International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 2
ISSN: 2088-5334

Development of an Emulator for Radiation Physics Simulator Using the Stochastic Variational Gaussian Process Model

Yonggwan Shin [a], Yun Am Seo [b,*]

[a] *R&D Center, XRAI Inc., Gwangju, 61186, Republic of Korea*
[b] *Department of Data Science, Jeju National University, Jeju-si, 63243, Republic of Korea*
Corresponding author: [*]seoya@jejunu.ac.kr

***Abstract*—This paper describes an emulator that uses a Stochastic Variational Gaussian process (SVGP) regression model to parameterize radiation in a numerical weather prediction (NWP) model that meteorologically models the Earth's weather system. The computation of radiative processes is considerable, accounting for most of the total NWP model computation. Statistical emulators are surrogate models representing simulators and can overcome the computational limitations of complex simulators such as radiative processes. Recently, artificial neural network-based radiative transfer emulators have been developed, and in this study, a statistical model, GP, is used to create a radiative transfer emulator. The GP model has the advantage of calculating the uncertainty of the prediction along with the prediction, so the uncertainty of the prediction can be utilized appropriately. However, the computational complexity of the conventional GP model is very high, making it challenging to apply to extensive data. An approximate approach, the SVGP model, was utilized to solve this problem. To further reduce the dimensionality of the input variables, we used a combined neural network and SVGP model. As a result, the SVGP-based radiative physics emulator improved its accuracy by about 20% compared to the artificial neural network emulator. However, the computation speed was about 3 to 9 times slower than the neural network emulator but faster than the NWP model's computation speed. This suggests that statistical emulators can be used to replace NWP model simulators.**

***Keywords*—Surrogate model; emulator; gaussian process.**

Manuscript received 20 Oct. 2023; revised 19 Dec. 2023; accepted 22 Jan. 2024. Date of publication 30 Apr. 2024.

I. INTRODUCTION

Nowadays, computational experiments utilizing computers are frequently executed across diverse domains. As computers have advanced significantly and sophisticated algorithms have been developed, numerous physical phenomena can now be elucidated and forecasted through computational experiments [1]-[4]. Recently, computational experiments aimed at predicting climate and weather have also gained substantial momentum [5], [6]. The integration of machine learning into weather prediction has been highlighted by a benchmark dataset aimed at evaluating AI-driven weather forecasting models to improve predictions and reduce computational costs [7]. A simulator constitutes a mathematical depiction of a physical system realized through computer implementation [8]-[10]. Simulators have been utilized in many scientific and technological fields to analyze actual systems. Generally, simulators produce consistent output when subjected to identical initial conditions and input values. It is common for simulators to demand significant computational time, mainly when dealing with complex physical systems. A potential solution to address this limitation is the use of statistical emulators. A statistical emulator serves as a statistical model that effectively captures the behavior of such simulators [11]-[15].

Currin et al. [1] and Santner et al. [2] pioneered research on statistical emulators. Notably, O'Hagan [3] explored the development of Gaussian process (GP) models as emulators for unknown functions, effectively serving as meta-models for simulators. Implementing a well-mimicking emulator that closely replicates the simulator's behavior offers the possibility of alleviating the computational time challenges encountered during simulator execution.

Emulators can also find applications in the calibration of simulators. Park [16] introduced an approximate nonlinear least squares approach for calibration, employing a Gaussian process regression model. Cox et al. [17] presented a similar approximate nonlinear least squares estimator for calibration,

utilizing a GP model as a surrogate for intricate simulators. In a more recent development, Lee and Park [8] introduced generalized nonlinear least squares as a calibration technique for GP models, and Seo and Park [9] proposed a method employing the EM algorithm.

In the field of weather and climate, there is a lot of research on developing emulators to replace existing numerical models. Recently, there has been an increasing number of studies on developing emulators to replace existing numerical models using modern machine learning methods such as artificial neural networks. Research on model improvement based on theory is rapidly changing to research on improvement based on data and machine learning techniques [10]. Roh and Song [10] devised an artificial neural network-based simulator tailored to the radiative transfer process within the Korea Meteorological Administration (KMA) Rapid Radiative Transfer Model for General Circulation Models-Korea (RRTMG-K). Radiative transfer processes contribute over 80% of the computational load in numerical weather prediction (NWP) models. Findings from the study demonstrated that a factor of at least twenty could improve computational efficiency. Nonetheless, during extended emulator operation, the cumulative errors in both the simulator and emulator may lead to deviations, particularly towards the end of the forecast period. Such deviations can compromise the accuracy of specific predictors within the NWP model and even induce model termination. Thus, ensuring the numerical stability of the emulator becomes pivotal for sustaining the reliable functioning of the NWP model reliant on emulator-driven operations.

This study aims to formulate a statistical emulator based on the GP to advance prediction accuracy and incorporate uncertainty considerations [18], [19]. Traditional GP models often demand a substantial temporal investment for training or may be challenged by extensive datasets [20], [21]. In response, researchers have introduced alternative models, such as the Local Approximate Gaussian Process (LAGP) and the SVGP, engineered to curtail training and inference durations, even when confronted with voluminous datasets [22]. The emulator advocated in this paper adopts a neural network architecture proficient in reducing the dimensionality of input variables and subsequently making forecasts using SVGP as input. This approach mitigates computational expenditures and facilitates the quantification of output value uncertainty. This reduces the computational cost and allows the uncertainty of the output to be expressed quantitatively so that the numerical model can be operated reliably even if an emulator replaces the simulator.

II. Materials and Method

A. Data

We utilized 196 input variables and eighty-six output variables to develop the radiative physics emulator, detailed in Table 1. The variables include pressure, air temperature, water vapor, and ozone across thirty-nine vertical layers, alongside single-level variables such as surface temperature, solar constant, and the solar zenith angle cosine. The vertical distribution of cloud cover is crucial in our study as an input variable. For modeling shortwave radiation, both the solar constant and the cosine solar zenith angle are vital, dictating the solar radiation received by Earth [10]. Our dataset comprises approximately 270,000 instances for training and 130,000 for accuracy evaluation, consistent with the dataset used by Roh and Song [10] to develop their emulator.

TABLE I
INPUT/OUTPUT VARIABLES FOR LEARNING RADIATIVE PHYSICS EMULATORS

Input Variables	
Vertical Pressure	#1-39
Vertical Temperature	#40-78
Vertical Water Vapor Mixing Ratio	#79-117
Vertical Ozone Mixing Ratio	#118-156
Vertical Cloud Fraction	#157-192
Forecast Time	#193
Surface Temperature	#194
Solar Constant	#195
Cosine Solar Zenith Angle	#196
Output Variables	
Vertical Total Sky Longwave Radiative Heating Rate	#1-39
Vertical Total Shortwave Radiative Heating Rate	#40-78
Total Sky Longwave Upward Flux at Top	#79
Clear Sky Longwave Upward Flux at Top	#80
Total Sky Longwave Upward Flux at Bottom	#81
Clear Sky Longwave Upward Flux at Bottom	#82
Total Sky Longwave Downward Flux at Top	#83
Clear Sky Longwave Downward Flux at Bottom	#84
Total Sky Shortwave Upward Flux at Top	#85
Total Sky Shortwave Upward Flux at Bottom	#86

B. Approximate Gaussian Process

The GP is a robust probabilistic method for modeling nonlinear relationships. Represented as an infinite-dimensional normal distribution, a GP offers a distribution over function spaces, incorporating correlations among data points to predict outcomes for new instances. Nonetheless, traditional GPs involve all training data in predictions, leading to a computational complexity of $O\ (N^3)$, which hampers their application to a large dataset [19], [23].

The approximate Gaussian process enables efficient inference on large datasets. Quinonero-Candela and Rasmussen [19] proposed a method to significantly reduce the computational complexity of GP by using inducing points instead of the entire dataset.

The idea behind the approximate Gaussian process model is to compute the covariance matrix using derived points instead of the entire dataset. The inducing points are a small subset representative of the entire dataset and are used to summarize the characteristics of the whole dataset effectively. This approach is computationally efficient and makes GP models practical for large datasets. An approximate Gaussian process of using induction points is defined as follows.

$$f \mid f_u \sim N(K_{uf}\, K_{uu}^{-1}\, f_u\ , K - K_{uf}\, K_{uu}^{-1}\, K_{uf}^{T}) \qquad (1)$$

where, K_{uu}, and K_{uf}^{T} are the covariance matrices between the induction points and the training data, between the inducing points, and between the inducing points and the training data, respectively. To compute the covariance matrix K, various functions can be used.

This study used the Matérn kernel function, which is defined as follows:

$$K_{ij} = k_{Matérn}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x-x'\|}{\rho}\right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}\|x-x'\|}{\rho}\right) \qquad (2)$$

The inducing point X_u is a set of data points used to summarize the characteristics of the entire dataset X. These points play an important role in effectively capturing the covariance structure of the entire dataset and reducing the computational complexity of GP.

$$X_u = \{X_{u1}, X_{u2}, \dots, X_{uM}\} \quad (3)$$

where M is the number of inducing points and x_{ui} denotes the inducing point. The induction points X_u and other hyperparameters are optimized to maximize the performance of the GP model. This optimization is typically done by maximizing the log-likelihood or variational lower bound.

By using inducing points, the computational complexity is reduced from $O\ (N^3)$to $O\ (M^2N)$This approach significantly reduces computational complexity while maintaining GP's flexibility, and it plays an important role in finding a balance between prediction accuracy and computational efficiency on large datasets.

C. Variational ELBO

The variational evidence lower bound (ELBO) is a method for fitting GP models more efficiently when dealing with complex datasets. This method is beneficial when the data does not follow a normal distribution [6]. Recent studies have shown that the ELBO can be expressed as a sum of three entropies at convergence, highlighting its efficiency and theoretical underpinnings in variational inference [24]. The variational ELBO is used to approximate the posterior distribution of the model and is defined as follows.

$$L = \sum_{i=1}^{N} E_{q(f_i)} [log\ p(y_i \mid f_i)] - KL[q(f)||p(f)] \quad (4)$$

where L represents the variational lower bound and $q(f)$ is the variational approximation to the posterior distribution of the GP. The first term represents the expected value of the likelihood on the data, and the second term represents the Kullback-Leibler divergence between $q(f)$ and the prior distribution $p(f)$.

D. Stochastic Variational Gaussian Process with Neural Network Feature Extractor

The Stochastic Variational Gaussian Process (SVGP) is a model designed to overcome computational limitations on large datasets by combining the approximate Gaussian process in Section II-B and the variational ELBO method in Section II-C [25]-[29]. SVGP uses inducing points to approximate the posterior distribution of a Gaussian process. To do this, the variance distribution $q(f)$ is optimized as follows.

$$q(f) = \int q(f|u)\, q(u)\, du \quad (5)$$

where $q(u)$ is the variance distribution over the inducing points and $q(f|u)$ is the conditional distribution over the given inducing points. The prediction for the new data point x_* is computed as follows:

$$q(f_*) = \int p(f_*|\, u) q(u)\ du \quad (6)$$

The conditional distribution $p(f_*|\, u)$ of the inferred point u over the new data points x_* can be used to compute the predicted mean and variance.

$$\mu_* = E[f_*] = K_{*u} K_{uu}^{-1} \mu_u \quad (7)$$

$$\Sigma_* = Var[f_*] = K_{**} - K_{*u} K_{uu}^{-1} (K_{uu} - \Sigma_u) K_{uu}^{-1} K_{u*} \quad (8)$$

In equations (7), (8), K_{*u} is the covariance between the new data point and the inferred point, K_{uu} is the covariance between the inferred points, and μ_u and Σ_u are the mean and covariance of $q(u)$, respectively.

If we directly apply the training data described in Section II-A to the SVGP model, we may encounter convergence problems because the input dimensionality is huge at 196. To mitigate this dimensionality curse, the number of inducing points, M, should be set large enough, but this increases the computational complexity of the SVGP model to $O(M^2N)$. However, as the number of inducing points increases, the size of the covariance matrix increases and the computational cost increases exponentially.

Fig. 1 shows the overall architecture of NN-SVGP, which combines neural networks and SVGP models. In the figure, Y_d represents the output values from each independent SVGP model, and d is the number of output variables in the data. The feature extractor exists as a single neural network, and there are d SVGP models. The feature extractor and SVGP models are not trained separately but are combined in an end-to-end learning method using gradient descent.

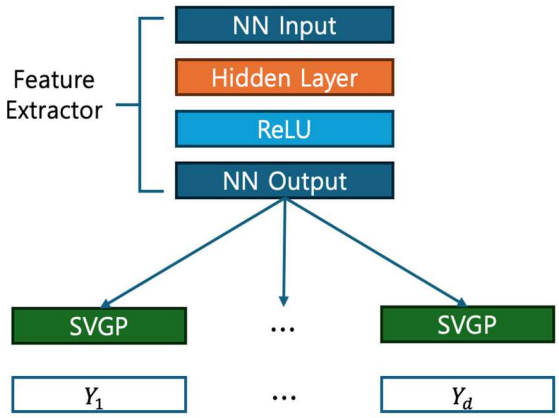

Fig. 1 Neural Network with Stochastic Variational Gaussian Process (NN-SVGP) Model Architecture

Moving beyond traditional computational methods, the field has seen notable enhancements in scientific computing capabilities through intelligent algorithms. Neural network models, with their ability to learn and adapt, have demonstrated significant potential across various scientific domains. These models, characterized by their fault tolerance, robustness, and ability to perform parallel computations, have become invaluable tools for tackling complex problems in scientific computing. Additionally, integrating machine learning with applied mathematics has led to significant innovations. For instance, the concept of 'graph coarsening' draws parallels between mesh simplification techniques in scientific computing and data reduction strategies in machine learning, underscoring the versatile applications of these methods in both fields. Such techniques, including the use of generative models and the application of machine learning to solve differential equations, are now prominent in computational science, providing fresh solutions to long-standing issues. The merging of machine learning with scientific computing is gradually enhancing the

methodologies employed in computational experiments, notably in climate and weather prediction [30], [31].

III. Results and Discussion

A. Selection of hyperparameters

The optimization of the NN-SVGP-based radiative physics emulator required meticulous analysis of error outcomes across various hyperparameter settings, a critical process for enhancing prediction accuracy while maintaining computational efficiency. The primary hyperparameters under scrutiny were the size of the induction points (Ips), the number of neurons in the hidden layer (Hidn), and the number of neurons in the output layer (Fout) of the neural network.

We computed the Root Mean Squared Error (RMSE) for an array of hyperparameter configurations to assess the model's performance. RMSE, the square root of the average squared differences between predicted and actual values, is a widely recognized measure for evaluating predictive accuracy. Beyond assessing the overall RMSE, this study delved into the errors associated with Long Wave (LW) Heating Rate, Short Wave (SW) Heating Rate, LW Flux, and SW Flux.

Table 2 presents the RMSE values across various hyperparameter settings, facilitating an in-depth comparison of their effects on model accuracy. Notably, the configuration with Ips=128, Hidn=256, and Fout=128 emerged as the most effective, demonstrating the lowest overall error. Such insights were instrumental in refining our emulator's design, guiding us toward configurations that promise precision and efficiency. This table enumerates the RMSE across various configurations, highlighting the nuanced impact of each hyperparameter on model accuracy. Notably, the configuration with Ips=128, Hidn=256, and Fout=128 emerged as the most effective, demonstrating the lowest overall error. Such insights were instrumental in refining our emulator's design, guiding us toward configurations that promise precision and efficiency.

TABLE II
DIFFERENT RMSE VALUES FOR HYPER-PARAMETER SETTINGS

Hyper-Parameters	**Root Mean Squared Error**				
Ips-Hidn-Fout	**Total**	**LW Heating Rate**	**SW Heating Rate**	**LW Flux**	**SW Flux**
64-64-32	2.839	1.399	0.958	1.687	16.795
64-64-64	2.851	1.374	0.972	1.709	16.898
64-64-128	2.551	1.287	0.894	1.455	15.020
64-64-256	2.696	1.315	0.907	1.557	15.983
64-128-32	2.412	1.291	0.872	1.466	14.017
64-128-64	2.709	1.305	0.901	1.565	16.100
64-128-128	2.576	1.245	0.893	1.441	15.278
64-128-256	2.400	1.225	0.844	1.465	14.072
64-256-32	2.498	1.226	0.873	1.457	14.760
64-256-64	2.359	1.174	0.824	1.477	13.882
64-256-128	2.530	1.252	0.894	1.475	14.922
64-256-256	2.347	1.229	0.882	1.438	13.636
128-64-32	2.460	1.259	0.877	1.438	14.430
128-64-64	2.420	1.213	0.835	1.395	14.275
128-64-128	2.383	1.232	0.849	1.436	13.941
128-64-256	2.429	1.217	0.872	1.443	14.271
128-128-32	2.364	1.200	0.832	1.381	13.890
128-128-64	2.246	1.179	0.821	1.356	13.083
128-128-128	2.768	1.322	0.958	1.614	16.417
128-128-256	2.456	1.211	0.878	1.485	14.464
128-256-32	2.374	1.258	0.838	1.437	13.841
128-256-64	3.720	1.692	1.198	3.273	21.891
128-256-128	**2.162**	1.141	0.792	1.354	**12.567**
128-256-256	2.318	1.166	0.837	1.449	13.588
256-64-32	2.346	1.215	0.840	1.376	13.725
256-64-64	2.206	1.174	0.821	**1.337**	12.799
256-64-128	2.499	1.196	0.863	1.425	14.835
256-64-256	2.560	1.203	0.856	1.436	15.267
256-128-32	2.351	1.184	0.814	1.422	13.830
256-128-64	2.169	1.149	0.801	1.344	12.593
256-128-128	2.956	1.343	0.992	1.805	17.649
256-128-256	2.185	**1.132**	**0.784**	1.341	12.766
256-256-32	2.518	1.257	0.853	1.526	14.855
256-256-64	2.259	1.171	0.799	1.388	13.211
256-256-128	2.283	1.133	0.807	1.358	13.449
256-256-256	2.253	1.135	0.820	1.397	13.196

B. Evaluation of NN-SVGP Emulator Performance

The emulator was trained using the Gpytorch Framework within Python, utilizing the Adam optimization algorithm with an initial learning rate of 0.01. We designated 10% of our roughly 270,000 training samples for validation, reducing the learning rate by half whenever the validation data's MSE did not improve over five epochs. Training occurred on an NVIDIA A100-PCI-40GB system with 256 CPUs and 1TB of memory.

To compare the performance of our NN-SVGP emulator with that of Roh and Song's NN emulator [9], we examined the accuracy and the computational efficiency. Figure 2 shows our NN-SVGP emulator demonstrates an RMSE of 2.162, approximately 28% better than the NN emulator's RMSE of 3.003. This improvement in accuracy is quantified by the Brier Skill Score (BSS).

$$BSS = \left(1 - \frac{NN\text{-}SVGP_{rmse}}{NN_{rmse}}\right) \times 100\,\% \tag{9}$$

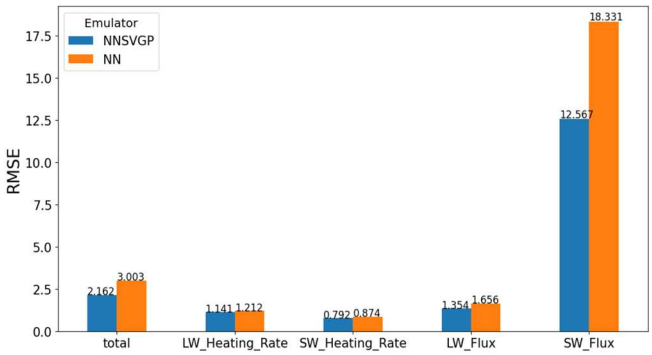

Fig. 2 Comparison of RMSE between NN emulator and NN-SVGP emulator

Additionally, to assess the computational demand of each emulator, we measured the average calculation speed, leveraging a GPU for enhanced performance analysis. As depicted in Figure 3, we found the average computation speed for the NN to be 0.7 ms, while the NN-SVGP averaged 3.2 ms. This increase in computation time for the NN-SVGP emulator is attributed to its greater computational complexity. However, considering the higher accuracy and the ability of the NN-SVGP to quantify uncertainty, the trade-off for increased computation time is justified in applications where predictive precision is crucial.

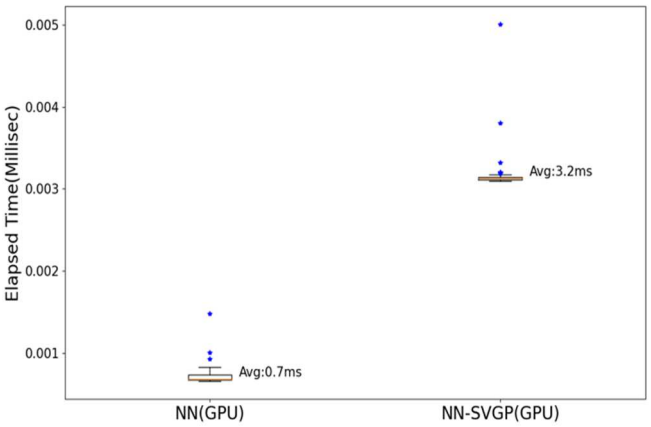

Fig. 3 Comparative Boxplot of Computation Speeds for Neural Network (NN) and Neural Network with Stochastic Variational Gaussian Process (NN-SVGP) Emulators

IV. Conclusion

This study developed an NN-SVGP emulator based on the Gaussian Process Regression (GPR) model using simulation data of radiative physics processes in a numerical forecast model. The model significantly improves prediction accuracy and uncertainty estimation compared to conventional NN emulators. In particular, NN-SVGP can provide confidence intervals for prediction results, an important tool for managing and understanding uncertainty in complex physical processes.

However, regarding computational speed, NN-SVGP is much slower than the NN model. This means that NN-SVGP may be limited in environments requiring high-speed computations. Nevertheless, it is noteworthy that applying the Gaussian process regression model and its potential use in limited environments confirms the possibility of developing an emulator capable of fast computation. These results suggest that NN-SVGP emulators have the potential to contribute to improving the performance of numerical forecasting models. In addition, the ability to estimate uncertainty opens the possibility of application in various fields, such as climate change research and disaster prediction systems.

In assessing the computational efficiency of our models, we noted a significant difference in calculation speeds between NN and NN-SVGP, with NN-SVGP averaging slower computation times. Despite this, the precision, robustness, and ability of NN-SVGP to provide valuable uncertainty estimates far outweigh its computational demands. The enhanced predictive performance and confidence supplied by NN-SVGP affirm its utility, especially in applications where accuracy is critical. Therefore, while the NN model's faster computation may be advantageous in time-sensitive scenarios, the accuracy and reliability benefits of NN-SVGP support its use even with its slower computational speed.

In future research, we will explore ways to improve the NN-SVGP emulator's computational efficiency and increase the uncertainty estimation accuracy. We aim to develop an emulator that can deliver faster and more accurate predictions, thereby contributing to the advancement of numerical forecasting models and their applications in diverse fields. This suggests that integrating sophisticated modeling techniques, such as those employed in NN-SVGP, can lead to more stable and reliable forecasting tools, notwithstanding the comparative slowdown in computation times.

Computationally, it took about three times longer than the NN-SVGP emulator but produced results faster than the radiative physics simulator (RRTMG-K). Despite losing computational time compared to the NN emulator, the NN-SVGP emulator proposed in this study delivered more accurate results. This underlines the potential benefits of adopting advanced emulators over traditional simulators, ensuring more stable operations even with increased computational loads.

Acknowledgment

This work was supported by a research grant from Jeju National University in 2022.

References

[1] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. “A Bayesian approach to the design and analysis of computer experiments,” Oak Ridge National Lab., TN, USA, Tech. Rep. ORNL-6498, Sep. 1988.

[2] T. J. Santner, B. J. Williams, W. I. Notz, and B. J. Williams. “The design and analysis of computer experiments,” New York: Springer, 2003.

[3] A. O'Hagan. “Curve fitting and optimal design for prediction,” *Journal of the Royal Statistical Society: Series B (Methodological)*., vol. 40, no. 1, pp. 1-24, 1978, 10.1111/j.2517-6161.1978.tb01643.x.

[4] T. Beckers. “An introduction to Gaussian process models,” arXiv preprint, arXiv:2102.05497, 2021.

[5] J. Hensman, N. Fusi, and N. D. Lawrence. “Gaussian processes for big data,” arXiv preprint, arXiv:1309.6835, 2013.

[6] P. Ghasemi, M. Karbasi, A. Z. Nouri, M. S. Tabrizi, and H. M. Azamathulla. “Application of Gaussian process regression to forecast multi-step SPEI drought index,” Alexandria Engineering Journal., vol. 60, no. 6, pp. 5375-5392, 2021.

[7] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey. “WeatherBench: a benchmark data set for data-driven weather forecasting,” Journal of Advances in Modeling Earth Systems, vol. 12, no. 11, 2020, doi: 10.1029/2020MS002203.

[8] Y. Lee and J. S. Park. “Generalized Nonlinear Least Squares Method for the Calibration of Complex Computer Code Using a Gaussian Process Surrogate,” Entropy., vol. 22, no. 9, pp. 985, 2020.

[9] Y. A. Seo and J. S. Park. “Expectation-Maximization Algorithm for the Calibration of Complex Simulator Using a Gaussian Process Emulator,” Entropy., vol. 23, no. 1, pp. 53, 2020.

[10] S. Roh and H. J. Song. “Evaluation of neural network emulations for radiation parameterization in cloud-resolving model,” Geophysical Research Letters., vol. 47, no. 21, 2020.

[11] J. Luo, X. Ma, Y. Ji, X. Li, Z. Song, and W. Lu. “Review of machine learning-based surrogate models of groundwater contaminant modeling,” Environmental Research., vol. 238, part. 2, 2023, doi:10.1016/j.envres.2023.117268.

[12] M. S. Go, J. H. Lim, and S. Lee. “Physics-informed neural network-based surrogate model for a virtual thermal sensor with real-time simulation,” International Journal of Heat and Mass Transfer., vol. 214, 2023, doi: 10.1016/j.ijheatmasstransfer.2023.124392.

[13] R. M. Slot, J. D. Sørensen, B. Sudret, L. Svenningsen, and M. L. Thøgersen. “Surrogate model uncertainty in wind turbine reliability assessment,” Renewable Energy., vol. 151, pp. 1150-1162, 2020, doi:10.1016/j.renene.2019.11.101.

[14] M. Tang, Y. Liu, and L. J. Durlofsky. “A deep-learning based surrogate model for data assimilation in dynamic subsurface flow problems,” Journal of Computational Physics., vol. 413, 2020, doi:10.1016/j.jcp.2020.109456.

[15] P. Jiang, Q. Zhou, X. Shao, P. Jiang, Q. Zhou, and X. Shao. “Surrogate-model-based design and optimization,” Springer Singapore., pp. 135-236, 2020.

[16] J. S. Park. “Tuning complex computer codes to data and optimal designs,” University of Illinois at Urbana-Champaign, 1991.

[17] D. D. Cox, J. S. Park, and C. E. Singer. “A statistical method for tuning a computer code to a database,” Computational statistics & data analysis., vol. 37, no. 1, pp. 77-92, 2001, doi:10.1016/S0167-9473(00)00057-8.

[18] M. D. Hoffman, D. M. Blei, and J. Paisley. “Stochastic variational inference,” Journal of Machine Learning Research., 2013.

[19] J. Quinonero-Candela and C. E. Rasmussen. “A unifying view of sparse approximate Gaussian process regression,” The Journal of Machine Learning Research., vol. 6, pp. 1939-1959, 2005.

[20] C. Ding, H. Rappel, T. Dodwell. “Full-field order-reduced Gaussian Process emulators for nonlinear probabilistic mechanics,” Computer Methods in Applied Mechanics and Engineering., vol. 405, 2023, doi:10.1016/j.cma.2022.115855.

[21] Y. A. Seo, Y. Lee, and J. S. Park. “Iterative method for tuning complex simulation code,” Communications in Statistics-Simulation and Computation., vol. 51, no. 7, pp. 3975-3992, 2022, doi:10.1080/03610918.2020.1728317.

[22] H. Liu, Y. S. Ong, X. Shen, and J. Cai. “When Gaussian process meets big data: A review of scalable GPs,” IEEE transactions on neural networks and learning systems., vol. 30, no. 11, pp. 4405-4423, 2020, doi: 10.1109/TNNLS.2019.2957109.

[23] M. M. Noack, H. Krishnan, M. D. Risser, and K. G. Reyes. “Exact Gaussian processes for massive datasets via non-stationary sparsity-discovering kernels," Scientific reports., vol. 13, no. 1, pp. 3155, 2023.

[24] S. Damm, D. Forster, D. Velychko, Z. Dai, A. Fischer, and J. Lücke. “The ELBO of Variational Autoencoders converges to a Sum of Three Entropies,” arXiv preprint, arXiv:2010.14860, 2020.

[25] I. Torroba, C. I. Sprague, and J. Folkesson. “Fully-probabilistic Terrain Modelling with Stochastic Variational Gaussian Process Maps,” arXiv preprint, arXiv:2203.10893, 2022.

[26] M. Ketenci, A. Perotte, N. Elhadad, and I. Urteaga. “A Coreset-based, Tempered Variational Posterior for Accurate and Scalabel Stochastic Gaussian Process Inference,” arXiv preprint, arXiv:2311.01409, 2023.

[27] I. Torroba, M. Cella, A. Teran, N. Rolleberg, and J. Folkesson. “Online stochastic variational gaussian process mapping for large-scale bathymetric slam in real time,” IEEE Robotics and Automation Letters., vol. 8, no. 6, 2023, doi:10.1109/LRA.2023.3264750.

[28] H. Yu and Y. Chen. “Stochastic Motion Planning as Gaussian Variational Inference: Theory and Algorithms,” arXiv preprint, arXiv:2308.14985, 2023.

[29] R. Meng, H. K. Lee, and K. Bouchard. “Stochastic Collapsed Variational Inference for Structured Gaussian Process Regression Networks,” In Conference of the International Federation of Classification Societies, Cham: Springer International Publishing, pp. 253-261, 2022.

[30] C. Hua, X. Cao, B. Liao, and S. Li. “Advances on intelligent algorithms for scientific computing: an overview,” Frontiers in Neurorobotics, vol. 17, 2023, doi:10.3389/fnbot.2023.1190977.

[31] J. Chen, Y. Saad, and Z. Zhang. “Graph coarsening: from scientific computing to machine learning,” SeMA Journal., vol. 79, pp. 187-223, 2022, doi:10.1007/s40324-021-00282-x.