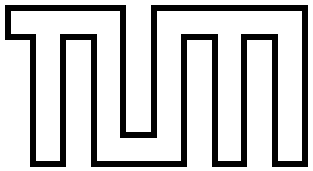


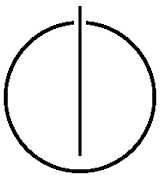
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Interdiscplinary Project Paper

# Adaptive Construction of Surrogate Models Based on Sparse Grid Interpolants for Bayesian Inverse Problems


| | |
|---|---|
| Author: | Pablo Gómez |
| Supervisor: | Univ.-Prof. Dr. Hans-Joachim Bungartz |
| Advisor: | Emily Mo-Hellenbrand |
| Date: | July 15, 2015 |

# Abstract

In this paper we present an adaptive surrogate model based on sparse grid interpolation for a bayesian inverse Navier-Stokes problem. We introduce the concept of inverse problems from a bayesian perspective and a finite difference method as forward model for the Navier-Stokes problem. The basic problem setup is a channel in which up to four obstacles are placed and velocity magnitude measurements are taken. From these measurements we want to infer the locations of the obstacles using a Metropolis-Hastings Markov chain Monte Carlo solver. Our surrogate model relies on a sparse grid interpolation of the forward model, which we then refine online by creating higher level sparse grids in areas of interest, i.e. likely obstacle locations. We present detailed results in the form of histograms for both the finite difference method and our adaptive surrogate model. Overall we achieve a higher accuracy using a computationally less expensive model. Our approach is also applicable to other inverse problems which will be the feature of future research.

# Contents

# 1 Introduction

With the vast increase in the amount of computational power available, the potential to solve a variety of numerical problems has steadily grown and new methods are actively researched and evaluated every day. Fueled by a similar development in regard to the availability of data and computational power statistics has also been on the rise.
One particular problem set often dealt with using methods in the field of statistics are inverse problems, which are often ill-posed and therefore particularly hard to solve. However, they are of interest to a multitude of research fields ranging from the creation of astrophysical images from telescope data to the inference of oil deposit locations.
In this paper we analyze an adaptive sparse grid method to increase the accuracy in the solution of an inverse Navier-Stokes problem, where we seek to determine the location of flow obstacles in a two-dimensional space. In particular we employ a Markov chain Monte Carlo (MCMC) method in form of a Metropolis-Hastings algorithm first on a finite-difference discretization of the equation and secondly on a sparse grid interpolant that is adaptively refined to aquire a higher accuracy in space regions that are likely obstacle locations.
Affected by the infamous *curse of dimensionality* the convergence rate of the MCMC method slows down significantly with increased dimensionality, i.e. a higher number of obstacles in our case. Research has been conducted in the parallelization of the MCMC method as shown in [11].
Another way to cope with the *curse of dimensionality* is the application of sparse grids. They have been successfully used in a variety of problem settings as shown, e.g., in [5]. In our setting, they allow an efficient parallelization additionally. Using a spatial refinement of our domain we are furthermore able to increase the overall accuracy in solving the inverse problem.
There has already been some research into similar approaches, e.g. in [7], where the sparse grid is refined instead of the spatial domain and in [2] where a combination of sparse grid quadrature, reduced basis approximation and empirical interpolation is used.
In the following sections we first introduce the general inverse problem and then a specific case, an inverse Navier-Stokes problem. Thereafter a brief introduction into sparse grid methodology and a detailed explanation of the algorithm we used ensue. Finally we present our results and analyze possible conclusions and further research.

# 2 Solving inverse problems

In this section we introduce inverse problems and discuss the Bayesian approach to solving inverse problems. Furthermore, we describe the Metropolis-Hastings algorithm, a basic Markov chain Monte Carlo method for solving inverse problems.

## 2.1 Inverse problems

Following the procedure in [8], we focus on an inverse problems of the form

$$y = G(x), \tag{1}$$

where $y \in \mathbb{R}^N$ describes data that is observed or measured, whereas *the forward model* $G(\cdot) : \mathbb{R}^M \to \mathbb{R}^N$ relates the input parameters $x \in \mathbb{R}^M$ to the output data. If there is no analytical way to aquire the input data $x$ that lead to the desired output $y$, it is necessary to find a strategy that relies on computing different results of the forward model for different input parameters. A more detailed description of this is given in [10].
Furthermore, as measurements are often affected by noise, we also introduce a noise term $\eta$, so that

$$y = G(x) + \eta. \tag{2}$$

In this paper, we focus on so-called statistical inverse problems, i.e. we introduce the Bayesian approach to inverse problems. For events $A$ and $B$, Bayes' Theorem is given as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, P(B) > 0, \tag{3}$$

where $P(A)$ and $P(B)$ are the probabilities of the events $A$ and $B$ respectively. $P(A|B)$ and $P(B|A)$ are the probabilities of event $A$ if event $B$ takes place and respectively vice versa. Furthermore, we now assume the following:

1. All variables included are modelled as random variables
2. The randomness of those variables describe the observer's degree of information
3. The probability distributions $\pi(x), \pi(y), \pi(x|y), \pi(y|x)$ - also referred to as prior, marginal likelihood, posterior distribution and likelihood - respectively describe all information regarding above mentioned random variables
4. The solution to the inverse problem is described by the posterior distribution $\pi(x|y)$

The inverse problem can then be written as

$$Y = G(X) + H, \tag{4}$$

for $X \in R^M, Y, H \in R^N$. The solution to it is then

$$\pi_{XY}(x|y) = \frac{\pi_{XY}(y|x)\pi_X(x)}{\pi_Y(y)}, \tag{5}$$

where $\pi_X(x) = P(X = x)$, i.e. the probability of $X$ taking value $x$ and $\pi(y), \pi(x|y), \pi(y|x)$ are used analogously. To decrease the notational burden, we omit the capital letters. Finally, as shown in [8], this can be simplified and used, so that the final posterior distribution $\tilde{\pi}_{pos}(x)$ takes the form

$$\tilde{\pi}_{pos}(x) = \pi_{noise}(y - G(x)|x)\pi(x), \tag{6}$$

if we disregard the uncertainty of the simulation model and where $\pi_{noise}$ is the probability distribution of the noise $H$ and $\tilde{\pi}_{pos}(x) \propto \pi_{pos}(x)$.

## 2.2 Metropolis-Hastings algorithm

Markov chain Monte Carlo (MCMC) methods are a standard tool in the setting of Bayesian inference. A more detailed introduction to them can be found in [9].
The algorithm used in this paper is the popular Metropolis-Hastings algorithm. It is based on the first MCMC method, the Metropolis algorithm, that was able to construct a Markov chain that will converge to the exact posterior distribution. Following [8] we can define it as

**Algorithm 2.1** ***Metropolis-Hastings Algorithm***
*For a target distribution $p(x)$ and a distribution $q(a|b)$, given a state $x_n$ at step $n$:*
*1. Draw a proposal $y$ from $q(x^*|x_n)$*
*2. Evaluate the acceptance ratio*

$$a(x_n, x^*) = min\left\{1, \frac{p(x^*)q(x_n|x^*)}{p(x_n)q(x^*|x_n)}\right\}$$

*3. Accept the candidate with probability $a(x_n, x^*)$ and set $x_{n+1} = x^*$, respectively , i.e. with probability $1 - a(x_n, x^*)$, set $x_{n+1} = x_n$*

In the following sections, we use a normal distribution as proposal distribution $q(x^*|x_n) = \mathcal{N}(x_n, \sigma)$ , where $\sigma$ is the random walk step size,which in our case is set to be $\frac{1}{20}$ of the domain size for each dimension. Furthermore, we only update one component per step, i.e. for a step $X^n = [x_1^n, \ldots, x_i^n, \ldots, x_1^n]^T \in \mathbb{R}^M$, the next step is given by $X^{n+1} = \left[x_1^n, \ldots, x_i^{n+1}, \ldots, x_1^n\right]^T \in \mathbb{R}^M$.

# 3 An inverse Navier-Stokes problem

This section introduces the setup of the inverse Navier-Stokes problem. Once again we refer to [8] for a more detailed description. Further research on inverse Navier-Stokes scenarios can be found in [3].
Our basic problem setup is a $10 \times 2$ channel filled with incompressible fluid, in which we place up to four obstacles that have a fixed size but unknown location. Based on measurements of the magnitude of the velocity in space and time we want to infer the locations of the obstacles.
As forward model we use a finite-difference method to solve the Navier-Stokes equations, which is explained first. Thereafter the inverse problem is formulated.

## 3.1 A finite-difference method for the Navier-Stokes equations

The forward model used in this paper is created using a finite-difference method on the Navier-Stokes equations based on the procedure in [1]. We consider a $10 \times 2$ channel filled with a non-stationary incompressible fluid, i.e. $\Omega = [0, 10] \times [0, 2]$, as in [8]. We ignore gravity-related terms, as our problem does not include gravity. The Navier-Stokes equations describe it as

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\text{Re}}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}), \tag{7}$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\text{Re}}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}), \tag{8}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{9}$$

where $x$ and $y$ describe the space dimensions, $t$ is the time, $u$ and $v$ are the fluid velocities in $x$ and respectively $y$ direction and finally $p$ is the pressure. Re is in this context the Reynolds number describing the fluid's viscosity as relative magnitudes of the pressure and inertial forces [4]. Note that equation 7 and 8 are also referred to as the *momentum equations* and likewise 9 is called *continuity equation*.
The boundary conditions of our channel are modelled by an inflow condition on the left boundary , i.e. at $x = 0$, outflow on the right ($x = 10$), and no-slip boundaries at the top and bottom ($y = 0$, $y = 2$). The initial condition for $t = 0$ is given by $u_0 = u(x, y, 0)$ and $v_0 = v(x, y, 0)$.

As in [8] we use a staggered grid spatial discretization with $\Delta x = 0.1$ and $\Delta y = 0.1$ resulting in a $100 \times 20$ grid and follow the implementation of the obstacles in there.
The time discretization is performed using a forward Euler scheme, leading to

$$u_{i,j}^{n+1} = F_{i,j}^{n} - \frac{\Delta t}{\Delta x}(p_{i+1,j}^{n+1} - p_{i,j}^{n+1}), \tag{10}$$

$$v_{i,j}^{n+1} = G_{i,j}^{n} - \frac{\Delta t}{\Delta y}(p_{i,j+1}^{n+1} - p_{i,j}^{n+1}), \tag{11}$$

where $u_{i,j}^{n+1}$, $v_{i,j}^{n+1}$ and respectively $p_{i,j}^{n+1}$ are the $u$ and $v$ values at the cells - or cell-interfaces, see [8] - at indices $i$ and $j$ in timestep $n+1$. $F_{i,j}^{n}$ and $G_{i,j}^{n}$ are given by

$$F_{i,j} = u_{i,j} + \delta t \left( \frac{1}{\mathrm{Re}} \left( \left[ \frac{\partial^2 u}{\partial x^2} \right]_{i,j} + \left[ \frac{\partial^2 u}{\partial y^2} \right]_{i,j} \right) - \left[ \frac{\partial u^2}{\partial x} \right]_{i,j} - \left[ \frac{\partial uv}{\partial y} \right]_{i,j} \right) \tag{12}$$

$$G_{i,j} = u_{i,j} + \delta t \left( \frac{1}{\mathrm{Re}} \left( \left[ \frac{\partial^2 v}{\partial x^2} \right]_{i,j} + \left[ \frac{\partial^2 v}{\partial y^2} \right]_{i,j} \right) - \left[ \frac{\partial u^2}{\partial x} \right]_{i,j} - \left[ \frac{\partial uv}{\partial y} \right]_{i,j} \right) \tag{13}$$

Using the continuity equation and the finite-difference schemes

$$\left[ \frac{\partial u}{\partial x} \right]_{i,j}^{n+1} + \left[ \frac{\partial v}{\partial y} \right]_{i,j}^{n+1} = 0, \text{with} \tag{14}$$

$$\left[ \frac{\partial u}{\partial x} \right]_{i,j}^{n+1} = \frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x}, \tag{15}$$

$$\left[ \frac{\partial u}{\partial x} \right]_{i,j}^{n+1} = \frac{v_{i,j}^{n+1} - v_{i,j-1}^{n+1}}{\Delta y}, \tag{16}$$

the computation of $u^{n+1}$ and $v^{n+1}$ requires solving the equation

$$\frac{p_{i+1,j}^{n+1} - 2p_{i,j}^{n+1} + p_{i-1,j}^{n+1}}{\Delta x^2} + \frac{p_{i,j+1}^{n+1} - 2p_{i,j}^{n+1} + p_{i,j-1}^{n+1}}{\Delta y^2} = \frac{1}{\Delta t} \left( \frac{F_{i,j}^{n} - F_{i-1,j}^{n}}{\Delta x} + \frac{G_{i,j}^{n} - G_{i,j-1}^{n}}{\Delta y} \right). \tag{17}$$

This is equivalent to solving

$$Ap = b, \tag{18}$$

where $A$ is sparse matrix representing the Laplace operator and $b$ the right hand side of the previous equation.
Finally to achieve stability, we require $\Delta t$ to be satisfy

$$\Delta t = \tau \min \left( \frac{1}{2\mathrm{Re}} (\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2})^{-1}, \frac{\Delta x}{|u_m ax|}, \frac{\Delta y}{|v_m ax|} \right), \tag{19}$$

with $|u_m ax|$ and $|v_m ax|$ being the absolute maximum of the respective velocities and $\tau \in [0, 1]$.

### 3.2 The inverse problem

Given the above forward model, our inverse problem takes the following form. Following [8] we assume some sensors placed across the channel that are able to measure the magnitude of the fluid's velocity at different timesteps. Using this measurement data $d$ our inverse problem is then

$$d = G(\theta) + \eta, \tag{20}$$

where $\theta$ denotes the obstacle location as $\theta := \{(\theta_{x_1}, \theta_{y_1}), (\theta_{x_2}, \theta_{y_2}), \dots\}$. Note that these coordinates describe the left bottom corner of each obstacle and, to simplify, we assume each obstacle to be of size $0.4 \times 0.4$. Furthermore we consider all parameters in the forward model to be known. In particular we set $Re = 100, u_0 = 1.0, v_0 = 0$ and $p_0 = 0$.
The measurements are taken at the times $t = \{12.16, 24.66, 37.16, 49.66\}$ at the locations

$$x \in \begin{Bmatrix} (1.5\ \ 0.6) & (3.1\ \ 0.6) & (4.7\ \ 0.6) & (6.3\ \ 0.6) & (7.9\ \ 0.6) \\ (1.5\ \ 1.3) & (3.1\ \ 1.3) & (4.7\ \ 1.3) & (6.3\ \ 1.3) & (7.9\ \ 1.3) \end{Bmatrix}. \tag{21}$$

Note that this means our forward model $G(\theta)$ now maps obstacles coordinates $\theta \in \Omega_\theta$ to measurement data $d_* \in \mathcal{D} \subset \mathbb{R}^{40}$, which is noise-free at this point. As prior we assume a constant uniform distribution of the obstacles so that

$$\pi(\theta) = \mathcal{U}_{\Omega_\theta}. \tag{22}$$

To avoid complications we neglect overlapping obstacle locations as it would be aequivalent to merging two or more obstacles into one of a more complex shape. Finally we include noise by perturbing our measurements with additive Gaussian noise $\eta$, i.e.

$$d = G(\theta) + \eta, \ \text{ where } \ \eta \sim \mathcal{N}(0, \sigma^2 I). \tag{23}$$

We used $\theta_i \in \{(1.0\ \ 0.5), (8.0\ \ 1.0), (5.5\ \ 0.3), (3.5\ \ 1.3)\}$ and $\sigma = 0.1 * \overline{d}_*$ to produce the "observed "data, where $\overline{d}_*$ is the average value of the measurement data. Using this and 6, the solution can be written as

$$\pi_{pos}(\theta) = \pi_{noise}(d - G(\theta)\pi_{pr}(\theta) \propto \exp\left(-\frac{1}{2\sigma^2}(d - G(\theta))^T(d - G(\theta))\right). \tag{24}$$

# 4 Sparse grid interpolation

In this section, sparse grids are introduced, which allow us to cope with the *curse of dimensionality* to some degree [5]. We then describe our adaptive algorithm, in which we analyze spatial regions of interest in more detail to improve the accuracy of our solution.

## 4.1 Sparse grid basics

Sparse grids have been used in a multitude of settings because of their ability to lessen the computational demands of full grids in higher dimensions. Some of them can be seen in [6]. As this introduction is rather brief, we refer to [5] for a more detailed description.
As shown in [6], to create a sparse grid we use a so-called hierarchical basis function, which we define as

$$\phi_{l,i}(x) := \phi(2^l x - i),$$

where $\phi(x) = \max\{1 - |x|, 0\}$ for a given level $l \in \mathbb{N}$ and index $i \in \mathbb{N}$. The $d$-dimensional hierarchical basis functions are derived from this using the tensor product

$$\phi_{l,i} = \prod_{j=1}^{d} \phi_{l_j,i_j}, \tag{25}$$

with $l = (l_1, ..., l_d)$ and $i = (i_1, ..., i_d)$.
From this the so-called hierarchical increments $C_l$ are created as

$$C_l = \operatorname{span}\{\phi_{l,i} | i \in \mathcal{I}_l\}, \tag{26}$$

where

$$\mathcal{I}_l = \left\{ i_j \in \mathbb{N}; 1 \leq i_j < 2^{l_j}, i_j \text{ is odd and } i_j \in \mathbb{N}_0, 1 \leq j \leq d \right\}. \tag{27}$$

Following [8] the sparse grid is then

$$\mathcal{V}_\ell^{(1)} = \bigoplus_{|l|_1 \leq \ell + d - 1} C_l, \tag{28}$$

with $|l|_1 = \sum_j l_j$. This allows us to represent a sufficiently smooth function $f \in \mathcal{V}_n$ as

$$f(x) = \sum_{|l|_\infty \leq n} \sum_{i \in \mathcal{I}_l} \alpha_{l,i} \phi_{l,i}(x) \tag{29}$$

where $|l|_\infty = \max_{1 \leq j \leq d} l_j$ and the $\alpha_{l,i}$ are called hierchical surpluses given by

$$\alpha_{l,i} = f(x_{l,i}) - \frac{f(x_{l,i-1} - f(x_{l,i+1})}{2}, l \geq 1. \tag{30}$$

In the case of our inverse problem this means that we can interpolate the posterior $\pi_{pos}(\theta)$ from 24 using a sparse grid interpolation $\tilde{\pi}^{lv}_{pos}(\theta)$ of level $lv$

$$\pi_{pos}(\theta) \approx \tilde{\pi}^{lv}_{pos}(\theta) = \sum_{|l|_1 \leq n} \sum_{i \in \mathcal{I}_l} \alpha_{l,i} \phi_{l,i}(x). \tag{31}$$

One consequence of this is that we have to evaluate our finite-difference forward model at every sparse grid point. This can lead to a lot of evaluations as, e.g., an $8$-dimensional - i.e. in our case corresponding to four obstacles - sparse grid of level $8$ has $580865$ points. But, in contrast to the evaluations necessary during a run of the Metropolis-Hastings algorithm using the finite-difference foward model, these evaluations can be performed independently of each other and thereby fully in parallel.

## 4.2 A space-adaptive sparse grid approach for an inverse Navier-Stokes problem

Using a sparse grid surrogate model instead of the original finite-difference foward model can greatly improve the performance of our MCMC solver as the evaluations of the sparse grid are comptutationally cheaper than the evaluation of the finite-difference model. [8] However, it is necessary for the sparse grid to be of a level high enough to attain the desired accuracy. With an increasing number of obstacles, we face both the problem, that the convergence rate of the MCMC solver is slowing as well as that the number of sparse grid points is still increasing significantly. To deal with this, we propose an adaptive sparse grid approach that allows us to use a lower sparse grid level and increase overall accuracy. Instead of selecting a fixed level and spatial domain for the sparse grid, we start on a lower level sparse grid and then continue to investigate regions of high probability regarding the obstacle locations on a higher level sparse grid. In detail, the resulting algorithm can be described as

**Algorithm 4.1** ***Adaptive sparse grid algorithm***
*For $k$ obstacles in the channel of size $x_c \times y_c$ and a surrogate sparse grid model $\tilde{\pi}^{l}_{pos}(\theta)$ created using a sparse grid of level $l$ on a domain of size $\Omega^l_\theta$ and for a refinement parameter $\kappa_l$ and an error tolerance $\epsilon$*

*1. Select a starting level $l = l_1$ and a number of MCMC steps $N_{mcmc}$*

*2. Create a surrogate model $\tilde{\pi}^{l}_{pos}(\theta)$ by interpolating the foward model*

*3. Find the maximum point $\theta^{SG}$ of the sparse grid points , which is available because we had to evaluate these points to create the sparse grid*

*4. Run $N_{mcmc}$ MCMC steps with $\tilde{\pi}^l_{pos}(\theta)$ on the domain $\Omega^l_\theta$ starting at $\theta^{SG}$ in the first step and obtain as result some point $\theta^{max} \in \Omega^l_\theta$ , which is the point for which $p(x^*)q(x_n|x^*)$ was maximal*

*5. Compute the error of the measurement $d^{max}$ taken for $\theta^{max}$ compared to to original measurement $d$ using the finite-difference foward model. If it is smaller than the tolerance $\epsilon$ terminate, otherwise continue.*

*6. Create a domain $\Omega^{l+1}_\theta$ as*

$$\begin{aligned} \Omega^{l+1}_\theta = \quad & [max(0,\ \theta^{max}_{x_1} - \kappa_l \tfrac{x_c}{2}),\ \ min(x_c,\ \theta^{max}_{x_1} + \kappa_l \tfrac{x_c}{2})] \quad \times \\ & [max(0,\ \theta^{max}_{y_1} - \kappa_l \tfrac{y_c}{2}),\ \ min(y_c,\ \theta^{max}_{y_1} + \kappa_l \tfrac{y_c}{2})] \quad \times \\ & \ldots \end{aligned} \tag{32}$$

*7. Set $\kappa_{l+1} = \kappa_l \kappa_{l_1}$ and then increase the level $l$ by 1. Return to step 2.*

This algorithm has one main advantage over using a surrogate model of a set level on the entire domain. The starting level can be relative small, i.e. it only needs to give a very coarse representation of the posterior $\pi_{pos}(\theta)$ which makes it faster to compute and evalute during the MCMC. Still, it has to be chosen in such a way, that the likelihood of the actual obstacle locations lying in the next domain $\Omega^{l+1}_\theta$ is as high as possible. The final level of the surrogate model can be smaller as the domain size decreases with each iteration and thereby provides addiontal accuracy to the surrogate model. Consequently the accuracy will be better allowing us to either use either surrogate models of smaller levels and maintain accuracy or to get a higher overall accuracy. Finally all advantages of using a surrogate model in the first place remain.
However, the use of this algorithm also leads to potential pitfalls, which require the parameters to be chosen carefully. If a value of $\kappa$ is selected that is too small, the actual obstacle location might not lie in the smaller domains, which would interrupt the algorithm's convergence. If the number of MCMC steps $N_{MCMC}$ is too small the result might not be likeliest point of the surrogate model. Finally if the error tolerance is too small the algorithm might not terminate as our original measurement data contains noise and some error in the result is therefore almost unavoidable. Note also, that it is possible to run the algorithm without increasing the level of the sparse grid model inbetween as the decreasing domain size also increases the surrogate model's accuracy. This does affect the convergence rate though and was therefore not explored in more detail in this paper.
Overall, this algorithm provides a more flexible surrogate model, that allows for adaptivity in the online phase leading to a higher accuracy using smaller sparse grid levels.

# 5 Results

In this section, we present the results of running the finite-difference model in comparison to our adaptive surrogate model. Detailed results of the underlying non-adaptive surrogate model can be found in [8].
All results were obtained on the Sandy Bridge nodes of the $Munich\ Centre\ of\ Advanced\ Computing$[1] using a C++ implementation on one node for the finite-difference model and up to $16$ nodes using *MPI* for the surrogate model given its potential for parallelism.
For the implementation of the sparse grid we relied on the $SG^{++}$[2] library and to solve the system in 18 we used the biconjugate gradient stabilized method solver implemented in the $Eigen$[3] library.
As test-cases we present results for the obstacle locations, measurement points, channel size and discretization parameters mentioned in the previous sections. We present histogram-like plots for all results. We do not present runtime results because the surrogate models creation can be fully parallelized so that its potentially better performance is thereby apparent. Compared to [8] we recognize that our lack of runtime measurements might be confusing, but point out that as previously mentioned our algorithm can be used to either lessen the necessary computations or increase accuracy and we focussed on the latter.
Both models rely on the same MCMC solver implementation with the exception that the starting point for the finite-difference model is chosen randomly because of the lack of prior information whereas the surrogate model uses the best sparse grid point evaluated.

## 5.1 Finite-difference model

The finite-different model was run using $N_i^{MCMC} \in [10000, 15000, 20000, 20000]$ MCMC steps for $1, 2, 3$ and $4$ obstacles respectively. The first $30\%$ of the MCMC samples serve as burn-in and only samples after that are considered in the following figures.
In figures 1,2,3 and 4 we present the finite-difference results for up to four obstacles. Note that some error is unavoidable because of the noise added to the measurement. Furthermore as the finite-difference discretization uses $\Delta x = 0.1$ and $\Delta y = 0.1$ obstacles at e.g. $(0.93, 0.31)$ and $(0.97, 0.38)$ would result in the same output since their coordinates are in the same cell of the spatial discretization.
Overall the shown results for one and four obstacles provide a reasonable accuracy. In the four obstacle case the result is affected by the higher dimensionality. However, we encounter a few problems here. The one and four obstacle cases are mostly best-case scenarios, where the random first step of the MCMC solver was good enough for the solver to converge. Often though, especially for a higher dimensionality, i.e. the number of obstacles we have, it is possible, that the MCMC solver converges to a local maximum of the

[1] http://www.mac.tum.de/
[2] http://www5.in.tum.de/SGpp
[3] http://eigen.tuxfamily.org/

posterior distribution, where it is then stuck. This occured in the test cases for two and three obstacles.
The other problem is the increasingly slow convergence, note how we used only $N^{MCMC} = 10000$ MCMC steps for the one obstacle case. This does not suffice for higher dimensional cases and the convergence rate will be decreasing significantly with higher obstacle counts so that there is a clear limit to the possible dimensionality we are able to tackle using this approach.
A potential approach to tackle both problems would be using a more sophisticated MCMC method or, as shown in this paper a surrogate model that can deal with them.

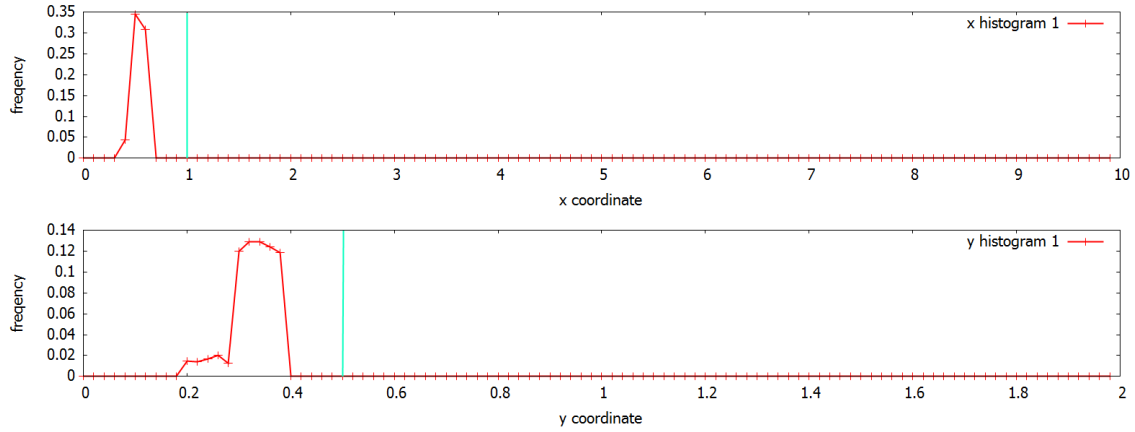



Figure 1: Finite-difference only inference results for one obstacle at $(1.0, 0.5)$ denoted by the cyan lines.

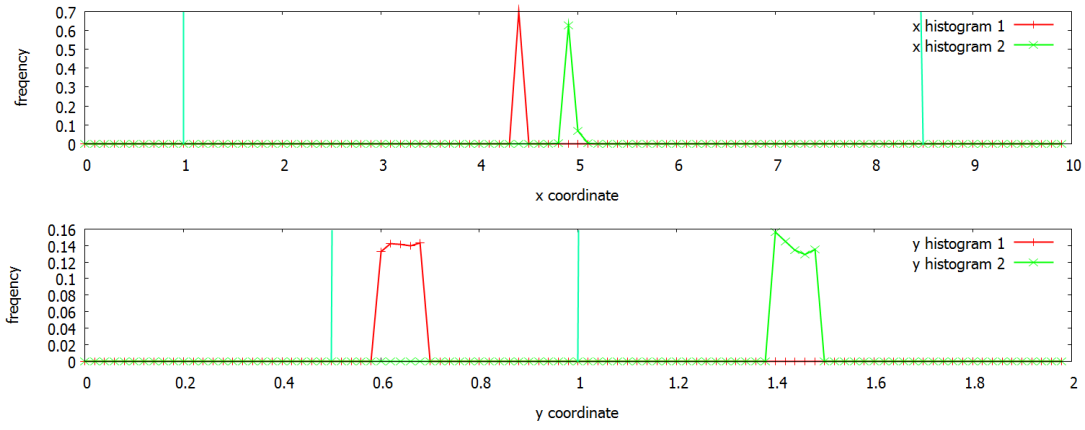



Figure 2: Finite-difference only inference results for two obstacle at $(1.0, 0.5), (8.5, 1.0)$ denoted by the cyan lines.

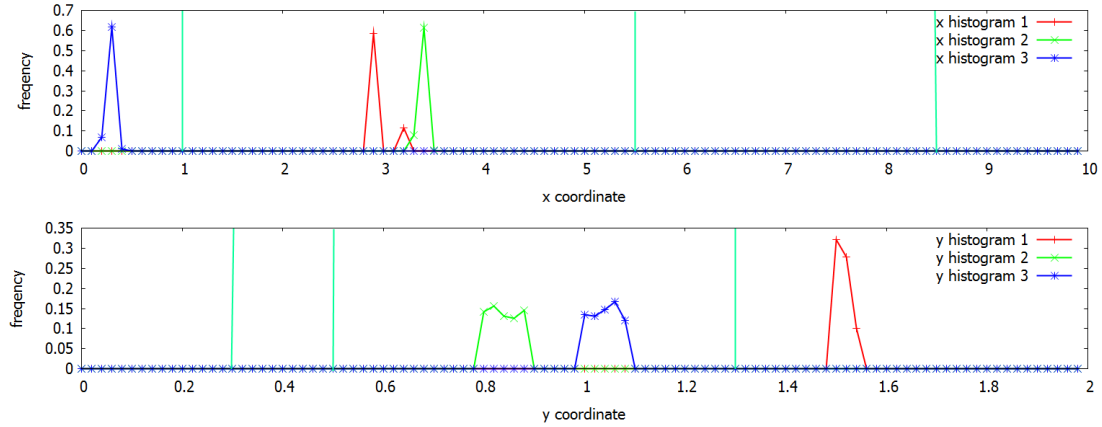



Figure 3: Finite-difference only inference results for three obstacle at $(1.0, 0.5), (8.5, 1.0), (5.5, 0.3))$ denoted by the cyan lines.

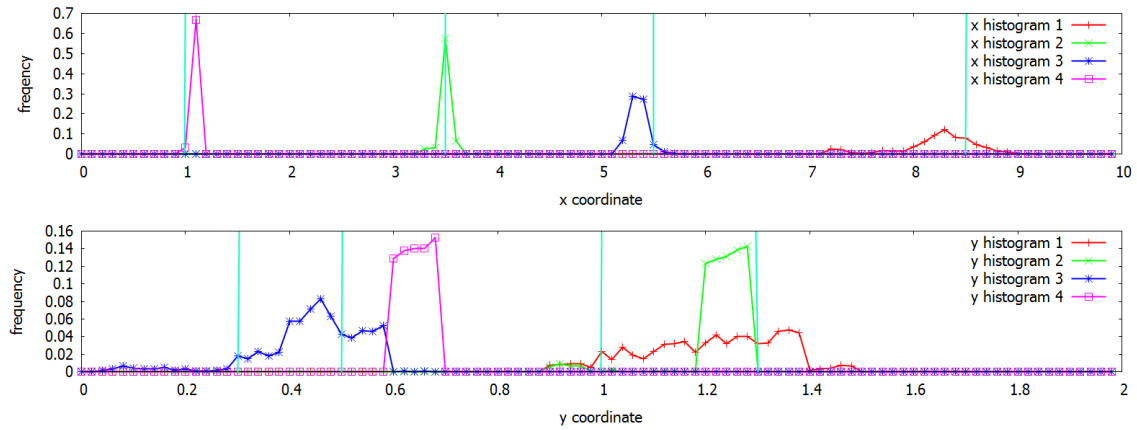



Figure 4: Finite-difference only inference results for four obstacle at $(1.0, 0.5), (8.5, 1.0), (5.5, 0.3), (3.5, 1.3)$ denoted by the cyan lines.

## 5.2 Adaptive surrogate model

The surrogate model was run using $N^{MCMC} = 20000$ MCMC steps for $1, 2, 3$ and $4$ obstacles as the evaluations of the sparse grid are computationally cheap. No burn-in was used in this case, because a suitable starting point for the MCMC is attained during the sparse grid creation.
In figures 5,6,7 and 8 we present the surrogate model results for up to four obstacles. Once again, some error is unavoidable due to the reasons stated in the previous subsection.

In the case of one obstacle, the algorithm started on level 7 (769 sparse grid points) and ran up to level 11 (20491 sparse grid points), where the domain was refined to $\Omega_\theta^{11} = [0.71, 1.29] \times [0.45, 0.56]$. Note the higher accuracy in the result and the vastly reduced domain and relatively small number of finite-difference model evaluations necessary to get there. For two obstacles we also started on level 7 (7937 sparse grid points) and went up to level 11 (471041 sparse grid points) resulting in the domain $\Omega_\theta^{11} = [0.67, 1.25] \times [0.52, 0.64] \times [8.18, 8.75] \times [0.99, 1.11]$. We attain higher accuracy again, however looking at the final domain, one of the main problems of this algorithm arises, as we have refined our domain to a point, where it does not include the real obstacle location of the first obstacle anymore. Therefore convergence would halt here. Also, the number of sparse grid points is significantly higher already, but due to the parallelism of the algorithm this does not present a problem at this point. The three obstacle case was performed on starting level 7 (40193 sparse grid points) and ran up to level 9 (471041 sparse grid points). The domain was refined to $\Omega_\theta^9 = [3.22, 8.12] \times [0.05, 0.91] \times [4.81, 9.55] \times [0.68, 1.55] \times [0.05, 3.05] \times [0.05, 0.75]$. Note the vastly increased accuracy in this case attained using a similar number of sparse grid points as in the two obstacle case, but on a bigger domain. Furthermore all obstacles are still in our refined domain at our final step indicating a potential to furtherly increase the accuracy.
The final test case was the four obstacle case, where we again chose starting level 7 (141569 sparse grid points) and went up to level 9 (2228225 sparse grid points) resulting in the domain $\Omega_\theta^9 = [0.05, 2.97] \times [0.36, 1.34] \times [4.39, 9.29] \times [0.05, 1.03] \times [5.83, 9.55] \times [0.05, 1.03] \times [0.70, 5.60] \times [0.22, 1.20]$. The accuracy in this case was unfortunately reduced which is probably caused by the fact, that not all obstacles lie in the refined domain and the MCMC solver's convergence is therefore hindered. A potential fix for this problem might be an adaptively chosen refinement parameter $\kappa$ or the use of a surrogate model that refines the domain but still allows the MCMC solver to run on the whole domain.

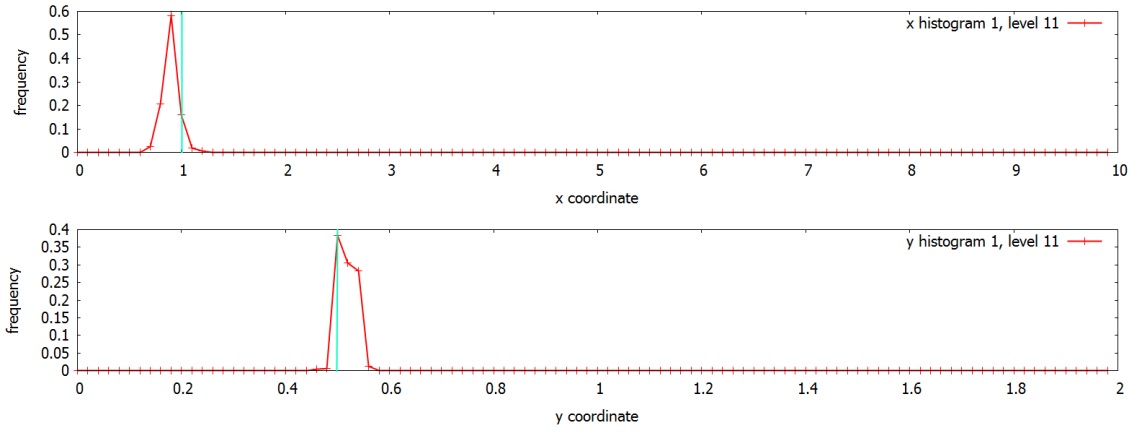



Figure 5: Adaptive Surrogate model inference results for one obstacle at $(1.0, 0.5)$ denoted by the cyan lines.

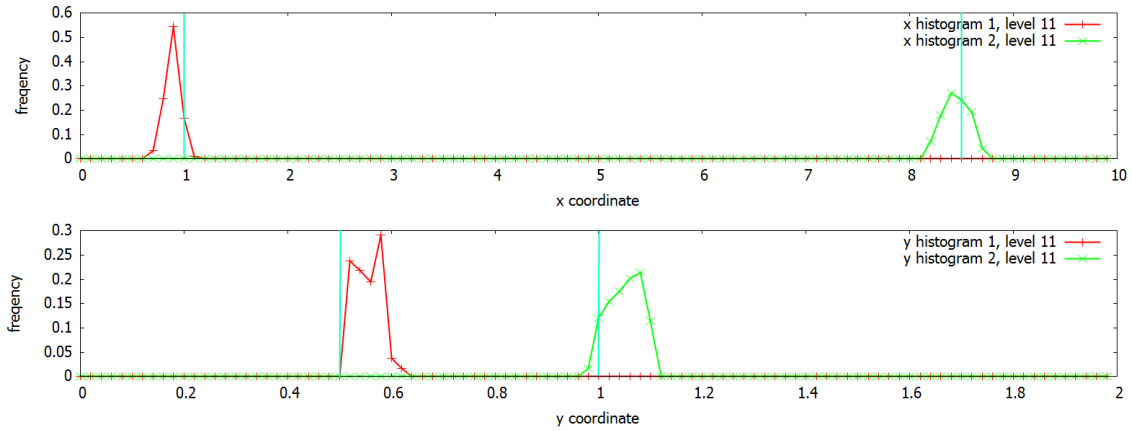



Figure 6: Adaptive Surrogate model inference results for two obstacles at $(1.0, 0.5), (8.5, 1.0)$ denoted by the cyan lines.

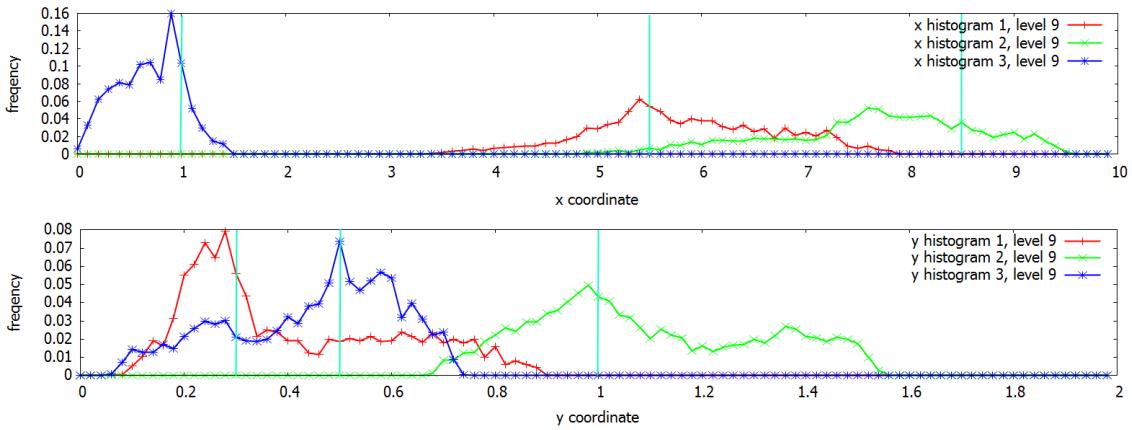



Figure 7: Adaptive Surrogate model inference results for three obstacles at $(1.0, 0.5), (8.5, 1.0), (5.5, 0.3)$ denoted by the cyan lines.

Overall our algorithm results in a higher accuracy and presents us with the option to choose a compromise of increased performance and higher accuracy. Note however, that especially after many iterations it can happen that true locations are outside of our refined domain, which will require further research.

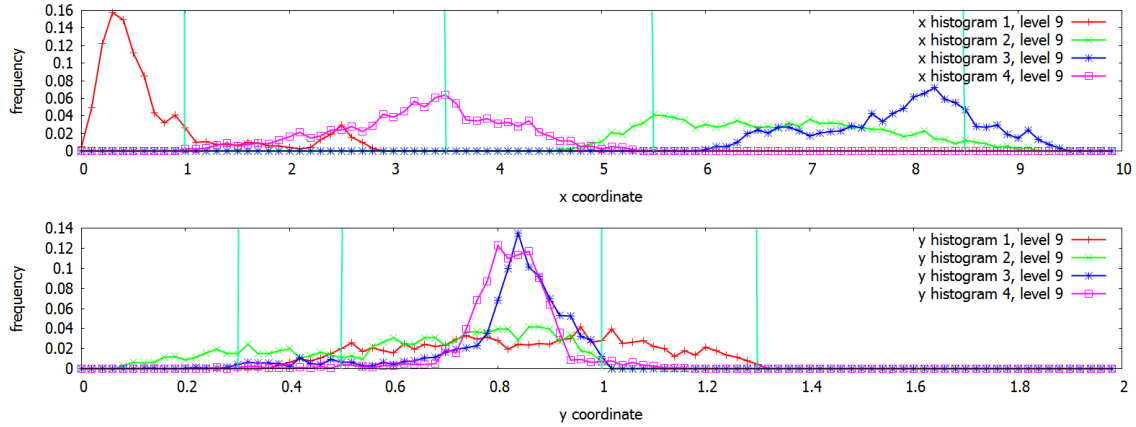



Figure 8: Adaptive Surrogate model inference results for four obstacles at $(1.0, 0.5), (8.5, 1.0), (5.5, 0.3), (3.5, 1.3)$ denoted by the cyan lines.

## 6 Conclusion and Outlook

In this paper we presented an adaptive surrogate model for an inverse Navier-Stokes problem using sparse grids. Our approach can be tailored to either increase the accuracy of the surrogate model used in [8] or speed up the necessary computations. We provide detailed results regarding the increased accuracy attained using this approach. However it can be seen that the achieved accuracy is still limited in the case of $4$ obstacles, which is probably due to the increased dimensionality of the problem, which slows convergence of the MCMC solver.

To sum up, the approach shown in this paper can be used to achieve a high accuracy in this problem setting. However, there is room for improvement left in future research. One potential avenue to pursue might be the use of a more sophisticated MCMC solver than the Metropolis-Hastings algorithm, which is somewhat outdated and there is research into parallel MCMC approaches too, e.g. [11].

Another way of improvement might lie in reusing the lower level surrogate models to improve the accuracy of the higher level ones and to avoid problems arising, if the predicted obstacle locations are outside of the new, smaller domains.

Finally, a thorough optimization of our code might allow us to make better use of the parallelism to improve overall performance and thereby also potential accuracy. It should also be relatively easy to test the approach on different problem settings. In particular it would be applicable to linear problems too. Future work will show, if it is possible to cope with its current dependency on careful calibration while maintaining the great accuracy gain.

# Bibliography


[1] Philipp Atanas Atanasov Christoph Kowitz Neumann. Praktikum wissenschaftliches rechnen: Computational fluid dynamics. 2012.

[2] P. Chen and Ch. Schwab. Adaptive sparse grid model order reduction for fast bayesian estimation and inversion. Technical Report 2015-08, Seminar for Applied Mathematics, ETH Zürich, Switzerland, 2015.

[3] S.L. Cotter, M. Dashti, J.C. Robinson, and A.M Stuart. Bayesian inverse problems for functions and applications to fluid mechanics. *Inverse Problems*, 25:115008, 2009.

[4] C. Foias, O. Manley, R. Rosa, and R. Temam. *Navier-Stokes Equations and Turbulence*, page 4f. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2001.

[5] J. Garcke. Sparse grids in a nutshell. In J. Garcke and M. Griebel, editors, *Sparse grids and applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 57–80. Springer, 2013. extended version with python code `http://garcke.ins.uni-bonn.de/research/pub/sparse_grids_nutshell_code.pdf`.

[6] Alexander Heinecke, Stefanie Schraufstetter, and Hans-Joachim Bungartz. A highly parallel black-scholes solver based on adaptive sparse grids. *Int. J. Comput. Math.*, 89(9):1212–1238, June 2012.

[7] Xiang Ma and Nicholas Zabaras. An efficient bayesian inference approach to inverse problems based on an adaptive sparse grid collocation method. *Inverse Problems*, 25(3), 2009.

[8] Ao Mo-Hellenbrand. Sparse grid interpolants as surrogate models in statistical inverse problems. Master's thesis, Institut für Informatik, Technische Universität München, September 2013.

[9] Wolfgang Polasek. Handbook of markov chain monte carlo edited by steve brooks, andrew gelman, galin jones, xiao-li meng. *International Statistical Review*, 80(1):184–185, 2012.

[10] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*, pages 1–40. SIAM, 2005.

[11] D. N. VanDerwerken and S. C. Schmidler. Parallel Markov Chain Monte Carlo. *ArXiv e-prints*, December 2013.