in diverse fields such as geophysics [3], medical imaging [4], multi-sensor fusion [5] and sensor placement [6]. While most uses of GPs were traditionally in 2D or 3D spaces, in the last decade, researchers in the Machine Learning community have embraced GPs to work in high dimensional spaces.

However, the main limitation of traditional GP regression is its sensitivity to noisy input and the computational complexity. In previous work [7], we reformulated the GP as a Weak Constraint Gaussian Process to overcome the computational complexity issues and optimally deal with noisy data. The former was done by decomposing the domain and therefore enabling parallel computations, while the latter was archived by a Data Assimilation approach that incorporates noisy input in a time window.

In the current paper, we are extending that theoretical contribution to a real use-case, that of optimal sensor placement. And particularly, to study the optimal sensor placement for collecting air pollution data in big cities. In order to build a computational system to accurately predict air pollution in urban environments, we need to continuously assimilate data into the simulation model, so predictions do not greatly diverge from reality. This in turn requires the placement of enough sensors to measure the independent variable for the entire space with a minimal error. However, as the custom developed sensors are very expensive, it is of utmost importance to place as few sensors as possible. A Gaussian Process (GP) based solution constitutes the ideal solution to this optimisation problem [8].

Specifically, in this paper, we developed a modified version of the algorithm from [8] using our Weak Constraint Gaussian Process in parallel on sub-domains obtained by the domain decomposition methodology described in [7]. To perform this in a real-case scenario, we dealt with issues of monotonicity and numerical stability, and we developed a way to identify relevant sub-domains. For each relevant sub-domain, we ran a separate sensor placement algo-

* Corresponding author.
E-mail address: r.arcucci@imperial.ac.uk (R. Arcucci).

https://doi.org/10.1016/j.jocs.2020.101110
1877-7503/© 2020 Elsevier B.V. All rights reserved.

rithm in parallel, using multi-threading to fully exploit all available resources.

As test-case, we placed sensors in a very large domain around the London South Bank University [9] in London, UK. We validated the results by using a GP fitted with the sensor placements to predict the mean pollution value of all nodes in the domain. By comparing them to the real mean pollution value and the mean pollution value predicted with random placement, we confirmed that our sensor placements are indeed close to optimal, with near-perfect validation results.

The rest of the paper is structured as follows: Section 2 reviews previous work on GPs and their application to the optimal sensor positioning problem. Section 3 describes Weak Constraint Gaussian Processes, and Section 4 describes our proposal. Section 5 presents and discusses the experimental results. We conclude the paper with a summary and a brief discussion on future work.

2. Related works and contribution of the present work

2.1. Gaussian Processes

GPs are widely used for various applications in spatial statistics and meteorology. However, due to their limitations in computational complexity and sensitivity to noise, and the vast amount of real-life, noisy data usually available in spatial statistics, the prediction quality in most applications is non-optimal at best. To solve these problems, several works have been proposed.

To deal with the issue of high sensitivity to noise, the authors in [10], for instance, expanded the GP around the input mean (delta method), assuming the random input is normally distributed. They then derive a new process whose covariance function accounts for the randomness of the input. Another example is [11], where the input noise variances are inferred from the data as extra hyper-parameters.

In contrast, we propose a Weak Constraint Gaussian Process (WCGP) model to improve the accuracy of the posterior distribution of the classical GP [12]. Noisy inputs are integrated into the GP through a data assimilation approach (i.e. by considering information provided by observed values of the noisy input). As *assimilated observations* are not verified exactly [13], 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 show in Section 3. The number of processed data points for this model increases with respect to the classical GP while the GP time complexity is $\mathcal{O}(N^3)$.

A sound approach to reduce time complexity is to introduce approximation methods. In [14] 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 $\mathcal{O}(NM^2)$, making the run less computationally expensive. However, parameters must still be selected a-priori and, consequently, unfavourably affecting sensitivity [15].

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 [16] a scalable sparse Gaussian Process (GP) regression [17] and Bayesian Gaussian Process latent variable model (GPLVM) [18] were presented. This work represented the first distributed inference algorithm able to process datasets with millions of points. However, even with sparse approximations it is inconceivable to apply GPs to training sets of size larger than $\mathcal{O}(10^7)$.

In [19] a distributed GP model is introduced. Their key idea is to recursively distribute computations to independent computational units to later 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 [20], the partitioning problem (i.e. decomposability: to break the problem into small enough independent less complex sub-problems) is a universal source of scalable parallelism. In [21] a domain decomposition approach for the classical GP (or [22] applied to urban flows simulation) are presented based on the definition of boundary conditions for the sub-problems. While the introduced approach allows big data problems to be tackled, there is a subsequent loss in accuracy.

In this paper, we formally address the parallelism problem by defining a parallel Weak Constraint GP (WCGP) model based on the previously introduced domain decomposition approach. The accuracy of the proposed approach is demonstrated (Section 4) by showing that the solution obtained by the parallel algorithm is the same as obtained by the sequential one. We also study the algorithm's scalability, taking into account the time complexity of the algorithm when implementing a domain decomposition.

2.2. Optimal sensor placement

Regarding the sensor placement problem to which we will be applying our proposal, the most straightforward approach for sensor placement is to assume fixed sensing radii of sensors, and then place them such that the entire space is covered [23,24]. However, this assumption is too strong as sensors make noisy measurements that cannot be described by spheres or any other fixed-size representation [8]. Rather, their sensing field depends on the correlation between sensors which can be negative or positive. This means that the combined sensing radius can be smaller or larger than just the sum of fixed-size spherical sensing radii of two sensors. Therefore, a single sensor can often not held responsible for sensing one location. This fundamentally wrong assumption leads to sub-optimal results.

Classical design criteria in this domain essentially tries to minimise the estimation error with the maximum likelihood methodology. In contrast, Bayesian design criteria uses Bayesian statistics to solve the regression. The Bayesian design criteria are defined as the change in expected utility between prior and posterior distribution [8]. Information-theoretic design criteria are special cases of the Bayesian experimental criteria where an information-theoretic function such as entropy or mutual information is defined as the utility function [8]. Minimising the estimation error is equivalent to maximising entropy. This is an NP-hard combinatorial optimisation problem and the basic approach means trying every possible combination of the sensor placements until the best one is found, which is computationally not feasible [25].

Therefore, a typical approach suggests to greedily place the sensors at the point of the highest entropy [1,26]. That is, the locations we are most uncertain about given the already placed sensors akin to the exploitative objective of the acquisition function in Bayesian optimisation. This approach, however, places sensors at the border of the area we are interested in and therefore wastes sensing information [27,28]. Mutual information, as implemented in this paper, on the other hand, does not have this problem as it measures the effect of sensor placement on the posterior uncertainty of the GP directly [8]. Our goal is, therefore, to find the best sensor locations such that mutual information is maximised using the WCGP to estimate the posterior uncertainty.

In this paper, the problem of finding the optimal sensor placement for pollutant dispersion within an urban environment is addressed using the WCGP model. Data are provided by simulations

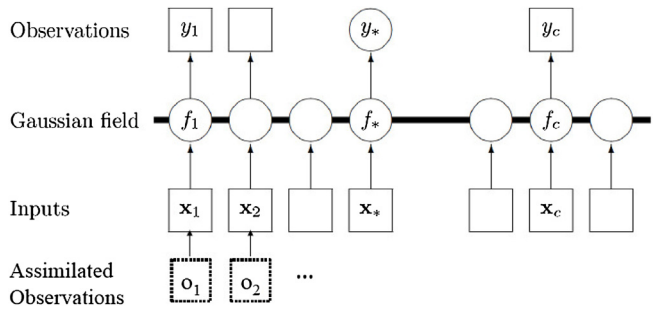


Fig. 1. Graphical model (chain graph) for a WCGP for regression. Squares represent observed variables and circles represent unknowns. Dotted squares represent innovation with respect a classical GP [19]. The novel observed data are assimilated into the input data of the classical GP. This process results in a modified GP minimisation problem we named WCGP.

performed using the open-source, finite-element, fluid dynamics model Fluidity [29]). The details of the equations solved and their implementation can be found in [30,29]. The state variable consists of values of pressure, velocity and pollution concentration. Observed values of the state variable are assumed from positions mainly located on the roof of the buildings. However, the algorithm and numerical methods proposed in this work can be applied to other physical problem involving other equations and/or state variables.

3. Weak Constraint Gaussian Processes (WCGP)

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

$$x = [x_1, x_2, \ldots, x_N]^T \tag{2}$$

and

$$y = [y_1, y_2, \ldots, y_N]^T. \tag{3}$$

The joint distribution of (f_*, y) is

$$(f_*, y) = \mathcal{N}\left(0, \begin{bmatrix} k_{x_*x_*} & k_{xx_*}^T \\ k_{xx_*} & \sigma_y^2 I + K_{xx} \end{bmatrix}\right) \tag{4}$$

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

$$P(f_* \| y) = \mathcal{N}\left(k_{xx_*}^T A^{-1} y, k_{x_*x_*} - k_{xx_*}^T A^{-1} k_{xx_*}\right) \tag{5}$$

where

$$A = \sigma_y^2 I + K_{xx} \tag{6}$$

Data Assimilation (DA): At each step i, $i = 1, \ldots, N$ (see Fig. 1), let

$$o = H(x), \quad o = [o_1, o_2, \ldots, o_N]^T \tag{7}$$

be the observations vector, where H is a non-linear operator collecting the assimilated observations at each step. The aim of a DA problem is to find an optimal trade-off between the current estimate of the system state (background) defined in Eq. (5) and the available assimilated observations. Let R be a covariance matrix whose elements provide the estimate of the errors[1] on o in Eq. (7), the assumption of input noise and the assimilation of it by a data assimilation approach [31] introduce a corrective term

$$O = H^T R^{-1} H \tag{8}$$

to the output noise. Then the resulting Gaussian Process is

$$P(f_* \| y, o) = \mathcal{N}\left(k_{xx_*}^T \hat{A}^{-1} y, k_{x_*x_*} - k_{xx_*}^T \hat{A}^{-1} k_{xx_*}\right) \tag{9}$$

where

$$\hat{A} = A + O \tag{10}$$

with A and O defined in Eqs. (6) and (8) respectively, and where we assume that each input dimension is independently corrupted by noise, thus R is diagonal:

$$R = \sigma_x^2 I \tag{11}$$

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 [21]. Consider all linear predictors

$$\mu(x_*) = u(x_*)^t y, \tag{12}$$

satisfying the unbiasedness requirement $E[\mu(x_*)] = 0$ where E denotes the expectation [12]. We want to find the vector $u(x_*)$ which minimises 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

$$\sigma(x_*) = u(x_*)^t E(yy^t) u(x_*) - 2u(x_*)^t E(yf_*) + E(f_*^2)$$

$$= u(x_*)^t (\sigma_y^2 I + K_{xx} + H^T R^{-1} H) u(x_*) - 2u(x_*)^t k_{xx_*} + k_{x_*x_*} \tag{13}$$

Eq. (13) is a quadratic form in $u(x_*)$. It is easy to see $\sigma(x_*)$ is minimised 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 Eq. (9) or the best linear unbiased predictor can be obtained by solving the following minimisation problem: for $x_* \in \Omega$, compute

$$\overline{u}(x_*) = argmin_{u(x_*)\in\Re^N} J(u(x_*)) \tag{14}$$

with

$$J(u(x_*), \sigma_y, K_{xx}, H, R, \Omega) = u(x_*)^t (\sigma_y^2 I + K_{xx} + H^T R^{-1} H) u(x_*)$$
$$- 2u(x_*)^t k_{xx_*} \tag{15}$$

To compute the minimum of J in Eq. (15), 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 to the ill conditioning of the matrix K_{xx},

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

the matrix $\hat{A}$ is ill conditioned as well, then is mandatory to introduce a preconditioning [12]. Since K_{xx} is symmetric and positive definite, it is possible to compute its Cholesky factorisation

$$K_{xx} = VV^t. \tag{16}$$

where [32]:

$$V_{i,j} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(i-j)^2}{2\sigma^2}\right). \tag{17}$$

Let be $c(\cdot)$ denote the condition number, by this way it is:

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

i.e., the Cholesky factorisation of the covariance matrix K_{xx} mitigates the ill conditioning of the minimisation problem. Let be $v = V^+u$, where $+$ denotes the generalised inverse of the matrix V. After the preconditioning the minimisation problem in Eqs. (14) and (15) is written as:

$$\bar{v}(x_*) = argmin_{v(x_*)\in\Re^N} J(v(x_*)) \tag{18}$$

with

$$J(v(x_*), \sigma_y, V, H, R, \Omega) = v^tV^t(I + \sigma_y^2 I + H^TR^{-1}H)Vv - 2v^tV^tk_{xx_*} \tag{19}$$

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 [11,10]. Here we face the problem concerning the high computational cost by introducing a domain decomposition approach into the mathematical model. In next sections we introduce an optimal sensor placement algorithm we developed based on this WCGP model defined in a decomposition of the domain.

4. WCGP for optimal sensor placement

The algorithm we developed for optimal sensors placement is based on the minimisation of the entropy.

Here, entropy tries to quantify uncertainty by finding a function whose output continuously increases with uncertainty in a random variable. According to Shannon [33], such a function should be additive, transitive and maximal when $P_X(x)$ is uniform. As proven in [33], the only function that satisfies these conditions is the formal definition of entropy defined in Eq. (20):

$$H(x) = -\sum_{x_i} P_x(x_i)\log P_x(x_i) = -E_{P_x}\log P_x \tag{20}$$

where x is defined in Eq. (2), P is defined in Eq. (5) and E denotes the expectation [12].

Intuitively, entropy is the minimum number of yes/no questions it takes to guess a random variable given perfect knowledge of the underlying distribution. Naturally, conditional entropy is the uncertainty of the random variable x given knowledge about another random variable y as seen in Eq. (21):

$$\begin{aligned} H(x|y) &= \sum_{y_i} P_y(y_i)\left[-\sum_{x_i} P_{x|y}(x_i|y_i)\log\left(P_{x|y}(x_i|y_i)\right)\right] \\ &= E_{P_y}\left[-E_{P_{x|y}}\log P_{x|y}\right] \end{aligned} \tag{21}$$

Mutual information, on the other hand, is the reduction in uncertainty after observing random variable y [34], as defined in Eq. (22):

$$I(x;y) = H(x) - H(x|y) \tag{22}$$

Optimising for any of the described design criteria yields a combinatorial optimisation problem where one must search for the best sensors placement among a very large number of candidate solutions. Instead, as the number of candidate solutions is too large for an exhaustive search approach, commonly some heuristic is taken. Assuming that the random variables are normally distributed, as they have to be for GP to work, the conditional entropies $H(y|x)$ is defined as in Eq. (23):

$$H(x|y) = \frac{1}{2}\log(2\pi\exp\sigma^2_{x|y}) = \frac{1}{2}\sigma^2_{x|y} + \frac{1}{2}(\log(2\pi)+1) \tag{23}$$

where σ is given by the WCGP as defined in Eq. (13), i.e. minimising J defined in the control space as in Eq. (19).

The exact treatment of function J 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 [11,10]. Here we face the problem concerning the high computational cost by introducing a domain decomposition approach into the mathematical model.

The mathematical formalisation of the WCGP model based on a domain decomposition approach is presented in the following. 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 such that:

$$DD(\Omega) = \left\{\Omega_i\right\}_{i=1,\dots,p} \tag{24}$$

with $\Omega_i \subset \Re^{r_i}$, $r_i \leq N$ and for $i = 1,\dots,p$, 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 sub-domain Ω_i and by adding a *local* constraint onto the overlap region between adjacent domains Ω_i and Ω_j such that:

$$J_{\Omega_i}(v_i) = v_i^tV_i^t(I_i + \sigma_y^2 I_i + H_i^TR_i^{-1}H_i)V_iv_i - 2v_i^tV_ik_{xx_*} + \|v_{ij} - v_{ji}\| \tag{25}$$

where v_i, V_i, I_i, R_i and H_i are restrictions on Ω_i of vectors and matrices in Eq. (15), 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 $\Omega_i \cap \Omega_j \neq \emptyset$.

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 WCGP solution. Let $\bar{v}_i = argmin_{v_i} J_{\Omega_i}(v_i)$ denote the minimum of J_{Ω_i} and let $\tilde{v}$ denote *extension vector* defined as the sum of the extensions of $\bar{v}_i$ $\forall i$, to the domain Ω:

$$\tilde{v}_i = \begin{cases} \bar{v_i} & in \quad \Omega_i \\ 0 & in \quad \Omega - \Omega_i \end{cases} \qquad and \quad \tilde{v} = \sum_{i=1}^{p}\tilde{v}_i \tag{26}$$

A theorem which ensures that the solution obtained by the parallel algorithm $\tilde{v}$ 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 sub-domains and the global domain respectively. Let $\tilde{J}_i$ denote the *extension function* of J_{Ω_i} to Ω:

$$\tilde{J}_i = \begin{cases} J_{\Omega_i} & in \quad \Omega_i \\ 0 & in \quad \Omega - \Omega_i \end{cases} \tag{27}$$

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

$$\sum_{i=1}^{p}\tilde{J}_i = J. \tag{28}$$

Theorem 4.1. *Let $DD(\Omega)$ be a decomposition of the domain Ω. With $\tilde{v}$ as defined in Eq. (26) and $\bar{v} = \bar{v}(x_*)$ as defined in equation (18), it follows that:*

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

Proof. Let $\bar{v}_i$ be the minimum of J_{Ω_i} on Ω_i such that

$$\nabla J_{\Omega_i}[\bar{v}_i] = 0, \quad \forall i : \Omega_i \in DD(\Omega). \tag{29}$$

$$\nabla \sum_i J_{\Omega_i}[\bar{v}_i] = 0 \tag{30}$$

$$\nabla \sum_i \tilde{J}_i(\tilde{v}_i) = 0 \tag{31}$$

which gives from Eqs. (28) and (26):

$$\nabla J(\tilde{v}) = 0 \tag{32}$$

$$J(\bar{v}) < J(\tilde{v}), \tag{33}$$

on Ω Eq. (33) gives $J(\bar{v}(x_j)) < J(\tilde{v}(x_j))$ for all $x_j \in \Omega$. In particular, let Ω_i be a subset of Ω, then $J(\bar{v}(x_j)) < J(\tilde{v}(x_j))$ for all $x_j \in \Omega_i$. Hence, considering the restriction to Ω_i, $\forall i$ and by assuming Eq. (33), we have

$$J_{\Omega_i}(\bar{v}/\Omega_i) < J_{\Omega_i}(\tilde{v}/\Omega_i), \quad \forall i : \Omega_i \subset \Omega \tag{34}$$

which gives from Eq. (26):

$$J_{\Omega_i}(\bar{v}/\Omega_i) < J_{\Omega_i}(\bar{v}_i). \tag{35}$$

Eq. (35) is an absurd. In fact, if $\bar{v}/\Omega_i \neq \bar{v}_i$, Eq. (35) says that it 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 in the placement algorithm. In fact, mathematically, this means that the conditional entropy $H(x_{\Omega\setminus S_i}|x_{S_i})$ is minimised on sub-domains Ω_i. Here, Ω_i are all nodes (e.g. points of interest) in the sub-domain Ω_i and S_i the chosen sensor placements. As $H(x_{\Omega_i\setminus S_i}|x_{S_i})$ is equivalent to to $H(x_{\Omega_i}) - H(x_{S_i})$, this means that minimising $H(x_{\Omega_i\setminus S_i}|x_{S_i})$ is equivalent to maximising entropy $H(x_{S_i})$. This is an NP-hard combinatorial optimisation problem and the basic approach means trying every possible combination of the sensor placements x_{S_i} until the best one is found, which is computationally not feasible [25].

Therefore, the typical approach suggests to greedily place the sensors at the point of the highest entropy [1,26]. That is, the locations we are most uncertain about given the already placed sensors akin to the exploitative objective of the acquisition function in Bayesian optimisation. This approach, however, places sensors at the border of the area we are interested in and therefore wastes sensing information [27,28]. Mutual information, which is described in Eq. (22), on the other hand, does not have this problem as it measures the effect of sensor placement on the posterior uncertainty of the WCGP directly. Our goal is, therefore, to find the best sensor locations S_i such that $I(S_i; \Omega_i\setminus S_i)$ is maximised. As this is not computationally feasible for the same reasons as above, we use a greedy approach in which we maximise $I(S_i \cup y) - I(S_i)$ instead. This is equivalent to maximising $H(y|S_i) - H(y|\hat{S}_i)$ with $\hat{S}_i = \Omega_i\setminus(S_i \cup y)$. Intuitively, this does the following: maximising $H(y|S_i)$ means searching for a position y with high uncertainty given the already placed sensor S_i (i.e a position that is different). Maximising $-H(y|\hat{S}_i)$ means searching for a position y with low uncertainty given the rest of the field that we are interested in $\Omega_i\setminus S_i$ (i.e. y is informative). This means that greedily selecting placements with the highest mutual information is finding positions are different and informative at the same, which is a little bit like the exploration and exploitation objective of the acquisition function in Bayesian optimisation.

We can show that, maximising $H(y|S_i) - H(y|\hat{S}_i)$ is equivalent to maximising $\sigma^2_{y|S_i}/\sigma^2_{y|\hat{S}_i}$ [35]. In fact:

$$\begin{aligned} &\max(H(y|S_i) - H(y|\hat{S}_i)) = \max\left(\frac{1}{2}\sigma^2_{y|S_i} + \frac{1}{2}(\log(2\pi)+1)\right. \\ &\left. - \left(\frac{1}{2}\sigma^2_{y|\hat{S}_i} + \frac{1}{2}(\log(2\pi)+1)\right)\right) \\ &= \max\left(\frac{1}{2}\sigma^2_{y|S_i} - \frac{1}{2}\sigma^2_{y|S_i}\right) \\ &= \max(\sigma^2_{y|S_i}/\sigma^2_{y|\hat{S}_i}) \end{aligned} \tag{36}$$

where $\sigma^2_{y|S_i}$ is the variance update rule of GP which is described in Eq. (13). Essentially, the greedy algorithm computes two WCGP for each y_i in $\Omega_i\setminus S_i$. The sensor placement is then chosen to be the one with the highest $\sigma^2_{y|S_i}/\sigma^2_{y|\hat{S}_i}$.

In Algorithm 1 we show the pseudo-code for our described sensor placement methodology, with k, Ω_i^{poss} and Ω_i^{unposs} being the number of sensors to be placed, the possible sensor placements and the impossible sensor placements respectively. To calculate one sensor selection, mutual information is calculated for each node $\Omega_i^{poss}\setminus S_i$. The node with the highest mutual information value is taken as the next sensor placement. This process is done k times with respect to information gathered from previous sensor selections.

The algorithm is modified with a priority queue and a local kernel approximation to reduce time-complexity significantly. The prior lowers the number of mutual information evaluation which is the bottleneck of the algorithm as described earlier. The latter works by only considering sensor placements for which the covariance is greater than ϵ when calculating entropy. Further, instead of evaluating mutual information for all nodes in $\Omega_i^{poss}\setminus S_i$, we are only evaluating it for nodes $x \in \Omega_i^{poss}$ for which $|K(y^*, x)| > \epsilon$. As the covariance matrix in the case of sensor placement is usually very sparse, this lowers the size of the matrix that we are inversing and the number of evaluations significantly. Depending on the value of ϵ and the covariance matrix, however, this improvement in time-complexity might decrease accuracy.

Algorithm 1. $\mathcal{A}(\Omega_i, p)$: WCGP for Optimal Sensor Placement

Input: Covariance matrix $K(\cdot,\cdot)$, number of sensor k, $\Omega_i = \Omega_i^{poss} \cup \Omega_i^{unposs}$, S_i
Output: Sensor selection $S_i \subseteq \Omega_i^{unposs}$

$S_i \longleftarrow S_i$;
$\epsilon \longleftarrow 1e-10$
foreach $y \in \Omega_i^{poss}$ **do** $\delta_y \longleftarrow +\infty$;
for $j = 1$ **to** k **do**
  foreach $y \in \Omega_i^{poss}\setminus S_i$ **do** $current_y \longleftarrow False$;
  while True **do**
  $y^* \longleftarrow argmax_{y \in \Omega_i^{poss}\setminus S_i} \delta_y$;

8: **if** $current_{y^*}$ **then break;**
9: $\delta_{y^*} \longleftarrow \tilde{H}_\epsilon(y|S_i) - \tilde{H}_\epsilon(y|\hat{S}_i)$;
10: $current_{y^*} \longleftarrow True$
11: **end while**
12: $S_i \longleftarrow S_i \cup y^*$;
13: **end for**

It has been proved in [8] that the approach described in Algorithm 1 ensures an approximate solution with an accuracy of $(1 - \frac{1}{e})$ of the perfect solution if the function is monotonic sub-modular [36].

However, while mutual information is indeed sub-modular, it is not always monotonic. As shown in [8], for spaces with less than $2k$, where k is the number of sensors, the approximation holds even if mutual information is not monotonic. For spaces with more than $2k$, nonetheless, mutual information has to be monotonic for the approximation guarantee to hold [8]. As we are dealing with a space of size $N > 2k$, this means that we have to ensure that monotonicity holds in our dataset. In fact, in contrast to the case discussed in [8], we are dealing with data at much larger scale as it is generated through simulations rather than actual measurements. Monotonicity in the case of mutual information holds if $H(y|S_i) - H(y|\hat{S}_i) \geq 0$. If it is not, [8] suggests to maximise for $H(y|A) - H(y|Z \cup \hat{S}_i)$ instead with Z_i being a grid of unobservable nodes. Adding a large enough Z ensures that $H(y|Z_i \cup \hat{S}_i)$ is small enough for mutual information to be approximately monotonic [37]. Our methodology checks whether monotonicity holds in the dataset and suggests to alternative approaches if it does not.

4.1. Complexity and scalability

In this section we provide an estimate of the scalability of our approach through a study of time complexity. First we provide the following definition of *Scaling factor*, which measures the performance gain in terms of time complexity reduction, as the ratio

$$S_p = \frac{\tau\left(\mathcal{A}(\Omega, 1)\right)}{\tau(\mathcal{A}(\Omega_i, p))}. \tag{37}$$

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.

The time complexity of a WCGP, when no decomposition is introduced, is $\mathcal{O}(kN)$ where k denotes the number of sensors to place. Then it yields

$$\tau(\mathcal{A}(\Omega, 1)) = kN \tag{38}$$

By introducing the domain decomposition of Ω as defined in Eq. (24), and by considering a partition of size p, we are going to solve, by $\mathcal{A}$, p sub-problems of size about N/p, i.e. the time complexity is related to the sub-problems and we have

$$\tau(\mathcal{A}(\Omega_i, p)) = \left(\frac{1}{p} k_p N\right) \tag{39}$$

where k_p denotes the number of sensors to place in the sub-domains of the partition, $k_p \leq k$. From Eq. (37), Eq. (38), and Eq. (39), we have the scaling factor of Algorithm 1 such that

$$S_p = \frac{kN}{\left(\frac{1}{p} k_p N\right)} = \frac{k}{\left(\frac{k_p}{p}\right)} = p\frac{k}{k_p} \tag{40}$$

In Section 5, our Algorithm 1 is used and validated on a real test case for optimal sensor placement in the context of air pollution. We study the impact of the optimal sensor placement computed by Algorithm 1 analysing the GP's mean pollutant concentration prediction and the impact in reducing the errors of a predictive model using data assimilation. The efficiency and the scalability of Algorithm 1 is also discussed.

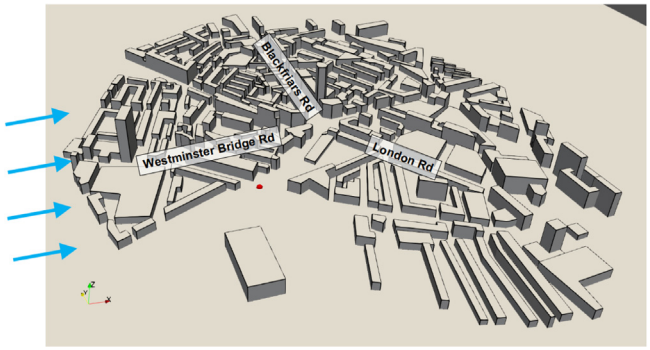


Fig. 2. Computational domain of the test site: the London South Bank University (LSBU), London (UK). The red sphere denotes the location of the source releasing pollution and the blue arrows the wind direction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Results and discussion

5.1. Test case description

The methodology described in this paper is tested for outdoor sensor placement on a real test case. The test-site comprises 148 buildings and is located around London South Bank University (LSBU), Elephant and Castle, London, UK (Fig. 2). The size of the computational model used in this paper is $2041 \times 2288 \times 250\,\mathrm{m}^3$ following CFD good practices [38]. This test case has been extensively used in the context of the MAGIC research project [9,39,5] (http://www.magic-air.uk/).

With the aim of better understanding and predicting air pollution level in cities, sensors optimally positioned can help in increase the accuracy of a predictive air pollution model. We use Fluidity [29] (an open-source finite-element Computational Fluid Dynamic (CFD) software) as predictive model. The flow field is computed from the three-dimensional incompressible Navier–Stokes equations, while the transport of the pollution is resolved using an advection-diffusion equation. In addition, a Large Eddy Simulation (LES) approach is used for the turbulence model. All the equations are solved using second-order scheme in space and time. For sake of brevity, the equations are not reported in this paper but their details can be found in [29,40–42]. The atmospheric boundary layer (ABL) is represented using a turbulent inlet boundary condition based on a synthetic eddy method (SEM) [41]. The SEM requires four given parameters: the number of eddies N_{eddies}, the mean velocity profiles (u, v, w), the Reynolds stresses $(\overline{u'u'}, \overline{v'v'}, \overline{w'w'})$ and the turbulence lengthscales (L_{xx}, L_{yy}, L_{zz}). In our simulation, these variables have been set up to mimic the behaviour of a typical ABL following a log wind profile: $N_{eddies} = 4000$, $(u, v, w) = \left(0.98 \ln\left(z/0.01\right), 0, 0\right)$, $(\overline{u'u'}, \overline{v'v'}, \overline{w'w'}) = (0.8, 0.8, 0.8)$ and $(L_{xx}, L_{yy}, L_{zz}) = (100, 100, 100)$. Zero velocity is prescribed on the bottom of the domain and along wall boundaries. A zero stress condition is used at the outlet which sets $p = 0$. Perfect slip conditions are used on the vertical lateral boundaries as well as top boundary of the domain. This simulation has already been used and described in previous papers [5,9,22,39].

To mimic a constant release of pollution, from a busy crossroads for instance, a source term is added to the transport equation. In this example, the pollutant concentration field considered is the one generated by the point source depicted by the red sphere in Fig. 2. The release of the source is equal to $1\ kg/m^3/s$ and the dispersion is

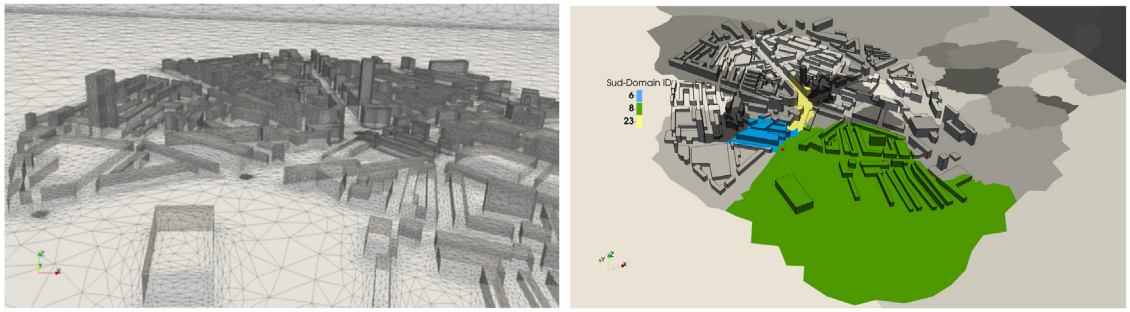


Fig. 3. (a) Surface mesh and (b) the decomposition into 32 sub-domains of the test site: the London South Bank University (LSBU), London (UK). In (b) the red sphere denotes the location of the source releasing pollution, each grey level of colour delimited a sub-domain area. The sub-domain 6 (in blue) and 8 (in green) are the sub-domain considered in this work. The red iso-surface shows the time-averaged spread of pollution for an iso-value of concentration equal to 0.5 kg/m^3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

simulated for a westerly wind (direction shown by the blue arrows in Fig. 2).

As an initialisation step, the simulation is first run during a period long enough to let the flow establish (∼1 H real time). After that, the simulation runs for an extra 536 time-steps with an interval of 4/3 s. (∼10 min real time), which is sufficient to capture the variability of the pollution spreading within the domain. Data from this second period are the one used in the following.

5.2. Numerical and parallel implementation

Fluidity is a fully parallel CFD software using the Message Passing Interface (MPI) library to communicate information between processors. Let $\Omega = \{x_j, y_j\}_{j=1,\dots,N}$ be the discrete spatial domain representing the mesh and let $DD(\Omega) = \left\{\Omega_i\right\}_{i=1,\dots,s}$ be a partition of Ω in sub-domains as implemented in Fluidity [29]. The decomposition of the mesh into sub-domains is based on the number of nodes x_j: although not a strict constraint, the number of nodes is evenly balanced on each processor and it is assumed to be a good proxy for the expected computational load. Other requirements include a minimisation of edge cut and data migration. The domain decomposition into Fluidity using overlapping regions between adjacent sub-domains to ensure continuity, for more details about the domain decomposition in Fluidity see [29]. The mesh is decomposed into sub-domains *.vtu files and *.halo files, the latest ones containing information on overlapping regions. The *.vtu files are read in our algorithm. The mesh used in this work includes 767,559 nodes (Fig. 3(a)) and the mesh was decomposed into 32 sub-domains as shown in Fig. 3(b). The CFD simulation was therefore run on 32 processors, rendering an average of 24,000 nodes per processor.

The optimal sensor placement is computed in parallel only on the relevant sub-domains of the LSBU test-site. To identify which sub-domains are of relevance, for all time-steps, the data were analysed numerically and only sub-domain 6 and 8 were identified as relevant, respectively in blue and green in Fig. 3(b). As a threshold, the time-averaged iso-surface for a concentration equal to 0.5 kg/m^3 is shown in Fig. 3(b). It can be seen that the source of pollution is located into the sub-domain 8 and the pollutant spreads only towards the sub-domain 6. Moreover, sub-domain 6 and 8 are adjacent domains ensuring the continuity of the data. Although a small amount of pollution can be detected in sub-domain 23 (in yellow in Fig. 3(b)), it is not considered significant enough to be taken into account. The concentration of pollution decay very quickly with distance and in all the other sub-domains (depicted in different level of grey colour in Fig. 3(b)), the level of pollution is not only lower than this threshold but also close to zero with values in the range of $[1 \times 10^{-8}, 0]$. To ensure that this choice based on a pollution threshold is correct, the maximum and minimum values, means and standard deviations were analysed and compared for each sub-domains. Fig. 4 shows such a comparison for sub-domains 8 and 15. The concentration in sub-domain 8 is much less centred around zero and covers a range from 0 to approximately 49 compared to 0 to approximately 6.8×10^{-5} for sub-domain 15. The very high maximum concentration in sub-domain 8 is directly attributed by the presence of the source. Comparisons with all other sub-domains yielded similar results, thus confirming that only sub-domains 6 and 8 are of relevance in this work.

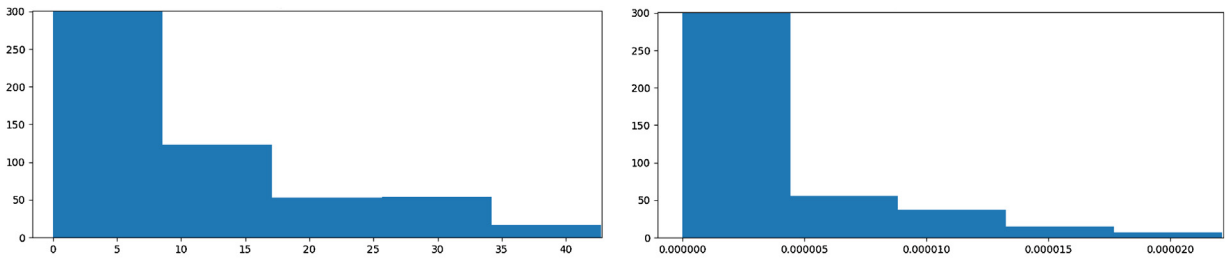
The methodologies for optimal sensor placement presented in this paper were implemented in an home-made code written in Python and is available at https://github.com/tolgadur/sensor-placement. In particular, we used the NumPy python library for all matrix operations. A mix of multi-processing and multi-threading was used to achieve efficient parallelisation. A separate process is run for each sub-domain of interest and multiple threads are used within each of these processes. As a separate sensor placement algorithm is run for each sub-domain of interest, this only utilises CPU's to the number of sub-domains in consideration and is therefore still too slow. For instance, in this application test case, sensors placement is considered in two sub-domains but had over 50 CPU's available. Hence, multi-threading is utilised within each process to fully take advantage of all computational resources. All the numerical tests were run on a cloud computing platform with Intel Cascade Lake CPU and 32 GBP Memory.

The practical limit of a GP is said to be approximately 10, 000 observations [43]. However, even after parallelisation, the individual sub-domains have up to 30, 000 nodes. Furthermore, the sensor placement algorithm works by fitting many GPs for all nodes available in order to calculate the conditional entropy $H(y|\hat{A}_i)$. As such, our tests show that 15, 000 nodes are the limit for our algorithms. To overcome this, we decided to only consider every second node in each sub-domain. The associated precision loss is not significant as the nodes are placed very closely together.

5.3. Results and discussion

In this section we describe the implementation of Algorithm 1 in a parallel computing environment. We introduce the regularisation methods we use to ensure the numerical stability and we discuss the results in terms of both accuracy and efficiency. We also validate the presented results studying the impact of our results in:

1. Gaussian Processes for fluid dynamic predictions: in order to quantify how well the optimal sensor placement performs, we



(a) Sub-domain 8. Max. value: 4.9×10^{1}. (b) Sub-domain 15. Max. value: 6.8×10^{-5}.

Fig. 4. Enlarged histograms for the pollution concentration for sub-domain 8 and 15.

validate the results by fitting a GP with our sensors placements and then used it to predict pollution values at each node. We then compare these to the real pollution values to see how far the predictions are from the real values. Furthermore, we compare this performance indication to that of hundred random placements for further insight. This is directly measuring what our algorithms are optimising for: selecting the most informative k sensor such that pollution in the entire space can most accurately be sensed.

2. Post-processing Data Assimilation: the data measured at the optimally placed sensors are then used as the observed data to improve the accuracy of our predictive model Fluidity. These observed data are assimilated using an optimal variational data assimilation methodology as described in [39,42]. This step will allow us to check how well our sensors are performing in reducing the predictive model error. Indeed, the performance of data assimilation depends on sensor placements as already observed and discussed in [42]. In addition of the sensors locations suggested by the three different approaches, the reduction in error is also compared to the one of sensors randomly chosen (averaged result over a large number of random experiments).

In the following, the error is the mean squared error

$$MSE(c) = \frac{\|c - c^{obs}\|_{L^2}}{\|c^{obs}\|_{L^2}}, \tag{41}$$

where c is either $c^{\mathcal{M}}$ the pollutant concentration from our predictive model or c^{DA} the corrected concentration using data assimilation, and c^{obs} is the real observed data. The lower $MSE(c^{DA})$ is in comparison to $MSE(c^{\mathcal{M}})$, the higher the error reduction and the better our sensors are performing for this specific application.

The reduction in error r_e is computed by equation

$$r_e = \frac{MSE(c^{\mathcal{M}}) - MSE(c^{DA})}{MSE(c^{\mathcal{M}})} \times 100 \tag{42}$$

5.3.1. Numerical stability and efficiency

The sparse nature of the data and covariance matrix V in Eq. (16) is leading to numerical stability problems when calculating conditional variances, even when using the double-precision floating-point format. This sometimes leads to the strange case of negative conditional variances. Naturally, negative conditional variances, which should never occur, lead to undefined entropies due to undefined negative logarithms. This problem can also be seen when computing the determinant of the covariance matrix: although inverting the matrix works without issues, the determinant is almost zero or too small. There are two ways to solve this issue: first, we could take absolute values of the conditional variance, which would ensure that the anomaly of negative conditional variances is eliminated. However, this alters sensor placements as conditional variances are in the heart of the sensor placement algorithms. Second, as often done in GPs, we could add a jitter $I \quad \varepsilon$ to the covariance matrix with I the identity matrix and ε being a very small number in the range of $[1 \times 10^{-3}, 1 \times 10^{-10}]$. However, as some of our variance values are smaller than 1×10^{-20}, adding a number at this range alters the sensor placements as well. Interestingly, we found that it does not matter which of the two we use when not using the local kernel approach. If we do, however, we found that the latter solution leads to much better results than the former solution.

Table 1
Computational time needed to run the three different sensor placement algorithms.

	Time (H)
Algorithm (original)	99.44
Algorithm 1 (jitter)	11.59
Algorithm 1 (abs.)	18.43

The value of ε was optimised. As mentioned, the higher it is, the faster Algorithm 1 due to the decreased size of the covariance matrix. However, accuracy also depreciates. Therefore after several tests, a value of 1×10^{-10} is suggested to be sufficiently small to preserve accuracy.

In the following three different approaches are considered:

- Algorithm 1 (jitter): using Algorithm 1 with preconditioning by adding a jitter to the covariance matrix.
- Algorithm 1 (abs.): using Algorithm 1 with preconditioning by absolute value of the conditional variance.
- Algorithm (original) [8]: using the state of the art algorithm for sensor placement as originally proposed in [8].

Due to the time complexity ($O(kN)$) of the Algorithm (original) [8] and the size of our domain, our implementation of Algorithm (original) [8] is parallelised and also based on a domain decomposition approach to ensure a comparison in the same computational environments.

The three approaches aim to position optimally 4 sensors in each sub-domains, i.e. 8 sensors in total. The choice of 8 sensors was made based on the number of outdoor sensors actually available by the MAGIC research project (see [9] for more details).

Efficiency

The computational time needed to run the three previously mentioned approaches are given in Table 1. The data set consists of the pollutant concentration values in sub-domain 6 (13, 898 nodes) and sub-domain 8 (12, 366 nodes) simulated for 536 time steps. It can be seen that the Algorithm (original) [8] takes, as expected, more than 5 times longer to run than the new Algorithm 1 proposed in this paper. Moreover, Algorithm 1 using a jitter preconditioning is 1.6 times faster than Algorithm 1 using an absolute preconditioning, i.e. 37% faster.

Gaussian Process validation

The validation results of the GP can be seen in Table 2. *Real Mean* values are the mean value of the real pollution observations and

Table 2
Means of predicted (*Estimated Mean*) and real (*Real Mean*) pollution concentration. For *Random*, the average over 100 random placements was taken.

	Sub-domain six		Sub-domain eight	
	Real mean	Estimated mean	Real mean	Estimated mean
Algorithm (original)	8.5413e−03	2.3832e−03	2.4662e−01	1.9598e−01
Algorithm 1 (jitter)	8.5413e−03	1.2502e−02	2.4662e−01	2.2771e−01
Algorithm 1 (abs.)	8.5413e−03	−2.5045e02	2.4662e−01	−2.6403e02
Random	8.5413e−03	−1.9248e00	2.4662e−01	2.3900e02

Table 3
Mean square errors of the predicted model before ($MSE(c^{\mathcal{M}})$) and after ($MSE(c^{\mathcal{M}})$) data assimilation. *MSE* computed using observation at the optimal sensor placements suggested by three different algorithms and random sensor positioning. r_e is the reduction in error in %. For *Random*, the average over 100 random placements was taken.

	Sub-domain six			Sub-domain eight		
	MSE($c^{\mathcal{M}}$)	MSE(c^{DA})	r_e	MSE($c^{\mathcal{M}}$)	MSE(c^{DA})	r_e
Algorithm (original)	7.55e−01	2.12e−01	72	2.24e−01	5.25e−02	77
Algorithm 1 (jitter)	1.43e−01	4.07e−03	97	1.77e−01	3.35e−02	81
Algorithm 1 (abs.)	1.68e−05	1.37e−05	18	1.25e−03	1.01e−07	99
Random	4.24e01	1.30e00	86	6.54e00	8.90e−01	82

Estimated Mean are the mean values of the GP predictions in the respective sub-domain.

The predictions of the average random placement are very bad. For sub-domain six it estimates, on average, a mean value of approximately −1.9248 compared to the real mean of 0.0085. For sub-domain 8, it even estimates a mean of approximately 239 compared to the real mean of 0.2466. The estimations of Algorithm (original) [8] and Algorithm 1 (jitter) are very close to real pollution values. Indeed, for Algorithm (original) [8], the order of magnitude of the predicted mean is the same than the real one for both sub-domains. Although one order of magnitude lower in sub-domain 6, the predicted means by Algorithm 1 (jitter) are really close to the real ones for both sub-domains. Hence, both, Algorithm (original) [8] and Algorithm 1 (jitter), are much better than random placement. This means that the GPs with only four observations at the positions suggested by the respective algorithms are almost perfect in predicting pollution for the entire sub-domain with over 25, 000 nodes. It has to be noted that Algorithm 1 (abs.) performs much worse: it predicts negative mean pollutant concentration which is physically speaking not consistent (the concentration of a species cannot be negative – it can be equal to 0 at a minimum).

Overall, as conclusion, the optimal sensor placements suggested by Algorithm (original) [8] and Algorithm 1 (jitter) are the more accurate and efficient ones as they sensed the space in an optimal way.

Impact in data assimilation

The different $MSE(c)$ and the associated reduction in error r_e for the different sensor placements are shown in Table 3.

Although the random sensors placement has a reduction in error of one order of magnitude (rendering of about 86% and 82% error reduction) in sub-domain 6 and 8 respectively, the global error of the predicted model is still quite high. In sub-domain 6, the order of magnitude of the error is not reduced when using the optimal sensors provided by Algorithm (original) [8] and Algorithm 1 (abs.), while it is decreased by two order of magnitude when using the ones from Algorithm 1 (jitter). For sub-domain 8, the error is reduced by one order of magnitude when using the optimal sensors provided by Algorithm (original) [8] and Algorithm 1 (jitter) and up to four order of magnitude when using the ones from Algorithm 1 (abs.). The percentage of error reduction is about 72% and 77% in sub-domain 6 and 8 respectively when using Algorithm (original) [8]. This reduction in error is even better when using Algorithm 1 (jitter) reaching reduction of 97% in sub-domain 6. The use of Algorithm 1 (abs.) gives a very good reduction in error in sub-domain 8 (99%), however the reduction is quite poor for sub-domain 6 (18%).

Table 4
Coordinates of the 8 sensors optimally positioned in sub-domain 6 and 8 computed using Algorithm 1 (jitter) using the traditional Gaussian Process approach and the novel Weak Constraint Gaussian Process proposed in this paper. The results are rounded to two decimals.

	Sub-domain six				Sub-domain eight		
	X	Y	Z		X	Y	Z
Gaussian Process							
(6.1)	−112.65	−49.66	43.44	**(8.1)**	155.89	−532.08	19.09
(6.2)	−93.19	−48.22	62.22	**(8.2)**	−13.44	−280.54	18.24
(6.3)	−94.41	−11.56	116.93	**(8.3)**	22.85	−354.75	18.83
(6.4)	−122.06	−73.90	72.66	**(8.4)**	−14.11	−167.59	3.53
Weak Constraint Gaussian Process							
(6.1)	−125.17	−72.99	54.71	**(8.1)**	−47.81	−351.21	26.19
(6.2)	−117.08	10.12	60.39	**(8.2)**	90.99	−201.51	32.18
(6.3)	−105.97	−56.00	91.75	**(8.3)**	−137.72	−151.44	7.79
(6.4)	−116.51	−72.27	83.40	**(8.4)**	12.08	−409.59	6.15

In conclusion, the two algorithms performing the best in both sub-domains at the same time are Algorithm (original) [8] and Algorithm 1 (jitter), with Algorithm 1 (jitter) giving the best balanced reduction in error.

5.3.2. Weak Constraint Gaussian Process vs. Gaussian Process

In this section, results and performance of the traditional Gaussian Process (GP) and the novel Weak Constraint Gaussian Process (WCGP) proposed in this paper are compared using the (Algorithm 1 (jitter)) as it has been proved in Section 5.3.1 to be the more reliable approach. The coordinates of the optimal sensor locations are given in Table 4 and visualisations of these placements can be seen in Fig. 5. Visually, in sub-domain 6, it can be seen that the optimal sensors placement provided using the traditional GP is quite close to the one predicted using a WCGP. In sub-domain 8: the sensors (8.2) suggested by the GP and (8.1) suggested by the WCGP are quite close to each other and similarly, the sensors (8.3) (GP) and (8.4) (WCGP) are predicted in the same area.

Gaussian Process Validation

The validation results of the Gaussian Process can be seen in Fig. 6. *Real* values are the mean value of the real pollution observations and *Prediction* values are the mean values of the Gaussian Process predictions when using a WCGP. The comparison of the *Real* and *Prediction* mean values are shown for 20 nodes in each sub-domain. It can be seen that the predicted mean concentration values is very close to the real one, i.e. same order of magnitude, in both sub-domains, suggesting that the optimal sensor place-

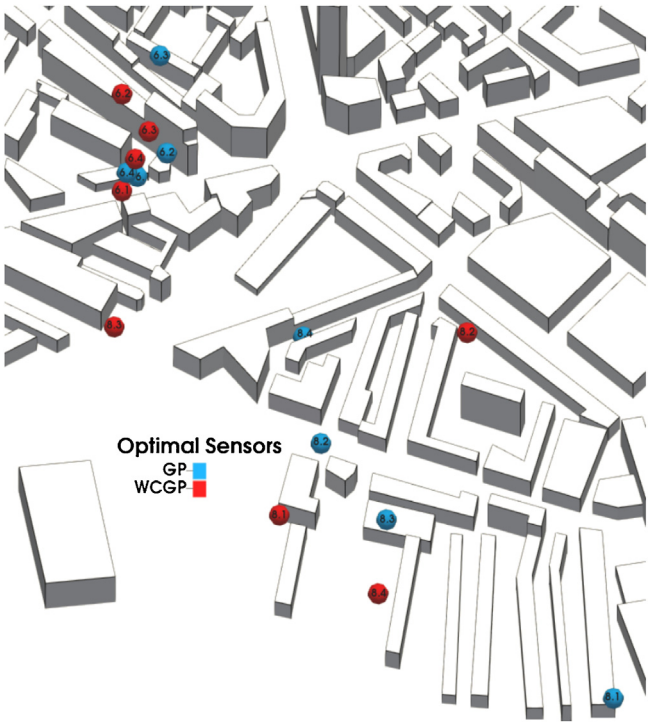


Fig. 5. Optimal sensor placements when using a traditional Gaussian Process (GP) approach (blue spheres) or a Weak Constraint Gaussian Process (WCGP) approach (red spheres) in Algorithm 1 (jitter). The numbers within the spheres denote the sensors IDs as given in Table 4. Note that the radius of the spheres have been intentionally exaggerated for visualisation purposes.

ment provided by the WCGP is sensing the entire space of the sub-domains in a reliable and optimal way.

Impact in data assimilation

The different $MSE(c)$ and the associated reduction in error r_e for the different sensor placements obtained when using a traditional GP and a WCGP are shown in Table 5. The observed data are assimilated at the optimal sensor locations given in Table 4.

In sub-domain 6, the error is reduced by four and two order of magnitudes when using a GP or a WCGP respectively, rendering a global error reduction equal to 99% in both cases. This result is not surprising as the predicted sensor positions are quite close to each other. In sub-domain 8, the error is reduced by one order of magnitude for both method. However, the reduction in error when using a traditional GP is equal to 93%, while the error is reduced by up to 98% when using a WCGP. In sub-domain 8, the WCGP is clearly performing better than the traditional GP. Globally, using the new WCGP approach presented in this paper, allows us to improve in a more effective way our predicted model when using data assimilation.

5.3.3. Scalability of WCGP

The parameter p in Eq. (37) denotes the rank of the partition which constitutes the decomposition. If we consider the WCGP 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*:

$$Measured \quad S_p = \frac{T^1_{\mathcal{A}}(N)}{T^p_{\mathcal{A}}(N)}. \tag{43}$$

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 yields:

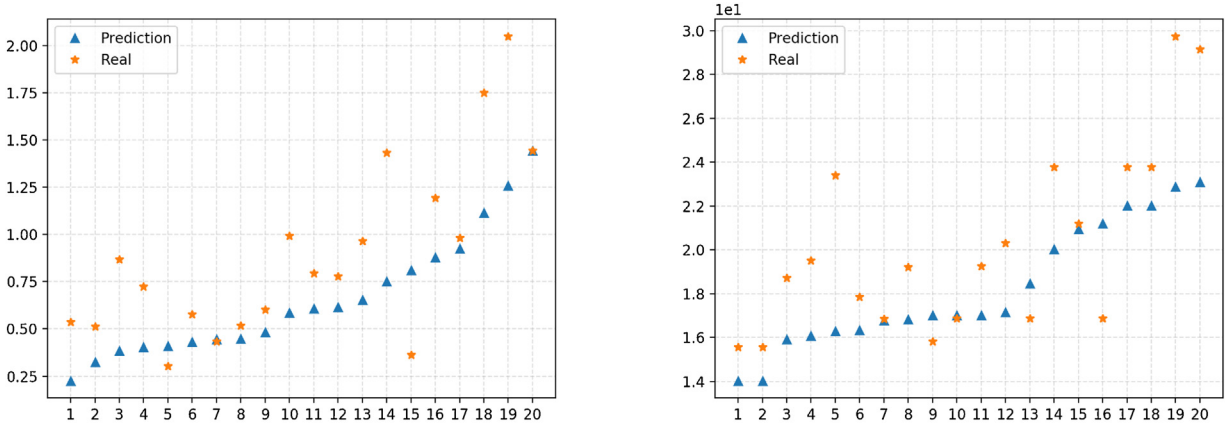
$$T^p_{\mathcal{A}} = T^p_{flop}(N) + T^p_{comm}(N) \tag{44}$$

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

$$T^p_{flop}(N) = \tau(\mathcal{A}(\Omega, p)) \quad t_{flop} \tag{45}$$

In absence of overlapping regions, i.e. in absence of communication among the processors, by using Eqs. (44) and (45) in Eq. (43), we clearly obtain: $Measured \quad S_p \simeq S_p$.

Here we analyse the scalability of our Weak Constraint Gaussian Process (WCGP) as implemented in Algorithm 1 (jitter) with respect to the number of processing cores p. The data set consists of the pollutant concentration values in sub-domain 6 (13, 898 nodes) and sub-domain 8 (12, 366 nodes) simulated for 250 time steps as we know this does not affect the generality of this study. We computed the sensor placements with 2, 4, 8 and 16 cores as can be seen in Table 6 for $k_p = k = 4$. Here, T_p is the time taken and $Measured \quad S_p$ is defined in (43).



(a) Sub-domain 6 (b) Sub-domain 8

Fig. 6. Mean values of the pollutant concentration for 20 chosen nodes in (a) sub-domain 6 and (b) sub-domain 8. *Real* values are the mean value of the real pollution observations. *Prediction* values are the mean values of the Gaussian Process predictions when using a WCGP. x-axis is the node ID and y-axis is the mean pollutant concentration.

Table 5
Mean square errors of the predicted model before ($MSE(c^{\mathcal{M}})$) and after ($MSE(c^{\mathcal{M}})$) data assimilation. *MSE* computed using observation at the optimal sensor placements suggested by the Algorithm 1 (jitter) when using a Gaussian Process (GP) or a Weak Constraint Gaussian Process (WCGP). r_e is the reduction in error in %.

	Sub-domain six			Sub-domain eight		
	$MSE(c^{\mathcal{M}})$	$MSE(c^{DA})$	r_e	$MSE(c^{\mathcal{M}})$	$MSE(c^{DA})$	r_e
Algorithm 1 (jitter) - **GP**	9.98e−01	8.95e−05	99	5.32e−02	3.68e−03	93
Algorithm 1 (jitter) - WCGP	8.86e−01	3.74e−03	99	9.60e−02	2.07e−03	98

Table 6
Comparison of computational time needed to place 4 sensors in sub-domain 6 when using our novel Weak Constraint Gaussian Process (WCGP) Algorithm 1.

p	Execution time: T_p (mins)	Scaling factor: *Measured* S_p
1	2989	1
2	1563	1.9
4	651	4.6
8	365	8.2
16	225	13.3

As it can be seen, the results show that the scaling factor of our algorithm is *Measured* $S_p \simeq p$ as expected. For $p = 4$ and $p = 8$, we observe an improvement in term of scalability. It is due to the reduction in the data transferred from/to the memories and it is given by the efficient memory accesses in the Intel's architecture we are using. We do not have the same improvement when using 16 cores as, for 16 cores, we involve two cards and there is an overhead given by the memories communications. In fact, the measured scaling factor for Algorithm 1 implemented on 16 cores is 13.3, which is still a good results as the optimal sensor locations is computed in few minutes only (less than 5 min for $p = 16$) instead of hours.

6. Conclusion

This paper has successfully developed and implemented the Weak Constraint Gaussian Process (WCGP) developed in [7] and applies the WCGP to optimally place sensors in large datasets.

The issue of scale was solved by decomposing the domain and then parallelising computations on relevant sub-domains. The final solution was validated on a real test-site located in London and including 767,559 nodes. Our algorithm overtakes the GP's practical limit which is said to be around 10, 000 nodes. The non-stationary and non-isotropic prior covariance matrix necessary for optimal results was obtained by taking the sample covariance matrix across all time-steps in the simulation data. Normally, estimating such a covariance matrix is very difficult. However, as we were using simulation data, we were able to take the sample covariance matrix across the time-steps to get a very good estimate of the real covariance matrix. We extended the base algorithm proposed in [8]. This new algorithm combines the priority queue with the local kernel approach. Hence, it is just as precise as the local kernel approach, but much faster. Further, the number of optimisation and hyper-threading methods were used for further optimisation. We found that solving numerical stability issues by adding a jitter to the covariance matrix is much better than taking absolute values of conditional variances. We developed a comprehensive validation methodology that measures the performance of sensor placements by calculating the mean prediction error of the posterior GP. It works by fitting sensor placements to the prior covariance matrix to obtain a posterior mean and covariance function. These can then be used to predict pollution values at every node in the domain of interest. The mean of this prediction is compared to the real mean of pollution values to see how far predictions of the Gaussian Process are off. Overall, the Weak Constraint Gaussian Process (WCGP) approach is a reliable, accurate and efficient way to find the optimal sensor locations as it can also reduced the error in a predicted model in a more significant way than the traditional GP. It has also be shown that our parallel approach is scalable.

The algorithm and overall methodology described in this paper can be used in several other applications in which selecting the most expressive data of a very large set is of interest. For instance, in evolutionary algorithms, generating the first initial population is generally done randomly. The sensor selection algorithm could be used instead to ensure the best possible variety in optimal solutions.

Declaration of interests

None declared.

Acknowledgments

This work is supported by the EPSRC Grand Challenge grant "Managing Air for Green Inner Cities" (MAGIC) EP/N010221/1, by the EPSRC Centre for Mathematics of Precision Healthcare EP/N0145291/1 and by the Health assessment across biological length scales for personal pollution exposure and its mitigation (INHALE) EP/T003189/1.

References

[1] N. Cressie, Statistics for Spatial Data, Wiley, 1993.
[2] L. Neil, Probabilistic non-linear principal component analysis with gaussian process latent variable models, Mach. Learn. Res. 6 (2005) 1783–1816.
[3] M. Osborne, A. Rogers, A. Ramchurn, S. Roberts, N.R. Jennings, Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes, International Conference on Information Processing in Sensor Networks, 2008 (2008).
[4] D. Xiao, F. Fang, J. Zheng, C.C. Pain, I.M. Navon, Machine learning-based rapid response tools for regional air pollution modelling, Atmos. Environ. 199 (2019) 463–473.
[5] D. Xiao, C.E. Heaney, L. Mottet, F. Fang, W. Lin, I.M. Navon, Y. Guo, O.K. Matar, A.G. Robins, C.C. Pain, A reduced order model for turbulent flows in the urban environment using machine learning, Build. Environ. 148 (2019) 323–337.
[6] C. Guestrin, A. Krause, A. Singh, Near-optimal sensor placements in gaussian processes, In 22nd International Conference on Machine Learning (ICML) (2005) 265–272.
[7] R. Arcucci, D. Mcllwraith, Y.-K. Guo, Scalable weak constraint gaussian processes, in: J.M.F. Rodrigues, P.J.S. Cardoso, J. Monteiro, R. Lam, V.V. Krzhizhanovskaya, M.H. Lees, J.J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2019, Springer International Publishing, 2019, pp. 111–125.
[8] A. Krause, A. Singh, C. Guestrin, Near-optimal sensor placements in gaussian processes: theory, efficient algorithms and empirical studies, J. Mach. Learn. Res. 9 (February) (2008) 235–284.
[9] J. Song, S. Fan, W. Lin, L. Mottet, H. Woodward, M. Davies Wykes, R. Arcucci, D. Xiao, E. Aristodemou, M. Carpentieri, M. Herzog, G. Hunt, R. Jones, C. Pain, D. Pavlidis, A. Robins, A. Short, J.-E. Debay, H. ApSimon, P. Linden, Natural ventilation in cities: the implications of fluid mechanics, Build. Res. Inf. 46 (8) (2018) 809–828.
[10] A. Girard, R. Murray-Smith, Learning a Gaussian Process Model with Uncertain Inputs, Department of Computing Science, University of Glasgow, 2003, Technical Report TR-2003-144.
[11] A. McHutchon, C.E. Rasmussen, Gaussian process training with input noise, Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11) (2011) 1341–1349.
[12] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.
[13] R. Arcucci, L. D'Amore, J. Pistoia, R. Toumi, A. Murli, On the variational data assimilation problem solving and sensitivity analysis, J. Comput. Phys. 335 (2017) 311–326.
[14] C.K.I. Williams, M. Seeger, Using the nyström method to speed up kernel machines Advances in Neural Information Processing Systems, 13, MIT Press, 2000, pp. 682–688.
[15] D.G. Cacuci, I.M. Navon, M. Ionescu-Bujor, Computational Methods for Data Evaluation and Assimilation, CRC Press, 2013.

[16] G. Yarin, M. van der Wilk, C.E. Rasmussen, Distributed variational inference in sparse gaussian process regression and latent variable models Advances in Neural Information Processing Systems, 27, 2014, pp. 3257–3265.
[17] M. Titsias, N. Lawrence, Variational learning of inducing variables in sparse gaussian processes, 12th International Conference on Artificial Intelligence and Statistics (AISTATS) (2009).
[18] M. Titsias, N. Lawrence, Bayesian gaussian process latent variable model, 13th International Conference on Artificial Intelligence and Statistics (AISTATS) (2010).
[19] M.P. Deisenroth, J.W. Ng, Distributed gaussian processes, in: Proceedings of the International Conference on Machine Learning (ICML), Lille, France, 2015.
[20] G.C. Fox, R.D. Williams, P.C. Messina, Parallel Computing Works! Morgan Kaufmann, 1994.
[21] C. Park, J.Z. Huang, Y. Ding, Domain decomposition approach for fast gaussian process regression of large spatial data sets, J. Mach. Learn. Res. 12 (2011) 1697–1728.
[22] D. Xiao, C.E. Heaney, F. Fang, L. Mottet, R. Hu, D.A. Bistrian, E. Aristodemou, I.M. Navon, C.C. Pain, A domain decomposition non-intrusive reduced order model for turbulent flows, Comput. Fluids 182 (2019) 15–27.
[23] D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and vlsi, J. ACM (JACM) 32 (1) (1985) 130–136.
[24] H. González-Banos, A randomized art-gallery algorithm for sensor placement, Proceedings of the Seventeenth Annual Symposium on Computational Geometry (2001) 232–240.
[25] C.-W. Ko, J.L.M. Queyranne, An exact algorithm for maximum entropy sampling, Oper. Res. 43 (4) (1995) 684–691.
[26] M.C. Shewry, H.P. Wynn, Maximum entropy sampling, J. Appl. Stat. 14 (2) (1987) 165–170.
[27] A. O'Hagan, Curve fitting and optimal design for prediction, J. R. Stat. Soc.: Ser. B (Methodol.) 40 (1) (1978) 1–24.
[28] N. Ramakrishnan, C. Bailey-Kellogg, S. Tadepalli, V.N. Pandey, Gaussian processes for active data mining of spatial aggregates, Proceedings of the 2005 SIAM International Conference on Data Mining (2005) 427–438.
[29] AMCG, Fluidity Manual (Version 4.1), 2015.
[30] E. Aristodemou, T. Bentham, C. Pain, R. Colvile, A. Robins, H. ApSimon, A comparison of mesh-adaptive LES with wind tunnel data for flow past buildings: mean flows and velocity fluctuations, Atmos. Environ. 43 (39) (2009) 6238–6253.
[31] M. Asch, M. Bocquet, M. Nodet, Data Assimilation: Methods, Algorithms, and Applications, 11, SIAM, 2016.
[32] S. Cuomo, R. Farina, A. Galletti, L. Marcellino, An error estimate of gaussian recursive filter in 3dvar problem, in: 2014 Federated Conference on Computer Science and Information Systems, IEEE, 2014, pp. 587–595.
[33] C.E. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (3) (1948) 379–423.
[34] P.E. Latham, Y. Roudi, Mutual Information, 2009.
[35] W.F. Caselton, J.V. Zidek, Optimal monitoring network designs, Stat. Probab. Lett. 2 (4) (1984) 223–227.
[36] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functions-i, Math. Program. 14 (1) (1978) 265–294.
[37] A. Krause, A. Singh, C. Guestrin, Near-Optimal Sensor Placements in Gaussian Processes Presentation Slides, 2008.
[38] J. Franke, A. Hellsten, H. Schlünzen, B. Carissimo, Best Practice Guideline for the CFD Simulation of Flows in the Urban Environment, Meteorological Institute, 2007.
[39] R. Arcucci, L. Mottet, C. Pain, Y.-K. Guo, Optimal reduced space for variational data assimilation, J. Comput. Phys. 379 (2019) 51–69.
[40] C. Pain, A.P. Umpleby, C.R.E. De Oliveira, A.J.H. Goddard, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Comput. Methods Appl. Mech. Eng. 190 (2001) 3771–3796.
[41] D. Pavlidis, G.J. Gorman, J.L.M.A. Gomes, C. Pain, H. ApSimon, Synthetic-Eddy method for urban atmospheric flow modelling, Bound.-Layer Meteorol. 136 (2010) 285–299.
[42] E. Aristodemou, R. Arcucci, L. Mottet, A. Robins, C. Pain, Y.-K. Guo, Enhancing CFD-LES air pollution prediction accuracy using data assimilation, Build. Environ. 165 (2019) 106383.
[43] P. Marc, A. Deisenroth, Aldo Faisal, Ong. Cheng Soon, Mathematics for Machine Learning, Cambridge University Press, 2019, Available from: https://mml-book.com [Online].

Tolga Hasan Dur is a Machine Learning Engineer in a startup in London. He studied Economics at Goethe University of Frankfurt in Germany and at Keio University in Japan. Furthermore, he completed his MSc in Computer Science at Imperial College London with distinction with a project on "Parallel Gaussian Processes for Optimal Sensor Placement" as part of the DataLearning working group at Data Science Institute.

Dr Rossella Arcucci is a Research Fellow at Data Science Institute, Department of Computing, Imperial College London. She leads and coordinates the activities of the DataLearning Working Group and supervises MSc students and early career researchers. She works on numerical and parallel techniques for accurate and efficient Data Assimilation by exploiting the power of Machine Learning models. Efficiency is achieved by virtue of designing models specifically to take full advantage of massively parallel computers. PhD in Computational and Computer Science on February 2012. The subject of her thesis was Data Assimilation (DA). Her expertise covers the main models for Data Assimilation and Machine Learning. She received the acknowledgement of Marie Sklodowska-Curie fellow from European Commission Research Executive Agency in Brussels on the 27th of November 2017.

Dr Laetitia Mottet is a Post-Doctoral Research Associate at Imperial College London. She received an engineering degree in Thermal Sciences and Energy from Polytech'Nantes, France in 2012 and a Ph.D degree in Fluid Mechanics from the INP Toulouse, France in 2016. After her PhD she joined the University of Cambridge and the Imperial College London in UK to work on two EPSRC projects (LoHCool EP/N009797/1 and MAGIC EP/N010221/1). Her expertises are on computational fluid dynamic applied to urban environment and indoor-outdoor exchanges, as well as meshing technics for complex geometries. Previously, during her Ph.D at the Institut de Mécanique des Fluides de Toulouse (IMFT), she developed a novel numerical approach to access the heat and mass transfers within the capillary (porous) evaporator of a two-phase loop.

Dr Miguel Molina-Solana is a Marie Curie Research Fellow with Universidad de Granada and an Honorary Research Fellow with the Department of Computing, Imperial College London. Before, he was a Marie Curie Research Fellow at Imperial, where he led the Data Visualisation group at its Data Science Institute from 2016 to 2019. He received his PhD in Computer Science from Universidad of Granada in 2012. His research experience comprises applied work in the areas of Data Science, Machine Learning and Knowledge representation.

Prof Christopher Pain received a B.Sc. degree in Mathematics in 1988, from University of Reading, UK, an M.Sc. degree in Mathematical Techniques for CAD in 1989 from Cranfield University, UK, and received a Ph.D. degree in Computational Fluid Dynamics in 1991 from University of Exeter, UK. He leads the Applied Modelling and Computation Group (AMCG) at ICL- the largest research group at Imperial College, comprises of about 60 scientists and recipient of the ICL Research Excellence award in 2011. Award in recognition of its high academic achievement and significant future potential. He is visiting Prof at BU and USC, the director of the data assimilation (DA) lab. in the Data Science Institute (DSI) at ICL, leads the modelling for the MEMPHIS and MAGIC consortia and is PI of SmartGeoWells Newton consortium. He developed new balanced mixed finite elements for multi-phase flow and geophysical fluid dynamics. >180 journal papers, graduated 42 PhD students, completed 42 industry and research council grants, and won L23.5M in funding over 15 years.

Prof Yi-Ke Guo is a Professor of Computing Science in the Department of Computing at Imperial College London. He is the founding Director of the Data Science Institute at Imperial College. He is a Fellow of the Royal Academy of Engineering (FREng), Member of Academia Europaea (MAE), Fellow of British Computer Society and a Trustee of The Royal Institution of Great Britain. Professor Guo received a first-class honours degree in Computing Science from Tsinghua University, China, in 1985 and received his PhD in Computational Logic from Imperial College in 1993. He has been working on technology and platforms for scientific data analysis since the mid-1990s, where his research focuses on data mining, machine learning and large-scale data management. He has published over 200 papers on data analysis/applications, educated 80 PhD students & won L120M over 15 years. He has published over 250 articles, papers and reports. Projects he has contributed to have been internationally recognised, including winning the "Most Innovative Data Intensive Application Award" at the Supercomputing 2002 conference for Discovery Net, the Bio-IT World "Best Practices Award" for U-BIOPRED in 2014 and the "Best Open Source Software Award" from ACM SIGMM in 2017.