PROCEEDINGS of the
23rd International Congress on Acoustics
9 to 13 September 2019 in Aachen, Germany

Meta-Modeling for urban noise mapping

Antoine LESIEUR[(1)], Pierre AUMOND[(2)], Vivien MALLET[(3)], Arnaud CAN[(4)]

[(1)]INRIA, Sorbonne Université, CNRS, LJLL, France, antoine.lesieur@inria.fr
[(2)]IFSTTAR, Cerema, UMRAE, France, pierre.aumond@ifsttar.fr
[(3)]INRIA, Sorbonne Université, CNRS, LJLL, France, vivien.mallet@inria.fr
[(4)]IFSTTAR, Cerema, UMRAE, France, arnaud.can@ifsttar.fr

Abstract

Urban noise maps are usually computed by noise software which couples emission and attenuation rules like CNOSSOS. These models can require several hours to produce a map over a city for a single set of input data. This computational cost makes the models unusable for applications like uncertainty quantification where hundreds of simulations may be required. One solution is to replace the physical model with a meta-model which is extremely fast and yet fairly reproduces the results of the physical model. The strategy is first to reduce the dimension of both inputs and outputs of the physical model, which leads to a reduced model. This reduced model is then replaced by a statistical emulator. The emulator is trained with calls to the reduced model for a set of chosen inputs. The emulator relies on interpolation between the training output values. We applied this approach to the NoiseModelling software over the city of Lorient (France), using Kriging at the emulation step. It required a training set of 2000 calls to the physical model, but then the meta-model is 25 000 times faster than the physical model, while preserving the main behavior of the physical model, with 1.58 dB of mean quadratic error.

Keywords: Noise maps, Computing

INTRODUCTION

Noise pollution is associated with multiple psychological and physical disorders such as sleep disturbance, nervousness, cognitive impairment or hypertension, and is a major challenge to tackle for urban areas. While the World Health Organization (WHO) recommends, in its guideline, an outdoor night value of 40 dB(A) [4], 8 million European adults suffer sleep disturbance due to environmental noise [2].

The European Noise Directive (END) 2002/49/EC [1] defines a common approach for European cities intended to avoid, prevent or reduce the harmful effects of noise. In particular, European cities with more than 100 000 inhabitants are required to produce noise maps at least every 5-year term. These maps should present the distribution of a given indicator over the studied area, in such a way that the exposure of dwellers to environmental noise can be estimated.

A regulatory noise map is a 2D representation of the equivalent average noise level field over an urban area. In software such as NoiseModelling [3], the noise propagation algorithm used for the generation of the noise level field is based on a path finding algorithm which computes the wave amplitude attenuation along the acoustic path between the sources and the receptors, taking into account the reflections and the diffractions. The attenuation rules obey the European standard CNOSSOS [5] which has become the reference since the application of the directive 2002/49/EC in 2015. When this method is applied at a city level, the high number of sources and receptors makes this method computationally expensive. It is not adapted to the generation of hourly or even daily noise maps. The computation time would exceed the time step. An annually averaged equivalent noise level field which relies on annual traffic and weather data is usually represented in standard noise maps.

Recent works [9] have tried to enrich noise maps with data from sensor networks. Yet they have been confronted to a computation time limitation. Also, too much computation time leads to the intractability of large

Monte Carlo simulations and prevents us to apply advanced methods for uncertainty quantification, data assimilation, inverse modeling or network design which can require tens of thousands of calls to the model.

The objective of this paper is to circumvent this limitation by building a meta-model that reproduces the main features of the noise map simulator and requires negligible computational resources. This work relies on previous studies that have been done for urban air quality models [6]. In this paper, we explain the construction of a meta-model based on dimension reduction and interpolation methods applied to each projection on the basis of the reduced output space. The interpolation method we selected in this paper is the Kriging interpolation, it is trained by a Latin Hypercube Sampling (LHS) in the input space and their corresponding simulations at the sample points. Explanations about the reduction and emulation strategy are provided in section 2. In section 3, the case study is described. The meta-model is generated for an area of the city of Lorient (France) using the mean traffic data (speed and flow rate) over a year. The performance of the dimension reduction and emulation are analyzed in dB. The paper concludes on how the meta-model opens the way to several applications.

REDUCTION AND EMULATION OF THE NOISE MAP SIMULATOR

2.1 Description of the framework

Noise mapping simulators use a large amount of input data: flow rate and mean speed in road sections, buildings distribution, general topography, weather data, ground and building absorption and the receptors grid.

Statistical emulation is usually applied to models with a relatively small number of inputs. If we consider each building and road section separately, the Noisemodelling simulator has hundreds of inputs. A common way to reduce the large dimension of input data into a low dimension vector $\mathbf{p} = (p_1 \ \dots \ p_k)^T \in \mathbb{R}^k$ of k parameters is to apply multiplicative coefficients to given input sets. For instance, p_1 can be the multiplicative coefficient applied to the yearly averaged traffic data in all roads of the study area, so that we can change the total traffic by changing p_1—but we cannot change the traffic in a single specific road. We can define through $\mathbf{p}$ all the inputs that we want to vary, e.g., all uncertain inputs if we carry out uncertainty quantification. See section 3.3 for the list of parameters p_i that we considered in our case.

Let $\mathbf{x}^b \in \mathbb{R}^n$ be the output of our model, i.e., the noise levels at the n receptors. Then, we mathematically describe the simulator $\mathscr{M}$ as a multivariate function

$$\begin{array}{rccc} \mathscr{M} & : & \mathbb{R}^k & \to & \mathbb{R}^n \\ & & \mathbf{p} & \mapsto & \mathbf{x}^b = \mathscr{M}(\mathbf{p}) \end{array} \tag{1}$$

It takes several hours to get $\mathscr{M}(\mathbf{p})$ from $\mathbf{p}$. Assume we need to compute a large amount of simulations by varying $\mathbf{p}$ in $I \subset \mathbb{R}^k$. The meta-modeling strategy is to call the model for a restricted amount of inputs $S = \left\{\mathbf{p}^{(1)}, \dots, \ \mathbf{p}^{(r)}\right\} \subset I$ which are optimally distributed in I. With S and $\left\{\mathbf{x}^{b(1)} = \mathscr{M}(\mathbf{p}^{(1)}), \dots, \ \mathbf{x}^{b(r)} = \mathscr{M}(\mathbf{p}^{(r)})\right\} \subset \mathbb{R}^n$, we try to learn the dependency between the inputs and the outputs, which is translated in a meta-model $\widehat{\mathscr{M}}$ which is computationally cheap and has an acceptable performance loss compared to the original simulator:

$$\forall \mathbf{p} \in I, \ \widehat{\mathscr{M}}(\mathbf{p}) = \hat{\mathbf{x}}^b \simeq \mathbf{x}^b \tag{2}$$

2.2 Dimension Reduction

Before the emulation step, which applies to scalar functions, we need to represent the model outputs into a few scalars to be emulated. Indeed, a direct application of the emulation to the simulator output would imply to design an emulator for each receptor. With tens of thousands of receptors, the overall computation time would be too high. The strategy is to project $\mathbf{x}^b$ onto a subspace spanned by a reduced basis $(\mathbf{\Psi}_i)_{i \in [\![1,d]\!]}$ with $d \ll n$.

If $\mathbf{\Psi} = [\mathbf{\Psi}_1, \dots, \mathbf{\Psi}_d] \in \mathbb{R}^{n \times d}$ and $\bar{\mathbf{x}}^b = \dfrac{1}{r} \sum_{i=1}^{r} \mathbf{x}^{b(i)}$, we would expect that

$$\mathbf{x}^b \simeq \bar{\mathbf{x}}^b + \mathbf{\Psi}\mathbf{\Psi}^T (\mathbf{x}^b - \bar{\mathbf{x}}^b) \tag{3}$$

Since $\bar{\mathbf{x}}^b$ and $\mathbf{\Psi}$ are known, one would only have to generate d emulators for $\mathbf{\Psi}^T\mathscr{M}(\mathbf{p}) \in \mathbb{R}^d$ instead of n emulators for $\mathscr{M}(\mathbf{p}) \in \mathbb{R}^n$.

The reduced basis is chosen so as to represent the variability of the noise level field. It is computed with the training set $\mathbf{X} = [\mathbf{x}^{b^{(1)}}, \ldots, \mathbf{x}^{b^{(r)}}]$ obtained with the LHS strategy, and it is determined by a principal component analysis (PCA). The components are sorted so that the i-th component accounts for the i-th largest variability λ_i of the data. The total quadratic error of the approximations of the training set is

$$\sum_{h=1}^{r} \|\mathbf{x}^{b^{(h)}} - \bar{\mathbf{x}}^b - \sum_{j=1}^{d}((\mathbf{x}^{b^{(h)}} - \bar{\mathbf{x}}^b)^T\mathbf{\Psi}_j)\mathbf{\Psi}_j\|^2 = \sum_{\ell=d+1}^{n} \lambda_\ell \tag{4}$$

$(\mathbf{x}^{b^{(h)}} - \bar{\mathbf{x}}^b)^T\mathbf{\Psi}_j$ is the projection of $\mathbf{x}^b - \bar{\mathbf{x}}^b$ in the direction $\mathbf{\Psi}_j$.

If we call $\bar{\mathbf{X}}$ the centered training set $\bar{\mathbf{X}} = [\mathbf{x}^{b^{(1)}} - \bar{\mathbf{x}}^b, \ldots, \mathbf{x}^{b^{(r)}} - \bar{\mathbf{x}}^b]$ then for any $\ell \in [\![1, n]\!]$, the variability of the ℓ-th components satisfies $\bar{\mathbf{X}}\bar{\mathbf{X}}^T\mathbf{\Psi}_\ell = \lambda_\ell\mathbf{\Psi}_\ell$. As we will see in the case study, we can get a high explained variance, like 97%, with only a few vectors in the reduced basis (about five).

2.3 Emulation

For any $\mathbf{p} \in I$, we wish to emulate $\mathbf{\Psi}^T\mathscr{M}(\mathbf{p}) \in \mathbb{R}^d$. We denote $\alpha_i = \mathbf{\Psi}_i^T\mathscr{M}(\mathbf{p}) \in \mathbb{R}$, and $\hat{\alpha}_i$ its emulator which should satisfy $\forall \mathbf{p} \in I,\ \hat{\alpha}_i(\mathbf{p}) \simeq \alpha_i(\mathbf{p})$. If we denote $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_d)$ then

$$\forall \mathbf{p} \in I,\ \hat{\boldsymbol{\alpha}}(\mathbf{p}) \simeq \mathbf{\Psi}^T\mathscr{M}(\mathbf{p}) \tag{5}$$

2.3.1 Kriging

The Kriging emulator is a statistical interpolator which assumes that α_i is a centered stationary stochastic process, follows a Gaussian distribution with a fixed variance σ_i^2: $\forall \mathbf{p} \in I,\ \alpha_i(\mathbf{p}) \sim \mathscr{N}(0, \sigma_i)$, and for any $m \in \mathbb{N}$ and $(\mathbf{p}_1, \ldots, \mathbf{p}_m) \in \mathbb{R}^{k\times m}$, $(\alpha_i(\mathbf{p}_1), \ldots, \alpha_i(\mathbf{p}_m))$ follows a Gaussian distribution of dimension m.

With these assumptions, it is possible to build a unique estimator [7] with the following properties:

- linear, it is a linear combination of the training data;
- unbiased, the prediction matches the data at the training points;
- optimal, in the sense that it minimizes the variance of the prediction error $\mathbb{E}\left[(\hat{\alpha}_i(\mathbf{p}) - \alpha_i(\mathbf{p}))^2\right]$.

A key offline computation of the emulator consists in estimating the covariance kernel of the Gaussian process. We assume the second-order stationarity of the Gaussian process, which implies the covariance between two points only depends of the distance separating them. In addition, for computing considerations, we choose a separable kernel, i.e., a tensor product of k univariate functions, each depending on only one parameter. Finally, for regularity considerations, we choose the so-called Matern 5/2 kernel $\phi(r, \theta)$ which is a good trade-off between the rough exponentially decreasing covariance function and the smooth Gaussian covariance function. This choice is compatible with the assumption that the process is one time differentiable.

The covariance kernel has the following form

$$\forall (\mathbf{p}_1, \mathbf{p}_2) \in I^2,\ Cov(\alpha_i(\mathbf{p}_1), \alpha_i(\mathbf{p}_2)) = \prod_{i=1}^{k} \phi(|p_{1_i} - p_{2_i}|, \theta_i) \tag{6}$$

The covariance kernel is differentiable with respect to θ, hence it is possible to infer θ_{opt} under the maximum likelihood estimation with a gradient descent algorithm.

2.4 Complete Meta-Model

Finally, the complete meta-model reads

$$\hat{\mathbf{x}}^b = \widehat{\mathscr{M}}(\mathbf{p}) = \bar{\mathbf{x}}^b + \mathbf{\Psi}(\hat{\boldsymbol{\alpha}}(\mathbf{p}) - \mathbf{\Psi}^T \bar{\mathbf{x}}^b) \tag{7}$$

The meta-model now has the same outputs as NoiseModelling. Its performance can be compared with NoiseModelling itself or with the physical measurements after interpolation at the observed locations.

Since $\bar{\mathbf{x}}^b$ and $\mathbf{\Psi}\mathbf{\Psi}^T\bar{\mathbf{x}}^b$ are fixed and already known, a call to $\hat{\mathscr{M}}$ for a new set of parameters $\mathbf{p}$, only implies the computation of $\hat{\boldsymbol{\alpha}}(\mathbf{p})$, which consists in computing the covariance vector $\left(Cov\left(\alpha_i(\mathbf{p}), \alpha_i(\mathbf{p}^{(j)})\right)\right)_{j\in[\![1,r]\!]} \in \mathbb{R}^r$ and a sequence of matrix multiplications of a rather cheap computational cost.

META-MODELING OF NOISEMODELLING APPLIED TO LORIENT (FRANCE)

3.1 NoiseModelling

NoiseModelling[1] [3] is an open source software designed to produce noise maps for evaluating the noise impact on urban mobility plans. It is implemented in a GIS software (OrbisGIS[2]) and uses traffic, topographic and meteorological data to generate noise maps. The noise calculation method implemented within NoiseModelling is based on the standard European method CNOSSOS, as a reference method to be used under the Directive 2002/49/EC relating to the assessment and management of environmental noise [5]. In the version used for this paper, the noise calculation method implemented is a slightly simplified version of the standard.

The noise level field is represented by receptors organized as a regular grid in our study. NoiseModelling follows the rules defined by CNOSSOS to compute the noise map.

3.2 Case study

The CENSE project is an interdisciplinary project funded by the French research national agency (ANR) and aims to characterize urban landscapes combining numerical predictions and local observations. The city of Lorient has been chosen to conduct the experiment, and a network of noise sensors will be deployed across the city. The need for a flexible model is crucial to check the compatibility of the observations with the simulation, hence a meta-model is an adequate tool for this project.

The receptors grid is a regular grid with a space step of 10 m where we removed the points inside the buildings. They are located 4 m above the ground, which is approximately the height of the physical sensors (with some variations though). The total number of receptors is 9780.

The traffic inputs are derived from the annual average traffic data gathered by the CEREMA and the city of Lorient traffic department, the French national study center for territory development, for the year 2016. Soil, topography and buildings data come from the IGN[3], the French national geographical institute and the city of Lorient.

3.3 Input parameters

Table 1 shows the list of the $k = 12$ coefficients $\mathbf{p}$, split into two types of parameters, condition parameters and uncertainty parameters.

3.4 Sampling

Since the dimension of the input space is relatively large ($k = 12$), a regular grid to sample I would include too many points. For instance, a grid with 10 discretization points in each direction would require 10^{12} simulations,

[1] See http://noise-planet.org/noisemodelling.html
[2] See http://orbisgis.org/
[3] See http://professionnels.ign.fr/

Table 1. Input parameters and their input range for the meta-model. They are all dimensionless multiplicative coefficients applied to the reference values, except the altitude variation which is an additive value, and the temperature and absorption/reflection coefficient which are the actual values.

Parameter	Minimum value	Maximum value
Condition parameters		
Light vehicle speed	0	2
Heavy vehicle speed	0	2
Light vehicle flow	0	2
Heavy vehicle flow	0	2
Temperature (°C)	-5	30
Uncertainty parameters		
Buildings height	0.5	1.6
Receptors height	0.5	1.6
Slope amplitude	0.9	1.2
Distance to junction	0.9	1.2
Wall absorption coefficient	-0.2	0.2
Ground absorption – G value	0	0.7
Altitude variation (m)	-0.5	0.5

which is computationally intractable. Instead, we make use of a Latin hypercube sampling (LHS) which allows to explore I with a fairly low number of points. The parameters boundaries are slightly enlarged so that the effective boundaries of the input space lives within the interior of the LHS.

In order to evaluate the meta-model performance, we picked a testing sample whose points are chosen to be the farthest from the initial LHS points. The size of the testing sample is 90.

RESULTS

4.1 Dimension reduction

We applied the principal component analysis to a training set which is a Latin hypercube sampling of size 2000. With 4 principal components, the remaining unexplained variance goes below 3 %. Table 2 displays the RMSE of the projection against the real simulation under the number of selected principal components. The RMSE decreases fast with the first vectors, and slowly goes to zero as we increase the number of principal components.

Table 2. Scores of the projection against the size of the basis. The simulation is compared to its projection on the testing sample, at all the receptors

Basis size	3	4	12	20	50	100
RMSE (dB)	2.54	2.43	2.01	1.17	0.91	0.81
NRMSE (%)	4.20	4.01	3.31	1.94	1.50	1.33

4.2 Comparison of the meta-model with the direct model

The Kriging emulator has been generated with the R package **Dicekriging 1.5.5** [8]. The evaluation part consists in comparing the meta-model outputs with the actual outputs of the NoiseModelling simulator, and in observing the performance with the scores described in table 3.

Table 3. Scores for the performance evaluation of a model. $(c_i)_i$ is the simulated sequence. $(o_i)_i$ is the corresponding observed sequence. n is the total number of elements in the sequence. $\bar{c}$ and $\bar{o}$ are respectively the mean of $(c_i)_i$ and $(o_i)_i$

Score	Formula
Bias	$\frac{1}{n}\sum_{i=1}^{n}(c_i - o_i)$
Correlation	$\frac{\sum_{i=1}^{n}(c_i-\bar{c})(o_i-\bar{o})}{\sqrt{\sum_{i=1}^{n}(c_i-\bar{c})^2}\sqrt{\sum_{i=1}^{n}(o_i-\bar{o})^2}}$
Root mean square error (RMSE)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(c_i-o_i)^2}$
Normalized mean square error (NRMSE)	$\frac{RMSE}{\bar{o}}$

The objective is to show the spatial distribution of the scores in order to observe what spatial organization influences the loss of performance, and then to see the distribution of the scores, which is the most intuitive way to check the validity of the meta-model.

4.2.1 Spatialized scores

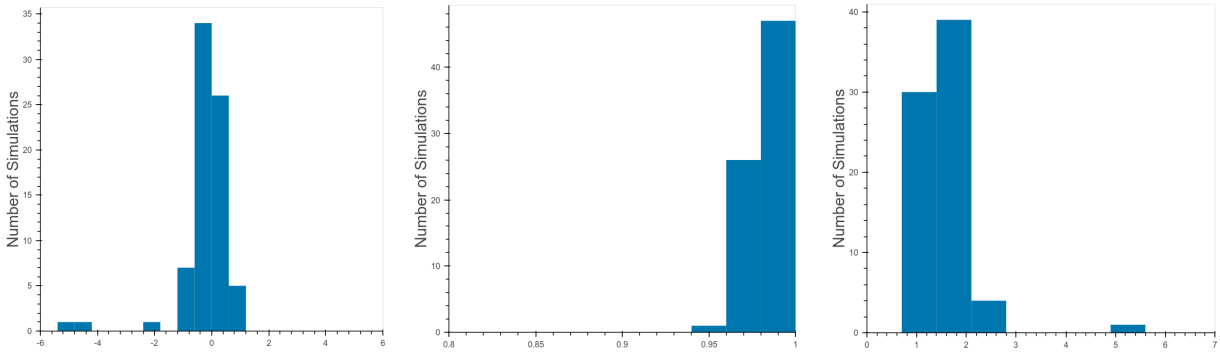
Figure 1 shows the spatial repartition of the bias, correlation and RMSE scores for all the receptors of the study. The results show that the meta-models perform best in area with low buildings density where the influence of reflections and diffractions is lower. The value of the noise sources has little impact on the performance.

(a) Spatialized Bias (dB) (b) Spatialized Correlation (dimensionless) (c) Spatialized RMSE (dB)

Figure 1. Maps of bias, correlation and RMSE between the Kriging meta-model results and the complete simulation.

4.2.2 Scores distribution and local evolution

The scores distribution, over all the simulations of the testing sample, is shown in figure 2. It reflects the distribution of bias, correlation and RMSE over the parameters space. We see on the bias histogram (2a) that the majority of the error is centered around 0 dB and it has a very narrow error variation range. More than 95 % of the simulations have a bias lower than 1 dB, and 93 % of the simulations have a RMSE below 2 dB. The average RMSE across all simulations is 1.58 dB.



(a) Bias distribution (dB) (b) Correlation distribution (c) RMSE distribution (dB)

Figure 2. Distribution of the scores (per simulation) of the meta-model results compared to the simulation.

4.3 Computational costs : building and applying the meta-model

The 2000 simulations have been fully run in parallel during 3 days on 192 cores. The outputs allowed us to generate the reduced basis $\mathbf{\Psi}$ and the interpolation parameters for the Kriging emulator. The offline computation consists in the building of the meta-model. The online time consists in the calls to the meta-model whose computation time are hugely negligible compared to the offline computation time: 65 ms for each call. A simulation with NoiseModelling takes 27 min on 3 cores.

CONCLUSIONS

The generation of a NoiseModelling meta-model required 2000 simulations. The meta-model results remain very close to the results of NoiseModelling with an average RMSE of 2 dB. The computational cost of the meta-model to generate one noise map is 65 ms, compared to 27 min for NoiseModelling, which corresponds to a speed-up of 25 000.

The two stages of the meta-model, dimension reduction and interpolation, have been carried out with a LHS training sample. This sample sweeps the whole input space with a limited size, even in high-dimension spaces. With this training data, we have generated a reduced basis of size 4, which explains 97 % of the variance. We then emulated the projections of the noise maps on each basis vector. With a training sample of 2000 simulations, we obtained satisfactory results with respect to the bias, correlation and RMSE scores, when comparing the meta-model with NoiseModelling on a testing sample.

The computation time saving is the main benefit of the meta-model. With an acceptable loss of performance compared to the original NoiseModelling simulator, we gained an extraordinary amount of computation time. The loss of performance against observed values from local sensors should be carried out in future work. The CENSE project will provide these data since a network of sensors will be installed in the area we presented in this paper.

Several applications can emerge from the meta-model design. Uncertainty quantification is the first that comes to mind, since it requires a very large amount of calls to the simulator which would be intractable with Noise-

Modelling. Data assimilation [10] is also a major investigation area where meta-modeling can be used, by quickly generating a simulation map in real time and correcting it with observations.

REFERENCES

[1] European Commission. Directive 2002/49/ec of the european parliament and of the council of 25 june 2002 relating to the assessment and management of environmental noise, 2002. URL http://data.europa.eu/eli/dir/2002/49/oj.

[2] European Environment Agency. *Noise in Europe 2014*, 2014.

[3] Nicolas Fortin, Erwan Bocher, Judicaël Picaut, Gwendall Petit, and Guillaume Dutilleux. An open-source tool to build urban noise maps in a GIS. In *Open Source Geospatial Research and Education Symposium (OGRS)*, pages 9p, cartes, YVERDON-LES-BAINS, Switzerland, October 2012. URL https://hal.archives-ouvertes.fr/hal-00845701.

[4] Charlotte Hurtley. *Night noise guidelines for Europe*. World Health Organization Europe, Copenhagen, Denmark, 2009. ISBN 978-92-890-4173-7. OCLC: ocn475454508.

[5] Stylianos Kephalopoulos, Marco PAVIOTTI, and Fabienne Anfosso-Lédée. *Common noise assessment methods in Europe (CNOSSOS-EU)*. PUBLICATIONS OFFICE OF THE EUROPEAN UNION, January 2012. doi: 10.2788/31776. URL https://hal.archives-ouvertes.fr/hal-00985998.

[6] Vivien Mallet, Anne Tilloy, David Poulet, Sylvain Girard, and Fabien Brocheton. Meta-modeling of adms-urban by dimension reduction and emulation. *Atmospheric Environment*, 184:37 – 46, 2018. ISSN 1352-2310. doi: https://doi.org/10.1016/j.atmosenv.2018.04.009. URL http://www.sciencedirect.com/science/article/pii/S1352231018302346.

[7] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[8] Olivier Roustant, David Ginsbourger, and Yves Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software, Articles*, 51(1):1–55, 2012. ISSN 1548-7660. doi: 10.18637/jss.v051.i01. URL https://www.jstatsoft.org/v051/i01.

[9] Xavier Sevillano, Joan Claudi Carrié, Francesc Alías, Patrizia Bellucci, Laura Peruzzi, Simone Radaelli, Paola Coppi, Luca Nencini, Andrea Cerniglia, Alessandro Bisceglie, R Benocci, and Giovanni Zambon. Dynamap – development of low cost sensors networks for real time noise mapping. *Noise Mapping*, 3, 01 2016. doi: 10.1515/noise-2016-0013.

[10] Raphaël Ventura, Vivien Mallet, and Valérie Issarny. Assimilation of mobile phone measurements for noise mapping of a neighborhood. *The Journal of the Acoustical Society of America*, 144(3):1279–1292, 2018. doi: 10.1121/1.5052173. URL https://doi.org/10.1121/1.5052173.