Journal of Computational Physics 520 (2025) 113460

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A Kronecker product accelerated efficient sparse Gaussian Process (E-SGP) for flow emulation

Yu Duan [*], Matthew Eaton, Michael Bluck

Nuclear Engineering Group, Department of Mechanical Engineering, City and Guilds Building (CAGB), Imperial College London, Exhibition Road, South Kensington Campus, SW7 2BX, United Kingdom

A R T I C L E I N F O

Keywords:
Bayesian Machine Learning
Gaussian processes
Variational energy free Gaussian process
Structured & unstructured fluid mechanics datasets

A B S T R A C T

In this paper, we introduce an efficient sparse Gaussian process (E-SGP) for the surrogate modelling of fluid mechanics. This novel Bayesian machine learning algorithm enables efficient model training using databases of different structures. It extends the approximated sparse GP algorithm by combining the concept of efficient GP (E-GP) and variational energy free sparse Gaussian processes (VEF-SGP). The developed E-SGP approach leverages the arbitrariness of inducing points and the monotonically increasing nature of the objective function with respect to the number of inducing points in VEF-SGP. By specifying the inducing points on the orthogonal grid/input subspace and using the Kronecker product, E-SGP significantly enhances computational efficiency without imposing constraints on the covariance matrix or increasing the number of parameters that need to be optimised during training.

The E-SGP algorithm outperforms E-GP not only in terms of scalability but also in model quality, as evidenced by mean standardised logarithmic loss (MSLL). The computational complexity of E-GP grows cubically with an increasing structured training database, while E-SGP maintains computational efficiency as the model resolution (i.e., the number of inducing points) remains the same. The examples show that E-SGP produces more accurate predictions compared to E-GP when model resolutions are similar. In the application to a partially structured database, E-GP benefits from more training data but comes with higher computational demands, while E-SGP achieves a comparable level of accuracy but is more computationally efficient, making E-SGP a potentially preferable choice for fluid mechanics. Furthermore, E-SGP can produce more reasonable estimates of model uncertainty, whilst E-GP is more likely to produce over-confident predictions. In the application to partially structured databases, the performance of E-SGP is also compared to the structured Gaussian processes latent variable model (SGPLVM). In this case, E-SGP and SGPLVM both exhibit robust performance. However, E-SGP demonstrates a slight advantage over SGPLVM, particularly in terms of MSLL, indicating that E-SGP is better at producing trustworthy and accurate estimates of the uncertainty in the predictions. . While SGPLVM also improves computational efficiency by exploiting the Kronecker product and a small number of inducing points, E-SGP's orthogonal grid specification of inducing points further enhances its scalability and robustness, making it a superior choice for complex fluid mechanics problems with partially structured data.

[*] Corresponding author.
E-mail address: y.duan@imperial.ac.uk (Y. Duan).

https://doi.org/10.1016/j.jcp.2024.113460
Received 13 December 2023; Received in revised form 1 August 2024; Accepted 26 September 2024
Available online 27 September 2024

1. Introduction

Fluid mechanics is pivotal to industries as diverse as transport, power generation, pharmaceutical industries, and more. Physics-based approaches are essential for understanding complex, three-dimensional (3D) turbulent thermo-fluid phenomena. However, it remains challenging to use a physics-based methodology to efficiently address the challenges posed in performing extensive parametric CFD simulations, e.g., uncertainty quantification (UQ), design exploration and optimisation. There is no doubt that the analysis of complex 3D turbulent thermo-fluid behaviour within complex engineering systems is a computationally demanding field of engineering involving very large data sets. This makes it an ideal field of engineering application to apply machine learning (ML) algorithms for producing surrogate models as well as performing design optimisation, data mining and anomaly or fault detection [1,2]. The successful, and disruptive, impact of ML algorithms in applications such as language and image processing [3], advertising [4], and finance [5], has attracted much attention from the fluid mechanics community. Developing interpretable, generalisable, and robust ML algorithms has become one of the goals of ML research related to fluid mechanics. Many reviews have been published recently on the development and application of ML in fluid mechanics, e.g. [6–8]. In this paper, we propose an efficient analytic Bayesian ML algorithm for learning the complex flow dynamics from structured and unstructured computational mesh databases.

1.1. *Why Bayesian?*

The success of deterministic ML models, e.g., deep neural networks (DNNs), is often plagued by several issues, such as overfitting due to small or noisy datasets [9]. Both computational and experimental fluid mechanics often produce very large data sets, or quantities of interest (QoI), such as temperature, pressure, velocity and other associated thermo-fluid data. However, the production of high-fidelity computational and experimental data can be costly to produce in terms of computational resources as well as financially in terms of the cost of thermo-fluid experiments. Consequently, the amount of high-fidelity thermo-fluid data available may be small compared to the complexity of the problem being analysed. Furthermore, the lack of reliable confidence estimates and other robustness issues in deterministic ML makes it vulnerable to adversarial attacks [10] and also reducing the models' reproducibility [11]. As we know, the confidence/uncertainty quantification (UQ) of model predictions is essential for reliability, and risk assessment, in safety-critical industries [12]. Finally, prior knowledge of fluid mechanic problems (e.g., the governing equations or empirical knowledge) are often available. Therefore, this makes it an attractive field of research for the application of Bayesian ML algorithms.

1.2. *Motivation for the development and use of Gaussian process based Bayesian ML algorithms*

Gaussian process (GP) based methods represent a set of powerful analytical Bayesian ML algorithms. It assumes that any combination of observations of a complex system forms an adjoint multivariate Gaussian distribution. The key to GP model training is to find the appropriate covariance matrix that maximises the marginal likelihood. The predictions of the GP model can then be interpreted as using existing knowledge to calculate the conditional distribution of function outputs at unknown points. This conditional distribution also has the form of a Gaussian distribution, and thus the prediction of the GP model consists of two parameters, the mean, and the variance. The variance can be used to calculate confidence intervals of model predictions. Even more attractive is the fact that the entries in the covariance matrix are computed using the covariance kernel function (kernel function for short), which can be re-derived from the PDF of the problem that is being analysed. This enables the integration of the governing physics laws within the algorithm, e.g. Latent force models (LFM) [13–17], numerical GP [18] and GP for linear [19] and non-linear PDFs [20]. (For the sake of clarity, we will name the original definition of GP as standard GP or STD-GP in the following discussion.)

1.3. *Background*

Covariance matrix inversion and storage are two major obstacles to the general application of Gaussian processes (GP) to engineering and thermo-fluid problems involving very large data sets. For a standard GP model with N training data points, the computational complexity is (N^3) and the storage requirement is (N^2). For a problem with more than 10^4 training data points, STD-GP becomes impractical. Therefore, recent research into GP methods has focussed on developing algorithms that address this challenge. Various scalable GPs have been developed to improve the scalability of the method. Recently, Liu et al. [21] conducted a comprehensive review of the scalable GP, in which they classified the scalable GPs into two major categories: 1) global approximation and 2) local approximation. The global approximation is further divided into 1.1) subset of data, 1.2) sparse kernel, and 1.3) sparse approximation. The work in this paper is consistent with the sparse approximation scalable GPs.

To improve the scalability of the STD-GP, the sparse approximate inference GP method adopts the concept of the inducing inputs e.g. [22–25]. The inducing points is a set of representative points being used to summarise the information from the data. Titsias [26] proposed the variational inducing points sparse GP or VEF-SGP method (the name of this approach refers to [21]). VEF-SGP treats the outputs of a function at inducing points as latent variables and derives a procedure for approximating latent variables and hyper-parameters. The method reduces the risk of over-fitting by introducing an extra regulation term in the objective function. Hensman et al. [27,28] scales up the VEF-SGP using stochastic variational inference (SVI) and generalised the approach to classification problems. Although sparse approximate inference GP methods greatly improve the efficiency of STD-GP, the limitation is also obvious. The number of inducing point is usually restricted to a number smaller than (10^4), which again may not be enough for a complex and high-dimensional problem.

The STD-GP can also be accelerated by exploiting the structure of the training dataset. Saatçi [29] proposed an efficient GP (E-GP) that utilises kernel functions in the form of tensor products and a structured training dataset. In E-GP, the covariance matrix is calculated as the Kronecker product of the sub-covariance matrices for each input. Similar approaches have been applied to accelerate the sparse approximate inference GP approaches. Wilson and Nickisch [30] proposed KISS-GP which uses the structured kernel interpolation (SKI) to approximate the covariance matrix. A development of KISS-GP for the product kernel function can be found in [31], while the online version of the KISS-GP, WISKI-GP was developed by Stanton et al. [32]. The scalability of the SKI-GP method is further improved in [33]. Undoubtedly, the adoption of the SKI approaches greatly improves the time efficiency of GP model training, but it may also lead to discontinuity and overconfidence in model prediction [21], which are unfavourable for fluid mechanics problems.

Izmailov, Noikov and Kropotov [34] proposed the TT-GP which uses the tensor train decomposition for the high-dimensional dataset, and also assumes that the covariance matrix of the inducing points can always be factorised using a Kronecker product. However, this assumption restricts the covariance matrix representation [35] and also increases the number of parameters that need to be optimised during training (e.g. entries in the sub-covariance matrices of inducing points). Atkinson and Zabaras [36] proposed an unsupervised learning algorithm, structured Gaussian process latent variable model (SGPLVM), based on the Bayesian GP-LVM for datasets of which inputs are defined as the cartesian product of the spatial and latent variable inputs. SGPLVM is computationally tractable and achieves a complexity similar to the SKI-GP algorithm without placing strict constraints on the covariance matrix or greatly increasing the number of parameters. Although our work shares some similarities with SGPLVM [36], such as being based on the variational sparse Gaussian process, VEF-SGP, and applying the Kronecker product to improve computational efficiency, there are distinct differences that set our approach apart. First, SGLVM [36] focuses on problems where the input can be decomposed into a Cartesian product of the spatial input space and other inputs, such as temporal or stochastic input spaces. Our work focuses on providing an efficient Bayesian type surrogate modelling algorithm for high-dimensional data typical in engineering applications, where the inputs are usually known. Second, the inducing points in the SGPLVM must possess the same structure as the observations. It will impose great limitation on its applications on a large dataset with unstructured spatial inputs, which is normally seen in engineering applications, especially, for fluid dynamic problems. In our E-SGP, we will explicitly derive the objective function and predictions with respect to the inducing inputs which are defined on the Cartesian grid.

In the following context, we refer to this novel VEF-GP approach as E-SGP. The rest of this paper is organised as follows. In Section 2, the mathematics of STD-GP, VEF-GP, as well as the E-GP are reintroduced, while the mathematical description of the E-SGP is included in Section 3. Applications of the E-SGP to both spatially structured and unstructured mesh fluid dynamics datasets are discussed in Section 4. We compare the quality and scalability of E-SGP to those of E-GP for a fully structured dataset in Section 4.1. In Section 4.2, the performance of E-SGP, E-GP, and SGPLVM is compared for cases where the spatial input of the training data is unstructured. Section 5 summaries our work and discusses the potential of E-SGP.

2. STD-GP, VEF-GP and E-GP revisited

In this section, we begin with a mathematical description of the STD-GP followed by a concise description of the variational learning sparse GP (VEF-GP) [26] and the efficient GP formula [29]. This allows us to introduce notation and derive expressions so that the Kronecker product accelerated efficient SGP (E-SGP) can be formulated. If readers require more information on GPs, the following references are recommended [21,24,37–39].

2.1. Gaussian processes (STD-GP) revisited

In the standard GP (STD-GP), we assume that any finite subset of $\{f(\vec{x})|\vec{x}\in\mathbb{R}^D\}$ follows a multivariate Gaussian distribution. Considering the vector of noisy observations $\vec{y}=\{y_i\in\mathbb{R}\}_{i=1}^n$ of the latent function acquired at n locations$X_n=\left\{\vec{x}_i\in\mathbb{R}^D\right\}_{i=1}^n$, $\vec{y}$ have the priors given by

$$p\left(\vec{y}\middle|\vec{f},X_n\right)=\mathcal{N}\left(\vec{y}\middle|\vec{f},\sigma^2I_{nn}\right),\tag{1}$$

$$p\left(\vec{f}\middle|X_n\right)=\mathcal{N}(\vec{y}|\vec{m}(X_n),K_{nn}).\tag{2}$$

In Eq. (1), $\vec{f}$ is the function outputs at locations X_n. $\vec{m}(X_n)$ denotes the mean function values at X_n. For the sake of simplicity, $\vec{m}(X_n)$ can be fixed to zeros in many GP applications. $K_{nn}=K(X_n,\ X_n)$ in Eq. (2) is an $n\times n$ real symmetric positive semi-definite matrix, whose elements are evaluated using kernel functions ($k(x,x')$). σ represents the noise/error level of $\vec{y}$ and I_{nn} is an n-by-n identity matrix.

The training of the GP data-driven model means optimising the hyperparameters of the kernel function to maximise the logarithmic marginal likelihood $p(\vec{y}|X_n)$, which is written as

$$\log p(\vec{y}|X_n)=-\frac{n}{2}\log 2\pi-\frac{1}{2}\vec{y}\left(\sigma^2I_{nn}+K_{nn}\right)^{-1}\vec{y}^T-\frac{1}{2}\log\left|\sigma^2I_{nn}+K_{nn}\right|,\tag{3}$$

or using the eigen-decomposition of the covariance matrix

$$\log p(\vec{y}|X_n) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\vec{y}V_{nn}\left(\sigma^2 I_{nn} + E_{nn}\right)^{-1}V_{nn}^T\vec{y}^T - \frac{1}{2}\log\left|\sigma^2 I_{nn} + E_{nn}\right|. \tag{4}$$

In Eq. (4), V_{nn} is the square $n \times n$ matrix whose columns are the eigenvectors of K_{nn}, and E_{nn} is the diagonal matrix and its diagonal elements are the corresponding eigenvalues.

Based on the prior assumption of the STD-GP, we can write the joint distribution of the unknown function values $\vec{f}_*$ at the input locations $X_* = \left\{\vec{x}_i^* \in \mathbb{R}^D\right\}$ and noisy observation $\vec{y}$ as

$$\begin{bmatrix} \vec{y} \\ \vec{f}_* \end{bmatrix} \sim N\left(\vec{0}, \begin{matrix} K_{nn} + \sigma^2 I_{nn} & K(X_*, X_n)^T \\ K(X_*, X_n) & K(X_*, X_*) \end{matrix}\right), \tag{5}$$

which also lead to the predictive equations of the STD-GP model:

$$\left\langle \vec{f}_* \right\rangle = K(X_*, X_n)\left(\sigma^2 I_{nn} + K_{nn}\right)^{-1}\vec{y}^T, \tag{6}$$

$$cov\left(\vec{f}_*\right) = K(X_*, X_*) - K(X_*, X_n)\left(\sigma^2 I_{nn} + K_{nn}\right)^{-1}K(X_*, X_n)^T. \tag{7}$$

where $\left\langle \vec{f}_* \right\rangle$ is the expected values of predictions, and $cov\left(\vec{f}_*\right)$ is the covariance matrix of the predictions providing information about how predictions at different points are correlated, with the diagonal elements representing the variances of the predictions at those points. The computational burden of the STD-GP arises from the need to invert n-by-n matrix in Eq. (3) (6) and (7), which are naively O(n^3).

2.2. Efficient Gaussian process (E-GP)

Saatçi [29] proposed an efficient GP (E-GP) algorithm for structured datasets. More specifically, the data are on a multidimensional Cartesian grid $X = X_1 \times X_2 \times \cdots \times X_n$, and the kernel function has a tensor product form. In particular, $k(\cdot,\cdot)$ is a tensor product if it can be written as $k(\vec{x}, \vec{x}') = k(x_1, x_1')k(x_2, x_2')\cdots k(x_n, x_n')$. For inputs defined on X, the covariance matrix K_{nn} can then be written in the form of a Kronecker product:

$$K_{nn} = \otimes_{i=1}^{n} K_i, \tag{8}$$

where K_i is the square covariance matrix of the inputs on the X_i axis. For the sake of clarity, K_i is called a sub-covariance matrix. The eigen-decomposition of K_{nn} can now be expressed as

$$\begin{aligned} &K_{nn} = VDV^T = \otimes_{i=1}^{n}\left(V_i E_i V_i^T\right) = \left(\otimes_{i=1}^{n} V_i\right)\left(\otimes_{i=1}^{n} E_i\right)\left(\otimes_{i=1}^{n} V_i^T\right), \\ &and \\ &K_{nn}^{-1} = \left(\otimes_{i=1}^{n} V_i\right)\left(\otimes_{i=1}^{n} E_i^{-1}\right)\left(\otimes_{i=1}^{n} V_i^T\right) \end{aligned} \tag{9}$$

In Eq. (9), E_i is a diagonal matrix containing the eigenvalues of the K_i.

Eq. (3) is now written as

$$\log p(\vec{y}|X_n) = -\frac{n}{2}\log 2\pi - \frac{1}{2}\vec{y}\left(\otimes_{i=1}^{n} V_i\right)\left(\sigma^2 I + \otimes_{i=1}^{n} E_i\right)^{-1}\left(\otimes_{i=1}^{n} V_i^T\right)\vec{y}^T - \frac{1}{2}\log\left|\sigma^2 I + \otimes_{i=1}^{n} D_i\right| . \tag{10}$$

The computational complexity of $\left(\otimes_{i=1}^{n} V_i^T\right)\vec{y}^T$ can be reduced to (N) using the mixed Kronecker matrix-vector product:

$$\left(B^T \otimes A\right)\vec{y}^T = vec(AYB), \tag{11}$$

in which vec is the vectorisation operator e.g. $\vec{y}^T = vec(Y)$ and $\vec{y}^T$ is a column vector. The computational complexity for evaluating the marginal likelihood represented by Eq. (4) is now reduced to $\mathscr{O}\left(N\sum_{i=1}^{n} n_i\right)$, where $N = \prod_{i=1}^{n} n_i$. A similar approach is also applicable to acceleration of the predictions in (6) and (7).

E-GP can be easily extended to problems where the input space consists of distinct and independent subspaces. In other words, X_i in X is not only interpreted as inputs on an axis but also inputs in an independent subspace. This eases the restrictions of E-GP on the structure of the dataset and is especially useful in fluid mechanics problems. For instance, the inputs of a fluid mechanics problem can be easily divided into several orthogonal subspaces, like spatial inputs, time inputs, operational parameters, design parameters, etc. Whilst better (in performance terms) than the STD-GP, there remain obvious limitation of E-GP (notably the number of inputs on each grid/input subspace must be smaller than (10^4)). This will ultimately hinder its generality in recovering a complex flow field. In real applications, the number of spatial inputs is normally more than (10^4).

2.3. Variational energy free sparse Gaussian process (VEF-SGP)

Let us define a set of inducing variables $\vec{u} = \{u_i\}_{i=1}^m$ defined at $X_m = \left\{\vec{x}_i \in \mathbb{R}^D\right\}_{i=1}^m$. And we also have $m \ll n$. Following the assumptions of GP, the distribution of $\vec{u}$ with knowledge of $\vec{y}$ can be written as $p(\vec{u}|\vec{y})$. To reduce the overfitting risk, the VEF-SGP algorithm introduces a variational distribution of $\vec{u}$, $q(\vec{u}) \sim N(\vec{u}|\vec{\mu}, S_{mm})$, in which $\vec{\mu}$ is the mean vector and S_{mm} is the covariance matrix. In other words, $q(\vec{u})$ is the variational distribution of $p(\vec{u}|\vec{y})$. X_m, $\vec{\mu}$ and S_{mm} together are called variational quantities. Now, the posterior GP mean and variance for the latent function at the unknown location $\vec{x}^*$ w.r.t $q(\vec{u})$ is written as

$$\vec{y}^* = K_{x^*m}K_{mm}^{-1}\vec{\mu}, \tag{12}$$

$$var(y^*) = k\left(\vec{x}^*,\ \vec{x}^*\right) - K_{x^*m}K_{mm}^{-1}K_{mx^*} + K_{x^*m}K_{mm}^{-1}S_{mm}K_{mm}^{-1}K_{mx^*}, \tag{13}$$

where $var(\cdot)$ is the variance of model prediction.

In VEF-SGP, $\vec{\mu}, S_{mm}$ and the hyperparameters are determined by maximising the evidence lower bound (EBLO) of the logarithm marginal likelihood. The EBLO is written as

$$\mathscr{l} = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\vec{y}\left(\sigma^2 I_{nn} + K_{nm}K_{mm}^{-1}K_{mn}\right)^{-1}\vec{y}^T - \frac{1}{2}\log\left|\sigma^2 I_{nn} + K_{nm}K_{mm}^{-1}K_{mn}\right| - \frac{1}{2\sigma^2}tr(\widetilde{K}), \tag{14}$$

where $\widetilde{K} = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}$ and $tr(\widetilde{K}) = tr(K_{nn}) - tr\left(K_{mm}^{-1}K_{mn}K_{nm}\right)$. The optimal parameters of $q(\vec{u})$ ($\vec{\mu}$ and S_{mm}) are written as:

$$S_{mm} = K_{mm}\left(K_{mm} + \sigma^{-2}K_{mn}K_{nm}\right)^{-1}K_{mm} \ and \ \vec{\mu} = \sigma^{-2}S_{mm}K_{mm}^{-1}K_{mn}\vec{y} \tag{15}$$

It is worth noting that the 4th term in Eq. (10), $\frac{1}{2\sigma^2}tr(\widetilde{K})$, is the regularisation term which reduces the overfitting risk and ensures the monotonically increasing nature of $\mathscr{l}$ as the number of inducing points increases.

Both K_{mm} and K_{mm}^{-1} are the real symmetric positive definite matrices, therefore K_{mm}^{-1} can be represented as the product of a lower triangular matrix (L_{mm}) and its transpose using a Cholesky decomposition. Hereafter we write $K_{nm}K_{mm}^{-1}K_{mn} = Q_{nm}Q_{mn}$, where $Q_{nm} = K_{nm}L_{mm}$ and $Q_{mn} = Q_{nm}^T$. Note that $Q_{mn}Q_{nm}$ is also real and symmetric and thus guarantees the orthogonality of its eigen vectors. The eigen decomposition of $Q_{mn}Q_{nm}$ can then be written as $W_{mm}E_{mm}W_{mm}^T$, while E_{mm} is the diagonal matrix containing eigenvalues and the columns of W_{mm} are the eigenvectors of $Q_{mn}Q_{nm}$. Based on the matrix inversion lemma, also known as Woodbury formula, we now have:

$$\begin{aligned}&\left(\sigma^2 I_{nn} + K_{nm}K_{mm}^{-1}K_{mn}\right)^{-1} = \left(\sigma^2 I_{nn} + Q_{nm}Q_{mn}\right)^{-1} = \sigma^{-2}I_{nn} - \sigma^{-2}Q_{nm}W_{mm}\left(\sigma^2 I_{mm} + E_{mm}\right)^{-1}W_{mm}^T Q_{mn},\\ &and\\ &\log\left|\sigma^2 I_{nn} + K_{nm}K_{mm}^{-1}K_{mn}\right| = (n-m)\log\sigma^2 + \log\left|\sigma^2 I_{mm} + E_{mm}\right|.\end{aligned} \tag{16}$$

After subtituting Eq. (16) into Eq. (14), we have the new form of the ELBO

$$\mathscr{l} = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sigma^{-2}\vec{y}\vec{y}^T - \frac{1}{2}(n-m)\log\sigma^2 + \frac{1}{2}\sigma^{-2}\vec{y}Q_{nm}W_{mm}\left(\sigma^2 I_{mm} + E_{mm}\right)^{-1}W_{mm}^T Q_{nm}^T\vec{y}^T - \frac{1}{2}\log\left|\sigma^2 I_{mm} + E_{mm}\right| - \frac{1}{2\sigma^2}tr(\widetilde{K}). \tag{17}$$

The optimal parameters of $q(\vec{u})$ ($\vec{\mu}$ and S_{mm}) in [26] can also be re-written as:

$$S_{mm} = K_{mm} - \sigma^{-2}K_{mn}K_{nm} + \sigma^{-2}K_{mn}Q_{nm}W_{mm}\left(\sigma^2 I_{mm} + E_{mm}\right)^{-1}W_{mm}^T Q_{mn}K_{nm}, \tag{18}$$

$$\vec{\mu} = \sigma^{-2}\left[K_{mn} - \sigma^{-2}K_{mn}Q_{nm}Q_{mn} + \sigma^{-2}K_{mn}Q_{nm}W_{mm}\left(\sigma^2 I_{mm} + E_{mm}\right)^{-1}W_{mm}^T Q_{mn}Q_{nm}Q_{mn}\right]\vec{y}. \tag{19}$$

Except for $\vec{\mu}, S_{mm}$ and the hyperparameters, the quality of the approximation in SGP is also dependent on the locations of the subset-of-data or inducing points in VEF-SGP. Many methods have been proposed to optimise the location of the inducing point with respect to posterior likelihood [25,40,41], information gain [25] or prediction error [42]. However, it will not present a major issue. It is demonstrated in [26], that ELBO ($\mathscr{l}$) (Evidence Lower Bound) of the VEF-SPG converges to the marginal likelihood of the STD-GP as the number of inducing points increases in the case where the inducing points are a subset of the observations. Therefore, it is reasonable to infer that the impact of inducing points can be minimised using a sensitivity analysis similar to the mesh sensitivity study employed in CFD simulations. In this paper, we treat the locations of the inducing points as user-defined and empirically validate our hypothesis regarding their sensitivity.

3. Kronecker product accelerated VEF-SGP (E-SGP)

In this section, we combine concepts of E-GP and VEF-SGP to develop the Kronecker product accelerated SGP (E-SGP) algorithm. In our work, we assume that the input space can be divided into orthogonal subspaces for the specific applications we consider. For instance, the spatial space is orthogonal to the temporal space or other operating parameter spaces, such as inlet velocities. This assumption allows us to exploit the computational advantages of the Kronecker product, significantly enhancing scalability and computational efficiency. However, we recognise that this approach may need adjustments for problems where the input dimensions are not orthogonal.

Suppose that the inputs of the observations $\overrightarrow{y}$ is on $\chi^1 \times \chi^2 \times \cdots \times \chi^s$, $\chi^i \in R^{d_i}$, $i = 1, \cdots,$ where, $\times$ denotes the Cartesian product between subsets. R^{d_i} is an input subspace with dimension d_i, and the total dimension of the input space $d = \sum_{i=1}^{s} d_i$, $i = 1, \cdots, s$. χ^i is a $n_i \times 1$ vector if $d_i = 1$ or a $n_i \times d_i$ matrix otherwise, where n_i is number of inputs in χ^i and the number of observations $n = \prod_{i=1}^{s} n_i$. We also define the inducing points using the form $\chi_u^1 \times \chi_u^2 \times \cdots \times \chi_u^s$. $\chi_u^i \in R^{d_i}$, $i = 1, \cdots, s$ and $m = \prod_{i=1}^{s} m_i$, while m_i is the number of inducing points in χ_u^i. Since the inducing point can be arbitrary, for the sake of the computational simplicity, we can furtherly define that inputs in any χ_u^i always locate on a Cartesian grid, namely $\chi_u^i = X_u^{i,1} \times X_u^{i,2} \times \cdots X_u^{i,d_i}$.

Like E-GP, the kernel functions with the tensor product format will also be used here. Thence, Q_{nm} and $Q_{mn}Q_{nm}$ in the Eq. (16) can be expressed as

$$Q_{nm} = K_{nm}L_{mm} = \otimes_{i=1}^{s}\left(K_{nm}^{i}L_{mm}^{i}\right) and,$$
$$Q_{mn}Q_{nm} = \otimes_{i=1}^{s}\left(Q_{mn}^{i}Q_{nm}^{i}\right) = \otimes_{i=1}^{s}\left[W_{m_i m_i}E_{m_i m_i}W_{m_i m_i}^{T}\right] = \left(\otimes_{i=1}^{s}W_{m_i m_i}\right)\left(\otimes_{i=1}^{s}E_{m_i m_i}\right)\left(\otimes_{i=1}^{s}W_{m_i m_i}^{T}\right). \tag{20}$$

In Eq. (20), K_{nm}^i is the covariance matrix between inputs in χ^i and inducing inputs in χ_u^i while L_{mm}^i is the triangular matrix obtained using Cholesky decomposition for the covariance matrix of inducing inputs in χ_u^i. Furthermore, $L_{mm}^i = \otimes_{j=1}^{d_i} L_{mm}^{i,j}$, as $\chi_u^i = X_u^{i,1} \times X_u^{i,2} \times \cdots X_u^{i,d_i}$. In addition, the regulation term $tr(\widetilde{K})$ in Eq. (17) can now be calculated as

$$tr(\widetilde{K}) = \prod_{i=1}^{s}\left(tr\left(K_{nn}^{i}\right)\right) - \prod_{i=1}^{s}\left(tr\left(\left(K_{mm}^{i}\right)^{-1}K_{mn}^{i}K_{nm}^{i}\right)\right). \tag{21}$$

Hereby, the computational complexity of Eq. (17) is now reduced to $\mathscr{O}\left(N\sum_{i=1}^{n} m_i^2\right)$ and is much more efficient than E-GP when $m_i \ll n_i$. Eq. (17) can be further accelerated as we enforce the inducing points on a Cartesian grid in each input subspace, since K_{mm} can be rewritten in the form of Kronecker product with respect to each input.

The approximated GP mean and variance in Eqs. (12 and 13) can be expend as follows:

$$\overline{y}^{*} = \sigma^{-2}B_{x^*n}\overrightarrow{y} - \sigma^{-4}B_{x^*n}Q_{nm}Q_{mn}\overrightarrow{y} + \sigma^{-4}B_{x^*n}Q_{nm}W_{mm}\left(\sigma^2 I_{mm} + E_{mm}\right)^{-1}W_{mm}^{T}Q_{mn}Q_{nm}Q_{mn}\overrightarrow{y}, \tag{22}$$

$$var(y^*) = k\left(\overrightarrow{x}^{*}, \overrightarrow{x}^{*}\right) - \sigma^{-2}B_{x^*n}B_{x^*n}^{T} + \sigma^{-2}B_{x^*n}Q_{nm}W_{mm}\left(\sigma^2 I_{mm} + E_{mm}\right)^{-1}W_{mm}^{T}Q_{mn}B_{x^*n}^{T}, \tag{23}$$

where $B_{x^*n} = K_{x^*m}K_{mm}^{-1}K_{mn}$.

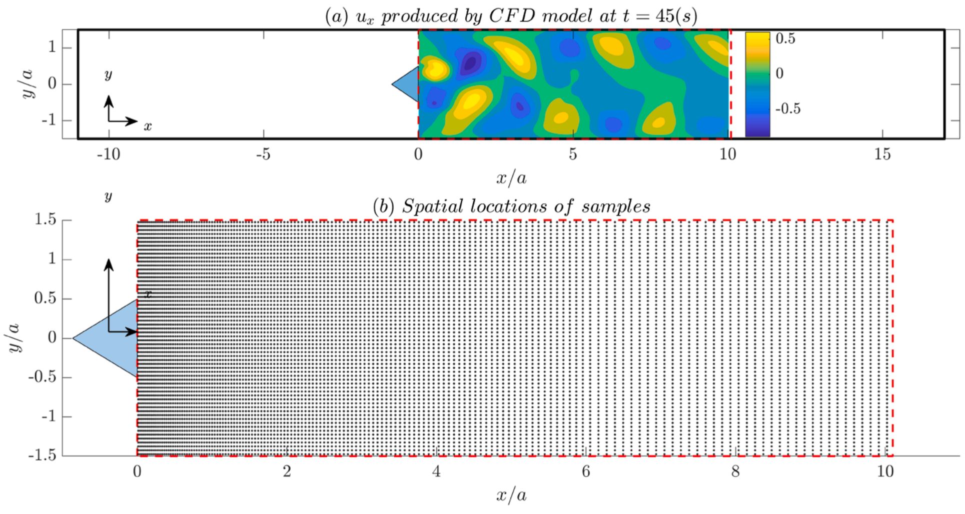

Fig. 1. (a) Cross section of the flow domain, the considered sampling range of training data for data-driven models, as well as a single snapshot of the streamwise fluctuating velocity component (u_x), (b) spatial locations of samples.

Once the unknown inputs has the similar form as the training data, B_{x^*n}, $B_{x^*n}Q_{nm}Q_{mn}$, an $B_{x^*n}Q_{nm}W_{mm}$ in Eq. (22) can be written in the Kronecker product form. The first two term in Eq. (22) can be effectively evaluated using Eq. (11). For the third term in Eq. (22), $(\sigma^2 I_{mm} + E_{mm})^{-1} W_{mm}^T Q_{mn} Q_{nm} Q_{mn} \overrightarrow{y}$ is eventually a $m \times 1$ vector and can be easily calculated suing Eq. (11) and element wise production between a column vectors ($diag\left((\sigma^2 I_{mm} + E_{mm})^{-1}\right)$) and $m \times 1$ vector $W_{mm}^T Q_{mn} Q_{nm} Q_{mn} \overrightarrow{y}$. Similar method can also be applied to accelerate the calculation of GP variance.

4. Experiments

Both E-GP and E-SGP algorithms are used to learn the periodic bluff body flow and the lid-driven cavity flow generated using CFD with different types of meshing strategies. In the first example, the periodic bluff-body flow field is generated using a structured grid (Section 4.1). The structured database allows both algorithms to take full advantage of the Kronecker product. In the second example, E-GP, SGPLVM and E-SGP models are trained to learn the relationships between the moving wall and the flow field in a square cavity (Section 4.2). The quality and scalability of both algorithms will be evaluated on a database with unstructured spatial inputs. It should be noted that the training database is generated using different meshes. E-GP, SGPLVM and E-SGP algorithms are implemented in our in-house MATLAB code. All experiments are performed on a workstation with Intel XI(R) E5–2687 w 3.0 GHz CPU and 254 GB RAM.

4.1. *Transient streamwise velocity behind a prism bluff body*

In this section, E-GP and E-SGP are applied to learn the transient velocity field behind a prism bluff body. The cross-section of the prism bluff body is an equilateral triangle with side length $a = 0.04\ m$. The prism bluffy body is in a channel with height, length, and depth of $3a$, $28a$, and $8a$, respectively. The cross-section of the flow channel and prism bluff body is depicted in Fig. 1, where the x-axis represents the flow direction, and the y-axis represents the normal direction of the wall. The flow Reynolds number (Re_a) calculated from the side length (a) is 100. The presence of the prism bluff body causes a large-scale flow separation, leading to unsteady flow and periodic asymmetrical vortex shedding.

The training data is generated using the unsteady laminar flow solver and structured mesh grid in the commercial CFD software STAR-CCM+. The time step interval in the CFD model is set as 0.01 s, while the grid resolution in the x, y, and z directions is $157 \times 60 \times 80$. The z-axis is perpendicular to the xy-plane. For simplicity, we only consider the fluctuating component of the streamwise velocity (u_x) recorded between 0 and 10a in the xy-plane, see the red rectangular in Fig. 1. Also, the grey dots ($\cdot$) in Fig. 1(b) represent the spatial locations where the training data is sampled. The temporal resolution of the training data is ten times (0.1 s) the time step interval (0.01 s) in the CFD model, whilst the time span of the recorded training data is 15 s.

Before constructing the kernel function for GP models of the transient flow domain, we assume the quantity-of-interest has the form of $f(\overrightarrow{x}) = \prod_{i=1}^{d} f_i(x_i)$, where d is the dimension of the input space or the number of independent subspace. The kernel function for the latent process $f(\overrightarrow{x})$ can then be expressed as $k(\overrightarrow{x}, \overrightarrow{x}') = \prod_{i=1}^{d} k_i(x_i, x_i')$. Here, the fluctuating component of the streamwise velocity, $u(x, y, t)$, is assumed to be the product of three independent functions $u_1(x)$, $u_2(y)$, *and* $u_3(t)$. And the kernel function $k(x, y, t, x', y', t')$

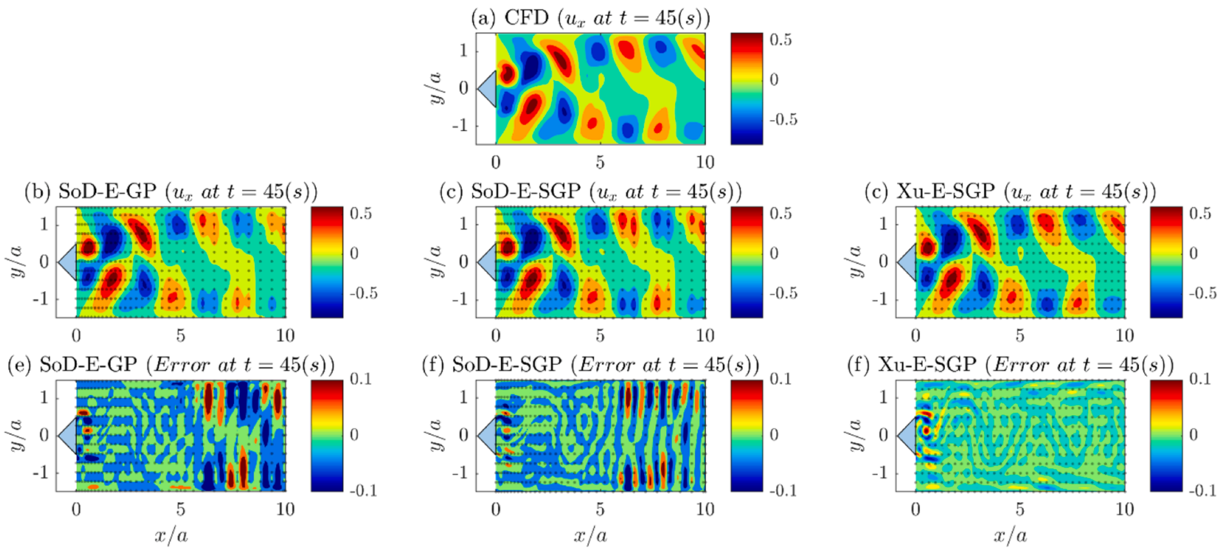

Fig. 2. A single snapshot of the streamwise fluctuating velocity component (u_x) for the incompressible flow passing a prism bluff body produced by (a) the finest CFD simulation together with three data-driven models (b) SoD-E-GP (c) Xu-E-SGP with spatial resolution as 13×33; (d) the error contour of the prediction by SoD-E-GP model and (e) the error contour of the prediction by Xu-E-SGP model.

for $u(x, y, t)$ can then be written as $k_x(x, x')k_y(y, y')k_T(t, t')$. In the example, the squared exponential (SE) kernel function is adopted for the spatial inputs and the periodic kernel is used for the temporal inputs. Hereby, the kernel function for $u(x, y, t)$ can be written as

$$k((x, y, t), (x', y', t')) = \sigma_f^2 \exp\left(-\frac{(x-x')^2}{2l_x^2}\right)\exp\left(-\frac{(y-y')^2}{2l_y^2}\right)\exp\left(-\frac{1}{2w_a^2}\sin^2\left(\pi\left|\frac{t-t'}{\tau}\right|\right)\right). \tag{24}$$

Fig. 2 shows snapshots of u_x, predicted using the physics-based and data-driven methods, as well as errors in the data-driven model predictions. It is worth mentioning that the number of data points in the E-GP model is the same as the number of inducing points in the two E-SGP models. Specifically, 13 points are assigned in the y-direction and 33 points in the x-direction. Furthermore, the temporal inputs in all three data-driven models is the same as that of the training data.

Fig. 2(a) is the CFD predictions of the u_x field downstream of the prism bluff body at $t = 45$ s, which is 15 s away from the training dataset. The predictions made by the trained E-GP model with subset of data (SoD-E-GP), E-SGP model with the previous SoD (SoD-E-SGP) as the inducing input and E-SGP with uniformed distributed inducing inputs (Xu-E-SGP) are shown in Fig. 2 (b, c, and d). Fig. 2 (e, f, and g) shows errors in the prediction of the related data-driven models. As illustrated in Fig. 2, the accuracy of the SoD-E-GP model (Fig. 2 (b & e)) and the SoD-E-SGP model (Fig. 2 (c & f)) deteriorates towards the outlet. This is because both SoD-E-GP and SoD-E-SGP inherit the non-uniform nature of the training data in the x-direction. The spatial resolution of both models becomes coarser toward the outlet. By evenly distributing the inducing points, the Xu-E-SGP model reduces the error significantly, see Fig. 2 (d and g). Finally, the accuracy of the SoD-E-SGP is slightly improved in comparison with the SoD-E-GP. Later, the statistical performance of E-GP and E-SGP due to varying input locations will be discussed.

To create a baseline, predictions of CFD models with different structured mesh resolutions are used to assess errors due to varying mesh densities. Similar resolutions are then applied to selecting subsets of data (SoD) for the E-GP model and determining the inducing points (Xu) for the E-SGP model. A total of five different spatial resolutions (in the xy-plane) are adopted, namely '8×21', '13×33', '16×41', '21×80' and '31×95'. Spatial coordinates of SoD or Xu are chosen randomly using the 'randperm' or 'lhsdesign' functions in MATLAB. This process is repeated a hundred times, giving rise to the statistics of the accuracy of the data-driven models which are shown in Fig. 3. The root mean square error (RMSE) defined in Eq. (25) and mean standardised log loss (MSLL) defined in Eq. (26) of both E-GP and E-SGP gradually reduces as more data points/inducing points (model resolutions in other words) are included in the model. For the same data input resolution, the RMSE and MSLL of the E-SGP model are likely smaller than those of the E-GP model. In particular, E-SGP evidently surpasses E-GP when the data points/inducing points are much sparser than the resolution of the training data, i.e. '8×21'.

$$RMSE = \left(\frac{1}{n_t}\sum_{i=1}^{n_t}\left(y_i^* - \bar{y}_i^*\right)^2\right)^{1/2}. \tag{25}$$

$$MSLL = \frac{1}{n_t}\left\{\sum_{i=1}^{n_t}\left[\frac{1}{2}\log(2\pi var(y_i^*)) + \frac{\left(y_i^* - \bar{y}_i^*\right)^2}{2var(y_i^*)}\right] - \sum_{i=1}^{n_t}\left[\frac{1}{2}\log\left(2\pi var\left(\vec{y}_{train}\right)\right) + \frac{\left(y_i^* - mean\left(\vec{y}_{train}\right)\right)^2}{2var\left(\vec{y}_{train}\right)}\right]\right\}, \tag{26}$$

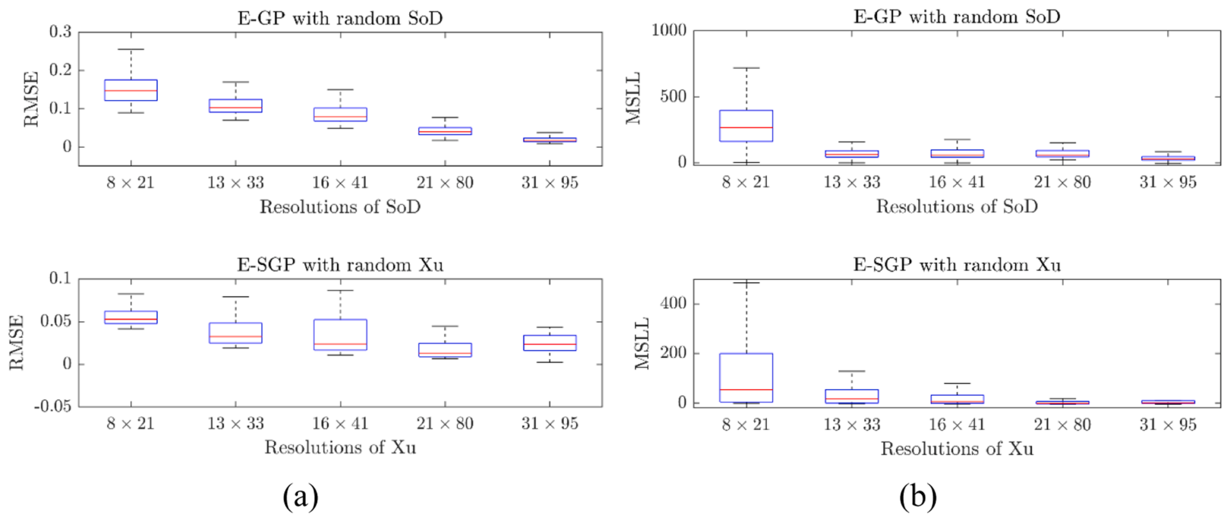

Fig. 3. (a)The box plot of the rooted mean square error (RMSE) and (b) the box plot of the mean standardised log loss (MSLL) against the configurations of randomly selected subset of data (SoD) in efficient GP (E-GP) and randomly defined inducing points (Xu) in efficient SGP (E-SGP).

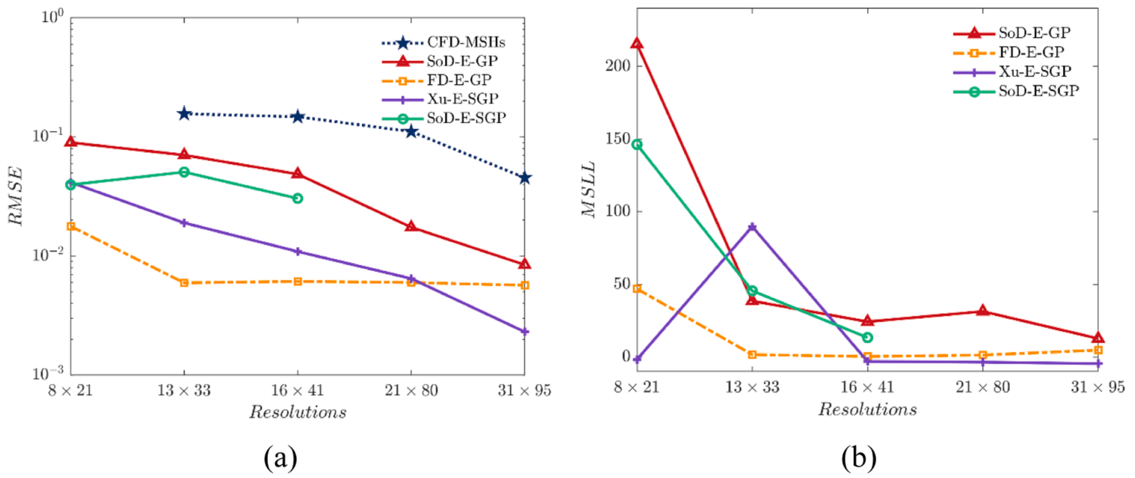

Fig. 4. (a) The minimum RMSE and (b) related MSLL value against the configuration of grids for five modelling methods, namely, 1. CFD-MSHs (CFD with different meshes, only for RMSE comparison); 2. SoD-E-GP(Subset of data with Efficient GP); 3. FD-E-GP (Full training data with efficient GP and hyperparameters from SoD-E-GP methods); 4. Xu-E-SGP (random inducing data points with efficient SGP); 5. SoD-E-SGP (Efficient SGP model with subset of data as inducing data points and hyperparameters from SoD-E-GP methods).

where $mean\left(\vec{y}_{train}\right)$ and $var\left(\vec{y}_{train}\right)$ are the mean and variance of the observations in training data.

The minimum RMSE and associated MSLL for E-GP and E-SGP models are plotted in Fig. 4. For comparison purposes, the RMSE of the coarse mesh CFD model, the metrics of the E-GP model with full dataset, as well as the E-SGP model with the same spatial inputs of the SoD in E-GP models are also presented in the figure. It is clearly shown in Fig. 4 (a & b) that the E-SGP can lead to lower RMSE and MSLL even with the same spatial input as in the E-GP model. Furthermore, the E-SGP can create a more accurate data-driven model with less spatial input even compared to the E-GP method with full dataset. Surprisingly, the errors of the data-driven model are much lower than the errors of the coarse-grid CFD models.

4.2. Lid-driven cavity flow with unstructured meshes

The performance of the E-SGP, SGPLVM and E-GP algorithms in reproducing unstructured fluid data are compared in this section. Is should be noted that the SGPLVM is also implemented in our inhouse code. Here we considered the flow in a 1 (m) × 1 (m) cavity where the flow is driven by the top moving wall, as shown in Fig. 5(a). Data-driven models using E-SGP, SGPLVM or E-GP are trained to learn the trend of the x-direction velocity component due to changes in the moving wall. The time taken to train the model (hyper-parameters optimisation) and the ability of the trained model in predicting unknowns will then be assessed.

CFD models, solved using the laminar flow solver in the commercial CFD software Star-CCM+, are used to provide the training and validation databases. In the training database, u_{wall} varies as 0.02 m s^{-1}, 0.04 m s^{-1}, 0.08 m s^{-1}, 0.2 m s^{-1}, 0.64 m s^{-1}, 1.0 m s^{-1} and 1.5 m s^{-1}, whilst cases with $u_{wall} = 0.7$ m s^{-1} and 0.9 m s^{-1} are treated as the validation database. The training database here is produced using the CFD model with unstructured triangular mesh containing 6437 mesh elements, as illustrated in Fig. 5(b), thus the full training dataset contains 6437 × 7 points. The validation data is generated using different meshes, a quadrilateral mesh for $u_{wall} = 0.7m\ /s$ (Fig. 5(c)) and a polygonal mesh for $u = 0.9m/s$ (Fig. 5(d)). It is worth noting here that the spatial input space (i.e., locations of the data points in the flow domain) is orthogonal to the operating input space (i.e., moving wall velocity).

The performance of the E-GP and E-SGP algorithms on unstructured fluid datasets will be evaluated by comparing the quality of the E-GP and E-SGP models trained on 10%, 20%, 40%, 80% and 100% of the full training dataset. Furthermore, the scalability of both approaches will be examined by comparing the execution time of 1000 Metropolis-Hastings samplings (MH samplings) versus the number of training data. It is known that spatial inputs and moving wall velocities belong to two orthogonal input subspaces and the number of spatial inputs (6437) is much larger than the number of considered moving wall velocities (7). Therefore, in this example, we scale down the training data by reducing the number of spatial inputs and their related observation (u_x).

In the E-GP model, the full covariance matrix is the Kronecker product of two sub-covariance matrices, one for the spatial inputs and the other for the moving wall velocities. In terms of the data points in the subset of training data (SoDs), we try to distribute the data points as evenly as possible in the flow domain. An example of the SoD is shown in Fig. 6(a). For the SGPLVM model, the input spaces of the inducing points share the same structure as the observations; therefore, they are treated as the Cartesian product of the spatial coordinates and the moving wall velocity. To facilitate comparison between SGPLVM and E-SGP models, the inducing points in both models are identical and located on a grid, specifically with 20 points on the x-axis, 30 points on the y-axis, and 7 points (same as in the full training dataset) on the moving wall velocity axis. The inducing points on the xy-plane is presented in Fig. 6(b). It should be noted that, in SGPLVM model, the spatial points are grouped into a single subspace containing 600 pairs of coordinates. The data-driven models are trained using the Metropolis-Hastings (MH) algorithm. One hundred epochs, with 1000 MH-Samplings in each

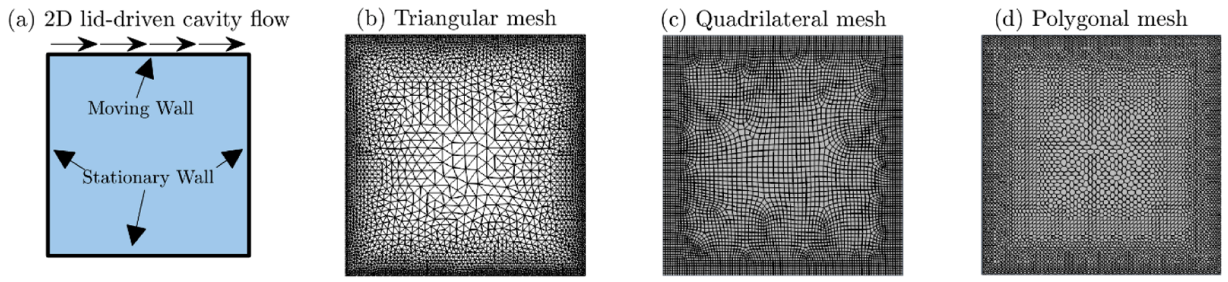

Fig. 5. (a) Geometry of the lid-driven cavity flow; (b) a view of the triangular mesh of the CFD model for generating the training dataset; (c) the quadrilateral mesh for the validation case of $u_{wall} = 0.7\ m/s$; (d) the polygonal mesh for the validation case of $u_{wall} = 0.9\ m/s$.

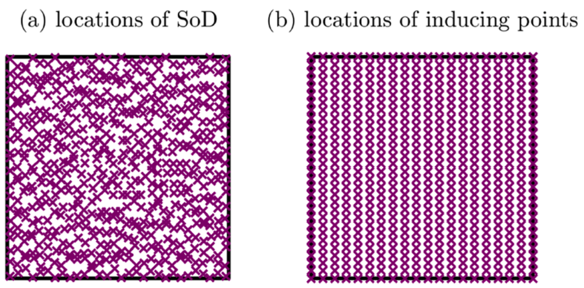

Fig. 6. (a) Spatial locations randomly selected subset of data for the E-GP-SoD (597 spatial points), (b) the uniformly distributed inducing points in the flow domain with resolution as 20 × 30.

epoch, are used in the training, except for the E-GP models trained using 80% and 100% of the full training dataset. E-GP models with 80% and 100% of the full dataset scale very poorly, the training epochs for both models are set to ten.

Fig. 7 illustrates the execution time (in seconds) of three algorithms (E-GP, SGPLVM and E-SGP) for 1000 MH sampling steps as a function of the percentage of training data used. Although the E-GP model benefits from the orthogonality of the input subspaces, it shows the highest execution time, increasing steeply from around 40 s at 10% training data to nearly 15,000 s at 100%, indicating poor scalability for the unstructured dataset. This is because the sub-covariance matrix for unstructured spatial input grows cubically with the increment of the training data. By leveraging the advantages of the Kronecker product and the small number of inducing points, the scalability of the SGPLVM model exhibits significant improvement, with execution times ranging from about 60 s to just over 240 s. The E-SGP model demonstrates the best performance, with execution times remaining relatively constant and the lowest among the three models, increasing slightly from around 40 s to just over 200 s, highlighting its efficiency and scalability with increasing data sizes. Notably, the execution times of E-SGP and SGPLVM gradually converge as the amount of training data increases, as the benefit from imposing orthogonality on the spatial inducing points is overshadowed by the larger matrix multiplication.

The statistics of the rooted-mean-square-error (RMSE) and the mean-standardised log loss (MSLL) of E-GP and E-SGP models are plotted in Fig. 8 and Fig. 9. Fig. 8 shows box plots of the RMSE for three models—E-GP, SGPLVM, and E-SGP—across five different percentages of training data: 10%, 20%, 40%, 80%, and 100%. SGPLVM and E-SGP show comparable RMSE values, with slight variations across training data percentages. The E-GP model shows lower RMSE values compared to SGPLVM and E-SGP, with a decreasing trend as training data increases. In the meantime, the variances of the RMSE of E-SGP and SGPLVM are generally larger than those of E-GP. This is because the resolution of the E-GP model increases with the number of training data, while the resolutions (configurations of the inducing point) in the E-SGP and SGPLVM remains the same (30 × 20 × 7).

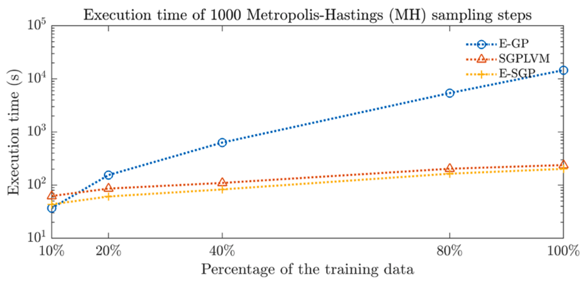

Fig. 7. Execution time (s) of the 1000 MH sampling of E-GP, SGPLVM, and E-SGP with 10%, 20%, 40%, 80% and 100% of full training dataset.

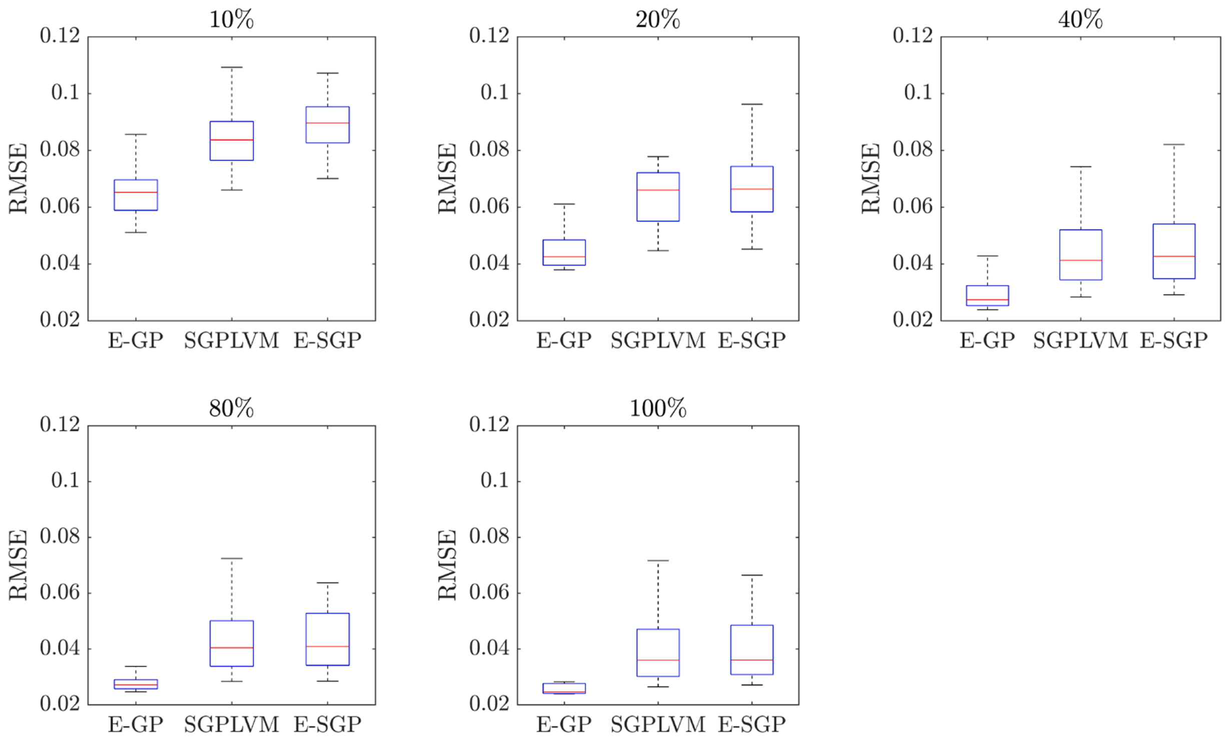

Fig. 8. The box plot of the rooted mean squared error (RMSE) of E-GP, SGPLVM, and E-SGP model trained using 10%, 20%, 40%, 80% and 100% of full training dataset.

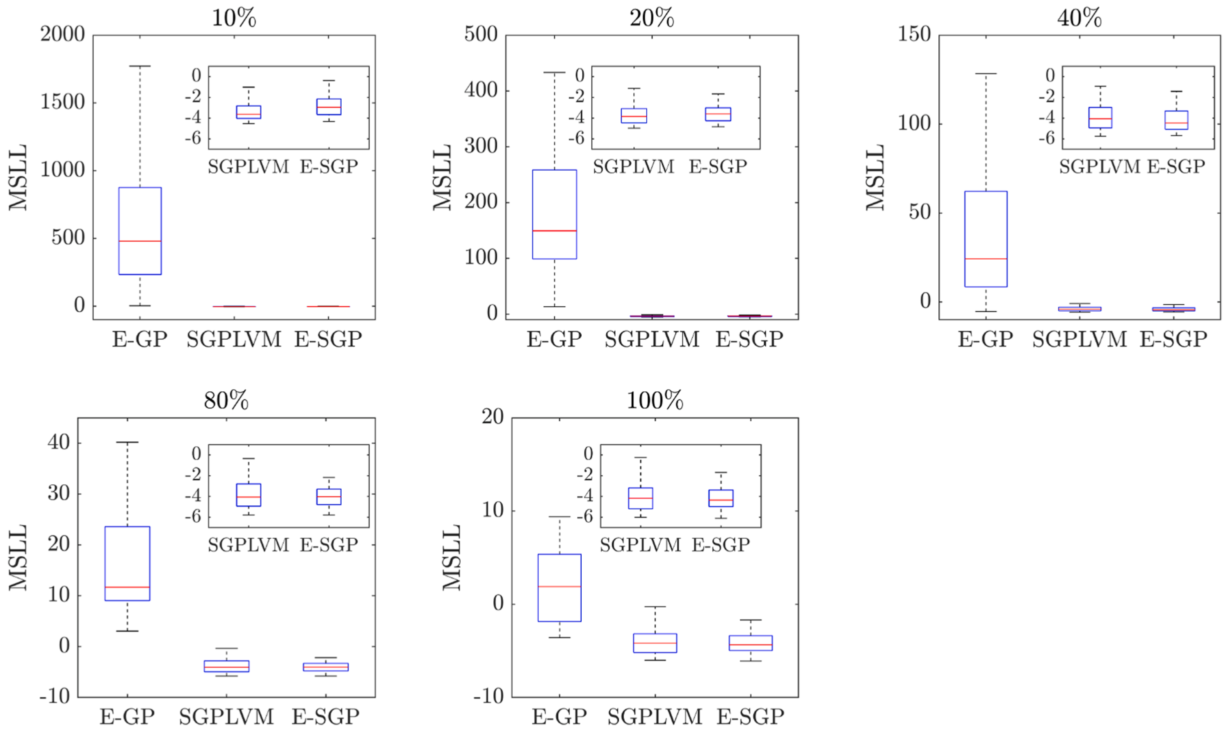

Fig. 9. The box plot of the mean standardised log loss (MSLL) of E-GP, SGPLVM and E-SGP model trained using 10%, 20%, 40%, 80% and 100% of full training dataset.

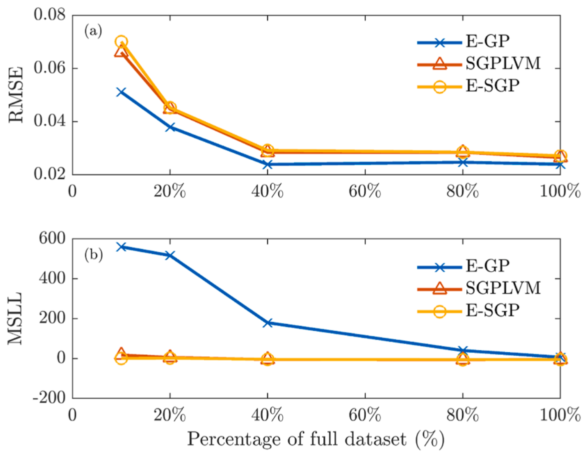

Fig. 10. (a)The relationship between percentage of full training dataset with the minimum RMSE and (b) related MSLL of the model trained using E-GP, SGPLVM, and E-SGP model respectively.

Fig. 9 presents box plots of the MSLL for the three models across the same five percentages of training data. MSLL consists of two parts, the one is the level of model variance, the other is the ratio of the squared error and the model variance. For a high-quality Bayesian machine learning model, the MSLL should be smaller than zero and the smaller the better. However, MSLLs of all E-GP models are generally above zero, implying the model variance is either too large or too small. Considering the RMSEs of E-GP models are lower than 0.1, the high MSLL of E-GP models means that the trained E-GP models are over-confident. In other words, the model variances of E-GP models are unrealistically low. Both SGPLVM and E-SGP models show consistently low MSLL values with minimal variability, while the inset plots provide a closer view, indicating that E-SGP performs slightly better than SGPLVM, with marginally lower MSLL values.

Similar observations can be found in Fig. 10, which depicts the relationship between the percentage of the training data and the minimum RMSE and corresponding MSLL of the E-GP and E-SGP models. Regarding the minimum RMSE, E-GP shows a consistent downward trend in RMSE as the dataset size increases, outperforming the other two models, whilst SGPLVM and E-SGP follow a similar trend, converging at higher data percentages but with slightly higher values than E-GP. In terms of MSLL, E-GP starts with a high MSLL ($\sim$580) that decreases sharply with increasing dataset size. SGPLVM and E-SGP maintain low and stable MSLL values across all dataset sizes, with E-SGP slightly outperforming SGPLVM. In comparison with the significant improvement in the MSLL, the RMSEs of E-SGP and SGPLVM are only marginally smaller than their counterparts, especially when the percentage of the full training data is larger than 40%. Overall, E-GP outperforms SGPLVM and E-SGP in terms of RMSE, indicating better predictive accuracy, but at the cost of scalability. However, E-GP's high and variable MSLL indicates poor reliability and performance in uncertainty quantification. In contrast, SGPLVM and E-SGP demonstrate significantly better performance and robustness in terms of MSLL, with E-SGP having a slight edge over SGPLVM.

5. Summary and discussion

In this work, we derive the Kronecker product accelerated efficient sparse Gaussian process (E-SGP) for both structured and unstructured mesh fluid datasets. This novel algorithm significantly enhances the computational tractability and efficiency of the approximated sparse GP algorithm VEF-GP without introducing additional parameters or imposing stringent restrictions on the excessively large covariance matrix. Moreover, the separation between the input of training data and the structured inducing input allows for the use of databases with varying resolutions, such as CFD data generated using different meshes. We empirically examine the performance of the E-GP and E-SGP using fully structured and partially structured databases. For the latter, we also compare the performance of the SGPLVM and our E-SGP algorithm.

In the case of a fully structured database, with similar model resolutions (i.e., the number of training data points in E-GP being the same as the number of inducing points in E-SGP), the E-SGP produces lower RMSE values. This is because E-SGP can utilise the full training dataset without significantly increasing the computational burden of solving the inverse of the covariance matrix, whereas E-GP only considers a subset of the full training data.

When the dataset becomes partially structured and the amount of training data in the E-GP, SGPLVM, and E-SGP models is the same, the scalability of the E-SGP outperforms the other two approaches. Although E-GP marginally outperforms E-SGP and SGPLVM in terms of RMSE, it does so at the cost of extremely increased training time. Notably, the resolution of the inducing data points in all E-SGP and SGPLVM models remains the same as that of E-GP with the smallest subset of data (SoD). The RMSE differences between the E-GP, SGPLVM, and E-SGP models are minimal and decrease as more training data is fed to the SGPLVM and E-SGP models. Additionally, in our experiments, E-GP is more likely to lead to unreasonably large MSLLs compared to E-SGP and SGPLVM. This suggests that the uncertainty estimates for E-GP model predictions may be unrealistic, and such overconfidence could result in biased conclusions in the reliability analysis of safety-critical engineering problems. Moreover, E-SGP generally outperforms SGPLVM in both MSLL and RMSE, making it the most balanced model in terms of predictive accuracy and uncertainty quantification.

Although the primary goal of this work is to extend the application of GP to different types of datasets in fluid dynamics problems, the method itself can be applied more broadly to other applications. For instance, it would be interesting to apply this method to generate Bayesian ML models for interdisciplinary problems such as parametric analysis of thermal fatigue phenomena caused by thermal stripping or thermal stratification. Another important area for future research is the integration of this algorithm with other physics-informed, physics-constrained, or physics-guided ML concepts to develop an efficient Bayesian ML solver for fluid dynamics problems.

CRediT authorship contribution statement

Yu Duan: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Matthew Eaton:** Writing – review & editing, Supervision. **Michael Bluck:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

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

Data availability

In accordance with EPSRC funding requirements, all supporting data used to create the plots in this paper may be accessed via https://doi.org/10.5281/zenodo.13885956.

Acknowledgements

Dr. Michael Bluck and Dr. Yu Duan would like to thank the financial support from the UK Engineering and Physical Sciences Research Council (EPSRC) via the research grant entitled “ATF Solutions to Light Water-Cooled SMRs” (EPSRC grant number: EP/X011313/1).

References

[1] S.L. Brunton, Applying machine learning to study fluid mechanics, Acta Mech. Sin. Xuebao. 37 (2021) 1718–1726, https://doi.org/10.1007/s10409-021-01143-6.
[2] G. Qi, Z. Zhu, K. Erqinhu, Y. Chen, Y. Chai, J. Sun, Fault-diagnosis for reciprocating compressors using big data and machine learning, Simul. Model. Pract. Theory. 80 (2018) 104–127, https://doi.org/10.1016/j.simpat.2017.10.005.
[3] OpenAI, GPT-4 Technical Report, in: GPT-4 Technical Report, 4, 2023, pp. 1–100.
[4] N. Shah, S. Engineer, N. Bhagat, H. Chauhan, M. Shah, Research trends on the usage of machine learning and artificial intelligence in advertising, Augment. Hum. Res. 5 (2020) 1–15, https://doi.org/10.1007/s41133-020-00038-8.
[5] S. Emerson, R. Kennedy, L. O'Shea, J. O'Brien, Trends and applications of machine learning in quantitative finance, in: 8th Int. Conf. Econ. Financ. Res., 2019.
[6] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech 52 (2020) 477–508, https://doi.org/10.1146/annurev-fluid-010719-060214.
[7] F. Sofos, C. Stavrogiannis, K.K. Exarchou-kouveli, D. Akabua, G. Charilas, T.E. Karakasidis, Current trends in fluid research in the era of artificial intelligence: a review, Fluids 7 (2022) 1–25, https://doi.org/10.3390/fluids7030116.
[8] M. Lino, S. Fotiadis, A.A. Bharath, C.D. Cantwell, Current and emerging deep-learning methods for the simulation of fluid dynamics, Proc. R. Soc. A Math. Phys. Eng. Sci. 479 (2023), https://doi.org/10.1098/rspa.2023.0058.
[9] K. Osawa, S. Swaroop, A. Jain, R. Eschenhagen, R.E. Turner, R. Yokota, M.E. Khan, Practical deep learning with Bayesian principles, Adv. Neural Inf. Process. Syst. 32 (2019) 1–13.
[10] J. Bradshaw, A.G. de G. Matthews, Z. Ghahramani, Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks, (2017) 1–33. http://arxiv.org/abs/1707.02476.
[11] V. Volodina, P. Challenor, The importance of uncertainty quantification in model reproducibility, Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. 379 (2021), https://doi.org/10.1098/rsta.2020.0071.
[12] I. Pan, L.R. Mason, O.K. Matar, Data-centric Engineering: integrating simulation, machine learning and statistics. Challenges and opportunities, Chem. Eng. Sci. 249 (2022) 117271, https://doi.org/10.1016/j.ces.2021.117271.
[13] M. Alvarez, D. Luengo, N.D. Lawrence, Latent force models, in: 12th Int. Conf. Artif. Intell. Stat. 2009, Florida, USA, PMLR, Clearwater Beach, 2009, pp. 9–16.
[14] M.A. Álvarez, J. Peters, B. Schölkopf, N.D. Lawrence, Switched latent force models for movement segmentation, in: Adv. Neural Inf. Process. Syst. 23 24th Annu. Conf. Neural Inf. Process. Syst. 2010 2010, NIPS, 2010.
[15] D. Luengo, M. Campos-taberner, G. Camps-valls, Latent force models for earth observation time series prediction, in: 2016 IEEE Int. Work. Mach. Learn. SIGNAL Process., IEEE, 2016, pp. 0–5.
[16] G. Camps-Valls, L. Martino, D.H. Svendsen, M. Campos-Taberner, J. Muñoz-Marí, V. Laparra, D. Luengo, F.J. García-Haro, Physics-aware Gaussian processes in remote sensing, Appl. Soft Comput. J. 68 (2018) 69–82, https://doi.org/10.1016/j.asoc.2018.03.021.
[17] S. Sarkka, M.A. Alvarez, N.D. Lawrence, Gaussian process latent force models for learning and stochastic control of physical systems, IEEE Trans. Automat. Contr. 64 (2019) 2953–2960, https://doi.org/10.1109/TAC.2018.2874749.
[18] M. Raissi, P. Perdikaris, G.E. Karniadakis, Numerical gaussian processes for time-dependent and nonlinear partial differential equations, SIAM J. Sci. Comput 40 (2018) A172–A198.
[19] M. Raissi, P. Perdikaris, G.E. Karniadakis, Machine learning of linear differential equations using Gaussian processes, J. Comput. Phys. 348 (2017) 683–693, https://doi.org/10.1016/j.jcp.2017.07.050.

[20] Y. Chen, B. Hosseini, H. Owhadi, A.M. Stuart, Solving and learning nonlinear PDEs with Gaussian processes, J. Comput. Phys. 447 (2021) 110668, https://doi.org/10.1016/j.jcp.2021.110668.

[21] H. Liu, Y. Ong, X. Shen, J. Cai, S. Member, When Gaussian process meets big data: a review of scalable GPs, IEEE Trans. Neural Netw. Learn. Syst. 30 (2020) 4405–4423, https://doi.org/10.1109/TNNLS.2019.2957109.

[22] C.K.I. Williams, M. Seeger, The using nystrom method to speed up kernel machines, in: Adv. Neural Inf. Process. Syst. 13 (NIPS 2000), 2000, pp. 1–7.

[23] L. Csató, M. Opper, Sparse on-line Gaussian processes, Neural Comput. 14 (2002) 641–668, https://doi.org/10.1162/089976602317250933.

[24] J. Quinonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate gaussian process regression, J. OfMachine Learn. Res. 6 (2005) 1939–1959.

[25] M. Seeger, C. Williams, N. Lawrence, Fast forward selection to speed up sparse Gaussian process regression, Artif. Intell. Stat. 9 (2003) 1–8.

[26] M.K. Titsias, Variational learning of inducing variables in sparse gaussian pocesses, in: 12th Int. Conf. Artif. Intell. Stat, Clearwater Beach, Florida, USA, 2009, pp. 567–574.

[27] J. Hensman, N. Fusi, N.D. Lawrance, Gaussian processes for big data, Conf. Uncertain. Artif. Intell. (2013) 282–290, https://doi.org/10.1016/S0074-7696(01)08005-6.

[28] J. Hensman, A.G. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, J. Mach. Learn. Res. 38 (2015) 351–360.

[29] Y. Saatçi, Scalable Inference for Structured Gaussian Process Models, University of Cambridge, 2011.

[30] A.G. Wilson, Hannes Nickisch, Kernel interpolation for scalable structured Gaussian Processes (KISS-GP), in: 32nd Int. Conf. Mach. Learn, Lille, France, 2015, pp. 1–10.

[31] J.R. Gardner, G. Pleiss, R. Wu, K.Q. Weinberger, A.G. Wilson, Product kernel interpolation for scalable gaussian processes, in: Int. Conf. Artif. Intell. Stat. AISTATS 2018 84, 2018, pp. 1407–1416.

[32] S. Stanton, W.J. Maddox, I. Delbridge, A.G. Wilson, Kernel interpolation for scalable online Gaussian processes, in: 24th Int. Conf. Artifi- Cial Intell. Stat, 2021. http://arxiv.org/abs/2103.01454.

[33] F. Wesel, K. Batselier, Tensor-based kernel machines with structured inducing points for large and high-dimensional data, Proc. Mach. Learn. Res. 206 (2023) 8308–8320.

[34] P.A. Izmailov, A.V. Novikov, D.A. Kropotov, Scalable Gaussian process with billions of inducing input via tenser train decomposition, in: Int. Conf. Artif. Intell. Stat., Playa Blanca, Lanzarote, Canary Islands, Spain, 2018, pp. 726–735.

[35] Z. Dai, M.A. Álvarez, N.D. Lawrence, Efficient modeling of latent information in supervised learning using Gaussian processes, in: Adv. Neural Inf. Process. Syst. 2017-Decem, 2017, pp. 5132–5140.

[36] S. Atkinson, N. Zabaras, Structured Bayesian Gaussian process latent variable model: applications to data-driven dimensionality reduction and high-dimensional inversion, J. Comput. Phys. 383 (2019) 166–195, https://doi.org/10.1016/j.jcp.2018.12.037.

[37] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes For Machine Learning, the MIT Press, 2006. ISBN 026218253X. c 2006 Massachusetts Institute of Technology, www.GaussianProcess.org/gpml.

[38] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, S. Aigrain, Gaussian processes for time-series modelling, Philos. Trans. R. Soc. A. 371 (2013).

[39] A. Damianou, Deep Gaussian processes and variational propagation of uncertainty, 2015.

[40] S. Reece, S. Ghosh, A. Rogers, S. Roberts, N.R. Jennings, Efficient state-space inference of periodic latent force models, J. Mach. Learn. Res. 15 (2014) 2337–2397.

[41] E. Snelson, Z. Ghahramani, Sparse Gaussian Processes using Pseudo-inputs Edward, Adv. Neural Inf. Process. Syst 18 (2005) 1–8. https://papers.nips.cc/paper_files/paper/2005/file/4491777b1aa8b5b32c2e8666dbe1a495-Paper.pdf.

[42] J. Schreiter, D. Nguyen-Tuong, M. Toussaint, Efficient sparsification for Gaussian process regression, Neurocomputing. 192 (2016) 29–37, https://doi.org/10.1016/j.neucom.2016.02.032.