Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Gaussian Processes for Optimal Sensor Position

Background & Progress Report

Author:
Adrian LÖWENSTEIN

Supervisor:
Rossella ARCUCCI
Miguel MOLINA-SOLANA

Submitted in partial fulfillment of the requirements for the MSc degree in Computing (Machine Learning) of Imperial College London

June 2019

Contents

Chapter 1

Project Proposal

Project Proposal:

Gaussian processes (GP) have been widely used since the 1970's in the fields of geostatistics and meteorology. Current applications are in diverse fields including sensor placement In this project, we propose the employment of a GP model to calculate the optimal spatial positioning of sensors to study and collect air pollution data in big cities. We will then validate the results by means of a data assimilation software with the data at the proposed positions.

Dataset:

London South Bank University (LSBU) air pollution data (velocity, tracer)

Chapter 2

Background

In this chapter we will cover the literature and the theory that will be used throughout the project. First we will review the context of the project and how it fits into the **MAGIC** project. Then our focus goes to the definition of **Gaussian Processes** (GP) and how they are used in the context of geospatial data. Furthermore, the use of GP relies heavily on **Covariance** matrixes which needs to be estimated. Those tools enable us to create **optimisation** algorithms for the position of sensors. Finally we will quickly explore the concepts of **Data Assimilation** (DA) that will be used to validate the results of the optimisation.

2.1 The MAGIC Project

This work is done in the context of the **Managing Air for Green Inner Cities** project. This is a multidisciplinary project and has for objective to find solutions to the pollution and heating phenomenons in cities. Traditionally, urban environmental control relies on polluting and energy consuming heating, ventilation and cooling (HVAC) systems. The usage of the systems increases the heat and the pollution levels, inducing an increased need for the HVAC. The MAGIC project aims at breaking this vicious circle and has for objective to provide tools to make possible the design of cities acting as a natural HVAC system.

This has been extensively discussed by Song et al. (2018). For this purpose, integrated management and decision-support system is under development. It includes a variety of simulations for pollutants and temperature at different scales; a set of mathematical tools to allow fast computation in the context of real-time analysis; and cost-benefit models to asses the viability of the planning options and decisions.

As explained by Song et al. (2018), the test site which has been selected to conduct the study is a real urban area located in London South Bank University (LSBU) in Elephant and Castle, London. In order to investigate the effect of ventilation on the cities problem, researchers in the MAGIC project have created simulations and experiments both in outdoor and indoor conditions, on the test site. They used wind tunnel experiments and computational fluid dynamics (CFD) to simulate the out-

door environment. Further works include the development of reduced-order modelling (ROM) in order to make faster the simulations while keeping a high level of accuracy (Arcucci et al., 2018).

Another key research direction in the use Data Assimilation (DA) and more specifically Variational DA (VarDA) for assimilating measured data in real time and allowing better prediction of the model in the near future (Arcucci et al., 2018). The further use of those method would be the optimisation of the position of the sensors which provide information for the VarDA.

2.2 Gaussian Processes

In this chapter we will review Gaussian Processes (GP) which are probabilistic models for spatial predictions based on observations assumption.

As explained by Rasmussen and Williams (2006, p. 29), the history of Gaussian Processes goes back at least as far as the 1940s. A lot of usages were developed in various fields. Notably for predictions in spatial statistics (Cressie, 1991). Applied in particular in Geostatistics with methods known as *kringing*, and in Meteorology. Gradually GP started to be be used in more general cases for regression. Nowadays it is often used in the context of Machine Learning.

For our problem we will be modeling the data of the sensor with GPs.

2.2.1 Sensor Data Modelling

2.2.1.1 Multivariate Gaussian Distribution

GP will serve as a basis tool in our project. In the space we are monitoring we have a certain number of sensors measuring a certain quantity, such as temperature, pressure, speed of the wind or the concentration of a pollutant at a given position. We assume that the measured quantity has a *multivariate Gaussian joint distribution* between each point of the space. The associated random variable is $\mathcal{X}_\mathcal{V}$ for the set of locations $\mathcal{V}$ we would have the following distribution : $P(\mathcal{X}_\mathcal{V} = \boldsymbol{x}_\mathcal{V}) \sim \mathcal{N}(\mu_\mathcal{V}, K_{\mathcal{V}\mathcal{V}})$, or explicitly :

$$P(\mathcal{X}_\mathcal{V} = \boldsymbol{x}_\mathcal{V}) \frac{1}{(2\pi)^{n/2}|K_{\mathcal{V}\mathcal{V}}|} \exp^{-\frac{1}{2}(\boldsymbol{x}_\mathcal{V}-\mu_\mathcal{V})^T K_{\mathcal{V}\mathcal{V}}^{-1}(\boldsymbol{x}_\mathcal{V}-\mu_\mathcal{V})} \tag{2.1}$$

2.2.1.2 Prediction with Gaussian Processes

Let us still consider that we have the set of locations $\mathcal{V}$ and a set of sensors $\mathcal{A}$. In order to predict the quantity at positions were we have no sensors ($\mathcal{V}\backslash\mathcal{A}$) we can use a Gaussian Process. This GP is associated with a **mean function** $\mathcal{M}(\cdot)$ and a symmetric positive-definite **kernel function** $\mathcal{K}(\cdot,\cdot)$. We will denote the mean function values for a set of positions $\mathcal{A}$ by $\mu_\mathcal{A}$ and the kernel function values, or covariance matrix,

between those points by $K_{\mathcal{A}}$. More detailed definitions are available in Rasmussen and Williams (2006, p. 13-16).

For a set of observations $\boldsymbol{x}_{\mathcal{A}}$ at positions $\mathcal{A}$ we can express for a finite set of other positions $\mathcal{V}\backslash\mathcal{A}$ the conditional distribution of those values. This means that we are able, for each point $y \in \mathcal{V}\backslash\mathcal{A}$, to predict the mean and the variance of $\boldsymbol{x}_y$. Using conditional distribution for the Multivariate Gaussian Distribution (Deisenroth et al., 2018, p. 193), we are able to express the following :

$$P(\mathcal{X}_y|\boldsymbol{x}_{\mathcal{A}}) = \mathcal{N}(\mu_{y|\mathcal{A}}, K_{y|\mathcal{A}}) \tag{2.2}$$
$$\mu_{y|\mathcal{A}} = \mu_y + K_{y\mathcal{A}}K_{\mathcal{A}\mathcal{A}}^{-1}(y - \mu_y) \tag{2.3}$$
$$K_{y|\mathcal{A}} = K_{yy} - K_{y\mathcal{A}}K_{\mathcal{A}\mathcal{A}}^{-1}K_{\mathcal{A}y} \tag{2.4}$$

An important point to notice is that the predicted covariance for the point y is not dependent of the values measured at $\mathcal{A}$, this is really useful because if allows us to define the uncertainty at y without using only the covariance.

2.2.2 Scalable Gaussian Processes

The biggest weakness of Standard GPs is their complexity. For p training points, the algorithm requires the inversion of a $p \times p$ covariance matrix K_{pp}. Liu et al. (2018) gives an extensive review of all methods used to make the GPs more scalable.

Liu et al. (2018) provides a survey proposing several startegies, both global and local.
Note : This section needs a lot a further development, in order to find an implement the best possible approach.

2.2.3 Covariance Matrix Estimation

We have seen how GPs are defined and made more scalable. In order to have good results we need to have a good estimate of the covariance matrix between the points of our space.

In our specific case, we already have at our disposal a very dense network of measurement. With more than $100'000$ different locations we don't need to explore the space outside of those points. Unfortunately the sample covariance is a very bad estimator of the true covariance in high dimensional settings such as ours (Pourahmadi, 2011).

We will see the properties that our covariance to accurately model our space, before discussing the best ways of estimating the covariance matrix estimation.

2.2.3.1 Properties of Covariance

By definition a covariance matrix must be **positive-definite** and **symmetrical** (Rasmussen and Williams, 2006, p. 80).

A covariance function between two inputs x and x' is **stationary** when it is invariant to translation. Thus, when it is a function of $x' - x$.
In a covariance function that is **isotropic**, we remove the dependance of the direction and, it become a function of the distance between the two points : $|x' - x|$. The isotropy of the covariance function is a stronger assumption than

In our problem we can't assume that the process is stationary nor isotropic. Our space is 3-dimensional and not homogeneous. The presence of the buildings and other obstacles that a likely to make environment variables less smooth (Paciorek and Schervish, 2004). Also it has been shown by Krause et al. (2008) that non-stationary covariance matrixes give better results than stationary or isotropic. This makes us choose a non-stationary covariance matrix for the GPs of our problem.

2.2.3.2 Covariance Estimation

We consider the process Y in p different locations at T sampling times. $\boldsymbol{Y}_t = (Y_{1t}, \ldots, Y_{pt})$, with $t = 1 \ldots T$. We want to estimate the covariance matrix Σ.

Sample Covariance

The simplest estimator of the covariance matrix is the **sample covariance matrix $\boldsymbol{S}$** directly computed from the captured data.

$$\boldsymbol{S} = \frac{1}{T-1} \sum_{t=1}^{T} (\boldsymbol{Y}_t - \bar{\boldsymbol{Y}}) \cdot (\boldsymbol{Y}_t - \bar{\boldsymbol{Y}}_t)^T, \quad \bar{\boldsymbol{Y}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{Y}_t \tag{2.5}$$

Unfortunately this covariance is singular when $p >> T$ (Fan et al., 2015), and the accumulation of errors due to the number of parameters to estimate. One additional assumption that we need to make for this covariance is its the sparsity of the covariance which is needed to reduce the number of parameters and is often realistic in practice.

Estimating a Spatial Covariance

Nott and Dunsmuir (2002) proposed to fit local variograms in order to approximate the covariance matrix. It is a method that was developped in Cressie (1991).

An other method is the use of spatial deformation of kernel to orther of obtain non-stationarity (Sampson and Guttorp, 1992).

Several different approaches are developed in the literature, notably the works of (Fan et al., 2015) and Guttorp and Sampson (1994) which it more focused on spatial data.
Note : This section needs a lot a further development, in order to find an implement the best possible approach.

2.3 Sensor Position Optimisation

Now that we have modelled the relationship between the positions using GPS we can establish an algorithm that was developed by Krause et al. (2008). The process of placing sensors in an optimal way is called in spatial statistics, *sampling* or *experimental design*. We want to find the optimal way place a number of k sensors (indexed by $\mathcal{A}$) inside the set of possible sensor locations $\mathcal{S}$. So that we have $\mathcal{A} \subseteq \mathcal{S} \subseteq \mathcal{V}$.

For the rest of this section we assume that we have at our disposal a good estimate a of the covariance matrix between each point. In practise this is not that obvious as we have seen in section 2.2.3. The following is valid for any covariance matrix that is symmetric and positive-definite.

First we will define how to characterise a good design in term of sensor placement. Then we will define the main optimisation algorithm and its improvements.

2.3.1 Placement Criterion

2.3.1.1 Entropy Criterion

Intuitively a good way of measuring uncertainty is the *entropy*. By observing the conditional entropy of the location where no sensor was placed $\mathcal{V}\backslash\mathcal{A}$, we can estimate the uncertainty remaining for those locations. We define the following conditional entropy of the un-instrumented location knowing the instrumented ones (Cover and Thomas, 1991, p. 16) :

$$H(\mathcal{X}_{\mathcal{V}\backslash\mathcal{A}}|\ \mathcal{X}_{\mathcal{A}}) = \mathbb{E}_{p(\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}},\boldsymbol{x}_{\mathcal{A}})} \log p(\mathcal{X}_{\mathcal{V}\backslash\mathcal{A}}|\ \mathcal{X}_{\mathcal{A}}) \tag{2.6}$$

$$H(\mathcal{X}_{\mathcal{V}\backslash\mathcal{A}}|\ \mathcal{X}_{\mathcal{A}}) = -\sum_{\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}}\in\mathcal{X}_{\mathcal{V}\backslash\mathcal{A}}} \sum_{\boldsymbol{x}_{\mathcal{A}}\in\mathcal{X}_{\mathcal{A}}} p(\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}}, \boldsymbol{x}_{\mathcal{A}}) \log p(\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}}|\boldsymbol{x}_{\mathcal{A}}) \tag{2.7}$$

$$H(\mathcal{X}_{\mathcal{V}\backslash\mathcal{A}}|\ \mathcal{X}_{\mathcal{A}}) = -\int p(\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}}, \boldsymbol{x}_{\mathcal{A}}) \log p(\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}}|\boldsymbol{x}_{\mathcal{A}})\, d\boldsymbol{x}_{\mathcal{V}\backslash\mathcal{A}}\, d\boldsymbol{x}_{\mathcal{A}} \tag{2.8}$$

For the specific case of the *Multivariate Gaussian Distribution*, Krause et al. (2008) gives us the expression of the entropy of a point $y \in \mathcal{V}\backslash\mathcal{A}$ conditioned by the set $\mathcal{A}$ in a closed form depending exclusively on the covariance between those elements : $K_{y|\mathcal{A}}$. Thus we have :

$$H(\mathcal{X}_y|\ \mathcal{X}_\mathcal{A}) = \frac{1}{2}\log K_{y|\mathcal{A}} + \frac{1}{2}\left(1 + \log 2\pi\right) \tag{2.9}$$

This formulation is extremely useful because we can directly use the expression of the covariance given by the *Gaussian Process* expressed previously (2.4).

This conditional entropy can also be expressed using the *chain rule* (Cover and Thomas, 1991, p. 16) :

$$\begin{aligned} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}|\ \mathcal{X}_\mathcal{A}) &= H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}},\ \mathcal{X}_\mathcal{A}) - H(\mathcal{X}_\mathcal{A}) && (2.10)\\ &= H(\mathcal{X}_\mathcal{V}) - H(\mathcal{X}_\mathcal{A}) && (2.11)\end{aligned}$$

The optimal set of sensors $\mathcal{A}^*$ with size $|\mathcal{A}^*| = k$, is then defined for the minimum of this entropy. If we minimise this quantity we will reduce the uncertainty on the un-instrumented locations $\mathcal{V}\setminus\mathcal{A}$:

$$\begin{aligned} \mathcal{A}^* &= \arg\min_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}|\ \mathcal{X}_\mathcal{A}) && (2.12)\\ &= \arg\min_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_\mathcal{V}) - H(\mathcal{X}_\mathcal{A}) && (2.13)\\ &= \arg\max_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_\mathcal{A}) && (2.14)\end{aligned}$$

2.3.1.2 Mutual Information Criterion

The Entropy criterion provides an intuitive way to solve the problem, unfortunately, during experiments referenced by Krause et al. (2008), it was noted that this criterion has a tendency to induce placement at the border of the space, and therefore wastes a lot of precious information. This is due to the fact that the entropy criterion is *indirect* because it measures the uncertainty of the selected sensor position, instead of measuring the uncertainty of every other location of the space (which are the ones we are interested in).

An other criterion was proposed by Caselton and Zidek (1984) : the *Mutual Information* (MI) Criterion. We try to maximise the mutual information between the set of selected sensors $\mathcal{A}$ and the rest of the space $\mathcal{V}\setminus\mathcal{A}$. Using the definitions of MI provided by Cover and Thomas (1991, p. 19) :

$$\begin{aligned} \mathcal{A}^* &= \arg\max_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} I(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}},\ \mathcal{X}_\mathcal{A}) && (2.15)\\ &= \arg\max_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}|\mathcal{X}_\mathcal{A}) && (2.16)\end{aligned}$$

Experimentally, Krause et al. (2008) explain that the mutual information outperforms entropy placement optimisation. They also argue that this criterion relies heavily on the quality of the model $P(\mathcal{X}_\mathcal{V})$ (i.e. how the covariance is modelled, see section 2.2.3) for giving good results.

2.3.2 Approximation Algorithm

As stated by Krause et al. (2008), the problem is a *NP-complete problem*. Therefore we present here an algorithm that is able to approximate in polynomial time the optimal solution, with a constant factor guarantee.

Let us define the initial sensor set $\mathcal{A}_0 = \emptyset$. At each iteration we have a new sensor set : $\mathcal{A}$, and for a point y, we define : $\bar{\mathcal{A}} = \mathcal{V}\backslash(\mathcal{A} \cup y)$. We also have the set of positions that can be selected as sensors : $\mathcal{S}$, and the associated set : $\mathcal{U} = \mathcal{V}\backslash\mathcal{S}$.

The idea is to greedily add sensors until we reach the wanted number (k). This greedy approach in enabled by the use the *chain rule* of entropy. Starting from $\mathcal{A} = \mathcal{A}_0$, at each iteration we add to $\mathcal{A}$ the point $y \in \mathcal{S}\backslash A$ which decreases the least the current MI :

$$y* = \arg\max_{y \in \mathcal{S}\backslash A} MI(\mathcal{A} \cup y, \bar{\mathcal{A}}) - MI(\mathcal{A}, \mathcal{V}\backslash\mathcal{A}) \tag{2.17}$$

$$= \arg\max_{y \in \mathcal{S}\backslash A} H(y|\mathcal{A}) - H(y|\bar{\mathcal{A}}) \tag{2.18}$$

In our specific case with the Multivariate Gaussian Distribution, we can take the formulation presented in equation 2.9 and rewrite the objective to :

$$y* = \arg\max_{y \in \mathcal{S}\backslash A} \frac{1}{2}\log K_{y|\mathcal{A}} - \frac{1}{2}\log K_{y|\bar{\mathcal{A}}} \tag{2.19}$$

$$= \arg\max_{y \in \mathcal{S}\backslash A} \log\left(\frac{K_{y|\mathcal{A}}}{K_{y|\bar{\mathcal{A}}}}\right) \tag{2.20}$$

$$= \arg\max_{y \in \mathcal{S}\backslash A} \frac{K_{y|\mathcal{A}}}{K_{y|\bar{\mathcal{A}}}} \tag{2.21}$$

Here we can use the GP that we have defined earlier in equation 2.4 to finally write the problem to solve at each iteration of the algorithm :

$$y* = \arg\max_{y \in \mathcal{S}\backslash A} \frac{K_{yy} - K_{y\mathcal{A}}K_{\mathcal{A}\mathcal{A}}^{-1}K_{\mathcal{A}y}}{K_{yy} - K_{y\bar{\mathcal{A}}}K_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1}K_{\bar{\mathcal{A}}y}} \tag{2.22}$$

Then we update the set of sensors such that $\mathcal{A} = \mathcal{A} \cup y^*$, and then restart the process until $|\mathcal{A}| = k$. It is described in algorithm 1

This algorithm computes a solution that is very close to the optimal one if the discretisation of the space is small enough. The bound of solution obtained is approximately 63% of the optimal solution. If the true optimal set is $\mathcal{A}^*$ and $\hat{\mathcal{A}}$ is the solution returned by the greedy algorithm, then Krause et al. (2008) proves, for a small $\epsilon > 0$, that :

$$MI(\hat{\mathcal{A}}) \geq (1 - \frac{1}{e}) \cdot MI(\mathcal{A}^*) - k\epsilon \tag{2.23}$$

To prove that they use the notion of *submodularity* (Nemhauser et al., 1978) applied to the $MI(\cdot)$ function. Intuitively this represents the notion of *diminishing returns* : adding a sensor to a small set of sensors has more benefits than adding a sensor to a large set of sensors.

Data: Covariance matrix $K_{\mathcal{V}\mathcal{V}}$, k, $\mathcal{V}$
Result: Sensor Selection $\mathcal{A}$
begin;
for $j \leftarrow 1$ **to** k **do**
  for $y \in \mathcal{S} \backslash \mathcal{A}$ **do**
    $\delta_y \leftarrow \frac{K_{yy} - K_{y\mathcal{A}} K_{\mathcal{A}\mathcal{A}}^{-1} K_{\mathcal{A}y}}{K_{yy} - K_{y\bar{\mathcal{A}}} K_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1} K_{\bar{\mathcal{A}}y}}$
  end
  $y^* \leftarrow \arg\max_{y \in \mathcal{S} \backslash A} \delta_y$
  $\mathcal{A} \leftarrow \mathcal{A} \cup y^*$
end

Algorithm 1: Greedy Algorithm

2.3.3 Improvements over the Algorithm

We have exposed the main greedy algorithm to solve that optimisation problem. Krause et al. (2008) explains that the complexity is a big issue for scaling this algorithm up. If we consider that ne number of locations in our space is $|\mathcal{V}| = n$, and that the number of sensors to place is k, the complexity of the main algorithm is $\mathcal{O}(kn^4)$. They propose two solutions to this issue : a *lazy procedure* that cuts the complexity to $\mathcal{O}(kn^3)$ and a *local kernel* strategy that reduces it to $\mathcal{O}(kn)$

2.3.3.1 Lazy Procedure

This procedure uses strategically the notion of *submodularity* and *priority queues*. We can describe it intuitively : When a sensor is selected y^*, the other nearby points will have de decreased δ_y and will therefore be less desirable. Therefore if we maintain a priority queue with the orderer values of δ_y at each step we will take the top values and update them to they current value successively. If the current value needs to be updated and the location is close to the previous optimum, it would have a small δ_y and be send back in the queue. If we meet a point which has been updated and is still a the top of the queue, it means that this point is our new optimum. This technique can be efficiently applied using **binary heaps**. This algorithm is described in algorithm 2.

2.3.3.2 Local Kernels

This procedure takes advantage of the structure of the covariance matrix. For many GPs, correlation decreases exponentially with the distance between points. So, the idea here is to truncate the covariance matrix to points that are the most correlated.

If we want to compute the covariance between a point of interest y and a set of points $\mathcal{B} \subset \mathcal{V}$ we have the original covariance matrix $K_{y\mathcal{B}}$. When we remove from the set $\mathcal{B}$, the set of elements $x \in \mathcal{S}$ such that $|\mathcal{K}(y, x)| > \epsilon$ we define the new set $\tilde{\mathcal{B}}_y = N(y, \epsilon)$. This set is defined such as it contains less than d locations : $|N(y, \epsilon)| \leq d$. The

Data: Covariance matrix $K_{\mathcal{VV}}$, k, $\mathcal{V}$
Result: Sensor Selection $\mathcal{A}$
initialisation;
$\mathcal{A} = \emptyset$
foreach $y \in \mathcal{S}$ **do** $\delta_y \leftarrow +\infty$;
begin;
for $j \leftarrow 1$ **to** k **do**
  foreach $y \in \mathcal{S}\backslash\mathcal{A}$ **do** $current_y \leftarrow$ False ;
  while *True* **do**
    $y^* \leftarrow \arg\max_{y \in \mathcal{S}\backslash A} \delta_y$
    if $current_{y^*}$ **then** break;
    δ_{y^*} is updated with 2.22
    $current_y \leftarrow$ True
  end
  $\mathcal{A} \leftarrow \mathcal{A} \cup y^*$
end

Algorithm 2: Lazy Algorithm

truncated covariance associated with this set is named : $\tilde{K}_{y\mathcal{B}}$. Finally, we define the approximate conditional entropy $\tilde{H}_\epsilon(y|\mathcal{X}) \simeq H(y|\mathcal{X})$, computed with truncated covariance of the points of $N(y, \epsilon)$. The δ_y values are initialised by taking the difference between the true entropy and the truncated covariance, as :

$$\delta_y = \tilde{H}_\epsilon(y|\mathcal{A}) - \tilde{H}_\epsilon(y|\bar{\mathcal{A}}) \tag{2.24}$$
$$= H(y) - \tilde{H}_\epsilon(y|\mathcal{V}\backslash y) \tag{2.25}$$

Or equivalently by keeping the previous notations:

$$\delta_y = \frac{K_{yy}}{K_{yy} - \tilde{K}_{y\mathcal{V}\backslash y}\tilde{K}^{-1}_{\mathcal{V}\backslash y\mathcal{V}\backslash y}\tilde{K}_{\mathcal{V}\backslash yy}} \tag{2.26}$$

As for the other iterations (when $\mathcal{A} \neq \emptyset$), we have :

$$\delta_y = \tilde{H}_\epsilon(y|\mathcal{A}) - \tilde{H}_\epsilon(y|\bar{\mathcal{A}}) \tag{2.27}$$
$$= \frac{K_{yy} - \tilde{K}_{y\mathcal{A}}\tilde{K}^{-1}_{\mathcal{A}\mathcal{A}}\tilde{K}_{\mathcal{A}y}}{\tilde{K}_{yy} - \tilde{K}_{y\bar{\mathcal{A}}}\tilde{K}^{-1}_{\bar{\mathcal{A}}\bar{\mathcal{A}}}\tilde{K}_{\bar{\mathcal{A}}y}} \tag{2.28}$$

We explicitly define the steps this in algorithm 3.

Krause et al. (2008) proves that this algorithm approximates the optimal solution with complexity $\mathcal{O}(nd^3 + nk + kd^4)$. The solution found by this algorithm is close to the real optimum within given bounds.

This two strategies can be combined in order to make the problem even more scalable.

Data: Covariance matrix $K_{\mathcal{VV}}$, k, $\mathcal{V}$, ϵ
Result: Sensor Selection $\mathcal{A}$
initialisation;
$\mathcal{A} = \emptyset$
foreach $y \in \mathcal{S}$ **do** $\delta_y \leftarrow 2.26$;
begin;
for $j \leftarrow 1$ **to** k **do**
  foreach $y \in \mathcal{S} \backslash \mathcal{A}$ **do** $current_y \leftarrow$ False ;
  while *True* **do**
    $y^* \leftarrow \arg\max_{y \in \mathcal{S} \backslash A} \delta_y$
    $\mathcal{A} \leftarrow \mathcal{A} \cup y^*$
    foreach $y \in N(y^*, \epsilon)$ **do** $\delta_y \leftarrow 2.28$;
  end
end

Algorithm 3: Local Kernel Algorithm

Chapter 3

Progress

This chapter exposes the work done until now in the project. It will focus on the data exploration done, the issues encountered in the covariance estimation, and the results of the algorithms developed in the sensor optimisation procedure. All the commented codes are available on the Github repository of the project.

3.1 Framework

In order to implement the algorithm and manipulate the VTK files of the dataset and I relied on Python 3.7 scripts that I runned the cluster of the DSI.

I have implemented a set of function to load and save the dataset and various intermediary computation results. Additional methods for the quick manipulation and the visualisation of the space. And functions that implement the covariance matrix estimate and the optimisation algorithms.

3.2 Data Exploration

As a main dataset we use a the simulation results on an unstructured mesh of measurements with a very high density in the low middle of the space. The simulation used has $100'040$ points over 988 timestamps. Several fields are present in the simulation : the **pressure**, the **tracer** concentration, the **background tracer** concentration and the **velocity**. We represent in the next plots 2 cuts made to visualise the tracer and the tracer background.

3.3 Covariance Estimation

In order to test optimisation algorithm I have make the choice to first use a subset of the whole dataset containing $2'488$ point and created by cropping the space to a cube of $30 \times 30 \times 30 \times$ at the center of the original space. On this subspace I have made the choice to compute an isotropic covariance matrix (Exponential kernel, Matern 3/2

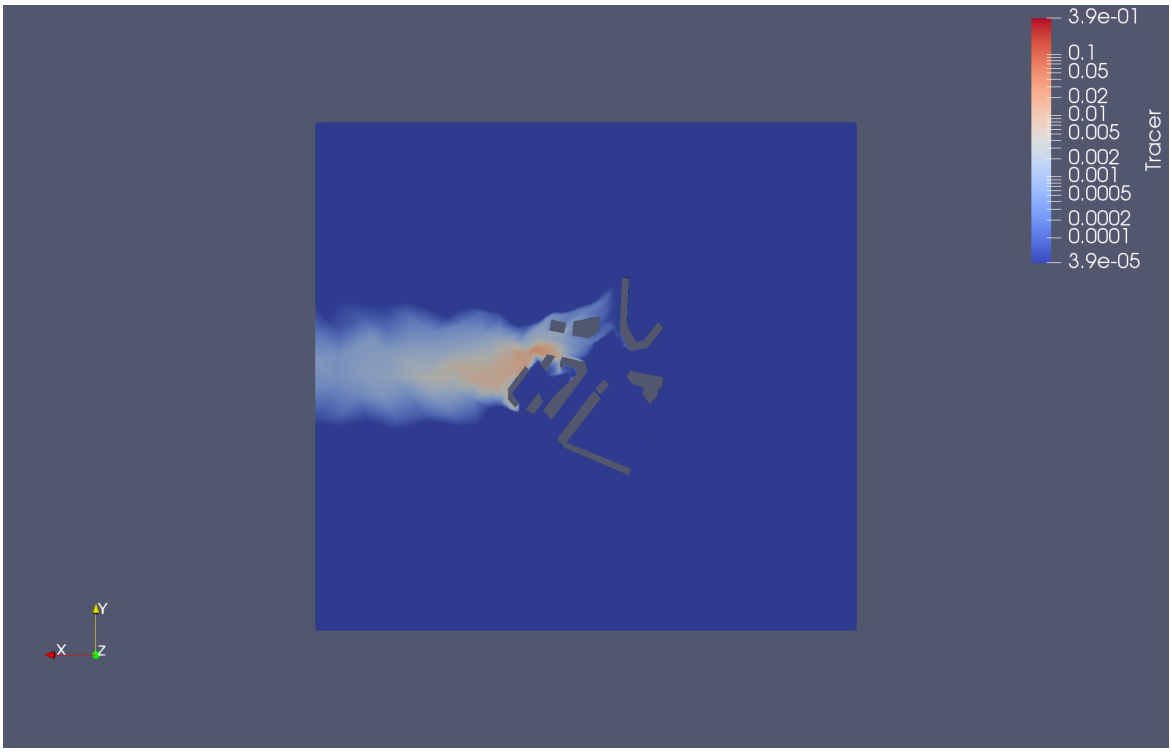

Figure 3.1: Cut of tracer at t = 639

and 5/2) with a length-scale of $l = 3$ for testing the performance of the optimisation algorithm.

I had initially tried it with the sample covariance. As a matter of fact, it did'nt work properly, resulting in problems due to the positive-definiteness and the singularity of the sample covariance.

Also, the memory taken by those covariance matrix shows the necessity of the usage of a sparse matrix representation for the covariance.

3.4 Sensor Optimisation

We have tested two algorithm exposed earlier :

The standard greedy approach (algorithm 1).
The Lazy approach developed using python heaps. (algorithm 2). This version is as expected much faster than the standard version.

We plot in the following figure the results obtained in the setting described earlier for a number of sensors $k = 10$.

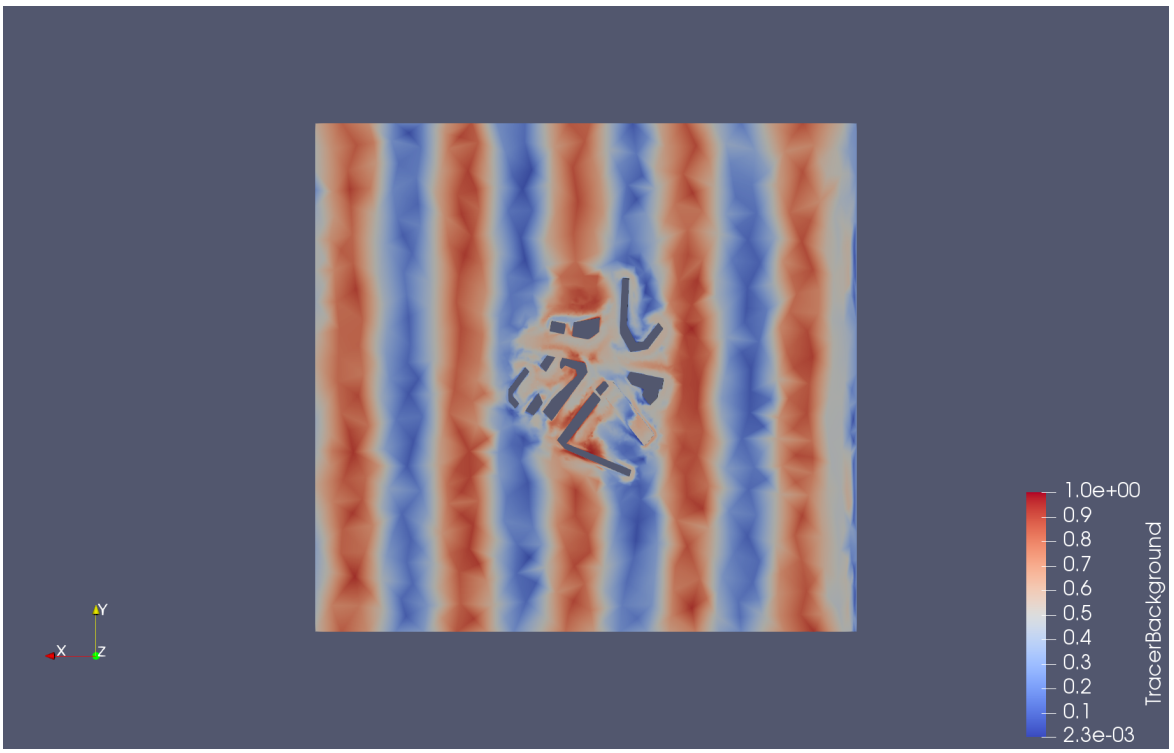

Figure 3.2: Cut of tracer background at t = 106

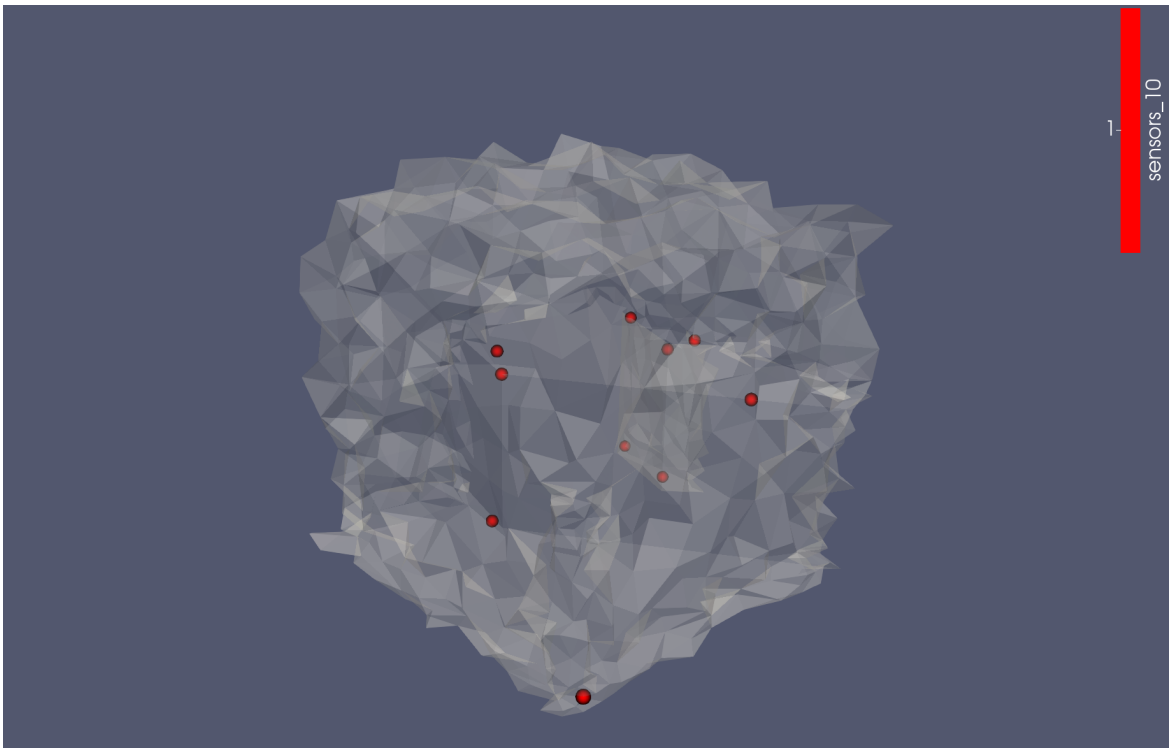

Figure 3.3: Position of the optimal set of 10 sensors

Chapter 4

Conclusion

In this report we have stated some of the research that was done in order to solve the sensor position optimisation problem.

The biggest challenge here is to make the algorithm scalable for optimising over $100'000$ sensor locations. It requires to develop p strategies to make the GPs scalable. The main other challenge is to find a method to accurately estimate the covariance matrix. The theory enabling this scalability still need to be developed

I have implemented some of the optimisation algorithm proposed on a smaller dataset and with a poor estimation of the covariance function. It has given consistent results and has proven that once the main challenges, described earlier are solved, we will have a good optimisation algorithm for that kind of setups.

Bibliography

Arcucci, R., Pain, C., and Guo, Y.-K. (2018). Effective variational data assimilation in air-pollution prediction. *Big Data Mining and Analytics*, 1(4):297–307. pages 3

Caselton, W. and Zidek, J. (1984). Optimal monitoring network designs. *Statistics & Probability Letters*, 2(4):223–227. pages 7

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. Wiley series in telecommunications. Wiley, New York. pages 6, 7

Cressie, N. A. C. (1991). *Statistics for spatial data*. Wiley series in probability and mathematical statistics. Wiley, New York. pages 3, 5

Deisenroth, M. P., Faisal, A., and Ong, C. S. (2018). Mathematics for Machine Learning. page 43. pages 4

Fan, J., Liao, Y., and Liu, H. (2015). An Overview on the Estimation of Large Covariance and Precision Matrices. *arXiv:1504.02995 [stat]*. arXiv: 1504.02995. pages 5, 6

Guttorp, P. and Sampson, P. D. (1994). 20 Methods for estimating heterogeneous spatial covariance functions with environmental applications. In *Handbook of Statistics*, volume 12, pages 661–689. Elsevier. pages 6

Krause, A., Singh, A., and Guestrin, C. (2008). Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. page 50. pages 5, 6, 7, 8, 9, 10

Liu, H., Ong, Y.-S., Shen, X., and Cai, J. (2018). When Gaussian Process Meets Big Data: A Review of Scalable GPs. *arXiv:1807.01065 [cs, stat]*. arXiv: 1807.01065. pages 4

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294. pages 8

Nott, D. J. and Dunsmuir, W. T. M. (2002). Estimation of nonstationary spatial covariance structure. *Biometrika*, 89(4):819–829. pages 5

Paciorek, C. J. and Schervish, M. J. (2004). Nonstationary Covariance Functions for Gaussian Process Regression. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pages 273–280. MIT Press. pages 5

Pourahmadi, M. (2011). Covariance Estimation: The GLM and Regularization Perspectives. *Statistical Science*, 26(3):369–387. arXiv: 1202.1661. pages 4

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge,

Mass. OCLC: ocm61285753. pages 3, 4, 5

Sampson, P. D. and Guttorp, P. (1992). Nonparametric Estimation of Nonstationary Spatial Covariance Structure. *Journal of the American Statistical Association*, 87(417):108–119. pages 5

Song, J., Fan, S., Lin, W., Mottet, L., Woodward, H., Davies Wykes, M., Arcucci, R., Xiao, D., Debay, J.-E., ApSimon, H., Aristodemou, E., Birch, D., Carpentieri, M., Fang, F., Herzog, M., Hunt, G. R., Jones, R. L., Pain, C., Pavlidis, D., Robins, A. G., Short, C. A., and Linden, P. F. (2018). Natural ventilation in cities: the implications of fluid mechanics. *Building Research & Information*, 46(8):809–828. pages 2