AUTOMATIC EMULATOR AND OPTIMIZED LOOK-UP TABLE GENERATION FOR RADIATIVE TRANSFER MODELS

Luca Martino, Jorge Vicent, Gustau Camps-Valls

Image Processing Laboratory (IPL), Universitat de València, Spain

ABSTRACT

This paper introduces an automatic methodology to construct emulators for costly radiative transfer models (RTMs). The proposed method is sequential and adaptive, and it is based on the notion of the acquisition function by which instead of optimizing the unknown RTM underlying function we propose to achieve accurate approximations. The Automatic Gaussian Process Emulator (AGAPE) methodology combines the interpolation capabilities of Gaussian processes (GPs) with the accurate design of an acquisition function that favors sampling in low density regions and flatness of the interpolation function. We illustrate the good capabilities of the method in toy examples and for the construction of an optimal look-up-table for atmospheric correction based on MODTRAN5.

***Index Terms*—** Radiative transfer model, Gaussian process, emulation, self-learning, look-up table, interpolation, MODTRAN

1. INTRODUCTION

Agape (Ancient Greek ὑγάπη, agápē) is "unconditional love and charity, love of God for man and of man for God."

Atmospheric correction of Earth Observation data aims to derive surface properties (e.g. reflectance) through the inversion of the atmospheric radiative transfer equations, and is perhaps the most crucial step for successful remote sensing applications. *Physically-based* atmospheric correction methods [1] are often preferred over faster *empirical* methods, such as in [2], as their accuracy is also generally higher. These *physically-based* methods rely on the inversion through a Radiative Transfer Model (RTM) [3], which are however computationally expensive and very often impractical for their execution on a pixel-per-pixel basis [4]. To overcome this limitation, large multi-dimensional look-up tables (LUTs) are precomputed for their later interpolation [5]. However, the computation of these LUTs still imposes a large computation burden, requiring techniques of parallelization and execution in computer grids. In order to further reduce this computation burden, a possible strategy is to select an optimal subset of anchor points in order to reduce the error of the interpolation of LUTs. Compact and informative LUTs give raise, in turn, to interesting possibilities for emulating RTMs [6].

The research was funded by the European Research Council (ERC) under the ERC-CoG-2014 SEDAL project (grant agreement 647423), and the Spanish Ministry of Economy and Competitiveness (MINECO) through the project TIN2015-64210-R.

In this work, we address the problem of optimal selection of the points to be included in the LUT. The field has received attention from (apparently unrelated) fields in statistical signal processing and machine learning. The problem has been cast as *experimental optimal design* [7, 8] of interpolators of arbitrary functions f. To reduce the number of direct runs of the system (evaluations of f), a possible approach is to construct an approximation of f starting with a set of support points. This approximation is then sequentially improved incorporating novel points given a suitable statistical rule. This topic is also related to different research areas: optimal nonuniform sampling, quantization and interpolation of continuous signals [9], the so-called Bayesian Optimization (BO) problem [10, 11], and active learning [12]. Finally, an interesting alternative approach is based on *adaptive gridding*, where the aim is to construct a partitioning of $\mathcal{X}$ into cells of equal size, where the cell edges have different lengths depending on their spatial direction. This was the approached followed in [13]. In order to find these lengths, the proposed method uses a Gaussian Process (GP) with an automatic relevant determination (ARD) kernel [14]. A clear problem of such approach is that the number of hyper-parameters to be estimated increases as the input dimension grows.

In this paper we introduce a simpler and more general approach. The proposed method is a sequential, adaptive and automatic construction of the emulator based on the notion of the *acquisition function*, similarly to the BO approach [10, 11]. Unlike in BO, our goal is not the optimization of the unknown underlying function f but its accurate *approximation* $\widehat{f}$. Given a set of initial points, the emulator is built automatically with the online addition of new nodes maximizing the acquisition function at each iteration. Theoretically, the acquisition function should incorporate (a) geometric information of the unknown function f, and (b) information about the distribution of the current nodes. Indeed, areas of high variability of $f(\mathbf{x})$ requires the addition of more points as well as areas with a small concentration of nodes requires the introduction of new inputs. Thus, the experimental design problem is converted into a sequential optimization problem where the function to be optimized involves geometric and spatial information (regardless of the dimensionality of the input space).

2. AUTOMATIC EMULATION

This section introduces the proposed scheme for automatic emulation. We start by fixing the notation and presenting the processing scheme, and then we discuss on the specifics of

the acquisition and interpolation functions. Some implementation details and remarks are finally given.

2.1. Notation and processing scheme

Let us consider a D-dimensional input space $\mathcal{X}$, i.e., $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$ and, for the sake of simplicity, we assume that $\mathcal{X}$ is bounded. Let us denote the system to be emulated as $f(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$, e.g. a complicated transfer equations modeled with an expensive RTM. Given an input matrix $\mathbf{X}_t = [\mathbf{x}_1, \cdots, \mathbf{x}_{m_t}]$ of dimension $D \times m_t$, we have a vector of outputs $\mathbf{y}_t = [y_1, \ldots, y_{m_t}]^\intercal$, where $y_t = f(\mathbf{x}_t)$, where the index $t \in \mathbb{N}^+$ denotes the t-th iteration of the algorithm. Essentially, at each iteration t one performs an *interpolation* $\widehat{f}_t(\mathbf{x}|\mathbf{X}_t, \mathbf{y}_t)$ followed by an *optimization* step that updates the acquisition function, $A_t(\mathbf{x})$, updates the set $\mathbf{X}_{t+1} = [\mathbf{X}_t, \mathbf{x}_{m_t+1}]$ adding a new node, set $m_t \leftarrow m_t + 1$ and $t \leftarrow t + 1$. The procedure is repeated until a suitable stopping condition is met such as a certain maximum number of points is included or a least precision error ϵ is achieved, $\|\widehat{f}_t(\mathbf{x}) - \widehat{f}_{t-1}(\mathbf{x})\| \leq \epsilon$. We assume scalar outputs in order to simplify the description of the technique, yet the algorithm can be easily extended to multi-output settings. Figure 1 shows a graphical representation of a generic automatic emulator.

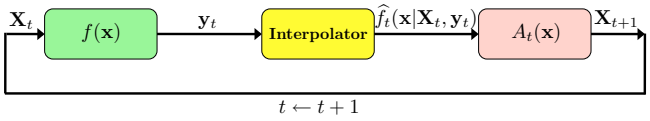

Fig. 1. Graphical representation of an automatic emulator.

2.2. The Acquisition Function

Let us start describing the general properties that a generic acquisition function $A_t(\mathbf{x})$ should satisfy. We consider an acquisition function defined as the product of two functions, a *geometry term* $G_t(\mathbf{x})$ and a *diversity term* $D_t(\mathbf{x})$, i.e.,

$$A_t(\mathbf{x}) \quad = \quad [G_t(\mathbf{x})]^{\beta_t} \, D_t(\mathbf{x}), \quad \beta_t \in [0, 1], \qquad (1)$$

where $G_t(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$, $D_t(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$ and hence $A_t(\mathbf{x}) : \mathcal{X} \mapsto \mathbb{R}$. Moreover, β_t is an increasing function with respect to t, with $\lim_{t\to\infty} \beta_t = 1$ (or $\beta_t = 1$ for $t > t'$).

The first function $G_t(\mathbf{x})$ represents some suitable geometrical information of the hidden function f. The second function $D_t(\mathbf{x})$ depends on the distribution of the points in the current vector $\mathbf{X}_t$. More specifically, $D_t(\mathbf{x})$ will have a greater probability mass around empty areas within $\mathcal{X}$, whereas $D_t(\mathbf{x})$ will be approximately zero close to the support points and exactly zero at the support points, i.e., $D_t(\mathbf{x}_i) = 0$, for $i = 1, \ldots, m_t$ and $\forall t \in \mathbb{N}$. Since f is unknown, the function $G_t(\mathbf{x})$ can be only derived from information acquired in advance or by considering the approximation $\widehat{f}$. Clearly, in this case, the approximation $\widehat{f}$ is usually not well-fit in the first iterations of the algorithm, so that the information provided by $G_t(\mathbf{x})$ should be disregarded or "tempered" (as in tempering strategies for optimization [15]) in these first iterations. This is the reason of using the tempering value β_t. If $\beta_t = 0$, we disregard $G_t(\mathbf{x})$ and $A_t(\mathbf{x}) = D_t(\mathbf{x})$ whereas, if $\beta_t = 1$, we have $A_t(\mathbf{x}) = G_t(\mathbf{x})D_t(\mathbf{x})$. An example of acquisition function is given latter in Figure 2 (cf. §3).

2.3. The Gaussian Process Interpolator

An important step is the selection of the interpolator function $\widehat{f}$. In our work we consider a Gaussian Process (GP) interpolator [14], which has been successfully used in remote sensing [16]. Hereafter, the automatic emulator using GP is called AGAPE. GPs give a full posterior probability from which we can estimate the predictive mean μ_{GP} and variance σ^2_{GP} for a new point $\mathbf{x}$. The predictive mean of the interpolating function for a new point $\mathbf{x}$ is thus given by

$$\mathbb{E}[\widehat{f}_t(\mathbf{x})] = \mu_{\text{GP}}(\mathbf{x}) = \widehat{f}_t(\mathbf{x}|\mathbf{X}_t, \mathbf{y}_t) = \mathbf{k}_x^\intercal \mathbf{K}^{-1} \mathbf{y}_t, \qquad (2)$$

where we defined a kernel function $k(\mathbf{x}, \mathbf{z}) : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, the corresponding kernel matrix $\mathbf{K}_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ of dimension $m_t \times m_t$ containing all kernel entries, and the kernel vector $\mathbf{k}_x = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_{m_t})]^\intercal$ that contains the similarities between the input point $\mathbf{x}$ and the observed ones at iteration t. The interpolation for $\mathbf{x}$ can be simply expressed as a linear combination of $\widehat{f}_t(\mathbf{x}) = \mathbf{k}_x \boldsymbol{\alpha} = \sum_{i=1}^{m_t} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$, where the weights $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{m_t}]^\intercal$ are $\boldsymbol{\alpha} = \mathbf{K}^{-1}\mathbf{y}_t$.[1]

The GP formulation provides also an expression for the predictive variance, which helps us define the diversity term:

$$\mathbb{V}[\widehat{f}_t(\mathbf{x})] = \sigma^2_{\text{GP}}(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_x^\intercal \mathbf{K}^{-1} \mathbf{k}_x, \qquad (3)$$

Here we consider the squared exponential kernel function,

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\delta^2}\right), \qquad (4)$$

where $\|\cdot\|$ is the ℓ_2-norm, and $\delta > 0$ is a positive scalar hyper-parameter. Other alternative kernels can be used depending on the specific application. Note that $\sigma^2_{\text{GP}}(\mathbf{x}_i) = 0$ for all $i = 1, \ldots, m_t$ and $\sigma^2_{\text{GP}}(\mathbf{x})$ depends on the distance among the support points $\mathbf{x}_t$, and the chosen kernel function k and associated hyper-parameter δ. For this reason, the function $\sigma^2_{\text{GP}}(\mathbf{x})$ is a good candidate to represents the distribution of the $\mathbf{x}_t$'s since it is zero at each $\mathbf{x}_i$ and higher far from the points $\mathbf{x}_i$'s. Moreover, $\sigma^2_{\text{GP}}(\mathbf{x})$ takes into account the information of the GP interpolator as previously remarked. Therefore, we consider as the diversity term $D(\mathbf{x}) := \sigma^2_{\text{GP}}(\mathbf{x})$, i.e., $D(\mathbf{x})$ is induced by the GP interpolator. As geometric information, we consider enforcing flatness of the interpolation function, and thus aim to minimize the norm of the the gradient of the interpolating function $\widehat{f}_t$ w.r.t. the input data $\mathbf{x}$, i.e.,

$$G(\mathbf{x}) = \left\| \nabla_x \widehat{f}_t(\mathbf{x}|\mathbf{X}_t, \mathbf{y}_t) \right\| = \left\| \sum_{i=1}^{m_t} \alpha_i \nabla_x k(\mathbf{x}, \mathbf{x}_i) \right\|. \qquad (5)$$

[1] Note that the solution reduces to invert the kernel matrix $\mathbf{K}$. This is related to the fact that *interpolation* is different from *regression*. In regression settings, we typically introduce an extra hyper-parameter to account for the noise variance, and that allows to control the complexity of the solution.

The intuition behind this choice is that wavy regions of f (estimated by $\widehat{f}_t$) require more support points than flat regions. The gradient vector of $k(\mathbf{x}, \mathbf{x}_i)$ in Eq. (4) with $\mathbf{x} = [x^{(1)}, \ldots, x^{(d)}]^\intercal$ and $\mathbf{x}_i = [x_i^{(1)}, \ldots, x_i^{(d)}]^\intercal$, can be computed in closed-form,

$$\nabla_x k(\mathbf{x}, \mathbf{x}_i) = -\frac{k(\mathbf{x}, \mathbf{x}_i)}{\delta^2}[(x^{(1)} - x_i^{(1)}), \ldots, (x^{(d)} - x_i^{(d)})]^\intercal, \quad (6)$$

which is used in Eq. (5) to estimate the geometry function $G(\mathbf{x})$. Several possibilities can be used as $G(\mathbf{x})$, for instance,

$$G(\mathbf{x}) = \left\| \frac{1}{m_t} \sum_{i=1}^{m_t} \nabla_x \left[k(\mathbf{x}, \mathbf{x}_i) \right] \right\|, \quad (7)$$

that reduces the dependence to the current approximation $\widehat{f}_t$. Finally, the acquisition function can be readily defined as $A_t(\mathbf{x}) = [G_t(\mathbf{x})]^{\beta_t} D_t(\mathbf{x})$ as given in Eq. (1). For the parameter we suggest $\beta_t = 1 - \exp(-\gamma t)$, where $\gamma \geq 0$ is a positive scalar, established by the user.

2.4. Optimization and Hyperparameter Tuning

An important advantage of using a GP interpolator is that the application and coding for high dimensional input space is straightforward. Furthermore, the GP formulation suggests directly a suitable function $D(\mathbf{x}) = \sigma^2_{\text{GP}}(\mathbf{x})$. However, unlike in classical piece-wise linear or spline interpolation, we have to tune the hyper-parameter δ. From a practical point of view, we desire to find the highest value δ^* such that the resulting kernel matrix $\mathbf{K}$ is well-conditioned and thus invertible. This was ensured by using a line search for the ℓ_2-norm condition number [17, Chapter 9]. In order to optimize the acquisition function $A_t(\mathbf{x})$, we have employed different parallel tempered-MCMC chains, that is parallel simulated annealing methods as in [15]. Furthermore, in high dimensional input spaces, the stop condition can be simplified considering the evaluation of $\widehat{f}_{t-1}$ and $\widehat{f}_t$ only in a set of checking points to control the variation between $\widehat{f}_{t-1}$ and $\widehat{f}_t$.

3. SIMULATIONS

3.1. Toy Example

In order to test AGAPE, first we consider a toy example where we can compare the achieved approximation $\widehat{f}_t(\mathbf{x})$ with the underlying function $f(\mathbf{x})$ which is unknown in the real-world applications. In this way, we can exactly check the true accuracy of the obtained approximation using different schemes. For the sake of simplicity, we consider the function

$$f(x) = \sum_{i=1}^{4} \exp\left[-a_i(x - b_i)^2\right] + c\log(x), \quad (8)$$

with $x \in \mathcal{X} = (0, 20]$ (hence $D = 1$), $a_1 = a_2 = 0.5$, $a_3 = 5$, $a_4 = 1$, $b_1 = 1$, $b_2 = 6$, $b_3 = 12$, $b_4 = 15$ and $c = 0.35$ (see Fig. 2). We start with $m_0 = 10$ support points, $\mathbf{X}_t = [x_1, \ldots, x_{m_0}]$, randomly chosen at each run within the interval $[0, 15]$. More specifically, $x_j \sim \mathcal{U}([0, 15])$ for $j = 1, \ldots, m_0$. We add sequentially and automatically other 21 points (so that the final number of points is $m_{21} = 31$) using different techniques: **(a)** random choice, uniformly within $\mathcal{X} = (0, 20]$, **(b)** a deterministic approach filling the greatest distance between two consecutive points, adding the middle point, **(c)** AGAPE with Eqs. (7) and $\gamma = 0$ (AGAPE-1) and **(d)** AGAPE with Eqs. (7) and $\gamma = 0.1$ in $\beta_t = 1 - \exp(-\gamma t)$ (AGAPE-2). We averaged the results over 10^4 independent runs. We compare the estimated ℓ_2 and ℓ_∞ distances between $\widehat{f}_t(x)$ and $f(x)$ after the new 11 points have been added. The results are given in Table 1, showing the ℓ_2 and ℓ_∞ distances, with the corresponding 95% confidence intervals, and the percentage of distance reduction w.r.t. the random approach. We can observe that the AGAPE schemes outperform the other strategies. Furthermore, the incorporation of the geometric information in AGAPE-2 provides the best results, reducing the final distances of 57.7% and 67.3% respectively for the ℓ_2 and ℓ_∞ distances, w.r.t. the random approach. Figure 2 depicts the function $f(x)$ in Eq. (8), the approximation $\widehat{f}(x)$, and the acquisition function $A_t(x)$ obtained after 3 iterations of AGAPE, in a specific run.

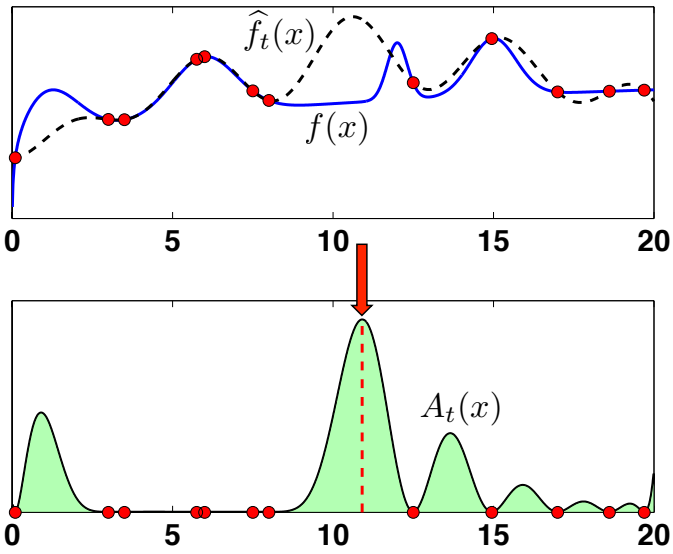

Fig. 2. The function $f(x)$ (top - solid line), its approximation $\widehat{f}_t(x)$ (top - dashed line) and the acquisition function $A_t(x)$ (bottom - solid line) of AGAPE after 3 iterations.

3.2. Emulation of MODTRAN5

Our second example focuses on the optimization of selected points for a MODTRAN5-based LUT. MODTRAN5 is considered as *de facto* standard atmospheric RTM for atmospheric correction applications [3]. This RTM solves the radiative transfer equation in the atmosphere considering the effect of scattering and absorption by gasses and aerosols for a flexible configuration of viewing and illumination conditions and surface reflectance. In our test application, and for the sake of simplicity, we have considered $D = 2$ with the Aerosol Optical Thickness at 550 nm (τ) and ground eleva-

Table 1. Results of the numerical example in Section 3.1.

Approach	Distance ℓ_2	Perc. of Reduction	Distance ℓ_∞	Perc. of Reduction
Random	0.0461 - $[0.0440, 0.0481]$	$--$	0.4744 - $[0.4471, 0.5016]$	$--$
Determ.	0.0388 - $[0.0345, 0.0431]$	-15.83%	0.3905 - $[0.3355, 0.4455]$	-17.68%
AGAPE-1	0.0209 - $[0.0205, 0.0213]$	-54.66%	0.1643 - $[0.1608, 0.1678]$	-65.36%
AGAPE-2	0.0195 - $[0.0191, 0.0199]$	-57.70%	0.1549 - $[0.1517, 0.1582]$	-67.34%

Table 2. Averaged (over 10^3 runs) number of nodes m_t.

Random	Latin Hypercube	AGAPE
28.43	16.69	9.16

tion (h) as key input parameters. The underlying function $f(\mathbf{x})$ consists therefore on the execution of MODTRAN5 at given values of τ and h at the single output wavelength of 760 nm (i.e. bottom of the O_2-A band). The input parameter space is bounded to 0.05-0.4 for τ and 0-3 km for h. In order to test the accuracy of the different schemes, we have evaluated $f(x)$ at all the possible 1750 combinations of 35 values of τ and 50 values of h. Namely, this thin grid represents the ground-truth in this example.

We test **(a)** a random approach choosing points uniformly within $\mathcal{X} = [0.05, 0.4] \times [0, 3]$, **(b)** the Latin Hypercube sampling (see, e.g., [13]) and **(c)** AGAPE (with $\gamma = 0.1$ and the parallel MCMC for the optimization [15]). We start with $m_0 = 5$ points $\mathbf{x}_1 = [0.05, 0]^\top$, $\mathbf{x}_2 = [0.05, 3]^\top$, $\mathbf{x}_3 = [0.4, 0]^\top$, $\mathbf{x}_4 = [0.4, 3]^\top$ and $\mathbf{x}_5 = [0.2, 1.5]^\top$ for all the techniques. We compute the final number of nodes m_t required to obtain an ℓ_2 distance between f and $\widehat{f}$ smaller than 0.03, with the different methods. The results, averaged over 10^3 runs, are shown in Table 2. AGAPE requires the addition of ≈ 4 new points to obtain a distance smaller than 0.03.

4. CONCLUSIONS

We introduced an automatic method to construct emulators and optimal look-up-tables for costly radiative transfer models (RTMs). We proposed an iterative scheme combining the interpolation capabilities of GPs with the design of an acquisition function that fosters a suitable choice of the nodes and flatness of the interpolation function. The combination of the geometric and diversity sampling criteria was possible because the gradient of GP predictive mean function yields a closed-form expression. We illustrated the good capabilities of the method in two examples, one involving the atmospheric correction based on the execution of MODTRAN5. Future work is tied to the development of multi-output schemes and testing in other costly RTMs.

5. REFERENCES

[1] P. North, C. Brockmann, J. Fischer, L. Gomez-Chova, W. Grey, A. Heckel, J. Moreno, R. Preusker, and P. Regner, "Meris/AATSR synergy algorithms for cloud screening, aerosol retrieval and atmospheric correction," *European Space Agency, (Special Publication) ESA SP*, 2008.

[2] L.S. Bernstein, X. Jin, B. Gregor, and S.M. Adler-Golden, "Quick atmospheric correction code: Algorithm description and recent upgrades," *Optical Engineering*, vol. 51, no. 11, 2012.

[3] A. Berk, G.P. Anderson, P.K. Acharya, L.S. Bernstein, L. Muratov, J. Lee, M. Fox, S.M. Adler-Golden, J.H. Chetwynd, M.L. Hoke, R.B. Lockwood, J.A. Gardner, T.W. Cooley, C.C. Borel, P.E. Lewis, and E.P. Shettle, "MODTRAN5: 2006 update," *The International Society for Optical Engineering*, 2006.

[4] G. Camps-Valls, D. Tuia, L. Gómez-Chova, S. Jiménez, and J. Malo, Eds., *Remote Sensing Image Processing*, Morgan & Claypool Publishers, LaPorte, CO, USA, Sept 2011.

[5] L. Guanter, R. Richter, and H. Kaufmann, "On the application of the MODTRAN4 atmospheric radiative transfer code to optical remote sensing," *International Journal of Remote Sensing*, vol. 30, no. 6, pp. 1407–1424, 2009.

[6] J.P. Rivera, J. Verrelst, J. Gmez-Dans, J. Muoz-Mar, J. Moreno, and G. Camps-Valls, "An emulator toolbox to approximate radiative transfer models with statistical learning," *Remote Sensing*, vol. 7, no. 7, pp. 9347–9370, 2015.

[7] K. Chaloner and I. Verdinelli, "Bayesian experimental design: A review," *Statistical Science*, vol. 10, no. 3, pp. 237–304, 1995.

[8] G. da Silva Ferreira and D. Gamerman, "Optimal design in geostatistics under preferential sampling," *Bayesian Analysis*, vol. 10, no. 3, pp. 711–735, 2015.

[9] F. Marvasti, "Nonuniform sampling: Theory and Practice," *Kluwer Academic Publishers*, 2001.

[10] M. U. Gutmann and J. Corander, "Bayesian optimization for likelihood-free inference of simulator-based statistical models," *Journal of Machine Learning Research*, vol. 16, pp. 4256–4302, 2015.

[11] J. Mockus, "Bayesian approach to global optimization," *Kluwer Academic Publishers, Dordrecht*, 1989.

[12] J. Verrelst, S. Dethier, J.P. Rivera, J. Muñoz-Mar, G. Camps-Valls, and J. Moreno, "Active learning methods for efficient hybrid biophysical variable retrieval," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 7, pp. 1012–1016, 2016.

[13] D. Busby, "Hierarchical adaptive experimental design for Gaussian process emulators," *Reliability Engineering and System Safety*, vol. 94, pp. 1183–1193, 2009.

[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2005.

[15] L. Martino, V. Elvira, D. Luengo, J. Corander, and F. Louzada, "Orthogonal parallel MCMC methods for sampling and optimization," *Digital Signal Processing*, vol. 58, pp. 64–84, 2016.

[16] G. Camps-Valls, J. Verrelst, J. Muoz-Mar, V. Laparra, F. Mateo-Jiménez, and J. Gomez-Dans, "A survey on gaussian processes for earth observation data analysis," *IEEE Geoscience and Remote Sensing Magazine*, , no. 6, June 2016.

[17] J. R. Rice, *Matrix Computations and Mathematical Software*, McGraw-Hill, New York, 1981.