Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

TECHNISCHE UNIVERSITÄT WIEN

ASC TUWIEN
Institut für Analysis und Scientific Computing

Fast Algorithms for Iterative Bayesian PDE Inversion

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Mathematik

eingereicht von

Benjamin Stadlbauer
Matrikelnummer 1226175

Ausgeführt am Institut für Analysis und Scientific Computing
der Fakultät für Mathematik und Geoinformation
der Technischen Universität Wien

Betreuer
Associate Professor Dipl.-Ing. Dr.techn. Clemens Heitzinger

Wien, am 22. 10. 2018 Unterschrift des Studierenden

Abstract

In this article we present algorithms to perform Bayesian inversion based on physical models, in particular based on partial differential equations. We are interested in identifying parameters of the PDEs that affect functionals of the solutions for which experimental data are available. Markov-chain Monte-Carlo methods like the Metropolis algorithm provide the algorithmic foundation. We present an adaptation and extension of this procedure to be able to perform multi-dimensional Bayesian inversion where not all measurements have to be present prior to the estimation, but become available in batches as time passes. Namely, based on the Delayed-Rejection Adaptive-Metropolis (DRAM) algorithm, we introduce an iterative approach, where we use the posterior of the last Metropolis run as the prior for the new run, where we use new measurements in each iteration. This allows to examine some information about the parameters already during the estimation process. Therefore a density estimator needs to be introduced. We make use of the Improved Fast Gauss Transform (IFGT) which allows us to perform a faster evaulation of the kernel density estimator, reducing the runtime from quadratic to nearly linear. Applications using a nano-capacitor sensor array are presented as well, where we estimate the radii of over 4000 nano-electrodes.

1 Introduction

One of the most important tasks in mathematics is to identify accurate models for processes in physics, chemistry, economics, finance etc. With the aid of mathematical models we are able to describe and understand the behaviour of a system and to draw forecasts for our quantities of interest. Usually, the models contain unknown parameters which have to be identified.

Let us denote such a model by the function

$$\begin{aligned} Q: \quad & X \to Y, \\ & Q(\mathbf{q}) = \mathbf{y}, \end{aligned} \tag{1}$$

which takes an element of the parameter space X and maps it to the quantity of interest, which is an element of the measurement space Y. We call this function Q the *observation* function.

The forward problem is to calculate the outcome $Q(\mathbf{q})$. Conversely, we can consider the case that the outcome $\mathbf{y}$ is known, e.g. from experiments, and the parameter $\mathbf{q}$ needs to be determined. In this case, we aim to find the inverse

$$F: \quad Y \to X, \quad F := Q^{-1}$$

of the observation function Q, which maps the observation $\mathbf{y}$ to the parameter $\mathbf{q}$ such that $Q(\mathbf{q}) = \mathbf{y}$ holds. This is also called the *inverse problem*.

It is typical for inverse problems of models $Q(\mathbf{q}) = \mathbf{y}$ to be ill-posed, which means that for a given $\mathbf{y}$ there may be no unique solution $\mathbf{q}$ or the solution may be very sensitive to small changes of $\mathbf{y}$. To solve this problem we use the Bayesian approach to examine the parameters $\mathbf{q}$ for a given $\mathbf{y}$. This way we interpret the parameters as random variables whose probability distributions we will calculate. Knowing the distribution, we can draw conclusions about the mean value, confidence intervals, and also about the standard deviation, a feature which is inaccessible using e.g. the maximum likelihood approach.

In Section 2 we present the theoretical background of Bayesian inversion, namely the general, infinite-dimensional Bayesian estimation approach, and later on the finite-dimensional approach. We will introduce the algorithms to perform numerical calculations in Section 3. We will go into details of the classic Metropolis algorithm in Section 3.1, followed by Section 3.2 introducing the Delayed-Rejection Adaptive-Metropolis procedure (DRAM). In Section 3.3 we explain the iterative algorithm, whereby a kernel-density estimation is necessary, which we will discuss in Section 3.4. In the latter subsection we will also describe the implementation of the Improved Fast Gauss Transform (IFGT) in order to reduce the computation time of evaluating the density. We present an application of the before mentioned algorithms in Section 4 to a biosensor for which we have experimental data. Sensors are a good example, since it is always useful to extract information about the analyte. Conclusions are drawn in Section 5.

2 Infinite-dimensional Bayesian Estimation

We will present the theory which enables us to express the inverse problem within the framework of probability theory.

To begin with, we denote the *parameter space* by X and the *measurement space* by $Y := \mathbb{R}^m$, $m \in \mathbb{N}$. X can be an infinite-dimensional Hilbert space or a Banach space, often $X = \mathbb{R}^n$, $n \in \mathbb{N}$.

Following the setup from [1] we introduce the statistical model

$$\mathbf{y} = Q(\mathbf{q}) + \boldsymbol{\eta},$$

which is an adaptation of eq. (1), where $\boldsymbol{\eta}$ is additive noise, namely a random variable with zero mean, $\mathbf{q} \in X$ is the sought parameter, and $\mathbf{y} \in Y$ is an observation. In our applications $\boldsymbol{\eta}$ is a Gaussian random variable with zero mean.

Using Theorem 1 we are able to express probability measures with respect to other probability measures.

Theorem 1 (Radon-Nikodym derivative)**.** *Let μ and ν be two measures on the same measure space $(\Omega, \mathcal{F})$. If $\mu \ll \nu$ and ν is σ-finite, then there exists a ν-measurable function $f : \Omega \to [0, \infty]$ such that*

$$\mu(A) = \int_A f(\mathbf{x}) \mathrm{d}\nu(\mathbf{x})$$

for all ν-measurable sets $A \in \mathcal{F}$.

This function f is also known as the *Radon-Nikodym derivative* of μ with respect to ν. We can also write this derivative as

$$\frac{\mathrm{d}\mu}{\mathrm{d}\nu}(\mathbf{x}) = f(\mathbf{x}).$$

In the finite-dimensional case, if we have a probability measure μ which is absolutely continuous with respect to the Lebesgue measure λ, we can express the classical probability density function in terms of the Radon-Nikodym derivative $\mathrm{d}\mu/\mathrm{d}\lambda(\mathbf{x}) = f(\mathbf{x})$. In this case, for a $A \in \mathcal{F}$, we can rewrite

$$\mu(A) = \int_A \frac{\mathrm{d}\mu}{\mathrm{d}\lambda} \mathrm{d}\mathbf{x},$$

where we have just used Theorem 1.

However, in infinite dimensions, the Lebesgue measure does not exist. Therefore, if we want to express the analogon to the PDF for a probability measure μ living on a Banach space X, we need a reference measure, e.g. a Gaussian measure μ_0, which is defined on a Banach space. The Radon-Nikodym derivative can be expressed as

$$\rho_g(\mathbf{x}) := \frac{\mathrm{d}\mu}{\mathrm{d}\mu_0}(\mathbf{x})$$

such that

$$\mu(A) = \int_A \rho_g(\mathbf{x})\mu_0(\mathrm{d}\mathbf{x})$$

holds for any $A \in \mathcal{F}$.

Now, we can state Bayes' Theorem for the infinite-dimensional case, namely on a separable Banach space $(X, \|\cdot\|)$. Let $\mathbf{q}$ be a random variable distributed according to the measure μ_0. We can observe the outcome using the statistical model

$$Q(\mathbf{q}) + \boldsymbol{\eta} = \mathbf{y},$$

where $\mathbf{y} \in \mathbb{R}^m$ is independent of $\mathbf{q}$ and has the density ρ with respect to the Lebesgue measure, and $\boldsymbol{\eta}$ is a Gaussian random variable $\mathcal{N}(0, \Gamma)$. We define the potential

$$\Phi(\mathbf{q}, \mathbf{y}) := -\log(\rho(\mathbf{y} - Q(\mathbf{q}))),$$

such that we can write the density $\rho(\mathbf{y} - Q(\mathbf{q}))$ in the form of an exponential as

$$\rho(\mathbf{y} - Q(\mathbf{q})) = \exp(-\Phi(\mathbf{q}, \mathbf{y})).$$

To ensure that for a given observation function Q and a prior μ_0 the posterior μ^y is well-defined, we asume that Q satisfies the following assumption of continuity and boundedness [1]. The assumption should be checked for the model equation in each application.

Assumption 1. The function $Q : X \to Y$ satisfies these two conditions:

1. For every $\varepsilon > 0$ there is an $M = M(\varepsilon) \in \mathbb{R}$ such that, for all $\mathbf{q} \in X$, the inequality

   $$\left|\Gamma^{-1/2} Q(\mathbf{q})\right| \leq \exp(\varepsilon \|\mathbf{q}\|_X^2 + M)$$

   holds.

2. For every $r > 0$ there is a $K = K(r) > 0$ such that, for all $\mathbf{q}_1, \mathbf{q}_2 \in X$ with $\max\left(\|\mathbf{q}_1\|_x, \|\mathbf{q}_2\|_X\right) < r$, the inequality

   $$\left|\Gamma^{-1/2}(Q(\mathbf{q}_1) - Q(\mathbf{q}_2))\right| \leq K \|\mathbf{q}_1 - \mathbf{q}_2\|_X$$

   holds.

Theorem 2. *Let Q satisfy Assumption 1 and assume μ_0 is a Gaussian measure satisfying $\mu_0(X) = 1$. Then μ^y is a well-defined probability measure and satisfies*

$$\frac{\mathrm{d}\mu^y}{\mathrm{d}\mu_0}(\mathbf{q}) = \frac{1}{Z(\mathbf{y})} \exp(-\Phi(\mathbf{q}, \mathbf{y})). \tag{2}$$

In eq. (2) the scalar $Z(\mathbf{y})$ is a normalizing constant.

Next, we state two results for a linear elliptic problem and for a nonlinear elliptic problem, showing that Assumption 1 is satisfied.

Theorem 3 ([2]). *We consider the elliptic boundary-value problem*

$$\begin{aligned} -\nabla \cdot (A \nabla u) &= f && \text{on } D, \\ u &= g && \text{on } \partial D_D, \\ \frac{\partial u}{\partial \mathbf{n}} &= 0 && \text{on } \partial D_N, \end{aligned}$$

where D is bounded and has a smooth boundary ∂D. Now let Q be the function which maps the parameter $\mathbf{q} := \log(A)$ to the unique solution $Q(\mathbf{q})$. Then Q satisfies Assumption 1 and therefore the according posterior distribution is well-defined.

In [2], where Theorem 3 is cited from, a Bayesian analysis has been performed to investigate a tomography problem based on the Poisson-Boltzmann equation. The proof can also be found there.

Theorem 4 ([3]). *We consider the nonlinear elliptic boundary-value problem*

$$\begin{aligned} -\nabla \cdot (A \nabla u) + \sinh(u) &= f && \text{on } D, \\ u &= g && \text{on } \partial D. \end{aligned}$$

Let Q again be the operator which maps the parameter $\mathbf{q} := \log(A)$ to the solution $Q(\mathbf{q})$. Then Q fulfills the Assumption 1 and therefore the according posterior is well-defined.

The proof can be found in [3].

From now on we assume that the parameter space $X = \mathbb{R}^n$ is finite-dimensional.

For the Bayesian setup we assume the parameter $\mathbf{q}$ and the observation $\mathbf{y}$ to be jointly distributed random variables $(\mathbf{q}, \mathbf{y}) \in X \times Y$ and that there exists a joint density $\pi(\mathbf{q}, \mathbf{y})$. We call the marginal density $\pi(\mathbf{q}) =: \pi_0(\mathbf{q})$ the *prior* density and the conditional density $\pi(\mathbf{y}|\mathbf{q})$ the *likelihood* function.

Theorem 5. *Let $f_{R,S}$ be the probability density function of a continuous random vector $(R, S) : \Omega \to \mathbb{R}^{n+m}$ with $R : \Omega \to \mathbb{R}^n$ and $S : \Omega \to \mathbb{R}^m$. Let us denote the marginal densities of the random vectors R and S as $f_R : \mathbb{R}^n \to \mathbb{R}$ and $f_S : \mathbb{R}^m \to \mathbb{R}$, respectively. Then for every $\mathbf{y} \in \mathbb{R}^m$ with $f_S(\mathbf{y}) \neq 0$, the equation*

$$f_{R|S}(\mathbf{x}|\mathbf{y}) = \frac{f_{S|R}(\mathbf{y}|\mathbf{x}) f_R(\mathbf{x})}{\int_{R^n} f_{S|R}(\mathbf{y}|\boldsymbol{\xi}) f_R(\boldsymbol{\xi})\, d\boldsymbol{\xi}} \tag{3}$$

holds. Here $f_{R|S}$ and $f_{S|R}$ are the conditional probability density functions.

Using Theorem 5, we can express the *posterior* density as

$$\pi(\mathbf{q}|\mathbf{y}) = \frac{\pi(\mathbf{y}|\mathbf{q}) \pi_0(\mathbf{q})}{\int_{R^n} \pi(\mathbf{y}|\xi) \pi_0(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi}} \tag{4}$$

or

$$\pi(\mathbf{q}|\mathbf{y}) \propto \pi(\mathbf{y}|\mathbf{q}) \pi_0(\mathbf{q}). \tag{5}$$

In many applications the prior and the likelihood are available. The prior distribution $\pi_0(\mathbf{q})$ plays the role of the initial believe in our parameter, e.g. if we already have some knowledge about the approximate value of our parameter. The likelihood function $\pi(\mathbf{q}|\mathbf{y})$ quantifies the probability of a parameter to yield the outcome $\mathbf{y}$.

3 Algorithms

For numerical applications we assume that the observation function Q, the observational noise $\boldsymbol{\eta}$, the observation $\mathbf{y}$, and the prior distribution π_0 are known. Using Theorem 5 we can express the posterior distribution in terms of the prior and likelihood, see eq. (4).

Now, the biggest numerical challenge would be to calculate the normalizing factor, the integral in eq. (4). Depending on the dimensionality of the parameter space X, this can be a very computationally expensive task to perform. In low dimensions one can use quadrature rules to evaluate the integral. As soon as the dimensionality is moderate, sparse-grid quadrature techniques can be applied. In high dimensions, Monte-Carlo methods are required [4]. This approach is still difficult because of the sheer fact that the support of the posterior is one aspect one does not know beforehand. The Metropolis algorithm remedies this problem.

To simplify the notation we define

$$\|\mathbf{x}\|_{\mathbf{A}}^2 := \mathbf{x}^\top \mathbf{A}\mathbf{x}.$$

for $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$.

3.1 Classic Metropolis algorithm

For implementations of Bayesian estimation, the basic algorithm is the Metropolis algorithm [4, 5]. This algorithm is a Markov-chain Monte-Carlo samling method, which constructs a sequence of realizations of random variables $(X_i)_{i\in\mathbb{N}}$, whereby the properties of X_i only depend on the previous random variable X_{i-1}, i.e.,

$$\mathbb{P}(X_i = k|X_{i-1} = k_{i-1}, X_{i-2} = k_{i-2}, \ldots, X_0 = k_0) = \mathbb{P}(X_i = k|X_{i-1} = k_{i-1}).$$

Once the Markov-chain has passed the *burn-in* period, i.e., the distribution of X_i is stationary and does not depend on i anymore, it should provide us with samples of the posterior distribution. This method is more suitable, especially in high dimensions, because one does not have to calculate the integral in eq. (4). For this procedure we start with the prior distribution $\pi_0(\mathbf{q})$, the likelihood-function $\pi(\mathbf{y}|\mathbf{q})$, and an initial educated guess $\mathbf{q}_0 \in \mathbb{R}^n$ for our Markov chain. This first value could also be the solution of the maximum-likelihood optimization problem [4]. Then we apply the Metropolis Algorithm 1.

If a symmetrical proposal function $J(\mathbf{q}^*|\mathbf{q}_{k-1}) = J(\mathbf{q}_{k-1}|\mathbf{q}^*)$ is chosen, e.g. a Gaussian distri-

Algorithm 1: Metropolis

Data: Initial value $\mathbf{q}_0$, proposal function J, prior $\pi_0(\mathbf{q})$, likelihood $\pi(\mathbf{y}|\mathbf{q})$, number of samples N, measurement $\mathbf{y}$

Result: Samples $(\mathbf{q}_i)_{i=0,\dots,N}$

```
for k = 1 to N do
    r ← Draw [U(0,1)];
    q* ← Draw J(·|q_{k-1});
    α ← π_0(q*)π(y|q*) / π_0(q_{k-1})π(y|q_{k-1});
    if α > r then
        q_k := q*;
    else
        q_k := q_{k-1}.
```

bution

$$J(\cdot|\mathbf{q}_{k-1}) \sim \mathcal{N}(\mathbf{q}_{k-1}|\mathbf{V}),$$

where $\mathbf{V}$ is the positive definite covariance matrix, then the choice

$$\alpha(\mathbf{q}^*|\mathbf{q}_{k-1}) := \frac{\pi(\mathbf{q}^*|\mathbf{y})}{\pi(\mathbf{q}_{k-1}|\mathbf{y})} = \frac{\pi_0(\mathbf{q}^*)\pi(\mathbf{y}|\mathbf{q}^*)}{\pi_0(\mathbf{q}_{k-1})\pi(\mathbf{y}|\mathbf{q}_{k-1})} \tag{6}$$

ensures that the stationary distribution of the constructed Markov chain is the sought posterior distribution [4].

To fully describe the algorithm, we assume that there are N_{obs} measurements $(\mathbf{y}_i)_{i=1,\dots,N_{\text{obs}}} \in \mathbb{R}^m$ given. The likelihood function $L(\mathbf{q}, \mathbf{y}) := \pi(\mathbf{y}|\mathbf{q})$ is a product of N_{obs} Gaussian densities, i.e.,

$$\begin{aligned} L(\mathbf{q}, \mathbf{y}) : &= \frac{1}{(2\pi)^{mN_{\text{obs}}/2} \det(\mathbf{\Sigma})^{N_{\text{obs}}/2}} e^{-S}, \\ S &:= \frac{1}{2} \sum_{i=1}^{N_{\text{obs}}} \|\mathbf{y}_i - Q(\mathbf{q})\|^2_{\mathbf{\Sigma}^{-1}}, \end{aligned} \tag{7}$$

where $\mathbf{\Sigma} \in \mathbb{R}^{m \times m}$ is the positive definite covariance matrix of the Gaussian density and S is the sum of squared errors. As described before, the likelihood quantifies the probability of having a parameter which results in an outcome which is close to the measurements. As we can see, the $L(\mathbf{q}, \mathbf{y})$ gets larger as the sum S of squared errors gets smaller, which happens iff $Q(\mathbf{q})$ coincides more with the measurements $\mathbf{y}_i$.

Inserting eq. (7) into the acceptance ratio eq. (6) gives

$$\alpha(\mathbf{q}^*|\mathbf{q}_{k-1}) = \frac{\pi_0(\mathbf{q}^*)\pi(\mathbf{y}|\mathbf{q}^*)}{\pi_0(\mathbf{q}_{k-1})\pi(\mathbf{y}|\mathbf{q}_{k-1})} = \frac{\pi_0(\mathbf{q}^*)}{\pi_0(\mathbf{q}_{k-1})} e^{-(S^* - S_{k-1})}.$$

To decide if a proposed value $\mathbf{q}^*$ is going to be accepted, we draw a uniformly distributed random number $r \sim \mathcal{U}(0,1)$. If $\alpha(\mathbf{q}^*|\mathbf{q}_{k-1}) > r$, the parameter $\mathbf{q}^*$ is accepted and $\mathbf{q}_k := \mathbf{q}^*$, if

not, the parameter is rejected and the old parameter $\mathbf{q}_{k-1}$ will be the realization of the Markov chain step, i.e., $\mathbf{q}_k := \mathbf{q}_{k-1}$.

3.2 DRAM

In the numerical examples, we use Algorithm 2, which contains two major improvements compared to the basic algorithm. It is called Delayed-Rejection Adaptive-Metropolis algorithm (DRAM) [4, 6].

Algorithm 2: DRAM

Data: Observation function Q, measurements $(\mathbf{y}_i)_{i=1,\dots,N_{\text{obs}}}$, number of samples N, first value $\mathbf{q}_0$, covariance matrix of proposal $\mathbf{V}$, covariance matrix of likelihood $\mathbf{\Sigma}$, prior π_0, factor γ^2, value s, update matrix every k_0 steps

Result: $(\mathbf{q}_i)_{i=0,\dots,N}$

1 $s_m \leftarrow s/m$;
2 **for** $k \leftarrow 1$ **to** N **do**
3 $\quad r \leftarrow$**Draw** $[\mathcal{U}(0,1)]$;
4 $\quad \mathbf{q}^* \leftarrow$**Draw** $[\mathcal{N}(\mathbf{q}_{k-1}, \mathbf{V})]$;
5 $\quad S^* \leftarrow 1/2 \sum_{i=1}^{N_{\text{obs}}} \|\mathbf{y}_i - Q(\mathbf{q}^*)\|^2_{\Sigma^{-1}}$;
6 $\quad \alpha \leftarrow \min(1, \pi_0(\mathbf{q}^*)/\pi_0(\mathbf{q}_{k-1}) \exp(-(S^* - S)))$;
7 $\quad$ **if** $r < \alpha$ **then**
8 $\quad\quad \mathbf{q}_k \leftarrow \mathbf{q}^*$;
9 $\quad\quad S \leftarrow S^*$;
10 $\quad$ **else**
11 $\quad\quad r \leftarrow$**Draw** $[\mathcal{U}(0,1)]$;
12 $\quad\quad \mathbf{q}^{*2} \leftarrow$**Draw** $\left[\mathcal{N}(\mathbf{q}_{k-1}, \gamma^2 V)\right]$;
13 $\quad\quad S^{*2} \leftarrow 1/2 \sum_{i=1}^{N_{\text{obs}}} \left\|\mathbf{y}_i - Q(\mathbf{q}^{*2})\right\|^2_{\mathbf{\Sigma}^{-1}}$;
14 $\quad\quad \alpha_2 \leftarrow$ Expression eq. (8);
15 $\quad\quad$ **if** $r < \alpha_2$ **then**
16 $\quad\quad\quad \mathbf{q}_k \leftarrow \mathbf{q}^{*2}$;
17 $\quad\quad\quad S \leftarrow S^{*2}$;
18 $\quad\quad$ **else**
19 $\quad\quad\quad \mathbf{q}_k \leftarrow \mathbf{q}_{k-1}$;
20 $\quad \overline{\mathbf{q}}^* \leftarrow \mathbf{q}^* + (k-1)/k \cdot (\overline{\mathbf{q}} - \mathbf{q}^*)$;
21 $\quad \widetilde{\mathbf{V}} \leftarrow (k-1)/k \cdot \widetilde{\mathbf{V}} + s_m/k \cdot \left((k-1)\|\overline{\mathbf{q}}\|^2 - k\|\overline{\mathbf{q}}^*\|^2 + \|\mathbf{q}^*\|^2\right)$;
22 $\quad$ **if** $mod(k, k_0) = 0$ **and** $k > k_0$ **then**
23 $\quad\quad \mathbf{V} \leftarrow \widetilde{\mathbf{V}}$;
24 $\quad \overline{\mathbf{q}} \leftarrow \overline{\mathbf{q}}^*$.

In the outer for-loop a new parameter $\mathbf{q}^*$ is proposed in line 4, followed by the calculation of the acceptance ratio α together with the sum S^* of squared errors. In the standard Metropolis-algorithm, if $\alpha > r$, whereby r is uniformly distributed random number, the new parameter is accepted and rejected otherwise.

However, this version also allows us to prevent rejections due to the adaptation added in lines

11–17. Whenever $\mathbf{q}^*$ is rejected, a new value $\mathbf{q}^{*2}$ is proposed from a normal distribution with covariance matrix $\gamma^2\mathbf{V}$, where $\mathbf{V}$ is the covariance matrix of the proposal distribution and γ^2 is a positive value smaller than one. Here we use $\gamma^2 := 0.2$ [4]. Then the new acceptance ratio α_2 is calculated as

$$\begin{aligned}\alpha_2 := \min\Bigg\{1, \frac{\pi_0\left(\mathbf{q}^{*2}\right)}{\pi_0\left(\mathbf{q}_{k-1}\right)}\cdot \\ \cdot\exp\left[-\left(S^{*2}-S\right)-\frac{1}{2}\left(\left\|\mathbf{q}^*-\mathbf{q}^{*2}\right\|^2_{\mathbf{V}^{-1}}-\left\|\mathbf{q}^*-\mathbf{q}_{k-1}\right\|^2_{\mathbf{V}^{-1}}\right)\right]\cdot \\ \cdot\frac{1}{1-\alpha}\left[1-\min\left(1,\frac{\pi_0(\mathbf{q}^*)}{\pi_0(\mathbf{q}^{*2})}\exp\left(-\left(S^*-S^{*2}\right)\right)\right)\right]\Bigg\},\end{aligned} \tag{8}$$

and $\mathbf{q}^{*2}$ is accepted with probability α_2. So after a first rejection, an alternative parameter $\mathbf{q}^{*2}$ "closer" to the previous parameter $\mathbf{q}_{k-1}$ is proposed, hence we have a *delayed-rejection*. Since the previous parameter was accepted, the alternative one has a higher chance of being accepted too.

If the components of the posterior are strongly correlated in an anisotropic way, it is not ideal to use a fixed covariance matrix for the proposal distribution. The *adaptive* contribution of the algorithm can change the covariance matrix during the random walk. This way more proposed points are accepted [6]. Another advantage of the adaptive covariance matrix procedure is that the sampling algorithm is more robust than a fixed covariance matrix for the proposal distribution in terms of the amount of rejected parameters. In lines 20–24 the procedure tracks the distribution of the $\mathbf{q}_k$ and updates the covariance matrix $\widetilde{\mathbf{V}}$ recursively to prevent calculating the whole covariance matrix in every step. Then every k_0 steps the matrix $\mathbf{V}$ is updated in line 23 and the proposal function changes its distribution accordingly in line 4. The parameter s_p is based on the dimension of the observation space $\mathbb{R}^m$, see line 1.

In Figure 1 the advantages of these improvements to the classic Metropolis algorithm are illustrated. Four different algorithms have been used to determine two parameters in an ideal setup. The one-dimensional, transient heat equation with a delta distribution δ_y as initial data and a constant thermal diffusivity constant A forms the setup. The task is to estimate the two parameters A and y. To emphasize the advantages of the DRAM algorithm, we performed (a) the classic Metropolis algorithm, (b) an adaptation where we implemented the adaptiveness of the proposal function, (c) a further adaption where we implemented the delayed-rejection procedure, and (d) the DRAM algorithm, which combines the two latter ones. We plot the error in dependence of the required number of samples in Figure 1. One can see that the adaptive implementation results in a significant speed up in convergence. However, the delayed-rejection implementation does not necessarily help in this regard. But it smoothes the density estimate, since long stagnations in the paths are prevented, see Figure 2.

3.3 The Iterative Metropolis Algorithm

It may be the case that not all of the N_{obs} measurements $(\mathbf{y}_i)_{i=1,\dots,N_{\text{obs}}} =: M$ are available from the very beginning. Maybe only a sublist $M_1 \subset M$ with $|M_1| = N^1_{\text{obs}} < N_{\text{obs}} = |M|$

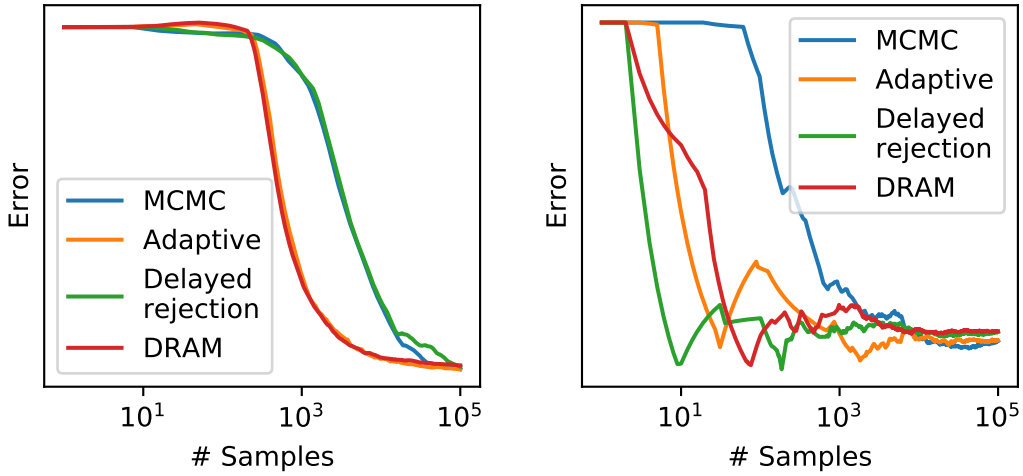


Figure 1: The error is the distance of the true value from the mean value, based on the first n samples. Left: Here we choose a proposal distribution, where the two-dimensional covariance matrix was rotated by 90° of the estimated variance matrix of the posterior and the starting value was far from the true value. One can see that the adaptive algorithm, as well as the DRAM, updated this matrix during the random walk to eventually lead to faster convergence. Right: Here the covariance matrix of the proposal distribution has been scaled by a factor of 10, which leads to a faster convergence of the delayed-rejection algorithm as well as for the DRAM compared to the classic MCMC.

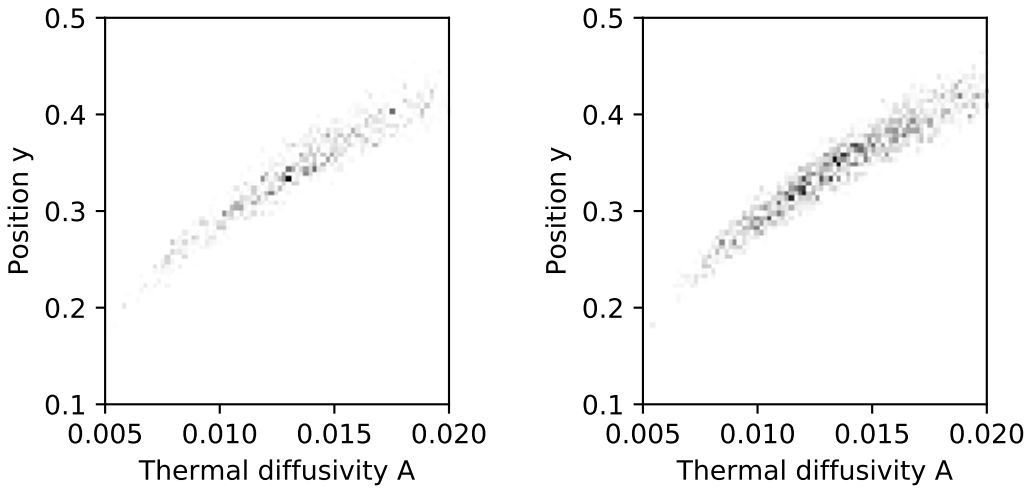


Figure 2: Two-dimensional histograms of samples, calculated using either the classic MCMC algorithm (left) or with the Metropolis algorithm which has the delayed-rejection procedure included (right). The setup is exactly the same, including the starting point, the proposal distribution and the number of samples. However, the delayed-rejection results in a much smoother posterior density estimation, which here is the advantage of this adaptation. Since this result is based on visualization we should note that for both of these color plots the histograms were normalized and the same color map was used.

can be used at the beginning, while a new list M_2 of measurements becomes available during the estimation process. Since the first N_{obs}^1 observations are taken into account in form of a likelihood function $L_1(\mathbf{q}, \mathbf{y})$, we obtain the posterior distribution

$$\pi_1(\mathbf{q}|\mathbf{y}) \propto L_1(\mathbf{q}, \mathbf{y})\pi_0(\mathbf{q}),$$

see eq. (5). Whenever the new list M_2 of observations is available, we can introduce the corresponding likelihood $L_2(\mathbf{q}, \mathbf{y})$. Since we already have some information about the parameter $\mathbf{q}$ in form of the first posterior π_1 we take this density into account as our prior density, perform a Bayesian estimation step, and find the second posterior

$$\begin{aligned}\pi_2(\mathbf{q}|\mathbf{y}) &\propto L_2(\mathbf{q}, \mathbf{y})\pi_1(\mathbf{q}|\mathbf{y}) \\ &\propto \underbrace{L_1(\mathbf{q}, \mathbf{y})L_2(\mathbf{q}, \mathbf{y})}_{\text{contains } M_1 \text{ and } M_2} \pi_0(\mathbf{q}),\end{aligned}$$

which includes already the first two lists of measurements.

One can *iterate* over the lists of measurements M_i until all observations $\bigcup_{i=1}^K M_i = M$ have been taken into account. The corresponding posterior density then looks like

$$\pi_K(\mathbf{q}|\mathbf{y}) \propto \prod_i^K L_i(\mathbf{q}, \mathbf{y})\pi_0(\mathbf{q}),$$

where the product of the likelihoods contains all measurements M. One should keep in mind that the very same posterior density results if we consider just one likelihood which contains all observations

$$L(\mathbf{q}|\mathbf{y}) = \prod_{i=1}^K L_i(\mathbf{q}|\mathbf{y}),$$

since the likelihood function is simply a product of Gaussians for every observation.

One important example for the application of this iterative procedure are sensors in general. Here we consider the timely application of nanoelectrode sensors in detail [7]. During data acquisition measurements are performed at several frequencies. The multiple lists of measurements correspond to the different frequencies. So with the iterative Metropolis algorithm we can determine the parameters, even though we do not have to consider all frequencies at once. We will use one after another, until we have included all frequencies to eventually get to the multi-frequency case.

In order to implement this procedure using the DRAM algorithm, the program needs to evaluate the density of the prior distribution, see line 6 in Algorithm 2. The very first prior distribution π_0 is given in closed form, but the latter densities $\pi_1, \pi_2, \ldots$, which then play the role of the prior, are only given in the form of finite amounts of samples. So we need a way to evaluate posterior densities – a density estimator.

3.4 Kernel Density Estimator

There exist different methods, e.g. histograms or kernel density estimators, to obtain a density estimator.

Histograms are easy to handle and well understood in low dimensions. However, many reasons speak against the use of histograms in our applications, since we also want to do Bayesian estimation in higher dimensions [8]. One of them is the difficulty to partition samples into regular bins [9].

In contrast, the kernel density estimator is quite robust in high dimensions, since one does not need to assign the source points to bins in advance and the evaluation of the kernel functions are not very expensive. Furthermore, the density estimations are very smooth, if we choose a smooth kernel. However, a bandwidth matrix has to be introduced, which is a crucial part [10].

Over the iterations of the various lists of measurements $M_1, M_2, \dots$ there is no reason to pick different numbers of samples for the posteriors, i.e., we can evaluate the kernel density estimator at N *target* points $(\mathbf{p}_j)_{j=1,\dots,N}$ based on N sample or *source* points $(\mathbf{x}_i)_{i=1,\dots,N}$ for every Metropolis iteration, whereby the $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{p}_j \in \mathbb{R}^m$. Let $\mathcal{I} := \{1, \dots, N\}$ be the index set of target points and source points, respectively. The density estimator based on the source points $(\mathbf{x}_i)_{i\in\mathcal{I}}$ and the bandwidth matrix $\mathbf{H}$ evaluated at the target point $\mathbf{p}_j$ is

$$\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{p}_j) := \frac{1}{N} \sum_{i\in\mathcal{I}} K_{\mathbf{H}}(\mathbf{p}_j - \mathbf{x}_i), \tag{9}$$

where

$$K_{\mathbf{H}}(\mathbf{x}) := \frac{1}{\sqrt{\det(2\pi\mathbf{H})}} \exp\left(-\frac{1}{2} \|\mathbf{x}\|^2_{\mathbf{H}^{-1}}\right) \tag{10}$$

is the Gaussian kernel function with the positive definite bandwidth matrix $\mathbf{H}$.

The computational runtime for one Metropolis iteration behaves like $\mathcal{O}(N^2)$, since for every target point $\mathbf{p}_j$ we have to calculate N evaluations $K_{\mathbf{H}}(\mathbf{p}_j - \mathbf{x}_i)$, $i = 1, \dots, N$, and there are N target points. Because we want to work with sample sizes of at least 10^4 to 10^6 we want to reduce the asymptotic runtime. Using the Improved Fast Gauss Transform [9] together with a greedy clustering algorithm we can achieve a computational runtime which behaves like $\mathcal{O}(N)$. We will introduce this procedure in detail.

Improved Fast Gauss Transform

The following algorithm is based on the *Fast Gauss Transform*. We will follow the procedure already established in [9].

The basic idea is to do something similar as the *Fast Multipole Method*. We want to partition our source points $\mathbf{x}_i$ into K clusters with cluster centers $\mathbf{c}_k$ such that for an evaluation of the density estimator $f_{\mathbf{H}}$ at a target point $\mathbf{p}_j$ it is only necessary to calculate the distances to these cluster centers rather than to all N source points. We also use a polynomial approximation of

our density estimation. All the information of the influence of the source points to the cluster centers will be stored in these coefficients.

For the evaluation of the kernel density estimator we will only use the Gaussian kernel function eq. (10). We assume our bandwidth matrix to be a positive multiple of the identity matrix $\mathbf{H} = (h^2/2)\mathbf{I} \in \mathbb{R}^{n\times n}$. We can thus rewrite eq. (9) as

$$\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{p}_j) = A(h)\sum_{i\in\mathcal{I}} e^{-\left\|\mathbf{p}_j-\mathbf{x}_i\right\|^2/h^2}, \tag{11}$$

where

$$A(h) := N^{-1}(h^2\pi)^{-n/2}. \tag{12}$$

The key is to partition all the source points $\mathbf{x}_i$ into K clusters with centers $\mathbf{c}_k$. Let us focus on one cluster only with its cluster center $\mathbf{c}$ and all the $\mathbf{x}_i$ are assigned to this one cluster. We can expand the term $-\left\|\mathbf{p}_j-\mathbf{x}_i\right\|^2/h^2$ such that

$$e^{-\left\|\mathbf{p}_j-\mathbf{x}_i\right\|^2/h^2} = e^{-\left\|\Delta\mathbf{p}_j\right\|^2}e^{-\|\Delta\mathbf{x}_i\|^2}e^{2\Delta\mathbf{p}_j\cdot\Delta\mathbf{x}_i/h^2},$$

where $\Delta\mathbf{p}_j = \mathbf{p}_j - \mathbf{c}$ and $\Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{c}$. Now, the term $e^{-\left\|\Delta\mathbf{p}_j\right\|^2/h^2}$ has to be calculated for every target point $\mathbf{p}_j$. Since the target points are not known until the very application of the algorithm there is no room for improvements or shortcuts for these computations. Similar for the second one, since $e^{-\|\Delta\mathbf{x}_i\|^2/h^2}$ only concerns the source points and this can be calculated prior of doing any calculations with the target points. In the third term, however, $e^{-2\Delta\mathbf{p}_j\Delta\mathbf{x}_i/h^2}$ the source points and the target points are still entangled. So here we have some potential of reducing the necessary steps. To break this entanglement, we introduce the series expansion

$$e^{2\Delta\mathbf{p}_i\Delta\mathbf{x}_i/h^2} = \sum_{\alpha\geq 0}^{\infty}\Phi_\alpha(\Delta\mathbf{p}_j)\Psi_\alpha(\Delta\mathbf{x}_i) \tag{13}$$

of this term, where the Φ_α and Ψ_α are the expansion functions

$$\Phi_\alpha(\Delta\mathbf{p}_j) = \left(\frac{\Delta\mathbf{p}_j}{h}\right)^\alpha, \quad \Psi_\alpha(\Delta\mathbf{x}_i) = \left(\frac{\Delta\mathbf{x}_i}{h}\right)^\alpha\frac{2^{|\alpha|}}{\alpha!}$$

with the usual operations

$$|\alpha| = \sum_{\ell=1}^{n}\alpha_\ell, \quad \alpha! = \prod_{\ell=1}^{n}\alpha_\ell!, \quad \mathbf{x}^\alpha = \prod_{\ell=1}^{n}x_\ell^{\alpha_\ell}$$

for multiindices $\alpha = (\alpha_1,\ldots,\alpha_n)$, where $\alpha_\ell \in \mathbb{N}$, $\ell = 1,\ldots,n$ and $\mathbf{x}\in\mathbb{R}^n$. If we truncate the series expansion in eq. (13) after multiindices with order greater than $p+1$ we can rewrite our

density estimator as

$$G(\mathbf{p}_j) := e^{-\left\|\mathbf{p}_j-\mathbf{c}\right\|^2/h^2} \sum_{|\alpha|\le p} C_\alpha \left(\frac{\mathbf{p}_j-\mathbf{c}}{h}\right)^\alpha,$$
$$C_\alpha = \frac{2^{|\alpha|}}{\alpha!} A(h) \sum_{i=1}^{N} e^{-\|\mathbf{x}_i-\mathbf{c}\|^2/h^2} \left(\frac{\mathbf{x}_i-\mathbf{c}}{h}\right)^\alpha, \tag{14}$$

where the coefficients C_α can be calculated prior to evaluating at target points $\mathbf{p}_j$. Now, our spatial data can be devided into several clusters. This can be accomplished using the *farthest-point clustering algorithm* [11]. This is a greedy algorithm subdividing the source points $\mathbf{x}_i$ into K clusters $(S_k)_{k=1,\dots,K}$ such that the maximum radius to the cluster center does not exceed a given threshold r_x, i.e.,

$$\max_{k=1,\dots,K} \max_{\mathbf{x}_i \in S_k} \|\mathbf{x}_i - \mathbf{c}_k\| \le r_x.$$

In Algorithm 3 at lines 2–4 the first cluster consists of all source points with the center $\mathbf{c}_1 = \mathbf{x}_1$.

Algorithm 3: Clustering

Data: Source points $(\mathbf{x}_i)_{i\in\mathcal{I}}$, maximum cluster radius r_x
Result: $(S_k, \mathbf{c}_k)_{k=1,\dots,K}$

```
1  K ← 1;
2  c_1 ← x_1;
3  S_1 ← {x_1, ..., x_N};
4  ℓ_i ← 1, i ∈ I;
5  while max_{i∈I} ‖x_i − c_{ℓ_i}‖ > r_x do
6      c_{K+1} ← source point with largest distance to its current cluster center;
7      K ← K + 1;
8      S_K ← {c_K};
9      for i ∈ I do
10         ℓ_i ← arg min_{k=1,...,K} ‖x_i − c_k‖;
11         assign x_i to cluster S_{ℓ_i}.
```

As long as some of the radii of the clusters are exceeding the threshold r_x a new cluster is going to be created as follows: The center of the new cluster is the source point with the largest distance to its current cluster center, line 6. Afterwards in lines 10–11, all source points are assigned to the cluster such that the distance to its center is minimized. This algorithm terminates because with an increasing number of clusters, the maximum distances to the cluster centers is decreasing.

We can now summarize the algorithm to evaluate the approximation of the density estimate $\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}$. In Algorithm 4 the clusters are calculated with the source points at line 1. Afterwards the coefficients are calculated in line 5. Finally, for every new target point $\mathbf{p}_i$ the density results in a sum over all the clusters which centers are within the cut-off radius r_y, line 7. Since we are using an approximation of our kernel density estimator we have to ensure that the

Algorithm 4: Improved Fast Gauss Transform

Data: Source points $(\mathbf{x}_i)_{i\in\mathcal{I}}$, maximum cluster radius r_x, cut-off radius r_y

Result: an approximate of the density $\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}$ for given target points $\mathbf{p}_i$

Subdivide source points $(\mathbf{x}_i)_{i\in\mathcal{I}}$ into clusters $(S_k, \mathbf{c}_k)_{k=1,\dots,K}$ using Algorithm 3;
choose p sufficiently large, see error estimate eq. (15);
for $k = 1, \dots, K$ **do**
    for $|\alpha| \leq p$ **do**
        $C_\alpha^k = \frac{2^{|\alpha|}}{\alpha!} A(h) \sum_{\mathbf{x}_i \in S_k} e^{-\|\mathbf{x}_i - \mathbf{c}_k\|^2/h^2} \left(\frac{\mathbf{x}_i - \mathbf{c}_k}{h}\right)^\alpha$;
for *target points* $\mathbf{p}_i$ **do**
    $\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{p}_j) \approx \sum_{\|\mathbf{p}_j - \mathbf{c}_k\| \leq r_y} e^{-\|\mathbf{p}_j - \mathbf{c}_k\|^2/h^2} \sum_{|\alpha|\leq p} C_\alpha \left(\frac{\mathbf{p}_j - \mathbf{c}_k}{h}\right)^\alpha$.

bound in the error estimate

$$\left|\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{p}) - G(\mathbf{p})\right| \leq A(h) \left(\frac{2^p}{p!}\rho_x^2\rho_y^2 + e^{-\rho_y^2}\right) \tag{15}$$

remains small, where we denote the ratios by $\rho_x = r_x/h$, $\rho_y = r_y/h$ and p is the degree of the polynomial approximation of eq. (13). It is not hard to see that if we keep the ratios ρ_x and ρ_y small and the degree p high, we can make the error arbitrary small.

The farthest-point clustering algorithm has a complexity of $\mathcal{O}(NK)$ [11] and the Improved Fast Gauss Transform also achieves a linear runtime for the number of samples N, see [9]. So in the end we are able to evaluate our density estimator based on N source points at N different target points with a linear runtime $\mathcal{O}(N)$.

Bandwidth Matrix

The crucial part of using a kernel density estimator is to calculate a suitable bandwidth matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$. The finite amount of sample points are realizations of a random variable with values in $\mathbb{R}^n$ where we assume this random variable has an underlying continuous density function $f : \mathbb{R}^n \mapsto \mathbb{R}$. We want to approximate this density function with a kernel density estimate $f_{\mathbf{H}}$ such that the mean integrated squared error

$$\mathrm{MISE}(\mathbf{H}) := \mathbb{E} \int_{\mathbb{R}^n} \left(\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\boldsymbol{\xi}) - f(\boldsymbol{\xi})\right)^2 \mathrm{d}\boldsymbol{\xi}$$

is minimized. Following the approach in [12] we introduce the leave-one-out estimator

$$\mathrm{KDE}_{\mathbf{H},i}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{p}) := \frac{1}{N-1} \sum_{i=1, i\neq j}^{N} K_{\mathbf{H}}(\mathbf{p} - \mathbf{x}_i),$$

where $1 \leq i \leq N$. We introduce a method how to estimate a good diagonal bandwidth matrix $\mathbf{H} = \mathrm{diag}(\mathbf{h})$, where $\mathbf{h} = (h_1, \dots, h_N)$ and $\mathrm{KDE}_{\mathbf{H}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}} = \mathrm{KDE}_{\mathbf{h}}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}$. Our goal is it to maximize

the likelihood function

$$L(\mathbf{h}|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \prod_{i=1}^{N} \mathrm{KDE}_{\mathbf{h},i}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{x}_i), \tag{16}$$

where we assume a prior for every h_k

$$\pi_0(h_k|\lambda) \propto \frac{1}{1+\lambda h_k^2}, \tag{17}$$

where λ is a hyperparameter controlling the shape of the prior density. So in the end we obtain the posterior density

$$\pi(\mathbf{h}|\mathbf{x}_1, \ldots, \mathbf{x}_N) \propto \prod_{k=1}^{n} \frac{1}{1+\lambda h_k^2} \cdot \prod_{i=1}^{N} \mathrm{KDE}_{\mathbf{h},i}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}(\mathbf{x}_i),$$

from which we can generate samples with the Metropolis algorithm.

Preparation of Data

First, we will calculate a bandwidth matrix $\mathbf{H}$. Let $(\mathbf{x}_i)_{i\in\mathcal{I}}$ be our source points. We calculate the empiric mean vector

$$\mathbf{m} = \frac{1}{N}\sum_{i\in\mathcal{I}} \mathbf{x}_i$$

and the empiric covariance matrix

$$\mathbf{\Sigma} = \frac{1}{N-1}\sum_{i\in\mathcal{I}} (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^\top.$$

Let $\mathbf{\Sigma}^{1/2}$ denote the principal square root and let $\mathbf{\Sigma}^{-1/2}$ denote the inverse of the principal square root of the symmetric and positive definite matrix $\mathbf{\Sigma}$. Then the data

$$\mathbf{z}_i := \mathbf{\Sigma}^{-1/2}(\mathbf{x}_i - \mathbf{m}), \quad i \in \mathcal{I}, \tag{18}$$

are the shifted, rotated and scaled (a.k.a. sphered) points of the $\mathbf{x}_i$ such that they have an estimated mean value of zero and a unit estimated covariance matrix

$$\frac{1}{N}\sum_{i\in\mathcal{I}} \mathbf{z}_i = \mathbf{0}, \quad \frac{1}{N-1}\sum_{i\in\mathcal{I}} \mathbf{z}\mathbf{z}^\top = \mathbf{I}.$$

Then, based on these manipulated data $\mathbf{z}_i$ we will use a Metropolis approach, described above, to estimate a diagonal bandwidth matrix $\mathbf{H} = \mathrm{diag}(\mathbf{h})$. Since the samples $\mathbf{z}_i$ are uncorrelated it is sufficient to introduce a diagonal bandwidth matrix instead of a full matrix. Furthermore, we get rid of differences in the magnitudes of different dimensions of the source points $\mathbf{x}_i$, which yields better performance of the evaluation of the kernel density estimator later on. The

corresponding bandwidth matrix for the original source points $\mathbf{x}_i$ results in

$$\boldsymbol{\Sigma}^{1/2}\mathbf{H}(\boldsymbol{\Sigma}^{1/2})^\top$$

because the calculation of $\mathbf{H}$ is based on the source points $\mathbf{z}_i$ calculated with eq. (18). This matrix is then a full matrix due to the transformation.

Overall Procedure for Evaluation

For the overall procedure let $\mathbf{p}_j$ be a target point at which we want to evaluate the kernel density destimator. First, we transform the target point and the source points accordingly by

$$\begin{aligned}\mathbf{v}_j :=& \mathbf{H}^{-1/2}\boldsymbol{\Sigma}^{-1/2}(\mathbf{p}_j - \mathbf{m}),\\ \mathbf{w}_i := \mathbf{H}^{-1/2}\mathbf{z}_i =& \mathbf{H}^{-1/2}\boldsymbol{\Sigma}^{-1/2}(\mathbf{x}_i - \mathbf{m}), \quad i \in \mathcal{I},\end{aligned}$$

such that

$$\begin{aligned}\mathrm{KDE}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}_{\boldsymbol{\Sigma}^{1/2}\mathbf{H}(\boldsymbol{\Sigma}^{1/2})^\top}(\mathbf{p}_j) &= \det\mathbf{H}^{-1/2}\det\boldsymbol{\Sigma}^{-1/2}\mathrm{KDE}^{(\mathbf{w}_i)_{i\in\mathcal{I}}}_{\mathbf{I}}(\mathbf{v}_j)\\ &= \det\mathbf{H}^{-1/2}\det\boldsymbol{\Sigma}^{-1/2}\frac{1}{N}\sum_{i\in\mathcal{I}}K_{\mathbf{I}}(\mathbf{v}_j - \mathbf{w}_i).\end{aligned} \tag{19}$$

The last expression uses the Gaussian kernel with a unit covariance. This way we can rewrite eq. (19) as

$$\begin{aligned}\mathrm{KDE}^{(\mathbf{x}_i)_{i\in\mathcal{I}}}_{\boldsymbol{\Sigma}^{1/2}\mathbf{H}(\boldsymbol{\Sigma}^{1/2})^\top}(\mathbf{p}_j) &= D\cdot A(1)\sum_{i\in\mathcal{I}}e^{-\left\|\mathbf{v}_j-\mathbf{w}_i\right\|^2},\\ D &:= \det\mathbf{H}^{-1/2}\det\boldsymbol{\Sigma}^{-1/2},\end{aligned} \tag{20}$$

and A from eq. (12). Then with the farthest-points Algorithm 3 we use the Improved Fast Gauss Transform Algorithm 4 with $h = 1$ to evaluate the transformed density at the target point $\mathbf{v}_j$. As mentioned before we are able to do this with a computational runtime $\mathcal{O}(N)$.

Iterative Metropolis

We summarise this section with the iterative Metropolis Algorithm 5. This procedure can be applied whenever we have K lists of measurements $(\mathbf{y}_i^k)_{i=1,\dots,N_{\mathrm{obs}}^k}$, $k = 1,\dots,K$.

In Algorithm 5 in lines 5–6 a bandwidth matrix is calculated using the classic Metropolis algorithm [4], whereby the prior distribution is defined in eq. (17) and we take the likelihood from eq. (16). For every iteration we transform the data as stated in lines 8–9, which we described in detail previously. For the application of the DRAM algorithm in line 11 we then use a composition of the kernel density estimator eq. (20), see line 10, and this transformation for the prior π_0. For the next iteration we treat the just received points $(\mathbf{x}_i)$ as the source points, therefore we calculate the estimated mean $\mathbf{m}$ and the estimated covariance matrix $\boldsymbol{\Sigma}$ at lines 12–13.

Algorithm 5: Iterative Metropolis

Data: Observation function Q, K lists of measurements $((\mathbf{y}_i^k)_{i=1,\dots,N_{\text{obs}}})_{k=1,\dots,K}$, number of samples N, first value $\mathbf{q}_0$, covariance matrix of proposal $\mathbf{V}$, covariance matrix of likelihood $\mathbf{\Sigma}_L$, prior π_0, factor γ^2, value s, update matrix every k_0 steps

Result: $(\mathbf{q}_i)_{i=0,\dots,N}$

$(\mathbf{x}_i)_{i=0,\dots,N} \leftarrow$ **DRAM** $\left(Q, (\mathbf{y}_i^1)_{i=1,\dots,N_{\text{obs}}^1}, N, \mathbf{q}_0, V, \mathbf{\Sigma}_L, \pi_0, \gamma^2, s, k_0\right)$;

$\mathbf{m} \leftarrow \frac{1}{N}\sum_{i=0}^{N} \mathbf{x}_i$;

$\mathbf{\Sigma} \leftarrow \frac{1}{N}\sum_{i=0}^{N} \|\mathbf{x}_i - \mathbf{m}\|^2$;

$\mathbf{z}_i \leftarrow \mathbf{\Sigma}^{-1/2}(\mathbf{x}_i - \mathbf{m})$;

$\mathbf{h} \leftarrow$ Sample Metropolis with prior from eq. (17), likelihood $L(\mathbf{h}|\mathbf{z}_0, \dots, \mathbf{z}_N)$ from eq. (16);

$\mathbf{H} \leftarrow \operatorname{diag}(\mathbf{h})$;

for $k = 2$ **to** K **do**

  $\mathbf{w}_i \leftarrow \mathbf{H}^{-1/2}\mathbf{\Sigma}^{-1/2}(\mathbf{x}_i - \mathbf{m}), i = 0, \dots, N$;

  $T \leftarrow \left[\mathbf{p} \mapsto \mathbf{H}^{-1/2}\mathbf{\Sigma}^{-1/2}(\mathbf{p} - \mathbf{m}) =: \mathbf{v}\right]$;

  $\pi \leftarrow$ see eq. (20);

  $(\mathbf{x}_i)_{i=0,\dots,N} \leftarrow$ **DRAM** $\left(Q, (\mathbf{y}_i^k)_{i=1,\dots,N_{\text{obs}}^k}, \mathbf{x}_N, \mathbf{V}, \mathbf{\Sigma}_L, \pi_0 = \pi \circ T, \dots\right)$;

  $\mathbf{m} \leftarrow \frac{1}{N}\sum_{i=0}^{N} \mathbf{x}_i$;

  $\mathbf{\Sigma} \leftarrow \frac{1}{N}\sum_{i=0}^{N} \|\mathbf{x}_i - \mathbf{m}\|^2$;

$\mathbf{q}_i \leftarrow \mathbf{x}_i, i = 0, \dots, N$.

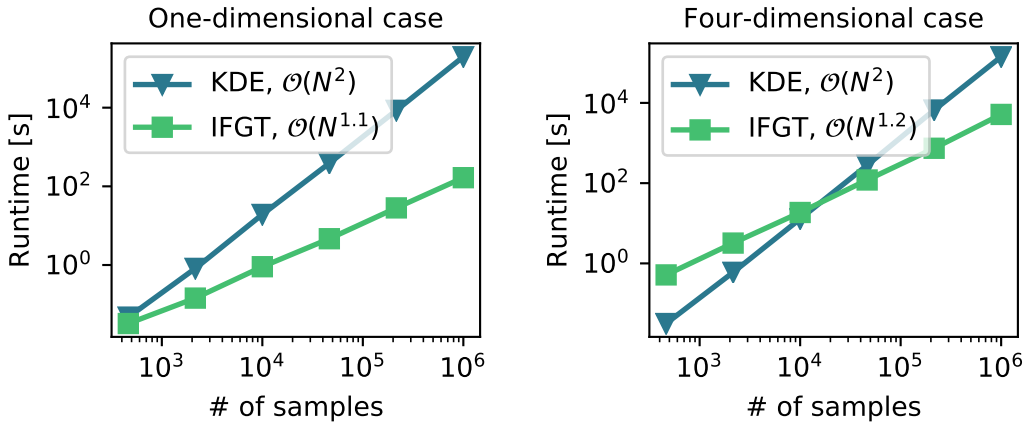


Figure 3: Runtime comparisons between naive kernel density estimation and evaluating the approximated function using the Improved Fast Gauss Transform.

4 Numerical Examples

We will now introduce examples for parameter estimation in high dimensions using the iterative algorithm. All numerical calculations were performed using Julia [13].

4.1 Introduction

Bayesian parameter estimation for this particular application was presented in [14, 15].

We consider a sensor consisting of nanoelectrodes, built for sensing particles in an electrolyte [16–18]. The main part consists of a 256×256 array of nano-electrodes which are excited with a high-frequency AC signal. These electrodes behave like capacitors. A chip with this array of capacitors placed in an electrolyte with no particles in it forms the initial state. Whenever a particle comes close to the chip it perturbes the field and changes the capacitances of the electrodes. This change can be measured and one can conclude the presence of a target object. This sensor is used to detect various particles such as biomolecules, viruses, and bacteria.

4.2 Model

The model is based on the Poisson-Boltzmann equation. We will focus on the AC small-signal regime formulation, since we consider the sensor in a high-frequency AC regime only. Then the Poisson equation is

$$\nabla \cdot \left(\varepsilon \tilde{V}\right) + \sum_{m=1}^{N_{ions}} \frac{q^2 Z_m^2}{k_B T} n_{0_m} \left(\tilde{\phi}_m - \tilde{V}\right) = 0,$$

where $\tilde{V}$ is the electrostatic potential phasor, $\tilde{\phi}_m$ and Z_m are the pseudo-potential and the signed valence of the m-th ionic species, respectively. The DC ion concentration of the m-th ionic species is described by

$$n_{0_m} = n_m^{\infty} \exp\left(\frac{Z_m q}{k_B T}(\phi_{0m} - V_0)\right).$$

The other part is the linearized drift-diffusion equation

$$q Z_m \mu_m \nabla \cdot (n_{0_m} \nabla \tilde{\phi}_m) - j\omega n_{0_m} \frac{q Z_m}{k_B T} \left(\tilde{\phi}_m - \tilde{V}\right) = 0,$$

which describes the current, where μ_m is the mobility of the m-th ionic species, j is the imaginary unit, and ω is the angular frequency. Together they form the model for the nano-electrodes. The model has been extensively validated in [19]. The simulations are done with the ENBIOS code [20–23].

To be able to perform the Metropolis algorithm we introduce our observation space to be the capacitances of the individual nano-electrodes, measured in aF.

Using Theorem 4 we see that this models yields a well-defined posterior, if we are only using

positively and negatively charged ion species and if the prior is a Gaussian.

4.3 Estimating the Radii of Electrodes

It is claimed [16–18, 24] that the nominal radius of one cylindrical shaped electrode is 90nm. However, to give an better estimation of this parameter we will perform a series of approaches to estimate the radii of individual electrodes. For all the following Bayesian estimation processes, we used the ENBIOS program to perform simulations. In addition, experimental data are available.

Global Radius

Our first approach is the following. We assume that all nano-electrodes have the same radius which we are going to estimate. This one-dimensional inference task was achieved with the Algorithm 2. We have chosen a uniformly distributed prior distribution on the interval $[70\text{nm}, 110\text{nm}]$ for the radius. For the observations, an array of 7×7 electrodes was simulated. One measurement for the capacitance is provided, 54.61aF with an uncertainty of 1aF (standard deviation), taken at one fixed frequency for the AC signal. With all this information we are able to examine the posterior distribution, which density is approximated with a histogram based on 10^5 samples, see Figure 4, and we take the mean value of 85.9nm to be our estimate of the global radius.

Local Radius

For the rest of paper we focus on a certain fixed 71×59 sub-array of nano-electrodes of the full 256×256 array. Previously we managed to estimate the radius to be 85.9nm. Now, we estimate the size of every individual electrode and consider the dimension of one electrode to be the random variable. We fix the remaining radii at 85.9nm, see Figure 5. We perform this one-dimensional estimation apprach for every electrode on the 71×59 array. This way we eventually get a full array of the estimated radii. For this task we also have more measurements we can take into account. Namely, for every frequency of a total of 8 different frequencies, we are provided with the capacitances of all $71 \cdot 59 = 4189$ electrodes. We perform our iterative Metropolis scheme for all 4189 electrodes, whereby we iterate over the 8 different frequencies. For one specific electrode we show the iteration over the posterior densities in Figure 6.

For an estimation of the radius we take the mean value of the last posterior (since this one consideres all the measurements from all frequencies). In Figure 7 on the left one can also examine a plot of the estimated radii for the whole array, and on the right is the distribution of the different estimated radii is shown as a histogram.

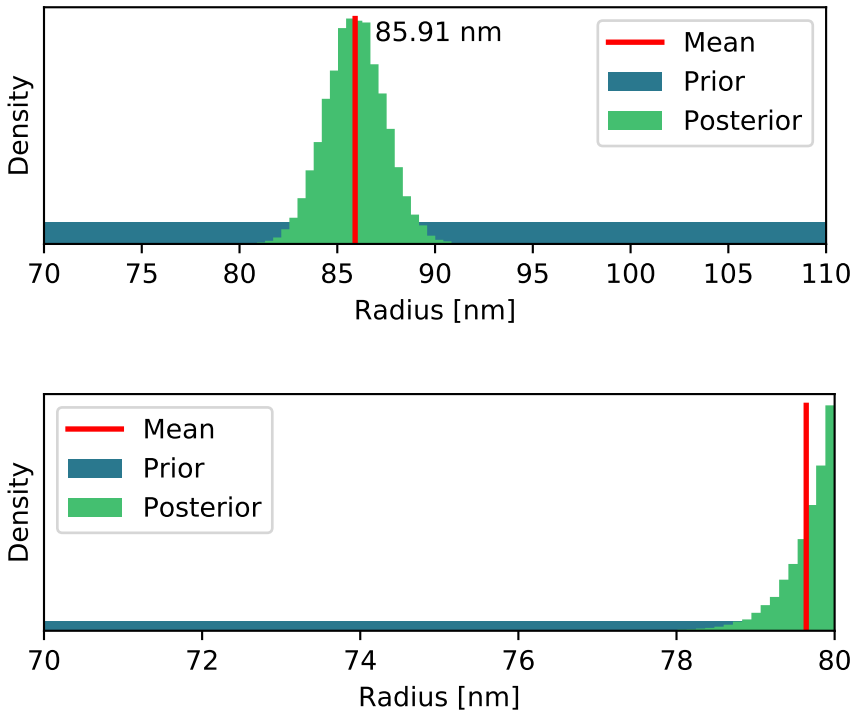


Figure 4: The prior densities and histograms of the posterior densities are shown. On the top, the support of the prior density seems to cover a sufficiently large domain in order that our MCMC algorithm can explore the whole domain of the posterior. The mean value of the global radius is estimated to be 85.9nm. In contrast: On the bottom apparently the prior density does not cover the whole domain of the posterior which results in a non-symmetrical histogram whereby the mean value is at the very edge.

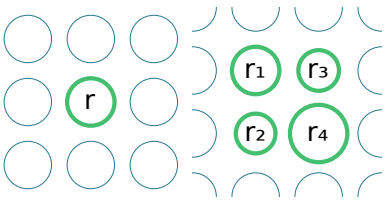


Figure 5: Left: Arrangement of the simulated nano-electrodes. The radius r of the center electrodes will be estimated, while the other electrodes (blue) are fixed in their size. Right: Analogue, but four radii are estimated simultaneously, whereby the surrounding electrodes are again fixed in their size.

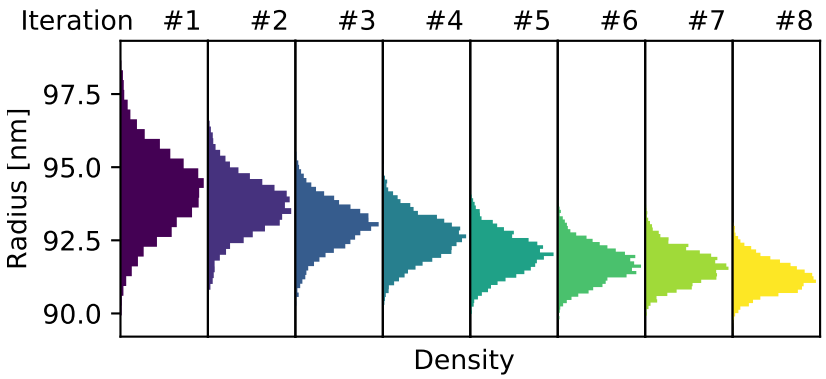


Figure 6: From left to right one can examine the posterior histograms for the radius of one electrode from the 71×59 *array. With every iteration (noted above), a different frequency is considered. Every posterior distribution plays the role of the prior for the following iteration.*

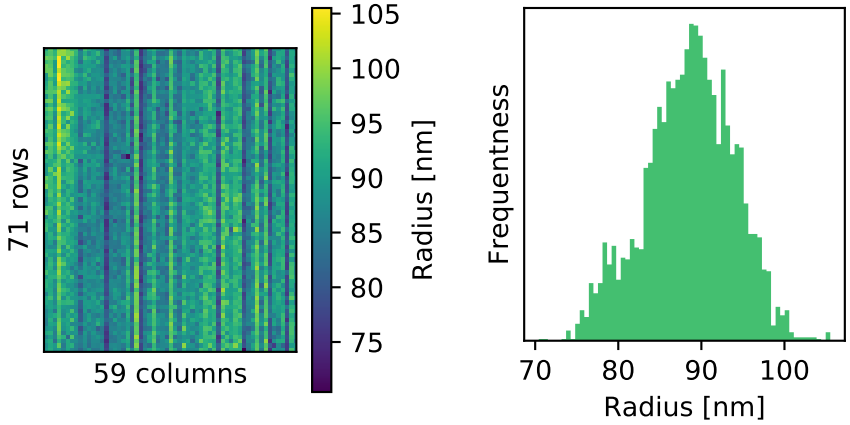


Figure 7: Left: Color map of the estimated radii (mean value of posterior) of the whole 71×59 *array. Right: Distribution of the different mean values from the color map to the left.*

Four Local Radii

Finally, we perform the same kind of estimation we did before, investigating the radius of the electrodes whereby we again consider the measurements taken at the 8 different frequencies. The only difference now is that we consider four neighbouring electrodes at the same time, whereby we again fix all the other electrode radii at 85.9nm. The arrangement for this case is also shown in Figure 5.

Performing this four-dimensional Bayesian estimation task with our iterative method results in a sequence of sets of four-dimensional samples, whose one-dimensional marginal histograms are shown in Figure 8. This four-dimensional approach was performed on the whole 71×59 array to get estimations for all electrodes.

To sum this section up, we are presenting the difference between the estimations of the radii, using the four-dimensional and the one-dimensional approaches. These results are presented in Figure 9.

Computation Time

Since we use the method of the Improved Fast Gauss Transform to evaluate an approximation of a density estimation, which we need for our iterative Metropolis algorithm, perfomed both for the one-dimensional case and the four-dimensional case, we present the actual runtimes depending on the number of samples see in Figure 3.

The runtimes for the clustering algorithm and for the calculations of the coefficients in eq. (14) are not considered.

5 Conclusions

We have presented an approach to perform Bayesian estimation with the Delayed-Rejection Adaptive-Metropolis algorithm, whereby we used the obtained posterior distribution as a prior distribution for an *iteration* of multi-dimensional Markov-chain Monte Carlo simulations. Hereby, a density estimation using a finite amount of random samples was necessary. Therefore we introduced a kernel density estimator, where we used the principle of the improved fast Gauss transform to receive a runtime of $\mathcal{O}(N)$ in dependence of the number of samples, compared to the naive implementation which would result in a worse performace of $\mathcal{O}(N^2)$. When a kernel density-estimator is used, the bandwidth matrix is essential. To calculate this matrix we used another Bayesian estimation approach, which minimizes the error due to using an approximation of the densities. With the use of the DRAM algorithm we achive an algorithm which is more robust against weak initial proposal distributions. It also prevents stagnations in the paths of the Markov chains which results in a smoother and better estimation of the underlying distribution. The algorithms were applied to the model of a nano-capacitor array where we estimated the radii of multiple nano-electrodes using multi-frequency measurements.

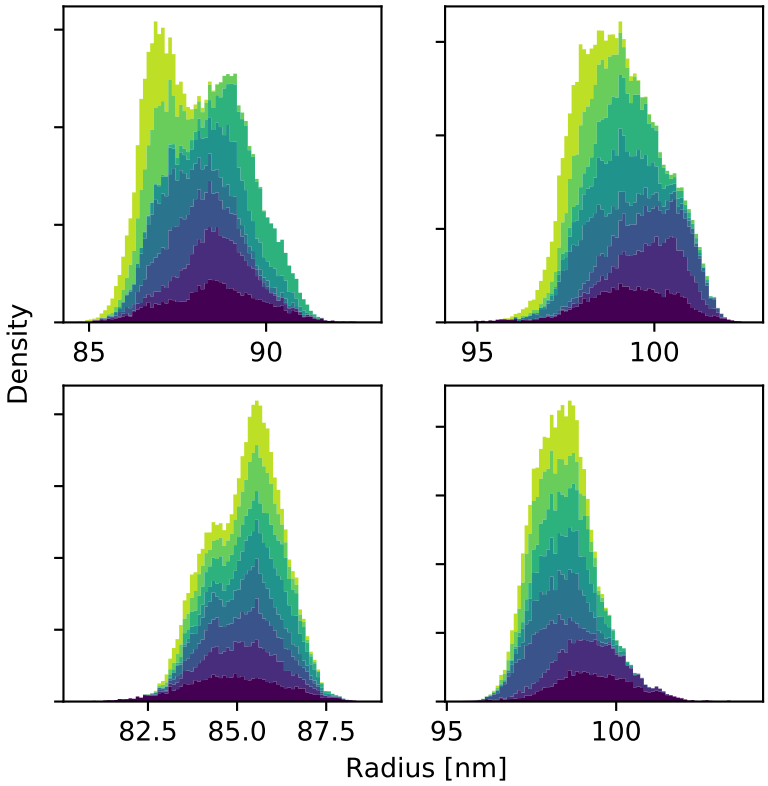


Figure 8: Marginal histograms of the sequence of four-dimensional posterior densities. For each iteration, the histogram is stacked on top of the previous one. Again, for the estimation, the mean value is taken from the last posterior density.

Acknowledgments

This work was funded by the Austrian Science Fund (FWF) START project No. Y660 *PDE Models for Nanotechnology*.

References

[1] Andrew M Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[2] Leila Taghizadeh, Benjamin Stadlbauer, Jose Morales, and Clemens Heitzinger. Bayesian inversion for electrical impedance tomography. pages 1–19, 2018. *In preparation*.

[3] Leila Taghizadeh, Samaneh Mokhtari, and Clemens Heitzinger. Bayesian estimation for parameter identification in Poisson-Boltzmann equation. 2018. *In preparation*.

[4] Ralph C Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*, volume 12 of *CS&E*. SIAM, 2013.

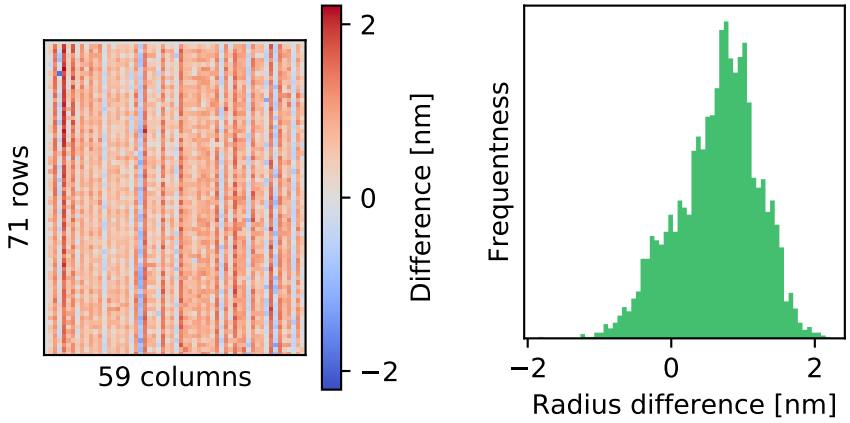


Figure 9: Left: Color plot of the signed difference between the four-dimensional and the one-dimensional Bayesian estimation approch for the radii ($r_{4dim} - r_{1dim}$). Right: Histogram of the distribution of these differences.

[5] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[6] Heikki Haario, Marko Laine, Antonietta Mira, and Eero Saksman. DRAM: efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, 2006.

[7] A. Cossettini, B. Stadlbauer, J. Morales E., L. Taghizadeh, L. Selmi, and C. Heitzinger. Determination of micro- and nano-particle properties by multi-frequency Bayesian methods and applications to nanoelectrode array sensors. pages 1–4, 2018. *In preparation*.

[8] David W Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

[9] Yang, Duraiswami, Gumerov, and Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 664–671 vol.1, Oct 2003.

[10] Tarn Duong and Martin L Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):485–506, 2005.

[11] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[12] Xibin Zhang, Maxwell L King, and Rob J Hyndman. A Bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 50(11):3009–3031, 2006.

[13] J. Bezanson, A. Edelman, S. Karpinski, and V. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[14] Benjamin Stadlbauer, Leila Taghizadeh, Jose Morales Escalante Clemens Heitzinger, Andrea Cossettini, and Luca Selmi. Bayesian estimation for transport equations for nanocapacitors. In *Proc. SIAM Conference on Uncertainty Quantification (UQ 2018)*, pages 69–70, Garden Grove, CA, USA, 16–19 April 2018.

[15] Andrea Cossettini, Paolo Scarbolo, Jose Morales Escalante, Benjamin Stadlbauer, Naseer Muhammad, Leila Taghizadeh, Clemens Heitzinger, and Luca Selmi. Calibration, compensation, parameter estimation, and uncertainty quantification for nanoelectrode array biosensors. In *Proc. SIAM Conference on Uncertainty Quantification (UQ 2018)*, page 81, Garden Grove, CA, USA, 16–19 April 2018.

[16] Cecilia Laborde, Federico Pittino, HA Verhoeven, SG Lemay, Luca Selmi, MA Jongsma, and FP Widdershoven. Real-time imaging of microparticles and living cells with CMOS nanocapacitor arrays. *Nature Nanotechnology*, 10(9):791, 2015.

[17] Serge G Lemay, Cecilia Laborde, Christophe Renault, Andrea Cossettini, Luca Selmi, and Frans P Widdershoven. High-frequency nanocapacitor arrays: Concept, recent developments, and outlook. *Accounts of Chemical Sesearch*, 49(10):2355–2362, 2016.

[18] Frans Widdershoven, Andrea Cossettini, Cecilia Laborde, Andrea Bandiziol, Peter Paul van Swinderen, Serge G Lemay, and Luca Selmi. A CMOS pixelated nanocapacitor biosensor platform for high-frequency impedance spectroscopy and imaging. *IEEE Transactions on Biomedical Circuits and Systems*, 2018.

[19] F. Pittino, P. Scarbolo, F. Widdershoven, and L. Selmi. Derivation and numerical verification of a compact analytical model for the AC admittance response of nanoelectrodes, suitable for the analysis and optimization of impedance biosensors. *IEEE Transactions on Nanotechnology*, 14(4):709–716, July 2015.

[20] Federico Pittino and Luca Selmi. Use and comparative assessment of the CVFEM method for Poisson–Boltzmann and Poisson–Nernst–Planck three dimensional simulations of impedimetric nano-biosensors operated in the DC and AC small signal regimes. *Computer Methods in Applied Mechanics and Engineering*, 278:902–923, 2014.

[21] Andrea Cossettini and Luca Selmi. On the response of nanoelectrode impedance spectroscopy measures to plant, animal, and human viruses. *IEEE Transactions on Nanobioscience*, 17(2):102–109, 2018.

[22] Andrea Cossettini, Matteo Dalla Longa, Paolo Scarbolo, and Luca Selmi. Modeling and simulation of small CCMV virus detection by means of high frequency impedance spectroscopy at nanoelectrodes. In *17th IEEE International Conference on Nanotechnology 2017 (IEEE-NANO)*, pages 416–419. IEEE, 2017.

[23] Clemens Heitzinger and Leila Taghizadeh. Analysis of the drift-diffusion-Poisson-Boltzmann system for nanowire and nanopore sensors in the alternating-current regime. *Communication in Mathematical Science*, 15(8):2303–2325, 2017.

[24] F. Widdershoven, D. Van Steenwinckel, J. Überfeld, T. Merelle, H. Suy, F. Jedema, R. Hoofman, C. Tak, A. Sedzin, B. Cobelens, E. Sterckx, R. van der Werf, K. Verheyden, M. Kengen, F. Swartjes, and F. Frederix. CMOS biosensor platform. In *2010 International Electron Devices Meeting (IEDM)*, pages 36.1.1–36.1.4, Dec 2010.