The Report Committee for Haocheng An
certifies that this is the approved version of the following report:

# Kalman filtering for state estimation of an advection diffusion PDE from sparse observation

APPROVED BY

SUPERVISING COMMITTEE:

Omar Ghattas, Supervisor

Clint N. Dawson

# Kalman filtering for state estimation of an advection diffusion PDE from sparse observation

**by**

**Haocheng An**

**REPORT**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**Master of Science in Computational Science, Engineering and Mathematics**

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2019

Dedicated to my parents for their selfless love and support

# Acknowledgments

My journey as a master student in the Oden Institute for Computational Engineering and Sciences has been a short but meaningful one. I have deepened my understanding of many materials in undergraduate and learned a lot new from this interdisciplinary program. It is with the great help of people around me that I can make it. I would like to express my most sincere gratitude in this acknowledgment.

First, I would like to say thank you to Dr. Omar Ghattas. He motivates me to think, to learn and to examine this topic. Also, he helps me settle down the scope of the problem for investigation base on my ability.

Then I would like to show my gratitude to Josh. He is very caring and offers me much guidance on deriving the algorithms, implementing in hIPPYlib and creating pictures/ graphs. It is with his help that I know what a true report should look like and what kind of metrics are really valuable.

Next, I would like to thanks to my parents. They sincerely support my study to chase my dream since I was a child. Now, they encourage me when I get stuck and never afraid of failures.

Also, I would like to appreciate Prof. Dawson. As the reader, he offers much useful advice on polishing this report.

I know I am far from an expert in this field and this report is only a starting point. It is with the help of all these people that make it happen.

# Kalman filtering for state estimation of an advection diffusion PDE from sparse observation

Haocheng An, M.S.C.S.E.M
The University of Texas at Austin, 2019

Supervisor: Omar Ghattas

Kalman filter is an important algorithm used in control theory. It takes an initial state as input and a series of observations over time and output the hidden state.

The advection-diffusion equation is a PDE that characterizes the combination effect of advection and diffusion of a given object in the solvent. Such a problem is within the domain that the Kalman filter can solve.

In this report, I will first derive the Kalman filter algorithm, then examine its application to an advection-diffusion equation. I will use different metrics to quantify the numerical performance of the algorithm. The contribution of this report lies in the combination of the Kalman filter algorithm with the advection equation. Also, an ample amount of graphs that can visually tell us the evolving trend of the state.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The Partial differential equation (PDE) plays an important role in mathematical modeling in science and engineering. Formally, it is defined as[8]

a differential equation that contains unknown multivariable functions and their partial derivatives.

PDE serves as an implicit function that describes the relationship between different variables and we can learn a lot of useful information from it.

hIPPYlib is a mathematical Python library that based on Fenics' discretization and PETSc for scalable performance. The library implements many operators required for solving inverse problems in matrix-free form.

In this report, under the guidance of numerical PDE solving algorithms, we will pick out necessary operators from hIPPYlib, with the help of numpy, re-organize the operator, implement the algorithm to solve the problem numerically.

## 1.1 Motivation

With the complex relations between variables in PDEs, very few of them can be solved explicitly. It motivates us to think of numerical solutions. By discretizing the domain space and extracting sample data points, we know how do some variables evolve with the change of others. However, such a construction may be hindered by other factors. For example, we may accidentally pick a data point that with very low probability, we may have very high variance and thus interferes the main evolving stream. Also, the previous knowledge at times can also guide the next state. It is very natural to come up with an idea of coordinating the current observation point and the prior knowledge together to achieve better performance. In this paper, we would like to review the Kalman filter algorithm, which exactly combining the aforementioned two parts and numerically estimate the hidden state.

In this report, we would like to consider a 2 dimension advection-diffusion equation. In order to intuitively understand the equation, let us consider the following scenario qualitatively: a river is running with some chemicals poured down at the source of the river (Assume the chemicals can be fully dissolved in water and does not have chemical reactions with water). The observer stands somewhere in the middle of the river and observes the concentration of the chemical. On one hand, as the concentration of the chemical is high in the place you initially placed the chemicals, it should spread around the water to lower it. This process is called $diffusion$ . On the other hand, the water is running inside, so the chemical will go along the way and thus

affect the concentration at the observation point. This process is called the *advection*. The concentration of the substance is jointly affected by these two and are independent of each other. so we have the following PDE:

The State equation [5]:

$$\frac{\partial \boldsymbol{u}}{\partial t} - \underbrace{\kappa \Delta \boldsymbol{u}}_{\text{diffusion}} + \underbrace{\boldsymbol{v} \cdot \nabla \boldsymbol{u}}_{\text{advection}} = 0 \text{ in } \mathcal{X} \times (0, T) \tag{1.1}$$

With initial and boundary condition:

$$u(\cdot, 0) = \boldsymbol{m_0} \text{ in } \mathcal{X} \tag{1.2}$$

$$\kappa \nabla u \cdot \boldsymbol{n} = 0 \text{ on } \partial\mathcal{X} \times (0, T) \tag{1.3}$$

Specifically, In Equation 1.1, $\boldsymbol{u}$ is the variable that we are interested in, refer to the concentration in the description above.

$\kappa$ describes the diffusivity, which is a proportionally constant between the molar flux and the gradient in concentration of the given species[7].

$\boldsymbol{v}$ is the velocity field[7], which describes how fast the object moves, just like the water speed mentioned above.

$\mathcal{X}$ specifies the state space where the variable $\boldsymbol{u}$ lives and $T$ specifies the time-space.

$\Delta$ is the Laplace operator. Specifically, $\Delta \boldsymbol{u}$ is the second-order derivative with respect to space.

$\nabla$ is the gradient operator and $\nabla \boldsymbol{u}$ represents the concentration gradient.

Equation 1.2 reflects the distribution of the concentration, i.e at time 0.

In Equation 1.3 , $\partial\mathcal{X}$ reflects the boundary of the state space, $\boldsymbol{n}$ is the normal vector that indicates the direction of the plane.

In this report, I will discretize the equation 1.1 for computation. For the non-derivative terms in it, the discretization is done by taking the value at a specific time and location. For the derivative terms we can use the finite difference method in time and finite element method(FEM) in space.

Then, we combine the observation and state at previous timestep by Hidden markov model(HMM) to estimate the next iteration. In the linear transformation settings, it is also known as kalman filter.

## 1.2 Report Organization

This report is composed of four chapters.

Chapter 1 is the current chapter. This chapter gives a high-level description of the PDE problem and introduces the structure of the report.

Chapter 2 is about the mathematical background of the Kalman Filter. It will discuss statistical and PDE background respectively. In the end, we'll combine these two and introduce the Kalman Filter algorithm. We will explicitly derive the algorithm from 0 to 1 and using induction to show the latter steps follows. The actual implementation of the algorithm will serve as the summary part of the whole chapter.

Chapter 3 is about the numerical experiments. We apply the algorithm introduced in the previous chapter to advection-diffusion function. We start from the simplest case where prior, observation noise and process noise are all independent. Then we explore more on the dependent case, examine how the dependence affects the prediction.

Chapter 4 is the conclusion and future work, where we will summarize the work first and then see what are possible options of advancing the algorithms or apply the algorithm to broader settings or we may develop some parallel way to run the algorithm efficiently.

Also to keep the consistent with the mathematical derivation and computation implementation, throughout the report, unless specified, the initial state or the first element is assumed to have index 0. For an array of length $n$, we assume the indices are $0, 1, \cdots n-1$.

# Chapter 2

# Mathematical Background

Solving the Bayesian inference inverse problem is not easy. It requires some prerequisite knowledge. Now, let us explore some statistical and PDE properties and then gradually move toward the actual algorithm.

## 2.1 Statistical Background

### 2.1.1 Bayes' Theorem

Bayes' theorem builds relations between random variables in conditional probability. The basic case of it is stated as below:

**Theorem 1.** Bayes' Theorem

Given random variale A and B, we have

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \propto P(B|A) * P(A) \tag{2.1}$$

With the formula shown in Theorem 1, in the Bayesian inference settings, we would label $P(A)$ as *prior*, $P(B|A)$ as *likelihood*, and $P(B)$ as *posterior* and we will use these terms throughout the whole report.

#### 2.1.1.1 Infinite Dimension Bayes' Theorem

The previous section illustrates finite dimensional settings of Bayes' theorem. Intuitively, it means transforming from prior to posterior through likelihood. In this report, we mainly deal with infinite dimension problem under Gaussian measure. For the initial step, initial state ($\boldsymbol{s}_0$) can serve as prior and the observation at the next iteration can serve as likelihood. Combination of these two yields the prediction $\boldsymbol{s}_1$. For later on, if we want $\boldsymbol{s}_k$, we can combine $\boldsymbol{s}_{k-1}$ and observation to get the prediction space. However, all $\{\boldsymbol{s_i}\}_{i=0}^{n-1}$ are infinite space so we need to generalize the previous theorem. Luckily, we know that the probability space is defined to be a measure space with a total measure of 1 and Radon-Nikodym theorem can generalize the Bayes' theorem.

**Definition 2.1.1.** Absolute continuity

Consider a measure space $(S, \mathcal{S})$ and two measures $\mu$ and $\nu$ defined in this space, $\nu$ is said to be absolute continuous with respect to $\mu$ if

$$\forall S \in \mathcal{S}, \text{ if } \mu(S) = 0, \nu(S) = 0 \tag{2.2}$$

holds. The relation is denoted as $\nu \ll \mu$.

**Theorem 2.** Radon-Nikodym theorem

Consider two measures $\nu$ and $\mu$ of a measurable space $(S, \mathcal{S})$: If $v \ll \mu$, then

$$\exists f \in \mathcal{L}_0^+, \forall A \subseteq S, \nu(A) = \int_A f d\mu \tag{2.3}$$

We omit the proof of this theorem as it is beyond the scope of the report.

Combine the measure theory with Bayes, let $\mu$ be the probability measure and $(M, \mathcal{M})$ be the measure space, we have the infinite dimension equation below:

$$\mu_{posterior}(\boldsymbol{m}|\boldsymbol{d}) \propto \mu_{likelihood}(\boldsymbol{d}|\boldsymbol{m})\mu_{prior} \tag{2.4}$$

Rewrite the equation above,

$$\forall M \in \mathcal{M}, \mu_{posterior}(M) \propto \int_M \mu_{likelihood} d\mu_{prior}. \tag{2.5}$$

By Theorem 2 , we have the Radon-Nikodym derivative:

$$\frac{d\mu_{posterior}}{d\mu_{prior}} \propto \mu_{likelihood} \tag{2.6}$$

As shown above, we have the transformation from prior to posterior in one iteration. However, to keep the system running, we still need bridge from the posterior to the prior of next iteration. Markov property is here to help.

### 2.1.2 Markov Property

Markov property can be intuitively understood as "memory-less" property. It means that the future distribution we would like to predict depends only on the current state. The state we have previously, however, are of no additional information revealed to us. This is a good reflection of the advection-diffusion system. For any given time $t_m$, our goal is to predict $t_n$

where $t_n > t_m$. All we care is the state at $t_m$ and how do advection and diffusion effects on the state at $t_m$ rather than how do we get to $t_m$ through the previous advection, diffusion operation. Formally, we have

**Definition 2.1.2.** Markov property for continuous time

let $\{X_i\}$ be a sequence of states. Given $0 \leq s < t$, we have:

$$P(X_t = j|X_s) = P(X_t = j|X_m, 0 \leq m \leq s) \tag{2.7}$$

In this report, the system changes over time continuously. However, for every given time $t$, theoretically, we have an infinite number of timestamps before and we need to ensure the independence regarding each of them. This is unrealistic as $\forall t_i, \forall t_j < t_i, \exists t_k$ s.t. $t_j < t_k < t_i$. So we will have to consider discretize the time and apply the Markov property to the discretized case:

**Definition 2.1.3.** Markov property for discrete time

$$P(X_{i+1} = j|X_i) = P(X_{i+1} = j|X_i, X_{i-1}, \cdots X_0) \tag{2.8}$$

For simplicity of implementation, we have the same interval length between two timestamp and define *time gap* as $\Delta t = t_i - t_{i-1}, \forall i > 0$. If $\Delta t$ is sufficiently small, the system won't change dramatically in that interval, so the discrete model serves as a good approximation to the original continuous model.

### 2.1.3 Vector Variance

Consider column vectors $\boldsymbol{X} = [x_0, x_1, \cdots x_{n-1}]^T$, by definition of covariance,

$$cov(\boldsymbol{X}) = E[(\boldsymbol{X} - E(\boldsymbol{X})) \otimes (\boldsymbol{X} - E(\boldsymbol{X}))] \tag{2.9}$$

$\otimes$ in Equation 2.9 is the tensor product between those two variables.

Now, let us consider the linear transformation on $\boldsymbol{X}$.

**Theorem 3.** Linear Transformation on Covariance

Given linear operator $A$ and random vector $b$ independent from $X$,

$$cov(A\boldsymbol{X} + b) = Acov(\boldsymbol{X})A^T + cov(b) \tag{2.10}$$

*Proof.* For simplicity, we will derive the discrete case here. For the continuous case, it follows the similar pattern.

As $b$ is independent from $\boldsymbol{X}$ and $A$,

$$cov(A\boldsymbol{X} + b) = cov(A\boldsymbol{X}) + cov(b) \tag{2.11}$$

Now consider plug in the definition of covariance, we have

$$cov(A\boldsymbol{X}) = E[(A\boldsymbol{X} - E(A\boldsymbol{X}) \otimes (A\boldsymbol{X} - E(A\boldsymbol{X}))] \tag{2.12}$$

Extract $A$, $A^T$ out from the equation before, we get the result.

$$cov(A\boldsymbol{X}) = A \underbrace{E((\boldsymbol{X} - E(\boldsymbol{X})) \otimes (\boldsymbol{X} - E(\boldsymbol{X})))}_{cov(\boldsymbol{X})} A^T \tag{2.13}$$

Combine equation 2.13 and 2.11, we get 2.10 and thus finish the proof.

□

## 2.2 PDE Properties

We need to define some operators on PDE and explore some properties before moving to the actual Kalman filter.

### 2.2.1 Space properties

PDE is about the relation between different spaces. We need to define various spaces before moving on to the relation describing them.

**Definition 2.2.1.** Inner-Product[1]

An Inner-Product on a vector space $H$ is a map $H(\cdot,\cdot) : H \times H \mapsto \mathbb{F}$ satisfy the follows

(a) The map is linear in the first arguments, $\forall \alpha, \beta \in \mathbb{F}$ and $x, y, z \in H$,

$$(\alpha x + \beta y, z) = \alpha(x, z) + \beta(y, z) \tag{2.14}$$

(b) It is conjugate symmetric. $\forall x, y \in H$,

$$(x, y) = \overline{(y, x)} \tag{2.15}$$

(c) The map is positive definite, meaning

$$\forall x \in H, (x, x) = 0 \text{ if and only if } x = 0 \tag{2.16}$$

**Definition 2.2.2.** Hilbert Space

Let H be a complete space that has the inner-prodcut described above, then H is a hilbert space.

In this report, we are mainly dealing with $\mathbb{R}^d$.

For $\mathbb{R}^d$, consider $x, y \in \mathbb{R}^d$, $x = (x_0, x_1, \cdots x_{d-1})$ and $y = (y_0, y_1, \cdots y_{d-1})$. Defined inner product: $(x, y) = \sum_{i=0}^{d-1} x_i y_i$, also define the norm as $||x|| = (x, x)^{\frac{1}{2}}$. For (a) in Definition 2.2.2,

$$(\alpha x + \beta y, z) = \sum_{i=0}^{d-1} (\alpha x_i + \beta y_i) z_i = \alpha \sum_{i=0}^{d-1} x_i z_i + \beta \sum_{i=0}^{d-1} y_i z_i = \alpha(x, z) + \beta(y, z) \tag{2.17}$$

(b) and (c) can be easily verified. Also $\mathbb{R}^d$ is comlete by construction so $\mathbb{R}^d$ is a hilbert space.

**Definition 2.2.3.** State Space $\mathcal{M}$

A state space is a hilbert space where the PDE function defined on. In other words, it reflects the domain of the PDE.

The PDE is also known as the *state equation* and we will use two terms interchangably throughout the report.

However, state space is hidden and no direct message revealed. Also, for the advection-diffusion equation, the state space is of infinite dimension that cannot be implement through computer. It motivates us to draw some observation samples from it to serve as an indicator of the state space behavior. Formally, we have

**Definition 2.2.4.** Observation Space $\mathcal{D}$

Sample points are taken from a specific state space and timestamp. The space where these points live in is called the observation space. It is the range of the state equation.

The Observation operator $B$ is the operator that extracts the point we need from the state space. so we have, $B : \mathcal{M} \mapsto \mathcal{D}$

We use the following diagram to summary definitions introduced in this section.

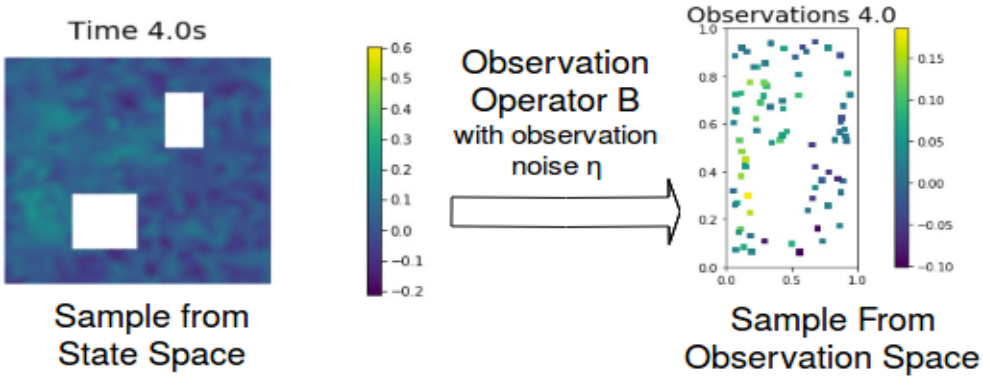


Figure 2.1: Relation between state space, observation space and observation operator.

### 2.2.2 Other properties

**Definition 2.2.5.** Hessian matrix

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $\boldsymbol{x} = (x_0, x_1, \cdots x_{n-1}) \in \mathbb{R}^n$, $H$ is the hessian matrix where $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$

**Theorem 4.** Given a multivariate gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\bar{x}, C_x)$, the covariance matrix $C$ is the inverse of the hessian matrix $H$, i.e $C_x = H^{-1}$

*Proof.* By definition, the probability density function(PDF) of $\boldsymbol{x}$ is given by

$$p(\boldsymbol{x}) \propto \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^T\right) C_x^{-1}(\boldsymbol{x} - \boldsymbol{x}^*) \tag{2.18}$$

Take the negative logrithm of 2.18, we have

$$J(\boldsymbol{x}) = -\ln\, p(\boldsymbol{x}) \propto \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^T C_x^{-1}(\boldsymbol{x} - \boldsymbol{x}^*) \tag{2.19}$$

Take the derivatives with respect to $x_i, x_j$ respectively, we have

$$H_{ij} = \left.\frac{\partial^2 J(\boldsymbol{x})}{\partial \boldsymbol{x}_i \partial \boldsymbol{x}_j}\right|_{\boldsymbol{x}=\boldsymbol{x}^*} = {C_x^{-1}}_{ij} \tag{2.20}$$

$\forall i, j$, Equation 2.20 holds and invert both right and left side, we have $C_x = H^{-1}$ □

Sometimes, the state equation is hard to solve directly, or it may have some quirky behavior on some random part, so we would like to consider weaken the restriction and get an approximate solution. Formally, the weak solution is set up as follows:

**Definition 2.2.6.** Weak Formulation and Weak Solution[6]

Given a differential equation, rewrite it in such a way that no derivatives of the solution of the equation show up. The new form is called the *weak formulation*. Usually, the new form is formulated by multiply by another

function and use integral by parts to retrieve the solution. In this process, the function we multiply is called *test function* and the function we would like to get the solution is *trial function*. The solution of the trial function is called *weak solution*.

In the pratical setting, usually we transfer the state equation to linear algebra like problems and use the linear algebra language to solve the problem. In other words, we would like to solve the equation about $\boldsymbol{u}$ like

$$A\boldsymbol{u} = f. \tag{2.21}$$

where $A : \mathcal{X} \mapsto \mathcal{X}'$, $\boldsymbol{u} \in \mathcal{X}$ and $f \in \mathcal{X}'$. $\mathcal{X}$ is a banach space and $\mathcal{X}'$ is the corresponding dual space of $\mathcal{X}$.

## 2.3 Kalman Filter

With the statistical and PDE properties introduced in previous sections, we can finally derive the Kalman filter algorithm. But before that, we need to quantitatively understand the model.

### 2.3.1 Data Assimilation

Given the fact that the combination of prior and likelihood can create a new probability distribution that can better quantify some random variable, we can use such property to conduct prediction on random variables. This combination and prediction process is formally known as *data assimilation*.

**Definition 2.3.1.** Data Assimilation [4]

Data assimilation is the integration of two information resources:

a) A mathematical model of a time-dependent physical system, or a numerical implementation of such model

b) A sequence of observations of that system, usually corrputed by some noise.

In general, we need two equations to describe the two-step of data assimilation respectively. In the following, $\hat{\boldsymbol{m}}_{p|q}$ means the estimate of $\boldsymbol{m}$ at time $p$ given the observation up to time(or only at by Markov property) $q$ and $p \geq q$. The similar notation also for the covariance of the distribution:$P_{p|q}$. The Kalman filter algorithm of the data assimilation works for the system like follows: $\forall \boldsymbol{m}_i \in \mathcal{M}$.

$$\boldsymbol{m}_{k|k-1} = F_k \boldsymbol{m}_{k-1|k-1} + \xi_k \tag{2.22}$$

$\boldsymbol{m}_k$ is the model that change over time. It reflects the state where the system is in on the $k$-th iteration.

$F_k : \mathcal{M} \mapsto \mathcal{M}$ is a state transition model. In this model settings, it is the combination of the advection and diffusion that serves as $F_k$.

Here we need to get the closed form of the transition operator before moving on. Let us recap equation 1.1. It is.

$$\frac{\partial \boldsymbol{u}}{\partial t} - \kappa \nabla \boldsymbol{u} + \boldsymbol{v} \cdot \Delta \boldsymbol{u} = 0 \tag{2.23}$$

As equation 1.1 is hard to solve, we consider formulate equation follow the pattern as Equation 2.21.

So let us multiply by test function $\boldsymbol{w} \in W$ at both right and left side of the equation, we have

$$\int_{\mathcal{X}} [\frac{\partial \boldsymbol{u}}{\partial t} - \kappa \nabla \boldsymbol{u} + \boldsymbol{v} \cdot \Delta \boldsymbol{u}] \boldsymbol{w} = 0 \tag{2.24}$$

Refactor the equation above, we have

$$(\frac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{w}) + a(\boldsymbol{u}, \boldsymbol{w}) = 0 \tag{2.25}$$

where $a(\boldsymbol{u}, \boldsymbol{w})$ is defined as follows:

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{w}) &= \int_{\mathcal{X}} [-\kappa \nabla \boldsymbol{u} + \boldsymbol{v} \cdot \Delta \boldsymbol{u}] \boldsymbol{w} \\ &= \int_{\mathcal{X}} -\kappa \nabla \boldsymbol{u} \boldsymbol{w} + \underbrace{\boldsymbol{v} \cdot \nabla (\nabla \boldsymbol{u} \boldsymbol{w})}_{\nabla u \cdot w = 0} - \boldsymbol{v} (\nabla \boldsymbol{u} \nabla \boldsymbol{w}) \\ &= -\int_{\mathcal{X}} \kappa \nabla \boldsymbol{u} \boldsymbol{w} + \boldsymbol{v} (\nabla \boldsymbol{u} \nabla \boldsymbol{w}) \end{aligned} \tag{2.26}$$

We can approximate the trial and test spaces by finite element methods.

$$\boldsymbol{u} = \sum_{j=0}^{N-1} \varphi_j \boldsymbol{u}_j, \boldsymbol{u}_j \in \mathcal{X}_h = \text{span}\langle \varphi_0, \varphi_1, \cdots \varphi_{N-1} \rangle \subset \mathcal{X} \tag{2.27}$$

$$\boldsymbol{w} = \sum_{i=0}^{N-1} \phi_i \boldsymbol{w}_i, \boldsymbol{w}_i \in \mathcal{W}_h = \text{span}\langle \phi_0, \phi_1, \cdots \phi_{N-1} \rangle \subset \mathcal{W} \tag{2.28}$$

where $\phi_i$ and $\varphi_j$ are both Lagrangian finite element basis, just represent in different variance of same greek letter for formulating $\boldsymbol{u}$ and $\boldsymbol{w}$ respectively. so plug in our results to Equation 2.25,

$$M \frac{d\boldsymbol{u}}{dt} + A\boldsymbol{u} = 0 \tag{2.29}$$

where $A$ is the assembled version of $a(\boldsymbol{u}, \boldsymbol{w})$.i.e:

$$A = \begin{bmatrix} a(\varphi_0, \phi_0) & \cdots & a(\varphi_{N-1}, \phi_0) \\ \vdots & \ddots & \vdots \\ a(\varphi_0, \phi_{N-1}) & \cdots & a(\varphi_{N-1}, \phi_{N-1}) \end{bmatrix} \tag{2.30}$$

The mass matrix is defined as $M_{ij} = (\varphi_j, \phi_i)$. In matrix format, it is

$$M = \begin{bmatrix} (\varphi_0, \phi_0) & \cdots & (\varphi_{N-1}, \phi_0) \\ \vdots & \ddots & \vdots \\ (\varphi_0, \phi_{N-1}) & \cdots & (\varphi_{N-1}, \phi_{N-1}) \end{bmatrix} \tag{2.31}$$

Rewrite Equation 2.29 in forward finite difference format, we have

$$M\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} + A\boldsymbol{u}^{n+1} = 0 \tag{2.32}$$

Re-organize the equation and we get the relation between $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n+1}$:

$$(\frac{M}{\Delta t} + A)\boldsymbol{u}^{n+1} = \frac{M}{\Delta t}\boldsymbol{u}^n \tag{2.33}$$

Combine all derivation above, we have the closed form for $F$:

$$F = \underbrace{(\frac{M}{\Delta t} + A)^{-1}(\frac{M}{\Delta t})}_{T^{-1}S} \tag{2.34}$$

This $F$ is only responsible for one iteration of updates. If we would like to get $u_i$ to $u_j, (j > i)$ without perturbing noise, we can use $u_j = F^{j-i}u_i$.

$\xi_k$ is the process noise. It characterizes the uncertainty of the PDE. With the evolving of the system, some outside factors may come into play and they affect the state that the system may land in.

To fulfill the requirement of part b), we have

$$\boldsymbol{d}_k = B\boldsymbol{m}_{k|k-1} + \eta_k \tag{2.35}$$

where at time $k$,

$B$ is the observation operator as shown in Figure 2.2.1 that transforms data from state space to observation space.

$\eta_k$ characterizes the observation noise. Basically this comes from the inaccuracy of measurement on $\boldsymbol{d}_k$.

Part (a) is also known as the prediction phase and part (b) is known as the update phase. Combine these two parts, we formulate the base model for the Kalman filters work. Intuitively, the whole process can be represented as the diagram shown below:

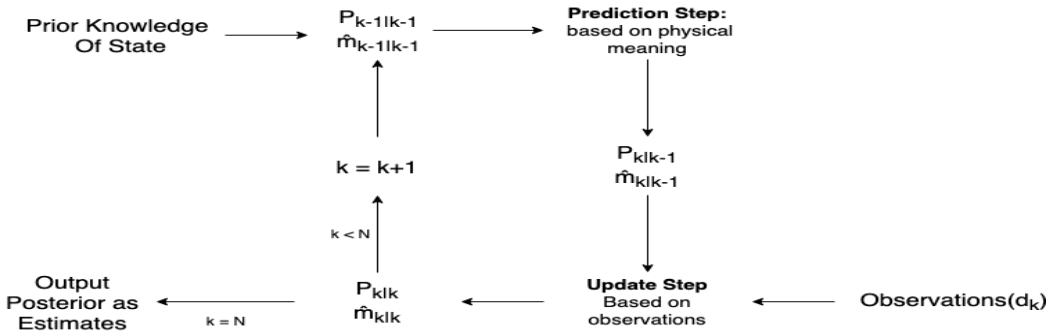


Figure 2.2: Visualization Diagram of Kalman filter.

### 2.3.2 Algorithm Derivation

We will develop the Kalman filter model in an induction manner. Given many nice properties that Gaussian distribution holds, the derivation here will mainly focus on variables that obey the Gaussian distribution.

Assume initial state: $\boldsymbol{m}_{0|0} = \boldsymbol{m}_0 \sim \mathcal{N}(\bar{\boldsymbol{m}}, P_0)$, $\bar{\boldsymbol{m}}$ is the initial mean and $P_0 = P_{0|0}$ for the initial covariance of the prior distribution.

$$\mu(\boldsymbol{m_0}) \propto \exp(-\frac{1}{2}(\boldsymbol{m_0} - \bar{\boldsymbol{m}})^T P_0^{-1}(\boldsymbol{m}_0 - \bar{\boldsymbol{m}})) \tag{2.36}$$

$Q_0$ characterizes the process noise of stochastic part of PDE so we assume process noise: $\boldsymbol{m}_1 - F\boldsymbol{m}_0 \sim \mathcal{N}(0, Q_0)$

Taking observation matrix $B$ into consideration, denote the data we can observe as $\boldsymbol{d}_0$, with random observation noise covariance$R_0$, i.e: $\boldsymbol{d}_0 - B\boldsymbol{m}_0 \sim \mathcal{N}(0, R_0)$ so we have follows:

$$\mu_{likelihood} = \mu(\boldsymbol{d}_0|\boldsymbol{m}_0) \propto \exp(-\frac{1}{2}(\boldsymbol{d_0} - B\boldsymbol{m_0})^T R_0^{-1}(\boldsymbol{d_0} - B\boldsymbol{m_0})) \tag{2.37}$$

By Theorem 2, plug in 2.6, we have

$$\mu_{posterior}(\boldsymbol{m}_1|\boldsymbol{d}_0) \propto \exp(-\frac{1}{2}(\boldsymbol{m_0}-\bar{\boldsymbol{m}})^T P_0^{-1}(\boldsymbol{m}_0-\bar{\boldsymbol{m}})-\frac{1}{2}(\boldsymbol{d_0}-B\boldsymbol{m_0})^T R_0^{-1}(\boldsymbol{d_0}-B\boldsymbol{m_0})) \tag{2.38}$$

As the product of two normal distributed random variables is also normal[2], the goal is to find the mean and covariance of the predicted value at a certain timestamp.

The mean of the posterior distribution is the parameter vector that maximizes the posterior. In other words, we want to find a $\boldsymbol{m}_{\text{MAP}}$ that maximizing 2.38. Ignore the exponential sign and flip the negative sign, we have

$$\boldsymbol{m}_{\text{MAP}} := \arg\min \mathcal{J}(\boldsymbol{m}) := \frac{1}{2}(\boldsymbol{m_0}-\bar{\boldsymbol{m}})^T P_0^{-1}(\boldsymbol{m}_0-\bar{\boldsymbol{m}})+\frac{1}{2}(\boldsymbol{d_0}-B\boldsymbol{m_0})^T R_0^{-1}(\boldsymbol{d_0}-B\boldsymbol{m_0}) \tag{2.39}$$

Our goal is to construct $\hat{\boldsymbol{m}}_{1|1}$ and $P_{1|1}$ based on inputs. Here, the derivation follow the flow chart in the Figure 2.2, that is: $\boldsymbol{m}_{0|0} \rightarrow \boldsymbol{m}_{1|0} \rightarrow \boldsymbol{m}_{1|1}$, $P_{0|0} \rightarrow P_{1|0} \rightarrow P_{1|1}$

#### 2.3.2.1 Derivation of Mean

By equation 2.22, as we assume noise $\xi_0 \sim \mathcal{N}(0, C_0)$, by definition, we have

$$\mathbb{E}[\hat{\boldsymbol{m}}_{1|0}] = \mathbb{E}[F_0\hat{\boldsymbol{m}}_{0|0} + \xi_0] \tag{2.40}$$

As $\xi_k$ is independent from $\boldsymbol{m}$, we can separate that out and remove it. So we have

$$\mathbb{E}[\hat{\boldsymbol{m}}_{1|0}] = \mathbb{E}[F_0\hat{\boldsymbol{m}}_{0|0}] + \mathbb{E}[\xi_0] = \mathbb{E}[F_0\hat{\boldsymbol{m}}_{0|0}] \tag{2.41}$$

As here we only care about the mean, remove the Expectation operator and finish the construction from $\hat{\boldsymbol{m}}_{0|0}$ to $\hat{\boldsymbol{m}}_{0|1}$. Now comes to construct $\hat{\boldsymbol{m}}_{1|1}$ from $\hat{\boldsymbol{m}}_{1|0}$. We actually "learn" more knowledge from the difference between the actual observation and the transformation from the state to observation. We hope the weighted combination of prior and the knowledge learned can

offer us smaller error estimation. At the time $k$, we denote such operator as $K_k$ and is known as *kalman gain*.

$$\hat{\boldsymbol{m}}_{1|1} = \hat{\boldsymbol{m}}_{1|0} + K_1(\boldsymbol{d_1} - B_1\hat{\boldsymbol{m}}_{1|0}) \tag{2.42}$$

where $\hat{\boldsymbol{m}}_{1|1}$ should minimize equation 2.39. Combine 2.41 and 2.42, we have finish the construction of $\hat{\boldsymbol{m}}_{1|1}$ from $\hat{\boldsymbol{m}}_{0|0}$

$$\hat{\boldsymbol{m}}_{1|1} = F_0\hat{\boldsymbol{m}}_{0|0} + K_0(\boldsymbol{d}_1 - BF_0\hat{\boldsymbol{m}}_{0|0}) \tag{2.43}$$

#### 2.3.2.2 Derivation of Covariance

Now comes the derivation on covariance matrix $P$.

First from $P_{0|0}$ to $P_{1|0}$. Apply Theorem 3 to variables in equation 2.22

$$P_{1|0} = F_0P_{0|0}F_0^T + Q_0 \tag{2.44}$$

Then consider the way from $P_{1|0}$ to $P_{1|1}$. By definition of $P_{1|1}$,

$$P_{1|1} = cov[(\boldsymbol{m}_1 - \hat{\boldsymbol{m}}_{1|1})] \tag{2.45}$$

Plug in the definition of $\hat{\boldsymbol{m}}_{1|1}$,

$$P_{1|1} = cov((\boldsymbol{m_1} - [\hat{\boldsymbol{m}}_{1|0} + K_1(\boldsymbol{d}_1 - B_1\hat{\boldsymbol{x}}_{1|0})])) \tag{2.46}$$

Substitute $\boldsymbol{d}_1$ equation 2.35 into equation above,

$$P_{1|1} = cov((\boldsymbol{m_1} - [\hat{\boldsymbol{m}}_{1|0} + K_1(B_1\boldsymbol{m}_1 + \eta_1 - B_1\hat{\boldsymbol{m}}_{1|0})])) \tag{2.47}$$

As $\eta_1$ is idnependent from all $\boldsymbol{m}$'s, we can separate the term with $\eta_1$ out and express the equation as

$$P_{1|1} = cov[(I - K_1 H_1)(\boldsymbol{m}_1 - \hat{\boldsymbol{m}}_{1|0})] + cov(K_1 \eta_1) \tag{2.48}$$

By Theorem 3 and plug in the definition of $P_{1|0}$ and $R_1$, we have

$$P_{1|1} = (I - K_1 H_1) P_{1|0} (I - K_1 H_1)^T + K_1 R_1 K_1^T \tag{2.49}$$

Plug in $P_{1|0}$ in 2.49 to 2.44, we have from $P_{0|0}$ to $P_{1|1}$:

$$P_{1|1} = (I - K_1 H_1)(F_0 P_{0|0} F_0^T + C_0) P_{0|0} (I - K_1 H_1)^T + K_1 R_1 K_1^T \tag{2.50}$$

#### 2.3.2.3 Derivation of Kalman Gain

Now let us tackle Kalman gain, a variable that can minimize the difference between $\boldsymbol{m}_k$ and $\hat{\boldsymbol{m}}_{k|k}$. In other words, it is the minimizer of $P_{k|k}$.

Expand Equation 2.49,

$$P_{1|1} = P_{1|0} - K_1 B_1 P_{1|0} - P_{1|0} B_1^T K_1^T + K_1 (B_1 P_{1|0} B_1^T + R_1) K_1^T \tag{2.51}$$

Treat $P$ as a function of $K$, take the derivative respect to $K$ and set it to 0.

$$\frac{\partial \operatorname{tr}(P_{1|1})}{\partial K_1} = -2(B_1 P_{1|0})^T + 2K_1 (B_1 P_{1|0} B_1^T + R_1) = 0 \tag{2.52}$$

Solve for $K_1$, we have the closed form for kalman gain:

$$K_1 = P_{1|0} B_1^T (B_1 P_{1|0} B_1^T + R_1)^{-1} \tag{2.53}$$

This concludes our derivation from state 0|0 to 1|1. For any state onwards, by the Markov property, it can always be regarded as the state at time 1 and follow the derivation above by referring the previous state. Plug in $K_k$, we have the Kalman filter algorithm.

### 2.3.3 Kalman Filter Algorithm

**Algorithm 1** Kalman Gain algorithm

$R \leftarrow R_0$, $P \leftarrow P_0$ ▷ $R_0$: observation noise, $P_0$: cov of $\boldsymbol{m_0}$
$B \leftarrow B_0$ ▷ $B_0$ :Observation matrix
$F \leftarrow F_0$ ▷ $F_0$ :Transition matrix
$Q \leftarrow Q_0$ ▷ $Q_0$ :process noise
Take a sample $\boldsymbol{m_0} \sim \mathcal{N}(\bar{m}, P_0)$ as initial state.
**for** $i = 0, 1, 2, \cdots N-1$ **do**
    $d \leftarrow d_i$ ▷ Observation at iteration $i$
    $P \leftarrow FPF^T + Q$ ▷ $P_{i-1|i-1} \rightarrow P_{i|i-1}$
    $K = PB(R + BPB^T)^{-1}$ ▷ Formulate Kalman Gain
    $P = (I - KH)P$ ▷ $P_{i|i-1} \rightarrow P_{i|i}$
    $m = Fm + K(d - BFm)$ ▷ $m_{i-1|i-1} \rightarrow m_{i|i}$
**end for**
**return** $m$, $P$ ▷ The estimator: $\boldsymbol{m_{N|N}} \sim \mathcal{N}(m, P)$

In the next chapter, we will investigate the actual performance of the Kalman filter against different initialization and setup of a PDE on different metrics to evaluate its performance from different aspects.

# Chapter 3

# Numerical Experiments

In this chapter, we implement the theoretical derivation from the previous chapters in hIPPYlib and numpy. We first examine the basic implementation of the algorithm and then move on to different variables in this algorithm to reveal different aspects of the Kalman filter.

## 3.1 Independent Prior

Let us first examine a simple case where the prior data are independent and process, observation noise are i.i.d. In this case, both $Q_0$ and $R_0$ are diagonal matrices.

First, we need to include the mesh (Figure 3.1). For simplicity, we use a simple mesh of 2023 points that looks like below and take 80 observation points through observe operator $B$. It can not only be representative of the space but also keep the computation performance.

$F$ is the transition operator as in equation 2.34. $F_k = T^{-1}S, \forall k \in \mathbb{R}^+$

Encoding process noise standard deviation $\xi_0 = 0.02$, observation noise standard deviation $\eta_0 = 0.01$, we have $R_0 = \eta_0^2 I$ and $Q_0 = \xi_0^2 I$. In this case, for simplicity, $\forall t, Q_t \equiv Q_0, R_t \equiv R_0$. Also assume $P_0 = Q_0$.

Including the time gap $\Delta t = 0.025s$ and the starting observation time $t_0 = 0s$

Later on, we will modify some of the initial conditions listed here and examine what their effect will have on different metrics.

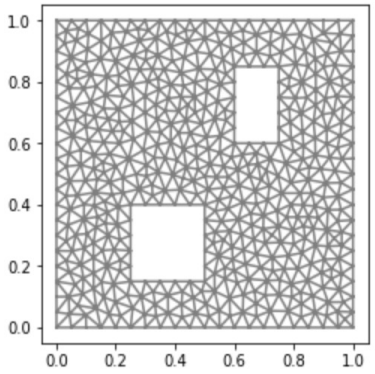


Figure 3.1: Mesh Graph for examining independent prior.

### 3.1.1 Eigenvalues of Independent prior

First, we will inspect eigenvalues. As $P_t$'s eigenvalue changes over time and the magnitude also varies, we need a uniform standard for comparison. Here, we would like to introduce a definition: *eigenvalue ratio*

**Definition 3.1.1.** Eigenvalue ratio

Let $A$ be a matrix and $\lambda_0, \lambda_1, \cdots \lambda_{n-1}$ be its eigenvalues. Eigenvalue ratio of an eigenvalue $\lambda_k$ is $|\frac{\lambda_k}{\max(\lambda_i)}|$

It can be easily verified $0 \leq$ eigenvalue ratio $\leq 1$, also as we are dealing

with covariance matrix, which is symmetric and positive definitem, we can ignore the absolute sign in the definition.

At $i$-th iteration, sort the eigenvalue ratio list in descending order, for $j < 200$, we have the following ratio trend for every second:

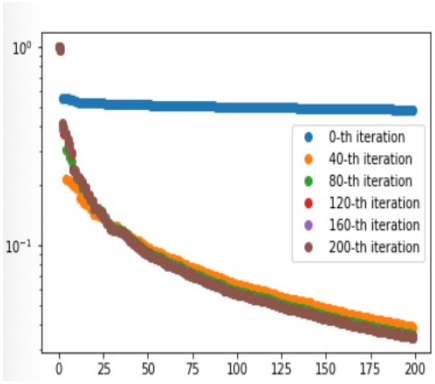


Figure 3.2: Eigenvalue ratio for independent prior change over time.

By Figure 3.2, at the 0-th iteration, the decay in eigenvalue ratio is negligible, reflecting the eigenvalues are approximately the same. With the elapse of time, we find that the gap between the 0-th and the 40-th iteration is very big. This means the top eigenvalues are stood out and thus should result in a smaller spectrum as the data are more concentrating to the top. and then it gets much smaller. The difference between 40 and 80, 80 to 120, $\cdots$ are negligible, this means convergence in later iterations.

In order to measure how the change of the hessian matrix behaves, we need a metric to quantify it. Here we use the trace as the metric. As we know that the trace of a matrix is equal to the sum of the (complex) eigenvalues

[9] and by Theorem 4, we find the reciprocal sum of all eigenvalues in every iteration, and then plot them with the increments in the iteration. The result is shown in Figure 3.3. It has a general decaying trend, reflecting the hessian matrix gets smaller and smaller. This verifies the effect of Kalman filter can help us get a lower variance estimation. Also, it is very clear that most of the decay in eigenvalues are concentrated in the first 25 iterations and later are rather smoother, which is coherent with Figure 3.2 where most of the decay of the eigenvalue ratio in the first couple of iterations and converges on later iterations.

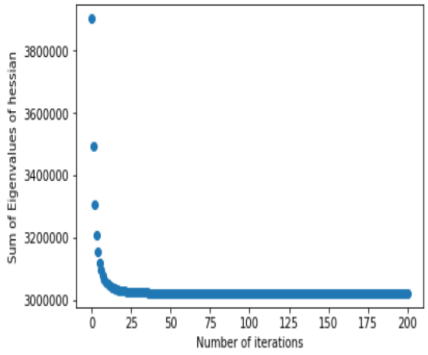


Figure 3.3: Decay of Hessian over time.

### 3.1.2 Eigenvectors of Independent Prior

Let us now examine the eigenvectors. Eigenvectors are of great importance to us as they characterize the distribution at that time. Here we examine the top 3 eigenvectors when sort by their associated eigenvalues in descending order.

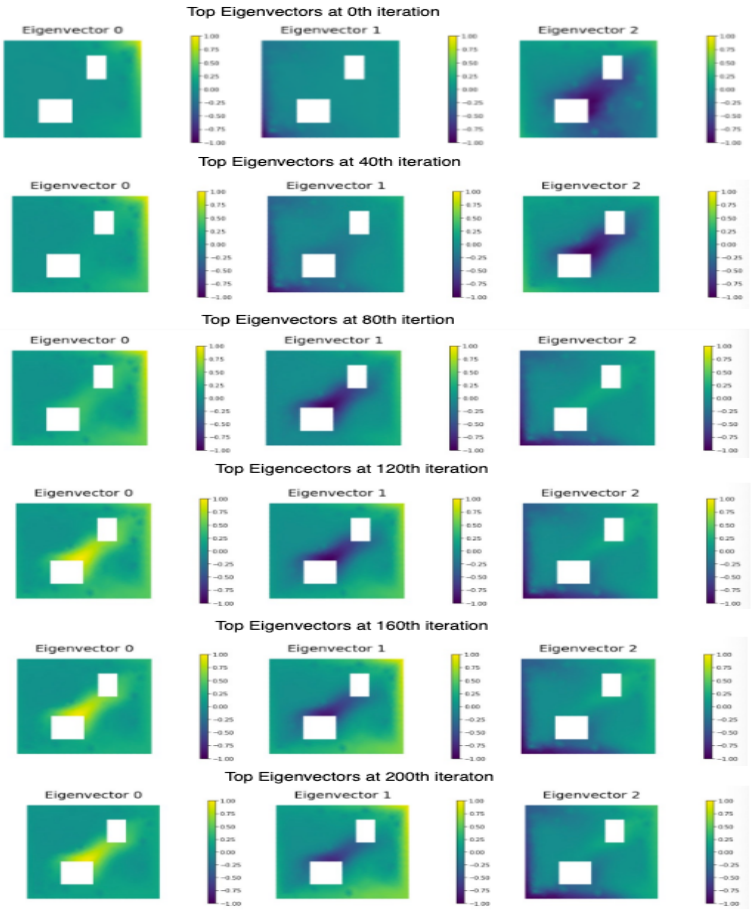


Figure 3.4: Top 3 Eigenvectors change over time given independent prior.

We can see some similarity from the graph above. For the eigenvectors in the same column are alike but also changes gradually. This reflects the gradual diffusion and advection process.

### 3.1.3 Samples of Independent Prior

Now, let us consider the sample taken at some specific iteration. It can intuitively reveal how the data behaves at a given time point. Same as the eigenvector, we plot the data for every second.


Figure 3.5: Change of mean and sample over time for independent prior

Figure 3.5 reflects the distribution of the state space. For a given iteration, we first plot the mean to get an idea of how the center of the distribution looks like, then plot 4 samples to avoid possible accidental outliers. By comparing these, we can get a rough idea of how the data evolves with the time elapsed. Observe the graph above, our samples generally follow the true state at that given time. On the other hand, it has a decreasing trend in the hessian, meaning we are more and more certain about the actual distribution.

### 3.1.4 Vary on initial mean

In the setup of the problem, we have the initial mean set to be 0. When compare with the true initial state, according to observation on Figure 3.5, they are rather similar except for the point where the concentration originally starts. Now the problem comes to for different initial states, will the estimation still get closer to closer to the true state as well? We consider 5 different initialization means. They are all 0's, same as the true initial condition, all 1's, half 1's and half 0's oscillate and that every entry follows the $\mathcal{N}(0.5, 0.5)$ distribution respectively. Here the second 0.5 is the standard deviation. Their initial covariance is all identical to be the $P$ introduced at first.

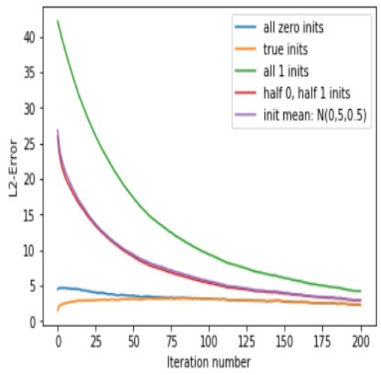


Figure 3.6: Error change for different mean with the progress of time.

From the graph shown above, we find that all zero initialization is the nearest to the actual initial state. With the evolve of the system, all of them have smaller and smaller errors and the best initialization have the best estimation, the other initialization, converge to the same error as well. This is consistent with what the stochastic theory states for the recurrent markov chain, that is, if the system has a steady state, it will reach there that satisfy $\pi = \pi F$, where F is the transition matrix between states, regardless of the starting state.

We can also visually see the different converge effects like below by examining the state every 100 iterations. so we will have the initial, middle and end state.

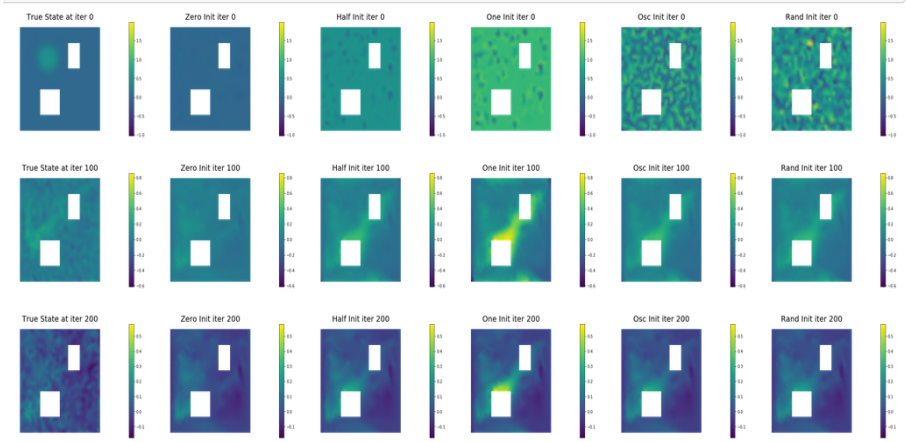


Figure 3.7: Visualization of Comparison of convergence of different initialization.

The graph in Figure 3.7 again verifies the trend of the change of the substance. At first, it may have different initialization, but with the pass of the time they are more and more alike, so they have similar errors when compare to the real solution.

### 3.1.5 Effect of observation points

Observation points give us further information that used to shrink the hessian of the observation. On one hand, generating enough observation points can be representative of the state space. On the other hand, ample amount of observation points hinder the performance.

Also, we can examine for a given specific initialization, how would the error change with the increase of data points. Here, we use the initialization

with all ones. As that is the initialization that farthest away from the true value, it can trigger relative significant changes in error as shown in Figure 3.8. Similar to the x-axis before, here we examine a geometric sequence of the number of data points. When examining the general trend, with the increase of number of observation points, the error decreases faster, meaning the more info gain from the observation actually contributes to the estimation in the coming iteration. Also, we find that 1536 observation points have the data almost converge but only 3 data points it just drops linearly and has not much clue on convergence. More data reveals more about how the distribution of the state will converge.

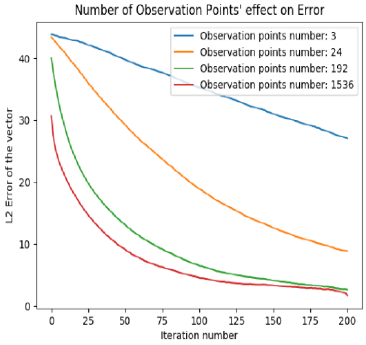


Figure 3.8: Effect of Different number of observation points on the change of error for a specific initialization

Now let us investigate the effect of the number of observation points on the error of the last estimation timestep. We know that with more observation point we are more and more certain about where the state should be. Here we are concerning the quantitative result, and also want to find out if different

initialization has different effects on the decay of error.

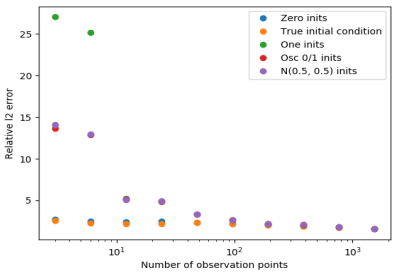


Figure 3.9: Change of error with different observation point.

From 3.9, we take the same 5 initialization method as the previous part and inspect the number of elements in a geometric sequence to be representative. With the increase of data points, the error in general are shrinking and the gap between different estimation in error are small, which is an indicator that it converges to the same point. Quantiatively, let us inspect the last two iteration error.

| ob point/type | zero | true | one | half-half | N(0.5,0.5) |
|---|---|---|---|---|---|
| 768 | 1.75112 | 1.75122 | 1.77799 | 1.78056 | 1.78285 |
| 1536 | 1.54470 | 1.54470 | 1.54509 | 1.54489 | 1.54450 |

Table 3.1: Different initialization last two iteration.

When the number of observation points is sufficiently large, the difference between these relative errors is about on the order of $10^{-2}$. In this way, we can conclude that the initial guess does not affect our final result as long as we have sufficient observation point. However, sufficient data point may be hard to obtain and farther away initializations take significantly longer time

to converge. A good strategy would be let the prior as close to the true initial as possible, then with the help of outsourced information, we may decrease the process noise and have small $P$ to give us better certainty.

### 3.1.6 Vary on Noise

Noise can also be a factor that affects the performance of the Kalman filter. In the exploration before, $R$ and $Q$ are of similar magnitude. For now, let us investigate what will happen when these two are tremendously different. Similar to the context before, we will use trace to measure the magnitude of a matrix. In the settings below, we don't change $P_0$ and keep it same as before, that is $P = \sigma^2 I$ where $\sigma = 0.02$.Also, to avoid the duplicate meaning of symbols, in context below, we use $\lll$ to represent far smaller than. i.e: $a \lll b$ means $a$ is far smaller than $b$.

#### 3.1.6.1 $R \lll Q$

Here we keep the problem simple. Observation noise standard deviation is $\eta_k = 10^{-4}$ and the process noise standard deviation is $\xi_k = 0.02$. The decay of the eigenvalue ratio and trace are like follows:

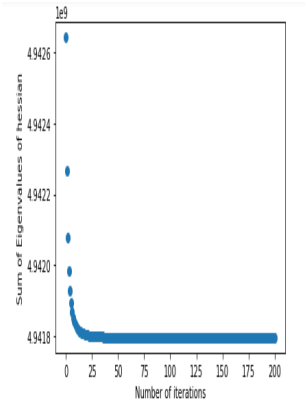


Figure 3.10: Decay of hessian over time for $R \lll Q$.

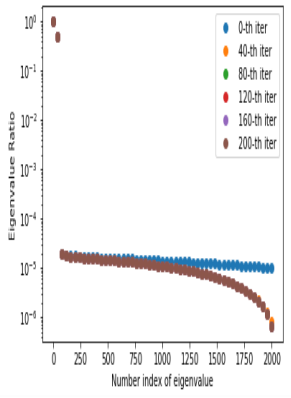


Figure 3.11: Decay of eigenvalue ratios over time for $R \lll Q$.

As we can see from the left graph above, when $R \lll Q$, the hessian got significantly larger but have smaller relative decay. From the right graph, the similar decay pattern also holds but it is much more skewed towards the biggest eigenvalues. In the previous settings, the smallest eigenvalue ratio is of magnitude $10^{-1}$. In this graph, however, it can be as small as $10^{-6}$, which means many associated eigenvectors are negligible.

#### 3.1.6.2 $Q \lll R$

Here we flip the initialization listed before so now the process noise has the standard deviation of $\xi_k = 10^{-4}$ and the observation noise standard deviation is $\eta_k = 0.02$. The decay of the eigenvalue ratio and change of trace are like follows:

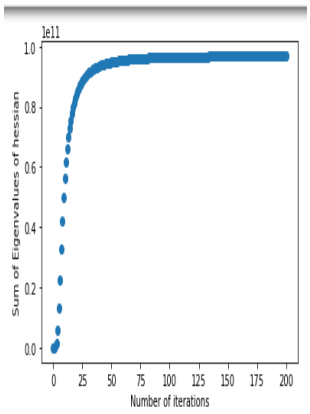


Figure 3.12: Rise of hessian over time for $Q \lll R$.

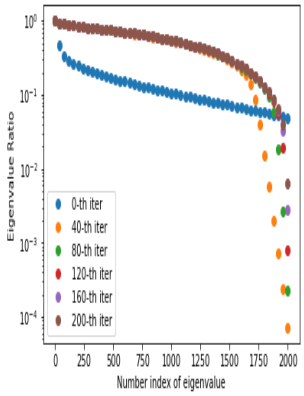


Figure 3.13: Decay of eigenvalue ratios over time for $Q \lll R$.

The performance for $Q \lll R$ is a reverse of $R \lll Q$. In this case, the observation noise accumulates in a much faster pace than that of the process noise. In other words, the Kalman filter does not promote but rather hinder the decrease of the hessian matrix. A similar scenario also applies to the eigenvalue ratio. The Kalman filter may help the skewness of the smallest eigenvalues but sacrifice the relatively bigger ones. As the operator is dominated by big eigenvalues, such a routine is not effective. In all, we can conclude that the Kalman filter does not work for the case $Q \lll R$.

The inaccuracy representation of the floating point number of computer contributes to the totally different behavior of this case in hessian and eigenvalue ratio. To be specific, the computer may represent small eigenvalues that are positive to be negative because of the round off error and thus make $P_k$ not symmetric positive definite anymore.

### 3.1.7 Refined Mesh

Now consider a refined Mesh. The mesh takes 7863 points and distributes similarly with the non-refined version. This mesh reflects the state space better but sacrifices the computation performance.

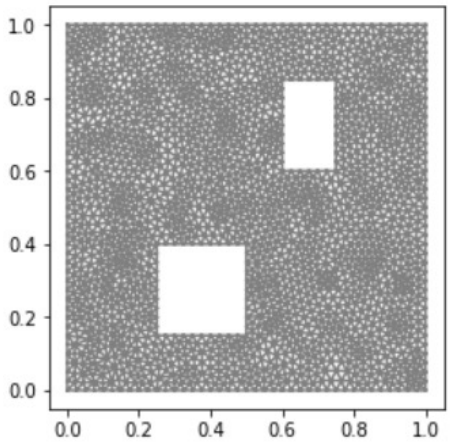


Figure 3.14: Refine Mesh Graph for Independent Prior.

As the Kalman filter algorithm is a quadratic order algorithm, When increasing the number of points to 7863, it takes about $(\frac{7863}{2023})^2 \approx 15$ times longer time than the non-refined one. With the huge computation complexity and limited computation power, we only care about the mean and change time gap $\Delta t = 0.25s$ to compensate for that.


Figure 3.15: Comparison of mean of distribution between refined and unrefined mesh.

Also, similar to the metric inspection for the simplified version of the mesh, we can inspect its eigenvalues as shown in Figure 3.16. From the graph, it follows the skewed property introduced in Figure 3.2. However, compared with Figure 3.2, at the same timestamp, the decrease in this is less skewed.

For example, the 4th iteration here for the 200th biggest eigenvalue ratio is about one third point to $10^{-1}$, which is about $10^{-\frac{2}{3}} \approx 0.21$ while in Figure 3.2, its about $10^{-\frac{3}{2}} \approx 0.07$. Such a difference mainly coming from the less granularity in time, so less iteration has been experienced and thus the top eigenvalues cannot standout.

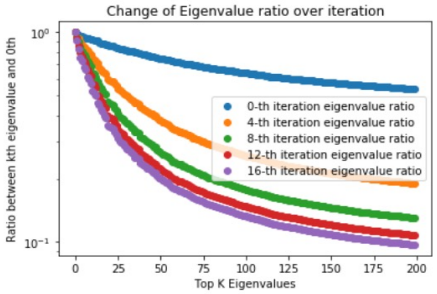


Figure 3.16: Change of eigenvalue ratio with refined mesh.

## 3.2 Smooth Prior

Now let us consider a more complicated case by weakening the independence requirements so that data points at the same time can have spatial dependence with each other. In other words, the data at a given spatial coordinate would be affected and also affects the other points. Such dependency

makes the prior and distribution data onward smoother. For $Q$ and $R$, we still keep them diagonal as $Q = \xi^2 I$ and $R = \eta^2 I$ . Also, we use the simple mesh as shown in Figure 3.1. Given the time-consuming part of sampling, here we only care about the mean but ignore the covariance to be the estimator.

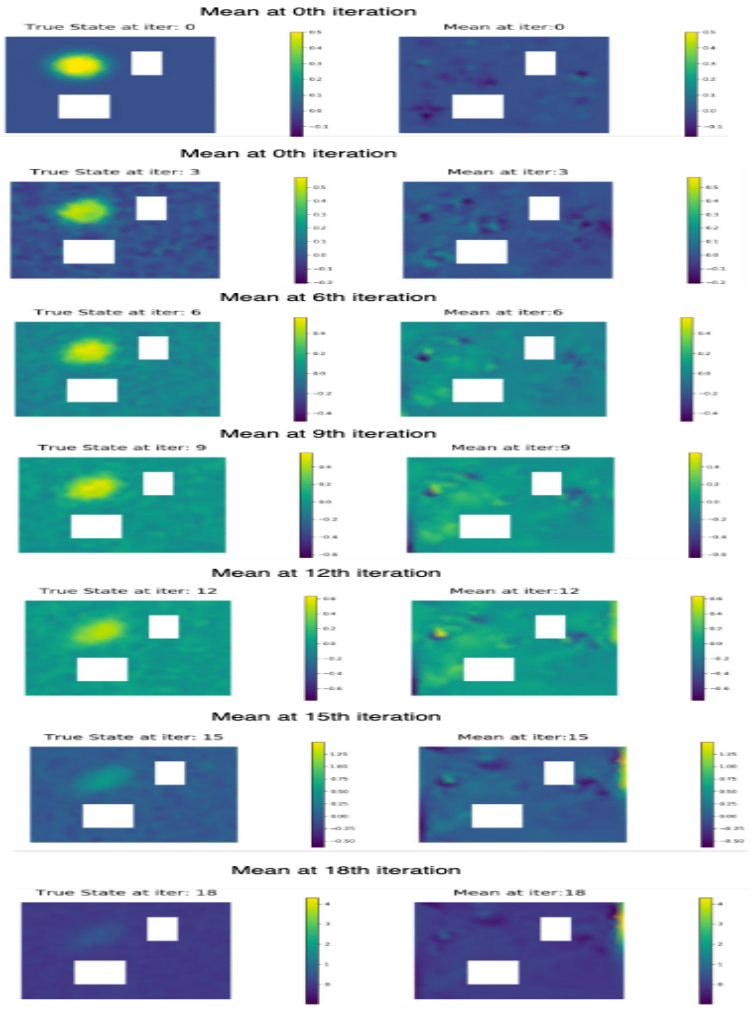


Figure 3.17: Mean change over time for dependent prior.

As we can see Figure 3.17, in the dependent settings where we use prior.R, the process diffuses much quicker than that of the independent case. After $18 \times 0.025 = 0.45$ second, the system almost totally diffused. Our mean constructed by the Kalman filter follows the similar pattern as the true state behaves but also have some extreme point on the boundary points, but the anomaly happens on less and less part, reflecting better and better predictions of the coming state.

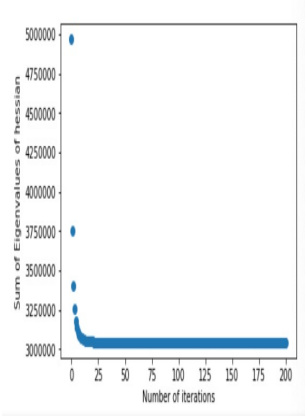


Figure 3.18: Decay of hessian over time for dependent prior.

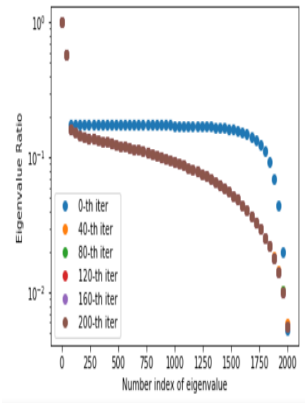


Figure 3.19: Decay of eigenvalue ratios over time for dependent prior.

Examining the decay of hessian and eigenvalue ratio, we find that the eigenvalues decay similar to the independent case and converge to similar value as the independent case. The initial trace is greater than that of the independent case as more dependency introduced. Also, Figure 3.19 verifies the similar decay with the independent case, which concludes similar results of most decay happen at first and then it converges.

# Chapter 4

# Conclusion and Future Work

## 4.1 Conclusion

As we can see from discussions before, Kalman filter is a great method that combines statistical background and applies to the PDE settings of an engineering problem. It did a great job in reducing the variance of the Hessian matrix and also reducing the gap between the true value and the predicted value. It triggers the Gaussian distribution more and more centered at the mean, which is verified by the eigenvalue ratio. Also, from the inspection on the starting point, we find they can have different starts but the error converges in the end, suggesting that the irrelevance of the starting point. We also removed the independence and simple mesh and also find out the more complex initial condition actually fosters the diffusion.

## 4.2 Future Work

As this report is just a starting point of inspection on the Kalman filter, many metrics and algorithms are still to explore. For example, the data in the real world maybe not complete, that is, we may have some timestamp that does not have the observation point available, while at other times we may

have duplicate data, so we might need to modify algorithm later and try algorithms such as extended Kalman filter and ensemble Kalman filter to function accordingly. Also, advection-diffusion equation has no control factor involved in the prediction phase, we may examine another PDE with the control factor and analyze the potential noise introduced by that as well.

# Bibliography


[1] T. Arbogast and J. L. Bona. *Methods of Applied Mathematics*. University of Texas at Austin,Austin, TX, 2008.

[2] K. B. Petersen and M. S. Pedersen. *The matrix cookbook*. University of Waterloo, 200 University Avenue West Waterloo, ON, Canada N2L 3G1, 2012.

[3] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations. An Introduction*, volume 92 of *Unitext*. Springer, 2016.

[4] T. J. Sullivan. *Introduction to Uncertainty Quantification*. Springer, 2017.

[5] U. Villa, N. Petra, and O. Ghattas. hIPPYlib: an Extensible Software Framework for Large-scale Deterministic and Bayesian Inverse Problems. 2016.

[6] Wikipedia contributors. Weak solution — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Weak_solution&oldid=791138101`, 2017. [Online; accessed 7-July-2019].

[7] Wikipedia contributors. Mass diffusivity — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Mass_diffusivity&oldid=890826942`, 2019. [Online; accessed 27-June-2019].

[8] Wikipedia contributors. Partial differential equation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Partial_differential_equation&oldid=903492467, 2019. [Online; accessed 11-June-2019].

[9] Wikipedia contributors. Trace (linear algebra) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Trace_(linear_algebra)&oldid=902571809, 2019. [Online; accessed 30-June-2019].

# Vita

Haocheng An was born in Beijing, China. He received the Bachelor of Science degree in Mathematics from the University of Texas at Austin with high honor in Fall 2017.

He worked under the guidance and supervision of Professor Robert van de Geijn in 2016. His research topic is on the condition number estimation of matrices.

Later in Summer 2017 and 2018, he worked as a software development engineering intern in Cisco Systems and Oracle Corporation respectively. Upon graduation, he will join Amazon's Fulfillment by Amazon(FBA) team upon his graduation to foster the storage, delivery, and transportation of inventories.

Permanent address: anhaocheng@hotmail.com

This report was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.