Sequential Inference For Dynamical Systems

Malcolm Morrison
Department of Physics, University of Otago
Supervisor: Dr. Colin Fox

A dissertation submitted for the fulfillment of a MSc in Physics, University of Otago, 2015.
Abstract

Sequential inference methods have played a crucial role in many of the technological marvels that we use today, from GPS and navigation systems [1] to machine learning [2]. Most current methods, such as the unscented Kalman filter (UKF) make several, occasionally crippling assumptions which allow them to work efficiently and accurately for approximately linear dynamics. The problem with this is that the majority of systems are not linear. Inference methods fully representing the dynamics and probability distributions were considered infeasible in the early days of sequential inference [3]. However, with the capabilities of modern computers this is no longer the case. In this thesis we propose a method to evolve a probability distribution on a dynamical system explicitly. This is done by using a finite volume partial differential equation solver to solve the continuity equation, combined with Bayesian observations. We present an example case of the simple pendulum and compare this with the UKF to examine several advantages.
# Contents

1 Introduction 1

2 Background 3
   2.1 Inference 4
      2.1.1 Probability 6
      2.1.2 Dynamical systems 7
   2.2 Computational Methods 10
      2.2.1 The Kalman Filter 12
      2.2.2 Unscented Kalman Filter 15
   2.3 The Pendulum 17

3 Early Work 21
   3.1 Partial Differential Equation Solvers 21
   3.2 Explicit ODE Solvers 23
      3.2.1 Runge-Kutta 4-5 24
   3.3 Crank-Nicholson 26

4 Finite Volume Method 29
   4.1 Finite Volume Solver 29
      4.1.1 Further Restrictions 32
   4.2 Consistency 34
      4.2.1 Kullback-Leibler Divergence 35
      4.2.2 Numerical Tests 36
   4.3 Inference 40

5 Application 41
   5.1 Behaviour Of Method 41
   5.2 Simple Pendulum 46
   5.3 Ambiguous Measurement and Multi-Modality 54
      5.3.1 Simulations 55

6 Conclusion 58

References 60
<table>
<thead>
<tr>
<th>Appendices</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1  Summary Of Method</td>
<td>63</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In the mid 1800’s a patient goes to see a doctor at the University of Edinburgh Medical School. After a small amount of time Dr Bell remarks that he knows the man was once in the army, not long discharged and that he was “...a non-commissioned officer in a Highland regiment stationed in Barbados.” The man was amazed, for Bell was correct on every count and they had never spoken a word to each other. Bell explained how he knew all of this: “... the patient was a respectful man, but did not remove his hat. They don’t in the army, but he would have learned civilian ways had he been long discharged. He has an air of authority and he is obviously Scottish. As to Barbados, his complaint is elephantiasis, which is West Indian and not British.” [4]

The observational powers and skills of inference possessed by Dr Joseph Bell would be immortalised in 1886 as the inspiration for the character Sherlock Holmes. Given very little evidence, Holmes could always reach the truth of a situation with unerring accuracy.

Through extraordinary cases like Dr John Bell and Sherlock Holmes, but also our everyday life, one can see the incredible power of inference that humans posses. We can tell by the way that someone runs how fit they are. Just by looking at the way a bag moves in someone’s hands we can roughly deduce the weight of its contents. If we see someone wearing a balaclava carrying a computer out a broken window at night we feel safe in assuming that they are a thief. However, these processes are surprisingly difficult to put into a formal framework. Even in a very intuitive case such as with the thief it is very hard to describe our mental process in a manner that one could program into a computer.

People have been trying to figure out how we deduce things using only limited knowl-
edge for centuries. Starting with Aristotle’s deductive reasoning, which was eventually encompassed by Boolean algebra in 1916 [5]. These algebras allow us to see what conclusions could be made with only the *a priori* knowledge of whether something is true or false and how they relate to one another. In the interim period, deductive and plausible reasoning was generalized to representing uncertain knowledge as probabilities from 0 to 1. People such as Rev. Thomas Bayes, and later Pierre-Simon Laplace extended upon this, culminating in Bayes’ Theorem, a formula that tells one how to update this uncertain knowledge with some new, related knowledge [6].

With the advent of computers there has been a desire to create predictive systems that utilise this framework to infer properties of a system through observing a sequence of events in real-time. These “sequential inference” methods have been developed over the years that do this but only for very specific circumstances. For example, one of the most well known prediction algorithms, the Kalman filter, only works for linear systems and falls apart when it encounters non-linearities [7]. This presents a problem in most practical situations as very few systems of interest are perfectly linear. In this thesis I will describe a new method of sequential inference that can work well for both linear and non-linear systems.

To put this work into perspective, the Electronics group here at the University of Otago are trying to weigh cows as they walk across a weighbridge. Currently measuring a cow’s weight is done by stopping the cows on the weighbridge and reading off the measurement once the bridge has stopped oscillating. This takes a very long time when done for large numbers of cows. Therefore, a faster measurement process is desired.

This is a generic sequential inference problem: there is a system that we wish to know about; the cow, and there is a means of gaining information about this system; the real-time force measurements from the weighbridge it as the cow walks. We want to use these measurements to predict the mass of the cow. However, cows are not linear systems. They have four legs, many bones and muscles, and don’t all walk the same way. Kalman filters and their non-linear extensions such as the unscented Kalman filter (UKF) [8] have been met with limited success, so a more accurate method was required.
Chapter 2

Background

For illustration purposes I would like to look at the theft example from the previous chapter. Say that we see the balaclava wearing person taking a computer out of a broken window at night. We would safely assume that this person is stealing the computer. Let’s walk through the reasoning to get to this conclusion.

Firstly observation: there is a person. People have the capacity to steal but they have the capacity to do many other things too, so we can’t make any good guess as to what they are doing.

Second observation: they are carrying a computer. Clearly, we are going to draw on past experiences to figure out what they might be doing with it. They may have just bought it, they may be transporting it, might have stolen it, but it is very unlikely that they are using it to clean their car because that would be nonsensical. Using your existing knowledge of what people do with computers and the observation that this person has a computer, you can make a far more educated guess at what this person is doing by effectively eliminating anything not related to the transport of computers.

Third observation: they are leaving through a broken window. Using our knowledge of why people go through broken windows and combining this with why people carry computers, we can deduce that these two things are most likely related. But still they could just be protecting their own property from looters and the window was the easiest
way out. But after just three facts we could make a well founded guess that they are stealing. And after adding the balaclava and the fact that it is night time, we can be almost certain that this person is a thief.

An interesting note about this example is that each observation has a different weight. For example if we just had three facts; that a person (1) was leaving through a broken window (2) with a computer (3), then we could guess that they are stealing. However a different three; that a person (1) is wearing a balaclava (2) at night (3), then we would probably guess that they were just warming their face. The two main reasons one wears a balaclava is to steal things or keep one’s face warm and we know that it is colder at night, hence the guess. However we can conclude that the more observations we have, the more accurate our guess is.

Through this example you can see that we need a way to represent our existing knowledge and combine it with a new observation. This was reasoned out by Bayes and formalized into what is now called Bayes’ Theorem.

In the above case we want to know the probability that they are a thief (\(A\)), denoted \(P(A)\). We make the observation that they are wearing a balaclava (\(B\)). What we need is a way to update the probability \(P(A)\), to the probability that they are a thief given that they are wearing a balaclava, denoted \(P(A|B)\), which is clearly going to be greater.

Note that we are thinking of probabilities as a representation of our current knowledge. What it says is: given our observations and past experience, what is the chance this person is currently stealing.

For this we use Bayes’ Theorem [6]:

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.1}
\]

This is the rule how to update our knowledge of our state, \(P(A)\), with the knowledge that is gained by an observation, \(B\). The term \(P(B|A)\) is the probability of the observation, \(B\), given the state is actually in \(A\), e.g. the probability that someone is wearing a balaclava given that they are stealing. Bayes’ Theorem is a very general statement and can apply to any system that we can make observations on[6].

2.1 Inference

Bayes’ Theorem is an incredibly powerful tool and significant result. The previous section explained the basics of equation 2.1. In this section we shall examine just what makes it so special and how it can be used.
Firstly, I would like to point out that the full derivation is remarkably simple. If we begin with the desire to have a quantitative rule for updating how plausible something is given some new information. If one has the desiderata that it will:

- represent degrees of plausibility as real numbers,
- have a qualitative correspondence with common sense\(^1\),
- be consistent,

then plausibility must be represented by probability and the only possible rule is Bayes’ Theorem \(^6\). So if we view probability as a representation on our knowledge, then equation \(^2.1\) is the only consistent rule for learning.

To put this in the framework of experimental science, there are three important things: the model \((m)\), the state of the current variables \((S)\), and the experimental data \((D)\). Theoretically we know that, given a model and state, what the distribution of the data is: \(P(D|S,m)\). We do an experiment and yield a data set, \(D_1\). Then in order to interpret how this data set updates our understanding of the state, we must use Bayes’ Theorem:

\[
P(S|D_1,m) = \frac{P(D_1|S,m)P(S|m)}{P(D_1|m)}.
\]  

(2.2)

I cannot emphasize enough that this is the only consistent way of inferring how some new data affects our knowledge and understanding. In this, one can interpret Bayes Theorem as a statement on the fundamentals of scientific theory.

An example of how this can be used in a scientific context is in the search for the Higgs Boson with the Large Hadron Collider. The model, \(m\), is the Minimal Supersymmetric Standard Model (MSSM). The experiments can be viewed as determining the probability of different regions of MSSM parameter space \(S\). With this, equation \(^2.2\) tells us how, under the assumption that the universe obeys the MSSM, what a certain dataset says about the parameters of the MSSM. As we have covered, Bayes Theorem is the only consistent way of interpreting this dataset\(^2\).

\(^1\)“Common sense”, as presented in \(^6\) means that it will reduce to Boolean algebra in the limiting case of true and false, correspond to Polya’s syllogisms on plausibility \(^9\) (plausible reasoning such as used earlier), and gain knowledge as more observations are made.

\(^2\)An analysis of this is done in reference \(^10\).
Bayes’ Theorem has a compounding behaviour for multiple pieces of new information. Say, if we had another dataset, $D_2$, then equation (2.2) becomes:

$$P(S|D_2, D_1, m) = \frac{P(D_2|D_1, S, m)P(S|D_1, m)}{P(D_2|D_1, m)}$$

$$= \frac{P(D_2|D_1, S, m)}{P(D_2|D_1, m)} \left( \frac{P(D_1|S, m)P(S|m)}{P(D_1|m)} \right). \quad (2.3)$$

If the data sets are independent and identically distributed then this simplifies to:

$$P(S|D_2, D_1, m) = \frac{P(D_2|S, m)P(D_1|S, m)P(S|m)}{P(D_2|m)P(D_1|m)}. \quad (2.4)$$

2.1.1 Probability

Since probability is the mechanism that we are using to represent our knowledge, we will briefly cover its properties. Firstly, the sum of the probabilities of all possible states sum to one. For a continuous range of possibilities, this is:

$$\int_{\text{all space}} \rho(x) \, dx = 1, \quad (2.5)$$

for a probability density function (pdf) $\rho$, defined by:

$$\int_{x_i}^{x_j} \rho(x) \, dx = P(x_i < x < x_j), \quad (2.6)$$

where $P(x_i < x < x_j)$ is the probability that $x$ is within the range $[x_i, x_j]$.

Secondly, probabilities and pdfs can not be negative:

$$\rho(x) \geq 0, \forall x. \quad (2.7)$$

While this may seem to be just stating the obvious, it will become important in later chapters.

Finally, we want to be able to make estimates on the properties represented by a pdf. We do this by looking at expectation values. For continuous variables $\{x_i\}$, this is
computed by:

$$E[g(x)] = \int g(x)\rho(x) \, dx$$

(2.8)

where $g(x)$ is a function describing the variable of interest. This could be as simple as the position, velocity or something more complex, such as the energy of the system. In this thesis we will only look at the mean ($E[x]$) and variance ($E[x^2] - E[x]^2$).

The continuous version of Bayes Theorem is [11]:

$$\rho(x|z) = \frac{\rho(z|x)\rho(x)}{\rho(z)},$$

(2.9)

for the continuous variables $x$ and $z$.

### 2.1.2 Dynamical systems

We have covered how to do inference on a static system, or where all the observations had no time dependence. However, in this thesis we will be performing sequential inference, i.e. making a sequence of observations in real-time. We will be performing this on something called a dynamical system.

A dynamical system is where a fixed, deterministic rules governing things in a region of space behave in time. Many of these rules are in the form of differential equations, such as systems that obey Newton’s second law:

$$F = m\frac{d^2x}{dt^2}.$$  

(2.10)

Newtonian dynamics functions well for particle motion, where an object is approximated as a point mass. Because this is a deterministic rule, we can calculate where a particle will be at a future point in time if the force is known. The solution to equation 2.10 in one dimension with a constant force is:

$$x(t) = \frac{F}{2m}t^2 + \dot{x}(0)t + x(0),$$

(2.11)

where the dot denotes the time derivative.

Let’s say that one you throw a ball up into the air and make notes of where it is at different times. Assuming that you know the mass of the ball, and initial velocity

\footnote{There are some problems with this that can produce misleading results. This will be discussed in chapter 5.}
you throw it at. Then if you wished to know the value of the local gravity, you could rearrange equation 2.11 to find the inverse map. It can easily be seen that with just two perfect observations of position at known times, you can explicitly calculate the force. However, our eyes are not perfect, nor is any other measurement device. All measurements that we make will have some associated error.

Our measurements will actually be in the following form:

\[
\hat{x}(t) = \frac{F}{2m} t^2 + \dot{x}(0)t + x(0) + \epsilon \\
= x(t) + \epsilon,
\]  

(2.12)

where the hat denotes a measurement and \( \epsilon \), a random value from the noise. Directly calculating the inverse is now no longer possible due to this noise term. Because of this we must look to Bayes Theorem.

What we are wanting to know is the pdf \( \rho(F|\hat{x}_1) \); the likelihood of where the true value of \( F \) lies, given an observation \( \hat{x}_1 \). This is given by:

\[
\rho(F|\hat{x}_1) = \frac{\rho(\hat{x}_1|F)\rho(F)}{\rho(\hat{x}_1)}.
\]

(2.13)

The two distributions \( \rho(\hat{x}_1) \) and \( \rho(F) \) are chosen in a sensible way such as to encapsulate our assumptions and knowledge. For the purposes of parameter estimation, as we are doing here, the denominator can be effectively ignored by treating it as a normalization term [12].

If the system did not change by the time of the next observation \( \hat{x}_2 \), then we could just apply equation 2.3. However, now that we are making inference on a dynamical system, the ball would have moved by the next time we can record its position. So we need a way to propagate our pdf forward in time in order to make this update meaningful.

One common way to do this called a particle filter. This is where one randomly chooses a set of particles that are distributed as the relevant pdf and move them all forward in time by equation 2.10. In this case it would be taking these samples from the spatial distribution and evolving them forward by a sample from \( \rho(F|\hat{x}_1) \). One would then construct a new updated pdf from these samples at the new time. This would then function as our new prior.

More generally one has some function or process, \( f \), that advances the pdf forward
to the time of the next observation;

\[ f(\rho_1(F|\hat{x}_1)) \rightarrow \rho_2(F|\hat{x}_1), \]  

(2.14)

that obeys the dynamics as described by the dynamical system, e.g. equivalent to equation [2.10] in this case. The subscripts \( \rho_i \) denote that the pdf is for timestep \( i \).

Then the next observation updates our new state in the following manner:

\[
\rho_2(F|\hat{x}_2, \hat{x}_1) = \frac{\rho_2(\hat{x}_2|\hat{x}_1)\rho_2(F|\hat{x}_1)}{\rho_2(\hat{x}_2|\hat{x}_1)} = \frac{\rho_2(\hat{x}_2|F, \hat{x}_1)f(\rho_1(F|\hat{x}_1))}{\rho_2(\hat{x}_2|\hat{x}_1)},
\]

(2.15)

and so on.

While particle filters would use equation [2.10], we are going to deal with the pdfs explicitly. For this we are going to turn to a law that all continuous dynamic quantities obey: the continuity equation. This is a mathematical representation of the fact that if the amount of a certain quantity in a volume changes, then that amount of this quantity must have left or entered the volume, or been created or destroyed. This is:

\[
\frac{\partial \rho}{\partial t} = -\nabla \cdot j + \sigma,
\]

(2.16)

where \( j \) is the flux of said quantity and sigma is a source term that can creates and destroy the quantity.

In our case the quantity is probability and the flux is the pdf multiplied by a velocity field, \( f \) that describes the dynamics at every point in space. From the fact that probability must be normalized [2.5], it can’t be spontaneously created or destroyed, and therefore \( \sigma = 0 \).

This simplifies the continuity equation for probability to:

\[
\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho f)
\]

(2.17)

This is the equation that describes any dynamical system with probability, in the same way that equation [2.10] is an equation which describes a dynamical system with particles. The velocity field, much like the force in [2.10] provides the information on how the probability will actually move.

---

4We are allowed to do the steps contained in equations [2.14] and [2.15] due to the Markovian nature of dynamical systems. This means that all of the historical information of a state is completely contained in the current state [11].
For later reference, the integral form of this is written as:

\[
\frac{\partial}{\partial t} \int_{V_i} \rho \, dV + \int_{A_i} \rho (f \cdot \hat{n}) \, dA = 0
\]  

(2.18)

for a volume \( V_i \), where \( \hat{n} \) is the outwards facing normal vector of the surface \( A_i \).

In this thesis we seek to devise a sequential inference method for dynamical systems that solves equation 2.17 explicitly with a PDE solver to propagate our pdf forward in time to the next observation step. That is have a method that:

1. Represents the probability distributions explicitly.
2. Calculates the filtering or observation step explicitly. That is, for variables \( x \), use Bayes’ Theorem to find the posterior given an observation returning \( z \):

\[
\rho_i(x|z) = \frac{\rho_i(z|x) \rho_i(x)}{\rho_i(z)}
\]  

(2.19)

through explicit calculation.
3. Use a PDE solver for prediction step, i.e., propagate forward one step in time by solving equation 2.17 to get:

\[
\rho_i(x|z) \xrightarrow{\text{PDE solver}} \rho_{i+1}(x)
\]  

(2.20)

Calculating these explicitly is something that was considered foolhardy in the early days of sequential inference as it is very computationally intensive \[3\]. However, nowadays we have personal computers that are orders of magnitudes more powerful than then, so this is no longer a significant obstacle.

2.2 Computational Methods

Now that we have a mathematical understanding of inference and probability, let’s take a look at some of the existing techniques for sequential inference. This will be useful for later comparisons against our method. However first I would like to talk about the concept called the “order”of a computational method and of functions, such as velocity

\[5\]To get between equations 2.17 and 2.18 Gauss’ Theorem has been applied to the integral.
Suppose we have a one-dimensional function, $f(x)$, that describes the dynamics of a system, such as a bag swinging, or a plane flying through the air. This could be a velocity field like in equation 2.17. Then we can construct a Taylor expansion about the point $x_0$.

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + ... \tag{2.21}$$

A first order representation of this function about $x_0$ contains the terms up to the linear one and is therefore a linear approximation of $f$. Likewise a second order representation has all the terms up to the quadratic, third order to the cubic, etc.

A first order approximation of a function is fine, provided the function itself is dominated by the linear term in the region of $x$ for which we are concerned. Naturally, the closer to $x_0$ we look, the more accurate the value of the function we shall receive.

The order of a computational method is a very different concept. It is the rate of which the method converges to the true solution, as will be described in sections 3.1 and 4.2. To say that a method is $n^{th}$ order accurate means that the error with time step size $\Delta t$ is proportional to the step size to the $n^{th}$ power:

$$\text{error}(\Delta t) \propto \Delta t^n. \tag{2.22}$$

Finally, to say that a probability distribution is represented perfectly up until the $n^{th}$ order means that the expectation values of the spatial coordinate up to the $n^{th}$ power are exact. For example, a Gaussian distribution is perfectly represented with second order statistics:

$$\mu_x = \mathbb{E}[x] \quad \sigma_x = \sqrt{\mathbb{E}[x^2] - \mu_x^2}. \tag{2.23}$$

\footnote{For a full explanation see [13]}
2.2.1 The Kalman Filter

In the 1960’s Bayes’ theorem was well known, but directly calculating the evolution of a probability distribution and updating with observations was far more than even the best computers could handle. The sequential inference methods from this time needed to be very computationally efficient, ideally only computing with a few numbers. The Kalman filter is one such method that has been very successful and is still used today.

The Kalman filter is a sequential inference method that was invented in 1960 by Rudolf E. Kalman [7] and was adapted quickly by Stanley F. Schmidt to be used by the Apollo rocket as the basis for it’s guidance system [14]. It relies on several assumptions about the system that we are looking at. Firstly, it models our knowledge of the system, the pdf, as a Gaussian distribution. That is:

\[ \rho(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \] (2.24)

in one dimension, where \( \mu \) is the mean and \( \sigma \) is the standard deviation, or width, of the distribution. It also assumes that the noise in our observation is distributed as a zero-mean Gaussian of width \( \sigma_z \).

The multi-variate Gaussian distribution is:

\[ \rho(x) = \frac{1}{|\Sigma|^{\frac{n}{2}} (2\pi)^{\frac{n}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}, \] (2.25)

where \( n \) is the dimension, \( \Sigma \) is the covariance matrix, \( |\Sigma| \) its determinant, and \( T \)

\footnote{This is not technically true. The original formulation of the Kalman filter does not assume this and is constructed to minimize the estimated covariance. I have written it this way because this is the most common used form of the filter. The Kalman filter with Gaussian distributions is provably exact [15].}

\footnote{The covariance matrix is closely related to the standard deviation. In one dimension this is just \( \sigma_z^2 \).}
Secondly, the dynamics in the system are assumed to be linear with additive noise, of any dimension. For a particle at time step $k$ in position $x^k$, its position at the next time step is assumed to be:

$$x^{k+1} = Ax^{k-1} + Bu^k + \epsilon_x$$  \hfill (2.26)

where $A$ is the transition kernel, $Bu^k$ forms an effective acceleration vector, and $\epsilon_x$ is a zero-mean Gaussian distributed random vector with covariance $E_x$. This noise is called the dynamic, or process noise.

These approximations are very powerful, as Gaussian distributions moving under linear dynamics maintain their Gaussian shape. Further, two Gaussian distributions multiplied together, say, if we were to combine them through Bayes’ theorem, is also a Gaussian distribution. This means that the Kalman filter need only ever deal with Gaussian distributions, which are characterised completely by their mean and covariance, $\mu$ and $\Sigma$. For $n$ dimensions that is only a vector of length $n$ and an $n \times n$ matrix per Gaussian. This lack of numbers allows for a very computationally efficient process [15].

The Kalman filter then makes one further approximation. This is that the observation at time $k$ is related to $x^k$ by the linear map:

$$z^k = Cx^k + \epsilon_z$$  \hfill (2.27)

where $z^k$ is the observation vector, $C$ is the observation model matrix and $\epsilon_z$ is the measurement noise, again zero-mean Gaussian distributed with covariance $E_z$.

Note: To avoid confusion we will be using $\Sigma$’s to denote covariance matrices of our predictions and estimates. These will change at each time step. We will use the symbol $E$ to denote the covariance matrices that do not change, namely for the process and observation noise. This is not to be confused with the expectation value $E$.

What follows is the algorithm for the Kalman filter at time step $k$, with some initial conditions $x^0$ and $\Sigma_{xx}^0$. Note that the tilde is used to denote the estimates.

Kalman filter algorithm [15]:

1. Predict next state’s mean:

$$\tilde{x}^k = Ax^{k-1} + Bu^k$$  \hfill (2.28)
2. Predict next state’s covariance:

\[ \tilde{\Sigma}^{k}_{xx} = A \tilde{\Sigma}^{k-1}_{xx} A^T + E_x, \]  

(2.29)

where the superscript \( T \) denotes the transpose.

3. Calculate observation mean and covariance:

\[ \tilde{z}^k = C \tilde{x}^k \]  

(2.30)

\[ \tilde{\Sigma}_{zz} = C \tilde{\Sigma}^k_{xx} C^T + E_z \]  

(2.31)

4. Calculate the Kalman gain, \( K \), otherwise known as the correction weight:

\[ K^k = \tilde{\Sigma}^k_{xx} C^T \tilde{\Sigma}^{-1}_{zz} \]  

(2.32)

5. Update mean:

\[ \tilde{x}^k = \tilde{x}^k + K^k (z^k - \tilde{z}^k), \]  

(2.33)

where \( z^k \) is the observed data.

6. Update covariance:

\[ \Sigma^k_{xx} = (I - K^k C) \tilde{z}^k, \]  

(2.34)

where \( I \) is the identity matrix.

This is the basic Kalman filter.

As mentioned earlier, this has been a very successful sequential inference technique. It was invented in 1960 and within a few years it was already been used in guidance systems. It is still in common use even today in places such as navigation systems such as GPS [1] devices and in machine learning algorithms [2]. It’s strength comes from it’s simplicity and efficiency.

The approximation that the noise is Gaussian is a very good assumption in numerous situations. The linear dynamics works very well for simple kinematic systems, such as missile guidance and navigation (like in equation 2.12). However, it becomes unreliable as soon as the system becomes more complex. Linear systems are the minority and approximating complex systems as linear at best will provide limited insight into said system. At worst it will provide completely wrong results. Another thing is that the approximation of our knowledge staying Gaussian shaped is no longer valid under non-
linear dynamics.

The observation being a simple linear map is also an assumption that in not true in
many cases, such as measuring the tension force on a pendulum\footnote{This measurement also
leads to a multi-modal distribution, while the Kalman filter only works for uni-modal
distributions.}. This measurement also

There have been many improvements on the Kalman filter that can been applied to
non-linear dynamics, such as the extended Kalman filter. This linearizes the dynamics at
every time step then applies the basic Kalman filter. It effectively calculates the Taylor
expansion up to the first order term as in equation\ref{2.21} which gets much more intensive
the higher the dimension\footnote{For reference sake, this involves calculating something called the Jacobian.}

\section{Unscented Kalman Filter} \label{2.2.2}

In the mid 90’s Jeffrey K. Uhlmann came up with the unscented transform, named
rather bizarrely after a deodorant can\footnote{See section\ref{5.3}.}. This is a method of representing a prob-
ability distribution by a set of $2n + 1$ weighted points, where $n$ is the dimension
of the system, that represent the statistics
perfectly up to second order. These points
are called sigma points and can be evolved
instead of using the mean and covariance.

The sigma points are calculated from
the mean and covariance by:

\begin{align}
\chi_0 &= \mu, \\
\chi_i &= \mu + \sqrt{(n + \kappa)\Sigma_{xx}}_i, \\
\chi_{i+n} &= \mu - \sqrt{(n + \kappa)\Sigma_{xx}}_i,
\end{align}

\begin{align}
W_0 &= \frac{\kappa}{n + \kappa}, \\
W_i &= \frac{1}{2(n + \kappa)},
\end{align}

\begin{align}
W_{i+n} &= \frac{1}{2(n + \kappa)}.
\end{align}

\begin{equation}
(2.35)
\end{equation}
where $\{\chi\}$ is the set of sigma points with associated weights, $W$. The method of square-rooting of the matrix does not matter but is often taken to be the Cholesky decomposition. The subscript on the square root denotes the $i$th row or column of the resulting matrix and $\kappa$ is a real, tuneable parameter.

Here the process and observation models no longer require linear maps. These are now generalised as functions $F$ and $H$ at time step $k$ by:

$$
\begin{align*}
x(k+1) &= F(x(k), u(k), k) \\
z(k) &= H(x(k), u(k), k)
\end{align*}
$$

where $u$ is a control or acceleration vector.

The algorithm for the Unscented Kalman Filter at time step $k$ is as follows, again using a tilde as before to denote estimates.

1. Use current mean and covariance matrix to create a set of sigma points as in equation (2.35):

$$
(x(k), \Sigma_{xx}(k)) \rightarrow \{\chi(k), W(k)\}. \tag{2.38}
$$

2. Pass the sigma points through your process mode to compute the expected points:

$$
\tilde{\chi}_i(k+1) = F(\chi_i(k), u_i(k), k). \tag{2.39}
$$

3. Predict mean, and then the covariance by the outer product:

$$
\begin{align*}
\tilde{x}(k+1) &= \sum_{i=0}^{2n} W_i \tilde{\chi}_i(k+1), \\
\Sigma_{xx}(k+1) &= \sum_{i=0}^{2n} W_i (\tilde{\chi}_i(k+1) - \tilde{x}(k+1)) (\tilde{\chi}_i(k+1) - \tilde{x}(k+1))^T.
\end{align*} \tag{2.40, 2.41}
$$

4. Pass the sigma points through the observation model to obtain observation sigma points, $Z$:

$$
\tilde{Z}_i(k+1) = H(\tilde{\chi}_i(k+1), u_i(k), k). \tag{2.42}
$$

5. Calculate the observation mean and the innovation covariance:

$$
\begin{align*}
\tilde{z}(k+1) &= \sum_{i=0}^{2n} W_i \tilde{Z}_i(k+1),
\end{align*} \tag{2.43}
$$
\[ \Sigma_{zz}(k+1) = 2n \sum_{i=0}^{2n} W_i (\tilde{Z}_i(k+1) - \tilde{z}(k+1)) (\tilde{Z}_i(k+1) - \tilde{z}(k+1))^T. \] (2.44)

6. Calculate the cross correlation:

\[ \Sigma_{xz}(k+1) = \sum_{i=0}^{2n} W_i (\tilde{\chi}_i(k+1) - \tilde{x}(k+1)) (\tilde{Z}_i(k+1) - \tilde{z}(k+1))^T. \] (2.45)

7. Calculate the gain:

\[ W = \Sigma_{xz}(k+1) \Sigma_{zz}^{-1}(k+1). \] (2.46)

8. The updated mean and covariance are then given by:

\[ 
\begin{align*}
\mathbf{x}(k+1) &= \tilde{\mathbf{x}}(k+1) + W(k+1)(\mathbf{z}_{\text{obs}}(k+1) - \tilde{\mathbf{z}}(k+1)) \\
\Sigma_{xx}(k+1) &= \Sigma_{xx}^*(k+1) - W(k+1)\Sigma_{zz}W^T(k+1)
\end{align*}
\] (2.47) (2.48)

where \( \mathbf{z}_{\text{obs}}(k) \) is the observed data at time \( t = k\Delta t \).

The Unscented Kalman Filter is a robust sequential inference method that is widely used today and will serve as a good benchmark to compare with the inference method introduced by this thesis.

2.3 The Pendulum

For the purpose of comparison we shall be take our dynamical system to be a simple pendulum. The simple pendulum, as shown in figure 2.3, is a mass, \( m \), on a string of fixed length, \( l \) that swings in a plane. The usual 2D Cartesian description of this is needlessly complicated, it is easier to describe the system in terms of the angular displacement and velocity, with the pendulum hanging down to be \( x = 0 \). By doing this we can fully represent the two dimensional system (4D in phase-space) as having only one dimension (2D in phase-space).

Figure 2.3: The simple pendulum. A point mass (\( m \)) hanging at fixed length (\( l \)) under the influence of gravity.
It is important to work in phase-space rather than ignoring the velocity component as this provides a complete description of the state that the system is in. For example, if I were to tell you the position of a pendulum, then you have no idea of where it will be at the next second. If you wanted to predict where the pendulum will be the next time you looked you need to know the the velocity as well. Then, combining this with the known dynamics/acceleration, we have a complete picture of its future behaviour.

Using \( x \) and \( v \) to denote the angular displacement and angular velocity, the phase-space velocity field for a pendulum is:

\[
\mathbf{f} = (\dot{x}, \dot{v}) = \left(v, -\frac{g}{l}\sin(x)\right),
\]

(2.49)

where \( g \) is the acceleration due to gravity and a dot denotes the time derivative.

The pendulum is of interest to us for a few reasons. Firstly, it is a Hamiltonian system.

To say that a system is Hamiltonian means two things:

• That one can write an expression for the Hamiltonian which is related to, and in many cases exactly the total energy in the system. For the simple pendulum this is:

\[
\mathcal{H} = \frac{1}{2} m(lv)^2 + mgl(1 - \cos(x)).
\]

(2.50)

This is time independent and therefore conserved.

• It also means that the system obeys Hamilton’s equations of motion. These are

\[
\dot{q} = \frac{\partial \mathcal{H}}{\partial p} \quad \text{and} \quad \dot{p} = -\frac{\partial \mathcal{H}}{\partial q},
\]

(2.51) \hspace{1cm} (2.52)

for canonical variables \( q \) and \( p \). For the pendulum these are \( x \) and \( ml^2v \), which is the position and momentum of the pendulum respectively.

Hamiltonian systems have one consequence in particular that we are interested in. For the velocity field \( \mathbf{f}_{\text{canonical}} = (\dot{q}, \dot{p}) \), the divergence is 0, i.e.

\[
\nabla \cdot \mathbf{f}_{\text{canonical}} = \frac{\partial}{\partial q} \left( \frac{\partial \mathcal{H}}{\partial p} \right) + \frac{\partial}{\partial p} \left( -\frac{\partial \mathcal{H}}{\partial q} \right).
\]
However, this also holds true of the variables that we are using:

\[
\nabla \cdot \mathbf{f} = \frac{\partial}{\partial x} (v) + \frac{\partial}{\partial v} \left( -\frac{g}{l} \sin(x) \right)
\]

\[= 0. \quad (2.53)
\]

You are probably asking, why is this fact relevant? Let’s take another look way back to the continuity equation in it’s differentiable equation form (equation 2.17). Using a vector identity to expand out the left hand side, we get:

\[
\frac{\partial \rho}{\partial t} = - \nabla \cdot (\rho \mathbf{f})
\]

\[= - \mathbf{f} \cdot \nabla (\rho) - \rho (\nabla \cdot \mathbf{f}). \quad (2.54)
\]

Now we can eliminate the second term, which yields:

\[
\frac{\partial \rho}{\partial t} = - \mathbf{f} \cdot \nabla (\rho). \quad (2.55)
\]

It may not seem that significant a result just by looking at equation 2.55, but let’s take a look at equation 2.54 again. Both the terms on the right hand side deal with different kinds of flow. The first term is the flow that travels in the direction of the velocity field. In fluid dynamics this is sometimes called the convection term. The second term is the one that we eliminate in Hamiltonian dynamics which tells us how the probability distribution expands or shrinks. In fluid dynamics this is referred to as the diffusive term. In eliminating this we know that the area of the probability distribution will not expand or shrink if we had a perfect PDE solver. This is known in Hamiltonian mechanics as Liouville’s theorem.

The second reason that we are using the simple pendulum is because of its non-linear properties.

Looking at regions where \( x \) and \( v \) are small, then we can apply the small angle approximation:

\[
\mathbf{f} \approx (v, -\frac{g}{l}x) \quad (2.56)
\]

This form of the simple pendulum is completely linear and is what most undergraduate physics students will be most familiar with. This is, in fact the dynamics for a

\[^{11}\text{We have included } v \text{ here because if } v \text{ is too large then } x \text{ will eventually leave the area where this approximation is valid.}\]
simple harmonic oscillator (SHO). In Figure 2.4, it is the area where the lines are circular near the center. If you would cast your mind back to section 2.2, you will remember that the Kalman Filter and the UKF work very well when under linear dynamics.

Compare with the position $x = \pi = 180$ and $v = 0$. This is a saddle point, which cannot exist in linear systems. So if we consider initial positions with $v_0 = 0$, the probability distribution will stay on the same closed path (see Figure 2.4) and therefore we effectively have scalable non-linearity dependant on the initial $x_0$. 

Figure 2.4: The separatrix for a simple pendulum shows the lines of motion a pendulum will follow in phase-space, given initial conditions. This is only possible to make because of the divergence-less property of time independent Hamiltonian systems. The red line is with initial conditions $(x = \pm \pi, v = 0)$. Source: Wikimedia commons
Chapter 3

Early Work

In this chapter I shall provide a basic introduction to partial differential equation (PDE) solvers and then cover all of our initial attempts at constructing a solver that meets our needs. This will highlight the care required when choosing a solver and the importance of the restrictions imposed in the next chapter. The final method used is described in the Chapter 4 and summed up in Appendix A.1.

3.1 Partial Differential Equation Solvers

A PDE solver is a computational method that attempts to approximate a solution to a given PDE. For the most part, these are used in situations where the analytical solution is not obtainable. However, in general no PDE solver will ever provide an exact solution. A good PDE solver is accurate enough, in regards to the properties that one is interested in, that it provides the correct solution within an predetermined level of error.

Partial differential equations are equations which contain more than one kind, but finite number, of partial differential operators. The general form is:

\[ F\left(x, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \ldots\right) = 0, \tag{3.1} \]

for some function \( F \), variables \( x = (x_1, x_2, \ldots) \) and function of interest \( u \).

In our case we are looking at the continuity equation (2.17) with the velocity field in equation (2.49)

\[ \frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x} \left(v\rho\right) - \frac{\partial}{\partial v} \left(-\frac{g}{l} \sin(x)\rho\right), \tag{3.2} \]

and the function of interest is \( \rho \).
A “solution” is a function $\rho(x,v,t)$ that satisfies equation 3.2 and the desired boundary conditions. The way that we deal with this is to discretize the function in some way. This could be by only taking values at certain points in space and interpolates (Figure 3.1), taking the integral of the function in an area, or some other way.

When one breaks a continuous function into a discrete one, information is lost. Logically, the coarser the discretization, the more information lost. This lost information can manifest in many different ways. The dynamical system may not behave properly, there may be boundary problems due to the fact we are representing the system in only a small region of space, or something else that leads to less accurate results. A part of creating a PDE solver is choosing how you discretize your system so that the properties one cares about are solved accurately.

People have come up with two measures of testing whether a PDE solver is not useful. These are [18]:

1. Consistency: Say we have a partial differential operator $P$, operating on a function $u$ to make a PDE, $P_u = f(t,x)$, with some $f$, and a discrete approximation to this, $P_{\Delta x \Delta t} v = f(t,x)$, for some function $v$. A Solver is said to be consistent if for any smooth function $\phi(t,x)$:

$$P \phi - P_{\Delta x \Delta t} \phi \to 0 \quad \text{as} \quad \Delta x, \Delta t \to 0. \quad (3.3)$$

This is a measure of whether or not the PDE solver represents your PDE properly.

2. Stability: A discrete PDE, $P_{\Delta x \Delta t} v^k_i = f(t,x)$, is stable in the region, $\Lambda$, if there is an integer, $J$, such that for any positive time, $T$, There is a constant, $C$ such
CHAPTER 3. EARLY WORK

that:

$$||v^k||_{\Delta x} \leq C \sum_{j=0}^{J} ||v^j||_{\Delta x},$$  \hspace{1cm} (3.4)

for $0 \leq k\Delta t \leq T$, $(\Delta t, \Delta x) \in \Lambda$, where $|| \cdot ||_{\Delta x}$ is the L2-norm of a grid function.

This means that the growth of a permutation from the true solution is bounded by some value.

The Lax-Richtmyer Equivalence Theorem tells us that the solution provided by any PDE solver, that is both consistent and stable, converges to the true solution as the discretization goes towards zero ($\Delta x, \Delta t \to 0$) \cite{19}.

One very common method used to discretize a system is the “method of lines” \cite{20}. This is where one separates the spatial and time dimensions. The spatial domain is discretized in some way, such as in Figure 3.1. This changes a complex PDE into a much simpler system of coupled ordinary differential equations (ODE) by changing all of the spatial derivatives into discrete approximations. The ODE solving methods are well established and understood \cite{21}.

3.2 Explicit ODE Solvers

Explicit methods are where one approximates the time gradient $\frac{\partial y}{\partial t}$ and then explicitly steps forward in time based off this. The simplest of these is the explicit Euler method, also known as a first order Runge-Kutta method.

The explicit Euler approximates the partial derivative as a finite difference. In one dimension at time $t_k = k\Delta t$, this is:

$$\frac{\partial y}{\partial t} \approx \frac{y_{k+1} - y_k}{\Delta t}. \hspace{1cm} (3.5)$$

Applying this, to an ODE, we get the explicit Euler method for a single time step:

$$\frac{y_{k+1} - y_k}{\Delta t} = f(t_k, y_k), \hspace{1cm} (3.6)$$

for some function $f$, where $t_k$ is the time at step $k$.  

Figure 3.2: Example of the forward Euler method. True solution in blue and Euler method output in red.
This method is well known to be consistent [21]. However, this method is only stable in very specific cases and since it is only a first order method its accuracy is limited.

### 3.2.1 Runge-Kutta 4-5

Our first true attempt at a PDE solver was using the method of lines with a standard Runge-Kutta 4-5 method (RK45), otherwise known as the Runge-Kutta-Fehlberg method. The RK45 is a very commonly used method in physics and engineering [22], and is a fourth order explicit ODE solver that compares with a fifth order one to adapt the time step as necessary to maintain a level of accuracy.

Higher order Runge-Kutta methods choose the gradient to be the combination of several gradients calculated at several optimized points. The 4th order method is as follows [13]:

\[
K_1 = f(t_k, y_k), \tag{3.7}
\]

\[
K_2 = f(t_k + \frac{1}{2} \Delta t, y_k + \frac{1}{2} K_1 \Delta t), \tag{3.8}
\]

\[
K_3 = f(t_k + \frac{1}{2} \Delta t, y_k + \frac{1}{2} K_2 \Delta t), \tag{3.9}
\]

\[
K_4 = f(t_k + \Delta t, y_k + K_3 \Delta t), \tag{3.10}
\]

and then finally

\[
y_{k+1} = y_k + \Delta t \left( \frac{1}{6} K_1 + \frac{1}{3} K_2 + \frac{1}{3} K_3 + \frac{1}{6} K_4 \right). \tag{3.11}
\]

Once again, this method is consistent, but stability is not easily determined for the number of dimensions that we use. Instead we ran several simulations to gauge this.

For our case, the PDE is the continuity equation as in equation 3.2. We shall be using the dynamics for the small angle approximation/simple harmonic oscillator (SHO) with all constants set to one for the purposes of testing the capabilities of PDE/ODE solvers. This is because there is an analytical solution that we can compare against, provided we use Gaussian distributions. Under these dynamics in phase-space equation 3.2 becomes:

\[
\frac{\partial \rho}{\partial t} = -v \frac{\partial \rho}{\partial x} + x \frac{\partial \rho}{\partial v} \tag{3.12}
\]

This is a PDE, however we want to create a system of coupled ODEs. For this we shall be dividing our spatial dimensions up into a square grid of \( n \times n \) points of spacing.
\( \Delta x = \Delta v \). For a discrete representation of our spatial partial derivatives we shall be using central finite differences. In one dimension this is:

\[
\frac{\partial \rho(x)}{\partial x} \approx \frac{\rho(x_{i+1}) - \rho(x_{i-1})}{2\Delta x},
\]

where \( x_i \) is the value of \( x \) at the \( i \)th point.

This allows us to obtain a coupled system of ODEs:

\[
\frac{\partial \rho_{ij}(t)}{\partial t} = -v_j \frac{\rho_{i+1,j}(t) - \rho_{i-1,j}(t)}{2\Delta x} + x_i \frac{\rho_{i,j+1}(t) - \rho_{i,j-1}(t)}{2\Delta v},
\]

where \( \rho_{ij}(t) = \rho(t, x_i, v_j) \).

The initial conditions used for our pdf, \( \rho(0, x, v) \), was a Gaussian with mean \( \mu = (x_0, 0) \) and covariance \( \Sigma = \sigma_0 I \), where \( I \) is the identity. The \( \rho_{ij}(0) \)’s were stored in a into a vector and the system of ODEs was solved with a RK45 algorithm in the Scipy python package (odeint) to obtain the following simulated solutions.

We quickly discovered that this was not stable and the pdf did not stay positive. Figures 3.3 and 3.4 are of a representative simulation. There are several obvious problems with this. Firstly it does not preserve positivity and the total probability is not constant. The pdf also did not stay Gaussian as it should have. A wake of positive and negative values follows the bulk of the probability, as is evident in Figure 3.4.

These problems demonstrate the difficulties we will need to account for when choosing our final PDE solver. It was unclear whether positivity something that we should just enforce explicitly, or have our solver implicitly deal with them. To improve the properties...
of our ODE solver we next experimented with the Crank-Nicholson method.

### 3.3 Crank-Nicholson

The Crank-Nicholson method is an implicit method [23]. While an explicit method uses the current state to calculate the state of the system at the next time by evaluating the gradient at the current state, an implicit method solves an equation that involves both the current state and the next one. In general, implicit methods are more stable than explicit ones, but are more computationally intensive [13].

We chose the Crank-Nicholson method because it is used as a textbook example for solving the 2d heat diffusion equation [23]. Since our pdf is in a 2d system and being evolved in a similar manner, we thought that the Crank-Nicholson method’s stability properties would be well suited for our situation. The Crank-Nicholson method is also unconditionally stable for 2d heat diffusion and many other cases [13].

The Crank-Nicholson method is the halfway point between the explicit Euler method and the implicit Euler method. The backwards Euler is:

\[
\frac{y_{k+1} - y_k}{\Delta t} = f(t_{k+1}, y_{k+1}).
\]  

The Crank-Nicholson is then:

\[
\frac{y_{k+1} - y_k}{\Delta t} = \frac{1}{2} \left( f(t_k, y_k) + f(t_{k+1}, y_{k+1}) \right).
\]

The Crank-Nicholson method applied to the SHO (equation 3.14) is:

\[
\frac{\rho_{k+1}^{i,j} - \rho_k^{i,j}}{\Delta t} = \left( \frac{v_j \rho_{k+1}^{i+1,j}(t) - \rho_{k+1}^{i-1,j}(t)}{2 \Delta x} - \frac{x_i \rho_{k+1}^{i,j+1}(t) - \rho_{k+1}^{i,j-1}(t)}{2 \Delta v} \right) + \left( \frac{v_j \rho_k^{i+1,j}(t) - \rho_k^{i-1,j}(t)}{2 \Delta x} + \frac{x_i \rho_k^{i,j+1}(t) - \rho_k^{i,j-1}(t)}{2 \Delta v} \right)
\]

where \( \rho_{ij}^k = \rho(t_k, x_i, v_j) \).

This was run with the initial condition of a Gaussian pdf at \( v_0 = 0 \). An example simulation is shown in figures 3.5-3.8.

As mentioned earlier, under simple harmonic motion, a Gaussian should evolve as a Gaussian. It is obvious from Figure 3.6 that this is not the case in our simulations. After half a period the tail end of the pdf has dipped into negative values. Figure 3.7 is the simulation after a full period. We can see here that the negative tail has become a...
CHAPTER 3. EARLY WORK

kind of wake, alternating negative and positive.

The analytical solution for this case is exactly the same as the initial conditions, Figure 3.5, so there is clearly something wrong with this simulation. The most obvious two reasons being that the pdf has negative values, which from section 2.1.2 we know is not allowed, and that the computed solution does not match up with the analytical solution. However, this may be arising from the same issue that allows negative values.

One possible reason for this may be the fact that the time step is too large. However when running the same conditions with a time step a tenth of the size we see no practical difference from later simulations (Figure 3.8).

What about the normalization of the pdf, as calculated by equation 2.5? Does this at least stay constant? Figure 3.9 tells us that it does not. This is a plot of the total normalization of the pdf over one full revolution. While it initially stays near one, it
soon starts oscillating, revealing that there is also an issue here to tackle.

![Graph showing Crank-Nicolson SHO normalization over time]

Figure 3.9: The normalization unstable, as seen here. Once the errors are large enough the normalization diverges from 1, in this case they oscillate about 1 with increasing amplitude. This is the normalization for the simulation in Figures 3.5-3.7

The exploration that was done into these solvers led us to focus our efforts on a PDE solver that implicitly preserves normalization: the finite volume method.
Chapter 4

Finite Volume Method

The goal of this thesis is to construct a method of performing inference on a system by evolving the probability in phase-space directly through solving the continuity equation. We shall be using a finite volume method to solve the PDE. A short summary of this method can be found in appendix A.1.

4.1 Finite Volume Solver

Finite volume methods are a type of PDE solver that are designed for systems with conservation laws. More specifically; systems where there is some continuous quantity that moves under a velocity field [24]. This is a very sensible choice for our purposes as it is derived from the continuity equation itself. Our starting point is the integral form of the continuity equation:

\[ \frac{\partial}{\partial t} \int_{V_i} \rho dV + \int_{A_i} \rho (\mathbf{f} \cdot \hat{n}) dA = 0. \]  

(4.1)

Here our space is divided up into volumes \( V_i \) with boundaries \( A_i \). Now remember that the first term is the change in the density of the volume and the second is the flux through the boundary.

We shall, again, be working in 1-dimensional phase-space and divide our space into a grid of square cells of width \( \Delta x = \Delta v \). This allows us to simplify the above expression...
CHAPTER 4. FINITE VOLUME METHOD

to:

\[
\frac{\partial}{\partial t} \int_{V_{i,j}} \rho dV + \int_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} (\rho (f \cdot \hat{n}))_x \bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} dx + \int_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} (\rho (f \cdot \hat{n}))_v \bigg|_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} dx = 0, \tag{4.2}
\]

where the \( \pm \frac{1}{2} \) subscript refers to the halfway point between \( i \) or \( j \) and \( i \pm 1 \) or \( j \pm 1 \), a.k.a. the boundary value. This equation is just the flux term expanded out into the four borders of the cell. With the simple pendulum velocity field, this looks like:

\[
\frac{\partial}{\partial t} \int_{V_{i,j}} \rho dV + \int_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} (\rho v) \bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} dv - \int_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} (\rho g_l \sin(x)) \bigg|_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} dx = 0. \tag{4.3}
\]

We shall now approximate \( f \) and \( \rho \) with constant values inside each cell. If the discretization is fine enough then this approximation is valid. Let \( \rho_{i,j}, x_i, \) and \( v_j \) be the average value of the \( (i,j) \)th cell for the respective variable. Now we can take the variables out of the integral and equation \( 4.3 \) becomes:

\[
\frac{\partial}{\partial t} \rho_{i,j} \int_{V_{i,j}} dV + (\rho_{i,j} v_j) \bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} dv - (\rho_{i,j} g_l \sin(x_i)) \bigg|_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} dx = 0 \tag{4.4}
\]

\[
\Rightarrow \Delta x \Delta v \frac{\partial}{\partial t} \rho_{i,j} + \Delta v (\rho_{i,j} v_j) \bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} - \Delta x (\rho_{i,j} g_l \sin(x_i)) \bigg|_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} = 0. \tag{4.5}
\]

A note here: due to the assumptions that we have made, we can equivalently work with the probability density, or we can work with the probability mass function (pmf). Our discrete pmf is defined by:

\[
P_{i,j} = \int_{V_{i,j}} \rho(x,v) dV = \Delta x \Delta v \rho_{i,j}, \tag{4.6}
\]

which is only differs from the discrete pdf by constant term. Substituting this into equation \( 4.4 \) produces:

\[
\frac{\partial}{\partial t} P_{i,j} + \frac{v_j P_{i,j}}{\Delta x} \bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} - \frac{g \sin(x_i) P_{i,j}}{\Delta v} \bigg|_{v_{j-\frac{1}{2}}}^{v_{j+\frac{1}{2}}} = 0. \tag{4.7}
\]

We shall be working with the pmf from this point on for a reason that will become clear in the next section.
An important note: we have not specified $\rho$ or $f$ on the boundaries between cells, so there is some ambiguity in this expression. We shall resolve this by using something called the method of up-streaming or the unwinding scheme.

Up-streaming is where we take the value for $P$ at the boundary from the cell that has a flux going through it.

The long version is that the upwinding scheme is a method that is used in solving hyperbolic partial differential equations (such as the continuity equation) which makes use of the direction of flow in our system to simplify our discretization [24].

If we were evolving our system forward an infinitesimal step, then taking the average of the two cells would be a very good approximation. However, we are stepping forward discrete steps in time. During this time the probability distribution would have moved. Every boundary will only have probability flowing through it from one cell. Therefore, that one cell’s probability will be closer to the true value of the probability flowing through the boundary over that time in the original, continuous formulation of the problem. So we will choose the probability at each boundary to be the same as the probability of the cell that flows into said boundary. Mathematically this is:

$$\left. P \right|_{x_i^{1/2}} = P_{ij} \quad \text{if } f \cdot n \geq 0, \quad (4.8)$$

$$\left. P \right|_{x_i^{1/2}} = P_{i+1j} \quad \text{if } f \cdot n < 0, \quad (4.9)$$

and similarly for $v_{j+1/2}$.

Finally, to solve the coupled system of ODEs given in equation 4.7 we use the explicit Euler method, where we take:

$$\frac{\partial}{\partial t} P_{ij} \approx \frac{P_{ij}^{k+1} - P_{ij}^k}{\Delta t}, \quad (4.10)$$
for time step $k$. Now, taking into account that $\Delta v = \Delta x$, this changes equation 4.7 into:

$$P_{i,j}^{k+1} = P_{i,j}^k - \frac{\Delta t}{\Delta x} \left( v_j \left( P_{i,j}^k \big|_{x_{i+\frac{1}{2}}} - P_{i,j}^k \big|_{x_{i-\frac{1}{2}}} \right) - \frac{g \sin(x_i)}{l} \left( P_{i,j}^k \big|_{v_{j+\frac{1}{2}}} - P_{i,j}^k \big|_{v_{j-\frac{1}{2}}} \right) \right). \quad (4.11)$$

We can construct the vector $P^k$ out of all of the $P_{i,j}^k$’s. Since the discretization and velocity field are not changing in time, we can write this as the matrix equation:

$$P^{k+1} = (I - \Delta t A) P^k, \quad (4.12)$$

where $I$ is the identity and $A$ is a matrix that contains the up-streaming information (equations 4.8 and 4.9), plus the associated velocities for each cell.

Equation A.7 shows that this solver essentially breaks down to a simple matrix multiplication for each time step. This can be easily handled by any linear algebra package and is relatively easy to code up.

### 4.1.1 Further Restrictions

Now that we have a PDE solver, we need to decide the size of our discretizations and what our global boundary conditions are. What we want the solver to do is preserve:

1. the normalization,
2. and positivity of our distribution,
3. plus the dynamics of the system.

We will create our grid with $n \times n$ cells across fixed spatial range. This value, $n$, will be referred to from here on as the “fineness” of the grid, the higher $n$, the finer the grid. Since we are working on a pendulum, the system repeats every $2\pi$ in $x$. Therefore it makes sense to have the range of the grid be $[-\pi, \pi]$, with $x = 0$ be vertically down, as in Figure 2.3. The angular velocity, $v$ will be divided up in the exact same manner.

Finite volume solvers evolve the state of a system by looking at the flux through the boundaries of each cell. The way that they are set up means that the flux “seen” through a boundary by one cell will be the negative of the flux seen by the cell on the other side of the boundary. Because these fluxes are equal and opposite, there is no change in the normalization in the system. All that we need to do is make sure that there is no flux out of the region of space of interest ($x,v \in [-\pi, \pi]$).
CHAPTER 4. FINITE VOLUME METHOD

For us this means that we must enforce no flux through the boundaries at \( v = \pm \pi \). For the \( x \) boundaries we shall have the boundaries at \( x = \pm \pi \) connect to the corresponding ones at \( x = \mp \pi \), so that they loop around like a angles should. With this, no probability can leave or enter the system and hence, the normalization will be preserved. This is shown in Figure 4.2.

Preserving the positivity is more difficult as there is no implicit conservation of it in our solver like for the normalization. To preserve the positivity we must limit our time step size \( \Delta t \), in some way. We find a restriction as follows.

Say that we have a one dimensional cell of width \( \Delta x \) with probability \( P \) contained within. In our discretization of this problem only neighbouring cells can communicate. If our time step is so long that the probability would move far enough to skip a cell, then this probability will be lost and to negative values would arise. This leads to an upper limit on \( \Delta t \) set by the velocity field \( f \):

\[
\Delta t \leq \frac{\Delta x}{f}
\]

(4.13)

For our two dimensional case there are four borders per cell. At most only two borders around any cell will have an outwards pointing velocity. Therefore it makes sense to limit \( \Delta t \) by twice times the maximum outwards velocity across our grid as follows:

\[
\Delta t \leq \frac{\Delta x}{2 \max(f \cdot \hat{n})},
\]

(4.14)

where max refers to the maximum value across all boundaries in our grid. This works with \( \Delta x \) because again \( \Delta x = \Delta v \) for our set up.

This is a well established condition on advective PDE solvers called the Courant-Friedrichs-Lewy (CFL) condition [25]. This condition says that we should choose a
number relating the spatial and time separation such that, in two dimensions:

$$C = \frac{f_x \Delta t}{\Delta x} + \frac{f_v \Delta t}{\Delta v} \leq C_{\text{max}}$$  \hspace{1cm} (4.15)$$

where $C$ is known as the Courant number and $C_{\text{max}}$ is often chosen to be 1 for explicit solvers, such as ours.

Due to our assumptions we can rephrase this as:

$$C = \frac{f_x \Delta t}{\Delta x} + \frac{f_v \Delta t}{\Delta v} \leq \frac{2 \max(f \cdot \hat{n}) \Delta t}{\Delta x} \leq C_{\text{max}}$$  \hspace{1cm} (4.16)$$

So if we keep

$$\frac{2 \max(f \cdot \hat{n}) \Delta t}{\Delta x} < 1$$ \hspace{1cm} (4.17)$$

then we will meet this condition. This may be rearranged to:

$$\Delta t < \frac{\Delta x}{2 \max(f \cdot \hat{n})},$$ \hspace{1cm} (4.18)$$

as above.

This is a limit, so ideally we should keep our time step much lower than this. For the purposes of this thesis we shall be using:

$$\Delta t = \frac{\Delta x}{10 \max(f \cdot \hat{n})},$$ \hspace{1cm} (4.19)$$

to ensure that our distribution does not go negative.

All that remains now is preserving properties of the dynamic system. This is one thing we cannot do perfectly. Although, the finer our discretization, the more accurate the dynamics and behaviour of our system is, provided the PDE solver is consistent and stable.

### 4.2 Consistency

It is known that first-order up-steaming finite volume methods such as the one described in the last section are consistent in advection-diffusion equations [24]. The continuity equation is purely an advection equation and therefore our system is a special case of this. However we shall show numerically that this PDE solver converges to the true solution as we increase the discretization fineness, which decreases the time and spatial
discretization step size.

4.2.1 Kullback-Leibler Divergence

For a particular run of this solver, we are going to need a way to tell how accurate our result is. In other words; how close our simulated solution is to the true solution. Hence, we can tell whether the simulation is accurate or not. For this we are going to be using the Kullback-Leibler divergence. I will not derive, nor explain the subtleties here, as they are done in great detail in [26]. It is a way of comparing how different one probability distribution is from another.

The divergence of the discrete probability distribution \( Q \) from \( P \) is:

\[
D_{KL}(P || Q) = \sum_i \log \left( \frac{P(i)}{Q(i)} \right) P(i).
\] (4.20)

Note that this is not symmetric. Again, this is the divergence of \( Q \) from \( P \) and not the other way around. Equation 4.20 only works for cases where \( Q(i) = 0 \implies P(i) = 0 \) as \( \lim_{x \to 0} x \log(x) = 0 \), so infinities are avoided. Also note that both probabilities must have the same discretization.

If \( Q \) in this case is our current numerical solution, what then is \( P \)? In cases where there exists an analytical solution, we take a discretized version of that such that it matches the numerical solution. But this presents an obvious problem; in most practical cases we will not have an analytical solution.

For these cases we will use the fact that as our discretization becomes finer and finer, the simulated solution converges to the true solution. If we take our “true” solution, \( P \), to be another simulated solution, only this one run with a finer grid for the same time, then we should get an estimate of how close we are to the actual true solution.

For the purposes of this thesis we shall be comparing a simulated solution with one where each square grid point is divided into 16 smaller ones, i.e. comparing a simulation of fineness \( n \) with one of fineness \( 4n \).

We also need the \( 4n \) case to have the same discretization as the \( n \) case. So we will match the cells for the \( n \) case to those that would be contained in it for the \( 4n \) case, then sum before calculating the divergence. So we calculate the Kullback-Leibler divergence
by:

\[ D_{KL}(P_n || P_{4n}) = \sum_{i=1}^{n^2} \log \left( \frac{P(i)}{\sum_{j=1}^{16} P_{4n}(j)} \right) P(i). \] (4.21)

### 4.2.2 Numerical Tests

For our numerical analysis we need to make sure that our PDE solver can evolve a probability distribution accurately enough that is represents the prior for each Bayesian observation we make accurately. We shall do this in a few parts.

Firstly, since there is an analytical solution to evolving a Gaussian on a SHO, we shall start there. We will increase \( n \) and observe whether or not the numerical solution converges to the true solution for an initial Gaussian distribution. If it does, then we will progress to do the same for the full, non-linear pendulum, exploring the full range

![Figure 4.3: Simulations with various grid resolutions compared with the analytical solution after one period.](image)

...
of non-linearities with Gaussian initial distributions again.

For both of these cases we shall evolve the system for $2\pi$ seconds, which is a full period for a SHO. This will then be compared against the analytical solution or the $4n$ solution. As we are planning on having several observations per period in our inference method, a reasonable level of convergence at $t = 2\pi$ will mean that there will be a high level of convergence over the time scales for which we are concerned.

Some simulated solutions with the SHO dynamical system are displayed alongside the analytical solution in Figure 4.3. The distributions after one period are clearly tending towards the analytical solution as $n$ is increased. A good assumption in regards to rate of convergence in numerical methods is that for large enough $n$, the method should converge like a power law [16]. A plot of the log of the divergences vs. grid size is shown on figure 4.4. We can see here that the values for $n$ are not large enough for this approximation to be valid, however it is decreasing at a super-linear rate.

An important note about our finite volume solver is that the simulations are not meant to run for the full period without an observation step. What we need is for the solver to propagate the probability accurately enough that the prior we use in a Bayesian observation step is essentially the correct prior. This means that we just need our solver to be accurate over the time scales separating observations. For reference, the maximum
Figure 4.5: Simulations with grid resolution $n = 50$ (top) compared with simulations with grid resolution $n = 200$ (bottom) after one period. The first column is initially at $x_0 = 0.2\pi$. The second is at $x_0 = 0.5\pi$. Both of these have initial $\sigma = 0.2$.

Some simulated solutions with the full pendulum dynamical system are shown in Figure 4.5 next to their respective $4n$ simulations, as discussed earlier. Note here the obviously non-Gaussian distribution at the higher initial angle. This also has a higher $D_{KL}$, as one might expect from a system with stronger non-linearities.

Repeating the previous process for different initial positions on the pendulum produces Figure 4.6. As expected the lower initial positions converge much like in the SHO, only with lower rates of convergence due to the fact that both the $n$ and the $4n$ simulation are increasing in resolution together. The convergence rate decreases the higher the initial angle is, again, as to be expected. However the when starting it at $x_0 = 2\pi$
we find the the divergence increases as we increase the grid size. This is due to two things: how the tests were performed, and the fact that we are not in the asymptotic region of numerical convergence yet. The ×4 method assumes that we are in, or close to the asymptotically convergence region and the high non-linearity at that point leads to a larger difference in the results.

Figure 4.7 is instead the Kullback-Leibler divergence calculated for different grid finenesses against a run with the high grid fineness of $n = 3200$. We see that there is a negative slope in the first few points showing that there it will indeed converge. The latter points are less important as there we are getting close to $n = 3200$ and will converge anyway; the final point is at $n = 1600$, which is only half that value.

Again it is important to point out that the simulations will not be run for this long without observation steps. The divergences are naturally much smaller the shorter the run time of the PDE.
4.3 Inference

Now that we have shown that the PDE solver is consistent and works over significant periods of time, we now can add an observation step. As discussed in section 2.1.2, we do this by applying Bayes Theorem:

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} , \tag{4.22} \]

where the probability distribution provided by the finite volume solver will be used as the prior, \( P(A) \).

We will assume our measurement has Gaussian noise with a known noise again as in section 2.2. The value of the measurement will function as the mean which, when combined with covariance \( E_z \), will form a Gaussian distribution that will function as \( P(B|A) \) in equation 4.22. The term, \( P(B) \), is just a normalization term and can be disregarded for the purposes of parameter estimation[12].

For the purposes of this thesis, the times between each observation shall be constant.

*Reminder*: A summary of the method can be found in appendix A.1.
Chapter 5

Application

In this chapter we shall explore the behaviour of the finite volume inference method (appendix A.1) on the pendulum. For comparison we shall be comparing this to the UKF algorithm as described in Section 2.2.2.

5.1 Behaviour Of Method

Firstly we shall explore how the finite volume method and UKF behave. We shall use the initial conditions of $x_0 = 0.2\pi$, $v_0 = 0$ with $\sigma = 0.2$. Some fake data is constructed by evolving the same initial position with an RK45 solver and selecting points at uniform spacing in time, adding Gaussian noise ($\sigma = 0.1$). These are the observations that we make on our system.

These shall represented by the mean and standard deviation of the distribution in time. An example of this is Figure 5.1 which is the finite volume method without any observations. We can see from this that the distribution expands as time goes on, but the mean follows the true solution very well.
CHAPTER 5. APPLICATION

Figure 5.1: The finite volume method at $n = 100$ evolved for $2\pi$ seconds with initial conditions $(x_0, v_0) = (0.2\pi, 0)$, $\sigma = 0.2$. The dashed black line is the “true” solution. The solid blue line is the mean of the evolved distribution. The green dotted line is one deviation from the blue.

Figure 5.2: The finite volume method at $n = 400$ otherwise the same as 5.1.
Figure 5.3: The finite volume method at $n = 200$ with 5 observations (cyan pluses).

Figure 5.4: The finite volume method at $n = 400$ with 5 observations.
Obviously, this will be more precise the finer the grid, as demonstrated by Figure 5.2 where the deviation is more consistent. With some observations this gives 5.4 for \( n = 100 \) and \( n = 400 \) respectively. We can see here that the predicted state is changed with every observation. Then this new state is evolved. This works very well with only five observations, but for comparison’s sake Figure 5.5 is the same case, but with 30 observations.

Now, the UKF as presented in this thesis can only step forward in time when there is an observation step. So we will be running this at 30 observations per period at a minimum. The pendulum has no dynamic noise, so it would make sense to represent \( F \) in equation 2.36 as precisely as possible, i.e. without and dynamic noise. However this can lead to the filter providing incorrect results as demonstrated in Figure 5.6. This even occurs with more observations. Because of this we shall run the UKF in this thesis under the false assumption that there is Gaussian dynamic noise of \( E_x = 0.01 \). This fudge factor prevents the UKF from getting stuck and means that it will stay near the “true” value indefinitely for the linear case. This is shown in Figure 5.7.

![Figure 5.5: The finite volume method at \( n = 200 \) with 30 observations.](image-url)
Figure 5.6: The UKF with 30 observations without dynamic noise. The predicted path and bounds differ from the true path a significant amount.

Figure 5.7: The UKF with 30 observations and dynamic noise $E_x = 0.01$. 
One important note is that the UKF tends to underestimate the covariance. In Figure 5.6 we can see that the estimate’s covariance is almost zero, despite the true path and observations lying far from the estimated mean. This happens to a lesser extent when dynamic noise is introduced. But we can see from this that the finite volume method places more importance on the observations.

5.2 Simple Pendulum

Now we shall compare the finite volume method with the UKF for different initial angles. Our initial conditions shall be a Gaussian as described in the previous section. We will have 30 observations that will be Gaussian distributed, $\sigma = 0.1$. Finite volume method will be run at $n = 200$.

Figures 5.8 and 5.9 are in the highly linear regime with the initial condition of $x_0 = 0.2\pi$. Both filters function essentially the same. However our finite volume method has the advantage that one can see the full probability distribution at every step; Figure 5.8 is only a representation of the second order statistics in the system. In contrast, that is all that one can get from the UKF. Just to name one advantage of this: it is much easier to tell if the filtering method is providing a false result. For example, if we had naively left out our assumed dynamic noise in setting up the UKF then the filter would have produced false results like in Figure 5.7. In highly non-linear systems it can be very difficult to check if the output is accurate or not.

Figures 5.10 and 5.11 are outside the small angle approximation at $x_0 = 0.4\pi$. We can see that the motion is no longer a perfect sinusoid. However, there is very little difference in behaviour from the linear regime. We again see that the finite volume method places more emphasis in the observations.

Figures 5.12 and 5.13, plus 5.14 and 5.15 are well outside the linear regime at $x_0 = 0.6\pi$ and $0.8\pi$ respectively. Once again both methods work very well. However in the first sub-plot of 5.15 if one looks closely one can see that the true path is just outside the one standard deviation bounds for a significant period of time. In contrast our method can be misled by several noisy measurements that land close to one another, but it can then readjust once more measurements are made.
Figure 5.8: The finite volume method with 30 observations at $x_0 = 0.2\pi$.

Figure 5.9: The UKF with 30 observations at $x_0 = 0.2\pi$. 
CHAPTER 5. APPLICATION

Figure 5.10: The finite volume method with 30 observations at $x_0 = 0.4\pi$.

Figure 5.11: The UKF with 30 observations at $x_0 = 0.4\pi$. 
Figure 5.12: The finite volume method with 30 observations at $x_0 = 0.6\pi$.

Figure 5.13: The UKF with 30 observations at $x_0 = 0.6\pi$. 
Figure 5.14: The finite volume method with 30 observations at $x_0 = 0.8\pi$.

Figure 5.15: The UKF with 30 observations at $x_0 = 0.8\pi$. 
Figure 5.16: The finite volume method with 30 observations at $x_0 = \pi$. Note: the spikes in the covariance are due to the fact that we are on the boundary and are nothing more than an artefact from how they were computed. They shouldn’t actually exist.

Figure 5.17: The UKF with 30 observations at $x_0 = \pi$. 
We have seen the filter’s both work very well in the practical cases one would encounter on a pendulum. The main differences are the lower estimate of the covariance in the UKF and the greater influence that observations have on the finite volume method.

Figures 5.16 and 5.17 are at the limit of the non-linearities on a pendulum: the unstable equilibrium at $x_0 = \pi$ and $v_0 = 0$. While this is a somewhat contrived circumstance, as a real pendulum will never stay on this point, in the majority of practical cases the model will only be an approximation of the true, complex system. For example, in the navigation of airborne vessels one can assume linear dynamics with some noise and return accurate results. When the air is calm, the small amount of turbulence is accounted for in the dynamic noise. However when the air is unusually turbulent - such as in stormy weather - this will not work nearly as well. So these simulations test how the filters deal with observations contradicting our assumed model.

We can see that this is where the different methods diverge. The UKF (Figure 5.17) returns a very noisy estimate with the true solution lying well outside the one sigma bounds for large periods of time. In contrast the finite volume method’s mean is relatively consistent with the true solution straddling the one sigma line (Figure 5.16). From this one might conclude that neither method has a significant advantage over the other, however I shall remind the reader again that this is only a second order representation of the actual output. What we are provided is shown in Figure 5.18.

When looking at Figures 5.16 and 5.17 it is not clear what exactly is happening, the full output of our method shows that there is a large spike in probability in the $x_0 = \pi$ and $v_0 = 0$ cell that does not decrease.

![Figure 5.18: Output after the first and the 10th observations of the finite volume method with $x_0 = \pi$.]
Figure 5.19: The finite volume method with 2 observations at $x_0 = 0.8\pi$, $n = 200$.

Figure 5.20: The finite volume method with 2 observations at $x_0 = 0.8\pi$, $n = 800$. 
Another advantage of this method is the fact that it can run very well with only a few observations. Figure 5.19 is of the finite volume method with only two observations. We can see that it still works remarkably well and importantly, provides us with uncertainties at any point in time that we wish. The accuracy of this can be increased merely by devoting more computational resources to it by way or increased grid fineness (Figure 5.20).

Note about viewing: As described earlier, this second order representation of the probability distribution in time can be misleading, especially when the distributions are like that in the right hand plots of Figure 4.3. It is, however, very simple to create an animation of the probability distribution directly. This is how I view them, alongside plots of relevant expectation values. But since this is made for print, these animations obviously could not be produced here and we have to settle for the second order representations displayed previously. These representations do, advantageously, allow us an easier comparison to the UKF.

5.3 Ambiguous Measurement and Multi-Modality

In the previous sections our observations were directly on the pendulum itself, however often the situation is more complex than this. The Kalman filter from Section 2.2.1 is designed to cope with noisy linear maps from the state to the observable quantity. The UKF extends this and allows for any noisy bijective maps from observation to state. However neither have any mechanism to deal with surjective, but not injective maps such as squares or sinusoidal functions.

In stark contrast the finite volume method has no such restrictions. It can deal with arbitrary noise, and distributions. It is not a conceptual stretch for it to be applied in hetsenesedastic situations, which are traditionally difficult to work with. For demonstration purposes, we shall consider our observations to be the tension force on the pendulum’s string as in Figure 5.21.

The force in the $r$ direction is on a pendulum is the combination of the force allowing
for circular motion and the component to gravity not in the \( \theta \) direction. This is:

\[
F_r = mv_0^2 + mg \cos(\theta) \hspace{1cm} (5.1)
\]

\[
= \frac{2mg}{l} \left( \cos(\theta) - \cos(\theta_0) \right) + mg \cos(\theta)
\]

\[
= \left( \frac{2}{l} + 1 \right) mg \cos(\theta) - \frac{2mg}{l} \cos(\theta_0) \hspace{1cm} (5.2)
\]

where \( \theta_0 \) is the maximum angle. This equation is symmetrical in \( \theta \) and therefore, a single measurement of \( F_r \) could be from two equally likely angles.

We shall be building some fake data by solving the motion of the pendulum as before, then measuring \( x \) only with some additive Gaussian noise. While not exactly the above case, this is sufficient for the purposes of demonstrating how our solver can work with multi-modality.

5.3.1 Simulations

Here we shall show a couple of example simulations with this ambiguous force measurement. Firstly there is Figure 5.22 which is the probability distributions at times \( t = 0 \), \( \pi/3 \), \( 2\pi/3 \), and \( \pi \). The initial conditions were \( x_0 = v_0 = 0 \), \( \sigma = 0.7 \) and observations in \( x \) with \( \sigma = 0.1 \).

We can see here that this leads to, on average, 2 modes on the opposite sides of the origin. If the observations are timed right one can even have 4 modes at once. As there were no assumptions made on the probability distribution, the filter doesn’t find this any more troublesome than a single mode.

To provide some idea as to why this is useful, we have a plot similar to those in the previous section. Figure 5.23 is the second order statistics of a force measurement simulation with 20 observations. By looking at just the mean and covariance, we get the impression that the distribution is a non-isotropic Gaussian centered at the origin slowly rotating. However the true path is obviously sinusoidal. We have overlayed this plot with the output of a very simple local maxima detection algorithm to give an indication of where the modes are at every point in time. Because of the algorithm’s simplicity, it can detect some “ghost” modes. But nonetheless we can see that there are two paths of \( \times \)'s following the true path, and the reflection of the true path about 0. However the actual output, again, is more like Figure 5.22.

The UKF and many other filtering methods cannot deal with a simple multi-modal system such as this. Therein lies one of the greatest strengths of this new finite volume method: it makes very few assumptions and therefore has very few restrictions.
Figure 5.22: Snippets every $\pi/3$ seconds of the finite volume method with 5 force observations at $x_0 = 0.2\pi$, $n = 200$. 

Phase-Space Probability
Figure 5.23: The finite volume method with 20 force observations at $x_0 = 0.2\pi$, $n = 200$. The black dotted-dashed line is the true path. The blue solid line is the mean with the green dotted lines one sigma away from this. Finally the red crosses are where a very basic detection algorithm places the local maxima, giving a rough idea of the center of our modes.
Chapter 6

Conclusion

In this thesis we took the idea of sequential inference back to it’s core. Most filtering methods take an approximation of the probability distribution, then evolve this as though it were a particle or through some other method. We evolve the probability itself directly through the continuity equation.

This was compared against a commonly used and efficient filter: the unscented Kalman filter. We found that for the two dimensional case both filters tested functioned very well. The UKF was more efficient and faster, but our finite volume method provided the full information of the probability distribution. The finite volume method also has the advantage that the accuracy can by increased merely by increasing the grid fineness and hence, the computational resources. More than that, by evolving the system explicitly, the accuracy is limited only by the accuracy of the PDE solver used and can therefore be arbitrarily accurate.

Another significant advantage our finite volume method has is that it has no limitations on the shape of the probability distribution itself. We saw in Section 5.3 that multi-modal systems and ambiguous observations were handled with equal ease. Multi-modal distributions occur in many areas of science, ranging from material sciences [27] to cancer detection [28]. Multi-modal distributions and ambiguous measurements are things that traditionally has been very difficult to do inference on; the UKF has no capacity for this at all.

Inference is a very powerful tool that is limited only by our creativity and computational power. We now have computers powerful enough to evolve and update probability distributions through observation explicitly. While the finite volume method presented
here scales poorly to high dimensional systems, this method was never meant to be the final product, just a proof of concept.

There has been many recent advances in efficient PDE solving methods. These include, but are not limited to, adaptive grid methods [29, 30], computationally efficient methods [24, 25], and more accurate solvers. These could be adapted to this problem with relative ease to fully realize this inference method’s potential.
References


Appendices

A.1 Summary Of Method

We are going to be performing inference on a system by evolving the probability in phase-space directly through solving the continuity equation:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{f})$$  (A.1)

for some quantity $\rho$ on a velocity field $\mathbf{f}$ which describes a dynamical system.

We will be using a finite volume method to solve this with the following details and approximations:

- Our space will be divided up into a square grid of cells of width $\Delta x$ ranging from $-\pi$ to $\pi$.
- This discretization must be fine enough that the values of $\rho$ and $\mathbf{f}$ are approximately constant in a cell and across a boundary, respectively.
- We will use the method of up-streaming to calculate the flux at the boundary between cell $(i, j)$ and $(i + 1, j)$. That is:

$$\rho \bigg|_{x_{i+1/2}} = \frac{P_{i,j}}{\Delta x^2} \quad \text{if} \quad \mathbf{f} \cdot \mathbf{n} \geq 0,$$  (A.2)

$$\rho \bigg|_{x_{i+1/2}} = \frac{P_{i+1,j}}{\Delta x^2} \quad \text{if} \quad \mathbf{f} \cdot \mathbf{n} < 0$$  (A.3)

where $P_{i,j}$ is the total probability in cell $(i, j)$, and the notation $x_{i+1/2}$ refers to the boundary between cell $(i, j)$ and $(i + 1, j)$ with associated outward pointing normal vector $\mathbf{n}$. 
We will be using a simple Euler step for our time partial differential:

$$\frac{\partial}{\partial t} P_{ij} \approx \frac{P_{ij}^{k+1} - P_{ij}^k}{\Delta t}$$  \hspace{1cm} (A.4)$$

with time step $\Delta t$.

The time step size is going to be limited by:

$$\Delta t \leq \frac{\Delta x}{2N \max(\|f \cdot n\|)}$$ \hspace{1cm} (A.5)$$

in order to preserve positivity, where $n$ is the outward pointing normal vector of a cell and $N$ is the number of neighbouring cells to a single one, 4 in our case.

This all leads to the equation for each cell to get from time $k$ to $k+1$:

$$P_{ij}^{k+1} = P_{ij}^k - \Delta t \left( F_{v \leftarrow 1/2} + F_{v \rightarrow 1/2} + F_{x \leftarrow 1/2} + F_{x \rightarrow 1/2} \right)$$ \hspace{1cm} (A.6)$$

where the $F$'s refer to the outward fluxes through each boundary and the subscripts $x$ and $v$ refer to the space and velocity directions respectively.

If we create a vector that contains all $P_{ij}$'s, this can then be summed up into the matrix equation:

$$P^{k+1} = (I - \Delta t A) P^k,$$ \hspace{1cm} (A.7)$$

where $I$ is the identity and $A$ is a matrix that contains the up-streaming information plus the associated velocities for each cell.

We can now use Bayes’ Theorem 2.1\[^{2,1}\] to update our probability from an observation through direct vector multiplication:

$$P_{\text{new}} \propto P_{\text{observation}} \cdot P_{\text{old}}$$ \hspace{1cm} (A.8)$$

These two sides just differ by a normalization term.