Model Reduction for Dynamic Sensor Steering: A Bayesian Approach to Inverse Problems

by

Sonja Wogrin

Submitted to the School of Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computation for Design and Optimization

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author .....
School of Engineering
May 16, 2008

Certified by .....
Karen E. Willcox
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Co-Director, Computation for Design and Optimization

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2008		**00-00-2008 to 00-00-2008**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Model Reduction for Dynamic Sensor Steering: A Bayesian Approach to Inverse Problems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,School of Engineering,Cambridge,MA,02139**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**101**	

Model Reduction for Dynamic Sensor Steering: A Bayesian Approach to Inverse Problems

by

Sonja Wogrin

Submitted to the School of Engineering
on May 16, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Computation for Design and Optimization

Abstract

In many settings, distributed sensors provide dynamic measurements over a specified time horizon that can be used to reconstruct information such as parameters, states or initial conditions. This estimation task can be posed formally as an inverse problem: given a model and a set of measurements, estimate the parameters of interest. We consider the specific problem of computing in real-time the prediction of a contamination event, based on measurements obtained by mobile sensors. The spread of the contamination is modeled by the convection diffusion equation. A Bayesian approach to the inverse problem yields an estimate of the probability density function of the initial contaminant concentration, which can then be propagated through the forward model to determine the predicted contaminant field at some future time and its associated uncertainty distribution. Sensor steering is effected by formulating and solving an optimization problem that seeks the sensor locations that minimize the uncertainty in this prediction.

An important aspect of this Dynamic Sensor Steering Algorithm is the ability to execute in real-time. We achieve this through reduced-order modeling, which (for our two-dimensional examples) yields models that can be solved two orders of magnitude faster than the original system, but only incur average relative errors of magnitude $\mathcal{O}(10^{-3})$. The methodology is demonstrated on the contaminant transport problem, but is applicable to a broad class of problems where we wish to observe certain phenomena whose location or features are not known *a priori*.

Thesis Supervisor: Karen E. Willcox
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

First of all, I would like to thank my advisor Professor Karen E. Willcox for giving me the chance to work with her and for her constant guidance and support whether in Cambridge or on Mont Blanc. My sincere gratitude goes out to Professor Alfio Borzi who encouraged me to apply to MIT. I also want to thank Professor Peter Grabner and Professor Hansjörg Albrecher for their great letters of recommendation.

Special thanks to our CDO Administrator Laura Koller, to Omar Bashir for letting me continue his work on the contaminant transport problem, and to my fellow ACDLers Garrett Barter and Chad Lieberman for answering all my Linux and vocabulary questions.

I also want to thank all my friends at MIT who have made me feel at home in the United States - in particular David Galbally Herrero for being my finite element guru, for never paying attention to my Austrian lessons, for the best birthday surprise ever and for being a monkey with zebra aspirations. To my cantik Rhea Patricia Liem I can only say: terima kasih for launching rockets together, for our late-night rehearsals in the GTL, for sharing the same chocolate addiction and for making all the hard work so much fun.

Finally I would like to express my gratitude to my family: to my grandma for passing on her Tyrolean stubbornness to me; to my father, my biggest role model, who always listened to me when my sensors were not going where they were supposed to go; to my mother for never losing interest in my life even though we were so far apart; and especially to my sister Melanie for believing in me like nobody else does.

This research was supported by the NSF under DDDAS grant CNS-0540186 (program director Dr. Frederica Darema) and the Air Force Office of Scientific Research (program manager Dr. Fariba Fahroo).

Contents

1 Introduction **17**

1.1 Motivation 17

1.2 Literature Review 18

1.3 Thesis Objectives 20

1.4 Thesis Outline 21

2 Dynamic Sensor Steering Algorithm - Methodology **23**

2.1 Problem Description 23

2.1.1 Introductory Definitions 23

2.1.2 Contaminant Transport Example 25

2.1.3 The Algorithm 25

2.2 Model Reduction Framework 28

2.2.1 Significance for Dynamic Sensor Steering Algorithm 28

2.2.2 Model Reduction Methodology 30

2.3 Bayesian Inverse Approach 34

2.3.1 Bayes Theorem 35

2.3.2 Solution of the Inverse Problem 35

2.3.3 Gaussian Case 37

2.4 Optimization Problem 37

2.5 Implementation Remarks 40

3 Dynamic Sensor Steering Algorithm - Application and Results **43**

3.1 Description of Rectangular Domain 43

3.2 Reduced-Order Model Performance in Forward Problem 44
3.3 Solution of the Inverse Problem 46
3.4 Reduced-Order Model Performance in Optimization Problem 50
3.4.1 Unconstrained Optimization 50
3.4.2 Constrained Optimization 52
3.4.3 Optimality Conditions 54
3.5 Moving Window Illustration 56

4 Model Order Reduction of Convection-Diffusion Equation with Parameterized Velocity Field 61
4.1 Problem Description 61
4.1.1 Weak Form and Finite Element Method 62
4.2 Dynamic Sensor Steering Algorithm with Parameterized Velocity Field 65
4.2.1 Offline Stage of the Dynamic Sensor Steering Algorithm with Parameterized Velocity Field 65
4.2.2 Online Stage of the Dynamic Sensor Steering Algorithm with Parameterized Velocity Field 66
4.3 Model Order Reduction Methodology in the Linear Case 67

5 Application to Convection-Diffusion Systems with Parameterized Velocity Fields 73
5.1 Dynamic Sensor Steering over Rectangular Domain 73
5.1.1 Description of Rectangular Domain 73
5.1.2 A Priori Velocity Fields over Rectangular Domain 74
5.1.3 Moving Window Illustration 75
5.2 Dynamic Sensor Steering over Backward Facing Step 77
5.2.1 Description of Backward Facing Step Domain 77
5.2.2 A Priori Velocity Fields over Backward Facing Step Domain . 78
5.2.3 Reduced-Order Model Performance in Forward Problem . . . 81
5.2.4 Solution of the Inverse Problem 82
5.2.5 Reduced-Order Model Performance in Optimization Problem . 83

5.2.6 Moving Window Illustration 86

6 Conclusions and Future Work **91**

6.1 Summary and Conclusions 91

6.2 Future Work 93

6.2.1 Recommendations 93

6.2.2 Model Order Reduction Methodology in the Nonlinear Case . 93

List of Figures

2-1 Offline stage of the Dynamic Sensor Steering Algorithm. 26

2-2 One cycle of the online stage of the Dynamic Sensor Steering Algorithm. 29

2-3 Trade-off between full and reduced-order model in one online cycle. . 31

2-4 Comparison between inverse and forward problem. 34

3-1 Triangular mesh of domain Ω. 44

3-2 This figure depicts the sample test initial condition used to compare reduced model to full model in the forward problem. 45

3-3 A comparison between full ($N = 4005$) and reduced outputs ($n = 212$) for sensors S_1 and S_2 using initial condition depicted in Figure 3-2 with $\kappa = 0.01$, $\Delta t = 0.005$ and hundred time steps. The error is $\varepsilon = 1.0232e^{-4}$. 47

3-4 The upper plot shows the increase in computational time in seconds of the forward problem as the size of the reduced-order model grows over the rectangular domain. The lower plot depicts the decrease of the error defined in (3.5) as the size of the reduced-order model increases. 48

3-5 This plot depicts the true initial contaminant concentration u_0 used in the inverse problem. 49

3-6 The upper plot shows the full model estimate $\hat{u}_0^{full}$ of the initial condition field. The lower plot shows the reduced model estimate $\hat{u}_0^{red}$ of the initial condition field with $n = 212$ which yields $\varepsilon_{u_0} = 0.0062$. . . 50

3-7 Marginal probability density $\sigma_{u_0^i}(u_0^i)$ with mean $\mu = 0.2105$ and with varying standard deviation σ. Note that i corresponds to grid point (0.1477,0.2105) in Ω. 51

3-8 The upper plot depicts the increase in computational time as the size of the reduced-order model increases. The lower plot depicts the error according to reduced-order model sizes in the inverse problem. 52

3-9 This figure contains the new sensor locations computed by the full model and several reduced-order models in the unconstrained case. . . 54

3-10 This figure contains the new sensor locations computed by the full model and several reduced-order models in the constrained case. . . . 55

3-11 The upper figure depicts the contaminant concentration at $t_f = 0.2$, current and optimal sensor locations of cycle one of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω. . 57

3-12 The upper figure depicts contaminant concentration at $t_f = 0.4$, current and optimal sensor locations of cycle two of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω. . 58

3-13 The upper figure depicts the contaminant concentration at $t_f = 0.6$, current and optimal sensor locations of cycle three of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω. . 59

3-14 The upper figure depicts the contaminant concentration at $t_f = 0.8$, current and optimal sensor locations of cycle four of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω. . 60

4-1 Offline stage of the Dynamic Sensor Steering Algorithm with parameterized velocity field. 66

4-2 One cycle of the online stage of the Dynamic Sensor Steering Algorithm with parameterized velocity field. 71

5-1 Contaminant concentration at $t_f = 0.2$, current and optimal sensor locations of cycle one of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0.75, 0.25, 0, 0, 0, 0, 0, 0\}$. 76

5-2 Contaminant concentration at $t_f = 0.4$, current and optimal sensor locations of cycle two of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0.25, 0.10, 0.25, 0, 0.15, 0, 0, 0.25\}$. 77

5-3 Contaminant concentration at $t_f = 0.6$, current and optimal sensor locations of cycle three of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0, 0, 0.25, 0, 0.75, 0, 0, 0\}$. 78

5-4 Triangular mesh of the backward facing step domain. Note that the mesh used for computations is much finer. 79

5-5 The upper plot depicts the streamlines over the backward facing step domain computed using a viscosity $\nu = 1/50$. The lower plot shows the pressure field over the backward facing step domain computed using a viscosity $\nu = 1/50$. 80

5-6 Velocity at selected grid points of the backward facing step domain computed using a viscosity $\nu = 1/50$. 81

5-7 Sample test initial condition used to compare reduced model to full model in the forward problem over backward facing step domain. . . 82

5-8 A comparison between full ($N = 2417$) and reduced outputs ($n = 185$) for sensors S_1 and S_2 using initial condition depicted in Figure 5-7 with $\kappa = 0.01$, $\Delta t = 0.0025$ and 100 time steps. The error is $\varepsilon = 8.5419e^{-4}$. 83

5-9 The upper plot shows the increase in computational time in seconds of the forward problem as the size of the reduced-order model grows over the backward facing step domain. The lower plot depicts the decrease of the error defined in (3.5) as the size of the reduced-order model increases. 84

5-10 The upper plot shows the full model estimate $\hat{u}_0^{full}$ of the initial condition field over the backward facing step domain. The lower plot shows the reduced model estimate $\hat{u}_0^{red}$ of the initial condition field with $n = 185$ which yields $\varepsilon_{u_0} = 0.4025$. 86

5-11 The upper figure depicts the contaminant concentration at $t_f = 0.2$, current and optimal sensor locations of cycle one of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{1, 0, 0, 0\}$. The lower figure shows the corresponding variance field based on the current optimal sensor locations over Ω. 88

5-12 The upper figure depicts the contaminant concentration at $t_f = 0.4$, current and optimal sensor locations of cycle two of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0, 0, 1, 0\}$. The lower figure shows the corresponding variance field based on the current optimal sensor locations over Ω. 89

5-13 The upper figure depicts the contaminant concentration at $t_f = 0.6$, current and optimal sensor locations of cycle three of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0, 0, 0, 1\}$. The lower figure shows the corresponding variance field based on the current optimal sensor locations over Ω. 90

List of Tables

3.1 Properties of various reduced-order models of a full-scale system with size $N = 4005$ and two output sensors in the forward problem. The error ε is defined in (3.5) and the initial condition used is depicted in Figure 3-2. 46

3.2 Properties of various reduced-order models of a full-scale system with size $N = 4005$ and two output sensors in the inverse problem. 53

3.3 Results of various reduced-order models in the unconstrained optimization problem. 53

3.4 Results of various reduced-order models in the constrained optimization problem. 55

5.1 Properties of various reduced-order models of a full-scale system with size $N = 2417$ and two output sensors over the backward facing step domain in the forward problem. The error ε is defined in (3.5) and the initial condition used is depicted in Figure 5-7. 85

5.2 Properties of various reduced-order models of a full-scale system with size $N = 2417$ and two output sensors in the inverse problem. 85

5.3 Results of various reduced-order models in the unconstrained optimization problem over the backward-facing step. 87

Chapter 1

Introduction

1.1 Motivation

In numerous fields of science and engineering there is a strong demand for precise observation of large-scale physical systems, modeled by partial differential equations, in order to detect phenomena of interest that can not be foreseen and to compute accurate predictions regarding those phenomena. For example, let us observe the amount of contamination in the air of the Boston city area as measured by mobile sensors and modeled by the convection diffusion equation. Imagine the following scenario: due to a terrorist attack, a highly poisonous contaminant is set free, endangering the life of many civilians. This situation calls for immediate results; counteractions, like evacuations or the deployment of the fire department, have to take place as soon as possible. It becomes apparent that the time available to come up with an accurate prediction of the ongoing contamination is extremely limited. Even though the results computed by a high fidelity, large-scale model are the most accurate for this application, the long computational time required by those models (hours or even days) makes them inappropriate under these circumstances. Furthermore, as the obtained prediction depends greatly on the location of sensors, we need a way to find the best sensor locations for this purpose.

The discipline of model order reduction studies properties of large-scale dynamical systems and reduces their complexity while maintaining their input-output behavior.

The reduced-order model is computationally much less expensive than the full model and thus yields real-time solutions for a negligible trade-off in accuracy. The Bayesian approach to inverse problems is based on observed data, which allow one to estimate unknown model parameters. These two techniques will be key ingredients for tackling the problem stated above. The remaining challenge is to develop a methodology that combines these two disciplines. This will enable accurate real-time predictions, which are optimal for the observed data in systems that change quickly, and to handle uncertainty in measurements.

1.2 Literature Review

Model order reduction is a technique that observes properties of large-scale dynamical systems and reduces their complexity while maintaining their input-output behavior. The majority of methods obtains the reduced-order model by projecting the full model onto a basis that spans a space of lower dimension. This reduced basis can be computed using proper orthogonal decomposition [29, 47], Krylov-subspace methods [18, 21, 26], reduced basis methods [42], balanced truncation [36] or a Hessian-based model reduction approach [4].

The Bayesian approach to inverse problems is thoroughly discussed in the literature, e.g. [30, 50] and deals with estimating unknown model parameters based on observed data. The solution of the inverse problem is represented by a probability density as oppose to just a single estimate of the model parameters as in the deterministic approach. In the Bayesian approach, model as well as measurement uncertainties can be incorporated into the solution very elegantly.

In [34, 35] an approach is proposed for the description of atmospheric flows based on proper orthogonal decomposition. In this thesis we assume different velocity fields and focus on the contaminant transport problem. Combining the work that was presented in [34, 35] and this thesis can yield even more efficient and more realistic results.

Optimal sensor placement is an issue that has been widely addressed in literature.

This discipline holds many challenges for several different applications. The most relevant papers, for the purpose of this thesis, shall be discussed now.

Both, [6] and [44], address the problem of placing sensors in water networks to detect maliciously injected contaminants, which has a similar motivation as our problem. [6] formulates a linear mixed-integer problem that minimizes the fraction of the population at risk and assumes a finite number of sensor locations. While efficiently solvable this approach, as many others, only considers a finite number of sensor locations. [44] focuses on finding a layout of early warning detection stations based on an optimization problem comprising water quality conditions and extended period unsteady hydraulics. The main limitation of this methodology is the real-time assumption and again the finite number of possible sensor locations.

An autonomous model-based reactive observing systems has been developed in [11] with a set of static and mobile sensors. The focus lies on a sample selection problem modeled by a subset selection problem for regression. They assume the physical phenomena that they wish to observe, do not change temporally (or change very slowly). In the contaminant transport setting on the other hand we have to be prepared for fast changing phenomena. In [11] the mobile sensors are steered by an algorithm based on local linear regression as oppose to our eigenvalue-based optimization problem. They distinguish between three different kinds of sensor faults and describe methods to detect those faults whereas we propose to incorporate measurement errors as uncertainties when solving the inverse problem.

In [24] a sensor placement strategy to obtain the most effective visual sensing of an area of interest is proposed and solved by solving a variant of the art-gallery problem [43]. This problem is NP-hard and once more the optimal solution is chosen among a finite number of possibilities. The issue of reducing uncertainty over a region of interest has been addressed by [13] and [27]. In [27] this is approached by an algorithm that maximizes mutual information in Gaussian processes and [13] presents an algorithm for an observation targeting problem formulated as a sensor selection problem.

In order to obtain the best prediction we have to find the Bayesian optimal exper-

imental design by solving an optimization problem aiming at minimizing uncertainty in the prediction based on an approximation of the variance-covariance matrix. The most common design criteria focus on minimizing the trace of this matrix (this is referred to as *A optimality*), other approaches [5, 38] involve the determinant or the largest eigenvalue and are known as *D* and *E optimal* designs [2]. The work of [1, 12, 28] involves *A optimal* design. In this thesis we focus on an *E optimal* approach.

Many of the optimal sensor placement approaches mentioned above compute an optimal solution from a finite number of sensor locations. In this thesis we rather choose to formulate the task as a continuous optimization problem, where the possible sensor locations are assumed to be continuous and the sensors are represented using mollified delta functions. This formulation has the advantage that it can be solved efficiently using a gradient-based algorithm, enabling the possibility of real-time computations.

1.3 Thesis Objectives

The main goal of this thesis is to develop a Dynamic Sensor Steering Algorithm, divided into an offline and online stage, that operates in real-time in order to observe quickly changing phenomena in physical processes. The specific objectives to reach this goal are the following:

1. As the physical processes are modeled by partial differential equations and lead to large-scale systems, which are too computationally expensive to be solved in real-time, we propose an algorithm that builds a reduced-order model only once in the offline stage and repeatedly runs through the online stage using only the reduced-order model.

2. We solve a Bayesian formulation of the ill-posed inverse problem to account for model and measurement uncertainties.

3. We formulate and solve an optimization problem based on the largest eigen-

value of the covariance matrix, yielding optimal sensor locations to minimize uncertainty in the prediction over the output region of interest.

4. We demonstrate the newly proposed methodology by applying it to a contaminant transport problem modeled by the convection diffusion equation and extending the methodology to be able to handle a parameterized velocity field.

1.4 Thesis Outline

In Chapter 2 we develop the methodology for a Dynamic Sensor Steering Algorithm based on a characterization of prediction uncertainty, computed using a Bayesian formulation of the inverse problem and the application of model order reduction. Each component of the approach, i.e. the Bayesian framework, the model order reduction and the optimization are discussed in detail.

In Chapter 3 the methodology developed in Chapter 2 is applied to a contaminant transport problem modeled by the convection diffusion equation. We also include a thorough comparison between full and reduced-order model performance in forward and inverse problem as well as in the optimization.

We introduce a parameterized velocity field in the Dynamic Sensor Steering Algorithm setting in Chapter 4, which enables a more realistic simulation of physical processes. The posed challenge is to maintain the algorithm's computational real-time property, by building reduced-order models that are not only dependent on the initial condition but also on the velocity field. The results are presented in Chapter 5 where we employ a Navier-Stokes solver to pre-compute the velocity fields.

Finally, Chapter 6 concludes the thesis with a summary and suggestions for future work.

Chapter 2

Dynamic Sensor Steering Algorithm - Methodology

This chapter provides a description of the methodology used in the newly developed Dynamic Sensor Steering Algorithm, which is based on a characterization of prediction uncertainty, computed using a Bayesian formulation of the inverse problem and the application of model order reduction. In Section 2.1 the algorithm's general work flow, its steps and its functionality are presented, followed by a detailed discussion in Section 2.2 on how a reduced-order model is incorporated to obtain an accurate predictive capability that operates in real-time. We will then analyze the Bayesian framework and the optimization problem employed, in Section 2.3 and Section 2.4 respectively.

2.1 Problem Description

2.1.1 Introductory Definitions

Let the two or three-dimensional domain that we wish to observe be named Ω. In order to obtain measurements of an ongoing physical process, Q *mobile* sensors are placed into this domain. For the purpose of this general discussion we do not assume any knowledge about Ω. Constraints describing the sensors' mobility have to be

adapted according to each individual problem setting. Therefore let us collect the corresponding sensor-related information in the so-called *sensor constraints*. One approach to defining these constraints is described in Section 2.4. Let the sensor locations be denoted as $S_1^i, \ldots, S_Q^i$ *for i=1,2,3...*, where the superscript refers to a time index and the subscript corresponds to the number of the sensor. Furthermore let T_{steer} denote the available steering time for the mobile sensors. We do not assume anything about the initial sensor placement except that they have to be within our domain Ω. Within this domain there are some specific regions that are of greater interest than others. It may be temporary interest because sensors detect some curious unexpected phenomenon or permanent interest due to additional knowledge on the structure of Ω, e.g. residential areas. The union of those regions will be referred to as the output region of interest and be denoted as Ω_I. Let $\boldsymbol{g}(\cdot)$ be the forward operator used to model the natural processes we want to observe e.g. the convection-diffusion equations, let $\boldsymbol{m}$ be the input vector for the model $\boldsymbol{g}$, and let $\boldsymbol{d}$ denote the vector containing the outputs of interest. Then this can be written as follows

$$\boldsymbol{d} = \boldsymbol{g}(\boldsymbol{m}). \tag{2.1}$$

In the case that $\boldsymbol{g}$ is a linear operator, equation (2.1) can be written as:

$$\boldsymbol{d} = \boldsymbol{G}\boldsymbol{m}. \tag{2.2}$$

In Section 2.2, we will present a model reduction framework. For now, we define the following notation. Let $\boldsymbol{g}_r(\cdot)$ denote the reduced-order model for $\boldsymbol{g}(\cdot)$ and $\boldsymbol{G}_r$ be the reduced-order model for $\boldsymbol{G}$, then we obtain the two reduced systems (2.3)–(2.4) that correspond to (2.1) and (2.2), respectively. The goal of model order reduction is to determine a reduced model so that $\boldsymbol{d} \approx \boldsymbol{d}_r$ while achieving a significant reduction in the computational cost.

$$\boldsymbol{d}_r = \boldsymbol{g}_r(\boldsymbol{m}), \tag{2.3}$$

$$\boldsymbol{d}_r = \boldsymbol{G}_r\boldsymbol{m} \tag{2.4}$$

2.1.2 Contaminant Transport Example

To demonstrate the results of the Dynamic Sensor Steering Algorithm we consider a two-dimensional contaminant transport problem, modeled by the convection-diffusion equation,

$$\begin{aligned}
\frac{\partial \mathfrak{u}}{\partial t} + \boldsymbol{v} \cdot \nabla \mathfrak{u} - \kappa \nabla^2 \mathfrak{u} &= 0 \quad \text{in } \Omega \times (0, t_f), && (2.5)\\
\mathfrak{u} &= 0 \quad \text{on } \Gamma_D \times (0, t_f), && (2.6)\\
\frac{\partial \mathfrak{u}}{\partial n} &= 0 \quad \text{on } \Gamma_N \times (0, t_f), && (2.7)\\
\mathfrak{u} &= \mathfrak{u}_0 \quad \text{in } \Omega \text{ for } t = 0, && (2.8)
\end{aligned}$$

where $\mathfrak{u}$ denotes the contaminant concentration, $\boldsymbol{v}$ denotes the velocity field, t_f denotes the observed time horizon and κ is the diffusivity constant. We impose homogeneous Dirichlet boundary conditions on the Dirichlet boundary $\Gamma_D \times (0, t_f)$ and Neumann boundary conditions on the Neumann boundary $\Gamma_N \times (0, t_f)$. We perform the time discretization using the Backward Euler method and a Streamline Upwind Petrov-Galerkin [8] finite-element method using triangular elements for the discretization in space to obtain the discrete-time system

$$\begin{aligned}
D_1\ u(k+1) &= D_2\ u(k), \quad k = 0, 1, \ldots, T-1 && (2.9)\\
d(k) &= C\ u(k), \quad k = 0, 1, \ldots, T && (2.10)\\
u(0) &= u_0, && (2.11)
\end{aligned}$$

where $u(k) \in \mathbb{R}^N$ represents the state at time t_k, $d(k) \in \mathbb{R}^Q$ is the output at time t_k and the initial condition u_0 is the state at time $t = 0$ and the matrices $D_1 \in \mathbb{R}^{N \times N}$, $D_2 \in \mathbb{R}^{N \times N}$, $C \in \mathbb{R}^{Q \times N}$.

2.1.3 The Algorithm

The Dynamic Sensor Steering Algorithm focuses on computing the best-possible prediction of an ongoing physical process in the output region of interest Ω_I in real-time

based on sensor measurements. In order to improve the amount of information obtained from the measurements and thereby increase the accuracy of the prediction, we solve an optimization problem to determine new sensor locations that lead to minimized uncertainty in the prediction. The algorithm is made up of two separate stages, the offline stage, which we execute only once, and the online stage, which consists of a measure-predict-optimize-steer cycle. This online cycle is repeatedly executed to improve the accuracy of the obtained prediction, while incorporating new measurement information as time progresses. The formation of a reduced-order model in the offline stage enables the successful application of the Dynamic Sensor Steering Algorithm to problems that require real-time processing. The online stage consists of five major steps computing a current prediction, solving an optimization problem on the sensor locations and steering each mobile sensor to its new and improved destination. Note that for this section the input vector $\boldsymbol{m}$ consists of an initial condition denoted by u_0 only.

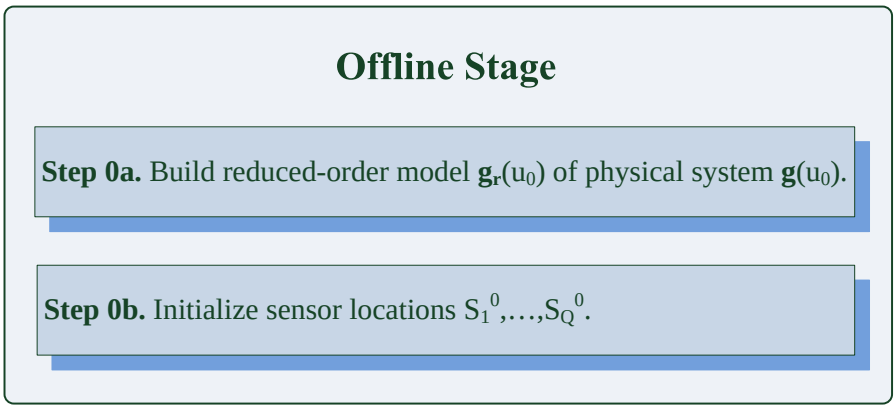

Figure 2-1: Offline stage of the Dynamic Sensor Steering Algorithm.

Offline Stage of the Dynamic Sensor Steering Algorithm

Figure 2-1 shows the offline stage of the algorithm. The details of each step are discussed in the following:

Step 0a. Build reduced-order model $\boldsymbol{g}_r(\cdot)$ of the physical system $\boldsymbol{g}(\cdot)$ and continue working with reduced system throughout the Dynamic Sensor Steering Algorithm.

Note that some well-known techniques to construct reduced-order models and the method applied in this thesis for the application to a contaminant transport problem are discussed in Section 2.2.

Step 0b. Place the Q mobile sensors at locations $S_1^0, \ldots, S_Q^0$ within Ω and activate them. We note that specification of the initial sensor locations is an important question; however, in the setting of this thesis we focus on the dynamic element, i.e. mobile sensors that are moved to optimal locations so as to minimize uncertainty in the prediction.

Online Stage of the Dynamic Sensor Steering Algorithm

Figure 2-2 shows the online stage of the algorithm. The details of each step are discussed in the following:

Step 1. The Q sensors create measurements of the environment at their current locations $S_1^{i-1}, \ldots, S_Q^{i-1}$ in *cycle i*. Let $\boldsymbol{d}^i = (d_1^i, \ldots, d_Q^i)$ be the obtained data at the Q different sensor locations in cycle i.

Step 2. Based on the data $\boldsymbol{d}^i = (d_1^i, \ldots, d_Q^i)$ and the known forward model $\boldsymbol{g}_r(\cdot)$ ($\boldsymbol{G}_r$ in the linear case), we solve the inverse problem using a Bayesian approach and compute the posterior probability density $\sigma_M(\boldsymbol{m})$. Please refer to Section 2.3 for further details on the Bayesian framework.

Step 3. In many applications some regions Ω_I within the observed domain Ω need to be watched more thoroughly than others at any time, meaning that the physical process that we want to predict interests us especially in those regions. Ω_I can be fixed, e.g. representing an urban region, or could change from cycle to cycle, e.g. if we wish to track regions of high concentration.

Step 4. We optimize the sensor locations within Ω such that the uncertainty in the prediction for the output regions of interest Ω_I is minimized. For details on how the optimization problem is set up and solved please refer to Section 2.4. As input parameters we use the current set of sensor locations $S_1^{i-1}, \ldots, S_Q^{i-1}$, a set of constraints specifying allowable sensor movements, knowledge about the domain Ω and Ω_I, and the model – either $\boldsymbol{g}$ or $\boldsymbol{g}_r$. As the output of this optimization problem

we obtain the *optimal* sensor locations $S_1^i, \ldots, S_Q^i$, whose quality will be reflected in *Step 2* of the next cycle of the online stage, where a new prediction will be produced.

Step 5. The last step in the algorithm steers the mobile sensors from their current locations $S_1^{i-1}, \ldots, S_Q^{i-1}$ to the optimized locations $S_1^i, \ldots, S_Q^i$ within T_{steer} and return to *Step 1.* Note that the choice of T_{steer} is connected to the above mentioned constraints on sensor movements. Depending on the application, there may be a trade-off between extending T_{steer}, thereby increasing the amount of reachable sensor locations and limiting the number of cycles that can be run in the time available, or rather focusing on gaining as many predictions as possible and restricting sensor movements.

2.2 Model Reduction Framework

2.2.1 Significance for Dynamic Sensor Steering Algorithm

In general, model order reduction is applied to highly complex large-scale dynamical systems to obtain a reduced-order model (ROM) whose dimensions are considerably smaller, thus leading to faster computational time while the input-output behavior remains approximately the same. In the majority of cases model reduction techniques project the large-scale systems onto a basis in a space of reduced dimension. Some well-known methods to obtain this reduced basis are proper orthogonal decomposition (POD) [14, 29, 47], approximate balanced truncation, Krylov-subspace methods [18, 21, 26] and Hessian-based model reduction [4] for initial value problems.

Let us emphasize once more the significance of model order reduction for realistic applications of the Dynamic Sensor Steering Algorithm. In order to run the online stage in real time, a ROM of the physical model has to be computed offline making sure computations remain accurate but are considerably faster. Figure 2-3 depicts the trade-off between full and reduced-order model.

After construction in the offline stage the reduced-order model is used at two points within the online stage: in *Step 2* when solving the inverse problem and in

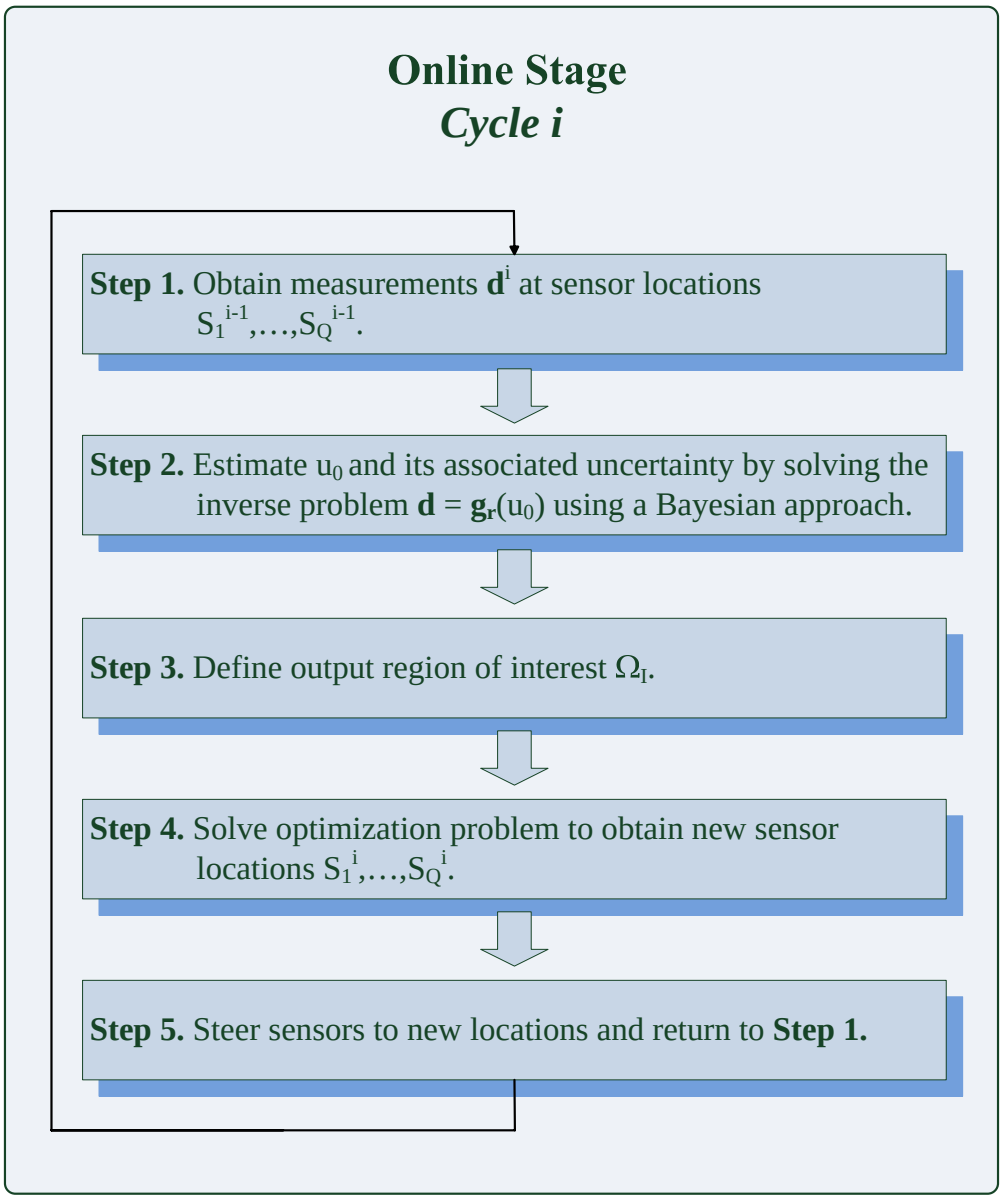

Figure 2-2: One cycle of the online stage of the Dynamic Sensor Steering Algorithm.

Step 4 when solving the optimization problem. For a detailed comparison between full and reduced-order model performance in a two-dimensional contaminant transport problem please refer to Chapter 3.

2.2.2 Model Reduction Methodology

In this section let us focus on how to build the reduced-order models intended for this thesis. Therefore let us restate the linear discrete-time system describing an initial value problem from Section 2.1.2 as

$$\begin{aligned} D_1\ u(k+1) &= D_2\ u(k), \quad k = 0, 1, \ldots, T-1 && (2.12)\\ d(k) &= C\ u(k), \quad k = 0, 1, \ldots, T && (2.13)\\ u(0) &= u_0, && (2.14) \end{aligned}$$

where $u(k) \in \mathbb{R}^N$ represents the state at a time t_k for $k = 0, \ldots, T$, $d(k) \in \mathbb{R}^Q$ is the output at time t and the initial condition u_0 is the state at time $t = 0$. We obtain $D_1, D_2 \in \mathbb{R}^{N\times N}$ and $C \in \mathbb{R}^{Q\times N}$ from spatial and temporal discretization methods. Then the system from (2.12) can be rewritten in matrix form

$$\begin{aligned} \boldsymbol{A}\ \boldsymbol{u} &= \boldsymbol{F}\ u_0, && (2.15)\\ \boldsymbol{d} &= \boldsymbol{C}\ \boldsymbol{u}, && (2.16) \end{aligned}$$

where

$$\boldsymbol{A} = \begin{bmatrix} I & 0 & \ldots & \ldots & 0 \\ -D_2 & D_1 & 0 & & \\ 0 & -D_2 & D1 & \ddots & \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & & 0 & -D_2 & D_1 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} C & 0 & \ldots & \ldots & 0 \\ 0 & C & 0 & & \\ \vdots & 0 & C & \ddots & \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & C \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad (2.17)$$

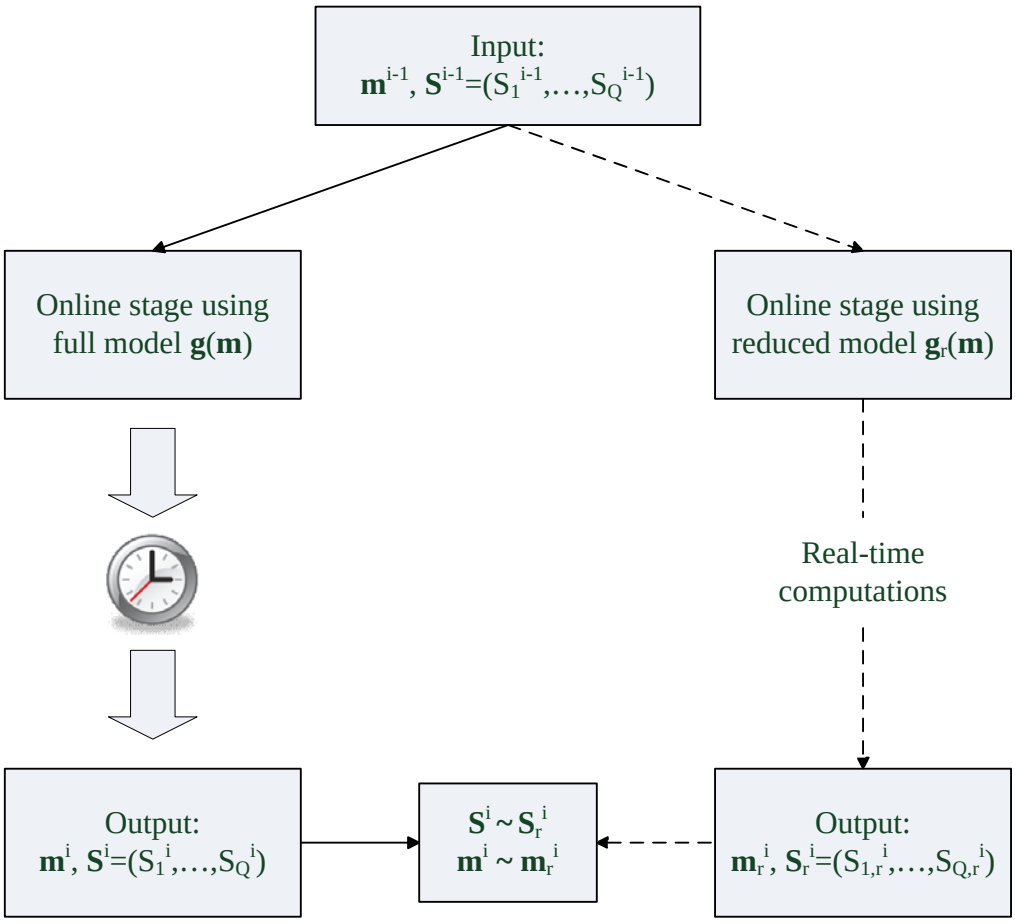

Figure 2-3: Trade-off between full and reduced-order model in one online cycle.

and the matrices' dimensions are given by $\boldsymbol{A} = \mathbb{R}^{N(T+1)\times N(T+1)}$, $\boldsymbol{F} = \mathbb{R}^{N(T+1)\times N}$ and $\boldsymbol{C} = \mathbb{R}^{Q(T+1)\times N}$, and where $\boldsymbol{u}$ and $\boldsymbol{d}$ are defined as

$$\boldsymbol{u} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(T) \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(T) \end{bmatrix}. \tag{2.18}$$

As N is usually large we need to create a reduced-order model by projecting the large-scale system onto a basis in a space of considerably reduced dimension n. Let $V \in \mathbb{R}^{N\times n}$ denote the matrix containing the n orthonormal basis vectors and $u_r(k) \in \mathbb{R}^n$ denote the reduced-order state, then $Vu_r(k)$ should provide us with a good approximation for the state $u(k)$ for $k = 0, 1, \ldots, T$. Then we apply a Galerkin projection method to the full system by projecting onto the space spanned by V and finally obtain the reduced-order model of the discrete-time system as

$$\begin{aligned} D_{1,r}\ u_r(k+1) &= D_{2,r}\ u_r(k), \quad k = 0, 1, \ldots, T-1 && (2.19) \\ d_r(k) &= C_r\ u_r(k), \quad k = 0, 1, \ldots, T && (2.20) \\ u_r(0) &= V^T\ u_0, && (2.21) \end{aligned}$$

where $D_{i,r} = V^T D_i V$ for $i = 1, 2$ and $C_r = CV$. Note that this can also be rewritten in matrix form

$$\begin{aligned} \boldsymbol{A}_r\ \boldsymbol{u}_r &= \boldsymbol{F}_r\ u_0, && (2.22) \\ \boldsymbol{d}_r &= \boldsymbol{C}_r\ \boldsymbol{u}_r, && (2.23) \end{aligned}$$

where $\boldsymbol{A}_r$, $\boldsymbol{C}_r$, $\boldsymbol{u}_r$ and $\boldsymbol{d}_r$ are defined exactly like $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{u}$ and $\boldsymbol{d}$ in (2.17) & (2.18) except for the fact that each inner component of the vectors or matrices has a subscript r now and thus we only have to exchange N to n to get the dimensions of the reduced

system. Only $\boldsymbol{F}_r \in \mathbb{R}^{n(T+1)\times N}$ is defined slightly differently, as

$$\boldsymbol{F}_r = \begin{bmatrix} V^T \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.24}$$

We use the Hessian-based model reduction approach in [4] where s seed initial conditions are computed as the dominant eigenvectors of the system's Hessian defined as

$$H = (\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{F})^T(\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{F}). \tag{2.25}$$

Then state trajectories at time steps $k = 0, 1, 2, \ldots, T$ are generated for each seed initial condition and stored in the snapshot matrix $X \in \mathbb{R}^{N\times M}$, where $M = s(T+1)$.

The technique used to efficiently compute the basis V from all the state trajectories is proper orthogonal decomposition (POD). There are several ways to compute the POD [31] but for our means we will concentrate on a method using the singular value decomposition of the snapshot matrix X. Let the singular value decomposition for the matrix of snapshots be defined as

$$X = USV^T, \tag{2.26}$$

then the following holds

$$XX^T = US^2U^T, \tag{2.27}$$

$$X^TX = VS^2V^T. \tag{2.28}$$

Due to the above equations, the singular values of X are the square roots of the eigenvalues of X^TX or XX^T and moreover the left and right singular vectors of X are in fact the eigenvectors of XX^T and X^TX. The proper orthogonal modes are the left singular vectors of X and the proper orthogonal values are defined as the

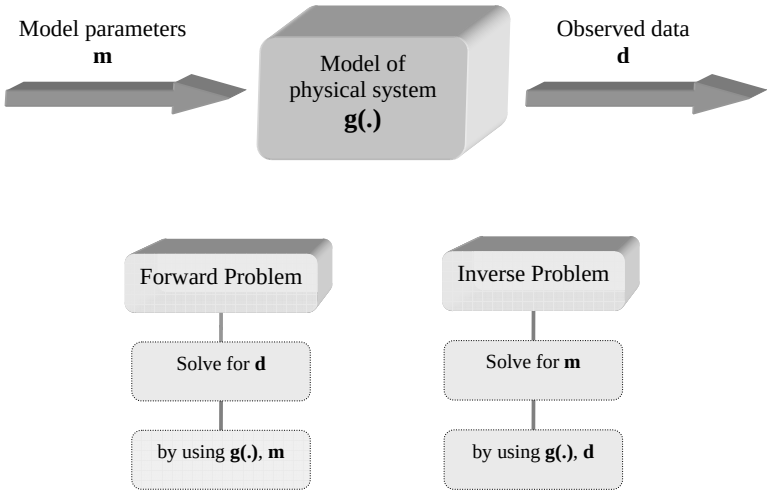

Figure 2-4: Comparison between inverse and forward problem.

corresponding singular values. As we started out to gain a basis containing n vectors, we just have to choose the dominant n left singular values of X to obtain the basis.

2.3 Bayesian Inverse Approach

Throughout this section we are following the notation introduced by [22, 50]. and thus let $\boldsymbol{g}(\cdot)$ denote the physical system we wish to observe, let $\boldsymbol{m}$ denote the model parameters and let $\boldsymbol{d}$ be the observable parameters, such that the theoretical forward problem can be written as

$$\boldsymbol{d} = \boldsymbol{g}(\boldsymbol{m}). \tag{2.29}$$

In each online cycle of the Dynamic Sensor Steering Algorithm we obtain measurements $\boldsymbol{d}^i$ that are related to model parameters $\boldsymbol{m}^i$ via the physical model $\boldsymbol{g}(\cdot)$ as described in (2.29). In order to predict $\boldsymbol{m}^i$ using the observed data $\boldsymbol{d}^i$ and the forward operator $\boldsymbol{g}(\cdot)$ an inverse problem needs to be solved. While the forward problem maps from the model parameters or the "cause" to the data, the inverse problem solves for the model parameters using the data and the knowledge of the physical model, see Figure 2-4. In practice the theoretical results for $\boldsymbol{d}$ obtained by using (2.29) will never exactly be the same as the actual observations that are made during the on-

line stage of the Dynamic Sensor Steering Algorithm due to model uncertainties and measurement errors. Thus, when reconstructing $\boldsymbol{m}$ based on $\boldsymbol{g}(\cdot)$ and $\boldsymbol{d}$, it is hard to quantify whether the obtained solution $\boldsymbol{m}$ is correct. What we are rather interested in is a way of saying that some solutions are in fact more likely than others. In order to accurately incorporate the model and measurement uncertainties and the likeliness of a solution $\boldsymbol{m}$ we introduce probability density functions to the inverse problem, leading us the Bayesian inverse approach where we compare theoretical predictions to observations. One basic tool needed in this framework is *Bayes' Theorem* relating cause to effect.

2.3.1 Bayes Theorem

Let A and B be two events, then the joint probability can be written as

$$P(A, B) = P(A \mid B)\ P(B) = P(B \mid A)\ P(A), \tag{2.30}$$

where $P(A \mid B)$ is the conditional probability of A in case event B has already happened. From this we immediately obtain Bayes' Theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}. \tag{2.31}$$

Let us now incorporate Bayes' Theorem into the Dynamic Sensor Steering setting in the following section.

2.3.2 Solution of the Inverse Problem

Let $\mathcal{M}$ denote the model space manifold and $\mathcal{D}$ denote the data space manifold. Furthermore let $\mu_M(\boldsymbol{m})$ and $\mu_D(\boldsymbol{d})$ denote the homogeneous probability density of the model space manifold and the data space manifold respectively, then their joint homogeneous probability density can be written as follows

$$\mu(\boldsymbol{d}, \boldsymbol{m}) = \mu_D(\boldsymbol{d})\ \mu_M(\boldsymbol{m}). \tag{2.32}$$

Let $\theta(\boldsymbol{d}|\boldsymbol{m})$ be the conditional probability density that provides knowledge about $\boldsymbol{d}$ when we have $\boldsymbol{m}$. This is determined by the forward operator and hence called the *likelihood function*. The joint probability density correlating $\boldsymbol{m}$ and $\boldsymbol{d}$ is defined as

$$\Theta(\boldsymbol{d}, \boldsymbol{m}) = \theta(\boldsymbol{d}|\boldsymbol{m})\ \mu_M(\boldsymbol{m}). \tag{2.33}$$

Due to the fact that there are measurement and model uncertainties let us introduce the joint *prior probability density* $\rho(\boldsymbol{d}, \boldsymbol{m})$. When solving the inverse problem we are interested in the information which will be provided by the *posterior density*. In particular, we are interested on the posterior information on the model parameters $\boldsymbol{m}$. From Bayes' Theorem we obtain the joint posterior probability density $\sigma(\boldsymbol{d}, \boldsymbol{m})$ as follows:

$$\sigma(\boldsymbol{d}, \boldsymbol{m}) = q\ \frac{\Theta(\boldsymbol{d}, \boldsymbol{m})\ \rho(\boldsymbol{d}, \boldsymbol{m})}{\mu(\boldsymbol{d}, \boldsymbol{m})}, \tag{2.34}$$

where q is a normalizing constant. From (2.34) we obtain the two marginal probability densities $\sigma_M(\boldsymbol{m})$ and $\sigma_D(\boldsymbol{d})$ as follows:

$$\sigma_M(\boldsymbol{m}) = \int_{\mathcal{M}} \sigma(\boldsymbol{d}, \boldsymbol{m})\ d\boldsymbol{m} \tag{2.35}$$

$$\sigma_D(\boldsymbol{d}) = \int_{\mathcal{D}} \sigma(\boldsymbol{d}, \boldsymbol{m})\ d\boldsymbol{d} \tag{2.36}$$

$\sigma_M(\boldsymbol{m})$ contains the posterior information on the model parameters given the observed data and is therefore the solution to the inverse problem in the Bayesian setting. Due to the fact that the number of model parameters may be large it might be difficult to be able to extract information from $\sigma_M(\boldsymbol{m})$. Thus in order to analyze this posterior probability density in more detail we want to observe marginal densities. Let $\boldsymbol{m} = (m_1, \ldots, m_P)$ where P denotes the number of model parameters and let $\mathcal{M} = \mathcal{M}_1 \times \ldots \times \mathcal{M}_P$ be the Cartesian product of P model manifolds, then the posterior marginal probability density for m_i for an $1 \leq i \leq P$ can be obtained by

$$\sigma_{m_i}(m_i) = \int_{\mathcal{M}_1} dm_1 \ldots \int_{\mathcal{M}_{i-1}} dm_{i-1} \int_{\mathcal{M}_{i+1}} dm_{i+1} \ldots \int_{\mathcal{M}_P} dm_P\ \sigma_M(\boldsymbol{m}). \tag{2.37}$$

2.3.3 Gaussian Case

This section is dedicated to restating the posterior probability density defined in Section 2.3.2 in case uncertainties are Gaussian. Let us assume theoretical modeling uncertainties, described by the covariance matrix $\boldsymbol{C}_T^{-1}$, are Gaussian. Then the conditional probability $\theta(\boldsymbol{d}|\boldsymbol{m})$ relating $\boldsymbol{m}$ and $\boldsymbol{d}$ becomes

$$\theta(\boldsymbol{d}|\boldsymbol{m}) \sim exp(-\frac{1}{2}(\boldsymbol{d}-\boldsymbol{g}(\boldsymbol{m}))^T\boldsymbol{C}_T^{-1}(\boldsymbol{d}-\boldsymbol{g}(\boldsymbol{m}))). \tag{2.38}$$

Now let $\boldsymbol{C}_d$ denote the measurement uncertainty matrix and $\boldsymbol{C}_M$ the prior uncertainty matrix, then let $\boldsymbol{C}_D = \boldsymbol{C}_d + \boldsymbol{C}_T$ denote the matrix combining modeling and measurement uncertainties. The resulting posterior probability density is as follows:

$$\sigma_M(\boldsymbol{m}) \sim exp(-\frac{1}{2}(\boldsymbol{g}(\boldsymbol{m})-\boldsymbol{d})^T\boldsymbol{C}_D^{-1}(\boldsymbol{g}(\boldsymbol{m})-\boldsymbol{d}) - \frac{1}{2}(\boldsymbol{m}-\boldsymbol{m}_{prior})^T\boldsymbol{C}_M^{-1}(\boldsymbol{m}-\boldsymbol{m}_{prior})) \tag{2.39}$$

If the forward problem is linear the term $\boldsymbol{g}(\boldsymbol{m})$ will be substituted by $\boldsymbol{G}\boldsymbol{m}$ and the posterior model covariance $\hat{\boldsymbol{C}}_M$ is the inverse of the Hessian $\hat{\boldsymbol{C}}_M = (\boldsymbol{G}^T\boldsymbol{C}_D^{-1}\boldsymbol{G} + \boldsymbol{C}_M^{-1})^{-1}$.

2.4 Optimization Problem

As already mentioned in Section 2.2.2, a discrete-time system can be rewritten in matrix form and we obtain a system such as that stated in (2.15)–(2.16). Let us consider the linear case where we can state the forward problem with forward operator $\boldsymbol{G}_i = \boldsymbol{C}_i\boldsymbol{A}^{-1}\boldsymbol{F}$ as follows

$$\boldsymbol{A}\ \boldsymbol{u} = \boldsymbol{F}\ u_0, \tag{2.40}$$

$$\boldsymbol{d} = \boldsymbol{C}_i\ \boldsymbol{u}, \tag{2.41}$$

where (as before) $\boldsymbol{u}$ is the space-time state, u_0 is the initial condition, and $\boldsymbol{d}$ are the observables in space-time. The matrix $\boldsymbol{C}_i(S)$ defines the current sensor locations S in

the domain Ω. Due to simplicity of notation we will be writing $\boldsymbol{C}_i$ instead of $\boldsymbol{C}_i(S)$. Similarly the prediction problem with prediction operator $\boldsymbol{G}_p = \boldsymbol{C}_p \boldsymbol{A}^{-1} \boldsymbol{F}$ is given by

$$\begin{aligned} \boldsymbol{A}\ \boldsymbol{u} &= \boldsymbol{F}\ u_0, & (2.42) \\ \boldsymbol{p} &= \boldsymbol{C}_p\ \boldsymbol{u}, & (2.43) \end{aligned}$$

where p are the prediction outputs of interest, defined by the matrix $\boldsymbol{C}_p$. The prediction and the inversion problem are formulated separately because in general they are not the same, e.g. sensor locations and locations of interest might not coincide or the problems have a differing time horizon.

Let $\boldsymbol{C}_D^{-1}$ denote the uncertainty matrix combining model and measurement uncertainties and $\boldsymbol{C}_M^{-1}$ denotes the priori uncertainty matrix. Then the expression for the posterior of the initial condition covariance is

$$\hat{\boldsymbol{C}}_{\boldsymbol{m}} = \hat{\boldsymbol{C}}_{\boldsymbol{m}}(S) = (\boldsymbol{F}^T \boldsymbol{A}^{-T} \boldsymbol{C}_i^T \boldsymbol{C}_D^{-1} \boldsymbol{C}_i \boldsymbol{A}^{-1} \boldsymbol{F} + \boldsymbol{C}_M^{-1})^{-1} = (\boldsymbol{G}_i^T \boldsymbol{C}_D^{-1} \boldsymbol{G}_i + \boldsymbol{C}_M^{-1})^{-1} \quad (2.44)$$

Assuming $\boldsymbol{C}_D^{-1}$ is the identity and $\boldsymbol{C}_M^{-1} \to 0$, which signifies that there is no prior information, we compute the simplified posterior model covariance as

$$\hat{\boldsymbol{C}}_{\boldsymbol{m}} = \hat{\boldsymbol{C}}_{\boldsymbol{m}}(S) = (\boldsymbol{F}^T \boldsymbol{A}^{-T} \boldsymbol{C}_i^T \boldsymbol{C}_i \boldsymbol{A}^{-1} \boldsymbol{F} + \beta I)^{-1} = (\boldsymbol{G}_i^T \boldsymbol{G}_i + \beta I)^{-1}. \quad (2.45)$$

Note that βI, with β small, is merely a regularization term needed for inversion. The goal addressed in the optimization problem is to minimize the uncertainty in the prediction, thus the posterior covariance of the prediction field will be observed, computed as

$$\hat{\boldsymbol{C}}_p = \hat{\boldsymbol{C}}_p(S) = \boldsymbol{G}_p \hat{\boldsymbol{C}}_{\boldsymbol{m}}(S) \boldsymbol{G}_p^T. \quad (2.46)$$

With p, the prediction outputs of interest, being known, the posterior covariance of the prediction field depends on the set of sensor locations S. Thus we have to move the sensors to locations such that the uncertainty is minimized. This can be achieved

by minimizing the spectral norm of $\hat{\boldsymbol{C}}_p$, which in turn is equivalent to minimizing the maximum eigenvalue of $\hat{\mathbf{C}}_p$. Therefore the dynamic sensor steering optimization problem reads

$$\min_{\tilde{S}\in\Omega} \quad \lambda_{\max}(\hat{\boldsymbol{C}}_p(\tilde{S})). \tag{2.47}$$

As we are working with mobile sensors, we need to impose constraints representing allowable sensor motions. Those sensor constraints are incorporated into the optimization problem, meaning that prediction uncertainty will be minimized taking into consideration only the sensor locations that can be reached within T_{steer} and that satisfy the imposed sensor constraints. Thus the constrained optimization problem reads

$$\begin{aligned} \min_{\tilde{S}\in\Omega} \quad & \lambda_{\max}(\hat{\boldsymbol{C}}_p(\tilde{S})) \\ \text{s.t. } & \tilde{S} \text{ satisfies sensor constraints} \end{aligned} \tag{2.48}$$

A realistic and for this setting very useful approach incorporates the previously discussed steering time T_{steer} in which the sensors move to the recently computed optimal locations that minimizes uncertainty in the prediction. Obviously we do not want T_{steer} to be too long as the Dynamic Sensor Steering Algorithm aims at achieving real-time measuring-predicting-optimizing-steering, where time lost in steering the sensors is needed badly during the optimization as well as during the computation of the prediction. Thus we assume that a sensor can only reach locations that are within a certain radius R within the available steering time, which can be formulated as

$$\begin{aligned} \min_{\tilde{S}\in\Omega} \quad & \lambda_{\max}(\hat{\boldsymbol{C}}_p(\tilde{S})) \\ \text{s.t. } & (S_j^x - \tilde{S}_j^x)^2 + (S_j^y - \tilde{S}_j^y)^2 \leq R^2 \quad \forall j = 1, \ldots, Q, \end{aligned} \tag{2.49}$$

where $S = (S_1, \ldots, S_Q)$ is the set of current sensor locations and each sensor location consists of an x and y coordinate, i.e. $S_j = (S_j^x, S_j^y)$.

We solve the optimization problem using a gradient-based method, i.e. a sequential quadratic programming (SQP) method. At each iteration in this method

a quadratic programming subproblem is solved, the estimate of the Hessian of the Lagrangian is updated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula [9, 19, 23, 46].

2.5 Implementation Remarks

In large-scale problems there arise various computational challenges for the newly introduced Dynamic Sensor Steering Algorithm. First, the resulting model matrices may have large dimensions. Moreover as the covariance matrix is the inverse of the Hessian, we need to compute this inverse and its eigenvalue spectra as well. Therefore it is important to avoid explicit formation of those matrices and apply matrix-free methods only requiring the generation of matrix-vector products.

For large-scale problems with high-dimensional parameter spaces, computing covariance matrices can become a very demanding problem because explicit formation could already exceed available storage. Furthermore to compute the covariance matrix the Hessian needs to be inverted (see Section 2.4) which could be too costly. Therefore we aim at approximating $\hat{\boldsymbol{C}}_{\boldsymbol{m}}$ by using both a truncated spectral decomposition i.e. a Lanczos method and the Sherman-Morrison-Woodbury formula for inversion.

Let $V\Lambda V^T$ be the eigenvalue decomposition of $\boldsymbol{G}_i^T\boldsymbol{G}_i$,where Λ is the diagonal matrix containing the eigenvalues and V contains the corresponding eigenvectors. It often occurs that the eigenvalue spectrum of $\boldsymbol{G}_i^T\boldsymbol{G}_i$ decays rapidly, enabling the approximation through a truncated spectrum, meaning that we can approximate $V\Lambda V^T$ based on the few dominant eigenvalues and vectors contained in V_r, such that $V\Lambda V^T \approx V_r\Lambda V_r^T$. Applying the Sherman-Morrison-Woodbury formula [52] now we obtain the following approximation

$$\hat{\boldsymbol{C}}_{\boldsymbol{m}} = (\boldsymbol{G}_i^T\boldsymbol{G}_i+\beta I)^{-1} = (V\Lambda V^T+\beta I)^{-1} \approx (V_r\Lambda V_r^T+\beta I)^{-1} = \frac{1}{\beta}(I-V_rDV_r^T), \quad (2.50)$$

where D is a diagonal matrix with $D_{ii} = \frac{\lambda_i}{\beta+\lambda_i}$.

A second challenge is related to representation of sensor locations. In order to make the problem differentiable, the sensor locations are represented by mollified delta functions, meaning the sensor locations are modeled by Gaussian functions centered at the according sensor location where the variance of the Gaussian needs to be chosen appropriately in conjunction with the local grid size, thereby enabling the sensor locations to be anywhere in the domain Ω (not just at grid points).

Chapter 3

Dynamic Sensor Steering Algorithm - Application and Results

In Chapter 3 we apply the methodology presented in Chapter 2 to a two-dimensional contaminant transport problem that is modeled by the convection-diffusion equation. This problem is introduced in Section 2.1.2. The following sections present results that include a comparison between the full and the reduced-order model, and a demonstration of the use of model reduction for uncertainty quantification.

3.1 Description of Rectangular Domain

Let us consider a specific example, where the dimensions of the domain Ω are described by

$$\Omega = \begin{cases} 0 \leq x \leq 1 \\ 0 \leq y \leq 0.4 \end{cases} \tag{3.1}$$

and the computational domain has spatial mesh size of $N = 4005$. Figure 3-1 shows the computational mesh with triangular elements as well as the dimensions of domain Ω. At the inflow boundary characterized by $\Gamma_D = \{x = 0, 0 \leq y \leq 0.4\}$ we impose homogeneous Dirichlet boundary conditions. On all the remaining boundaries of Ω

we impose homogeneous Neumann boundary conditions. Throughout this chapter we assume a diffusivity $\kappa = 0.01$, $\Delta t = 0.005$, forty time steps and a constant velocity field $\boldsymbol{v}_1$, defined as follows

$$\boldsymbol{v}_1(x, y) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{for } (x, y) \in \Omega. \tag{3.2}$$

The output region of interest, Ω_I, is defined as follows:

$$\Omega_I = \begin{cases} 0.6 \leq x \leq 0.8 \\ 0.15 \leq y \leq 0.25 \end{cases}. \tag{3.3}$$

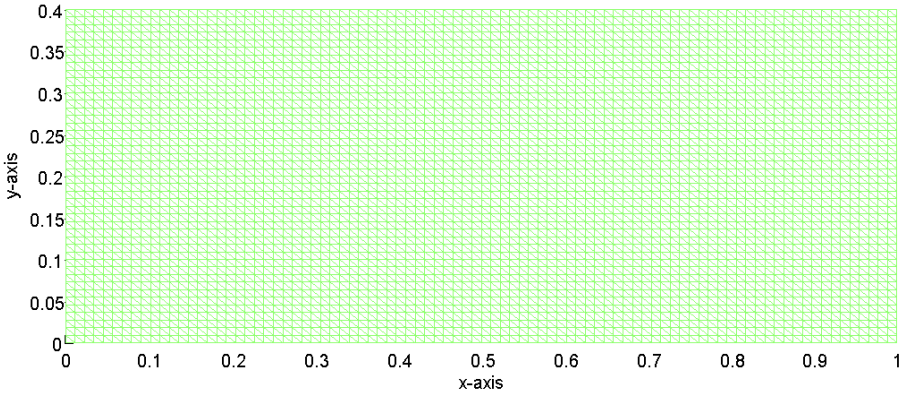

Figure 3-1: Triangular mesh of domain Ω.

3.2 Reduced-Order Model Performance in Forward Problem

We choose to demonstrate the results using a superposition of three Gaussian functions as the initial contaminant concentration depicted in Figure 3-2. Each Gaussian function is defined as

$$u_0(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}}, \tag{3.4}$$

where σ corresponds to the standard deviation and (x_c, y_c) represents the center of the Gaussian.

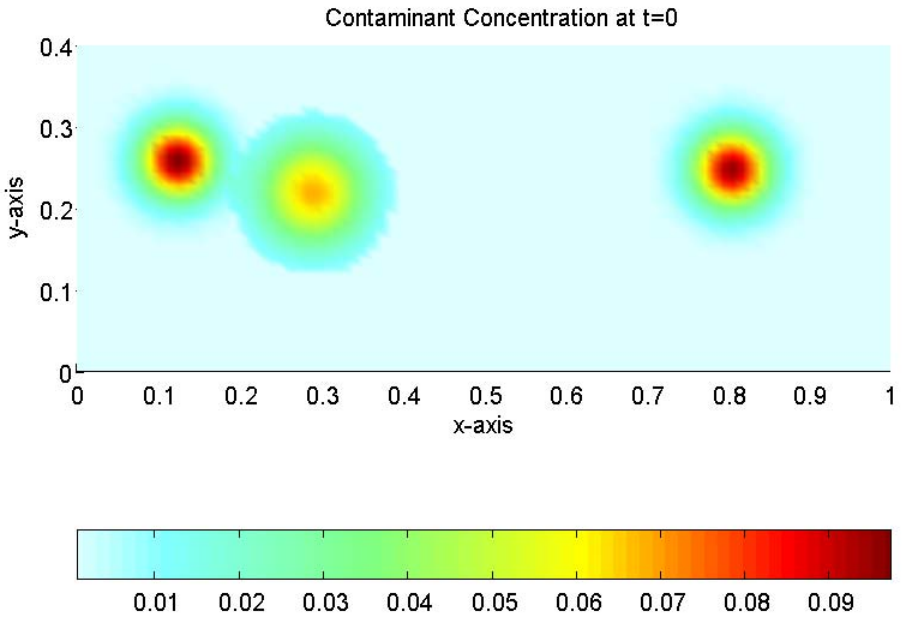

Figure 3-2: This figure depicts the sample test initial condition used to compare reduced model to full model in the forward problem.

In Figure 3-3 we compare reduced model outputs to full-scale outputs at two sensor locations $S_1 = (0.15, 0.25)$ and $S_2 = (0.45, 0.2)$. We define the error ε due to a particular initial condition u_0 as

$$\varepsilon \;=\; || \, \boldsymbol{d} - \boldsymbol{d}_r \, ||_2 \;\;=\;\; || \, (\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{F} - \boldsymbol{C}_r\boldsymbol{A}_r^{-1}\boldsymbol{F}_r) \; u_0 \, ||_2, \tag{3.5}$$

where $\boldsymbol{d}$ and $\boldsymbol{d}_r$ correspond to the full model output and the reduced-order model output respectively. Table 3.1 displays various reduced-order model properties, where ε denotes the error compared to the full model, $c - time$ denotes the computational time to obtain the outputs in seconds, n denotes the size of the reduced-order model, and the significance of $\bar{\lambda}$ and $\bar{\mu}$ are as follows. When creating a snapshot matrix using the Hessian-based method described in [4] we compute s eigenvectors corresponding to the s largest eigenvalues of the Hessian. Thus we specify $\bar{\lambda}$ so that eigenvector

i, which has eigenvalue λ_i, is in the pool of initial conditions if $\frac{\lambda_i}{\lambda_1} > \bar{\lambda}$, where λ_1 is the largest eigenvalue of the Hessian. Then we apply POD to the snapshot matrix to obtain a basis of size n for the reduced-order model. The number n is specified by relating the POD eigenvalues $\mu_1, \ldots, \mu_{(T+1)s}$ to the largest one μ_1. Thus the POD basis vector corresponding to eigenvalue μ_j is included in the basis V if $\frac{\mu_j}{\mu_1} > \bar{\mu}$. Note that the size of the reduced-order model used in our results will vary according to the choice of $\bar{\lambda}$ and $\bar{\mu}$.

Running the forward problem using the full model takes 5.5 seconds on a desktop computer with Intel Core 2 Duo processor and 2GB RAM. Note that all computational results in this thesis were computed using the same desktop computer. From Table 3.1 we see that the largest reduced-order model of size $n = 212$ is about ten times faster than the full model and the smallest reduced-order model of size $n = 38$ computes results 360 times faster. The performance with respect to error and computational time of each reduced-order model can be seen in Figure 3-4.

case	ε	$c-time$	n	$\bar{\lambda}$	$\bar{\mu}$
1	0.0290	0.0160	38	0.5	10^{-4}
2	0.0273	0.0280	53	0.5	10^{-6}
3	0.0039	0.0610	80	0.1	10^{-4}
4	0.0033	0.1430	118	0.1	10^{-6}
5	$1.9211e^{-4}$	0.1480	121	0.01	10^{-4}
6	$1.9087e^{-4}$	0.1950	135	0.001	10^{-4}
7	$1.1382e^{-4}$	0.4340	188	0.01	10^{-6}
8	$1.0232e^{-4}$	0.5690	212	0.001	10^{-6}

Table 3.1: Properties of various reduced-order models of a full-scale system with size $N = 4005$ and two output sensors in the forward problem. The error ε is defined in (3.5) and the initial condition used is depicted in Figure 3-2.

3.3 Solution of the Inverse Problem

In the contaminant transport problem introduced in (2.5)–(2.8) the vector of model parameters $\boldsymbol{m}$ corresponds to the initial condition u_0, and the output or the data parameters are $\boldsymbol{d} = (d_1, \ldots, d_Q)$, where d_j for $1 \leq j \leq Q$ corresponds to the conta-

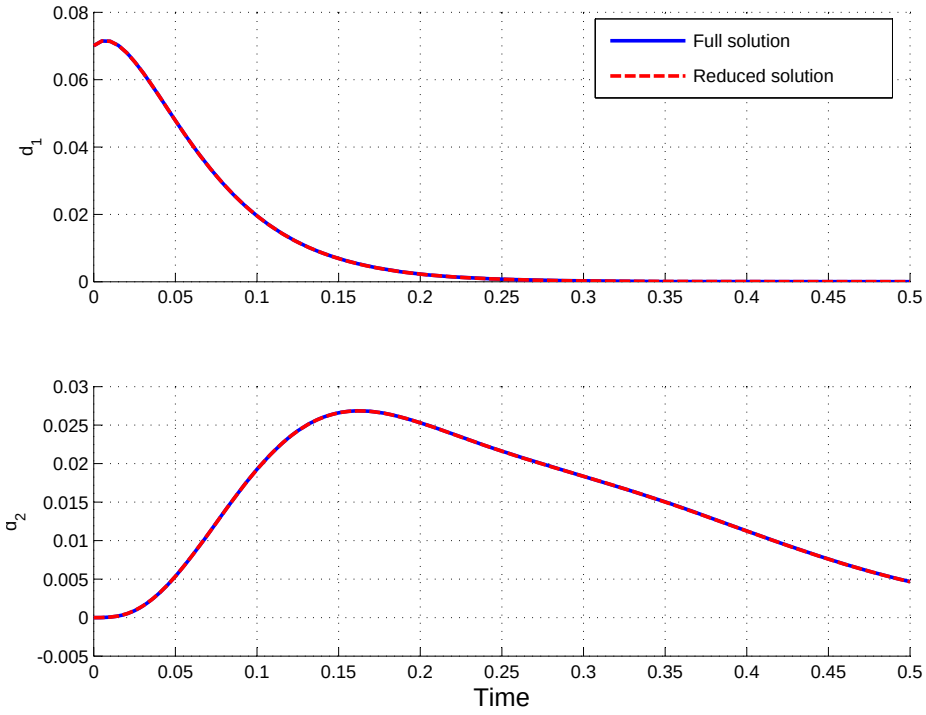

Figure 3-3: A comparison between full ($N = 4005$) and reduced outputs ($n = 212$) for sensors S_1 and S_2 using initial condition depicted in Figure 3-2 with $\kappa = 0.01$, $\Delta t = 0.005$ and hundred time steps. The error is $\varepsilon = 1.0232e^{-4}$.

minant concentration at sensor location j through time. The true initial condition u_0 that was chosen to present the results is shown in Figure 3-5. The observations $\boldsymbol{d}^{obs} = (d_1^{obs}, \ldots, d_Q^{obs})$ were obtained by propagating u_0 through the forward model and adding 5% noise to each component. The forward model is given by

$$d_j = \boldsymbol{G}_j u_0 = (\boldsymbol{C}_j \boldsymbol{A}^{-1} \boldsymbol{F}) u_0 \quad \text{for } 1 \leq j \leq Q, \tag{3.6}$$

where $\boldsymbol{A}$ and $\boldsymbol{F}$ are defined as in (2.17) and $\boldsymbol{C}_j$ is the matrix corresponding to sensor location j. As data uncertainties are Gaussian we can write the probability density distributions of the values of contaminant concentration as

$$\rho_D(d_1, \ldots, d_Q) \sim exp(-\frac{1}{2\sigma^2} \sum_{j=1}^{Q} (d_j - d_j^{obs})^T (d_j - d_j^{obs})). \tag{3.7}$$

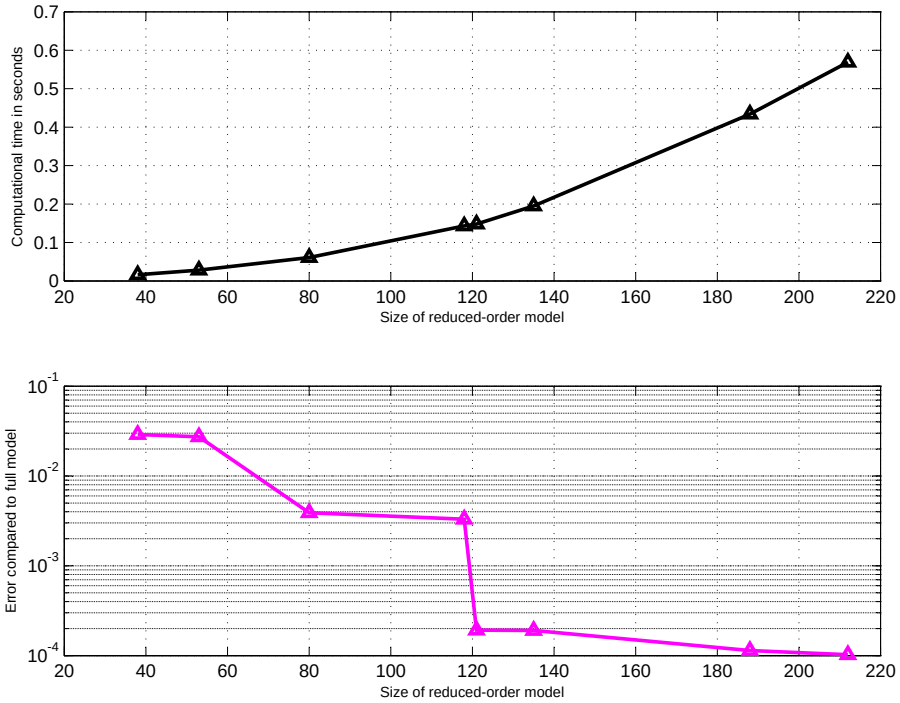

Figure 3-4: The upper plot shows the increase in computational time in seconds of the forward problem as the size of the reduced-order model grows over the rectangular domain. The lower plot depicts the decrease of the error defined in (3.5) as the size of the reduced-order model increases.

From this we obtain the probabilistic solution to the inverse problem, which is the posterior probability density of the model parameters given by

$$\sigma_M(u_0) \sim exp(-\frac{1}{2\sigma^2}\sum_{j=1}^{Q}(\boldsymbol{G}_j u_0 - d_j^{obs})^T(\boldsymbol{G}_j u_0 - d_j^{obs})). \tag{3.8}$$

This posterior probability density provides us with information on how likely each initial scenario u_0 is. In the deterministic approach, the solution of the inverse problem is just a single estimate of u_0, while the solution in the Bayesian approach provides the probability distribution function of u_0.

Here we have two sensors $S_1 = (0.15, 0.2)$ and $S_2 = (0.5, 0.2)$, hence $Q = 2$. Let us analyze this solution and the quality of reduced-order models via Figures 3-5, 3-6, 3-8 and Table 3.2. The mean of the initial condition field $\hat{u}_0$ obtained by the full model is referred to as $\hat{u}_0^{full}$ and depicted in the upper plot of Figure 3-6, whereas the

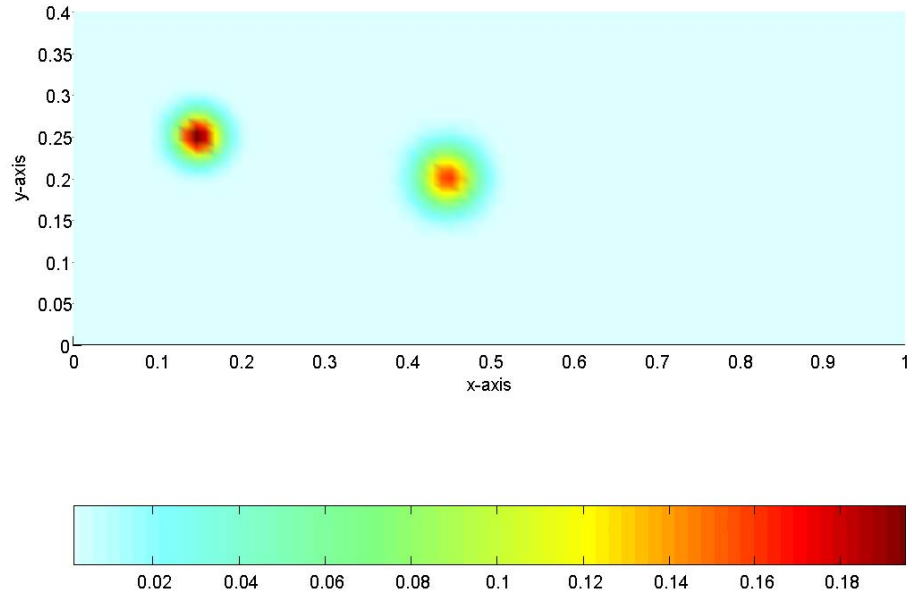

Figure 3-5: This plot depicts the true initial contaminant concentration u_0 used in the inverse problem.

mean of the initial condition field computed by the reduced model of size $n = 212$, referred to as $\hat{u}_0^{red}$, is shown in the lower plot of Figure 3-6. It takes 37.6 seconds to compute $\hat{u}_0^{full}$ using the full model and 4.2 seconds to compute $\hat{u}_0^{red}$, which is nine times faster. When using a reduced-order model of size $n = 38$ we can compute $\hat{u}_0^{red}$ about 123 times faster. In Table 3.2 and Figure 3-8 we present some reduced-order model results. The time to compute $\hat{u}_0^{red}$ is referred to as $c - time$, and the error between the estimate of the initial condition field using the full and the reduced model is denoted by ε_{u_0} and defined by

$$\varepsilon_{u_0} \;=\; || \, \hat{u}_0^{full} - \hat{u}_0^{red} \, ||^2 \, . \tag{3.9}$$

The full system has spatial size $N = 4005$ and hence $u_0 = (u_0^1, \ldots, u_0^i, \ldots, u_0^N)$, where u_0^i refers to the initial contaminant concentration at grid point i. For the remainder of this section $i = 2506$ which corresponds to the location (0.1477,0.2105) in Ω. In Figure 3-7 we depict the marginal probability density of u_0^i, namely $\sigma_{u_0^i}(u_0^i)$ which is a Gaussian with standard deviation σ and mean μ, where $\mu = \hat{u}_0^{full,i} = 0.2105$ with i

corresponding to grid point (0.1477,0.2105).

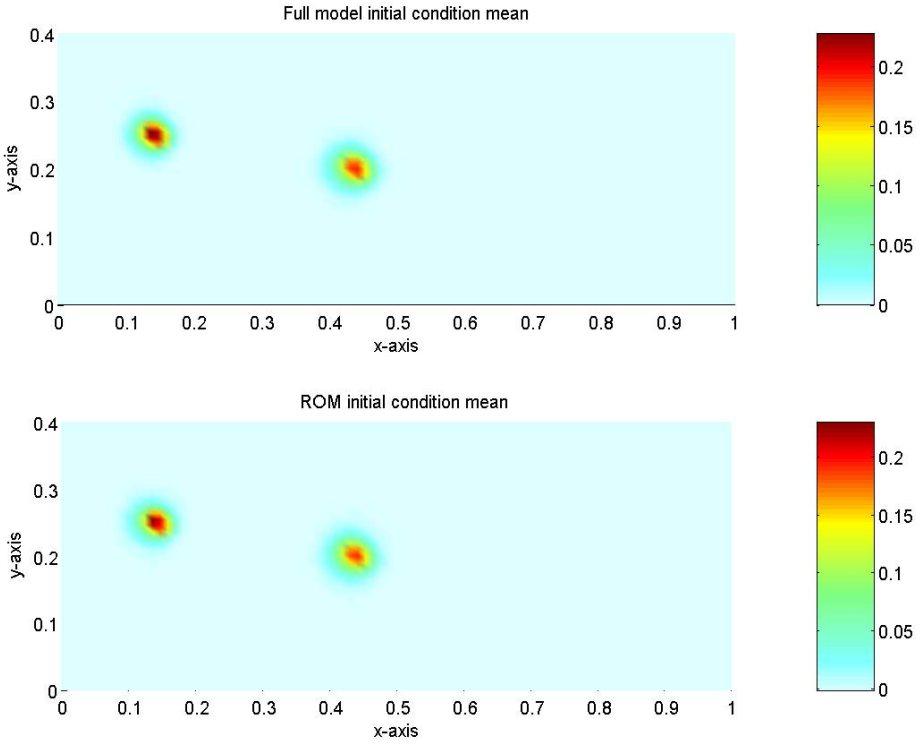

Figure 3-6: The upper plot shows the full model estimate $\hat{u}_0^{full}$ of the initial condition field. The lower plot shows the reduced model estimate $\hat{u}_0^{red}$ of the initial condition field with $n = 212$ which yields $\varepsilon_{u_0} = 0.0062$.

3.4 Reduced-Order Model Performance in Optimization Problem

3.4.1 Unconstrained Optimization

Let us consider the unconstrained optimization problem (2.47) introduced in Section 2.4. Due to the lack of sensor constraints the sensors are free to move anywhere in the domain Ω defined in (3.1). The output region of interest Ω_I is as stated in (3.3) and we expect the optimal sensor locations to be within Ω_I. For the following results we assume the number of sensors $Q = 2$. Solving the optimization problem using

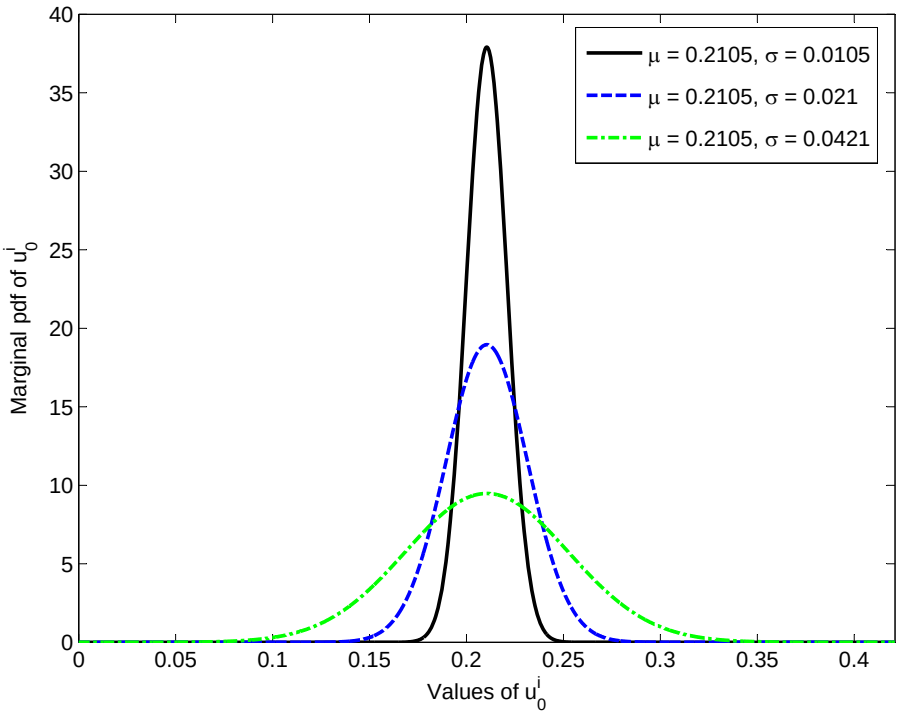

Figure 3-7: Marginal probability density $\sigma_{u_0^i}(u_0^i)$ with mean $\mu = 0.2105$ and with varying standard deviation σ. Note that i corresponds to grid point (0.1477,0.2105) in Ω.

the full model of size $N = 4005$ yields optimal locations of $S_1^* = (0.6783, 0.2034)$ and $S_2^* = (0.7751, 0.1999)$ after about 31.4 hours of computation time. Let us the define the average optimization error ε_{opt} as follows

$$\varepsilon_{opt} = \frac{1}{Q}\sum_{j=1}^{Q} dist_j, \quad \text{where} \tag{3.10}$$

$$dist_j = \sqrt{(S_j^{*,x} - S_{j,r}^x)^2 + (S_j^{*,y} - S_{j,r}^y)^2}, \tag{3.11}$$

where optimal sensor locations computed by the full model are $S_j^* = (S_j^{*,x}, S_j^{*,y})$ and optimal sensor locations computed by the reduced-order model are given by $S_{j,r}^* = (S_{j,r}^{*,x}, S_{j,r}^{*,y})$. In Table 3.3 we present the optimization results obtained using reduced-order models of size n. The average optimization error is defined in (3.10) and the time in seconds to compute an optimal solution is denoted by $c - time$.

From the obtained results we can see that the average optimization error decreases

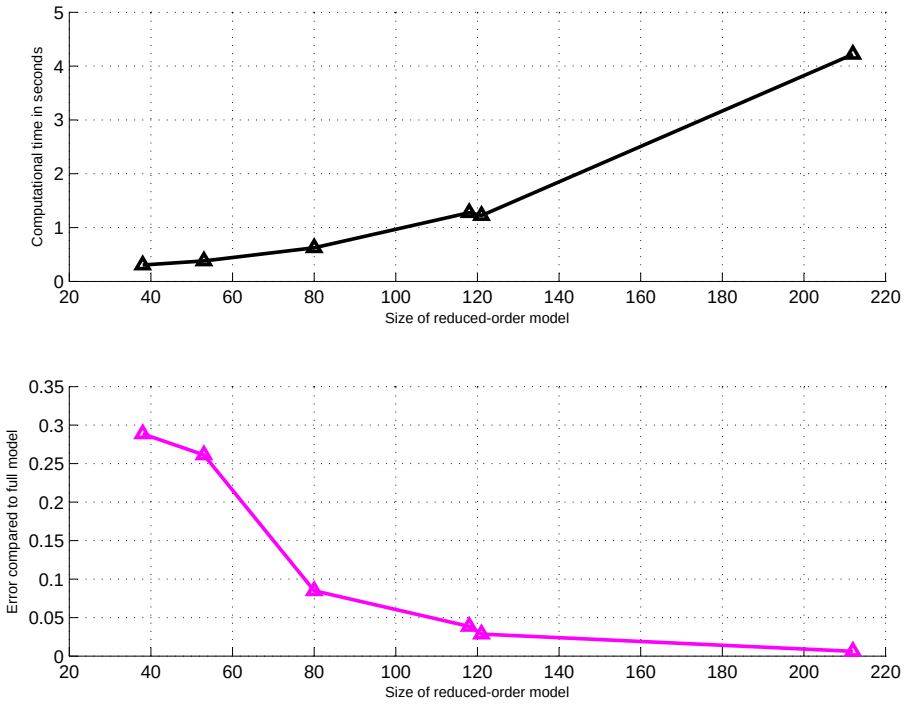

Figure 3-8: The upper plot depicts the increase in computational time as the size of the reduced-order model increases. The lower plot depicts the error according to reduced-order model sizes in the inverse problem.

with increasing n but only by very little. Hence we suggest to run the optimization problem with a reasonably small reduced-order model as the error doesn't change that drastically but the computing time increases quite fast. Note that in the Dynamic Sensor Steering Algorithm the optimization problem takes by far the most time compared to the forward and inverse problem. A reduced-order model of size $n = 212$ computes the optimal sensor locations about 159 times faster than the full model and with $n = 38$ we can compute up to 300 times faster. We depict the obtained optimal solutions in Figure 3-9.

3.4.2 Constrained Optimization

In this section we consider the constrained optimization problem (2.49) introduced in Section 2.4 with a radius R of 0.2. The output region of interest Ω_I is as stated in (3.3) and we expect the optimal sensor locations to be either within Ω_I, if the sensor

case	ε_{u_0}	$c-time$	n	$\bar{\lambda}$	$\bar{\mu}$
1	0.2887	0.305	38	0.5	10^{-4}
2	0.2613	0.381	53	0.5	10^{-6}
3	0.0848	0.624	80	0.1	10^{-4}
4	0.0386	1.276	118	0.1	10^{-6}
5	0.0286	1.224	121	0.01	10^{-4}
6	0.0062	4.218	212	0.001	10^{-6}

Table 3.2: Properties of various reduced-order models of a full-scale system with size $N = 4005$ and two output sensors in the inverse problem.

case	ε_{opt}	$c-time$	n
1	0.1613	373.921	38
2	0.0338	438.442	53
3	0.0267	651.141	80
4	0.0263	657.382	118
5	0.0119	711.401	212

Table 3.3: Results of various reduced-order models in the unconstrained optimization problem.

constraints allow it, or to move in the direction of Ω_I until the sensor constraint becomes active. For the following results we assume the number of sensors $Q = 2$ with initial locations $S_1^0 = (0.9, 0.3)$ and $S_2^0 = (0.4, 0.1)$. Solving the optimization problem using the full model of size $N = 4005$ yields optimal locations of $S_1^* = (0.7549, 0.2033)$ and $S_2^* = (0.5815, 0.1840)$ after about 17 hours of computation time. In Table 3.4 we present the optimization results obtained using reduced-order models of size n. The average optimization error ε_{opt} is defined in (3.10) and the time to compute an optimal solution is denoted by $c-time$ in seconds. As in the unconstrained case, the results show that increasing the size of the reduced-order model yields only small reductions in error, but has a large impact on computing time. In this case, a reduced-order model of size $n = 212$ computes the optimal sensor locations about 27 times faster than the full model, while $n = 38$ leads to a speed-up factor of about 94. We depict the obtained optimal solutions in Figure 3-10.

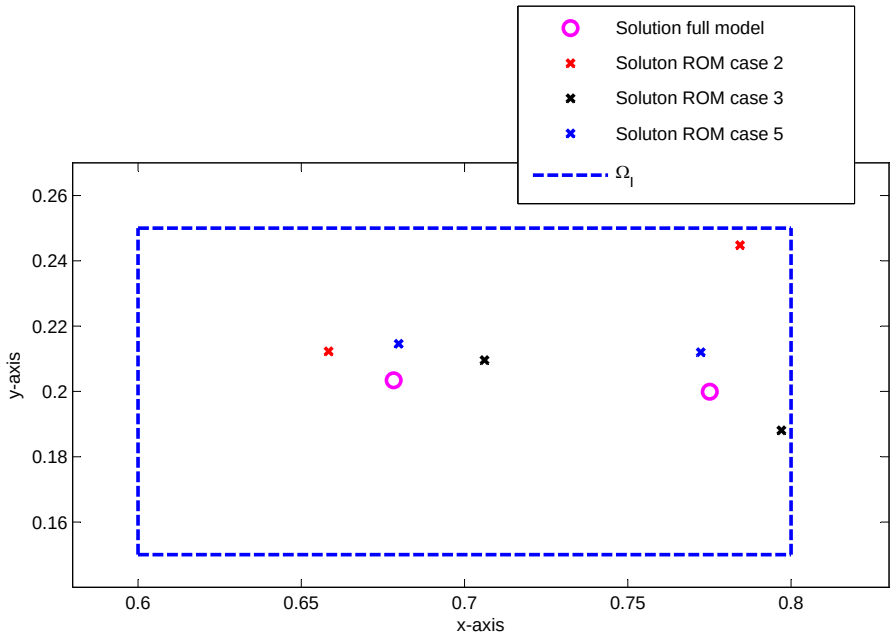

Figure 3-9: This figure contains the new sensor locations computed by the full model and several reduced-order models in the unconstrained case.

3.4.3 Optimality Conditions

In the unconstrained case the sensor can move anywhere within the bounds and hence we expect the optimal sensor locations to be within Ω_I. After the optimization algorithm terminates we verify that the gradient at the new sensor locations is zero. Hence we can state that we have found at least a local minimum. How confidently can we say that this local minimum is in fact a global minimum. When we fix $Q-1$ sensors and compute the objective function, i.e. the maximum eigenvalue of the covariance matrix, for the remaining sensor over the entire domain Ω we can observe that the local minimum is a global one for the remaining sensor. We can repeat this procedure Q times always letting one sensor run freely and we get similar results. Moreover we solved the two sensor optimization problem for hundred randomly chosen initial sensor locations and in 96% of the cases the SQP algorithm converged to the same solution. In the remaining 4% the algorithm got stuck near to the initialized points because they were chosen too close to the boundaries of the domain.

In the constrained case we obtain a solution that either has zero gradient, if we

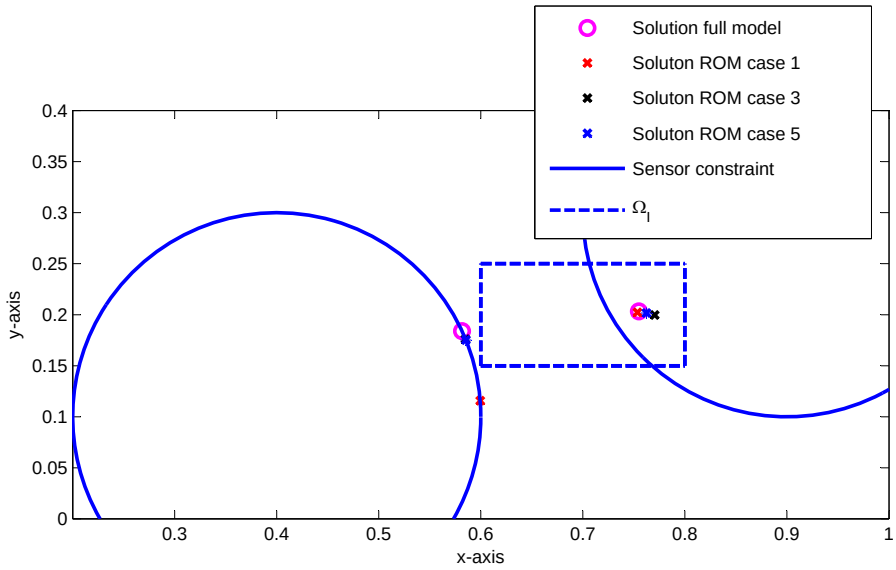

Figure 3-10: This figure contains the new sensor locations computed by the full model and several reduced-order models in the constrained case.

case	ε_{opt}	$c - time$	n
1	0.0361	655.644	38
2	0.0302	660.103	53
3	0.0122	688.332	80
4	0.0119	1957.057	118
5	0.0086	2304.569	212

Table 3.4: Results of various reduced-order models in the constrained optimization problem.

can reach Ω_I, or nonzero gradients, if the sensor constraints become active. We observed that, when repeating the constrained optimization problem long enough, always initializing with the previous optimal solution we end up at the same locations in Ω_I like we would have in the unconstrained case. This is only true if all the sensors can actually reach Ω_I. In the next section, we demonstrate how the sensors move during various cycles of the online stage.

3.5 Moving Window Illustration

In this section we demonstrate the work flow of the Dynamic Sensor Steering Algorithm over several cycles. We solve the optimization problem using a reduced-order model of size $n = 212$. The quality of the reduced solution compared to the full solution has already been discussed in Section 3.4. Note that we are also applying a set of sensor constraints, which were discussed in Section 2.4, and the corresponding optimization problem is stated in (2.49). The radius R is 0.1 and Ω_I is as defined in (3.3). Each figure shows the current contaminant concentration, the current sensor locations, the circle within which each sensor can steer, and the optimal sensor locations computed by solving the optimization problem.

In Figures 3-11–3-14 we observe four subsequent cycles of the online stage of the Dynamic Sensor Steering Algorithm. Figure 3-11 corresponds to Cycle 1 of the online stage and shows the current contaminant concentration at $t_f = 0.2$, the domain of interest Ω_I, the current sensor locations $S_1^0 = (0.3, 0.2)$, $S_2^0 = (0.4, 0.3)$, the sensor constraints and the optimal sensor locations $S_1^1 = (0.3962, 0.1727)$ and $S_2^1 = (0.4889, 0.2541)$. The lower plot in Figure 3-11 shows a low-rank approximation of the variance field. In Cycle 1 the variance in Ω_I is still quite large. Thus in the following cycles we move the sensors even closer to that region. Note that the variance is very small around the sensor locations.

Figure 3-12 corresponds to Cycle 2 of the online stage and shows the contaminant concentration at $t_f = 0.4$, the current sensor locations S_1^1, S_2^1 computed in Cycle 1, the sensor constraints and the optimal sensor locations $S_1^2 = (0.4895, 0.2087)$ and $S_2^2 = (0.5779, 0.2093)$. Due to the newly obtained sensor locations, the variance in Ω_I has decreased.

Figure 3-13 corresponds to Cycle 3 of the online stage and shows the contaminant concentration at $t_f = 0.6$, the current sensor locations S_1^2, S_2^2 computed in Cycle 2, the sensor constraints and the optimal sensor locations $S_1^3 = (0.5895, 0.2074)$ and $S_2^3 = (0.6775, 0.2008)$. The variance over Ω_I decreases further.

Figure 3-14 corresponds to Cycle 4 of the online stage and shows the contaminant

concentration at $t_f = 0.8$, the current sensor locations S_1^3, S_2^3 computed in Cycle 3, the sensor constraints and the optimal sensor locations $S_1^4 = (0.6492, 0.1980)$ and $S_2^4 = (0.7500, 0.2002)$. The variance field over Ω_I approaches zero.

In the first three figures we can observe that the optimal sensor locations lie on the boundary imposed by the sensor constraints and thus do not lead to a zero gradient. In Figure 3-14 the optimal sensor locations both lie within Ω_I as expected and furthermore none of the sensor constraints are active. Moreover with sensors at S_1^4 and S_2^4 the gradient is zero guaranteeing that we are at a local minimum.

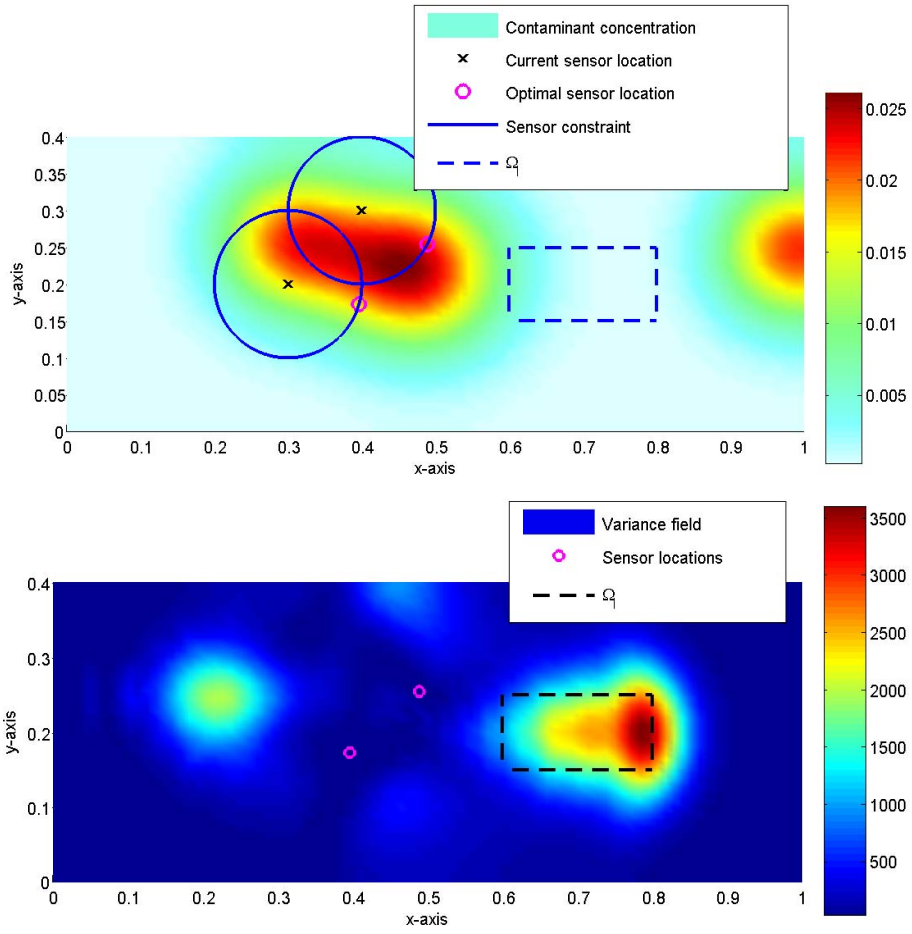

Figure 3-11: The upper figure depicts the contaminant concentration at $t_f = 0.2$, current and optimal sensor locations of cycle one of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω.

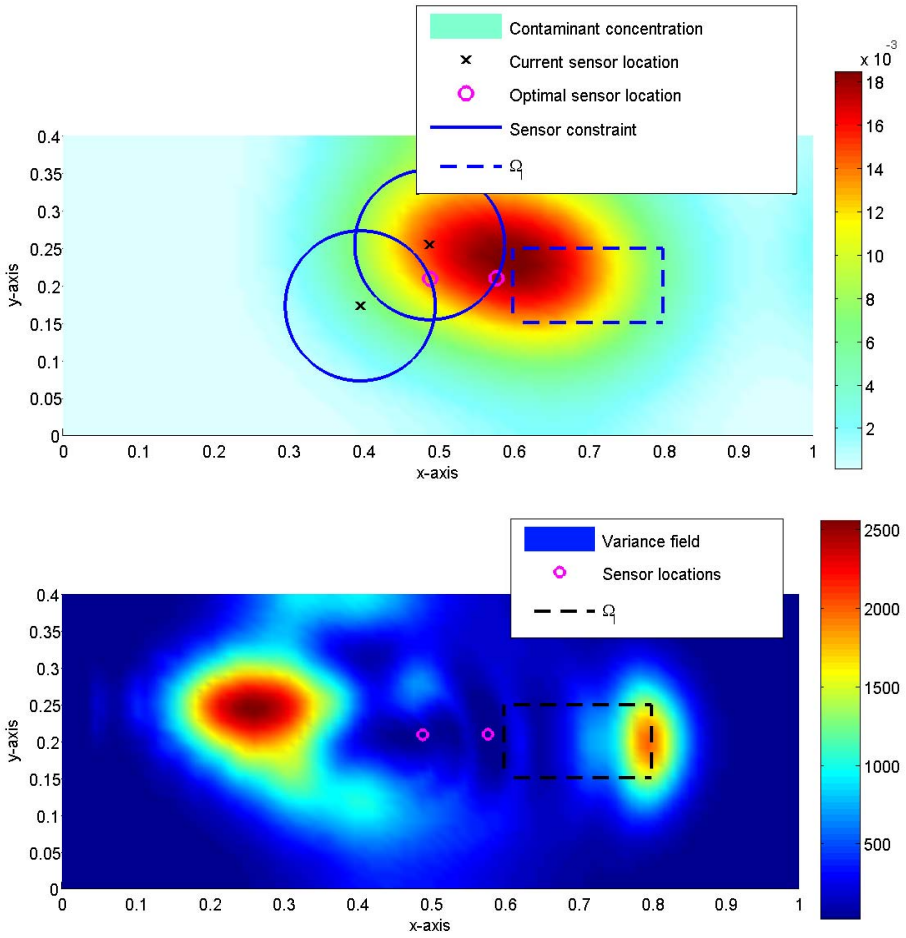

Figure 3-12: The upper figure depicts contaminant concentration at $t_f = 0.4$, current and optimal sensor locations of cycle two of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω.

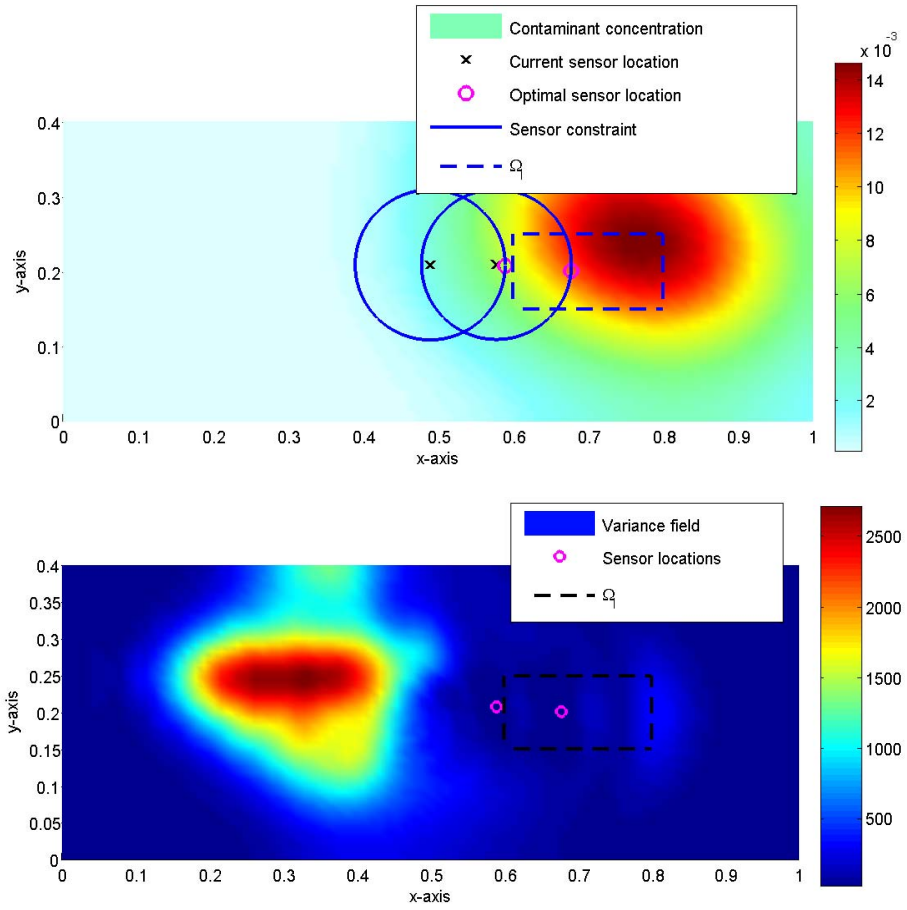

Figure 3-13: The upper figure depicts the contaminant concentration at $t_f = 0.6$, current and optimal sensor locations of cycle three of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω.

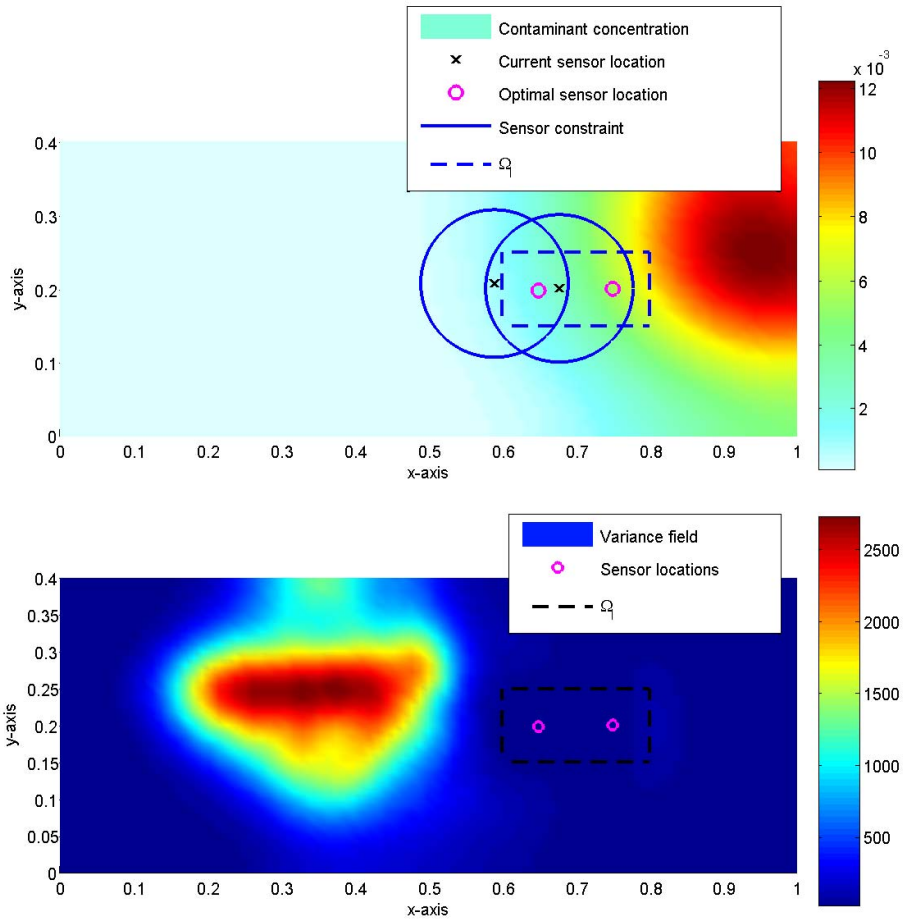

Figure 3-14: The upper figure depicts the contaminant concentration at $t_f = 0.8$, current and optimal sensor locations of cycle four of the online stage with constant velocity field $\boldsymbol{v}_1$. The lower plot shows the corresponding variance field based on the current optimal sensor locations over Ω.

Chapter 4

Model Order Reduction of Convection-Diffusion Equation with Parameterized Velocity Field

In Chapter 4 we consider the convection-diffusion equation and introduce a parameterized velocity field in the Dynamic Sensor Steering Algorithm setting, which enables a more realistic simulation of physical processes. The posed challenge is to maintain the algorithm's computational real-time property, by building reduced-order models that are not only dependent on the initial condition but also on the velocity field. In Section 4.1 we will provide a detailed problem statement, followed by a description of the extended Dynamic Sensor Steering Algorithm in Section 4.2 and by a discussion of the applied model order reduction methodology for linear dependence of the velocity field on the parameter vector in Section 4.3.

4.1 Problem Description

We consider the the convection-diffusion equation described in Section 2.1.2 but now the field variable, denoting the contaminant concentration, and the velocity field, and thus the state solution, depend on an input parameter vector $\boldsymbol{\mu}$. That is, we now have $\boldsymbol{v}(\boldsymbol{\mu})$ and $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^P$. The diffusivity constant is κ and the observed time

horizon is given by t_f. We imposed a homogeneous Dirichlet boundary condition on $\Gamma_D \times (0, t_f)$ and homogeneous Neumann boundary conditions on $\Gamma_N \times (0, t_f)$.

It is assumed that there exists a way of computing a velocity field $\boldsymbol{v}(\boldsymbol{\mu})$ over the entire domain Ω given an input parameter vector $\boldsymbol{\mu}$. Hence there are two separate cases, namely the linear case if the dependence of $\boldsymbol{v}(\boldsymbol{\mu})$ on $\boldsymbol{\mu}$ is linear and the non-linear case when this dependence is non-linear. In this chapter we will focus on the linear case.

For the linear case we assume that P different convective velocity fields are given $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_P$ and that $\boldsymbol{v}(\boldsymbol{\mu})$ is an linear combination of velocity fields that we specified a priori. Therefore the current velocity field $\boldsymbol{v}(\boldsymbol{\mu})$ in our domain Ω can be computed by:

$$\boldsymbol{v}(\boldsymbol{\mu}) = \sum_{j=1}^{P} \mu_j \; \boldsymbol{v}_j \tag{4.1}$$

with input parameter $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_P) \subset \mathbb{R}^P$. The P pre-computed velocity fields $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_P$ could come from e.g. a Navier-Stokes flow solver, or a reduced-order model of a flow solver, etc.

4.1.1 Weak Form and Finite Element Method

The weak formulation of the problem stated above is as follows: $\forall \boldsymbol{\mu} \in \mathcal{D}$, find $\mathfrak{u}(\boldsymbol{\mu}) \in X$, where $X = \{\mathfrak{u} \in H^1(\Omega) \mid \mathfrak{u} \mid_{\Gamma_D} = 0\}$, such that $\forall w \in X$,

$$m(\frac{\partial \mathfrak{u}(\boldsymbol{\mu})}{\partial t}, w) + a(\mathfrak{u}(\boldsymbol{\mu}), w; \boldsymbol{\mu}) = l(w), \tag{4.2}$$

where $a(\cdot, \cdot)$ is a bilinear functional, $l(\cdot)$ is a linear functional and $m(\frac{\partial \mathfrak{u}(\boldsymbol{\mu})}{\partial t}, w)$ is a bilinear form that results from the inner product of $\frac{\partial \mathfrak{u}(\boldsymbol{\mu})}{\partial t}$ and the test functions w. The time discretization is then performed using Backward Euler or any other appropriate scheme.

For $w, \bar{w} \in X$ the bilinear and linear functionals stated in (4.2) are given by

$$
\begin{aligned}
a(w, \bar{w}; \boldsymbol{\mu}) &= a_1(w, \bar{w}; \boldsymbol{\mu}) + a_2(w, \bar{w}), && (4.3)\\
a_1(w, \bar{w}; \boldsymbol{\mu}) &= \int_\Omega w \ \boldsymbol{v}(\boldsymbol{\mu}) \cdot \nabla \bar{w} \ d\Omega, && (4.4)\\
a_2(w, \bar{w}) &= \kappa \int_\Omega \nabla w \cdot \nabla \bar{w} \ d\Omega, && (4.5)\\
l(w) &= \int_\Omega w f \ d\Gamma, && (4.6)
\end{aligned}
$$

where κ corresponds to the diffusivity and f corresponds to the source term, which is zero for our problem.

Time Discretization and Matrix Equations

In Section 2.2.2 we have introduced the general linear discrete-time system (2.12)–(2.14), which can be expressed in the following continuous form

$$
\begin{aligned}
\mathcal{M} \ \dot{u}(k) &= \mathcal{A} \ u(k), && (4.7)\\
d(k) &= C \ u(k) && (4.8)
\end{aligned}
$$

where $\dot{u}$ denotes the vector of state derivatives with respect to time and $\mathcal{M} \in \mathbb{R}^{N \times N}$ and $\mathcal{A} \in \mathbb{R}^{N \times N}$ correspond to the mass matrix and the stiffness matrix respectively obtained from the finite-element discretization. For the discretization in time we will use a Backward Euler method with a time step Δt. In order to write (4.7)–(4.8) in matrix form let us define the following matrices according to the chosen Backward Euler method.

$$
\begin{aligned}
D_1 &= \frac{1}{\Delta t}\mathcal{M} + \mathcal{A} && (4.9)\\
D_2 &= \frac{1}{\Delta t}\mathcal{M} && (4.10)
\end{aligned}
$$

Inserting D_1 from (4.10), D_2 from (4.10) and C from (4.8) into (2.17) we can derive the same matrix form as stated in (2.15)–(2.16).

Streamline-Upwind / Petrov-Galerkin Stabilization

For small values of the diffusivity κ a solution to the discretization of (4.2) can exhibit non-physical oscillations due to the negative numerical diffusion of the Galerkin finite element method, which is described in detail in the literature. In [15] this instability occurs when the Péclet number is greater than one. In our problem however it can not be guaranteed to have the same Péclet number for each element in the triangulation because the current velocity field $\boldsymbol{v}$ may not be constant. Thus we define an elemental Péclet number Pe_k such that

$$Pe_k = \frac{h \mid\mid \boldsymbol{v}_k \mid\mid_{\ell^2}}{2\kappa} \quad \text{for } k = 1, \ldots, N_{elem}, \tag{4.11}$$

where N_{elem} corresponds to the number of elements in our mesh, h is the size of the element and $\boldsymbol{v}_k$ denotes the velocity at the centroid of element k. Therefore when $\max_{1 \le k \le N_{elem}} Pe_k > 1$ we will have add stabilization to (4.2). This can be done by deploying the Streamline-Upwind Petrov-Galerkin method by Brooks and Hughes [8], which leads to the weak form

$$m(\frac{\partial \mathfrak{u}(\boldsymbol{\mu})}{\partial t}, w) + a(\mathfrak{u}(\boldsymbol{\mu}), w; \boldsymbol{\mu}) + \sum_{k=1}^{N_{elem}} \int_{T_k^h} \mathcal{P}(w; \boldsymbol{\mu}) \; \tau \; \mathcal{R}(\mathfrak{u}(\boldsymbol{\mu}); \boldsymbol{\mu}) = l(w), \tag{4.12}$$

where the Streamline-Upwind Petrov-Galerkin stabilization terms are given by

$$\mathcal{P}(w; \boldsymbol{\mu}) = \boldsymbol{v}(\boldsymbol{\mu}) \cdot \nabla w \tag{4.13}$$

$$\mathcal{R}(\mathfrak{u}(\boldsymbol{\mu}); \boldsymbol{\mu}) = \frac{\partial \mathfrak{u}(\boldsymbol{\mu})}{\partial t} + \boldsymbol{v}(\boldsymbol{\mu}) \cdot \nabla \mathfrak{u}(\boldsymbol{\mu}) - \kappa \nabla^2 \mathfrak{u}(\boldsymbol{\mu}). \tag{4.14}$$

and the stabilization parameter τ can be computed by

$$\tau = \widetilde{\kappa} / \mid\mid \boldsymbol{v} \mid\mid_{\ell^2}^2, \tag{4.15}$$

$$\widetilde{\kappa} = \frac{\widetilde{\xi} \mid\mid \boldsymbol{v} \mid\mid_{\ell^2} h}{2}, \tag{4.16}$$

$$\widetilde{\xi} = \coth(Pe) - 1/Pe. \tag{4.17}$$

4.2 Dynamic Sensor Steering Algorithm with Parameterized Velocity Field

All introductory definitions from Section 2.1.1 remain unmodified. It should be emphasized though that contrary to Chapter 3, where $\boldsymbol{m} = (u_0)$, the input vector is now $\boldsymbol{m} = (u_0, \boldsymbol{\mu})$, where u_0 denotes the initial condition and $\boldsymbol{\mu}$ denotes the parameter vector. Moreover, the overall purpose of the Dynamic Sensor Steering Algorithm doesn't change but additionally to the methodology described in Chapter 2 we incorporate a parameterized velocity field into the problem. Thus we take into account that the velocity field in our domain Ω is subject to frequent changes, which was not captured by the initial design of the algorithm stated in Chapter 2. Therefore there are significant changes in the algorithm which will be described in this section.

4.2.1 Offline Stage of the Dynamic Sensor Steering Algorithm with Parameterized Velocity Field

Figure 4-1 shows the offline stage of the algorithm. The details of each step are discussed in the following:

Step 0a. Build reduced-order model $\boldsymbol{g}_r(u_0, \boldsymbol{\mu})$ of physical system $\boldsymbol{g}(u_0, \boldsymbol{\mu})$ and continue working with reduced system throughout the Dynamic Sensor Steering Algorithm. Note that the process of building a reduced-order model, that is now not only depending on an initial condition u_0 but also on a parameter vector $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^P$, becomes a rather complex, challenging task. The modus operandi differs depending on the linearity or non-linearity of the relationship between $\boldsymbol{\mu}$ and $\boldsymbol{v}(\boldsymbol{\mu})$. The linear approach is presented in Section 4.3.

Step 0b. Place the Q mobile sensors at locations $S_1^0, \ldots, S_Q^0$ within Ω and activate them. Note that in *Step 1* of the Online Stage we have to measure two values, the current contaminant concentration denoted by $\boldsymbol{d}^i = (d_1^i, \ldots, d_Q^i)$, where i corresponds to the cycle we are in, and the current parameter vector $\boldsymbol{\mu}^i$, which provides the current velocity field. Later on this issue shall be addressed and some solution approaches

shall be proposed.

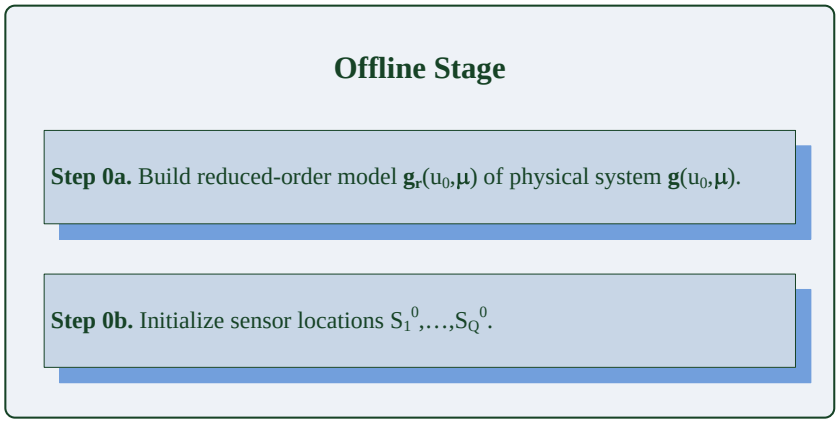

Figure 4-1: Offline stage of the Dynamic Sensor Steering Algorithm with parameterized velocity field.

4.2.2 Online Stage of the Dynamic Sensor Steering Algorithm with Parameterized Velocity Field

Figure 4-2 shows the online stage of the algorithm. The details of each step are discussed in the following:

Step 1. The Q sensors create measurements of the environment at their current locations $S_1^{i-1},\ldots,S_Q^{i-1}$ in *cycle i*. Let $\boldsymbol{d}^i=(d_1^i,\ldots,d_Q^i)$ be the obtained data on the contaminant concentration at the Q different sensor locations and $\boldsymbol{\mu}^i$ be the obtained data on the parameter vector.

Step 2. Compute the current velocity field $\boldsymbol{v}^i$ using $\boldsymbol{\mu}^i$.

Step 3. Based on $\boldsymbol{d}^i=(d_1^i,\ldots,d_Q^i)$, $\boldsymbol{\mu}^i$, $\boldsymbol{v}^i$ and the known reduced forward model $\boldsymbol{g}_r(u_0,\boldsymbol{\mu})$ (or $\boldsymbol{G}_r(u_0,\boldsymbol{\mu})$ in the linear case) we solve the inverse problem to obtain a prediction of the initial value u_0. By applying model order reduction to $\boldsymbol{g}(u_0,\boldsymbol{\mu})$ in the offline stage we made sure to maintain real-time computations throughout the online stage.

Step 4. In many applications some regions Ω_I within the observed domain Ω need to be watched more thoroughly than others at any time, meaning that the physical

process that we want to predict interests us especially in those regions. Ω_I can be fixed, e.g. representing an urban region, or could change from cycle to cycle, e.g. if we wish to track regions of high concentration.

Step 5. We optimize the sensor locations within Ω such that the uncertainty in the prediction for the output regions of interest Ω_I is minimized. Here, the full and reduced forward models depend on both u_0 and $\boldsymbol{\mu}$.

Step 6. The last step in the algorithm steers the mobile sensors from their current locations $S_1^{i-1}, \ldots, S_Q^{i-1}$ to the optimized locations $S_1^i, \ldots, S_Q^i$ within T_{steer} and return to *Step 1.*

4.3 Model Order Reduction Methodology in the Linear Case

In this section we assume that the dependence of the convective velocity field $\boldsymbol{v}(\boldsymbol{\mu})$ on the parameter vector $\boldsymbol{\mu}$ is linear. Now let us observe once more the weak formulation derived in Section 4.1.1 and pay particular attention to the functional $a(\cdot, \cdot; \boldsymbol{\mu})$ which depends on the parameter vector $\boldsymbol{\mu}$. In equation (4.2) we divided functional $a(\cdot, \cdot; \boldsymbol{\mu})$ into two independent functionals $a_1(\cdot, \cdot; \boldsymbol{\mu})$ and $a_2(\cdot, \cdot)$ where the latter corresponds to the diffusive part of the problem and $a_1(\cdot, \cdot; \boldsymbol{\mu})$ shows the dependence on $\boldsymbol{\mu}$ that is inherent to the convective part of the problem. Due to their independence of $\boldsymbol{\mu}$, both $l(\cdot)$ and $a_2(\cdot, \cdot)$ can be computed offline once, leading to the mass matrix $\mathcal{M}$ and the right-hand side vector $\mathcal{F}$ that will not change throughout the problem (assuming constant diffusivity κ). Note that $\mathcal{F}$ is zero in our problem. Conversely,

$$a_1(w, \bar{w}; \boldsymbol{\mu}) = \int_\Omega w \ \boldsymbol{v}(\boldsymbol{\mu}) \cdot \nabla \bar{w} \ d\Omega \quad \forall \ w, \bar{w} \in X \tag{4.18}$$

shows that we cannot compute $a_1(\cdot, \cdot; \boldsymbol{\mu})$ once, because $\boldsymbol{\mu}$ and thus the stiffness matrix $\mathcal{A}$ are subject to frequent changes. Potentially, this could mean that we can no longer compute a reduced-order model in the offline Stage of the Dynamic Sensor Steering Algorithm. The consequence is a loss of executing the algorithm in real-time. However

let us now incorporate (4.1) into (4.18), which yields

$$\int_\Omega w\ \boldsymbol{v}(\boldsymbol{\mu}) \cdot \nabla \bar{w}\ d\Omega = \int_\Omega w\ (\sum_{j=1}^{P} \mu_j\ \boldsymbol{v}_j) \cdot \nabla \bar{w}\ d\Omega \tag{4.19}$$

$$= \sum_{j=1}^{P} \mu_j \int_\Omega w\ \boldsymbol{v}_j \cdot \nabla \bar{w}\ d\Omega. \tag{4.20}$$

From this we can see that the integral over Ω in (4.20) is no longer dependent on $\boldsymbol{\mu}$ and can therefore be computed offline for each $\boldsymbol{v}_j,\ j = 1, \ldots, P$. This means that we now have to pre-compute P different stiffness matrices $\mathcal{A}_1, \ldots, \mathcal{A}_P$ corresponding to the P different velocity fields.

Thus, the continuous representation of the discrete-time system stated in (4.7)-(4.8) becomes

$$\mathcal{M}\ \dot{u}(k) = \mathcal{A}(\boldsymbol{\mu})\ u(k), \tag{4.21}$$

$$d(k) = C\ u(k), \tag{4.22}$$

$$\text{with } \mathcal{A}(\boldsymbol{\mu}) = \sum_{j=1}^{P} \mu_j\ \mathcal{A}_j, \tag{4.23}$$

and from the Backward Euler method for time discretization we obtain

$$D_1^{\boldsymbol{\mu}} = \frac{1}{\Delta t}\mathcal{M} + \sum_{j=1}^{P} \mu_j\ \mathcal{A}_j, \tag{4.24}$$

$$D_2 = \frac{1}{\Delta t}\mathcal{M}. \tag{4.25}$$

If we define the matrices

$$\boldsymbol{A}^{\boldsymbol{\mu}} = \begin{bmatrix} I & 0 & \dots & \dots & 0 \\ -D_2 & D_1^{\boldsymbol{\mu}} & 0 & & \\ 0 & -D_2 & D_1^{\boldsymbol{\mu}} & \ddots & \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & & 0 & -D_2 & D_1^{\boldsymbol{\mu}} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} C & 0 & \dots & \dots & 0 \\ 0 & C & 0 & & \\ \vdots & 0 & C & \ddots & \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & C \end{bmatrix} \tag{4.26}$$

and

$$\boldsymbol{F} = \begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(T) \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(T) \end{bmatrix}, \tag{4.27}$$

the new matrix form of the system becomes:

$$\boldsymbol{A^{\mu}}\ \boldsymbol{u} \ = \ \boldsymbol{F}\ u_0, \tag{4.28}$$

$$\boldsymbol{d} \ = \ \boldsymbol{C}\ \boldsymbol{u}. \tag{4.29}$$

Since we have decoupled $\boldsymbol{\mu}$ from the model matrices, we can compute all necessary reduced-order model matrices offline. The reduced matrices are defined by

$$\mathcal{M}_r \ = \ V^T \mathcal{M} V \tag{4.30}$$

$$\mathcal{A}_{j,r} \ = \ V^T \mathcal{A}_j V \quad j = 1, \ldots, P \tag{4.31}$$

$$C_r \ = \ CV, \tag{4.32}$$

where computation of the basis V was discussed in Section 2.2.2. In the reduced model, (4.24)-(4.25) becomes

$$D^{\boldsymbol{\mu}}_{1,r} \ = \ \frac{1}{\Delta t}\mathcal{M}_r + \sum_{j=1}^{P} \mu_j\ \mathcal{A}_{j,r}, \tag{4.33}$$

$$D_{2,r} \ = \ \frac{1}{\Delta t}\mathcal{M}_r, \tag{4.34}$$

which ultimately leads to the reduced system in matrix form

$$\boldsymbol{A}^{\boldsymbol{\mu}}_r\ \boldsymbol{u}_r \ = \ \boldsymbol{F}_r\ u_0, \tag{4.35}$$

$$\boldsymbol{d}_r \ = \ \boldsymbol{C}_r\ \boldsymbol{u}_r, \tag{4.36}$$

with

$$\boldsymbol{A}_r^{\boldsymbol{\mu}} = \begin{bmatrix} I & 0 & \dots & \dots & 0 \\ -D_{2,r} & D_{1,r}^{\boldsymbol{\mu}} & 0 & & \\ 0 & -D_{2,r} & D_{1,r}^{\boldsymbol{\mu}} & \ddots & \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & & 0 & -D_{2,r} & D_{1,r}^{\boldsymbol{\mu}} \end{bmatrix}, \quad \boldsymbol{C}_r = \begin{bmatrix} C_r & 0 & \dots & \dots & 0 \\ 0 & C_r & 0 & & \\ \vdots & 0 & C_r & \ddots & \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & C_r \end{bmatrix} \tag{4.37}$$

and

$$\boldsymbol{F}_r = \begin{bmatrix} V^T \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \boldsymbol{u}_r = \begin{bmatrix} u_r(0) \\ u_r(1) \\ \vdots \\ u_r(T) \end{bmatrix}, \quad \boldsymbol{d}_r = \begin{bmatrix} d_r(0) \\ d_r(1) \\ \vdots \\ d_r(T) \end{bmatrix}. \tag{4.38}$$

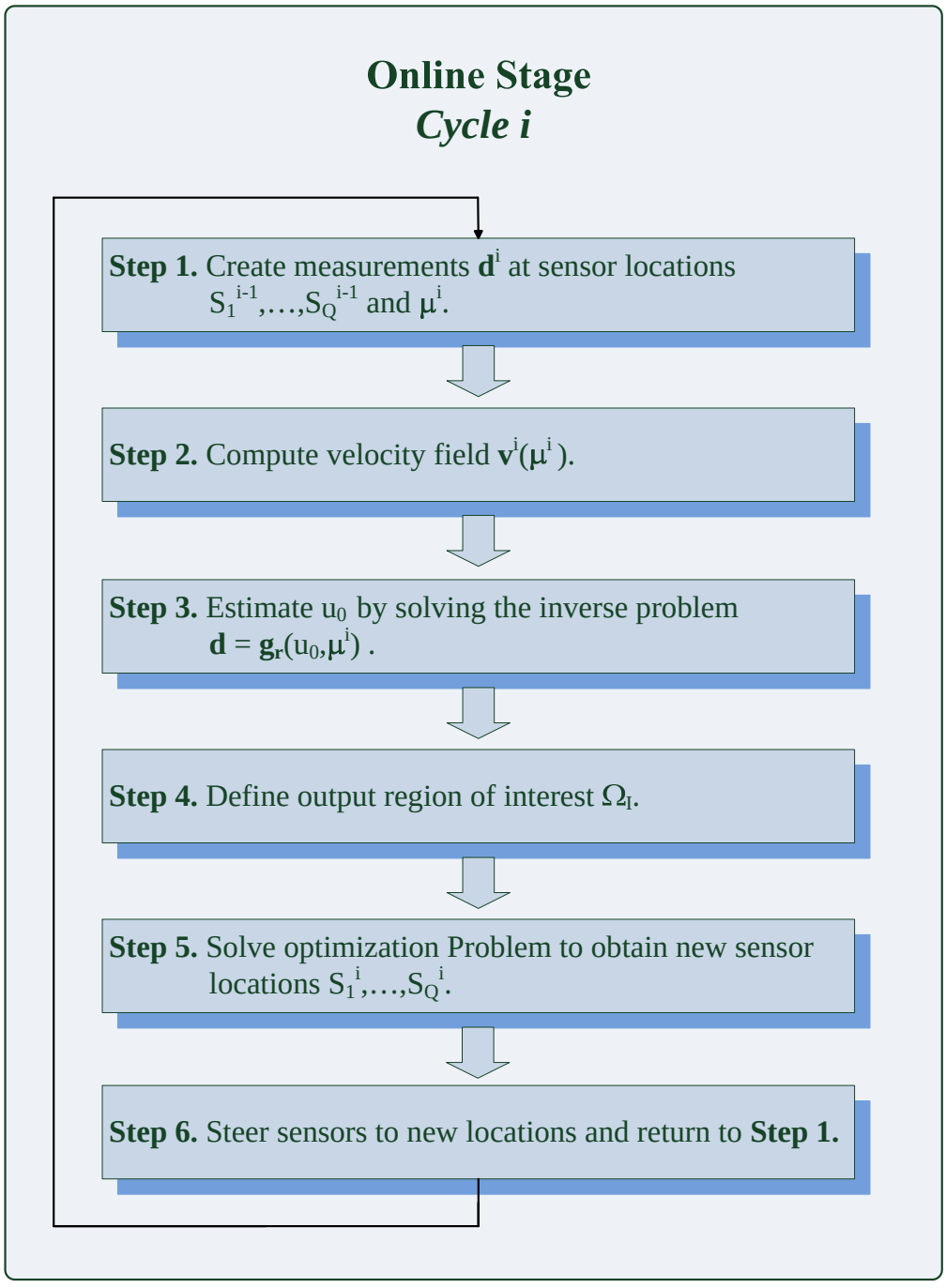

Figure 4-2: One cycle of the online stage of the Dynamic Sensor Steering Algorithm with parameterized velocity field.

Chapter 5

Application to Convection-Diffusion Systems with Parameterized Velocity Fields

In Chapter 5 we present the results obtained by the Dynamic Sensor Steering Algorithm with parameterized velocity field in the linear case described in Section 4.3. First, let us compare reduced-order model and full model performance over a rectangular domain in Section 5.1. Then in Section 5.2, we apply the algorithm to a backwards-facing step domain with velocity fields computed by a Navier-Stokes flow solver.

5.1 Dynamic Sensor Steering over Rectangular Domain

5.1.1 Description of Rectangular Domain

The rectangular domain used throughout this section was described in Section 3.1. Note that boundary conditions as well as the output region of interest remain unchanged and that all following results are obtained assuming two sensors. As previously mentioned in Section 4.1.1 we need to stabilize our system in case the maximum

elemental Péclet number Pe_k, defined in (4.11), exceeds one. Hence, let us choose a diffusivity κ ensuring

$$\max_{1 \leq k \leq N_{elem}} Pe_k < 1 \tag{5.1}$$

by finding an upper bound for (4.11). We consider only low Péclet number flows in order to avoid stabilization because otherwise we would have to pre-compute stabilization matrices which are described in Section 4.1.1 and introduce additional notation here.

5.1.2 A Priori Velocity Fields over Rectangular Domain

The Dynamic Sensor Steering Algorithm in the affine case requires P pre-computed velocity fields and an input parameter $\boldsymbol{\mu} \in \mathbb{R}^P$. In this section the a priori velocity fields shall be represented by polynomial functions in x and y. In particular we want to be able to represent velocity fields modeled by all polynomial functions $f_1, \ldots, f_U$, where U is the number of velocity fields, up to a degree of χ which causes

$$deg(f_u) \leq \chi \quad \text{for } u = 1, \ldots, U. \tag{5.2}$$

If there is no redundancy within $f_1, \ldots, f_U$ then $U = P$. From this we immediately get that $U \geq \chi + 1$ and the polynomials of degree smaller or equal to χ have a resulting vector space of dimension $\chi + 1$. In order to span this space let us assume a monomial basis

$$f_u(x) = x^{u-1} \quad \text{for } u = 1, \ldots, \chi + 1, \tag{5.3}$$

$$f_u(y) = y^{u-1} \quad \text{for } u = 1, \ldots, \chi + 1. \tag{5.4}$$

Then P has to be $2 * (\chi + 1)$ and let us define the corresponding velocity fields as

$$\boldsymbol{v}_k = \begin{bmatrix} f_k(x) \\ 0 \end{bmatrix} \quad \text{for } k = 1, \ldots, \chi + 1 \tag{5.5}$$

$$\boldsymbol{v}_{k+\chi+1} = \begin{bmatrix} 0 \\ -f_k(y) \end{bmatrix} \quad \text{for } k = 1, \ldots, \chi + 1 \tag{5.6}$$

which makes sure that we can adequately model polynomial flow up to an order of χ in both the x and y direction.

For the computational results over the rectangular domain we choose $\chi = 3$, which leads to $P = 8$ a priori velocity fields

$$\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \boldsymbol{v}_2 = \begin{bmatrix} x \\ 0 \end{bmatrix}, \ \boldsymbol{v}_3 = \begin{bmatrix} x^2 \\ 0 \end{bmatrix}, \ \boldsymbol{v}_4 = \begin{bmatrix} x^3 \\ 0 \end{bmatrix}, \tag{5.7}$$

$$\boldsymbol{v}_5 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \ \boldsymbol{v}_6 = \begin{bmatrix} 0 \\ -y \end{bmatrix}, \ \boldsymbol{v}_7 = \begin{bmatrix} 0 \\ -y^2 \end{bmatrix}, \ \boldsymbol{v}_8 = \begin{bmatrix} 0 \\ -y^3 \end{bmatrix}, \tag{5.8}$$

and $\boldsymbol{\mu}$ will be specified at the beginning of each new cycle of the Dynamic Sensor Steering Algorithm and shall be stated accordingly.

5.1.3 Moving Window Illustration

As we have already discussed the performance of reduced-order models in the forward problem, inverse problem and optimization problem over the rectangular domain in Chapter 3 the reader is referred to Sections 3.2 – 3.4. Let us now depict the work flow of the algorithm as the velocity field changes. Each figure below corresponds to a different cycle of the online stage of the Dynamic Sensor Steering Algorithm. The current velocity field that is represented by the vector $\boldsymbol{\mu}$ changes after each cycle. Again we apply sensor constraints stated in (2.49). The radius R is 0.1. We assumed $\kappa = 0.01$, $\Delta t = 0.005$, unchanged Ω_I and the initial condition displayed in 3-2.

Figure 5-1 corresponds to Cycle 1 of the online stage and shows the current con-

taminant concentration at $t_f = 0.2$, the domain of interest Ω_I, the current sensor locations $S_1^0 = (0.7, 0.35)$, $S_2^0 = (0.4, 0.2)$, the sensor constraints and the optimal sensor locations $S_1^1 = (0.7189, 0.2518)$ and $S_2^1 = (0.4992, 0.2123)$.

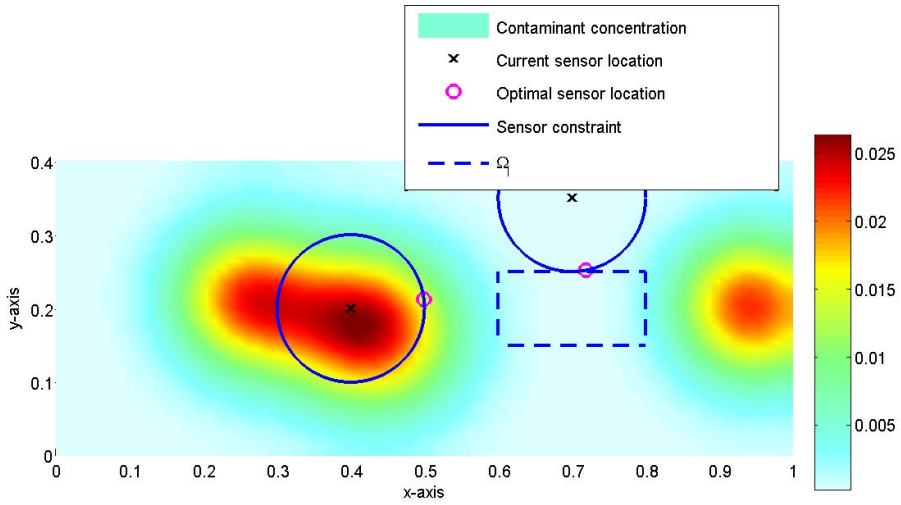

Figure 5-1: Contaminant concentration at $t_f = 0.2$, current and optimal sensor locations of cycle one of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0.75, 0.25, 0, 0, 0, 0, 0, 0\}$.

Figure 5-2 corresponds to Cycle 2 of the online stage and shows the current contaminant concentration at $t_f = 0.4$, the domain of interest Ω_I, the current sensor locations S_1^1 and S_2^1 as computed in Cycle 1, the sensor constraints and the optimal sensor locations $S_1^2 = (0.7392, 0.2029)$ and $S_2^2 = (0.5989, 0.2203)$.

Figure 5-3 corresponds to Cycle 3 of the online stage and shows the current contaminant concentration at $t_f = 0.6$, the domain of interest Ω_I, the current sensor locations S_1^2 and S_2^2 as computed in Cycle 2, the sensor constraints and the optimal sensor locations $S_1^3 = (0.7610, 0.1839)$ and $S_2^3 = (0.6684, 0.2011)$ which is a local minimum.

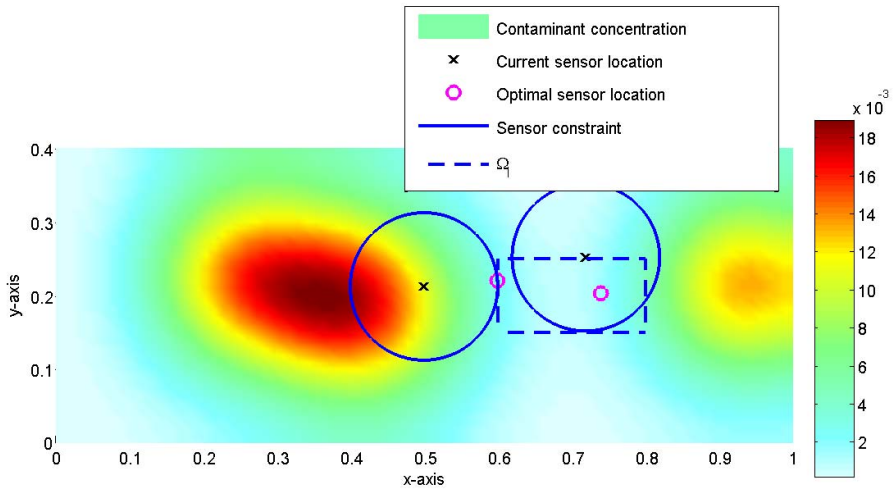

Figure 5-2: Contaminant concentration at $t_f = 0.4$, current and optimal sensor locations of cycle two of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0.25, 0.10, 0.25, 0, 0.15, 0, 0, 0.25\}$.

5.2 Dynamic Sensor Steering over Backward Facing Step

5.2.1 Description of Backward Facing Step Domain

The dimension of the backward facing step domain Ω is best described by (5.9) and the corresponding computational domain has spatial mesh size of $N = 2417$. Figure 5-4 depicts the subdivision of Ω in triangles. We have $\kappa = 0.01$, $\Delta t = 0.0025$ and the output region of interest is Ω_I. The following results were computed assuming two sensors. At the inflow boundary $x = 0$ and $0 \leq y \leq 0.4$ we apply homogeneous Dirichlet conditions and everywhere else homogeneous Neumann boundary conditions are applied.

$$\Omega = \begin{cases} 0.2 \leq y \leq 0.4 & \text{if } 0 \leq x < 0.2 \\ 0 \leq y \leq 0.4 & \text{if } 0.2 \leq x \leq 1 \end{cases} \tag{5.9}$$

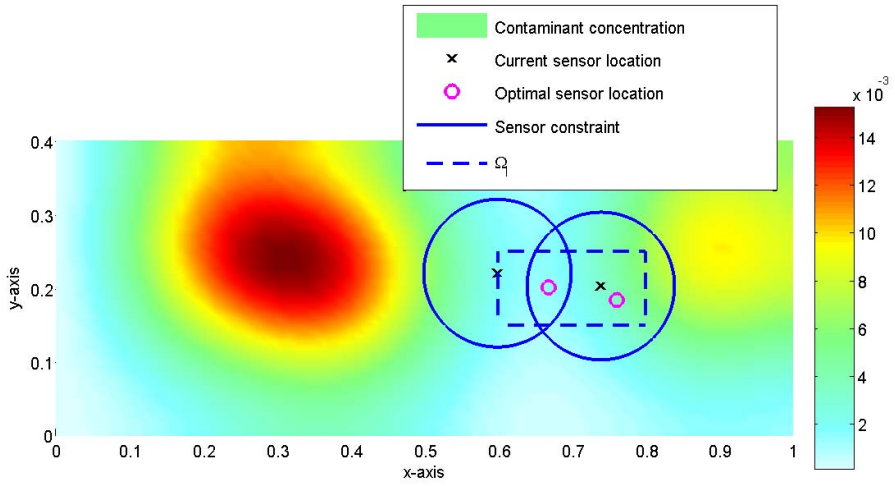

Figure 5-3: Contaminant concentration at $t_f = 0.6$, current and optimal sensor locations of cycle three of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0, 0, 0.25, 0, 0.75, 0, 0, 0\}$.

5.2.2 A Priori Velocity Fields over Backward Facing Step Domain

More realistic velocity fields for the backward facing step domain were computed using a Navier-Stokes flow solver, i.e. the Incompressible Flow Iterative Solution Software (IFISS) [16, 17]. The Navier-Stokes equations can be written as

$$-\nu\nabla^2\boldsymbol{v} + \boldsymbol{v}\cdot\nabla\boldsymbol{v} + \nabla p = \boldsymbol{f}, \tag{5.10}$$

$$\nabla\cdot\boldsymbol{v} = 0, \tag{5.11}$$

where ν corresponds to the kinematic viscosity constant, $\boldsymbol{f}$ corresponds to a source term, , $\boldsymbol{v}$ is the velocity field and p represents pressure. Boundary conditions are given by

$$\boldsymbol{v} = \boldsymbol{w} \text{ on } \Gamma_D, \tag{5.12}$$

$$\nu\frac{\partial\boldsymbol{v}}{\partial n} - \boldsymbol{n}p = \boldsymbol{0} \text{ on } \Gamma_N, \tag{5.13}$$

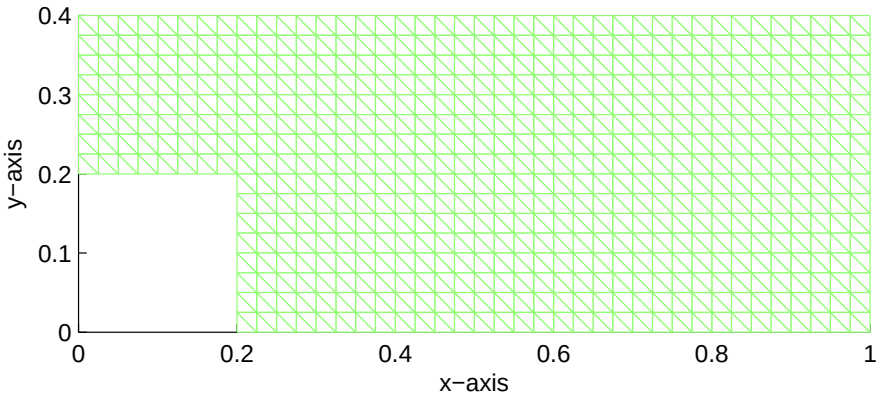

Figure 5-4: Triangular mesh of the backward facing step domain. Note that the mesh used for computations is much finer.

where $\boldsymbol{n}$ is the normal vector, Γ_D corresponds to the boundary where we apply homogeneous Dirichlet conditions and Γ_N corresponds to the boundary where we apply homogeneous Neumann conditions. In particular, a Poiseuille flow is imposed on the inflow boundary, a no-flow (zero velocity) condition is imposed on the top and bottom boundaries and a Neumann condition is applied at the outflow boundary automatically setting the mean outflow pressure to zero. To solve the Navier-Stokes equations stated in (5.12)–(5.13) a finite element method using a quadrilateral element mesh is applied. The resulting non-linear algebraic system is solved using iterative methods, i.e. a hybrid method performing a small number of Picard iterations first for getting a good starting point for Newton's method. For further details on Krylov subspace methods, stabilization and pre-conditioning please refer to [16].

To demonstrate the computational results over the backward facing step domain we choose $P = 4$ obtaining velocity fields $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_P$ corresponding to four different choices of the kinematic viscosities specified in (5.14). Note that the boundary conditions for the Navier-Stokes equations remain unchanged when computing the solution corresponding to different viscosities.

$$\nu_1 = 1/50,\ \nu_2 = 1/75,\ \nu_3 = 1/25,\ \nu_4 = 1/10 \tag{5.14}$$

As the resulting velocity fields can not be described analytically as in Section 5.1.2 let us present the solution of the Navier-Stokes equations graphically for ν_1. Figure 5-5 contains the solution of the velocity field by plotting the streamlines and the solution of the pressure field. In Figure 5-6 we show the velocity at certain nodes in the step domain. Note that $\boldsymbol{\mu}$ will be specified at the beginning of each new cycle of the Dynamic Sensor Steering Algorithm but due to the fact that we can not interpolate between velocity fields with different viscosities only one μ_j will be set to one whereas all the other ones must remain zero.

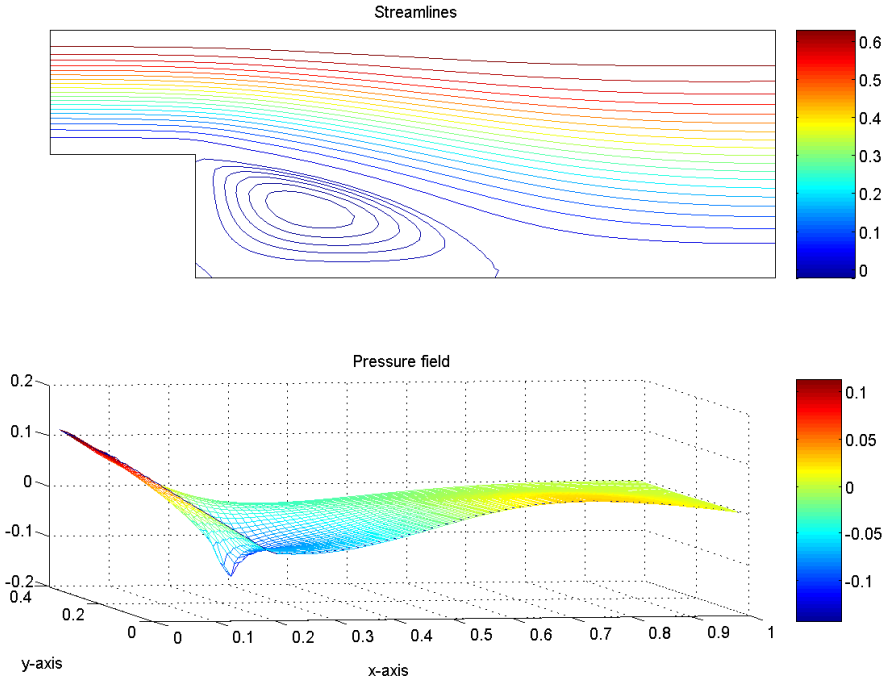

Figure 5-5: The upper plot depicts the streamlines over the backward facing step domain computed using a viscosity $\nu = 1/50$. The lower plot shows the pressure field over the backward facing step domain computed using a viscosity $\nu = 1/50$.

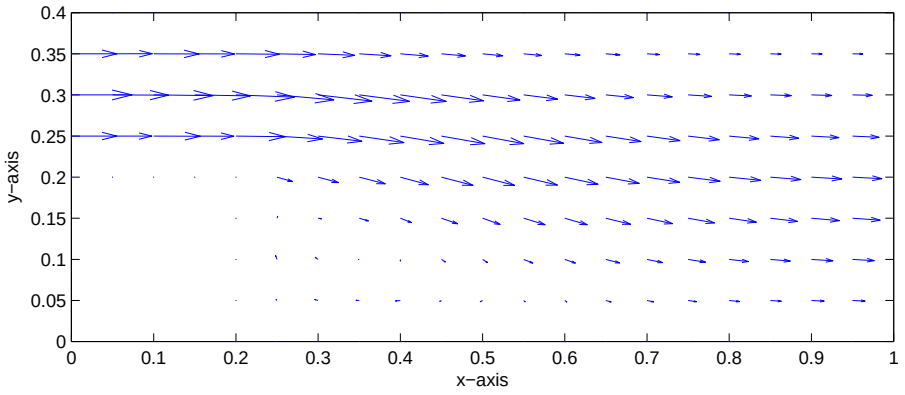

Figure 5-6: Velocity at selected grid points of the backward facing step domain computed using a viscosity $\nu = 1/50$.

5.2.3 Reduced-Order Model Performance in Forward Problem

In this section we assumed a diffusivity $\kappa = 0.01$, $\Delta t = 0.0025$ and an input vector $\boldsymbol{\mu} = \{1, 0, 0, 0\}$ specifying the velocity field

$$\boldsymbol{v}(\boldsymbol{\mu}) = \boldsymbol{v}_{\nu_1}, \tag{5.15}$$

where $\boldsymbol{v}_{\nu_1}$ corresponds to the viscosity ν_1 defined in (5.14). We choose to demonstrate the results using a superposition of five Gaussian functions as the initial contaminant concentration depicted in Figure 5-7. Each Gaussian function is defined as in (3.4). In Figure 5-8 we compare reduced model outputs to full-scale outputs at two sensor locations $S_1 = (0.15, 0.25)$ and $S_2 = (0.45, 0.2)$. Table 5.1 displays various reduced-order model properties, where ε denotes the error compared to the full model defined in (3.5), $c - time$ denotes the computational time to obtain the outputs in seconds, n denotes the size of the reduced-order model, and the significance of $\bar{\lambda}$ and $\bar{\mu}$ is discussed in Section 3.2. Running the forward problem using the full model takes 4.88 seconds. From Table 5.1 we see that the largest reduced-order model of size $n = 185$ is about twelve times faster than the full model and the smallest reduced-order model of size $n = 32$ computes results even 350 times faster. The performance

with respect to error and computational time of each reduced-order model can be viewed in Figure 5-9.

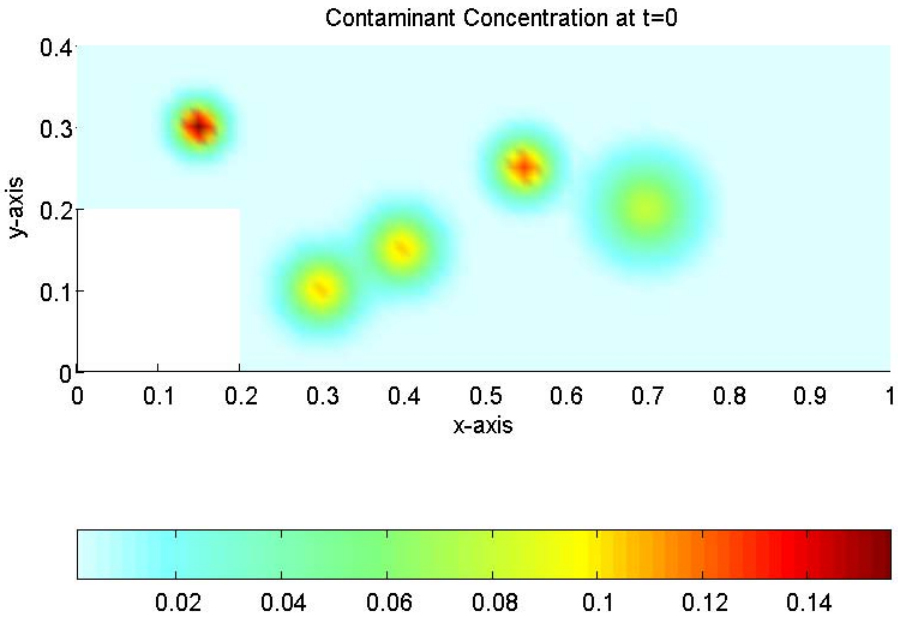

Figure 5-7: Sample test initial condition used to compare reduced model to full model in the forward problem over backward facing step domain.

5.2.4 Solution of the Inverse Problem

As theoretical details have already been discussed in Section 3.3, we will only present the numerical results over the backward facing step domain here. The true initial condition u_0 that was chosen to obtain the observations $\boldsymbol{d}_{obs}$ is depicted in Figure 5-7.

The mean of the initial condition field $\hat{u}_0$ obtained by the full model is referred to as $\hat{u}_0^{full}$ and depicted in the upper plot of Figure 5-10, whereas the mean of the initial condition field computed by the reduced model of size $n = 185$, referred to as $\hat{u}_0^{red}$, is shown in the lower plot of Figure 5-10. It takes 34.62 seconds to compute $\hat{u}_0^{full}$ using the full model and 5.55 seconds to compute $\hat{u}_0^{red}$. When using a reduced-order model of size 32 we can compute $\hat{u}_0^{red}$ about 150 times faster than the full model computations. In Table 5.2 we present some reduced-order model results. The time to compute $\hat{u}_0^{red}$ is referred to as $c - time$, and the error between the estimate of

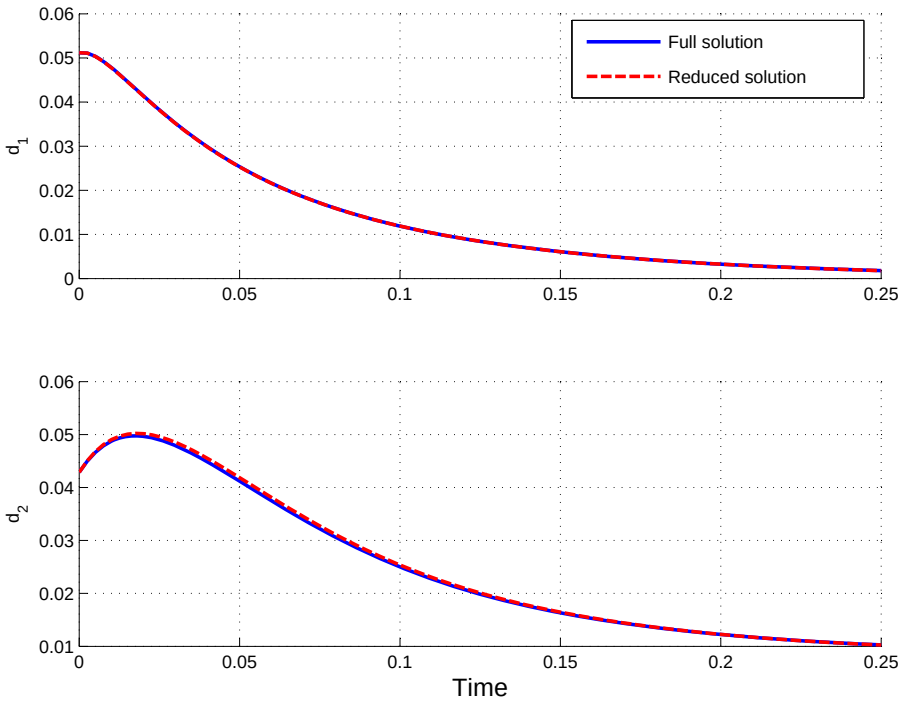

Figure 5-8: A comparison between full ($N = 2417$) and reduced outputs ($n = 185$) for sensors S_1 and S_2 using initial condition depicted in Figure 5-7 with $\kappa = 0.01$, $\Delta t = 0.0025$ and 100 time steps. The error is $\varepsilon = 8.5419e^{-4}$.

the initial condition field using the full and the reduced model is denoted by ε_{u_0} and defined by (3.9).

5.2.5 Reduced-Order Model Performance in Optimization Problem

Again we consider the unconstrained optimization problem (2.47) introduced in Section 2.4 with a velocity field corresponding to $\nu = 1/50$ and $\Delta t = 0.0025$. Due to the lack of senor constraints the sensors are free to move anywhere in the domain Ω defined in (5.9). The output region of interest Ω_I is as stated in (3.3) and we expect the optimal sensor locations to be within Ω_I. For the following presented results we assume the number of sensors $Q = 2$. Solving the optimization problem using the full model of size $N = 2417$ yields optimal locations of $S_1^* = (0.6463, 0.1901)$ and $S_2^* = (0.7100, 0.1998)$ after about 19 hours. The SQP algorithm terminated because

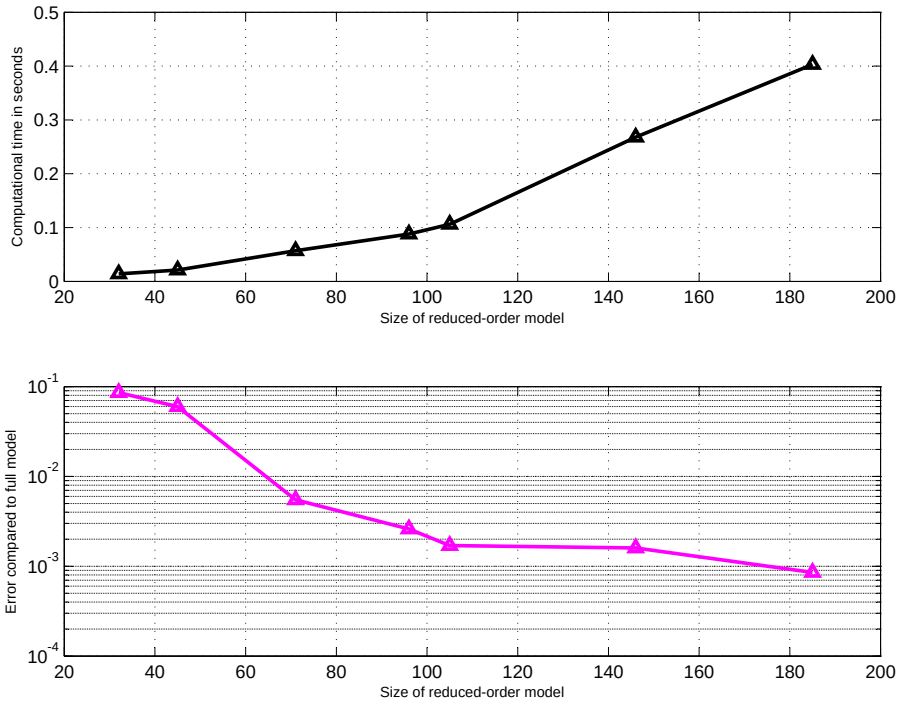

Figure 5-9: The upper plot shows the increase in computational time in seconds of the forward problem as the size of the reduced-order model grows over the backward facing step domain. The lower plot depicts the decrease of the error defined in (3.5) as the size of the reduced-order model increases.

the number of maximum function evaluations was exceeded. The gradient at the optimal solution was reasonably small to account for at least a local minimum. We have observed that when velocity fields are computed by a Navier-Stokes solver, the objective function is less well behaved. Compared to results in Chapter 3 only 73% of random initial locations converge to the above optimum. The remaining 27% either get stuck in local minima that are not within Ω_I (and yield a bigger objective value than the optimal point) or when initialized too close to a boundary do not move at all (we assume that this is due to flatness in the objective function). In Table 5.3 we present the optimization results obtained using reduced-order models of size n. The average optimization error ε_{opt} is defined in (3.10) and the time to compute an optimal solution is denoted by $c - time$. The following conclusions can be drawn from Table 5.3: the smallest reduced-order model does not perform the fastest. Even though a single function evaluation using the reduced-order model of

case	ε	$c-time$	n	$\bar{\lambda}$	$\bar{\mu}$
1	0.0858	0.0140	32	0.5	10^{-4}
2	0.0599	0.0210	45	0.5	10^{-6}
3	0.0055	0.0570	71	0.1	10^{-4}
4	0.0026	0.1060	105	0.1	10^{-6}
5	0.0017	0.0880	96	0.01	10^{-4}
6	0.0016	0.2680	146	0.01	10^{-6}
7	$8.5419e^{-4}$	0.4030	185	0.001	10^{-6}

Table 5.1: Properties of various reduced-order models of a full-scale system with size $N = 2417$ and two output sensors over the backward facing step domain in the forward problem. The error ε is defined in (3.5) and the initial condition used is depicted in Figure 5-7.

case	ε_{u_0}	$c-time$	n	$\bar{\lambda}$	$\bar{\mu}$
1	0.7492	0.224	32	0.5	10^{-4}
2	0.7125	0.320	45	0.5	10^{-6}
3	0.6070	0.664	71	0.1	10^{-4}
4	0.5264	1.533	105	0.1	10^{-6}
5	0.4256	3.000	146	0.01	10^{-6}
5	0.4025	5.551	185	0.001	10^{-6}

Table 5.2: Properties of various reduced-order models of a full-scale system with size $N = 2417$ and two output sensors in the inverse problem.

size $n = 45$ is indeed faster than using a reduced-order model of size $n = 105$, the number of iterations needed in the SQP algorithm is not dependent on the size of the reduced-order model. Hence the fastest result is not necessarily obtained by the smallest reduced-order model. However, in general (when comparing average run times) smaller reduced-order models yield a smaller computational time. Moreover we can observe that increasing the size of the reduced-order model by four leads to an error that is reduced by a factor of three. Hence we suggest that using a smaller reduced-order model in the optimization problem yields a small computational time with a reasonably small error. We omit details about the constrained optimization problem as we observe similar behavior as in the unconstrained case.

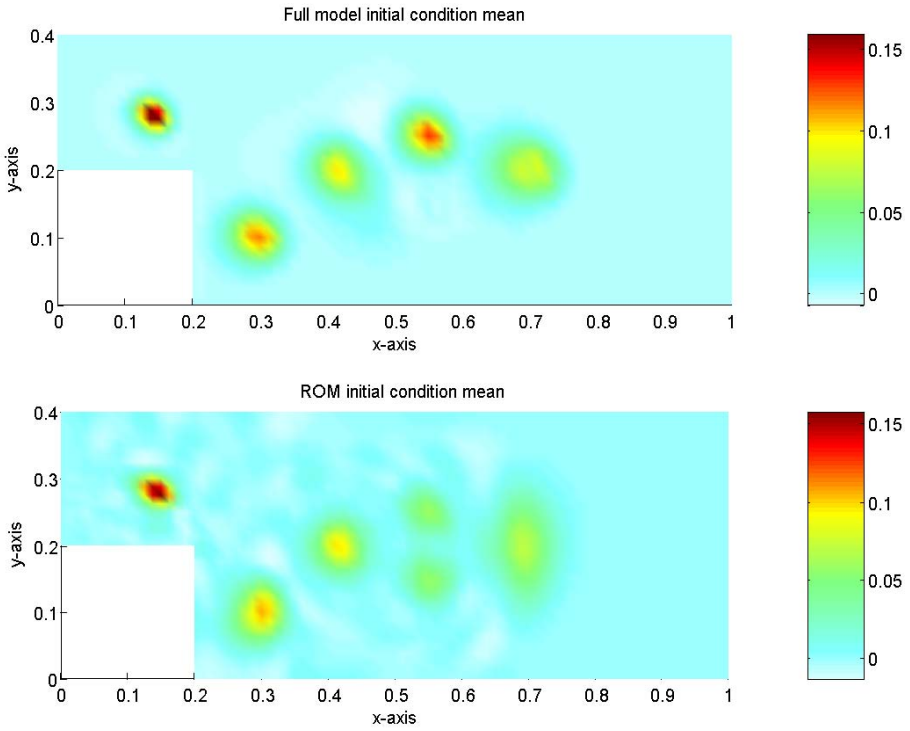

Figure 5-10: The upper plot shows the full model estimate $\hat{u}_0^{full}$ of the initial condition field over the backward facing step domain. The lower plot shows the reduced model estimate $\hat{u}_0^{red}$ of the initial condition field with $n = 185$ which yields $\varepsilon_{u_0} = 0.4025$.

5.2.6 Moving Window Illustration

In this section we demonstrate the work flow over the backward facing step domain. We solve the optimization problem using a reduced-order model of size $n = 185$ and apply a set of sensor constraints stated in (2.49). The radius R that was used here is 0.15.

Figure 5-11 corresponds to Cycle 1 of the online stage and shows the current contaminant concentration at $t_f = 0.2$, the domain of interest Ω_I, the current sensor locations $S_1^0 = (0.3, 0.2)$, $S_2^0 = (0.9, 0.1)$, the sensor constraints and the optimal sensor locations $S_1^1 = (0.4436, 0.2434)$ and $S_2^1 = (0.7714, 0.1772)$. We also include a plot corresponding to the not normalized variance field. In Cycle 1 the variance on the right boundary of Ω_I is very small because one sensor is located there. On the left boundary however the variance is still big as the other sensor is still to far away

case	ε_{opt}	$c-time$	n
1	0.0295	262.061	45
2	0.0195	242.479	105
3	0.0166	278.449	146
4	0.0105	464.317	185

Table 5.3: Results of various reduced-order models in the unconstrained optimization problem over the backward-facing step.

from Ω_I.

Figure 5-12 corresponds to Cycle 2 of the online stage and shows the current contaminant concentration at $t_f = 0.4$, the domain of interest Ω_I, the current sensor locations S_1^1 and S_2^1 as computed in Cycle 1, the sensor constraints and the optimal sensor locations $S_1^2 = (0.5887, 0.2814)$ and $S_2^2 = (0.6992, 0.1987)$. The variance in Ω_I is close to zero now that one sensor is in the output region of interest and the other one is very close to it.

Figure 5-13 corresponds to Cycle 3 of the online stage and shows the current contaminant concentration at $t_f = 0.6$, the domain of interest Ω_I, the current sensor locations S_1^2 and S_2^2 as computed in Cycle 2, the sensor constraints and the optimal sensor locations $S_1^3 = (0.7113, 0.1950)$ and $S_2^3 = (0.6511, 0.1740)$. As we decrease the variance in Ω_I by moving both sensors into this region, the variance at regions that are further away from the sensor locations increases. Once again let us mention that we only want to minimize uncertainty in predictions over Ω_I and hence neglect the rest of the domain.

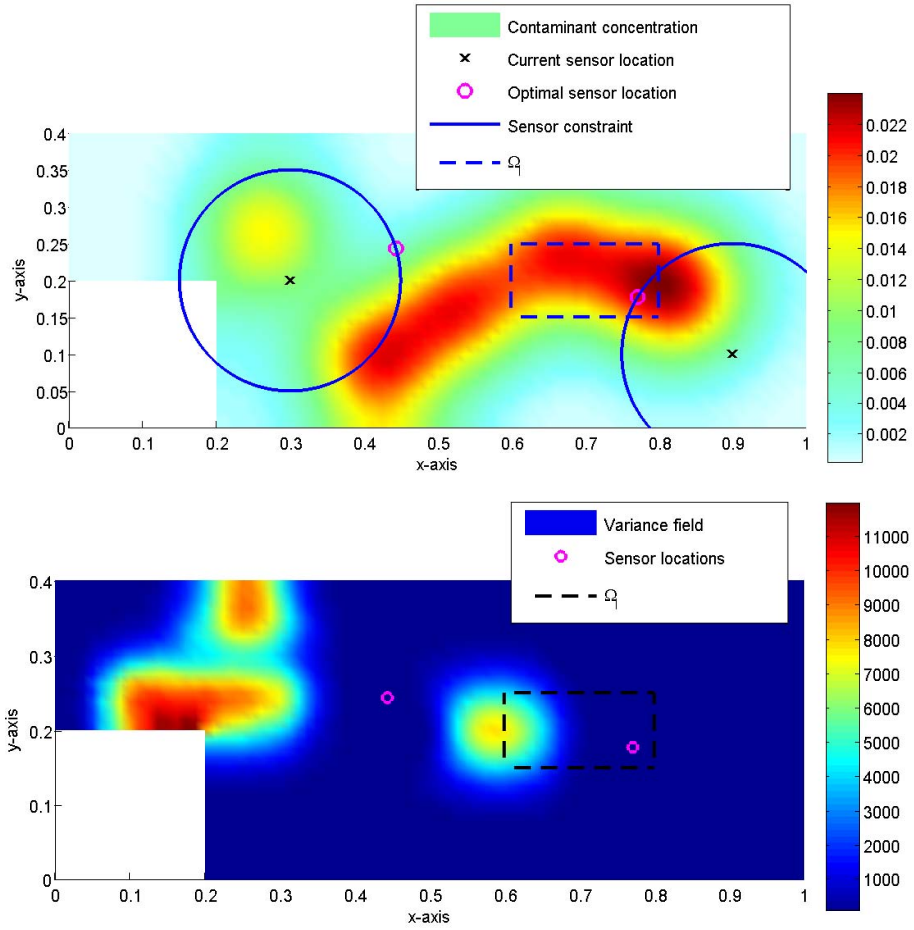

Figure 5-11: The upper figure depicts the contaminant concentration at $t_f = 0.2$, current and optimal sensor locations of cycle one of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{1, 0, 0, 0\}$. The lower figure shows the corresponding variance field based on the current optimal sensor locations over Ω.

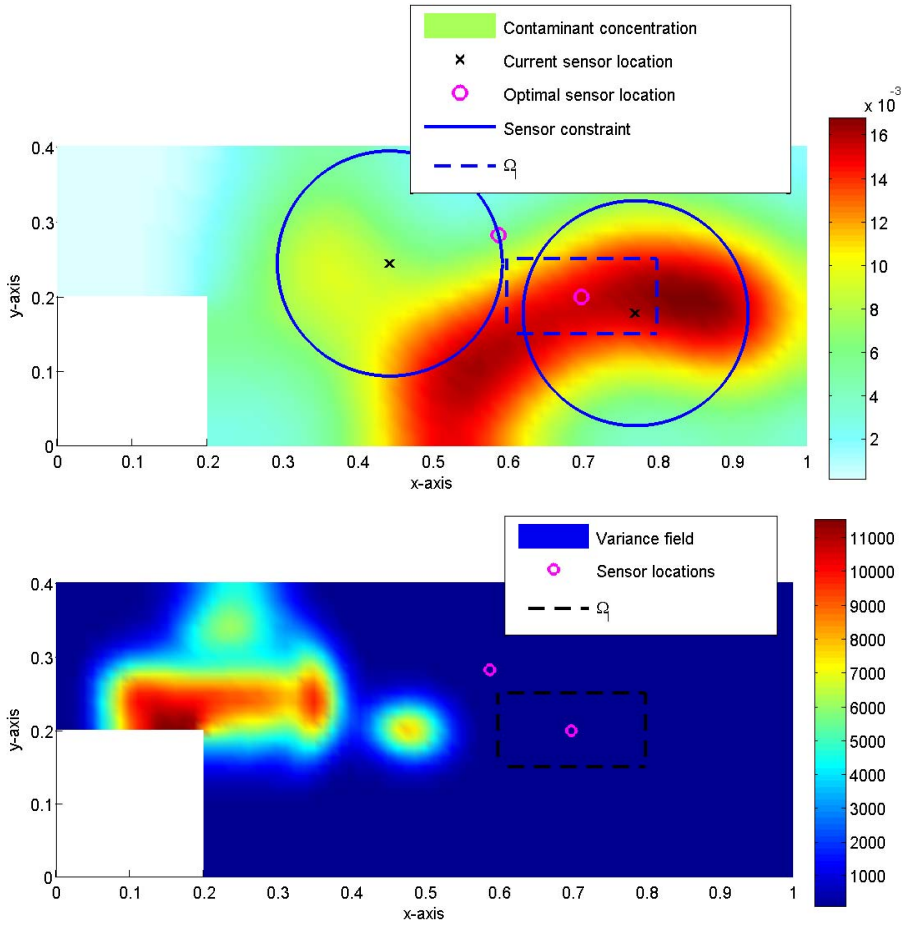

Figure 5-12: The upper figure depicts the contaminant concentration at $t_f = 0.4$, current and optimal sensor locations of cycle two of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0, 0, 1, 0\}$. The lower figure shows the corresponding variance field based on the current optimal sensor locations over Ω.

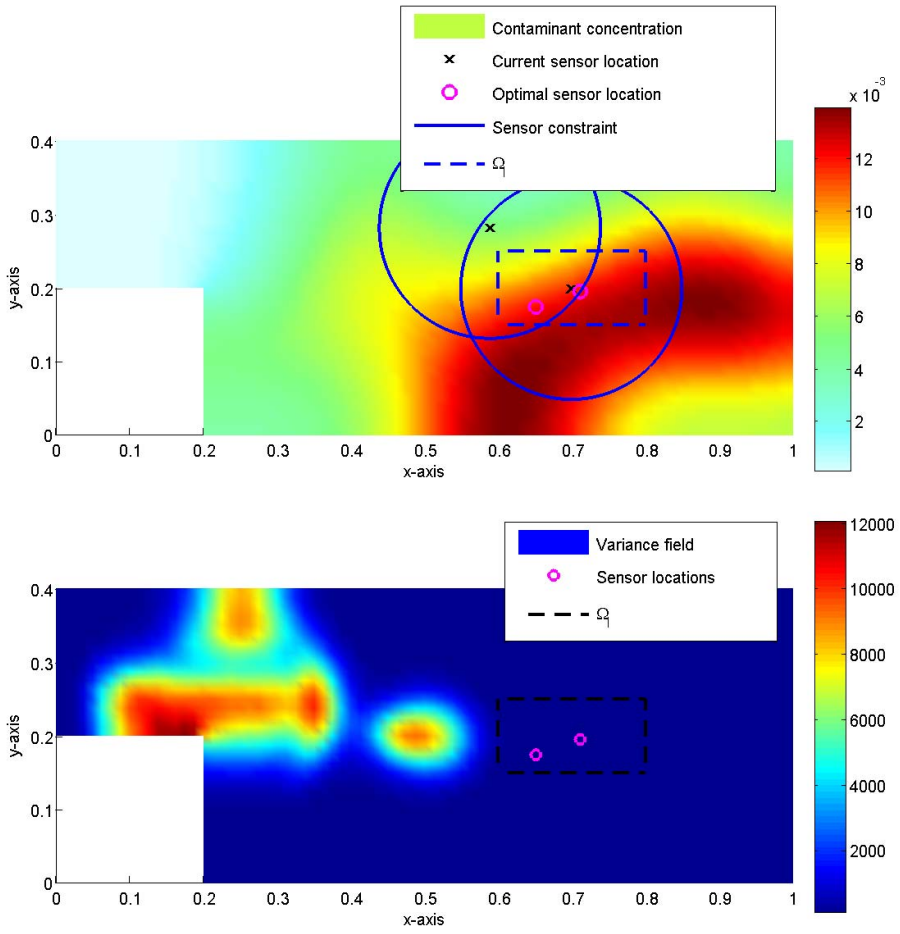

Figure 5-13: The upper figure depicts the contaminant concentration at $t_f = 0.6$, current and optimal sensor locations of cycle three of the online stage with parameterized velocity field by $\boldsymbol{\mu} = \{0, 0, 0, 1\}$. The lower figure shows the corresponding variance field based on the current optimal sensor locations over Ω.

Chapter 6

Conclusions and Future Work

6.1 Summary and Conclusions

In this thesis we develop the methodology for a real-time Dynamic Sensor Steering Algorithm that computes the best-possible prediction of an ongoing physical process by combining model order reduction with a Bayesian approach to inverse problems and with optimization. We apply the algorithm to a contaminant transport problem, paying particular attention to full model versus reduced-order model performance.

The algorithm is divided into two separate stages: the offline stage, which is executed only once, and the online stage, which consists of a measure-predict-optimize-steer cycle and is executed repeatedly. In the offline stage we place mobile sensors in the domain that we wish to observe and we build a reduced-order model of the physical system, using the reduced basis technique, which projects the large-scale system onto a basis in a space of reduced dimensions. This basis is obtained by proper orthogonal decomposition and Hessian-based model reduction.

In the online stage we first obtain observations of the contaminant concentration at current sensor locations by measurements and then we solve an inverse problem yielding a prediction of the initial contamination using a Bayesian approach. The solution to the inverse problem is a probability density function of the initial contaminant concentration. This probability density function provides us with information on how likely each possible contamination scenario is based on the obtained mea-

surements. The knowledge of the contamination distribution provides us with the advantage of being able to instruct fire fighters and police more effectively and if necessary even evacuate people who are in danger. The physical process, i.e. the contamination, changes with time and hence we move the sensors to new locations that minimize the uncertainty in the prediction. We obtain those new locations by solving an optimization problem, with non-linear constraints if sensor constraints are imposed, using the sequential quadratic programming algorithm. After the sensors are steered to their new locations, the online stage repeats itself.

In reality the velocity field, e.g. the wind velocity, is also subject to changes, which alters the way the contamination evolves. Hence we extend the Dynamic Sensor Steering Algorithm to handle a parameterized velocity field, i.e. we pre-compute several different velocity fields, using a Navier-Stokes flow solver and obtain different physical systems according to different velocity fields. In the offline stage we build reduced-order models for each of the physical systems. In the online stage we are now capable of approximating the full model by using the reduced-model, with a velocity field that is any linear combination of the previously computed velocity fields, enabling to model a much broader range of scenarios accurately. The performance in terms of computational time and accuracy of reduced-order models versus full models in the forward problem, the inverse problem and the optimization problem is discussed individually. Due to reduced-order modeling we are able to execute one cycle of the online stage about 260 times faster than with the full model with average relative errors of magnitude $\mathcal{O}(10^{-3})$. The methodology applied to the contaminant transport problem is widely applicable to all sorts of problems where we wish to observe certain phenomena whose location or features are not known before hand. The following section includes some suggestions for improvements and future work.

6.2 Future Work

6.2.1 Recommendations

In this thesis the general methodology was applied to a linear two-dimensional contaminant transport problem. It is a challenging and interesting task to extend the methodology to three-dimensional problems and non-linear problems.

In case the dependence of the velocity field $\boldsymbol{v}(\boldsymbol{\mu})$ on the input parameter $\boldsymbol{\mu}$ in the Dynamic Sensor Steering Algorithm with parameterized velocity field is nonlinear, the velocity field $\boldsymbol{v}(\boldsymbol{\mu})$ could be anything but in order to reduce the level of abstraction we can think of this model as a Navier-Stokes flow solver that computes the velocity field for every point in the domain Ω given $\boldsymbol{\mu}$. Then $\boldsymbol{\mu}$ could contain a specific Reynolds number, boundary conditions, initial conditions, etc. We can extend the algorithm to additionally sample for obtaining current velocity fields in each cycle instead of pre-computing velocity fields. An approach for this extension is proposed in the following section.

In references [34, 35] a technique based on proper orthogonal decomposition is proposed that reconstructs a close approximation to the velocity field. In this thesis we have focused on computing reduced-order models to accurately approximate the full model of a contaminant transport problem while assuming certain velocity fields. Combining the work that was presented in [34, 35] and this thesis can yield even more efficient and more realistic results.

6.2.2 Model Order Reduction Methodology in the Nonlinear Case

We consider the more general case where the dependence of parameter vector $\boldsymbol{\mu}$ on the convective velocity field $\boldsymbol{v}(\boldsymbol{\mu})$ is non-linear. In order to avoid online re-computation of the reduced-order model we need to obtain an affine decomposition of the bilinear functional $a_1(w, \bar{w}; \boldsymbol{\mu})$ in terms of products of parameter-dependent coefficients, $\Phi_m^{\boldsymbol{v}}(\boldsymbol{\mu})$ for $1 \leq m \leq n$ (computed online), and parameter-independent bilinear forms

$a_1^m(w, \bar{w})$ for $1 \leq m \leq n$ (computed offline). The resulting bilinear functional can be written as

$$a_1(w, \bar{w}; \boldsymbol{\mu}) = \sum_{m=1}^{n} \Phi_m^{\boldsymbol{v}}(\boldsymbol{\mu}) a_1^m(w, \bar{w}). \tag{6.1}$$

In order to achieve this we introduce a parameter sample that adequately covers the parameter space $\mathcal{D}$,

$$S_K = \{\boldsymbol{\mu}_1 \in \mathcal{D}, \ldots, \boldsymbol{\mu}_K \in \mathcal{D}\}. \tag{6.2}$$

Then we compute solution snapshots

$$\mathcal{S}_K^u = \{u(\boldsymbol{\mu}_k) \text{ for } 1 \leq k \leq K\}. \tag{6.3}$$

Then we apply POD to the snapshots $\mathcal{S}_K^u$ and hence obtain the basis functions for our reduced-order model $\{\zeta_m\}_{m=1}^n$ and define the approximation space $V = span\{\zeta_1, \ldots, \zeta_n\}$. For the following discussion let us assume the basis is orthonormalized. Now we also generate snapshots of the velocity field,

$$\mathcal{S}_K^{\boldsymbol{v}} = \{\boldsymbol{v}(\boldsymbol{\mu}_k) \text{ for } 1 \leq k \leq K\}. \tag{6.4}$$

Then we use the "Best Points" Interpolation Method (BPIM) introduced in [40] to construct a set of interpolation points $\{\boldsymbol{z}_m^{\boldsymbol{v}}\}_{m=1}^n$ and the basis functions $\{\psi_m^{\boldsymbol{v}}\}_{m=1}^n$ such that we can define a coefficient-function approximation, $\boldsymbol{v}_n(\boldsymbol{x}, \boldsymbol{\mu})$, for the convective velocity field $\boldsymbol{v}(\boldsymbol{x}; \boldsymbol{\mu})$ as follows,

$$\boldsymbol{v}_n(\boldsymbol{x}; \boldsymbol{\mu}) = \sum_{m=1}^{n} \boldsymbol{v}(\boldsymbol{z}_m^{\boldsymbol{v}}; \boldsymbol{\mu})\ \psi_m^{\boldsymbol{v}}, \tag{6.5}$$

where $\boldsymbol{x} = (x, y) \in \Omega$. Then $\{\psi_m^{\boldsymbol{v}}\}_{m=1}^n$, with $\psi_j^{\boldsymbol{v}}(\boldsymbol{z}_i) = \delta_{ij}$, where δ_{ij} is the Kronecker symbol, can thus be obtained by

$$\zeta_i(\boldsymbol{x}) = \sum_{m=1}^{n} \zeta_i(\boldsymbol{z}_m)\ \psi_m^{\boldsymbol{v}}(\boldsymbol{x}), \quad 1 \leq i \leq n. \tag{6.6}$$

Then the discrete version of the bilinear functional $a_1(w, \bar{w}; \boldsymbol{\mu})$ is given by

$$A^n_{1\ ij} = \sum_{m=1}^{n} v_1(\boldsymbol{z}^{\boldsymbol{v}}_m; \boldsymbol{\mu}) \int_\Omega \psi^{\boldsymbol{v}}_m(\boldsymbol{x}) \frac{\partial \zeta_j(\boldsymbol{x})}{\partial x} \zeta_i(\boldsymbol{x})\ d\Omega + \sum_{m=1}^{n} v_2(\boldsymbol{z}^{\boldsymbol{v}}_m; \boldsymbol{\mu}) \int_\Omega \psi^{\boldsymbol{v}}_m(\boldsymbol{x}) \frac{\partial \zeta_j(\boldsymbol{x})}{\partial y} \zeta_i(\boldsymbol{x})\ d\Omega \tag{6.7}$$

$$= \sum_{m=1}^{n} v_1(\boldsymbol{z}^{\boldsymbol{v}}_m; \boldsymbol{\mu})\ a^m_{1x}(\zeta_i, \zeta_j) + \sum_{m=1}^{n} v_2(\boldsymbol{z}^{\boldsymbol{v}}_m; \boldsymbol{\mu})\ a^m_{2y}(\zeta_i, \zeta_j) \quad 1 \le i, j \le n, \tag{6.8}$$

where $\boldsymbol{v}(\boldsymbol{z}^{\vec{\boldsymbol{v}}}_m; \boldsymbol{\mu}) = (v_1(\boldsymbol{z}^{\boldsymbol{v}}_m; \boldsymbol{\mu}), v_2(\boldsymbol{z}^{\boldsymbol{v}}_m; \boldsymbol{\mu}))$. Now we can solve the weak form (4.2) in the reduced basis space V. The same affine decomposition described above for $a_1(w, \bar{w}; \boldsymbol{\mu})$ should also be used for all the SUPG stabilization terms in (4.2).

When actually implementing this method we need to address the following issue. Throughout the algorithm the sensors that measure the ongoing contamination steer to locations which minimize uncertainty in the prediction, however the functionality of the best interpolation points that we use to approximate $a_1(\cdot, \cdot; \boldsymbol{\mu})$ is completely unrelated to the sensor locations. Thus we can either introduce a second set of sensors that merely depends on the best interpolation points or we can only work with one set of sensors and try to find a reasonable balance between the locations that minimize uncertainty and locations of best interpolation points. Although it is more costly to deploy another set of sensors the results will to be more accurate. However it should be considered that when working with one set of sensors due to limited resources finding the optimal trade-off between minimizing uncertainty in the prediction and approximating the functional $a_1(\cdot, \cdot; \boldsymbol{\mu})$ is a very interesting problem by itself and should be examined in future work.

Bibliography

[1] A. Altmann-Dieses, J. P. Schloeder, H. G. Bock, and O. Richter. Optimal experimental design for parameter estimation in column outflow experiments. *Water Resources Research*, 2002.

[2] A. Atkinson and A. Donev. *Optimum Experimental Designs.* Oxford University Press, New York, 1992.

[3] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations. *C. R. Acad. Sci. Paris, Série I*, 339(9):667–672, November 2004.

[4] O. S. Bashir. Hessian-based model reduction and applications to initial-condition inverse problems. Master's thesis, Massachusetts Institute of Technology, 2007.

[5] S. Bégot, E. Voisin, P. Hiebel, E. Artioukhine, and J. M. Kauffmann. D-optimal experimental design applied to a linear magnetostatic inverse problem. *IEEE Transactions on Magnetics*, 2002.

[6] J. W. Berry, L. Fleischer, W. E. Hart, C. A. Phillips, and J. P. Watson. Sensor placement in municipal water networks. *Journal of Water Resources Planning and Management*, June 2005.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[8] A.N. Brooks and T.J.R. Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, pages 199–259, 1990.

[9] C. G. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications, 6*, pages 76–90, 1970.

[10] T. Bui-Thanh. *Model-Constrained Optimization Methods for Reduction of Parameterized Large-Scale Systems.* PhD thesis, Massachusetts Institute of Technology, May 2007.

[11] D. Caron, A. Das, A. Dhariwal, L. Golubchik, R. Govindan, D. Kempe, C. Oberg, A. Sharma, B. Stauffer, G. Sukhatme, and B. Zhang. A generic multi-scale modeling framework for reactive observing systems: An overview. In *ICCS 2006, Part III, LNCS 3993*, pages 514–521, 2006.

[12] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statis. Sci.*, 1995.

[13] H. L. Choi, J. P. How, and J. A. Hansen. Ensemble-based adaptive targeting of mobile sensor networks. In *Proceedings of the 2007 American Control Conference*, 2007.

[14] A.E. Deane, I.G. Kevredidis, G.E. Karniadakis, and S.A. Orszag. Low-dimensional models for comples geometry flows: Application to grooved channels and circular cylinders. *Phys. Fluids*, pages 3(10):2337–2354, 1991.

[15] J. Donea and A. Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons Ltd, 2003.

[16] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics*. Oxford University Press, New York, 2005.

[17] H. C. Elman, A. Ramage, and D. J. Silvester. Ifiss: a matlab toolbox for modelling incompressible flow. 2005.

[18] P. Feldmann and R.W. Freund. Efficient linear circuit analysis by pad approximation via the lanczos process. In *IEEE Transactions on Computer-Aided Design of Integrated Curcuits and Systems*, pages 14:639–649, 1995.

[19] R. A. Fletcher. New approach to variable metric algorithms. *Computer Journal, 13*, pages 317–322, 1970.

[20] D. Galbally. Nonlinear model reduction for uncertainty quantification in large-scale inverse problems: Application to nonlinear convection-diffusion-reaction equation. Master's thesis, Massachusetts Institute of Technology, 2008.

[21] K. Gallivan, E. Grimme, and P. Van Dooren. Padé approximation of large-scale dynamic systems with Lanczos methods. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, 1994.

[22] O. Ghattas, V. Akcelik, P. Flath, J. Hill, B. van Bloemen Waanders, L. Wilcox, and K. Willcox. Model calibration: A bayesian inverse approach. 2007.

[23] D. A. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation, 24*, pages 23–26, 1970.

[24] H. González-Banos and J. C. Latombe. A randomized art-gallery algorithm for sensor placement. *SCG*, June 2001.

[25] M. A. Grepl. *Reduced-Basis Approximations and A Posteriori Error Estimation for Parabolic Partial Differential Equations*. PhD thesis, Massachusetts Institute of Technology, May 2005.

[26] E. Grimme. *Krylov Projection Methods for Model Reduction*. PhD thesis, Coordinated-Science Laboratory, University of Illinois at Urbana-Champaign, 1997.

[27] C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. *ICML*, 2005.

[28] E. Haber. Numerical methods for optimal experimental design of large-scale ill-posed problems. Technical report, January 2008.

[29] P. Holmes, J.L. Lumley, and G. Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, 1996.

[30] J. P. Kaipio and E. Somersalo. *Computational and Statistical Methods for Inverse Problems*. Springer, 2005.

[31] G. Kerschen, J.C. Golinval, A.F. Vakakis, and L.A. Bergmann. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems. *Springer, Nonlinear Dynamics*, pages 41:147–169, 2005.

[32] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11(2):431–444, 1963.

[33] The MathWorks, Natick, MA. *MATLAB Optimization Toolbox 3. User's Guide*, 2007.

[34] P. Mokhasi and D. Rempfer. Opimized sensor placement for urban flow measurement. *American Physical Society, 56th Annual Meeting of the Division of Fluid Dynamics*, 2003.

[35] P. Mokhasi and D. Rempfer. Optimized simulation of contaminant dispersion in urban flows. *American Physical Society, 57th Annual Meeting of the Division of Fluid Dynamics*, 2004.

[36] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, (1):17–31, August 1981.

[37] K. Mosegaard and A. Tarantola. Probabilistic approach to inverse problems. In *International Handbook of Earthquake & Engineering Seismology (Part A)*, pages 237–265. Academic Press, 2002.

[38] M. Mossberg. Optimal experimental design for identification of viscoelastic materials. *IEEE Transactions and Control Systems Technology*, 2004.

[39] N. C. Nguyen. *Reduced-Basis Approximation and A Posteriori Error Bounds for Nonaffine and Nonlinear Partial Differential Equations: Application to Inverse Analysis*. PhD thesis, National University of Singapore, Singapore-MIT Alliance, July 2005.

[40] N. C. Nguyen, A. T. Patera, and J. Peraire. A "best points" interpolation method for efficient approximation of parametrized functions. *Int. J. Numer. Meth. Engng.*, 2007. Accepted.

[41] N. C. Nguyen, K. Veroy, and A. T. Patera. Certified real-time solution of parametrized partial differential equations. In S. Yip, editor, *Handbook of Materials Modeling*, pages 1523–1558. Springer, 2005.

[42] A. K. Noor and J. M. Peters. Reduced basis technique for nonlinear analysis of structures. *AIAA Journal*, 18(4):455–462, 1980.

[43] J. O'Rourke. *Art-Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.

[44] A. Ostfeld and E. Salomons. Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management*, 2004.

[45] D. Otte. *Development and Evaluation of Singular Value Analysis Methodologies for Studying Multivariate Noise and Vibration Problems*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 1994.

[46] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation, 24*, pages 647–656, 1970.

[47] L. Sirovich. Turbulence and the dynamics of coherent structures. part 1: Coherent structures. *Quarterly of Applied Mathematics*, 1987.

[48] K. C. Sou, A. Megretski, and L. Daniel. A quasi-convex optimization approach to parameterized model-order reduction. In *IEEE/ACM Design Automation Conference*, June 2005.

[49] J. Staar. *Concepts for Reliable Modeling of Linear Systems With Application to On-Line Identification of Multivariate State Space Descriptions*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 1982.

[50] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, Philadelphia, 2005.

[51] K. Veroy, C. Prud'homme, D. V. Rovas, and A. T. Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations (AIAA Paper 2003-3847). In *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*, June 2003.

[52] M. A. Woodbury. Inverting modified matrices. *Memorandum Rept. 42, Statistical Research Group, Princeton University*, 1950.