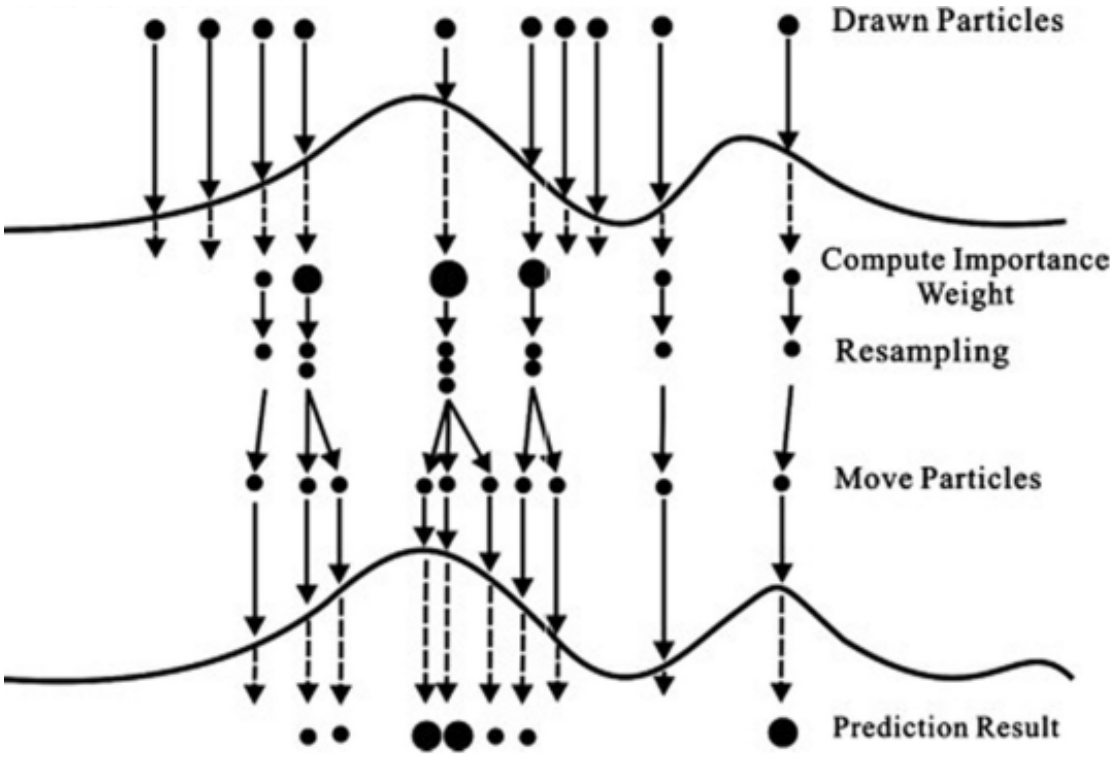
CHALMERS
UNIVERSITY OF TECHNOLOGY



Latent State Estimation for Financial Time Series

Estimating Financial Health with MCMC Methods and Particle Filters

Master's thesis in Engineering Mathematics and Computational Science

Erik Hermansson

Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Latent State Estimation for Financial Time Series

Estimating Financial Health with MCMC methods and Particle Filters

Erik Hermansson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2020

Latent State Estimation for Financial Time Series
Estimating Financial Health with MCMC methods and Particle Filters
Erik Hermansson

Supervisor: Moritz Schauer, Mathematical Sciences
Examiner: Umberto Picchini, Mathematical Sciences

Master's Thesis 2020
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of a Bootstrap Particle Filter.

Gothenburg, Sweden 2020

Abstract

Financial Modelling allows for prudent decision making for individual business owners and other stakeholders. The Financial Health can be seen as an underlying measure which governs the companies ability to meet its obligations and make profits. Therefore *Financial Health* is linked to the company's cash flow which can readily be observed.
We consider the *Financial Health* as a dynamic latent state and infer it from the cash flow. We are estimating this latent state under the Bayesian paradigm to take stylized properties of the cash flow into account, using a Particle Filter as part of a Monte Carlo method to sample the posterior distribution of latent state and model parameters.
We investigate the performance of this approach on a real data set consisting of real cash flow from small Swedish businesses.

Keywords: Bayesian Inference, Hidden Markov Models, Particle Filter, Financial Health

Contents

List of Figures

List of Tables

1
Introduction

Financial modelling is an important topic for the planning of company activities since this helps businesses with their financial planning and risk management. However, financial time series is hard to model since the data often show signs of heteroscedasticity, i.e. non-constant volatility, there is often a significant partial auto-correlation and the underlying model is often complicated. Financial data can contain paradigm shifts which can be viewed as changes of the latent state. These paradigm shift might be caused by several different real world events that might be hard to identify. By considering the paradigm as a latent, or hidden, state one can infer when there has been a structural change of the financial time series.
The aim of this thesis is to investigate the possibility to model the financial health of a company through its cash flow. Cash flow and financial health should however not be conflated since cash flow is not by itself a measure of a company's well being. The aim is to come up with a method which estimates the financial health as a latent variable from the observed cash flow. The method will try to capture the stylized properties of the cash flow.
The thesis will be using the Bayesian framework since this gives a natural way of working with missing data and the latent nature of financial health. By adopting a Bayesian perspective of the data modelling one can avoid many issues that is connected to overfitting. In [2] Bishop writes that overfitting is largely connected to Maximum Likelihood Estimation (MLE) and not present in the same way in the Bayesian paradigm. This is because no MLE is needed in Bayesian statistics and that the number of effective parameters adapts to the data set, which removes the issue of overfitting to some degree, according to Bishop. The reason for this is that in Bayesian statistics one does not optimize but instead marginalize over all possible choices. Due to this reason the Bayesian Paradigm is a good choice for this kind of data series.
The thesis will further use methods which can handle non-normal distributions and nonlinear observations which is a result by the hidden state. This is different from many models used today which assumes conditional Gaussanity and linearity that limits the applicability and accuracy of the models. Some methods which will be explored are Hamiltonian Monte Carlo, Variational Inference and Particle Filters, these methods are described in both theory and practice.
The dataset which is being used is real cash flows of companies with similar company and tax structure with no accounts Payable or Accounts Receivables. The anonymized data has been delivered by Company X and will not be disclosed due to secrecy reasons.
The thesis begins with a discussion regarding related work and followed by data

exploration which seeks to find structural patterns in the data set. Then follows an attempt to model the cash flow with a hierarchical Bayesian model. Lastly the model is utilized in a Hidden Markov Model, where a Bootstrap Particle Filter, which allows for non-linear and non-Gaussian emissions, is implemented. The output of this model will yield a hidden state which can be interpreted as measure of financial health and can be used to gauge the well-being of said company.

2
Background

This chapter will introduce some of related work that currently exist that is related to the topics that will covered in this thesis. The related work covers Bayesian modelling, Compound Poisson Processes, Markov Chain Monte Carlo Methods and Particle Filters. No individual article have the same aim as this thesis but all of them cover some part which is important to the thesis.

2.1 Related Work

Bayesian Modelling allows for uncertainty estimation of model parameters which are time dependent. Time dependency of variance in financial market have long been a known issue and with Bayesian modelling one can construct Stochastic Volatility Models that shows heteroscedasticity, i.e. non-constant volatility. This has been done by Gugushvili et al in [3] where the authors modeled the volatility of the exchange rate EUR/USD under micro-structure noise. By the use of a Gibbs sampler and the Forward Backward algorithm the authors were able to achieve good results. The authors built the model using Bayesian models which gives a natural interpretation of latent and missing data which will be adopted in this thesis.
Compound Poisson processes have Poisson counts and gamma distributed intensity where the two processes are independent from each other. This kind of process have been used to model the daily stochastic rain fall by Dzupire et al in [4]. The results showed that rainfall could be modeled with a relative simple model with only these two features. The implications from this is that the model could be used to estimate future rainfall which leads to easier risk assessment and pricing of weather related financial derivatives according to the authors. The authors used Maximum Likelihood Estimation which is a method often used in frequentist paradigm but that was not the scope of the thesis. However, the result shows that the model type, Gamma Poisson Process, is a very viable model to use and can be applied to real world problems.
The case of Non-Homogenuous Poisson Processes have been covered extensively by Gugushvili et al in [5] where the authors estimated the time dependent parameter for a Non-Homogeneous Poisson Process in a non-parametric way. A Non-Homogeneous Poisson Process is a stochastic process with a time dependent parameter, more details regarding the method can be found in the paper. The authors displayed the usability of the method on various data sets and the results showed that a time dependent parameters can often describe the data in a better way since many systems are dynamic and not static.

Particle filter have long been used in signal processing where the driving noise is non-gaussian. There have however been a large amounts of issues reported with these methods by Kantas et al in [6]. The paper showed how many Particle Filters have a degeneracy problem which means that few particles can have an excessive weight in the filter. This leads to high variance estimates that makes the model unreliable. It is also reported the the methods are very computationally expensive which might limit their application in the real world.

2.2 Final Remarks

The articles presented above shows that the methods which are intended to be used in this thesis have been used with success before. Many time series have the time dependent parameters which can be modeled with compound processes and non-homogeneous stochastic processes. There has been reported issues with the particle filter giving high variance estimates. This is an important aspect to keep in mind while working on the model and when evaluating it.

3
Data Analysis

The scope of this data analysis is to identify if there is any temporal structure to the data, such as seasonality, and with this information find a way to model the data in a Markovian way. From here on will $\pi()$ denote the density of a stochastic variable, capitalized variables are random variables and small letters are realizations of said variables, all random variables are real numbers. A stochastic process X_t, $t \in 0, 1, \ldots$ is Markovian if the condition in eq. (3.1) is fulfilled, where n is index of time above t .

$$\pi(x_{t+n} \mid x_1, ..., x_t) = \pi(x_{t+n} \mid x_t) \tag{3.1}$$

The meaning of this is that the probability of the next or an later state is only dependent on the current one. It is important that the data can be modeled with this expression since the intended model for this thesis, the Hidden Markov Model, requires the hidden state to be Markovian.
Certain aspects such as stochastic seasonality makes it hard to model the data in a Markovian way. To avoid this issue one can assume that there is only deterministic, i.e. fixed, seasonality.

3.1 Data Description

The data contains anonymized information regarding the main activities of the company, its initial liquidity, tax period and corporate structure. All the companies have have the same start and end of the fiscal year but the length of data series differs significantly from company to company. For the sake of simplicity, the main companies of consideration will be companies with the same tax period, corporate structure and main business activities. There are a lot of companies in the data set so there will only be a few on displayed in the paper. The results does however not differ much between companies.

3.2 Initial Data Exploration

Since one part of the project is centered around modelling cash flow, only the sums per day and amount of transactions per day will be analyzed in this chapter. The histogram of the sums of the transactions are shown in fig. 3.1 and the histogram of the transactions per day are shown in fig. 3.2, these plots are only for an individual company.

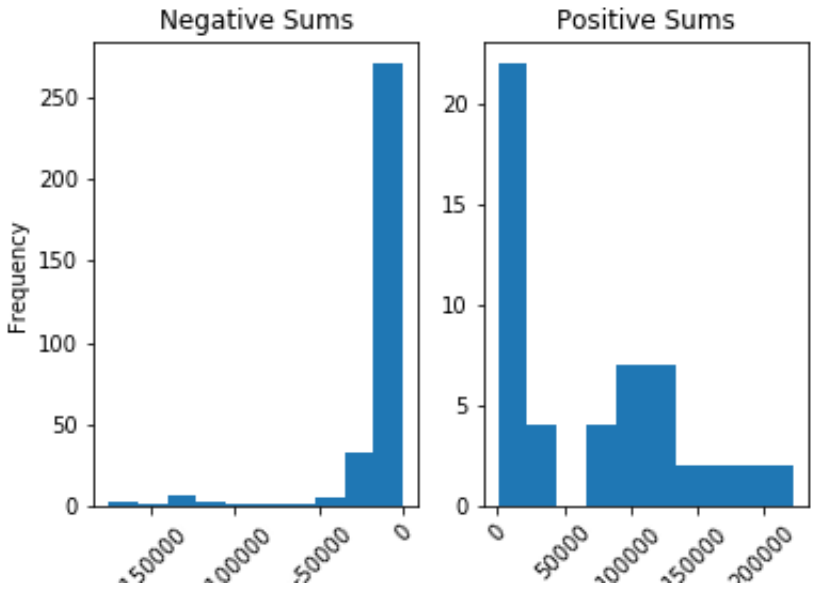


Figure 3.1: Histogram of positive and negative cash flow in SEK from individual transactions.

In fig. 3.1 one can see that the positive sums, which is defined as positive cash flow, and the negative sums, which is defined as negative cash flow, per day does not seem to come from the same distribution, instead the negative sums seems to have many small payments and few large ones. In contrast to this the positive sums seems to be bi-modal, i.e. that the sums comes from a mixture distribution.
What can be seen in fig. 3.2 is that the amount of daily transactions seems to decrease rapidly. Zero transactions per day is the most common amount of transactions and there are very few days with two or more transactions. With this information one can argue that the transactions per day is Poisson distributed.
What should be noted is that histograms does not capture the temporal structure, if there is one, but they do however show the empirical distribution. The histogram of the sums shows that the negative sums are power law distributed, there are many small amounts and few large amounts. The positive sums seems to be bimodal, there are many small amounts but also quite a few large ones with few in between.

3.3 Data Diagnostics

One of the main things of interest will be the auto correlation (ACF) and partial auto correlation (PACF). These are the most important temporal structures since they will dictate the model building.
The intuition of the ACF is that it measures the correlations of the different lags of the data. The PACF on the other hand measures the correlation of the lagged data and controls for intervening lags.The ACF is defined as

$$\phi_X(k) = Cor(X_{t+h}, X_t) \tag{3.2}$$

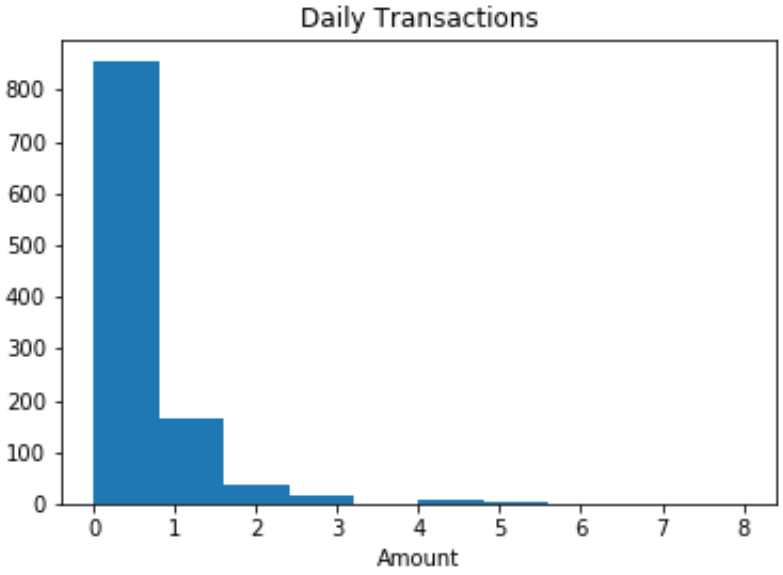


Figure 3.2: Histogram of transactions per day.

where h is the amount of lags and the PACF is defined as

$$\alpha(h) = \phi_{hh} \tag{3.3}$$

where ϕ_{hh} is the last component of

$$\phi_h = ((\gamma(i-j))_{i,j=1}^{h})^{-1}(\gamma(1), ..., \gamma(h)))'.$$

A more thorough explanation of these concepts can be found in [7]. The ACF and PACF only work for stationary time series which means that the mean and variance is constant over time, a more thorough definition is found in [7]. However, the ACF and PACF is still interesting in practice even if the process is not stationary.
In fig. 3.3 and fig. 3.4 shows the ACF and PACF of the transaction for the company is shown. A point can be seen as significant if it is above the 95% confidence interval. It looks like that there is a significant auto-correlation from month to month and there is also a strong PACF between several time instances. This is reasonable since companies have a monthly payroll and also pay taxes on a fixed time basis, yearly for example. These temporal structures will be important to incorporate into the model since there is a very robust structure.
In fig. 3.5 we see that the ACF for the cash flow does not seem to have any significant auto correlation and the PACF in fig. 3.6 seems to be significant for a lag of 30 days, 75 days and 160 days for the cash flow. This translates to that there seams to be an occurring transactions on a monthly basis. But one should be wary of using visual diagnostics in this way since it is vulnerable to the multiple comparison problem since during the project many companies were inspected. So a robust conclusion of the data diagnostics is that the transactions seems to have a temporal structure but that the sums does not seem to have any significant ACF or PACF.
In fig. 3.7 the average transactions per day if one partitions that data. For example, the average amount of transaction per day the first five days for the positive counts is

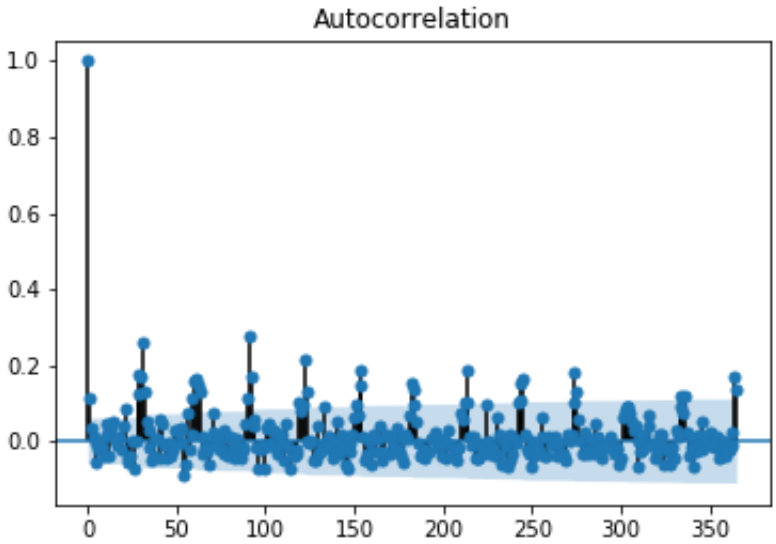


Figure 3.3: ACF of transactions with 95% confidence interval.

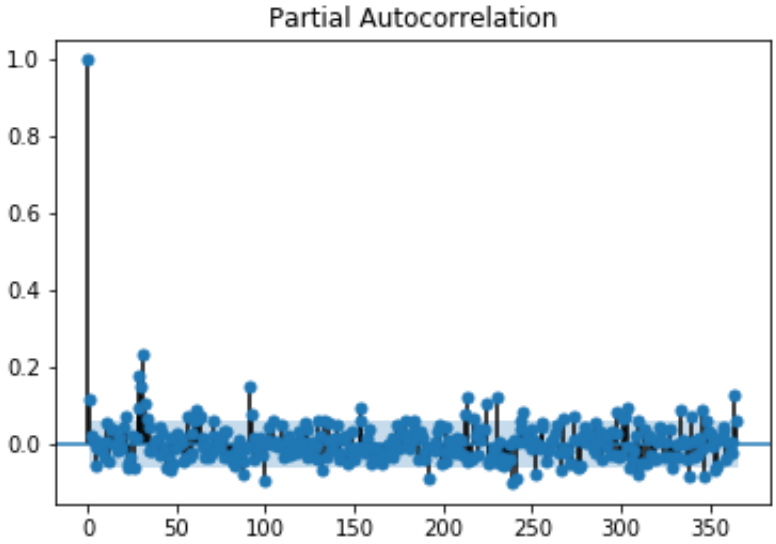


Figure 3.4: PACF of transactions per day with 95% confidence interval.

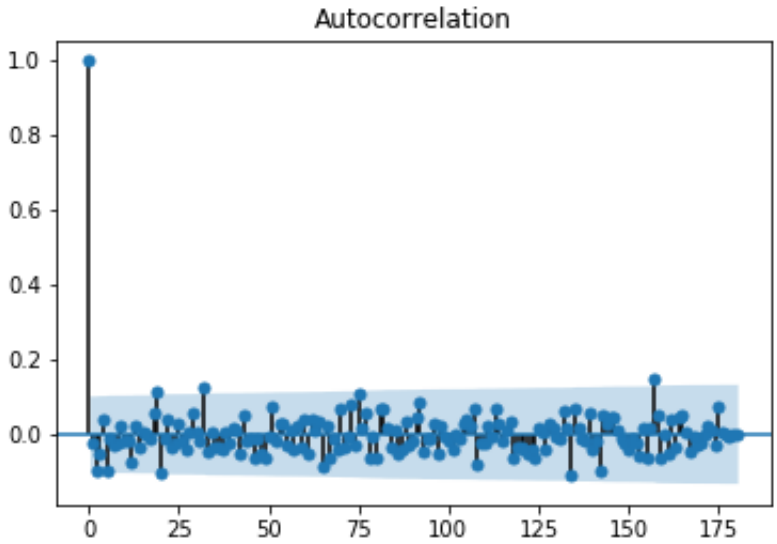


Figure 3.5: ACF of cash flow in SEK with 95% confidence interval.

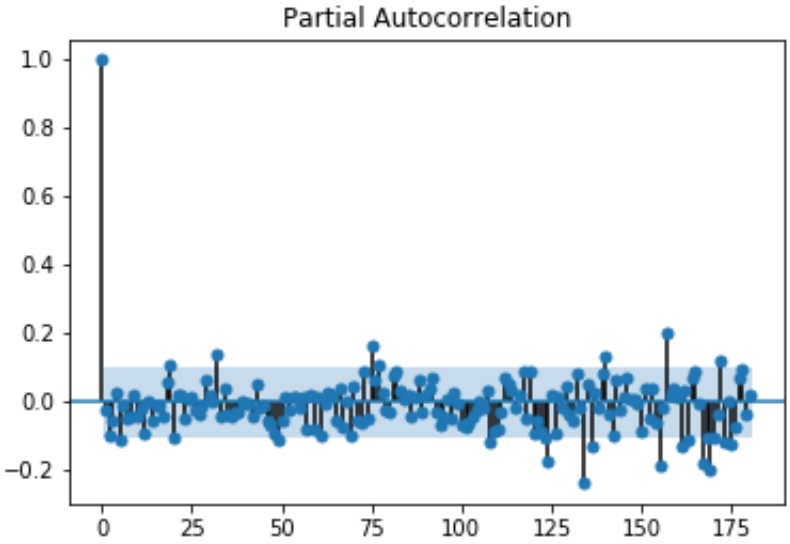


Figure 3.6: PACF of cash flow in SEK with 95% confidence interval

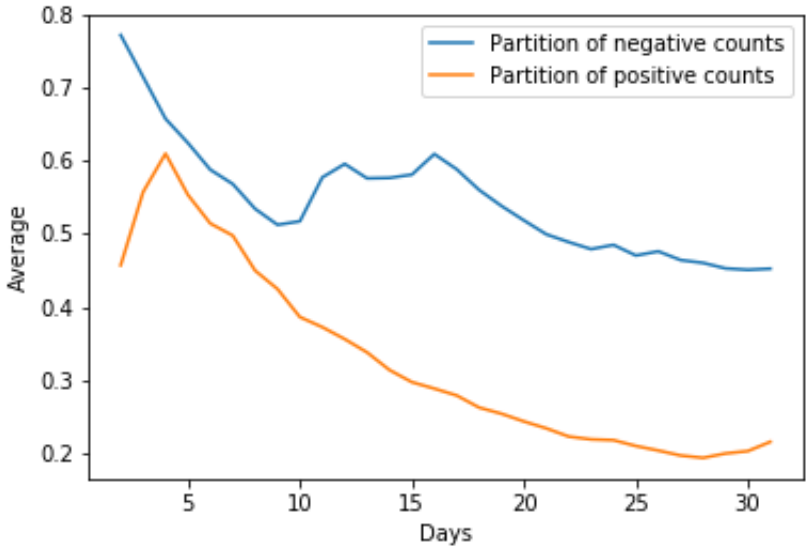


Figure 3.7: The average amount of transaction per day for the partitioned data.

0.6 while it is 0.2 for the first 30 days(i.e. the full month). What this tells us is that there is an heterogeneous distribution of transactions throughout the month, if it was homogeneous distributed the lines in the plot would be flat. Instead the parameter that governs this stochastic process seems to be time dependent. This encourages the use to describe the transactions as a Non-Homogeneous Poisson Process, which will be introduced later. The pattern is reoccurring and should therefore be used for the cash flow modelling since this is a stylized property of the cash flow.

3.4 Data Quality

There are some reasons to suspect that the data does not fully reflect the true nature of companies. The data indicates that the transactions spike in the beginning of the month and then decay over time. This is contrary to the more intuitive description that more transactions should be in the last days of the month as payroll and bills are due, one explanation for this is that companies delay their accounting entries in the system. This has an impact on the accuracy of the model since if the data does not fully reflect the reality of the behaviour of the cash flow the model will suffer. Another aspect of the data which needs to be considered is the potential survivorship bias of the data. Only companies with ongoing businesses have been included in the data set which means that according to the data there is no possibility that a company can ever go bankrupt. This is of course a troublesome property since the results will be skewed. The result from this bias is that the model will show that companies with a poor Financial Health will eventually rebound, but this is obviously far from an true in reality.

4

Mathematical Background

This chapter introduces the mathematical theory which the model stands on. The first part covers stochastic processes that is important for the intended statistical model. Various Markov Chain Monte Carlo methods are introduced and then there is a brief introduction to Hidden Markov Models. Both traditional algorithms such as Viterbi and Baum-Welch but also a non-linear method, the Particle Filter, are introduced.
The model in this thesis will be done in a hierarchical fashion, what that means is that there is a hierarchy of different distributions which will be used to capture the structure in the data. This structure makes it possible to capture some stylized properties of the data which simpler models cannot do.
The model in this thesis will utilize Bayesian Hierarchical Modelling which is a model type that works in several levels where each parameter has its own prior distribution itself. This makes it possible to build models which can adapt itself efficiently to the data. This specific type of model is introduced briefly in [2].
In this model the number of transactions is at the top level and the transaction amounts is at the lower level of the model. The modelling is done within the Bayesian paradigm which means that the observed data is seen as fixed and that the parameters are stochastic, this is orthogonal to the frequentist framework where the data is random and the parameters are fixed, but unknown.

4.1 Poisson Processes

The amount of transactions per day can be seen as a Poisson Process with an unknown parameter λ. A Poisson Process is a distribution of times of random events, where the number of events in different time intervals are independent from each other. $N(t)$ denotes the number of events up to time t, and $N(t_1) - N(t_2)$ is Poisson$(\lambda(t_2 - t_1))$ distributed. The density of the Poisson distribution is given in eq. (4.1) where λ is the hyper parameter for the distribution

$$\pi(n) = \frac{\lambda^n}{n!} e^{-\lambda}, \quad n = 0, 1, 2, ... \tag{4.1}$$

where n is the number of events up to time t. The process is used in many disciplines since many phenomenon in nature, social sciences and economics follow this kind of distribution. This is because an event which is strictly positive and discrete, the number of rainfalls during a week for example, is well suited to be modeled as a Poisson Process. More can be found in [8] and [9].

4.1.1 Non-homogeneous Poisson Process

There is an extension to the Poisson process, namely the Non-homogeneous Poisson Process (NHPP). A NHPP is a Poisson process with a non-constant and time dependent, rate parameter. The process is defined ineq. (4.2) where $\lambda(x)$ is the parameter for a specific time. The process is very similar to the one described in eq. (4.1).

$$\pi(N(t) = n) = \frac{(\Lambda(t)t)^n}{n!} e^{-\Lambda(t)} \tag{4.2}$$
$$\Lambda(t) = \int_0^t \lambda(x)dx < \infty$$

The process is described in detail in [9]. The Non-Homogeneous Poisson process makes it possible to model data which have time dependent parameters, which dynamic (changing) systems most often have. An example of this could be any seasonal pattern, such as how the revenue changes for a company depending on the season. As discussed in the data analysis financial data often exhibit time dependent behaviour, so by utilizing a NHPP the model will be able to catch this time dependent behaviour.

4.1.2 Compound Poisson Process

The model aims to capture the behaviour of cash flow of individual companies by modelling it as a compound Poisson process, which is defined in eq. (4.3). This will be done by the two features, sums per day and transactions per day. In this specific case the amount of transactions per day will be modeled as a Non-Homogeneous Poisson Process and the amount will be modeled as a Gamma distribution. One assumption that is usually made is that the Poisson Process and Gamma Distribution are independent, which can be seen in the defined Compound Process.

$$Y_t = \sum_{i=0}^{N(t)} D_i \tag{4.3}$$
$$N(t) \sim \text{Poisson}(\Lambda(t)), \quad D(t) \sim \text{Gamma}(\alpha, \beta)$$

The interpretation here is that the jump, measured in days, between transactions are Poisson distributed while the intensity is Gamma distributed. The rate parameter may change which could enable the process to capture some stylized behaviour of the data series. The the intensity of the cash flow is Gamma distributed.
There are many different ways to estimate the parameters for the Non-Homogeneous Poisson Process. One way would be to partition the data series by a fixed seasonal pattern, or in a non-parametric way which is discussed in [5].

4.2 Markov Chain Monte Carlo Methods

Markov Chain Monte Carlo (MCMC) methods is a collection of algorithms which allows sampling from a wide array of distributions which cannot be directly sampled from.

One general Markov Chain Monte Carlo method which is often used is the Metropolis-Hastings algorithm. A more detailed description can be found in [2].

4.2.1 Metropolis-Hastings

Metropolis-Hastings is a Markov Chain Monte Carlo Methods to sample (dependent samples) from a density p, while one only has an unnormalized version $\tilde{p}(z) = cp(x)$ of the density available. To do so one samples from a proposal distribution which can depend on the current state $z^{(t)}$ to propose a new state z^*. Whether z^* become the new state $z^{(t+1)}$, or the chain remains in the old state $z^{(t+1)} = z^{(t)}$ is decided by the flip of a biased coin, thus obtaining samples z^1, z^2....
The proposal distribution depends only on the current state so that the proposal density becomes $q(z \mid z^{(t)})$, in order for the samples z^1, z^2... to create a Markov Chain.
The probability for the proposal to be accepted is

$$A(z^*, z^{(t)}) = \min\left(1, \frac{\tilde{p}(z^*)q(z^{(t)} \mid z^*)}{\tilde{\pi}(z^{(t)})q(z^* \mid z^{(t)})}\right). \tag{4.4}$$

One does not need the normalized distribution since the normalizing constant cancels out.
In eq. (4.4) $q_k(\cdot \mid z^{(t)})$ is the proposal density and z^* is the proposal, and t denotes what step the algorithm is in. By following this procedure one can sample from a distribution without knowing the normalizing constant of the distribution. The Metropolis-Hastings, as all other Markov Chain Monte Carlo Methods utilizes random number generators to produce their output.
Important observation: If one has a symmetric proposal, that is $q(z^* \mid z) = q(z \mid z^*)$ (4.4), simplifies to

$$A(z^*, z^{(t)}) = \min\left(1, \frac{\tilde{\pi}(z^*)}{\tilde{\pi}(z^{(t)})}\right). \tag{4.5}$$

Thus for a symmetric proposal, one does not need to know the density of the proposed value. It should also be noted that

$$z^* = z + \mathcal{N}(0, \sigma^2) \tag{4.6}$$

where σ^2 is some variance, also creates symmetric proposals. Also different kernels q_k can be combined to create a sampler to sample from $\tilde{\pi}$. See [2], page 541.

4.2.2 Gibbs Sampling Algorithm

Gibbs sampling is an iterative algorithm which utilizes the Hamiltonian Monte Carlo, which was introduced earlier, in each step. The algorithm works by sampling from conditional marginal distributions for each parameters, by then using Bayes theorem one is able to approximate the true distribution. The algorithm is formally presented in [2] and is displayed in algorithm 1 where $\pi(z_i \mid z^{\tau}_{all|-i})$ is the density of z_i conditioned on all remaining variables.
Then (4.4) equals 1 and one can take the proposed value z^* as new state.

Algorithm 1 Gibbs Sampling

1: Initialize $z_i : 1, .., M$
2: **for** $\tau = 1, .., T$ **do**
3: **for** i=1,...,M **do**
4: Sample $z_i^{\tau+1} \sim \pi(z_i \mid z_{all|-i}^{\tau})$

One of the many things one needs to be cautious of while using the Gibbs sampler is that the first part of the sampling chain does often not represent the distribution very well, since it is far from convergence. There are two strategies one can employ here, either one removes the first iterations, also called the burn in, or one samples so many points that the initial part of the chain cannot skew the results in a meaningful way.

4.2.3 Hamiltonian Monte Carlo

An issue with many MCMC methods is that they can have a random walk behaviour and that they are sensitive to correlated parameters. A method which avoids these characteristics is the Hamiltonian Monte Carlo [10] which takes steps with information from the first order gradient.
The state z is augmented by a "momentum" variable r.
We give the momentum variable the density

$$p(r) := (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{1}{2} r^T r\right) \tag{4.7}$$

of a standard multivariate Normal distribution.
The state space of z and r is defined to be the product

$$\pi(z, r) = \pi(z)p(r). \tag{4.8}$$

Then samples from $\pi(z, r)$ can be used as samples of $\pi(z)$ discarding r.
Now we define the Hamiltonian H which can be understood the total energy system,

$$H(r, z) := E(z) + K(r) \tag{4.9}$$

where

$$K(r) := \frac{1}{2} r' r.$$

We also define $E(z)$,which is the potential energy of the variable z, as

$$\pi(z) =: \frac{1}{Z_p} \exp(-E(z)). \tag{4.10}$$

Then we can rewrite

$$\pi(z, r) = \frac{(2\pi)^{-\frac{N}{2}}}{Z_p} \exp(H(z, r)). \tag{4.11}$$

This can be seen by

$$\pi(z,r) = (2\pi)^{-\frac{N}{2}} \exp\left(-\frac{1}{2}r^T r\right) \frac{1}{Z_p} \exp(-E(z)) \tag{4.12}$$

$$= \frac{(2\pi)^{-\frac{N}{2}}}{Z_p} \exp\left(-\frac{1}{2}r^T r - E(z)\right).$$

Then we get from eq. (4.9) that

$$\pi(z,r) = \frac{(2\pi)^{-\frac{N}{2}}}{Z_p} \exp(-H(z,r)).$$

Our goal is now to sample from the joint distribution π.
The algorithm creates a Markov Chain of stochastic updates of the momentum variable r and Hamiltonian dynamical updates from the leapfrog algorithm. The momentum variable r is defined in eq. (4.13) and should be interpreted as the rate of change of the state variable z with respect to the time τ.

$$r_i = \frac{dz_i}{d\tau} \tag{4.13}$$

The momentum variable r can be deduced from $E(z)$ as in eq. (4.14) and r can then be given by integration.

$$\frac{\partial r_i}{\partial \tau} = -\frac{\partial E(z)}{\partial z_i} \tag{4.14}$$

The Leapfrog Discretization method is used to integrate differential equations numerically which is needed in the Hamiltonian Monte Carlo algorithm. This is done by discrete time approximations of the time variable τ and the positional variable z. The steps in the algorithm is described in eq. (4.15) where ϵ is the step length.

$$\hat{r}_i(\tau + \epsilon/2) = \hat{r}_i(\tau) - \frac{\epsilon}{2}\frac{\partial E}{\partial z_i}(\hat{z}(\tau)) \tag{4.15}$$

$$\hat{z}_i(\tau + \epsilon) = \hat{z}_i(\tau) + \epsilon \hat{r}_i(\tau + \epsilon/2)$$

$$\hat{r}_i(\tau + \epsilon) = \hat{r}_i(\tau + \epsilon/2) - \frac{\epsilon}{2}\frac{\partial E}{\partial z_i}(\hat{z}(r + \epsilon))$$

What the leapfrog algorithm does explicitly is that it works in two steps, first in half step updates of the momentum variables and then a full length step update of the position variable. Note that the Leapfrog Integration is time-reversible, which means that the starting point $z(\tau)$ will be reached using n negative time steps $-\epsilon$ starting from $z(\tau + n\epsilon)$.
The leapfrog integration then produces a deterministic candidate (z^*, r^*). For z^*, r^* to have a density also pertubes r^* randomly.
The Hamiltonian algorithm then takes $z^*, -r^*$ as Metropolis-Hastings proposal for the new state. This makes the proposal density $q(\cdot \mid \cdot)$ symmetric. Then the acceptance probability becomes

$$A((z,r),(z^*,-r^*)) = \min\left(1, \frac{\tilde{p}(z,r)q((z^*,-r^*) \mid (z,r))}{\tilde{\pi}((z^*,-r^*))q((z,r) \mid (z^*,-r^*))}\right) \tag{4.16}$$

$$= \min\left(1, \frac{\tilde{\pi}(z,r)}{\tilde{\pi}((z^*,-r^*)}\right)$$

$$= \min\left(1, \frac{\frac{(2\pi)^{-\frac{N}{2}}}{Z_p}\exp(H(z,r))}{\frac{(2\pi)^{-\frac{N}{2}}}{Z_p}\exp(H(z^*,-r^*))}\right)$$

$$= \min(1, \exp\{H(z,r) - H(z^*,-r^*)\}).$$

But since $K(r) = K(-r)$ by definition, we get that

$$= \min(1, \exp\{H(z,r) - H(z^*,r^*)\}).$$

This is similar to what was shown in eq. (4.5). The acceptance probability is then the difference between the Hamiltonian before and after the leapfrog integration.
The Hamiltonian Algorithm is a form of Metropolis-Hastings, the main difference to random walk Metropolis-Hastings is that the Metropolis Hastings Algorithm introduced earlier in eq. (4.4) is that the Hamiltonian takes the gradient of the log probability into account and not only the distribution as Metropolis Hastings does, and nowadays the gradient is almost always available through automatic differentiation. Hamiltonian Monte Carlo is preferred over other MCMC methods such as the Metropolis Hastings Algorithm and the Gibbs Sampler since it is much more effective in higher dimensions and for complicated distributions, as discussed by Gelman et al in [10], since the Hamiltonian converges a lot quicker and more accurate.

4.2.4 No-U-Turn Sampler (NUTS)

There are two parameters which needs to be set by the user of the Hamiltonian Monte Carlo, step length and the amount of steps. These two quantities usually needs to be set on a case by case basis and creates the need for hand tuning the method. This however is not trivial and it often a bespoke solution is needed and this limits the generality of the method. But one instead can use the No-U-Turn Sampler which was introduced by Hoffman and Gelman in [10] which resolves these issues for the user.
The number of steps, L, in the leapfrog part of the NUTS algorithm corresponds to the amount of steps it takes for the distance between the proposal $\hat{z}$ and the initial values z to no longer increase. This leads to that one should run leapfrog steps until the expression in eq. (4.17) becomes smaller than 0.

$$\frac{d}{dt}\frac{(\hat{z}-z)^2}{2} \tag{4.17}$$

This method does not always show time-reversibility which is necessary in the process. This is however dealt with by running the Hamiltonian simulation forward and backward in time which creates the time reversibility. The other question which the NUTS algorithm resolves is the question of the step length. A too small step size wastes computation time since is takes more steps to reach convergence and too large steps leads to high rejection rates, this is what NUTS aims to solve.

The step size ϵ is set in the tuning phase of the algorithm by dual averaging. The specific algorithm used to determine the step size is found in [10]. The algorithm is implemented with the package PYMC3, which is a library for Probabilistic Programming. PYMC3 is available for in the Python programming language, a more detailed description of the implementation can be found in [11].

4.2.5 Automatic Differentiation Variational Inference (ADVI)

An aspect of MCMC methods which cannot be overlooked is the importance of the selection of starting point for the procedure. In [12] Kucukelbir et al. developed a method to choose an optimal starting point through Variational Inference, which is a methods to approximate intractable integrals.

The model finds automatically a variational family and optimizes this variational objective. The first step in the ADVI algorithm starts with a differential probability model $p(x, \theta)$ and by introducing latent variables which relates to the observations in θ. These latent variables are then transformed to the real space variables ζ. The next step is to use variational inference to transform the approximate posterior inference into a optimization problem, the objective one want to optimize is the evidence lower bound (ELBO), which is what is being optimized in Variational Inference.

This objective is optimized with stochastic optimization which is done by stochastic gradient ascent, this guarantees reaching a local optimum under certain assumptions on the step size sequence, or learning rate as it is also called. The specific algorithm for this is outlined in [12] and is also implemented in PYMC3.

4.3 Hidden Markov Model (HMM)

Hidden Markov Models are a class of models which are used to estimate hidden, or latent, states in dynamical systems. HMMs are suitable for sequential data where there is some auto-correlation between the data points. These structures in the data is easily modeled and captured by the use of this kind of model.

In hidden Markov models the latent state at time t is defined by X_t and the observed state at time t is defined by Y_t, both X and Y are real numbers. The definition is that a stochastic process (X_n, Y_n) is a Hidden Markov Model if Y_n is an observation, X_n is not observable (hidden) and that the process is Markovian i.e. $\pi(X_n \mid X_1 = x_1, ..., X_{n-1} = x_{n-1}) = \pi(X_n \mid X_{n-1} = x_{n-1})$, it is also required that the observations are conditionally independent of each other.

A discrete HMM consists of a transition matrix, which determines the probability of different transitions between states, an emission matrix which determines the probability observations given the hidden state and finally initial matrix which sets the initial state of the model. The main idea behind the Hidden Markov Model is shown in fig. 4.1 for a model in discrete time and discrete states.

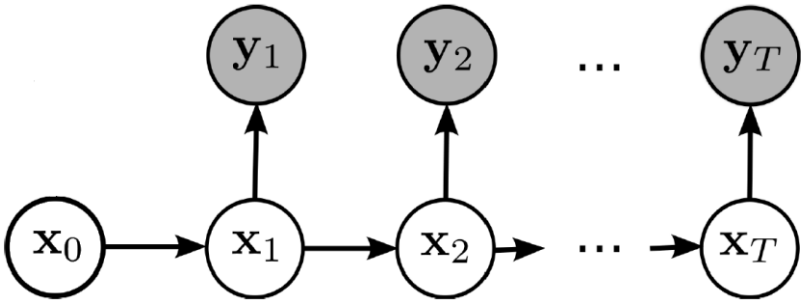


Figure 4.1: Illustration of how a HMM is built from [1]

There is however a wide array of different types of Hidden Markov Models since the definition allows for many different types of models. The most usual case is that the number of states is discrete and the amount is determined beforehand but there are other methods which do not rely on these assumptions. By having a fixed amount of states the transition and emission matrices are easily calculated with some algorithms which will be presented later on, however, the assumption of a fixed number of states is not always suitable.
What kind of Hidden Markov Model model one should choose depends on the data. If the data follows a Gaussian distribution one can choose a simpler model, which can easily be implemented via packages such as *hmmlearn*, which originally was a Scipy package in Python and more information can be found in [13]. But if the data follows a more complicated distribution, such as a multi modal distribution, one needs to implement a more complicated model to catch the structure of the data. If one chooses a model like this one cannot reside to already built models, instead one need to build a bespoke model.

4.3.1 Sequential Monte Carlo (SMC)

Sequential Monte Carlo methods are used for filtering, these methods are used in statistics to remove features from an observed signal which are unwanted. Examples of this could be observation noise in a data set or in this case extract the latent state from observations. So a SMC is a class of algorithms for statistical estimation of internal states of a process which have some noise.
SMC is preferred over Gaussian filtering methods when the distributions are multi-modal or discrete, in these cases the Gaussian methods are poor approximations according to Särkää in [14].
The general idea behind Monte Carlo methods is that one sample independent samples X_t from $\pi(X|Y)$, i.e. the posterior density. Then one can estimate the expected value of $E(f(x)|y)$ where $f()$ is an arbitrary function and N is the amount of samples such as

$$E(f(X)|Y) \approx \frac{1}{N}\sum_{i=1}^{N} f(X^{(i)}) \tag{4.18}$$

But often one cannot sample N independent samples from a distribution. Instead one can use Importance Sampling which utilizes an approximate distribution called

Importance Distribution which is denoted $p(x|y)$. Then it can be shown that the expected value of the posterior density is

$$E(f(X)|Y) \approx \sum_{i=1}^{N} w^{(i)} f(X^{(i)}) \tag{4.19}$$

where

$$w^{(i)} = \frac{1}{N} \frac{\pi(X^{(i)}|Y)}{p(X^{(i)}|Y)}. \tag{4.20}$$

But this method does not work for sequential data which is the main scope of the thesis. Instead one uses Sequential Importance Sampling to evaluate the expected value of the posterior density. This method is used to for models which can be expressed such as

$$\begin{aligned} X_i &\sim \pi(X_i|X_{i-1}) \\ Y_i &\sim \pi(Y_i|X_i). \end{aligned} \tag{4.21}$$

The expected value of the posterior density, where k is the time step, becomes

$$E(f(X_k)|Y_{1:k}) = \sum_{i=1}^{N} w_k^{(i)} f(X_k^{(i)}) \tag{4.22}$$

where

$$w_k^{(i)} \propto \frac{\pi(Y_k|X_k^{(i)})\pi(X_k^{(i)}|X_{k-1}^{(i)})}{p(X_k^{(i)}|X_{0:k-1}^{(i)}, Y_{1:k})} w_{k-1}^{(i)}. \tag{4.23}$$

The weights in eq. (4.23) do not sum to one but one can normalize the weights to get this property.
This method often have a degeneracy problem, this means that many particles have near a zero weight. To avoid this in SMC methods one uses Sequential Importance Re-Sampling where one replaces the old N samples for each time step with N new samples.
It is in this fashion SMC methods work, by utilizing Monte Carlo methods for sequential data where one av avoids the issue of not being able to sample independent samples from the distribution.

4.3.2 Bootstrap Filter

One of the simpler Sequential Monte Carlo methods is the Bootstrap Particle Filter which will be used in this thesis. This is a good choice since if a system is not Gaussian or have linear emission many filtering methods fail to work as described by Särkkä in [14]. Since the thesis aims to not introduce restrictive assumptions on the system, such as linearity or Gaussianity, the usage of the Bootstrap Filter becomes natural. By using the Bootstrap filter one can avoid making any of these assumptions.

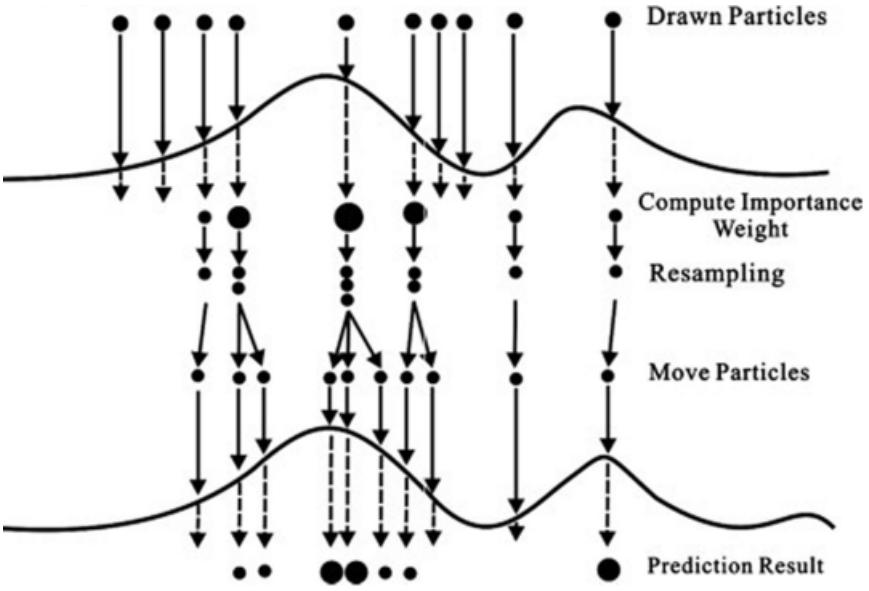


Figure 4.2: Illustration of how a bootstrap filter works from [1]

The main family of Sequential Monte Carlo methods is the Sequential Importance Sampling and this method is introduced in [15]. But this method fails to handle the degeneracy problem which can occur if some or all of the particles in the filter have close to zero weight. This can be handled by another method called Sequential Importance Re-sampling (SIR). There is a wide array of flavours of SIR methods but in this instance we will only consider the Bootstrap Particle Filter.
The result from a Particle Filter an estimation of a sequence of hidden states given a sequence of observations and a predefined model. The important thing to notice here is that it estimates a sequence of states, this is vital since time series can often be path dependent, i.e. the path of which states occur matter.
The outline of the algorithm can be seen in algorithm 2. An illustration of the bootstrap filter and how it works can be found in fig. 4.2. The illustration shows how one samples particles and then calculate the importance weight after each instance. The algorithm which governs the bootstrap filter is specified in [16]. One should note that one issue with bootstrap filters, and all other particle filters, is that a bespoke solution is almost always required by the user. So there is no quick and easy way to important it but on has to hard code a new filter for each process. In algorithm 2 the Bootstrap Particle Filter is outlined and the Categorical Distribution is a generalized Bernoulli distribution where the variable can take any of k categories with a certain probability.
There are some quantities which needs to be known beforehand and these are $\pi(X_0)$ and $\pi(Y_t \mid X_t)$. The idea is to view the observed data Y as a function such as $Y_t = X_t + \mathcal{N}(0, \sigma^2)$ where the parameters are set from the data. The stochastic process X_t is most easily seen as a model for the process, which should be built separately. Then one can model the process such as $\pi(Y_t \mid X_t) = \mathcal{N}(X_t, \sigma^2)$ which means that one just needs to calculate the probability density function.
To estimate $\pi(X_t|X_{t-1})$ one can assume that $X_t = aX_{t-1} + \mathcal{N}(0, \sigma^2)$, or just forward sample the model we have built. The vital part of the Bootstrap Filter is the weight

Algorithm 2 Bootstrap Filter

1: At $t = 0$ initialize:
   $x_0^i \sim \pi(x_0)$ for each $i = 1, ..., N$. Assign $w_0^i = 1/N$
2: **for** t=1 to T **do**
3: Resampling Step: $a_t^i \sim \text{Categorical}((W_{t-1}^j)_{j=1}^N)$
4: Propagate Forward: $x_t^i \sim \pi(x_t|x_{t-1}^{a_1})$
5: Compute Average of Hidden State: $\pi(x_{t-1}|y_{t-1}) \approx \frac{1}{N}\sum_i W_{t-1}(x_{t-1})$
6: Evaluate $\tilde{W}_t^i = \pi(Y_t|X_t^i)$ and normalize $W_t^i = \frac{\tilde{W}_t^i}{\sum_{j=1}^N \tilde{W}_t^j}$
   return $\pi(x|y)$

that is assigned to each particle since this determines that hidden state, the drawback of the method is that this part can often be close to zero. This is the case since y_{t+1} in eq. (4.24) can be far from the estimated latent state for that time partition.

$$\begin{aligned} x_{t+1}^i &\sim \pi(x_{t+1} \mid \tilde{x}_t^i) \\ w_{t+1}^i &= \pi(y_{t+1} \mid x_{t+1}^i) \end{aligned} \tag{4.24}$$

The higher the assigned variance of the emission density in eq. (4.24) the lower the probability of close to zero weight for the particles. But if one assigns a very high variance for the distribution, almost all particles will have the same weight. This would then render the bootstrap filter worthless, so the choice of the variance for the emission density will be an important aspect of the model fit.
The Bootstrap Filter is a very simple particle filter and can be implemented without much effort. With this simplicity comes some drawback such as its inefficient importance distribution $\pi(X_t \mid X_{t-1})$. The consequence of this inefficiency is that one may require a large amount of Monte Carlo samples for good estimation results. These aspects are discussed in depth in [14]. One thing that needs to be monitored with a particle filter is the concentration of weights among the particles. There is a possibility of clustering of weights such that only a few particles carry all the weights. This is not wanted since this means that there will be a more random error in the filter and this will yield inconsistent results.

5
Methods

In this part of the thesis the methods and algorithms which were introduced in *Mathematical Background* will be implemented and fitted to the observed data.The data is used to fit the Hierarchical Bayesian Model which will be done with some Markov Chain Monte Carlo Methods and some analysis of the convergence of the chains.
The Particle filter will then be used with the fitted cash flow model to estimate the latent process. All of the methods are implemented in Python but will only be shown in pseudo-code. The data in this part comes from a single company for illustration purposes, the methods have however been implemented on the full data set. Not all methods introduced in the section above will be implemented, only the ones that were found useful for this specific data set and aim.

5.1 Model Building

The model building was done in a Bayesian hierarchical manner. Transactions and sums are being viewed as separate processes and comes from different distributions and will be the main features of the model. The transactions will be viewed as discrete observations while the sums will be seen as a continuous process.
The two processes, transaction counts and transaction sums are assumed to be independent, this assumption has also been used in [4] with success.
All of the Markov Chain Monte Carlo algorithms were implemented with the python package PYMC3 which allows the user to build a wide array of algorithms in a modular way with different underlying distributions that can be hierarchical, as described in [11]. The package is open source and give an opportunity to use powerful algorithms without having to build new samplers for each new model. The current drawback of PYMC3 is that it does not use the GPU of the computer which makes the methods take quite some time to run.

5.1.1 Model Architecture

In section we will specify the model in detail. We first describe a model for the cash-flow without observation error. We model the cash-flow as follows.
We model the times of positive and negative transactions as independent Non-homogeneous Poisson processes, where we denote by $N^+(t)$ and $N^-(t)$ to be the number of events up to time t (respective), see eq. (4.2), where time is measure in days, so $t \in [i-1, i)$ are times during the i's day, $i \in \mathbb{N}$.

$S_i = N^+(i) - N^+(i-1)$, is the number of random time points of the non-homogeneous Poisson process during day i with a positive transaction. Therefore

$$S_i \sim \text{Poisson}(\lambda_p(i)) \tag{5.1}$$

where $\lambda_p(i) = \Lambda(i) - \Lambda(i-1)$.
$Q_i = N^-(i) - N^-(i-1)$ is the number of random time points of the non-homogeneous Poisson process with a negative transaction which is defined as

$$Q_i \sim \text{Poisson}(\lambda_n(i)) \tag{5.2}$$

where $\lambda_n(i) = \Lambda(i) - \Lambda(i-1)$.
The size of the positive and negative transactions are assumed to be independent. The positive transaction size are modeled to come from a mixture of two Gamma distributions, which is a bimodal distribution, this is in line with our empirical findings. The distribution of the positive transaction size K_i^k, the kth transaction of day i, has distribution given by

$$p \sim \text{Dirichlet}(\alpha) \tag{5.3}$$

$$\kappa | p \sim \text{Binomial}(N, p)$$

$$\alpha_1 \sim \text{Gamma}(\alpha, \beta)$$

$$\alpha_2 \sim \text{Gamma}(\alpha, \beta)$$

$$\beta_1 \sim \text{Gamma}(\alpha, \beta)$$

$$\beta_2 \sim \text{Gamma}(\alpha, \beta)$$

$$K_i^k | \alpha_1, \beta_1, \alpha_2, \beta_2, \kappa \sim \text{Gamma}(\alpha_\kappa, \beta_\kappa)$$

where α is the amount of mixture distributions, which in is this case is 2.
The negative transactions are modeled to come from a Gamma distribution. The negative transaction size L_i^k is defined as

$$\beta_n \sim \text{Gamma}(\alpha, \beta) \tag{5.4}$$

$$\alpha_n \sim \text{Gamma}(\alpha, \beta)$$

$$L_i^k | \alpha, \beta \sim \text{Gamma}(\alpha_n, \beta_n).$$

The cash flow on day i is the sum of all transactions done on day i and denoted as C_i. K^k is the kth positive transaction on day i and L^k denotes he kth negative transaction on day i. Therefore

$$C_i = \sum_{k=1}^{S_i} K_i^k + \sum_{k=1}^{Q_i} L_i^k. \tag{5.5}$$

The total accumulated cash flow at time i is said to be X_i and is defined as

$$X_i = \sum_{k=1}^{i} C_k \tag{5.6}$$

Note that we can sample X_i conditional on X_{i-1}, that is we can sample from

$$X_i \sim \pi(X_i \mid X_{i-1}). \tag{5.7}$$

We consider the daily data (after pre-processing) to be accumulated the sum of positive and negative transactions each day, observed with error. Hence

$$Y_i = X_i + \epsilon_i. \tag{5.8}$$

where ϵ_i is normal distributed with mean 0 and variance σ^2 and X_i is the latent state at time i.
This has the effect of smoothing the (now latent) process X_t conditional on the observations: the samples of the posterior distribution of X tend to be smoother, as part of the daily variation is absorbed by the noise quantities then they would be if fitting the process X directly to the observed data.
The non-normalized weight of the particles in the Particle Filter is calculated, where i is the index of particles such as $j = \{1, ..., N\}$ and N is the number of particles, as

$$\tilde{W}_i \sim \pi(Y_i \mid X_i^j). \tag{5.9}$$

This leads to

$$\pi(\tilde{W}_i) = \mathcal{N}(Y_t; X_t, \sigma_i) \tag{5.10}$$

The normalized weights W_j are the evaluated to be

$$W_j = \frac{\tilde{W}_j}{\sum_{j=1}^{N} \tilde{W}_j}. \tag{5.11}$$

5.2 Inference Procedure

This section introduces how the inference of the model will be conducted for the transactions and the transaction intensity.

5.2.1 Modelling of Transactions

It was further shown in the data analysis that the amount of transactions seem to be time dependent. The majority of the transactions are being made at the beginning of the month, this characteristic is something that ought to be captured in the model. For the intensities λ_p and λ_n of the NHPP we partition the month into two parts, and set λ constant on each part. The partition was chosen such that 50% of the transaction are being made on average in each part.
We estimate λ_p and λ_n separately by the following procedure. It is enough to describe it for λ corresponding to the first partition of the positive transactions. The other intensities are estimated similar.
We denote the days belonging partition 1 by $\mathcal{P}_1$.

The denote positive transactions S_i for days $i \in \mathcal{P}_1$ by assumption

$$S_i \sim \text{Poisson}(\lambda). \tag{5.12}$$

This can generalized for any partition. The prior used is a gamma prior such as $\lambda \sim \text{Gamma}(1, 1)$ for all the partitions. Then the likelihood becomes

$$\pi\left(\sum_{i\in\mathcal{P}_i} O_i|\lambda\right) = \frac{e^{-|\mathcal{P}_1|\lambda}\lambda^{\sum_{i\in\mathcal{P}_1} O_i}}{\prod_{j\in\mathcal{P}_1}(O_j!)} \tag{5.13}$$

It can then be shown that the posterior distribution $\pi(\lambda \mid O)$ is distributed such as

$$\text{Gamma}\left(\sum_{i\in|\mathcal{P}_1|} O_j + \alpha, |\mathcal{P}_1| + \beta\right) \tag{5.14}$$

This closed form expression is possible due to the Poisson-Gamma conjugacy.
This conjugacy can be use to estimate the intensities assuming no observation noise, but also as step in a Gibbs sampler in the full model assuming observations of Y_i.

5.2.2 Modelling of Sums

Is was assumed that the positive sums are bimodal distributed with two different gamma distributions. These hyper parameters are being estimated the NUTS algorithm and the hierarchical structure of the bimodal gamma distribution are given in eq. (5.3). In this setting α_p and β_p are priors for the distributions, p is Dirichlet distributed where $\alpha = 2$ which means that it is a Binomial distribution and N is the amount of days in the data set.
The negative sums were shown in the data analysis not be governed by a mixture distribution but generally something that looks like an exponential or Pareto distribution. The estimation of this distribution is done in the same way as for the mixture distribution but with a more simplistic architecture.
There is no nice conjugacy for for an unknown α and β,such as it is for the Poisson-Gamma conjugacy above, instead on can run the NUTS algorithm for many iterations. The proposal in the NUTS algorithm can also be used as a proposal in a Gibbs Sampler, sampling the joint posterior of parameters and latent path X from observations Y.

5.3 Estimation Results

The results of the estimation of the distribution are shown below with a discussion regarding the settings of the code and the choice of priors. All of the computations were run one a Dell with an Intel Core i7-870H CPU with 16 GB RAM.

5.3.0.1 Toy Example: Gamma Mixture

To illustrate the method 500 data points are generated from a mixture of two gamma distributions. 70% of the data comes from the Gamma$(1, 1)$ and the remaining data

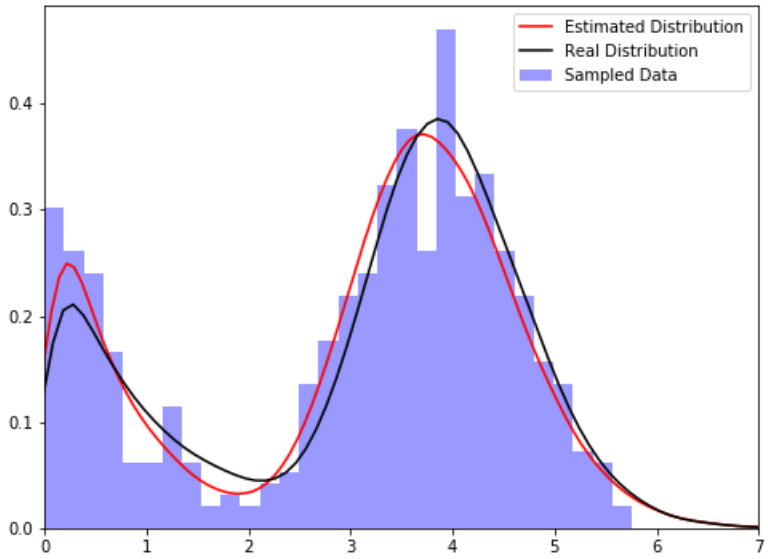


Figure 5.1: Histogram and density plot of real and estimated data.

Variabel	Mean	SD	HPD 3%	HPD 97%
q	0.290	0.024	0.244	0.335
p	0.710	0.024	0.665	0.756
α_1	0.907	0.103	0.722	1.105
α_2	26.492	2.231	22.602	30.838
β_1	1.004	0.215	0.627	1.413
β_2	6.650	0.553	5.594	7.653

Table 5.1: Statistics for the NUTS algorithm

points comes from Gamma$(30, 7.5)$. The priors are set to be $p \sim$ Dirichlet(2), $\alpha_1 \sim$ Gamma$(1, 1)$, $\alpha_2 \sim$ Gamma$(20, 1)$, $\beta_1, \beta_2 \sim$ Gamma$(1, 1)$. 4000 samples are drawn with the NUTS algorithm, 4000 tuning samples are being used as well, this makes it easier to get convergence of the chains. The results can be seen in table 5.1, HPD means Higher Posterior Density and is an interval for the value of an unobserved parameter. As one can see the estimated distribution is very close to the true underlying distribution. In fig. 5.1 it is obvious that the estimated density corresponds very well to the actual data.

An important part in MCMC methods is that the chain converges, this can be checked by visual inspection of the trace plots of the chains. In fig. 5.2 one can see the two of each variable which were estimated with the NUTS algorithm. It is clear that the chains converge quickly and there seems to be a good mixture of the chains. One should note however that there might be divergences, i.e. that the chain fails to converge, in the chains on occasions so therefore should one always evaluate the chain performance.

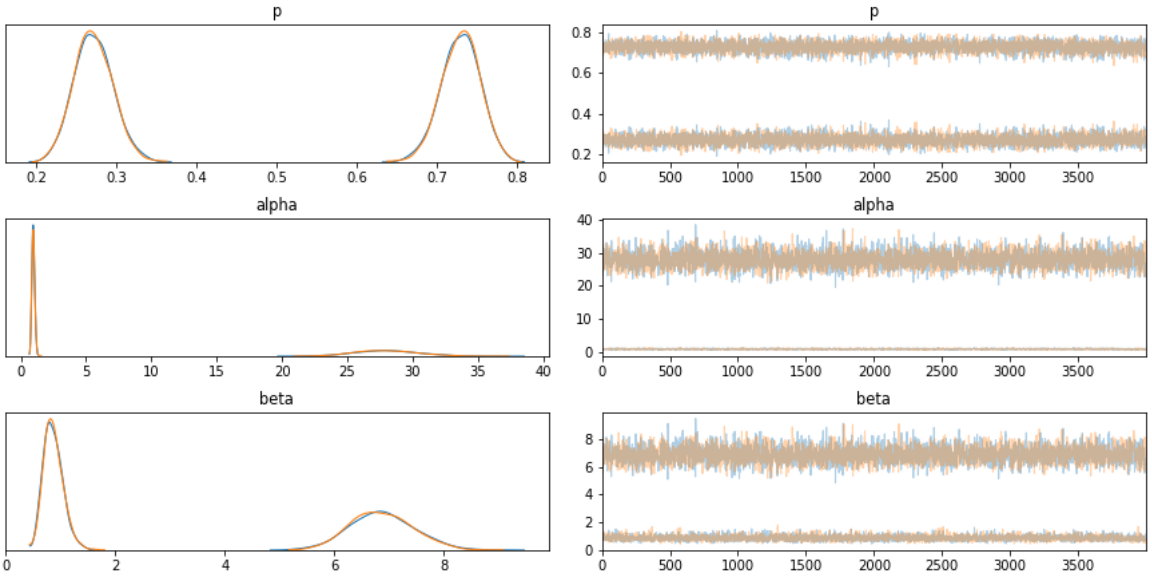


Figure 5.2: Chains for NUTS algorithm

5.3.1 Settings

There are some aspects of every MCMC sampler that needs to be evaluated. These are Burn-in, acceptance rate and tuning and as with any Byesian Model one needs to set priors. The general idea of this thesis is to sample enough to make sure the burn in does not have an impact on the sampler. The acceptance rate is directly proportionate to the step length of the sampler and since the step size is determined automatically by the NUTS sampler give an acceptance rate one can modify the step length by modifying the acceptance rate. If one increases the acceptance rate the step length will decrease. The default acceptance rate for the NUTS sampler is 80%, which is used for the non-mixture models. For the mixture model, which is more problematic, an acceptance rate of 95% is used since this decreases the step length. However, this does not have to be the actual acceptance probability. For problematic distributions is might be considerable less, but anything between 20% and the target acceptance is to be considered acceptable. If the acceptance probability is low one just have to increase the sampling so that one get large enough set of points.
When one is working with a complex and problematic distribution having a non informative prior might lead to the divergence of the chains in the sampler. When this occurs it might be helpful to give a somewhat informative prior to the sampler. Since a weakly informative prior is needed for the model one might need to set a unique prior for each company in the dataset.

5.3.2 Gamma Distribution - Negative Sums

The negative sums of the data set are being modeled as a gamma distribution, the estimation is being done with the NUTS method with a uninformative prior for both α and β. The prior is set to be $Gamma(1, 1)$ for both. The model is similar to 5.3 but the model is not bimodal, the full model for the negative sums is outlined in eq. (5.4).

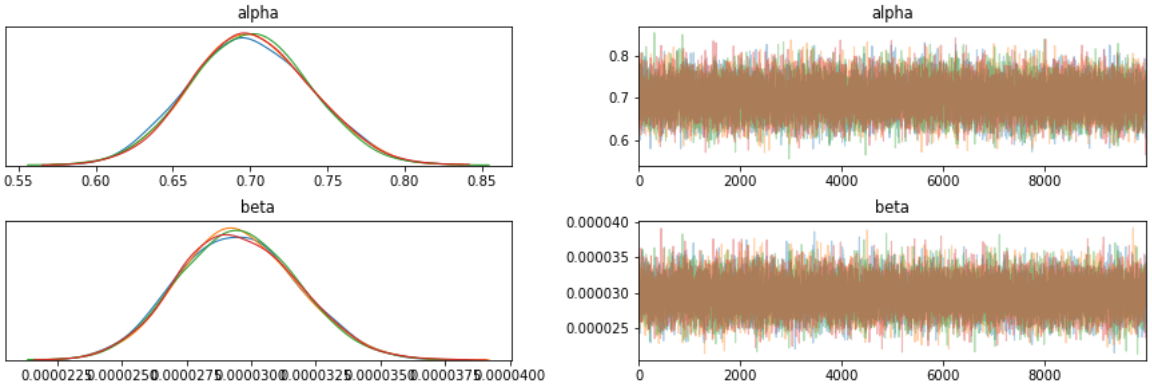


Figure 5.3: Chains for NUTS algorithm for the gamma

Variabel	Mean	SD	HPD 3%	HPD 97%
α	0.701	0.038	0.629	0.772
β	$2.95 * 10^{-5}$	0.00	0.00	0.00

Table 5.2: Statistics for the NUTS algorithm for gamma

The method does not seem to need any weakly informative prior in practice since the data is well behaved. The estimation of the parameters for the gamma distribution takes about 1 minute to estimate with the NUTS sampler. The result for one of the companies can be seen below in table 5.2. The chain convergence can be seen in fig. 5.3. The convergence indicates that we have found a good estimate of the underlying distribution.

5.3.3 Gamma Mixture - Positive Sums

During the data analysis it was shown that the positive sums had clear signs of a mixture distribution. To be able to estimate the true underlying distribution one has to make a series of assumptions. The assumptions for the positive sums are derived from the data analysis where it was deduced that the distributions seems to be bimodal and seems to follow something similar to a gamma distribution.
The first assumption that the distribution is a bimodal one can easily be changed and the gamma distributions is a very versatile one which can take many different shapes, so the assumptions are not limiting for the model. The chains for one company from the model simulation can be seen in fig. 5.4, as one can see the chains are well mixed, the numerical results are shown in table 5.3.

5.4 Non-Homogeneous Poisson Process - Transactions

The estimation of the NHPP was done using the NUTS method. This method is overly powerful for this simple problem where a simple Maximum A Posteriori (MAP) method would work just fine. The method was however already implemented

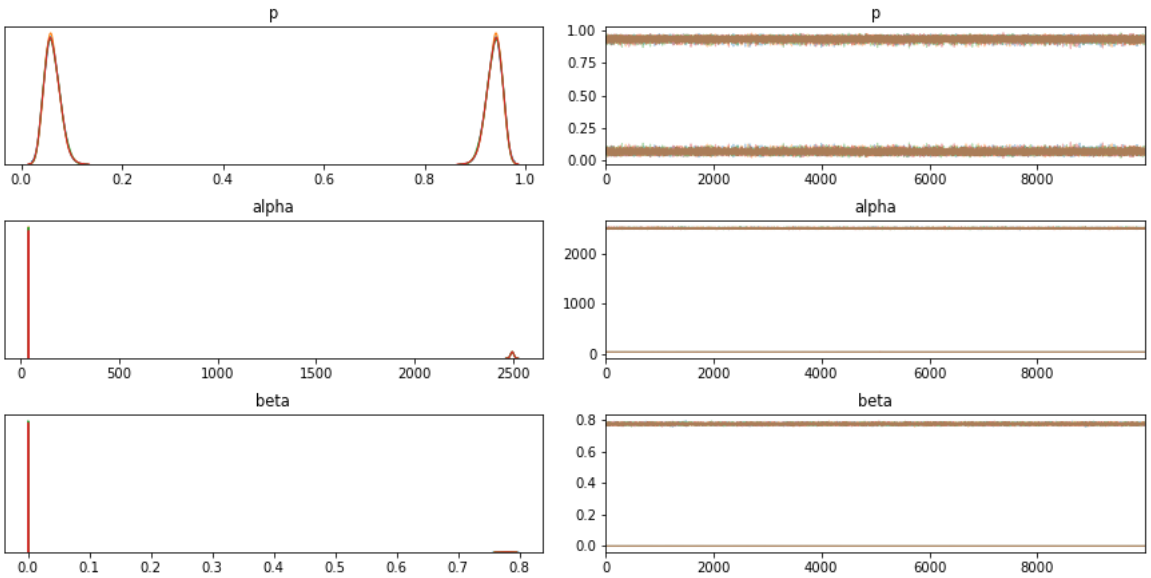


Figure 5.4: Chains for NUTS algorithm for the mixture of gammas

Variable	Mean	SD	HPD 3%	HPD 97%
q	0.94	0.02	0.91	0.07
p	0.06	0.02	0.03	0.09
α_1	38.53	0.38	37.81	39.25
α_2	2496.97	7.84	2482.09	2511.39
β_1	0.000745	0.00	0.00	0.00
β_2	0.78	0.00	0.77	0.79

Table 5.3: Statistics for the NUTS algorithm for gamma mixture

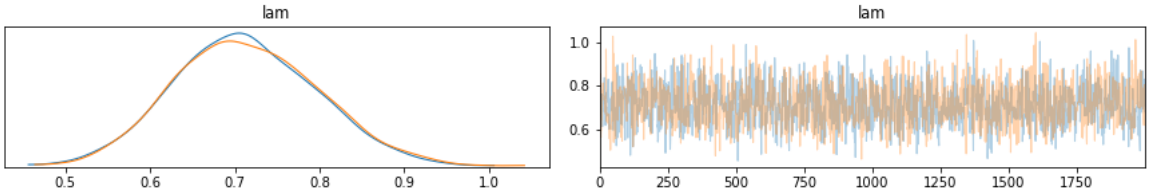


Figure 5.5: Chains for one of the rate parameter using the NUTS algorithm

Variable	Estimate
λ_{Pos1}	0.712
λ_{Pos2}	0.139
λ_{Neg1}	0.851
λ_{Neg2}	0.410

Table 5.4: Statistics for the NHPP parameters

for the other methods and was therefore reused. The NHPP was defined as two different homogeneous Poisson processes which is shifting depending on which part of the month the current day is. This partition is unique for each company and is decided by calculating the partition where 50% of the transaction are being made in each partition historically.
So the transactions of each company will be modeled as a mixture model of four different homogeneous Poisson processes, two for the positive transactions and two for the negative transactions. So these four homogeneous Poisson process make up a two non-homogeneous Poisson process. The result of the sampler for one of the processes are displayed in fig. 5.5 and the rate parameter for each of the four Poisson processes can be seen in table 5.4.
As can be seen there is a good mixture of the chains and there is significant difference between the rate parameters of both the negative and positive sums. This strengthens the view that there is a inter monthly seasonality of the transactions.

5.5 Bootstrap Particle Filter

The hidden state which will be interpreted as the financial health of the given company is estimated through a Bootstrap Particle Filter and this method allows for non-linear and non-Gaussian emissions. The particle filter is set up by using the cash flow as an observed state which is denoted as Y_t for time t. The cash flow is then assumed to follow the the model $Y_t = X_t + \mathcal{N}(0, \sigma_\epsilon^2)$ where X_t is the latent state.
The initial state X_0 is assumed to be identical as the first cash flow data point. The transition density is $\pi(X_t|X_{t-1})$ and a random sample from X_t is given from forward sampling from the model of the cash flow which were introduced above.
One assumption that is made with this model architecture is that the model is more reliable than the observations which means that it assumes that there are some significant observations errors. These observation errors can have many sources but does not need to be identified, but some potential errors is misreported numbers.

The assumptions for the bootstrap filter is very simplistic and might miss some important features. But in lack of better modelling these assumptions were made to make the model transparent.
In the bootstrap filter only the variance of the emission needs to be set. These properties can be estimated with a sample of X and a sample of Y as described in eq. (5.15).

$$Y_t - X_t \sim N(0, \sigma_\epsilon^2) \tag{5.15}$$

The obvious issue with this method is that one needs the latent state to estimate the variance , but at the same time one needs the variance to estimate the latent state. This creates a circular argument which leads to a Münchhausen trilemma. One solution to this problem is to assume a variance which has been done in this case. Another way to do this is a more robust way is to use a Gibbs sampler in the bootstrap filter to estimate the variance, this method will be discussed later on.
Since the variance cannot be estimated from the latent state it has to be assumed. As an initial choice the standard deviation of the observed data was chosen. This choice is based on the assumption that the distribution of the hidden state reassembles the distribution of the observed state. A good guess is that the observed state reassembles the hidden state and therefore one can make an argument that a good guess of the hidden states variance is the same as the observed state.
The amount of particles was set to 4000 since this creates a good trade of between robustness and run time. A general heuristic is that one should use at-least 1000 particles in a bootstrap filter. The only trade-off with choosing a larger amount of particles is that the time of computation of the particle filter increases. So if one chooses an unnecessary large amount of particles the computation will take a longer time. The needed amount of particles can however differ between different data sets, so one should monitor the variance of different estimations of the latent state. If the simulation differ from each other in a meaningful way it is an indication of high variance estimates which are discussed in [6]. If this is the case one needs to either increase the amount of particles or choose a different model.

6

Results

The results of the methods which were implemented in the previous chapter are presented in this chapter. The results consists of a cash flow simulation model and the state estimation of the financial health where the Cash Flow model is used. It should be noted that this estimation is not a prediction of any sort but is only an estimation of a latent state of an already observed time series. The results shown below is of a single company but the results are very similar throughout the data set.

6.1 Cash Flow Model

The cash flow is modeled with the methods which as introduced in *Methods*. First is the transactions per day modeled as a Non-Homogeneous Poisson Process, the transactions per day are being split by the sign of the cash flow (positive or negative) and the temporal structure is being taken into account by a partition of the month. Then is the transaction intensity modeled by two different processes.

The positive transactions are being modeled as a hierarchical Gamma mixture distribution while the negative sums are being modeled as a hierarchical gamma distribution. The cash flow model can be used as a prediction tool since the forward sampling of the model creates an array of potential outcomes, an example of this can be seen in fig. 6.1.

It should be noted that there are some problematic properties of the cash flow model, one of those things is that the model can simulate the transactions in a way that it becomes very unbalanced. An example of this is that the model does not exclude the possibility that one have three times the income but no expenses during some period of time.

Another aspect of the model which needs to be discussed is that if one would run a t-test of the final cash flow amount one would not be able to say that the final result is not significantly different from 0, this however does not mean that the model is insufficient. It can be the case that since most of the companies in the data set mostly break even and the best naive prediction of the cash flow is unchanged over the time period. The explanation for this phenomenon is that there is an incentive for the individual business owner not to make to much of a profit, or any profit at all, due to tax reasons.

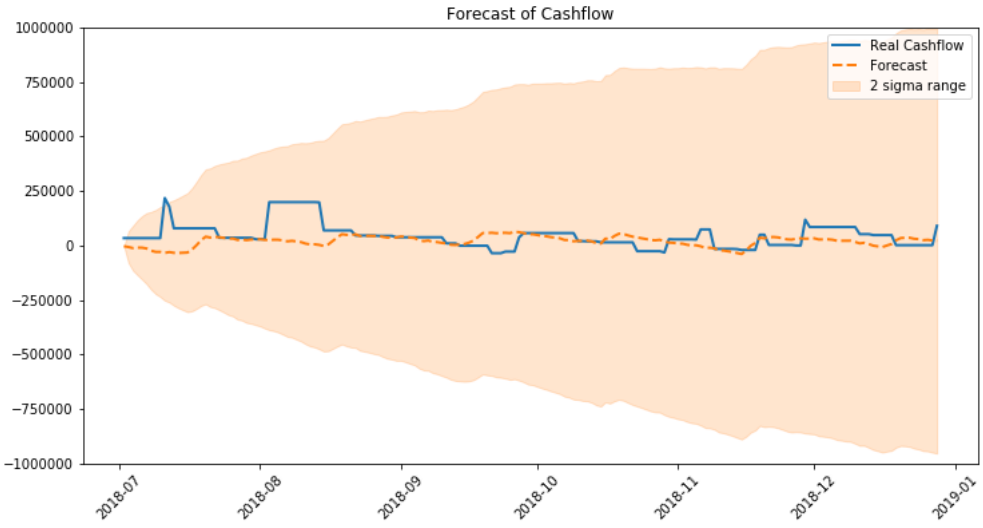


Figure 6.1: Forward sample of the Cash Flow Model in SEK

6.2 Particle Filter Implementation

In fig. 6.2 on can see the comparison of the observed cash flow and the estimates latent state estimation. As one can see the observed cash flow is much almost unchanged during the cycle while the hidden state seems to indicate a deteriorating financial health.

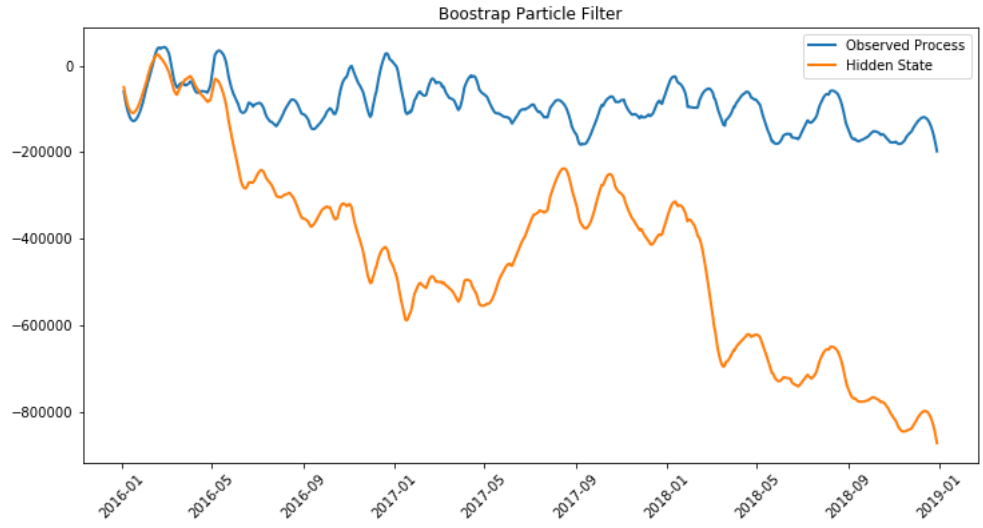


Figure 6.2: Hidden State compared to observed cash flow for an individual company

The hidden state is calculated using the Maximum A Posteriori (MAP) estimate of each parameter in the cash flow model. Then each particle in the filter is weighted by importance for each time step, where the importance is evaluated by $\pi(Y_t|X_t)$.

The result of the particle filter with a discrete state is shown in fig. 6.3. The states has been normalized and smoothed so that one can get a better interpretation of what the states are. This is done since a meaningful interpretation of the hidden state in this setting could be its derivative, i.e. is the financial health improving or deteriorating. Further, there is an indicator function showing -1 if the state is outside 1.5 standard deviation from the mean, 0 if it is within 1.5 standard deviation and +1 if it is outside 1.5 standard deviation on the positive side.
The interpretation of the particle filter could that when the hidden states are trending upwards, or are in state 1, the company is doing better give all other observations for that specific company. This is might be meaningful since the cash-flow is a very noisy process and a small different in the hidden state might only be confusing for the user.

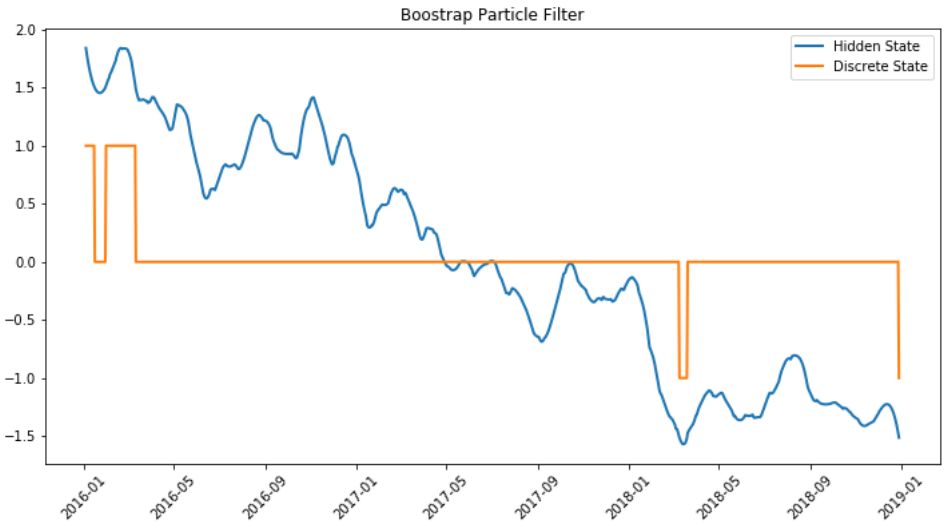


Figure 6.3: Particle Filter for an individual company

The Bootstrap Particle Filter was built using the method presented earlier in the thesis. There are a few parameters which needs to be tuned in the model and that is the starting value, the driving noise and the amount of particles. The initial hidden state is set to be equal to the first observed state and the driving noise is set to be normal with the same variance as the observed data. The variance of the latent state was estimated to be equal to the variance of the observed state. The number of particles should be set such that not just a few particles carries all weight.
The filter is unstable with a low amount of particles since this lead to high variance between the estimates, after calibrating the number of particles to 4000 the results became more stable. Each run will differ slightly from the previous one since this is a stochastic process. This simulation takes about 2 minutes to run for the whole sequence of observations.
Usually one validates a statistical model with some sort of train test split but since Hidden Markov Models is an unsupervised method, there is no "target", one cannot evaluate the model in the usual way. In this case one have to rely on the theory behind the method as use common sense to determine if the model is accurate or

not. In more detail, the Hidden Markov Model aims to estimate a hidden state, i.e. a state which is never observed , so there is no way to measure the accuracy of the hidden states. This is one of the obstacles in using this kind of model.

6.3 Comparison of Markov Model and Hidden Markov Model

Since the Hidden Markov Model is a more complex version of the already familiar Markov Model it might be the case that the Hidden Markov Model just adds unneeded complexity to the model. To investigate this one can compare the Hidden Markov Model to a Markov Model and see if there is a difference.
As one can see by allowing for a dynamic latent state one capture some dynamic of the time series which the fixed state model does not. This can be seen by the significant difference between the two models in fig. 6.4.

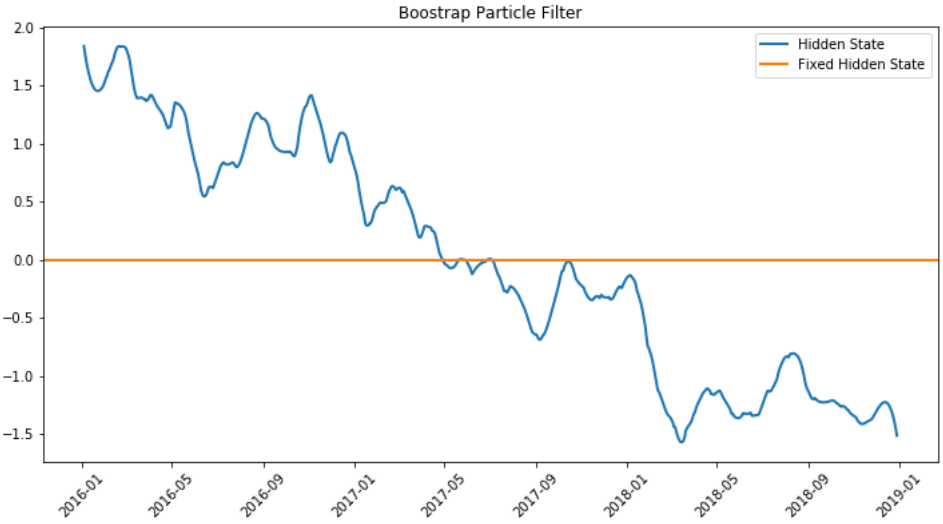


Figure 6.4: Illustration of a Fixed State Hidden Markov Model

From this comparison one can conclude that a Hidden Markov Model adds information and is able to explain more of the observed data. So therefore it can be said that there is some latent nature in the time series. One should however recall that the latent nature might not have a clear causal explanation, but rather one can just say that there is one.
One aspect of the latent structure is that it seems to be mean reverting throughout the data set. What this means is that the latent structure seems to drift towards the mean when it is either above or below the mean. If one keeps the intended interpretation that the latent state can be interpreted as some measure of financial health, then one can say that a company that is doing very well will soon be doing less well and a company that is doing poorly will soon be doing better.
There is however one flaw of the interpretation above and this is because the data has a survivor-ship bias, which was discussed as a potential flaw in the data analysis.

Since each company in that data set have not ceased their operations there is no data on any potential bankruptcies. So the mean reverting property of the financial health might only be a product of the survivor-ship bias in the data.

6.4 Potential Model Extension

It has been mentioned that this model can be seen as one iteration of a Gibbs sampler with the prior that the latent state is equal to the hidden state. Another simplification was discussed in the *Methods* chapter of the thesis, the assumptions of the model are very naive and is based on the idea that the hidden state is very similar to the observed state and that we are more confident regarding the model compared to the observed data. These simplification will limit the accuracy of the model.
These simplifications were made since one cannot estimate the hidden state transition state since it is hidden and since it is hidden the best guess one can make is that it is similar to the observed state. These assumptions would have less of a influence on the model if one would build a Gibbs Sampler which would iterate between the parameter estimation for the cash flow model and the hidden state estimation.
One could say that the current version of the model is one iteration of a Gibbs Sampler, which was introduced in *Mathematical Background*, but as with any other Gibbs sampler one needs several iterations to reach convergence. This model could be built with *PYMC3* which the current model uses.
The Gibbs Sampler could be built by first fitting the data onto the Bayesian Hierarchical Model and then use the model to estimate the hidden State through a particle filter. Then one adjusts the model by the new estimated hidden state and then run the particle filter again. By doing this procedure many times one will get a more accurate estimation.

7
Conclusion

The main goal of the thesis is to investigate if there is a way to estimate the latent state of cash flow data while taking some stylized properties of the cash flow into account. These properties were specified to be the seasonality and latent state was considered as a continues process while observing discrete cash flow transactions.
The thesis started with a data analysis that showed that there is a strong seasonality in the number of transactions while the size of the aggregated daily transactions showed no signs of any robust temporal structure. It was also assumed that the amounts of transactions and transaction amounts are independent, this is also in line with [4] where the authors built a similar model but for rain fall data. The positive sums and the negative sums have very different distributions which was taken into account.
The model was built by first modelling the cash flow with a Compound Poisson Process where Non-homogeneous Poisson Processes, Hierarchical Bayesian Modelling and Markov Chain Monte Carlo Methods were used. This model was then used as a forward sampler in a Bootstrap Particle Filter which then estimated the latent state.

7.1 Discussion

The cash flow model was built as a Poisson Compound Process where positive and negative sums were split due to the difference between the distributions. This way of building the model is similar to the model in [4] but built using the Bayesian paradigm. The cash flow model showed signs of high variance which might indicate of a poorly fitted model but could also reflect the fundamental uncertainty that small businesses face in the economy. However, there are properties of the data which the model does not take into account, such as it is not in the business operators interest to make excessive profit, due to tax reasons, but this feature could be added in a future model.
The Particle Filter implementation shows signs of giving high variance estimates which means that the Particle Filter might suffer from degeneracy. This was discussed in [6] and is a well known phenomenon with particle filters. This degeneracy problem needs to be supervised in future versions of the model.
The results shows that using a Hidden Markov Model makes it possible to find the signal in the noise for financial data which is discrete and have some deterministic seasonality. This method might be useful since financial data is notoriously noisy and many other statistical methods are not fit for this.

Another result was that there is a latent state which dictates the observations. This should not come as a surprise since the economy is a complex adaptive system and have a clear causal effect on a company's performance. However, the causal effects can be intricate and hard to define. What the proposed model makes possible is to define the magnitude of this latent state and how it changes over time, without needing an explicit understanding of the latent state.
There are currently two drawbacks in the implementation of the methods proposed in the model. The first drawback is the assumption of the magnitude of noise in the hidden state, to make it more accurate one would have to come up with a solution to this problem. The other drawback is that the model's run time which from data processing to particle filter takes about 7 minutes. This is done on CPU on a regular laptop so the run time will decrease if one runs it on a Virtual Machine with a large GPU.
One could mend the first issue with alternative implementations. An example could be to implement a Gibbs Sampler which could estimate the the hidden state and its properties in a better way, this was discussed breifly in *Results*.
For the results to be more conclusive one would also need an unbiased data set that includes companies that have gone bankrupt. Without this information one will always have a biased model which would present the possibility for the company to go out of business. One would also need longer time frames for the company data to catch monthly seasonality and yearly seasonality. These seasonal patterns cannot be estimated with a low amount of data but these patterns surely exists and knowing them would improve the model.

7.2 Applications

The application for the model could be to estimate how well a company is doing and use the method as an indicator if any action should be taken by the company itself or from a third party such as a lending institute. The model could also be used by the company itself which would makes it possible to take more intelligent business decisions since it provides insights of the well-being of the company.
The results of this thesis could be used for any kind of hidden state estimation for noisy time series. Another application could be to see if a financial market have changed latent state. Both these applications could improve risk management for the fields.
It could also be used in Bioinformatics to estimate latent states of highly dynamic systems, such as the human body. Combined with the suggested improvements the model could be proven to be powerful and solve issues with time series without any strict assumptions of linearity or Gaussianity which often haunt time series models.

7.3 Future Studies

Some further studies which might be interesting to conduct is if there is a way to implement Markov Chain Monte Carlo Methods such that the programs could be run faster. Ideas such as Parallel Programming, Threading and optimizing code

could be investigated. This would be within the field High Performance Computing and would make the MCMC methods more available for the industry. This would be beneficial since MCMC methods offers some unique features which should be useful for field such as quantitative finance and bioinformatics.
Some other studies which would be useful for the field of Bayesian Statistics is the development of methods to set vaguely informative priors systematically for complicated distributions. A systematic way to set priors would make Markov Chain Monte Carlo Methods more approachable by practitioners.

Bibliography

[1] Umberto Picchini. *Sequential Monte Carlo and the Bootstrap filter*. https://umbertopicchini.wordpress.com/2016/10/19/sequential-monte-carlo-bootstrap-filter/.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Shota Gugushvili, Frank van der Meulen, Moritz Schauer, and Peter Spreij. Nonparametric Bayesian Volatility Learning Under Microstructure Noise. arXiv 1805.05606, 2018.

[4] Nelson Dzupire and Philip Ngare. A Poisson-Gamma Model for Zero Inflated Rainfall Data. *Journal of Probability and Statistics*, 2018, 04 2018.

[5] Shota Gugushvili, Frank van der Meulen, Moritz Schauer, and Peter Spreij. Fast and Scalable Non-Parametric Bayesian Inference for Poisson Point Processes. *Researchers One*, 2019. `https://www.researchers.one/article/2019-06-6`.

[6] Nikolas Kantas, Arnaud Doucet, Sumeetpal S. Singh, Jan Maciejowski, and Nicolas Chopin. On Particle Methods for Parameter Estimation in State-Space Models. *Statist. Sci.*, 30(3):328–351, 08 2015.

[7] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting, 2nd Edition*. Springer New York, 2002.

[8] John A. Rice. *Mathematical Statistics and Data Analysis.* Belmont, CA: Duxbury Press., third edition, 2006.

[9] Geoffrey R. Grimmett and David Stirzaker. *Probability and Random Processes (Vol. 80)*. Oxford University Press, 2001.

[10] Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. arXiv 1111.4246, 2011. `https://arxiv.org/pdf/1111.4246.pdf`.

[11] Fonnesbeck C. Salvatier J, Wiecki TV. Probabilistic Programming in Python using PyMC3. 2016. `https://arxiv.org/pdf/1507.08050.pdf`.

[12] Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic Variational Inference in Stan. 2015. `https://arxiv.org/abs/1506.03431`.

[13] hmmlearn: Unsupervised Learning and Inference of Hidden Markov Models. `https://hmmlearn.readthedocs.io/en/latest/index.html`.

[14] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[15] Hans R. Künsch. Particle filters. *Bernoulli*, 19(4):1391–1403, 09 2013. `https://doi.org/10.3150/12-BEJSP07`.

[16] A.F.M. Smith N.J. Gordon, D.J. Salmond. Novel Approach to Nonlinear/non-Gaussian Bayesian State Estimation. *Radar and Signal Processing, IEE Proceedings F*, 1993.