Delft University of Technology
Faculty of Electrical Engineering, Mathematics en Coputer Science
Delft Institute of Applied Mathematics

Financial market crashes: Predicting bubbles using the Johansen-Ledoit-Sornette model.

Thesis submitted to the
Delft Institute of Applied Mathematics
as part of obtaining

the degree of

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

D.A.W. van Lange

Delft, The Netherlands
August 2018

BSc report APPLIED MATHEMATICS

"Financial market crashes: Predicting bubbles using the Johansen-Ledoit-Sornette model."

Dion Alex Willem van Lange

Delft University of Technology

Supervisor

Dr. M.T. Joosten

Other thesis committee members

Dr. J.G. Spandaw

Dr. Juan-Juan Cai

August, 2018

Delft

Abstract

In this project we looked into financial markets. The goal of this project was to find out if a bubble is forming and when the most probable time of bursting would be. that is what we call the critical time. In order to do this we studied the work of Professor Didier Sornette, who is an expert in this field of mathematics. In this bachelor thesis we use the Johansen-Ledoit-Sornette (JLS) model and the Levenberg-Marquardt algorithm to predict the critical time of bubbles. By critical time we mean the most probable time of a bubble to burst, but not for certain: there is always a probability to attain the end of the bubble without bursting.
We looked at the results Didier Sornette got in his work on Black Monday and tried to obtain the same results. Besides that we investigated the sensitivity of the JLS model and differences between variants of it, also when simulating our own data. In the end we looked at applying the model at real data. For the data we chose the Amsterdam Exchange Index and Bitcoin.

Preface

What you are reading is the thesis 'Financial market crashes: Predicting bubbles using the Johansen-Ledoit-Sornette model.', written in order to obtain the degree of Bachelor of Science from the Delft Institute of Applied Mathematics. This project took place in the department of statistics under the supervision of M.T. Joosten.

From May till August I spent a lot of time researching the mathematics of modelling bubbles in financial markets and working in R to simulate my models. Not everything I researched turned out the way I hoped, and had I had more time I would have loved to spend some more time researching and simulating. But after all, everything comes to an end.

I would like to thank M.T. Joosten for his guidance and help during the project and Dr. J.G. Spandaw and Dr. Juan-Juan Cai for taking part in my assessment committee.

Dion van Lange
Delft, August 2018

Contents

1 Introduction

Financial crashes have always been events with a lot of consequences. The direct result is a great loss in paper wealth which causes a big loss in wealth in general. Consequences like bankruptcy of companies, people losing their jobs and a loss of trust in each other and in the economy are some examples. One of the most famous financial bubbles is the tulip mania, since this is one of the first recorded speculative bubbles. The tulip mania happened in the Netherlands during the seventeenth century. Tulip bulbs were very rare in the Netherlands during the seventeenth century, since tulip bulbs are originally from Turkey. Tulip bulbs were considered very valuable, because of the beautiful colours they would show when fully grown. Because of this, people considered having tulips as something prestigious. Tulip bulbs were very rare at that time. The new desire and rarity of tulip bulbs caused the prices of the bulbs to rise very rapidly in a short period of time. Some very rare bulbs were even traded for more than thousand Dutch guilders. The Dutch guilder was the currency at that time. One Dutch guilder is approximately 0.45 euros, but because of inflation a Dutch guilder was worth more than 0.45 euros at that time. So people payed a ridiculous amount of money for the bulbs. As we said it was one of the first bubbles, the price of the tulip bulbs decreased in price even faster that it had risen. The tulip bulb you bought yesterday for one hundred guilders had a value of less than one guilder at the day of the crash. This example of a bubble is used as an example for other financial bubbles a lot. Today the tulip mania is compared frequently to cryptocurrencies like Bitcoin for example. Since cryptocurrencies have been gaining a lot of value in a small period of time.
With the upcoming technologies like blockchain the upcoming cryptocurrencies are very innovative and may have a big impact on the way of living. Cryptocurrencies have been a hot topic for quite a while. In this project we will only look at the biggest cryptocurrency right now: Bitcoin. Bitcoin has been in the news quite a lot lately, always asking the same question: Will Bitcoin continue growing in price? In this bachelor thesis we will try to answer this question by modeling Bitcoin as a bubble.
So one can wonder if we can predict financial bubbles. Because if we can, we might be able to stop them from bursting. That would be wonderful, because of all the negative consequences that would be avoided. A lot of mathematicians tried to model financial bubbles, but this appeared to be very hard. Mostly because every piece of information influences the actions of others. For example: If the headline in the newspapers says that tomorrow Bitcoin will be half of the value it is now, people will read it and often decide to sell of their Bitcoins. These interdependencies are making it very hard to model financial bubbles. The main question we will try to answer in this thesis is: "Is it possible to model a bubble and predict what the most probable time of bursting is when using the Johansen-Ledoit-Sornette model?"
In chapter two we will derive and explain the Johansen-Ledoit-Sornette model. We will also apply it at a famous marketer crash: Black Monday. In the third chapter we will look further in the sensitivity of the model. The fourth chapter describes the application of the model at the Amsterdam Exchange Index and Bitcoin.

2

The model

In this chapter we will look into the Johansen-Ledoit-Sornette model, which we use to predict bubbles. First we will look into assumptions, explanation and the derivation. After that we will apply our model on a well known crash: Black Monday.

2.1. Explanation and derivation

A financial crash is not certain: You never know that it will happen for sure, but there is always a chance of happening. To model this uncertainty, we introduce the hazard rate $h(t)$, which is the probability per unit time that the crash will happen in the next instant if it has not happened yet. A crash happens when a large group of people decide to place sell orders in the market simultaneously. Because of this large amount of sell orders, there will be an imbalance in the buy and sell orders, which will cause the price to drop. If this amount of sell orders is large enough to have a decrease of more than 20% we speak of a crash. The definition of a crash depends on the literature you are reading, but what is used usually is a drop of 20%-30% in price. In our case we will say a drop of more than 20% is a crash. The question remains however why, when and how does a large amount of sell orders occur simultaneously. People don't plan with each other to sell at the same time, because they think that would be fun. No, they all think at the same time that there is an overvaluation. But when do they think that there is an overvaluation and decide to sell off? When will there be an amount of sellers large enough to make the buy and sell orders go off balance? These questions are contained in the hazard rate.

In "mean field" theory of collective systems (see article [2]), this is the simplest way to describe imitation process:

$$\frac{dh}{dt} = Ch^{\delta} \tag{2.1}$$

with $\delta > 1$ and C a positive constant. δ represents the average interaction between traders minus one. We can see $h(t)$ as the collective result of the interactions between traders.

We can obtain the solution for the hazard rate $h(t)$ by integrating:

$$\begin{aligned}
\frac{dh}{dt} = Ch^{\delta} &\Rightarrow \frac{1}{h^{\delta}}\frac{dh}{dt} = C \\
&\Rightarrow \int \frac{1}{h^{\delta}}\frac{dh}{dt}dt = \int C dt \\
&\Rightarrow \int h^{-\delta} dh = Ct + K \\
&\Rightarrow \frac{1}{-\delta+1}h^{-\delta+1} = Ct + K \\
&\Rightarrow h^{-\delta+1} = (Ct+K)(-\delta+1) \\
&\Rightarrow h(t) = ((Ct+K)(-\delta+1))^{\frac{1}{-\delta+1}} \\
&\Rightarrow h(t) = \Big(\frac{1}{(Ct+K)(-\delta+1)}\Big)^{\frac{1}{\delta-1}} = \Big(\frac{1}{(\frac{K}{C}+t)(C-\delta C)}\Big)^{\frac{1}{\delta-1}}
\end{aligned}$$

Define $t_c := -\frac{K}{C}$. Notice that $t_c > 0$, since C is a positive constant and $K < 0$. $K < 0$ we obtain from $h^{-\delta+1} = (Ct+K)(-\delta+1)$ with $t = 0$, $\delta > 1$ and $h(t) > 0$. Then follows:

$$h(t) = (\frac{1}{(-t_c+t)(C-\delta C)})^{\frac{1}{\delta-1}} = (\frac{1}{(t_c-t)(\delta C-C)})^{\frac{1}{\delta-1}}$$
$$= \frac{B}{(t_c-t)^{\frac{1}{\delta-1}}} = \frac{B}{(t_c-t)^{\alpha}}$$

So we find

$$h(t) = \frac{B}{(t_c-t)^{\alpha}}$$

with $\alpha = \frac{1}{\delta-1}$ and $B = (C\delta - C)^{\frac{1}{-\delta+1}}$ constant. Notice that $\alpha < 1$ to make sure the hazard rate $h(t)$ does not diverge at t_c. Rewriting this condition for δ gives us $2 < \delta < \infty$. This means that every agent is connected to at least 2 other agents.
Assume that during a crash the price drops by a fixed percentage $k \in (0,1)$, say between 20% and 30% increase of the price above a reference value p_1. Then the dynamics of the asset price are given by:

$$\frac{dp}{dt} = \mu(t)p(t) - \kappa[p(t)-p_1]\frac{dj}{dt} \tag{2.2}$$

On the left we have the change in price. This is equal to the return $\mu(t)$ times the price at that time. Here j denotes a jump process whose value is zero before the crash and one afterwards. This means that $\frac{dj}{dt}$ is always zero except for the exact moment of the crash happening. At that point $\frac{dj}{dt}$ is equal to one. At that moment you will have an instant drop of $\kappa[p(t)-p_1]$ in value. Notice that this model is very simplified. We neglect for example: interest rate, risk aversion, information asymmetry, and the market-clearing condition. We also assume that the martingale property holds: traders do their best and price the asset fair:

$$\forall t' > t: \quad E_t[p(t')] = p(t) \tag{2.3}$$

In this equation E_t denotes the conditional expectation where everything up to time t is known already. In the same way P_t denotes the conditional probability where everything up to time t is known already. Note that $\forall t' > t$ we have $E_t[dp] = E_t[p(t') - p(t)] = E_t[p(t')] - E[p(t)] = p(t) - p(t) = 0$. Putting (2.2) into (2.3) and multiplying both sides by dt gives:

$$\begin{aligned}
0 = E_t[dp] &= E_t[\mu(t)p(t)dt - \kappa[p(t)-p_1]dj] \\
&= \mu(t)p(t)dt - \kappa[p(t)-p_1]E_t[dj] \\
&= \mu(t)p(t)dt - \kappa[p(t)-p_1](P_t(dj=0)\cdot(dj=0) + P_t(dj=1)\cdot(dj=1)) \\
&= \mu(t)p(t)dt - \kappa[p(t)-p_1](0 + h(t)dt) \\
&= \mu(t)p(t)dt - \kappa[p(t)-p_1]h(t)dt
\end{aligned}$$

Rewriting this gives us:

$$\mu(t)p(t) = \kappa[p(t)-p_1]h(t) \tag{2.4}$$

So we find that $\mu(t)p(t) = \kappa[p(t)-p_1]h(t)$. In words this means that if the crash hazard rate $h(t)$ increases the return μ increases as well. At first this seems a bit contradictory. However the return μ increases to compensate the traders for the increasing risk they take. So after giving it some second thoughts it is understandable. Plugging (2.4) into (2.2) gives us an ordinary differential equation:

$$\frac{dp}{dt} = \kappa[p(t)-p_1]h(t) - (k[p(t)-p_1]\frac{dj}{dt}) = \kappa[p(t)-p_1]h(t) \tag{2.5}$$

Since j is a jump process we have that j is a constant function $\forall t < t_c$. So before the crash we have $\frac{dj}{dt} = 0$. So before the crash we find that $\frac{dp}{dt} = \kappa[p(t)-p_1]h(t)$ holds. This gives us the following solution for $p(t) - p(t_0) < p(t_0) - p_1$:

$$p(t) \approx p(t_0) + \kappa[p(t_0)-p_1]\int_{t_0}^{t} h(t')dt' \tag{2.6}$$

Plugging $h(t) = \frac{B}{(t_c-t)^\alpha}$ with $\alpha = \frac{1}{\delta-1}$ into (2.6) gives the following price law:

$$\begin{aligned} p(t) &\approx p(t_0) + \kappa[p(t_0) - p_1] \int_{t_0}^{t} \frac{B}{(t_c - t')^\alpha} dt' \\ &\approx p(t_0) + \kappa(p(t_0) - p_1)\Big[\frac{B}{1-\alpha}(t_c - t')^{1-\alpha}\Big]_{t_0}^{t} \\ &\approx p(t_0) + \kappa(p(t_0) - p_1)\Big[\frac{B}{1-\alpha}(t_c - t)^{1-\alpha} - \frac{B}{1-\alpha}(t_c - t_0)^{1-\alpha}\Big] \end{aligned}$$

Notice that $\kappa(p(t_0) - p_1)\Big[\frac{B}{1-\alpha}(t_c - t_0)^{1-\alpha}\Big]$ is constant, so we can define the constant p_c in this way:

$$p_c = p(t_0) - \kappa(p(t_0) - p_1)\Big[\frac{B}{1-\alpha}(t_c - t_0)^{1-\alpha}\Big]$$

When we let $B^* = B((p(t_0) - p_1))$ we obtain:

$$p(t) \approx p_c - \frac{\kappa B^*}{z} \times (t_c - t)^z \tag{2.7}$$

Where $z = 1 - \alpha \in (0,1)$. Note that p_c is the price at the critical time conditioned on no other crash having been triggered.

If we let α be a complex number and use first order expansion of our hazard rate becomes:

$$h(t) \approx B_0(t_0 - t)^{-\alpha} + B_1(t_c - t)^{-\alpha} cos[\omega log(t_c - t) - \psi] \tag{2.8}$$

In this equation ψ is a phase constant. Notice that the hazard rate explodes near the critical date. However we will not use (2.8) for predicting the value of t_c. Because we have very few to no information about the hazard rate. We do have information about the price of the financial product. Equation (2.9) is also obtained with α a complex number in (2.7) and using the first order expansion.

$$p(t) \approx p_c - \frac{k}{z}[B_0(t_c - t)^z + B_1(t_c - t)^z cos[\omega log(t_c - t) - \phi]] \tag{2.9}$$

ϕ is another phase constant. We will use equation (2.9) to predict the critical time t_c. Mostly because we do have data about the price in the passed time. We will rewrite this equation, because it is more convenient for predicting the critical time t_c.
Instead of using formula (2.9) we can rewrite the equation in this way:

$$\begin{aligned} p(t) &\approx p_c - \frac{k}{z}\Big[B_0(t_c - t)^z + B_1(t_c - t)^z \cos\big[\omega log(t_c - t) - \phi\big]\Big] \\ &\approx p_c - \frac{kB_0}{z}\Big[(t_c - t)^z + \frac{B_1}{B_0}(t_c - t)^z \cos\big[\omega log(t_c - t) - log(e^\phi)\big]\Big] \\ &\approx p_c - \frac{kB_0}{z}(t_c - t)^z\Big[1 + \frac{B_1}{B_0}\cos\big[\omega log(\frac{t_c - t}{e^\phi})\big]\Big] \end{aligned}$$

When we redefine the parameters we get this equation:

$$p(t) \approx A_2 + B_2\Big[(t_c - t)^{m_2}\Big[1 + C\cos\Big(\omega log\Big(\frac{t_c - t}{T}\Big)\Big)\Big]\Big] \tag{2.10}$$

Notice that formula (2.10) has one less parameter than formula (2.9) has. This enhances the performance of the Levenberg-Marquardt algorithm. Notice that this equation has four non-linear parameters: t_c, m_2, ω and T. We can rewrite formula 2.10 using the difference formula for the cosine to eliminate one non-linear parameter:

$$\begin{aligned} p(t) &= A + B(t_c - t)^m + (t_c - t)^m C\cos(\omega\log((t_c - t)) - \phi) \\ &= A + B(t_c - t)^m + C(t_c - t)^m\cos(\omega\log((t_c - t)))\cos(\phi) + C(t_c - t)^m\sin(\omega\log((t_c - t)))\sin(\phi) \end{aligned}$$

Let $C_1 = C\cos(\phi)$ and $C_2 = C\sin(\phi)$. Then we obtain this formula for our new model:

$$p(t) = A + B(t_c - t)^m + C_1(t_c - t)^m\cos(\omega\log(t_c - t)) + C_2((t_c - t))^m\sin(\omega\log(t_c - t)) \tag{2.11}$$

The only non-linear parameters that remain are t_c, m and ω. This makes the performance of the Levenberg-Marquardt algorithm better, since the dimensionality of the non-linear problem is reduced from a four dimensional space to a three dimensional space. This decreases the complexity of the problem.

If the price drops by a fixed percentage $\kappa \in (0,1)$ of the price without a reference value p_1, the dynamics of the crash are given by

$$dp = \mu(t)p(t)dt - \kappa p(t)dj \tag{2.12}$$

We then get

$$E_t[dp] = \mu(t)p(t)dt - \kappa p(t)h(t)dt = 0$$

which yields $\mu(t) = \kappa h(t)$. Equation (2.12) also gives us:

$$\begin{aligned}
&dp = \kappa h(t)p(t)dt - \kappa p(t)dj \\
\Rightarrow &\frac{dp}{dt} = \kappa p(t)(h(t) - \frac{dj}{dt}) \quad \text{Notice that } \frac{dj}{dt} = 0. \\
\Rightarrow &\int_{t_0}^{t} \frac{1}{p(t)} dp = \kappa \int_{t_0}^{t} h(t')dt' \quad \text{Which gives the result.} \\
\Rightarrow &log[p(t)] - log[p(t_0)] = \kappa \int_{t_0}^{t} h(t')dt'
\end{aligned}$$

So we find:

$$log[p(t)] = log[p(t_0)] + \kappa \int_{t_0}^{t} h(t')dt' \tag{2.13}$$

Notice that (2.13) is almost the same equation as (2.6). The biggest difference is that we use the logarithm of the price instead of the price. In the same way as we derived (2.11) from (2.6), we can derive (2.16) from (2.13):

$$\begin{aligned}
log[p(t)] &\approx log[p(t_0)] + \kappa \int_{t_0}^{t} \frac{B}{(t_c - t')^{\alpha}} dt' \\
&\approx log[p(t_0)] + \kappa \Big[\frac{B}{1-\alpha} (t_c - t')^{1-\alpha} \Big]_{t_0}^{t} \\
&\approx log[p(t_0)] + \kappa \Big[\frac{B}{1-\alpha} (t_c - t)^{1-\alpha} - \frac{B}{1-\alpha} (t_c - t_0)^{1-\alpha} \Big]
\end{aligned}$$

Let $p_c = log[p(t_0)] - \kappa \frac{B}{1-\alpha}(t_c - t_0)^{1-\alpha}$ and $z = 1 - \alpha$, then:

$$log[p(t)] \approx p_c - \frac{\kappa B}{z} \times (t_c - t)^z \tag{2.14}$$

When we let α be a complex number the first order expansion becomes (see article [3]):

$$log[p(t)] \approx p_c - \frac{k}{z}[B_0(t_c - t)^z + B_1(t_c - t)^z cos[\omega log(t_c - t) - \phi]] \tag{2.15}$$

To reduce the complexity of the problem we can again rewrite formula 2.15 using the difference formula for the cosine:

$$\begin{aligned}
log[p(t)] &= A + B(t_c - t)^m + (t_c - t)^m C\cos(\omega \log((t_c - t)) - \phi) \\
&= A + B(t_c - t)^m + C(t_c - t)^m \cos(\omega \log((t_c - t)))\cos(\phi) + C(t_c - t)^m \sin(\omega \log((t_c - t)))\sin(\phi)
\end{aligned}$$

Let $C_1 = C\cos(\phi)$ and $C_2 = C\sin(\phi)$. Then we obtain formula 2.16.

$$log(p(t)) = A + B(t_c - t)^m + C_1(t_c - t)^m \cos(\omega \log(t_c - t)) + C_2((t_c - t))^m \sin(\omega \log(t_c - t)) \tag{2.16}$$

Sometimes it is preferable to use the logarithm of the price. Often when the price changes are not proportional to the price you should fit the log-price instead of the price (see article [1]). In this thesis we will always fit the price and not the log-price with one exception: We will apply model (2.16) only in chapter four on Bitcoin data.

2.2. Levenberg-Marquardt algorithm

The Levenberg-Marquardt algorithm is the algorithm I used in all of my models to estimate the parameters, most importantly the critical time. That is why I will give a brief explanation of how it works.
Like many other algorithms, the Levenberg-Marquardt algorithm uses the least square error method to find the parameters which fits best to the the data given (see source [8]). So let (x_i, y_i) be a dataset of n data points with $1 < i < n$. Let β be a vector in which all the parameters of the model are stored. Let f be the function trying to fit the y_i dependent of the parameters of β and x_i. Then the Levenberg-Marquardt algorithm gives us the β for which the residual sum of squares (2.17) is minimized

$$S(\beta) = \sum_{i=1}^{n} [y_i - f(x_i, \beta)]^2 \tag{2.17}$$

The Levenberg-Marquardt algorithm needs a first guess for β. From there on it takes iterative steps to estimate the beta for which the (2.17) is minimized. If there is only one minimum, the first guess for β does not really matter. But if you have multiple minimums, the first guess does matter (This almost always the case). Because in the case with multiple minimums the Levenberg-Marquardt algorithm probably converges to the minimum closest by: It converges to the nearest local minimum. However that is not what we want, because we want the best prediction for the critical time, so we want the global minimum: the β for which (2.17) is minimized. This means that when we get a prediction we have to check if it really is the global minimum. To do this we will run our model multiple times to conduct more results and then we can pick the best fit.
We already have data about the prices of the past time. We use this data to get the best fit and prediction for t_c. That's where we apply the Levenberg-Marquardt algorithm. It gives us estimations for all parameters of our model, which includes most interesting parameter: The critical time t_c.

2.3. Black Monday

In the financial sector Black Monday refers to the nineteenth of October 1987 (see source [10]). This day was one of the worst days in the history of financial markets all around the world world. Almost all stock markets crashed. In New Zealand for example this day is known as Black Tuesday instead of Black Monday, since it was already Tuesday in New Zealand. As outlined earlier, a lot of markets crashed at Black Monday, however in this paragraph we will restrict our view to the Standard & Poor's 500 (S&P 500). The S&P 500 is an American stock market index, which lists all the big companies. In figure 2.1 we can see how big the crash of Black Monday was.

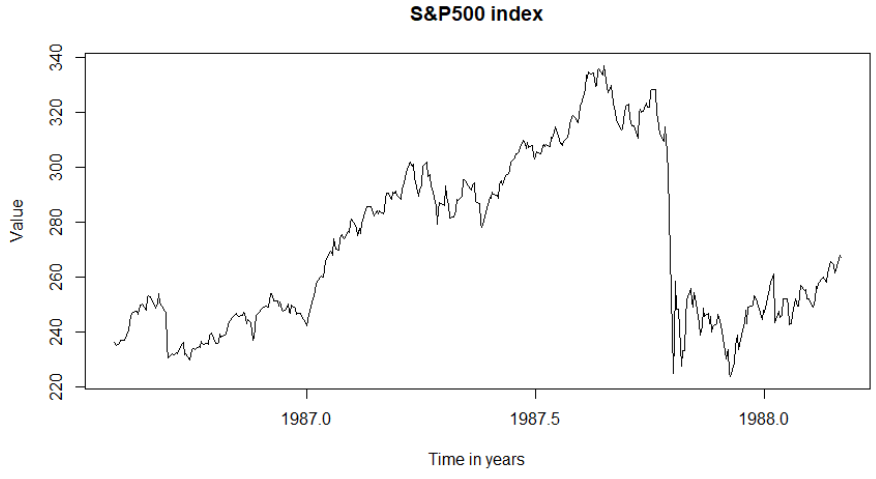

Figure 2.1: Black Monday at the S&P 500 index. The nineteenth of October 1987 was a big loss in value as we see in the graph. Data is downloaded from yahoo finance [6].

As we can see in figure 2.1 this was a huge drop in value of the S&P 500 index. In just a few days the index dropped from 320 to 220, a drop of more than 30%. This was very exceptional and no one could provide a clear answer why this happened. A lot of people blamed the new upcoming computer trading, however also

other markets crashed, which did not use computer trading yet. So this is not a likely reason.
This is a perfect example of a bubble that we can use to find out if our model works on this data.

Parameter	Estimate	standard error	t-value	P-value
A_2	457.9	14.07	32.53	$< 2e-16$
B_2	-190.3	13.19	-14.43	$< 2e-16$
t_c	88.05	0.02574	3420.29	$< 2e-16$
m	0.3624	0.02105	17.21	$< 2e-16$
C	-0.04628	0.00381	-13.69	$< 2e-16$
ω	9.251	0.2298	40.26	$< 2e-16$
T	1.673	0.03004	55.68	$< 2e-16$

Table 2.1: Estimations of the parameters we obtain when we run our model on the data of Black Monday from the S&P 500. The starting time of the data is 85.0 (1985.0) and the end time is 87.76 (1987.76)

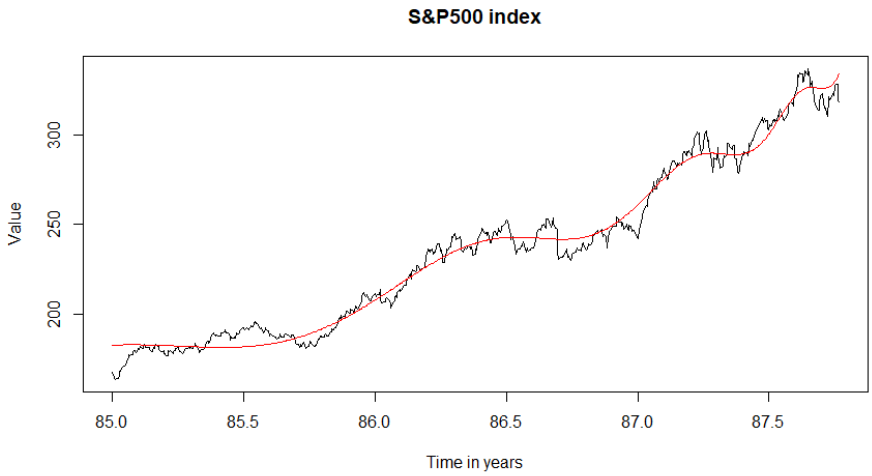

Figure 2.2: The red line is our fit on the data with the estimations of the parameters shown in table 2.1. The starting time of the data is 85.0 (1985.0) and the end time is 87.76 (1987.76)

The red line in figure 2.2 is our fit to the data and we can see that it fits the data quite well. All of the parameters are converging well, because the p-value is very small. The conclusion is that the most probable time (critical time) of the crash happening would be at 1988.05. This is 3 months later than the real bursting of the bubble, which is quite a lot. Keep in mind that the critical time is the time where the probability of the bubble bursting is highest, but it can always be earlier or later. When we vary our starting time of the data and the end time of the data we get different estimations for the critical time. Nonetheless do most of these estimations give an estimation of 88.05 for the critical time.
One of the reasons why we applied our model on this data specifically is the fact that Didier Sornette used model (2.10) as well to get estimations for Black Monday on the S&P 500 in article [2]. As starting time of the data he used 85.5 (1985.5) and as ending time of the data he used 87.6 (1987.6). We will use the same starting time and end time of the data to find out if we can obtain the same results as Didier Sornette did in [2]. Notice that the time span of the data he used is approximately two years. We will use this time span of two years frequently in this bachelor thesis.

Parameter	Estimate	standard error	t-value	P-value
A_2	909.2	216.1	4.208	$3.03 * 10^{-5}$
B_2	-635.1	214.4	-2.962	0.00319
t_c	88.14	0.03396	2595.41	$< 2e-16$
m	0.1219	0.03787	3.219	0.00136
C	0.01851	0.005921	3.126	0.00187
ω	10.11	0.2901	34.848	$< 2e-16$
T	2.388	0.03079	77.546	$< 2e-16$

Table 2.2: Estimations of the parameters we obtain when we run our model on the data of Black Monday from the S&P 500. The starting time of the data is 85.5 (1985.5) and the end time is 87.6 (1987.6)

We find that our estimations in table 2.2 are less trustworthy than our estimations in table 2.1, because the p-values are way higher. Unfortunately we find that out estimations of the parameters are not the same as Didier Sornette's estimations. In article [2] Didier Sornette found as estimations: $A_2 \approx 412$, $B_2 \approx -165$, $t_c \approx 87.65$, $m \approx 0.33$, $C \approx 12$, $\omega \approx 7.4$ and $T \approx 2.0$. When we compare our estimations we find that most of our estimations are close to Didier Sornette's estimations, except one: C. We got an estimation of 0.01851 while Didier Sornette has an estimation of 12. This is very strange, since this is a very significant difference in estimation. When we take a closer look at Didier Sornette's estimations we find that if we use the estimations in formula (2.10). We get strange values for the price:

$$p(t) \approx 412 - 165\Big[(87.74-t)^{0.33}\Big[1+12\cos\Big(7.4log\Big(\frac{87.74-t}{2}\Big)\Big)\Big]\Big]$$
$$\Rightarrow p(85.5) \approx 412 - 165\Big[(87.74-85.5)^{0.33}\Big[1+12\cos\Big(7.4log\Big(\frac{87.74-85.5}{2}\Big)\Big)\Big]\Big] = -1530.48$$

How can the value of the S&P 500 be negative? That is not possible, so there is reason to believe that Didier Sornette made a writing mistake in [2].

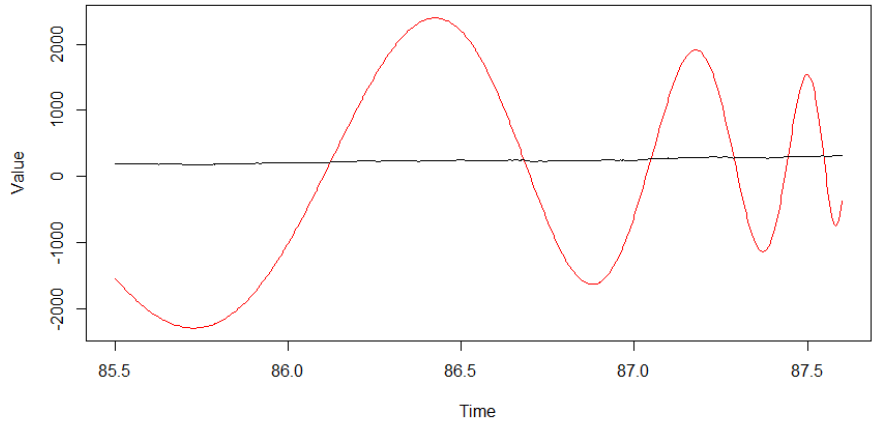

Figure 2.3: The black line in the graph is the data from S&p 500. the red line is the fit of model 2.10 with parameters $A_2 = 412$, $B_2 = -165$, $t_c = 87.65$, $m = 0.33$, $C = 12$, $\omega = 7.4$ and $T = 2.0$ The starting time of the data is 85.0 (1985.0) and the end time is 87.76 (1987.76)

In figure 2.3 we can find the fit of Didier Sornette's estimations of the parameters. This fit is clearly quite poor. This means it is fair to assume that Didier Sornette made a writing mistake of some sort in article [2]. In conclusion we found that our model performs all right on the Black Monday data, but it is sensitive to the starting time and end time of the data we choose. The critical time is estimated some months later than the date of Black Monday, however we can slightly explain this away by the fact that the critical time is the most probable time of the crash happening, but it can always happen earlier or later. In the next paragraph we will look further into the sensitivity of our model.

2.3.1. Sensitivity of the model

It is informative to investigate how sensitive our model is. Because the more sensitive the model is to changes in the parameters, the less trustworthy. There are a lot of parameters we can look at. The first one is the chosen starting time of the data taken into account. In figure 2.2 for example we took 1985.0 as starting time, but we could have started later or earlier as well. Secondly, the end time of the data taken into account is also very important for the estimations you get. In present markets you would take the present time as end time, however in this case we have a historical market where we can choose the end time ourselves. Thirdly we have the starting parameters we can choose for our Levenberg-Marquardt algorithm. This influences the convergence of the algorithm: To a local minimum or a global minimum. As told in paragraph 2.2.

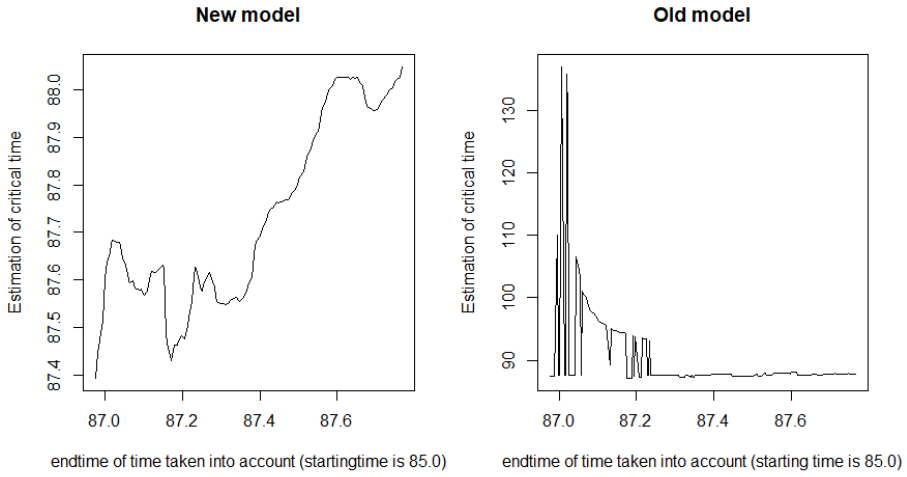

Figure 2.4: Difference in estimation of the critical time when taking various end times and using the new model and old model. (Old model referring to (2.9) and New model referring to (2.10))

In figure 2.4 we can see that our model is very sensitive to the end time we take. Our new model gives an estimation of the critical between approximately 1987.4 and 1988. This is a difference of more than half a year which is very much. Especially when you are consulting someone, who wants to know whether there is a bubble and when it is going to burst. We can see that if we take our end time earlier, our estimation of the critical time becomes earlier as well. The graphs in figure 2.4 show that the the sensitivity of the critical time is way higher in the old model (2.9) than in our rewritten model (2.10). The old model varies from more than 40 years in estimation, which is ridiculous ofcourse. It is great to see that rewriting our model has sufficient influence on the performance of estimating the critical time, when we vary the end time.

2.3.2. Looking at different models

In the beginning of my bachelor thesis I mostly used formula (2.10) to estimate the parameters and especially the critical time. However, often this model does not work that well. That is because the Levenberg-Marquardt algorithm converges to local minima often. Since a cosine is involved in our model we have a lot of local minima. Because of this we often get convergence to a minimum which is not the global minimum which we are looking for. That's why we used the rewritten version (2.11). This equation has one less non-linear parameter. This reduces the optimization problem from a four dimensional problem to a three dimensional problem which enhances the performance of the Levenberg-Marquardt algorithm. This makes our Levenberg-Marquardt algorithm perform better. This is explained in greater detail in article [11]. This means that the Levenberg-Marquardt algorithm will converge to the global minimum more often than to a local minimum when we use model (2.11). Let's see if we can find out its performance is significant better. We will use this model and the old model on Black Monday to see if there are differences in the performance. Both algorithms were used to estimate the critical time of the crash of October 1987 of the S&P500. To research the sensitivity of both models we looked at the sensitivity when we vary the value of the starting value of the non-linear parameters, when we vary the starting time of the date we use and when we vary the end

time of the date we use. On the next pages you can find the results we obtained shown in figures.

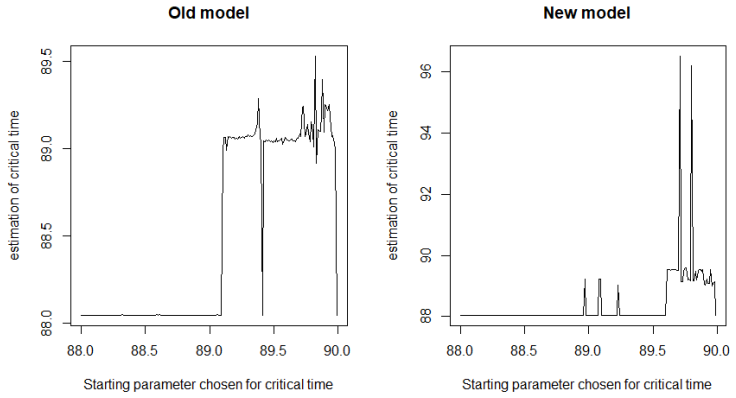

Figure 2.5: Difference in estimation of the critical time when taking various values of the starting parameter of t_c (old model referring to formula (2.10) and new model referring to formula (2.11)). The old model is not that stable in the beginning, but the new model is not so stable at the end. Notice that we let the starting value of the critical time vary in a two year range which is quite big. In this figure neither one of the models is performing significantly better

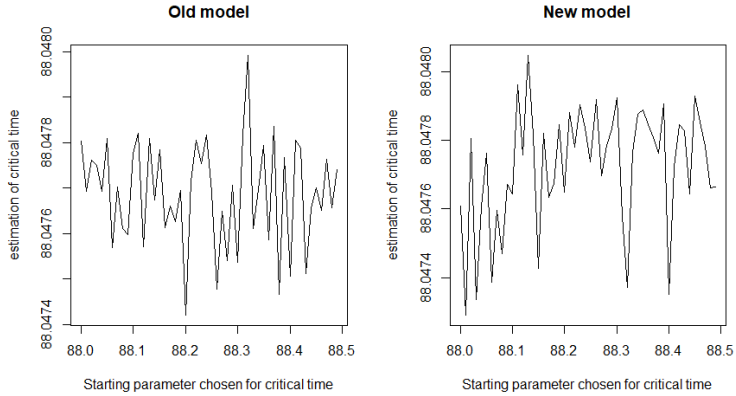

Figure 2.6: Difference in estimation of the critical time when taking various values of the starting parameter of t_c (old model referring to formula (2.10) and new model referring to formula (2.11)). We let the starting value of the critical time vary in a shorter time period than in our previous model. We can see that both models are very stable in this time period. All of the estimations of the critical time our between 88.047 and 88.048, which is really close together. In this figure neither one of the models is performing significantly better

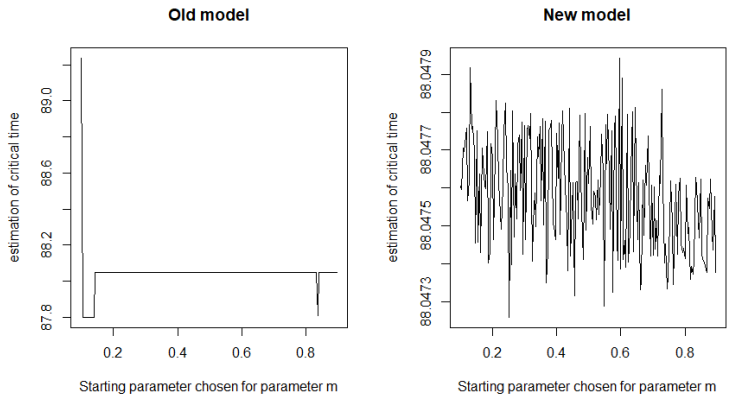

Figure 2.7: Difference in estimation of the critical time when taking various values of the starting parameter of m (old model referring to formula (2.10) and new model referring to formula (2.11)). The old model is not that stable when we take m very small. Our new model is performing slightly better when we look at different m.

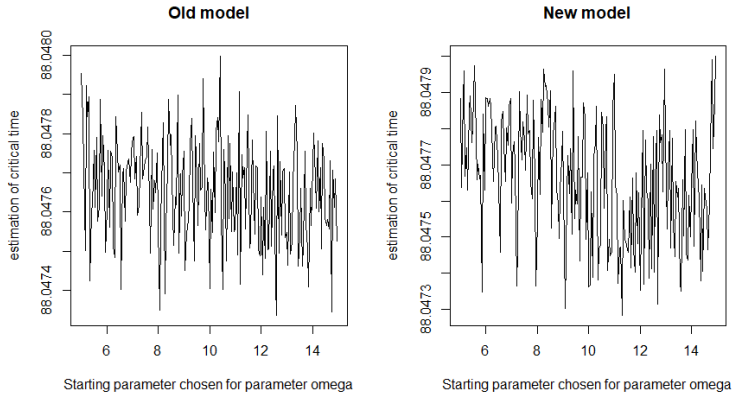

Figure 2.8: Difference in estimation of the critical time when taking various values of the starting parameter of ω (old model referring to formula (2.10) and new model referring to formula (2.11)). We see that both models are very stable under different values of ω. Both are performing well. Neither one of the models outperforming the other one.

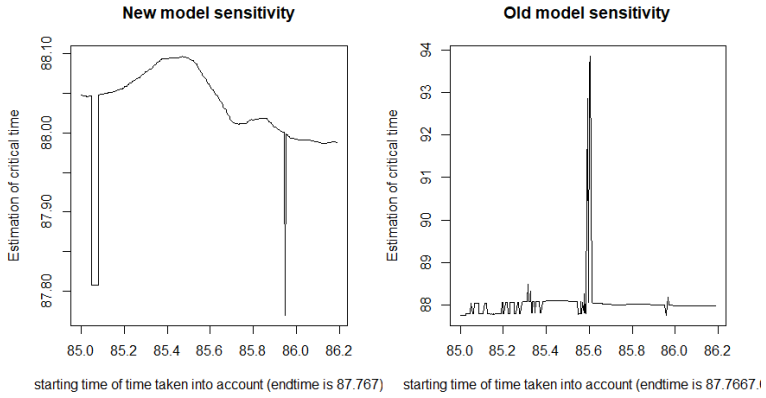

Figure 2.9: Difference in estimation of the critical time when taking various values of the starting time of the data. (old model referring to formula (2.10) and new model referring to formula (2.11)) In the figure we can see that our new model is more stable than our old model when we vary the starting time of the data we used.

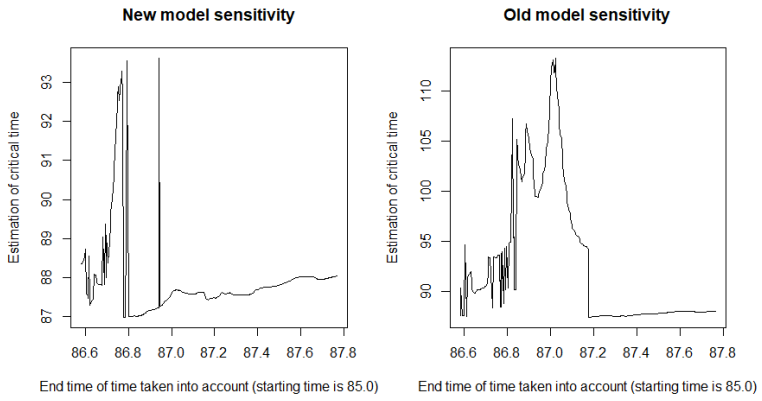

Figure 2.10: Difference in estimation of the critical time when taking various values of the end time of the data. (old model referring to formula (2.10) and new model referring to formula (2.11)). In this figure we let that end time of the data we used vary for a bit more than a year. We see that the old model is more sensitive, because the estimations of the critical time vary a lot more. In this figure the new model is performing slightly better

In conclusion we can say that our rewritten model (2.11) performs slightly better at this data than model (2.10) does. We already found that model (2.10) performs better than model (2.9), because it has one parameter less. This implies that model (2.11) performs better than model (2.9) as well. We find that (2.11) performs best, so from now one we will use model (2.11) to estimate the critical time.

3

Simulating data

To test my model, it is useful to simulate realistic bubbles. To do this we have to decide what values we will choose for the parameters and what kind of noise we will add to the the simulated bubble. First we have to decide how many trading days we will take into account. 500 trading days, which corresponds to almost two (trading) years, is a good amount. Didier Sornette took this timespan as well when he looked at Black Monday in article [2]. So we will take 500 trading days into account. As discussed in chapter two we will use formula (2.11) for the simulation of the bubble:

$$p(t) = A + B(t_c - t)^m + C_1(t_c - t)^m \cos(\omega \log(t_c - t)) + C_2((t_c - t))^m \sin(\omega \log(t_c - t))$$

In this table you can see the parameters we chose:

Parameter	A	B	t_c	m	C_1	ω	C_2
Chosen value	4000	-50	530	0.5	0.1	10	0.1

When we use these parameters to simulate a bubble we obtain figure 3.1

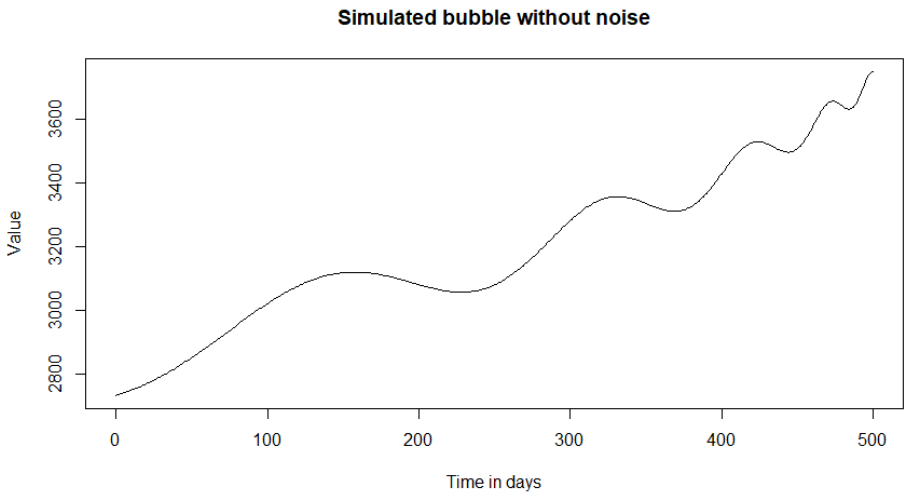

Figure 3.1: Simulated data without any noise added. The graph does not look like a real market. That is why we will add some noise to make it more realistic.

If we let x_i be the simulated data without noise with $1 < i < 500$. Let ϵ_i be the noise we want to add. Then we get

$$y_i = x_i + \epsilon_i$$

in which y_i is the simulated data with added noise. y_i is the data we will use. The question remains what kind of noise we want to add. In the next two paragraphs we will discuss to different types of noise: Normally distributed noise and t-student distributed noise.

3.1. Normally distributed noise

In this paragraph we will look at the results when we take normally distributed noise with mean zero. However, what should the standard deviation of the normally distributed noise be? In figure 3.2 you can see four plots of simulated data with different kind of noise:

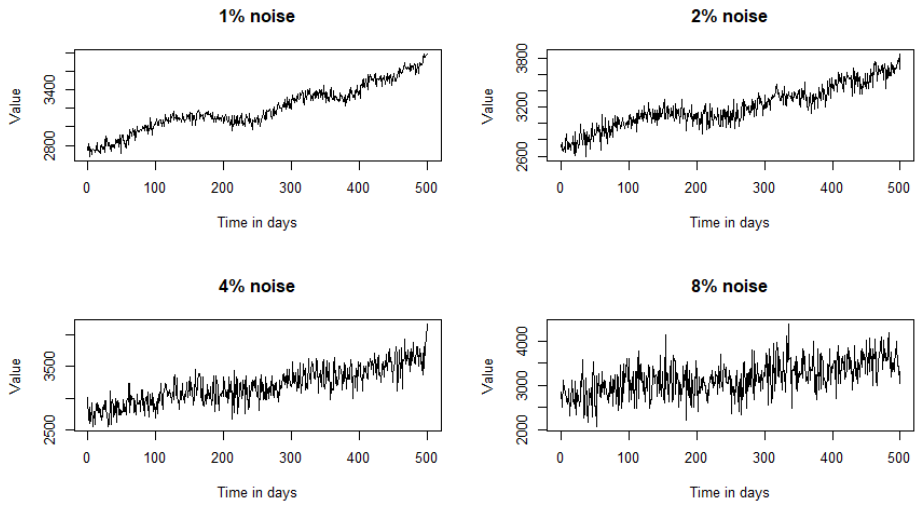

Figure 3.2: four examples of simulated data with normally distributed noise with different standard deviation are shown. The percentage noise is the percentage of parameter A, which is 4000, that we took as standard deviation. So in the 1% case we have a standard deviation of $0.01 \cdot 4000 = 40$.

Keep in mind that by 1% noise we mean a standard deviation of 0.01 times parameter A. In this case that is $0.01 \cdot 4000 = 40$. Until paragraph 3.2 we will keep using this notation. So 0.5% noise gives us normally distributed noise with a standard deviation of $0.005 \cdot 4000 = 20$ and 3.65% noise corresponds with normally distributed noise with a standard deviation $0.0365 \cdot 4000 = 146$. The graphs in figure 3.2 give an indication what kind of noise would be realistic. The graph with 8% noise for example is way to volatile for an ordinary market.

3.1.1. Ordinary market

As noted earlier 8% is way to volatile for an ordinary market. Perhaps this kind of noise corresponds better to markets like Bitcoin: Very volatile markets. In this case we wanted to simulate a bubble belonging to an index like the S&P500 or AEX, so we are not taking 8% noise for sure. An index like the S&P500 or AEX looks more like the one with one percent noise. Source [9] gives us that in the past 60 years 50% of the daily changes were between -0.41% and 0.49%. Because of this, we will take a normal distributed noise with a mean of 0 and a standard deviation of 0.5%. Figure 3.3 shows us how the datagraph looks like when we add the normally distributed noise with standard deviation 0.5% and mean zero:

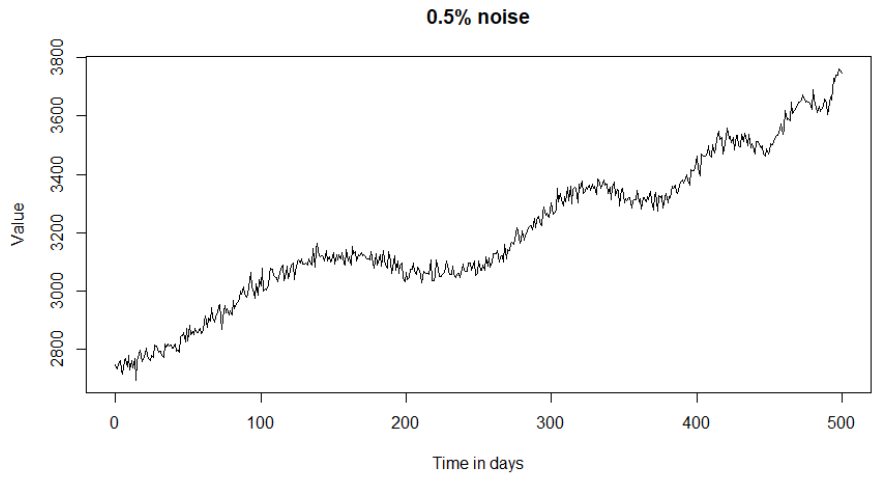

Figure 3.3: Example of simulated data with normally distributed noise with 0.5% standard deviation and mean zero.

Now we can use the simulated data to find out how sensitive the non-linear parameters are while also knowing the true value of the parameters beforehand. This is very useful for researching the sensitivity of the non-linear parameters. On the next pages we will look into the sensitivity of the non-linear parameters.
We first have to decide how much we will vary our non-linear parameters. Note that $0 < m < 1$ and ω is around 10 (See article [11]. That is why we will vary the starting value of m between 0.1 and 0.9 and the starting value of ω between 5 and 15. The starting value of t_c we will vary between 505 and 555. Remember that the true value ot t_c is 530.

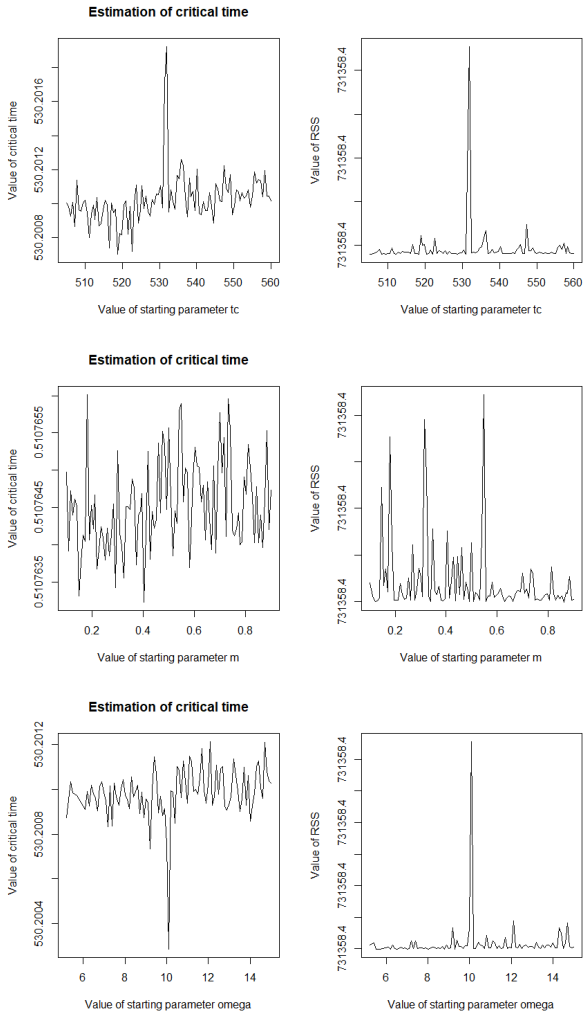

Figure 3.4: For one simulated bubble with 0.5% noise we varied one of the non-linear parameters in linear steps (varying t_c from 505 tot 555, m from 0.1 tot 0.9 and ω from 5 tot 15). In this figure we can find six graphs showing the estimation of the critical time on the left and the value of the residual sum of squares on the right. The estimations of the critical time look very stable when varying the non-linear parameter.

In figure 3.4 we cannot conclude that one of the non-linear parameters is more sensitive to changes in their starting value. The estimations of the critical time look very stable when varying all three of non-linear parameters. We simulated just one bubble here, however there is randomness involved, because of the noise we add. That is why we simulated the noise 100 times and looked at what happened when we varied starting parameters of the non-linear parameters. We looked at the mean and standard deviation of the estimations of the critical time when varying the non-linear parameter. Besides that, we looked at the residual sum of squares. We always want the lowest residual sum of squares, because that is the best fit.

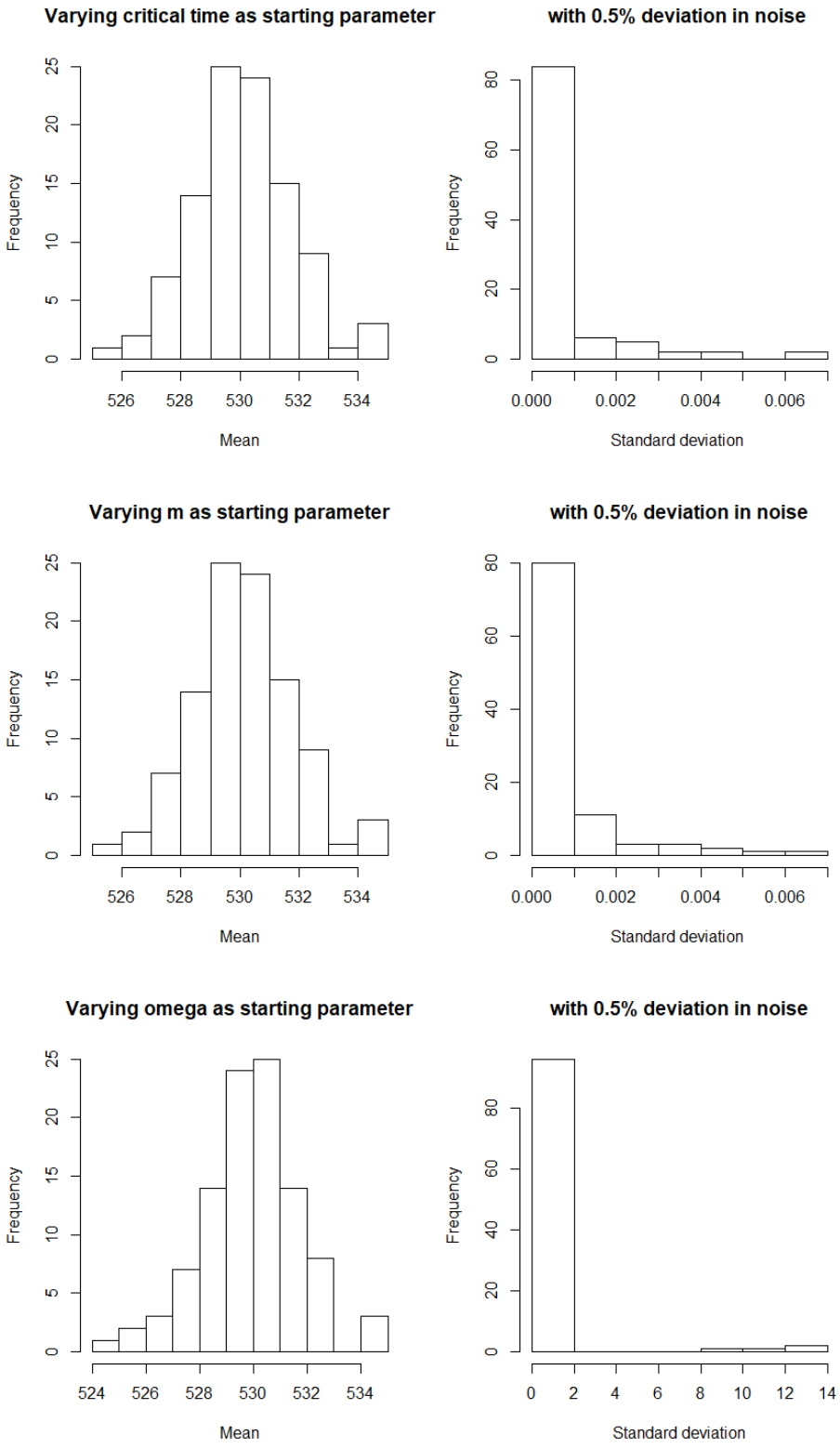

Figure 3.5: For 100 simulated bubbles with 0.5% normally distributed noise we varied the non linear parameters. In this figure we can find six histograms showing the frequency of means and standard deviation of the distribution of estimations of the critical time, when varying one of the three non-linear parameters in linear steps (varying t_c from 505 till 555, m from 0.1 till 0.9 and ω from 5 till 15). So in total we have 100 means and standard deviations calculated from the distribution of estimations of the critical times while varying t_c, m or ω.

Figure 3.5 shows us that varying the starting value of t_c and m does not have much influence on the estimation of the critical time, since the standard deviations are very small. Varying the starting value of ω has more influence on the estimation of the critical time, because the standard deviation is sometimes around 10 which is quite high. When looking at this figure we would say that ω is the most sensitive non-linear parameter.

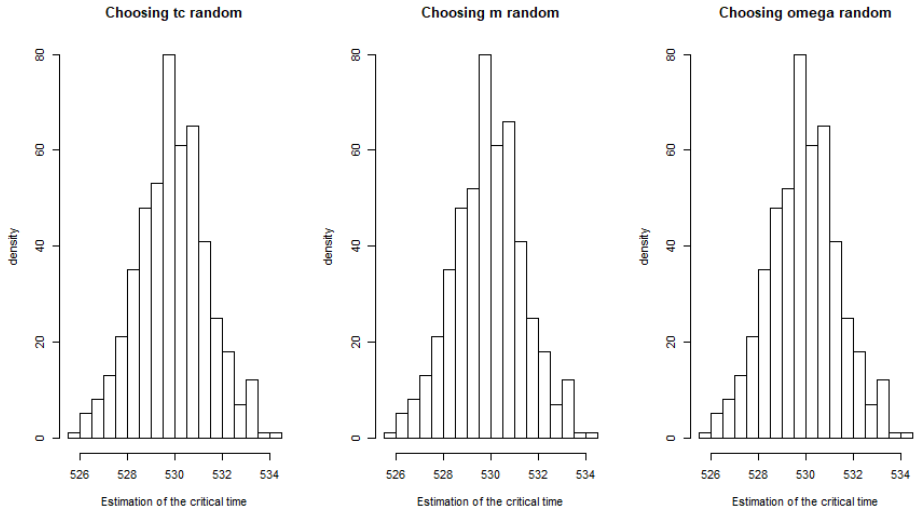

Figure 3.6: We simulated bubbles 500 times with 0.5% normally distributed noise. For every bubble we varied one of the starting values of the non-linear parameters ten times random (t_c randomly chosen out 505-550, m randomly chosen out 0.1-0.9 and ω randomly chosen out 5-15). After we did this ten times, we picked the best fit: The fit with the lowest residual sum of squares. So we picked the best fit 500 times and we did this for all three non-linear parameters. In the three histograms you can find the density of the value of the critical time for the best fits.

In figure 3.6 we can find three histograms showing us the density of the best estimations of the critical for 500 non-identical simulated data when varying one of the non linear parameters. We can see that all three of the histograms look very much alike. This is not strange at all, because it is likely that there is only one minimum in the neighbourhood. In that case varying the non linear parameters does not have any impact on the convergence. The means of all three histograms are all around 530 (to be precise 529.9). The histograms look like they could be normally distributed. When we run the Shapiro Wilk test we find these p-values: 0.643, 0.631 and 0.644. This implies that we cannot reject the null hypothesis that the data comes from a normal distribution. So it is a reasonable assumption that in all three cases the data comes from a normal distribution. When we look at the standard deviation of the histograms we find that they are all almost the same: approximately 1.46.
In conclusion, findings reveal that the model yields a moderate-to-good fit, when we consider the lowest residual sum of squares. At the same time, findings show that the model is very sensitive to variation in the starting values of the non-linear parameters, especially variation in the non-linear parameter ω.

3.1.2. Volatile market

In the previous paragraph we looked at ordinary markets. One of our goals of the bachelor thesis was to find out if we could apply the model on Bitcoin data as well. To do this it is useful to simulate a market with the same amount of volatility as Bitcoin. To do this we obtained the Bitcoin data from yahoo finance [5] and calculated the daily changes of Bitcoin over the past years. We can see that Bitcoin is a very volatile market. There are multiple peaks with a daily change of more than 20%. The highest peak at early 2014 is even a daily change of more than 100%. We can calculate the average daily change by taking the sum of all the daily changes and dividing them by the amount of trading days minus 1. This appears to be 0.0365 which is 3.65%. We will take normally distributed noise with mean zero and standard deviation 3.65%, add it to our generated data and investigate the sensitivity of our model.

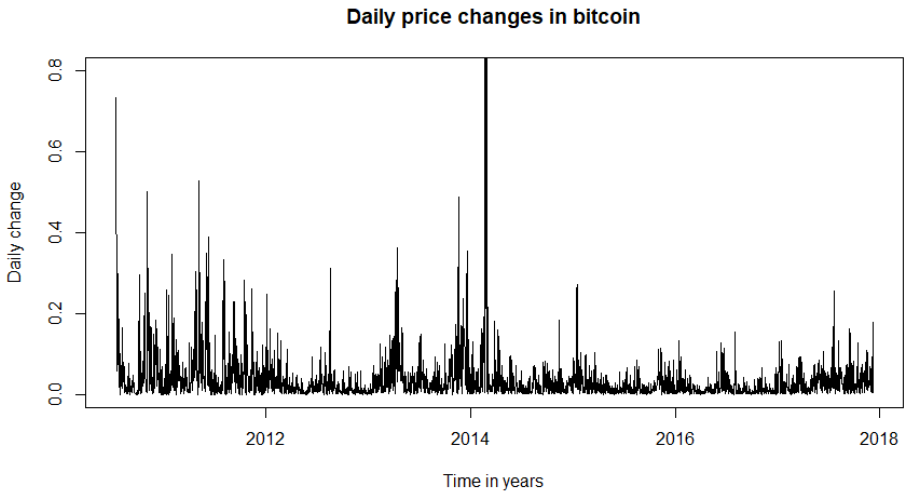

Figure 3.7: Daily price changes of Bitcoin's value in the past years. Notice that Bitcoin is very volatile.

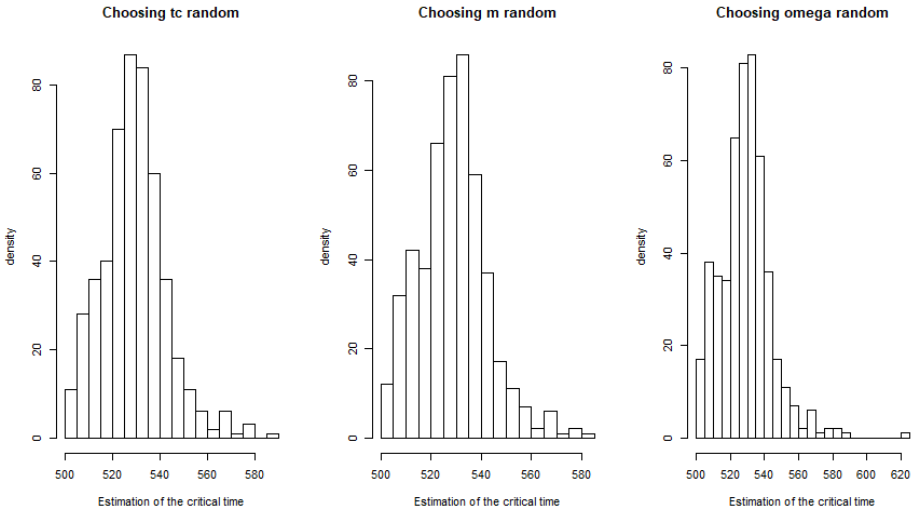

Figure 3.8: We simulated bubbles 500 times with normally distributed noise with 3.65% standard deviation. For every bubble we varied one of the starting values of the non-linear parameters ten times random (t_c randomly chosen out 505-550, m randomly chosen out 0.1-0.9 and ω randomly chosen out 5-15). After we did this ten times, we picked the best fit: The fit with the lowest residual sum of squares. So we picked the best fit 500 times and we did this for all three non-linear parameters. In the three histograms you can find the density of the value of the critical time for the best fits.

In figure 3.8 we did the same as in figure 3.6, but this time we used simulated data with normally distributed noise with 3.65% standard deviation. If we compare both figures, we find that the estimations of the critical time in a volatile market are more spread than in an ordinary market. This is also what we expected. The standard deviations from left to right are: 13.44, 13.57 and 14.89. This implies that it common that we are more than 13 days off point when estimating the critical time. The means of the three histograms are in order from left to right: 529.2, 528.8 and 529.0. This is still close to 530, which was the critical time we used to simulate the data. The histograms do not look like they are normally distributed anymore. Mostly because an estimation of the critical time lower than 500 cannot be made (since the 500th day is the last day we take into account for our simulated data). But frequently a best fit of the critical time is estimated higher than 560. We also find this when we run the Shapiro Wilk test in R. We get very small p-values, which implies that we reject the null hypothesis that the data comes from a normal distribution.
On the next page you can find figure 3.9 which is the same as figure 3.5, but for a volatile market this time. We can conclude that our model works decent on markets with the same amount of volatility as Bitcoin. There

is a lot of standard deviation sometimes even more than 80. This means that the model is very sensitive to changes in the starting value of the non-linear parameters.
In conclusion we find that the model performs decent when we take the best fit with the lowest residual sum of squares. We also find that the model is very sensitive to changes in the starting value of the non-linear parameters.
In the next paragraph we will look into adding t-student distributed noise instead of normally distributed noise.

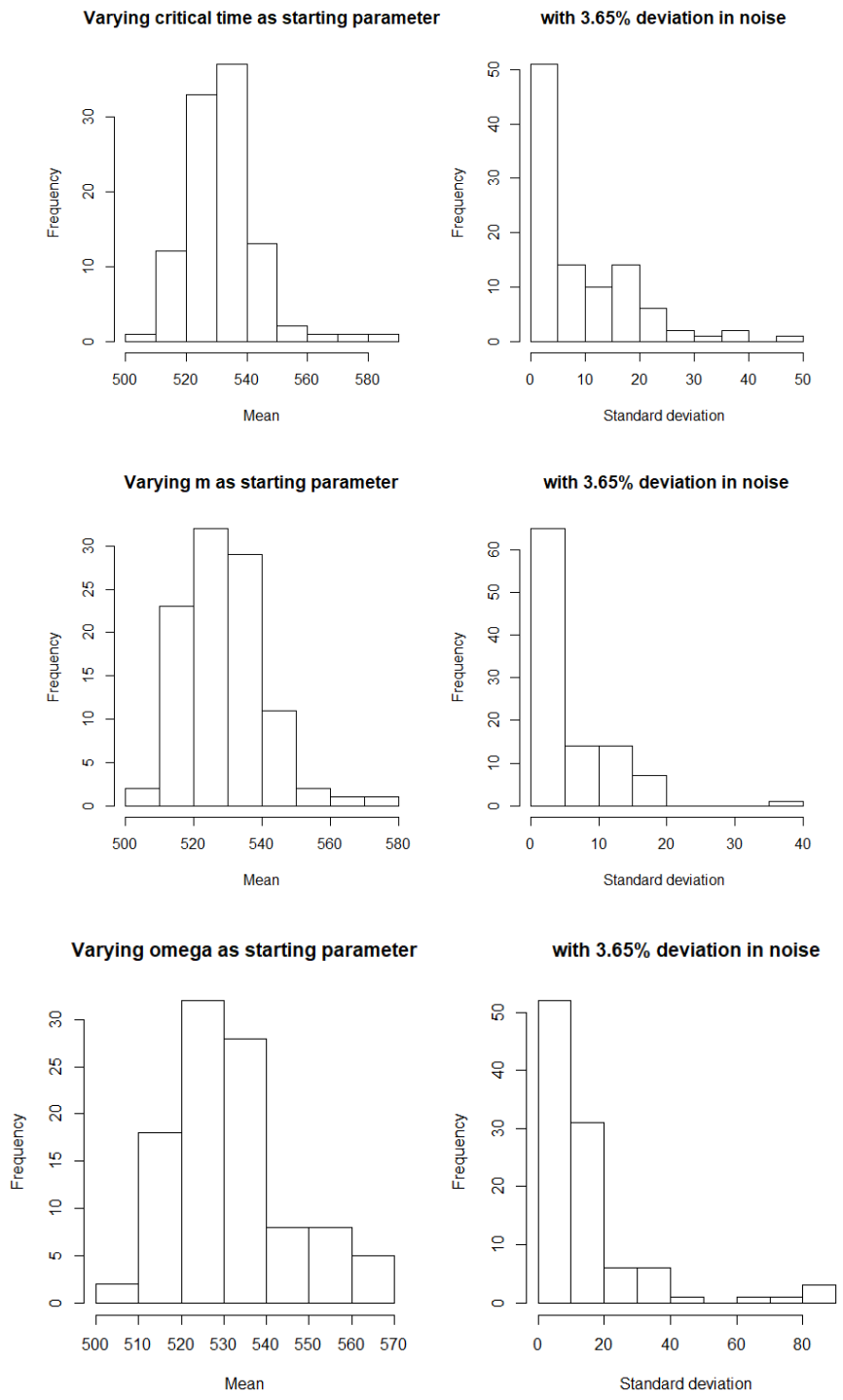

Figure 3.9: For 100 simulated bubbles with 3.65% normally distributed noise we varied the non linear parameters. In this figure we find six histograms showing the frequency of means and standard deviation when varying one of the three non linear parameters (varying t_c from 505 till 555, m from 0.1 till 0.9 and ω from 5 till 15). We can see that our model is way more volatile in a market with 3.65% volatility than in a market with 0.5% volatility.

3.2. t-student distributed noise

We will introduce t-student distributed noise, because noise from t-student distribution may be more realistic than normally distributed noise for simulating financial market data. Below you see the difference between a standard normal distribution and t-student distribution with four degrees of freedom.

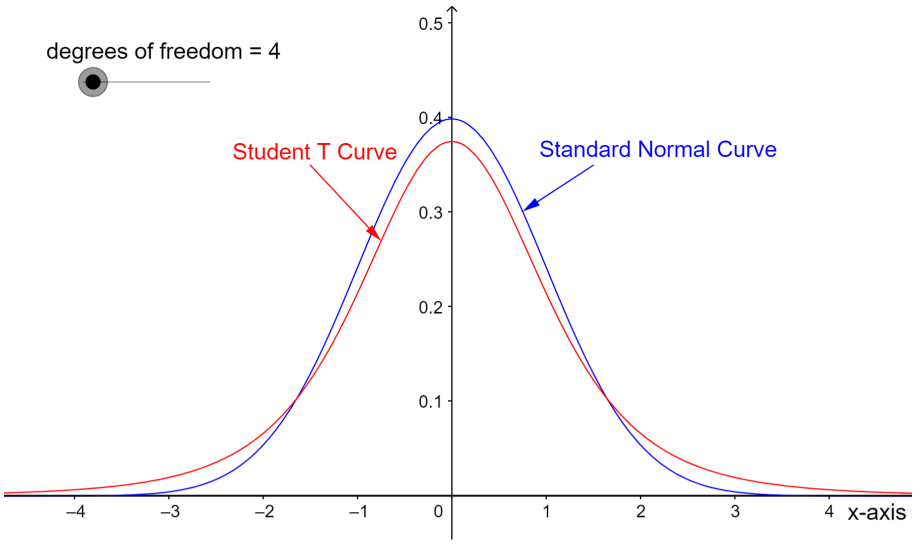

Figure 3.10: A visualization of the difference between standard normal distribution and t-student distribution with four degrees of freedom. Image is from source [7].

We can see in figure 3.10 that the t-student distribution has fatter tails than the standard normal distribution. That is why we think this is more realistic for a financial markets. Often when we have a big daily change in financial markets, it is a very big daily change. This implies that we need fatter tails than we have in a normal distribution. That is the argument why the t-student distribution may be more realistic. We will use four degrees of freedom because it exhibits a tail similar to that often reported in literature for the distribution of financial returns (see article [1]).

3.2.1. Ordinary market

We take t-student distributed noise with four degrees of freedom and mean zero. Then we scale it the same way we did with normally distributed noise: We multiply our noise with 0.5 times parameter A.

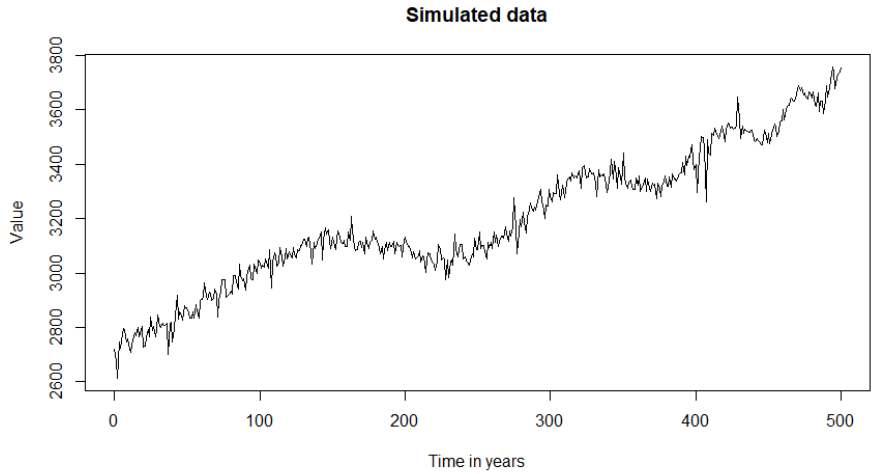

Figure 3.11: Example of simulated data with t-student distributed noise with four degrees of freedom scaled with 0.5% of parameter A.

The graph in figure 3.11 looks very much the generated data with the normal distributed noise with mean

zero and 0.5% standard deviation in figure 3.3. However the t-student distribution has fatter tails than the normal distribution. We can see this back in the graphs when we compare both: The graph in figure 3.11 is more volatile than the graph in figure 3.3.

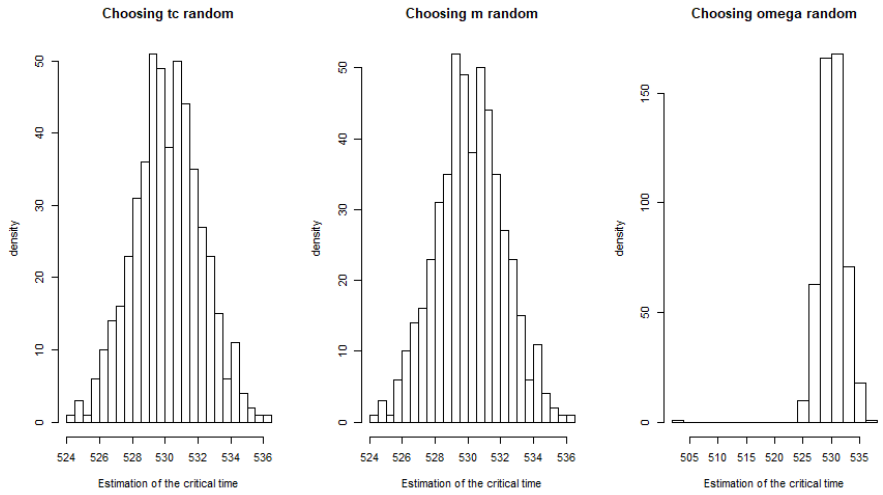

Figure 3.12: We simulated bubbles 500 times with t-student distributed noise and four degrees of freedom and 0.5% standard deviation. For every bubble we varied one of the starting values of the non-linear parameters ten times random (t_c randomly chosen out 505-550, m randomly chosen out 0.1-0.9 and ω randomly chosen out 5-15). After we did this ten times, we picked the best fit: The fit with the lowest residual sum of squares. So we picked the best fit 500 times and we did this for all three non-linear parameters. In the three histograms you can find the density of the value of the critical time for the best fits.

We find that the histograms in figure 3.12 look like the histograms we got in figure 3.6. However the histograms in figure 3.12 have higher standard deviations. Their standard deviations are from left to right: 2.06, 2.06 and 2.39. All of their means are approximately 530.1, which is very close to 530. When we test for normality with the Shapiro Wilk test we find from left to right the p-values of 0.93, 0.93 and $2.2 * 10^{-16}$. We find that the data from the histogram on the right does not come from a normal distribution. Very reasonable, because when we look at the graph on the right we see a big outlier. The data from the two other histograms could come from a normal distribution though.

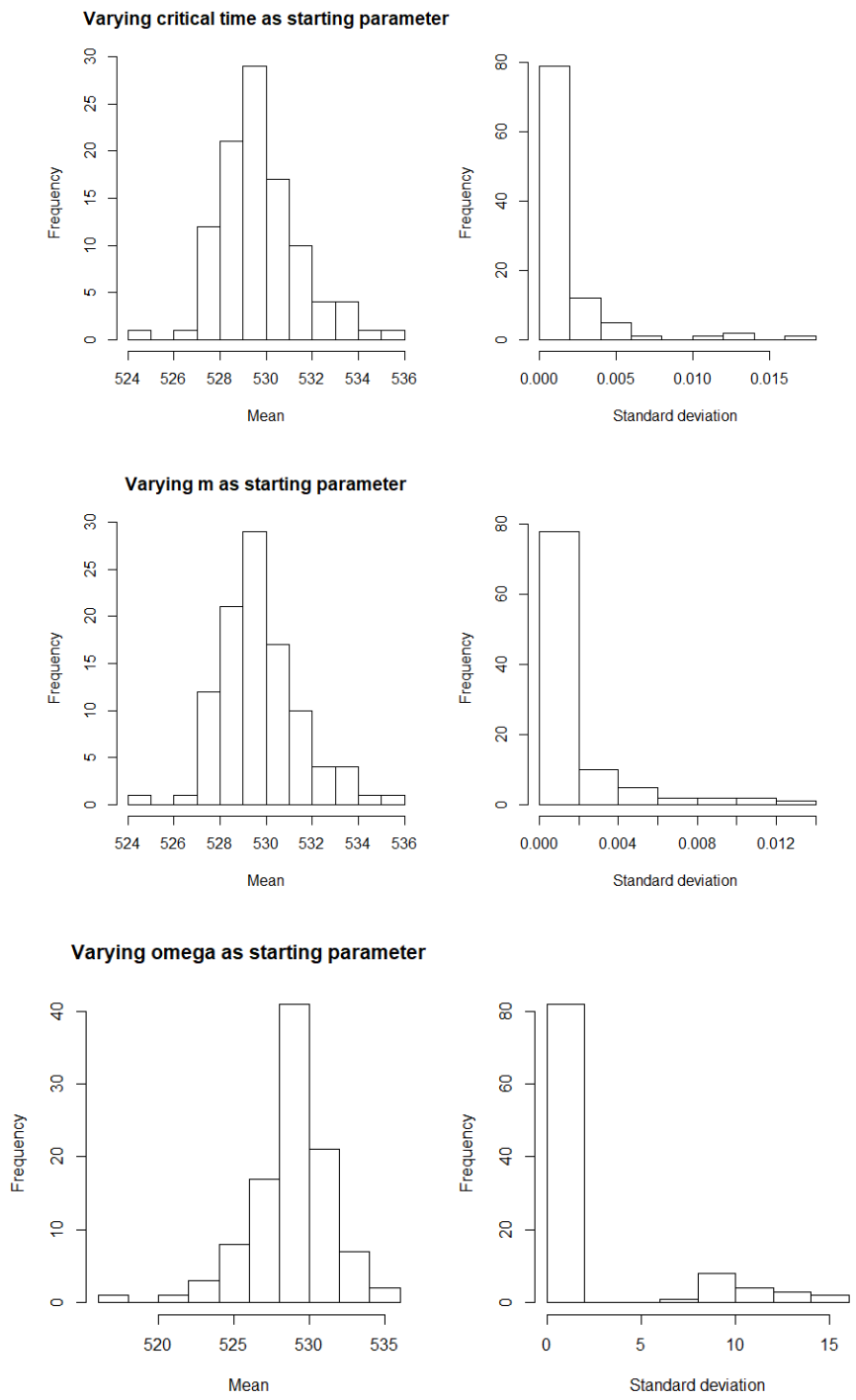

Figure 3.13: For 100 simulated bubbles with t-student distributed noise with four degrees of freedom and mean zero. We varied the non linear parameters and estimated the critical time. We want to look at the distribution of the estimations we get when varying the value of the non linear starting parameters. In this figure we can find six histograms showing the frequency of means and standard deviation when varying one of the three non linear parameters (varying t_c from 505 tot 555, m from 0.1 tot 0.9 and ω from 5 tot 15). We can see that our model is very stable for varying the value of starting parameters t_c and m. But when we vary ω we find that the estimations of the critical time are stable most of the times.

3.2.2. Volatile market

Again we will simulate a market with the same volatility as Bitcoin. As discussed in paragraph 3.1.2 we will take 3.65% of parameter A as scaling. Again we will use t-student distribution with four degrees of freedom and mean zero. This is how our generated data looks like when we add the noise:

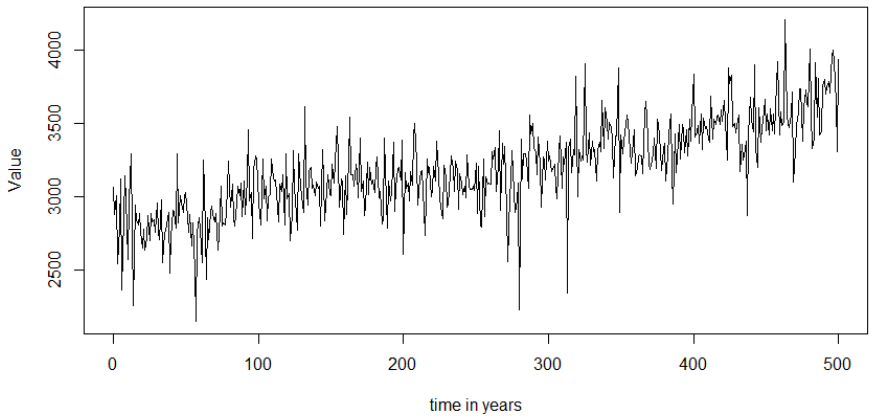

Figure 3.14: Example of the simulated data to generate a volatile market using t-student distributed noise with standard deviation scaled with 3.65% of parameter A.

The graph in figure 3.14 looks very volatile. It does not look like a graph of an average market. This simulated data looks more volatile than than the data generated in paragraph 3.1.2, so we are curious whether our model works fine here.

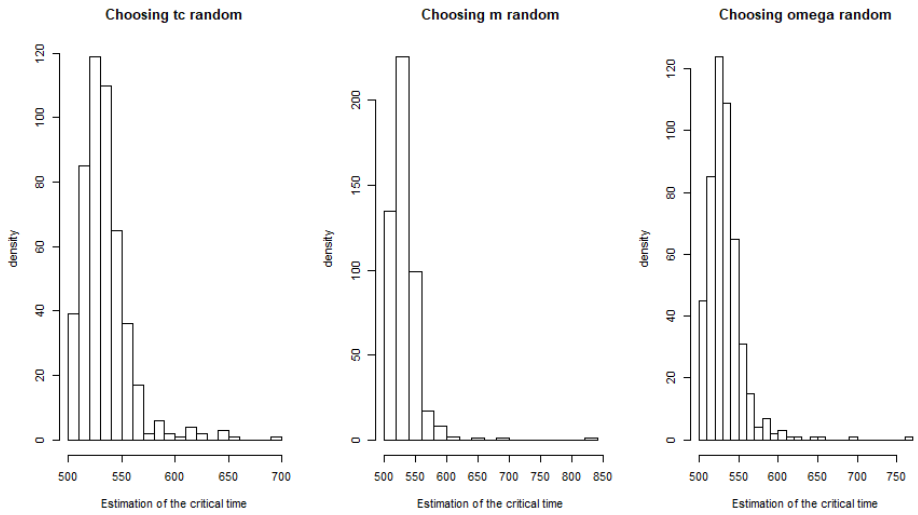

Figure 3.15: We simulated bubbles 500 times with t-student distributed noise with four degrees of freedom and scaled with 3.65% of parameter A. For every simulated bubble we varied one of the starting values of the non-linear parameters ten times random (t_c randomly chosen out 505-550, m randomly chosen out 0.1-0.9 and ω randomly chosen out 5-15). After we did this ten times, we picked the best fit: The fit with the lowest residual sum of squares. So we picked the best fit 500 times and we did this for all three non-linear parameters. In the three histograms you can find the density of the value of the critical time for the best fits for every non-linear parameter varied.

In figure 3.15 we have almost the same histograms as we have in figure 3.8. However this time we have even a wider spread of estimations of the critical time. Sometimes the best fit gives us even a estimation of the critical time higher than 700, which is way higher than 530. The means of the histograms in order from left to right are: 533.9, 532.3 and 533.1. This differs three trading days from the true value of the critical time.

The standard deviations are in order from left to right: 22.87, 23.99 and 23.73. The data from all three of the histograms does not look like it is normally distributed and the Shapiro Wilk test agrees on this.

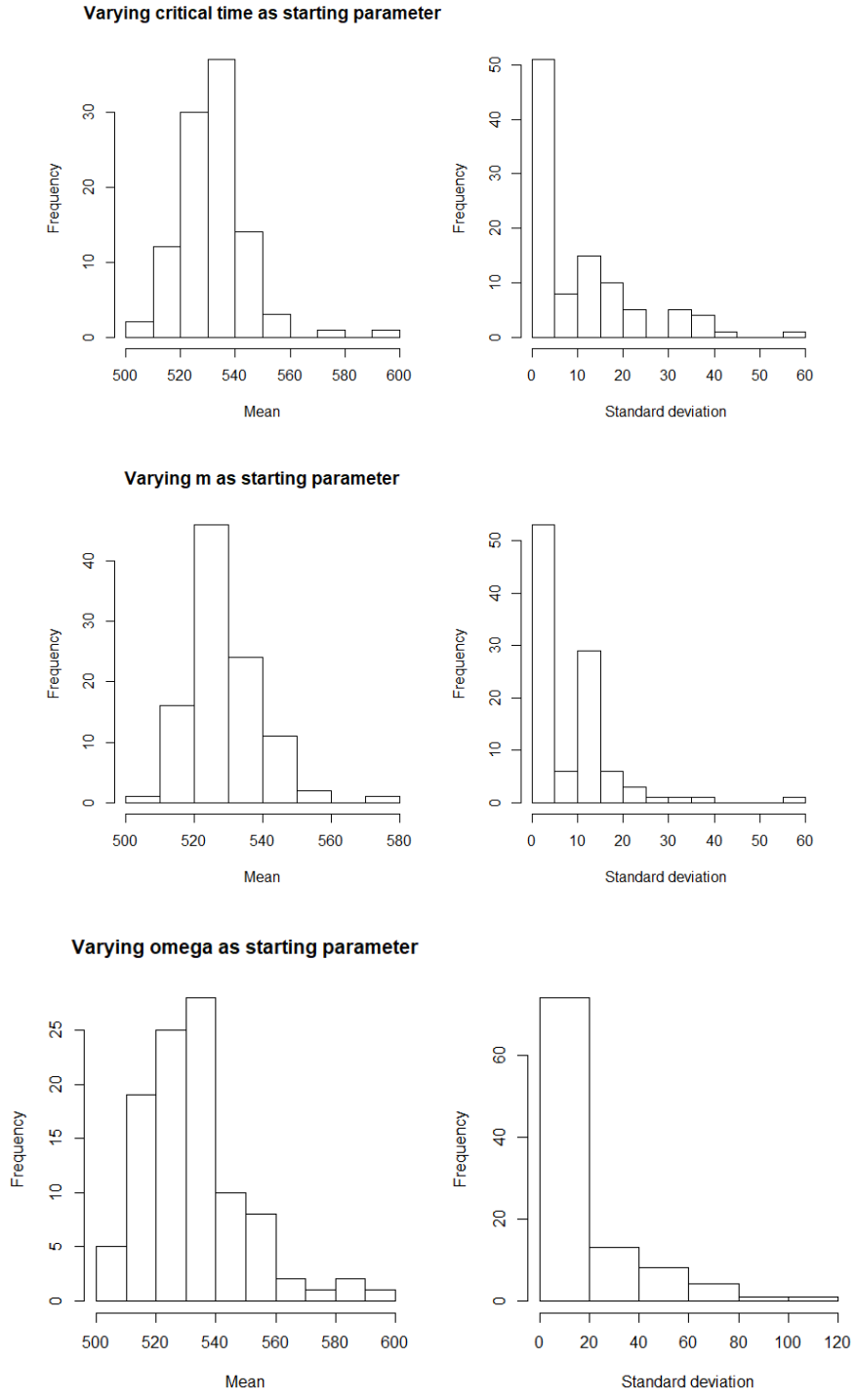

Figure 3.16: For 100 simulated bubbles with t-student distributed noise with four degrees of freedom and scaled with 3.65% of parameter A. We varied the non linear parameters and estimated the critical time. We want to look at the distribution of the estimations we get when varying the value of the non linear starting parameters. In this figure we can find six histograms showing the frequency of means and standard deviation when varying one of the three non linear parameters (varying t_c from 505 tot 555, m from 0.1 tot 0.9 and ω from 5 tot 15).

Figure 3.16 shows us that our model is very sensitive to changes in the starting values of the non-linear

parameters. A standard deviation of more than 30 is common, which means that the choice of the non-linear parameters can influence the estimation of the critical time with even 30 days or more. This is very much, so in a volatile market you have to choose your non-linear parameters carefully.
In conclusion we found in this paragraph that estimations of the critical time in simulated ordinary markets are not very sensitive to changes in the starting value of the non-linear parameters. In simulated volatile markets this is not so much the case: The estimations of the critical time do depend very much on the starting values of the non-linear parameters. We also found in various figures that the estimation of the critical time is most sensitive to changes in the non-linear parameter ω. This means that in the dimension of ω there are more (local) minima than in the other two dimensions of the non-linear parameters.
In the next chapter we will apply our model at two ongoing markets: The Amsterdam Exchange Index and Bitcoin.

4

Applications to ongoing markets

4.1. Amsterdam Exchange Index

The Amsterdam exchange index, abbreviated as AEX, is a stock market index composed of Dutch companies. Since we live in the Netherlands and this is the only index in the Netherlands, we found it interesting to research if we can apply our model on this data. Figure 4.1 shows the value of the AEX of the past years:

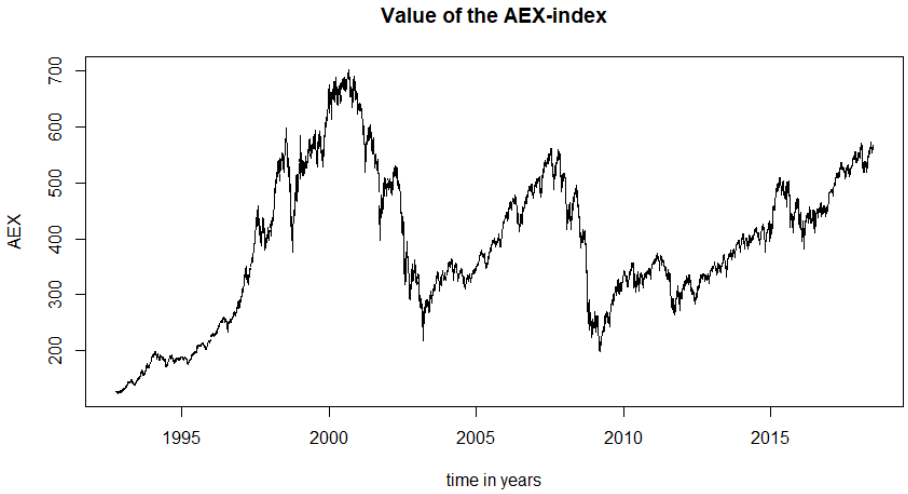

Figure 4.1: The value of the Amsterdam exchange index for the past 25 years. We can see that the value has been increasing for the past years, so one might wonder whether a bubble is forming. Data is downloaded from yahoo finance [4].

The graph in figure 4.1 shows us two financial bubbles bursting. The first one is the dot-com bubble in 2000/2001. Everyone was hyped for the new possibilities the internet offered and that caused a lot of speculation. In the end the dot-com bubble burst, but we do know what a big impact the internet has had to the whole world. The second one is the financial crisis of 2008. A lot of countries including the Netherlands suffered from this crisis and a lot of companies bankrupted. In the graph we can see that in the present moment we are hitting the same high value of the AEX as in 2008 when the bubble burst as we can see in the graph. One can wonder whether a bubble might be forming right now. Well that is precisely what we can use our model for to estimate the critical time of the bubble forming right now. We used model (2.11), since we found in preceding paragraphs that this model works best. We need to think about the value of our starting time of the data. A logical choice is the end of the 2008 crisis. On the next page you can find the fit and estimations for the parameters we obtained

Parameter	Estimate	standard error	t-value	p-value
A	808.4	28.18	28.689	$< 2e-16$
B	-83.86	13.31	-6.3	$3.53e-10$
t_c	2023.58	0.2716	7451.3	$< 2e-16$
m	0.7097	0.04397	16.14	$< 2e-16$
C_1	0.1049	0.10493	-0.357	0.721
ω	15.10	0.45703	33.04	$< 2e-16$
C_2	5.950	4.8390	1.216	0.224

Table 4.1: Estimations of the parameters by our model. All p-values are small except the the p-values of C_1 and C_2.

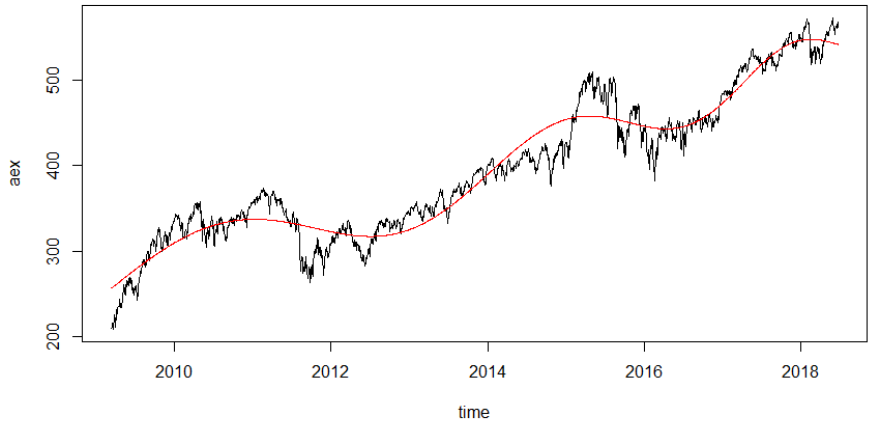

Figure 4.2: Fit of our model to the value of the Amsterdam exchange index on data from the past approximately ten years. The red line shows the fit of our model to the data.

The red line in figure 4.2 gives a moderate-to-good fit to the data. As estimation for the critical time we find 2023.58 as estimation with a very small p-value, which is good. However, not all estimations of the parameters have a p-value significant. The value of parameter C_1 and C_2 are both higher than 0.2, so they are not significant at all. This means that our results are not very trustworthy. What will happen to the estimation of the critical time when we choose another value for the starting time of our data?

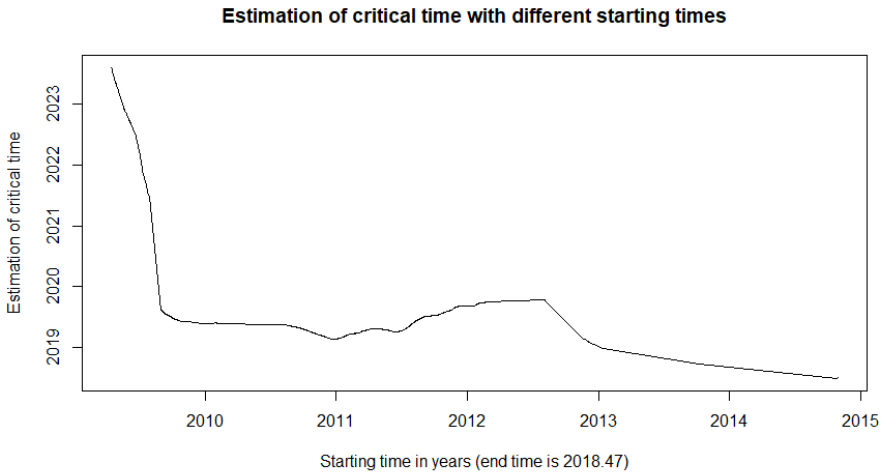

Figure 4.3: Estimations of the critical time when we vary the starting time of our data. Notice that the estimation of the critical time is very dependent of the starting time of our data.

We find in figure 4.3 that our model is very sensitive to the starting time of the data we take into account. Most of the estimations are between 2019 and 2020. It seems that our estimation of 2023.58 is a big outlier when we look at this graph, which is quite interesting. How can we draw conclusions when we have a graph like this? That is very hard, because the estimation of the critical time depends very much on the starting the time of the data we choose.

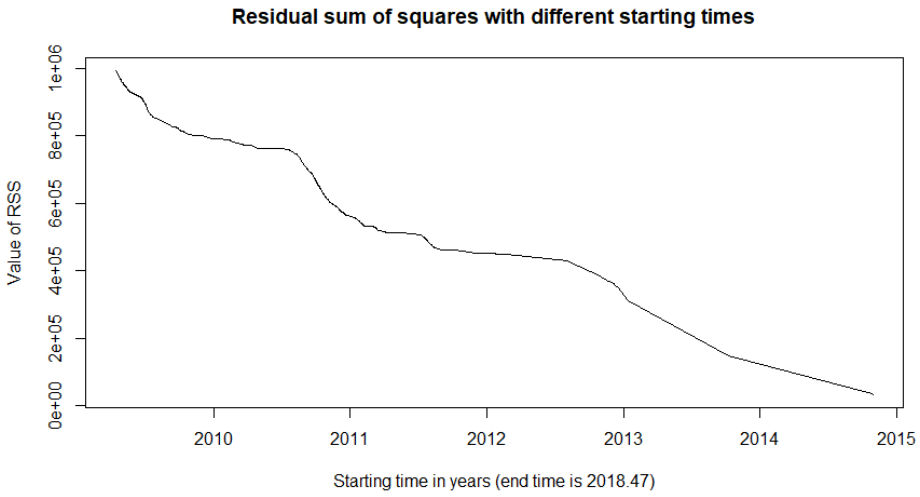

Figure 4.4: Shown is the value of the residual sum of squares when we vary the starting time of our data.

Looking at the residual sum of squares does not give us any new information about the best prediction of the critical time. After all, we have less residuals when we take less time into account, so it is logical that the residual sum of squares is smaller as well. This is not the way to find out what the best prediction of the critical time is.
What we can do is looking at the residual sum of squares when we look at the best starting value for the non-linear starting parameters we choose before running our algorithm. To do this we first have to set the starting date of the bubble. We will set the starting time to 2016.139, so that is mid February. We choose this date, because we will take approximately 500 trading days into account, just as we did when simulating our own data. Besides that, the value of the AEX was quite low in mid February. This means that it is also a good starting point for a bubble.

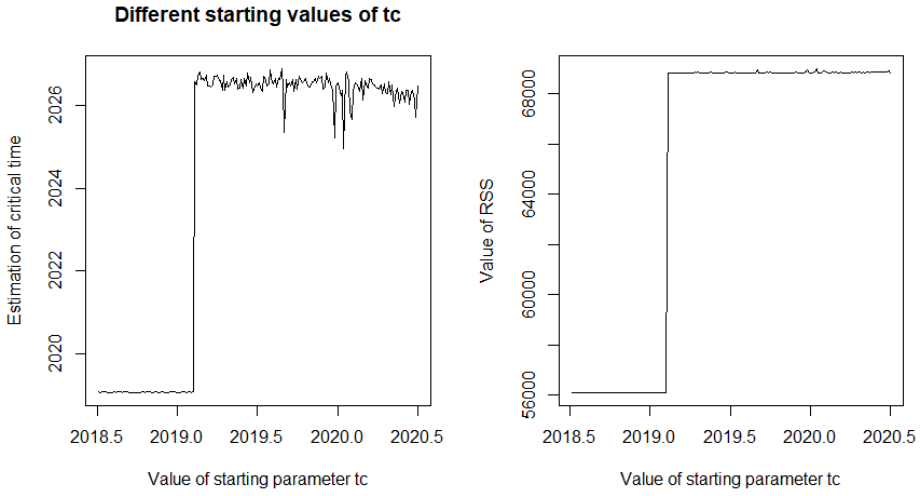

Figure 4.5: Shown are the estimations of the critical time on the left and on the right the value of the residual sum of squares when we vary the starting value of the non-linear parameter t_c.

Figure 4.5 shows us that the residual sum of squares is larger when we take a starting value of the critical time larger than some value around 2019. Because of that we will pick a starting value for the parameter t_c between 2018.5 and 2019, because this will give a better fit. But before doing that, we will take a closer look what happens when we vary the starting value of the critical time between 2018.5 and 2019.

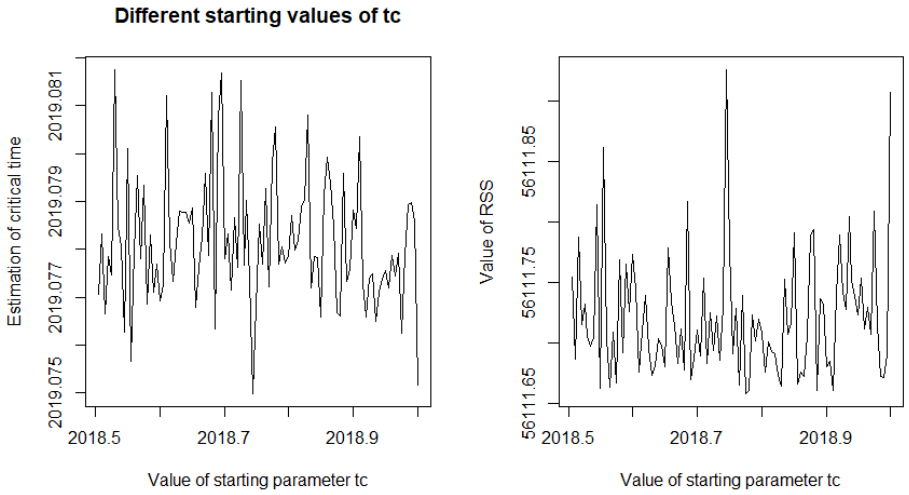

Figure 4.6: Shown are the estimations of the critical time on the left and on the right the value of the residual sum of squares when we vary the starting value of the non-linear parameter t_c for a shorter time period than in figure 4.5

Figure 4.6 gives us that varying the starting value between 2018.5 and 2019 does not give any significant difference in estimations and values of the residual sum of squares. In that case we will take 2018.75 as starting value for the critical time since this is right in between.
We will research the sensitivity of the starting value of parameter m now.

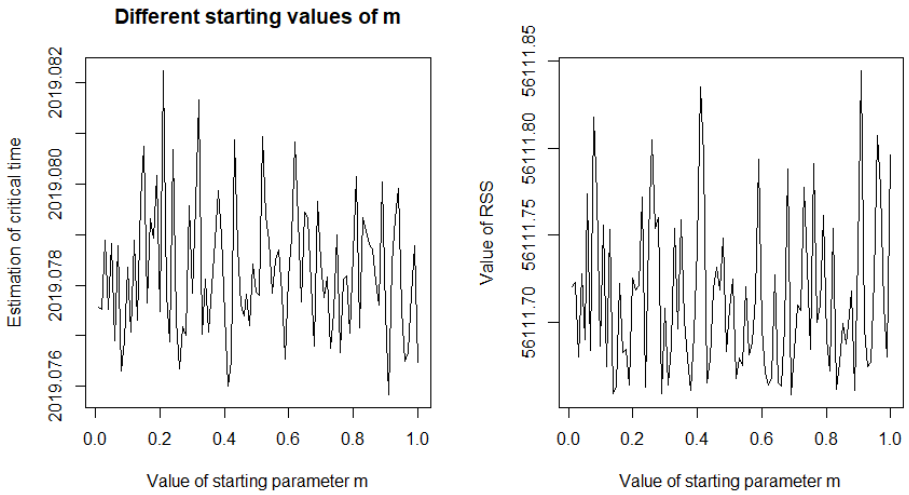

Figure 4.7: Two graphs are shown. On the left you can see the estimations of the critical time when we vary the non-linear parameter m in linear steps from 0 to 1. On the right you can see the value of the residual sum of squares when we vary m the same way.

When we look at figure 4.7 we find that m is very stable under choosing different starting values. There is no significant difference in the estimations of the critical time and the value of the residual sum of squares. Because of the we will just take $m = 0.5$ as starting value.
The last non-linear parameter that remains is ω.

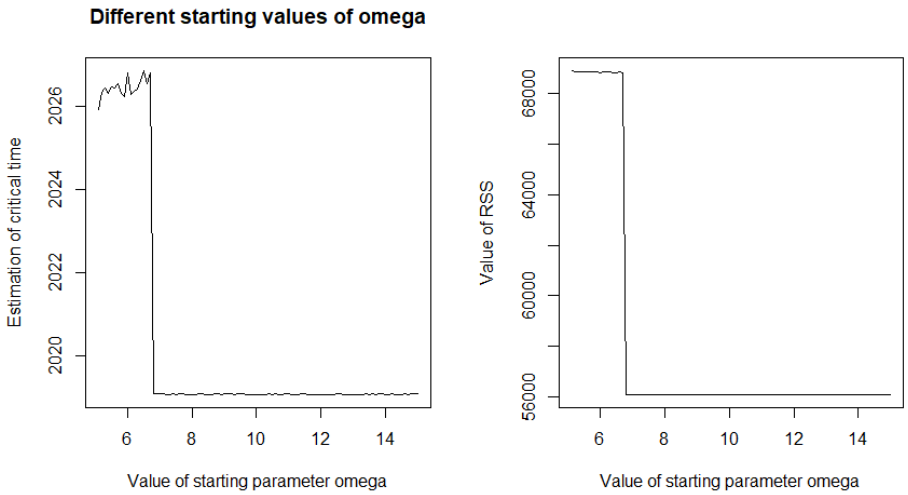

Figure 4.8: Shown are the estimations of the critical time on the left and on the right the value of the residual sum of squares when we vary the starting value of the non-linear parameter ω.

In figure 4.8 we find that we get higher residual sum of squares and higher estimations for the critical time when the starting value of ω is smaller than 7. Before choosing a good starting value for the parameter ω We can take a closer look what happens when we vary the omega between 8 and 14.

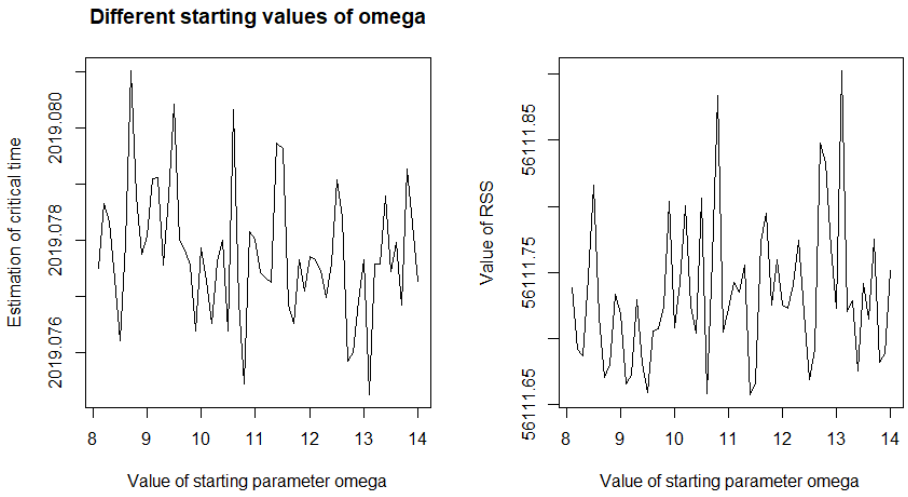

Figure 4.9: Shown are the estimations of the critical time on the left and on the right the value of the residual sum of squares when we vary the starting value of the non-linear parameter ω for a shorter time period than in figure 4.5

We do not see any significant differences in the estimations of the critical time and the value of the residual sum of squares in figure 4.9. As starting value for ω we can take 11, since that is the value in between.
So right now we have these starting values for the non-linear parameters: $t_c = 2018.75$, $m = 0.5$ and $\omega = 11$. With these parameters we can take a closer look when we vary the starting date of the data. We do hereby assume that these starting values are also optimal when we choose different starting dates of the data. In figure 4.10 we can see the results we obtain when we vary the starting dates.

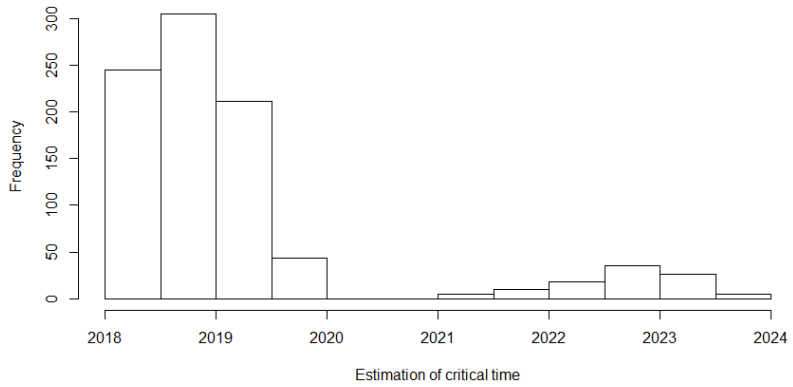

Figure 4.10: In this histogram you can find the frequency of the estimations when we vary our starting date of the data we take into account. Notice that a lot of the critical times are predicted in the coming two years.

In conclusion we find that it is very hard to estimate the critical time of the Amsterdam Exchange Index. Mostly because it is very dependent on the starting time of the data we choose. We see however in figure 4.10 that a lot of the critical time are estimated between 2018 and 2020. This indicates that a (big) loss in value of the Amsterdam Exchange Index in the coming two years would not be a surprise to us.

4.2. Bitcoin

With the upcoming technology of the blockchain a lot of cryptocurrencies emerged out of nowhere. The biggest one of them is Bitcoin. Bitcoin has been gaining a lot of value in the past years. Having a value of approximately 100 dollars in June 2013 and right now (summer of 2018) a value of approximately 7500 dollars. This is a huge increase and Bitcoin has been in the news because of that various times. The question remains whether Bitcoin is a bubble or not. This is where we can apply our model at the data of Bitcoin. Data is downloaded from yahoo finance [5].

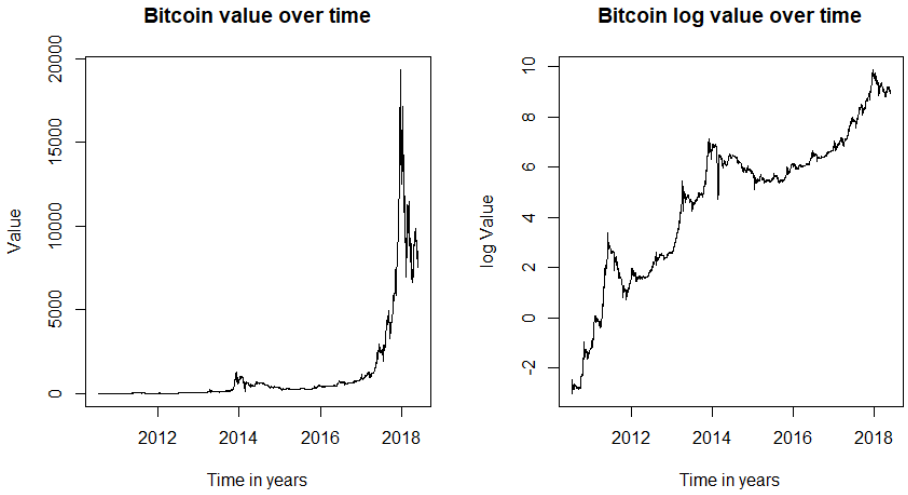

Figure 4.11: In the graph on the left we can find the value of Bitcoin over the past years. The value of Bitcoin has been growing massively. In 2010 a Bitcoin was not even worth 1 dollar. In the graph on the right we can find the logarithmic value of Bitcoin over the past years. It is often useful to look at the logarithmic value of something, to see it from a different perspective.

In figure 4.11 we can find can see that Bitcoin has been growing exponentially the past years. When we look at the logarithmic value we can find that the loss in January/February 2018 is not even that big of a loss when we compare it to other losses in the paste. The loss in 2011 for example was bigger relatively.

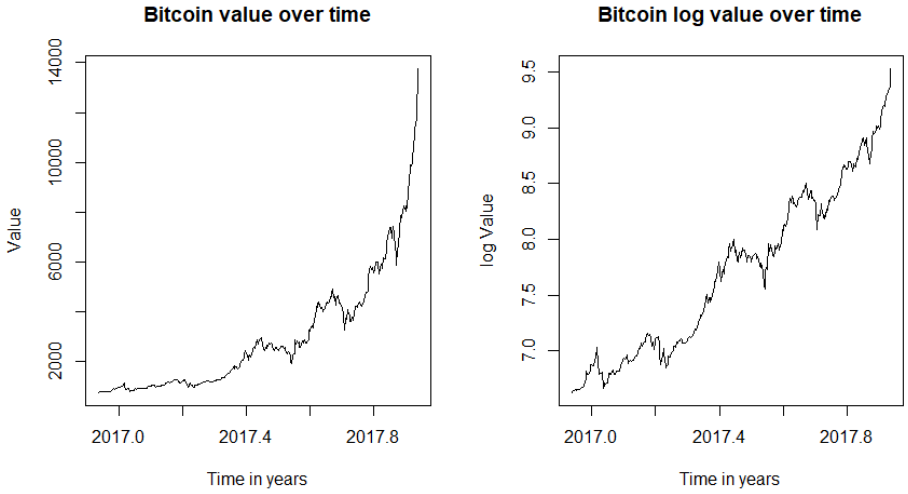

Figure 4.12: The same graphs as in figure 4.11, but with another time frame. We are looking here at the Bitcoin bubble, which burst in January/February 2018.

We see a huge increase of the value of Bitcoin at the end of 2017. We zoomed in on the data in figure 4.12. In January/February 2018 there was a huge loss in the value of Bitcoin, which can be seen as the bubble

bursting. This is very interesting for us, because we can try to apply our model to obtain a critical time which corresponds with reality. In the next paragraphs we will try to fit our model on this data frame from the beginning of 2017 till end of 2017. We will try to fit our model on both the value and the logarithmic value of Bitcoin.

4.2.1. Value of Bitcoin

When we apply the model on this data frame, we get an error when we ask to give us a summary with all the information about the estimations of the parameters. We do however get an estimation of the critical time which is 2017.937. This is our ending time of the data we took. This means that our model thinks that it is a huge bubble and will burst in the next instant.

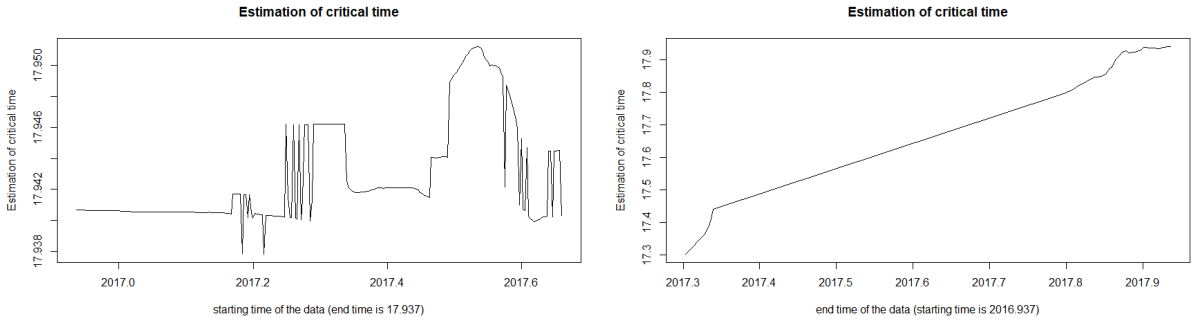

Figure 4.13: The graph on the left shows what happens with the estimation of the critical time when we vary the starting time of the data and keep the end time fixed. Notice that all the estimations are very close together. The graph on the right shows what happens with the estimation of the critical time when we vary the end time of the data and keep the starting time fixed.

Figure 4.13 shows us that varying the value of the starting time doesn't have any influence on the estimation of the critical time. It is always precisely the next instance after our last data point. When we vary our end time of the data we obtain that our model gives an estimation of the critical time always very close to our end time. With these two results combined we can conclude that our model is not working well on the data, because our model estimates the critical time to be the next moment after the end time of the data we choose. This is probably because the value of Bitcoin is very volatile and our model recognizes the data as a bubble all the time. However we simulated markets just as volatile as Bitcoin in chapter 3 and our model didn't give estimations of the critical time close to the end time of the data. So it remains unclear why our model is not working on Bitcoin, but it must have something to do with the properties of Bitcoin like it's volatility.

4.2.2. Logarithmic value of Bitcoin

When we look at the logarithmic value of Bitcoin we have less volatile data. This could help us getting better results. So to avoid the errors we get when looking at the value of Bitcoin we will look at the logarithmic value of Bitcoin instead. When we do this we don't get the error and we obtain a summary in R. These are the estimated parameters:

Parameter	Estimate	standard error	t-value	P-value
A	891.8	18480	0.005	0.996
B	-883.2	18480	-0.005	0.996
t_c	18.79	0.9009	20.857	$< 2e-16$
m	0.003766	0.7937	0.005	0.996
C_1	0.0001092	0.0236	0.005	0.996
ω	15.18	10.83	1.402	0.162
C_2	-0.03117	1.3116	-0..24	0.981

We find in the table that our only significant estimation is the estimation of the critical time, which is predicted to be 2018.79. However since all the other estimations of the parameter are very high this is not trustworthy at all. Also when we compare our estimation with reality we conclude that a prediction of 2018.79 is not good at all. Unfortunately also when we varied our starting time of the data or the starting values of the parameters, we did not get any trustworthy results. We wonder why this is the case and we will come back on this in the discussion.

5

Conclusion and Discussion

5.1. Conclusion

In the second chapter we found that the model is very sensitive to changes in starting time and end time of the data. We wanted to find out which model performs best. We found that formula (2.11) performs slightly better than the others in the results. We do know that it should perform better because formula (2.11) contains the least non-linear parameters which enhances the performance of the Levenberg-Marquard algorithm.
We tried to get the same estimations on Black Monday as Didier Sornette obtained in article [2]. We almost obtained the same estimation as Didier Sornette except for one of our parameters which was very off point. When we took a closer look at Didier Sornette's results we found that he might have made a typing error, which causes the differences between our results.
In chapter three we started simulating our own data. We got very good estimations for all parameters when we simulated the data ourselves. We looked at adding normally distributed noise and t-student distributed noise with four degrees of freedom. Our model provided a good fit when we simulated data for ordinary markets, but when we simulated more volatile markets our estimations were very sensitive to choices of the starting value of non-linear parameters.
Last but not least, we applied our model at ongoing markets. The first one is the Amsterdam Exchange Index (AEX). When we look at the past data of the value of the AEX it is a reasonable assumption that a bubble has been forming for some years already. We applied our model and found that the estimation of the critical time is very dependent on the starting value of the data we choose. This makes it very hard to give a conclusion about most probable time of the bubble bursting. However most estimations were in the year 2019 and 2020, so it would not be a surprise if we see a (big) loss in value of the AEX in 2019/2020.
The second market we looked at was Bitcoin. First we used model (2.11) to estimate the critical time. Secondly we used logarithmic value of Bitcoin in model (2.16) to estimate the critical time. Both times we got results which are not trustworthy. Unfortunately our model is not working properly on this data.

5.2. Discussion

Looking at the conclusions we have drawn, we used formula (2.11) mostly because we know that this model should work best, because it has the fewest non-linear parameters. When we tested if formula (2.11) works better than formula (2.10), we did get slightly better results. If we would have had more time we would have run more tests to investigate the performance of all models.
When we tried to get the same result as Didier Sornette got for estimating the parameters of for Black Monday, we got one estimation for parameter C which was not close to Didier Sornette's estimation at all. This might be a writing mistake of Didier Sornette. Unfortunately Didier Sornette does not release his code. So we were not able to check if he did something different than we did. Besides that it was unclear whether he used the opening or closing prices of the S&P 500. This gives however a very small change in estimations of the parameters. So this cannot explain the difference in estimation in parameter C.
When simulating our own data and adding noise we do get very nice estimations for the parameters close to the parameters we used for simulating the data especially in ordinary markets. However when we simulate the data, we add noise, but nonetheless does the 'form' of our model still remain in the data. In reality the data does not have to be in this 'form'. That is why we get very good results for simulated data and why it

more difficult to find good estimations in real markets. However it is great to see that our model and R code works properly on simulated data.
Looking at the Amsterdam Exchange Index (AEX) and Bitcoin gives us a hard time estimating the critical time. At the AEX the estimation of all parameters depends very much on the starting parameters we choose. Especially the starting value of the data we choose is very important as we get a lot of different estimations for the critical time. However a lot of the estimations for the critical time we got are in the year 2019/2020, so we would not be surprised if we see a big drop in value of the AEX in the coming two years. (Remember that the critical time is the time of a crash happening is the most probable, but not certain!)
For Bitcoin the estimations are untrustworthy. Looking both at the value of Bitcoin and the logarithmic value of Bitcoin give us estimations untrustworthy. This might have something with high volatility that Bitcoin has. This remains a little strange, because when we simulated our data with same volatility as Bitcoin we did get moderate-to-good results.
Not getting good results on all the data is not a bad thing at all. We learn from it and hope it inspires others to research this matter as well. In the end predicting financial bubbles is a very interesting topic, filled with many puzzles yet to be resolved.

A

R scripts

A.1. Bitcoin

```
library(minpack.lm)
bitcoin = read.csv("BTC-USD.csv",sep = ',')

#convert dates to numeric values
startdate <- as.Date(bitcoin$Date[1],"%Y-%m-%d")
Date <-  as.Date(bitcoin$Date,"%Y-%m-%d")
NumDays <- difftime(Date,startdate,units="days")

### Using the model to estimate parameters ###
time=as.numeric(10.5397+NumDays/365)[2335:2700] #max_date_is_2869, #2700 is end of bubble
sp = bitcoin$Close[2335:2700]

#When you want the logarithmic value of the price
sp=log(sp)

#define functions and starting parameters
price <- function(parS, xx) parS$a+(parS$b*(parS$c -xx)^parS$d)*(1+parS$e*cos(parS$f*log((parS$c-xx))))+
((parS$c -xx)^parS$d)*(parS$g*sin(parS$f*log((parS$c-xx))))
residFun <- function(p, observed, xx) observed - price(p, xx)
parStart = list(a=1000,b=-765,c=18.5,d=0.2,e=12,f=7.4,g=4.5)

#Levenberg-Marquardt algorithm
nls.out <- nls.lm(par=parStart,lower = c(-Inf, -Inf, 0, 0, -Inf, -Inf, -Inf),
                  upper =c(Inf, Inf, Inf, 1, Inf, Inf, Inf) , fn = residFun, observed = sp,
                  xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=1))
summary(nls.out)

### Two plots of value and log value of Bitcoin ###
par(mfrow = c(1,2))
time=as.numeric(2010.5397+NumDays/365)[2335:2700]
sp = bitcoin$Close[2335:2700]
plot(time,sp,type='l',xlab = 'Time in years', ylab='Value',main = 'Bitcoin value over time')
sp=log(sp)
plot(time,sp,type='l',xlab = 'Time in years', ylab='log Value',main = 'Bitcoin log value over time')


#### Calculating average fluctuation of bitcoin (daily price changes)  ###
change<-c()
sp = bitcoin$Close[1:2699]
for (i in 1:2699){
  change=c(change,abs((sp[i+1]-sp[i])/sp[i]))
}
time=as.numeric(2010.5397+NumDays/365)[1:2699]
plot(time,change,type='l',xlab='Time in years',ylab='Daily change',main='Daily price changes in bitcoin',ylim=c(0, 0.8))
average_change=sum(change)/2699
average_change
```

A.2. Black Monday

```
library(minpack.lm)
sp500 = read.csv("SP500.csv",sep = ',')

#convert dates to numeric values
startdate <- as.Date(sp500$Date[1],"%d/%m/%y")
Date <-  as.Date(sp500$Date,"%d/%m/%y")
NumDays <- difftime(Date,startdate,units="days")
time=as.numeric(85+NumDays/365)[110:665]
sp = sp500$Close[110:665]

#define functions
price <- function(parS, xx) parS$a+(parS$b*(parS$c -xx)^parS$d)*(1+parS$e*cos(parS$f*log((parS$c-xx)/parS$g)))
residFun <- function(p, observed, xx) observed - price(p,xx)

#start with parameters close to Sornette (2003, p.61)
parStart = list(a=400,b=-150,c=87.7,d=0.5,e=12,f=7.4,g=2.0)
nls.out <- nls.lm(par=parStart, fn = residFun, observed = sp, lower = c(-Inf, -Inf, 0, 0, -Inf, -Inf, -Inf),
                  upper =c(Inf, Inf, Inf, 1, Inf, Inf, Inf),
                  xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=1))
summary(nls.out)

#use the obtained parameters as input for nls()
par <- nls.out$par
df=data.frame(Price=sp,t=time)
nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t)/g))),data=df,start=par)
summary(nls.final)

#plot real prices with fit
plot(time,sp,type='l',xlab = 'Time', ylab='Value',main = 'Value of S&P500 index')
lines(time,price(nls.out$par, time), col=2)
```

A.3. Amsterdam Exchange Index

```
library(minpack.lm)
AEX = read.csv("AEX.csv",sep = ',',stringsAsFactors=FALSE)

#convert dates to numeric values
startdate <- as.Date(AEX$Date[1],"%Y-%m-%d")
Date <-  as.Date(AEX$Date,"%Y-%m-%d")
NumDays <- difftime(Date,startdate,units="days")
vector=c()
RSS=c()
time4=c()

time=as.numeric(1992.78+NumDays/365)[6042:6639]#[4269:6639]
aex = AEX$Close[6042:6639]
aex<-as.numeric(aex)
plot(time,aex,type="l")
#defining model
price <- function(parS, xx) parS$a+(parS$b*(parS$c -xx)^parS$d)*(1+parS$e*cos(parS$f*log((parS$c-xx))))
                            +((parS$c -xx)^parS$d)*(parS$g*sin(parS$f*log((parS$c-xx))))
residFun <- function(p, observed, xx) observed - price(p,xx)

for(i in 1:100){
  parStart = list(a=700,b=-55,c=2018.5+0.005*i,d=0.5,e=1,f=10,g=5.0)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = aex,
                      xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=1))
    summary(nls.out)
    #use the obtained parameters as input for nls()
    par <- nls.out$par
    df=data.frame(Price=aex,t=time)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
   summary(nls.final)}, error=function(e){})
  RSS<-c(RSS,nls.out$deviance)
  time4=c(time4,parStart$c)
  vector<-c(vector,par$c)
}
par(mfrow=c(1,2))
plot(time4,vector,type = 'l',ylab = "Estimation of critical time", xlab = "Value of starting parameter tc",
     main = "Different starting values of tc")
plot(time4,RSS,type = 'l', ylab = "Value of RSS", xlab = "Value of starting parameter tc", main = "")

#  Looking at other non linear parameter m
vector2=c()
RSS2=c()
time5=c()
for(i in 1:100){
  parStart = list(a=700,b=-55,c=2018.8,d=0+0.01*i,e=1,f=10,g=5.0)
  tryCatch({
```

```
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = aex,
                      xx = time, control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=1))
    summary(nls.out)
    #use the obtained parameters as input for nls()
    par <- nls.out$par
    df=data.frame(Price=aex,t=time)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    summary(nls.final)}, error=function(e){})
  RSS2<-c(RSS2,nls.out$deviance)
  time5=c(time5,parStart$d)
  vector2<-c(vector2,par$c)
}
par(mfrow=c(1,2))
plot(time5,vector2,type = 'l',ylab = "Estimation of critical time",
     xlab = "Value of starting parameter m", main = "Different starting values of m")
plot(time5,RSS2,type = 'l', ylab = "Value of RSS", xlab = "Value of starting parameter m", main = "")



##Looking at other non linear parameter omega
vector3=c()
RSS3=c()
time6=c()
for(i in 1:100){
  parStart = list(a=700,b=-55,c=2018.8,d=0.5,e=1,f=5+0.1*i,g=5.0)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = aex,
                      xx = time, control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=1))
    summary(nls.out)
    #use the obtained parameters as input for nls()
    par <- nls.out$par
    df=data.frame(Price=aex,t=time)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    summary(nls.final)}, error=function(e){})
  RSS3<-c(RSS3,nls.out$deviance)
  time6=c(time6,parStart$f)
  vector3<-c(vector3,par$c)
}
par(mfrow=c(1,2))
plot(time6,vector3,type = 'l',ylab = "Estimation of critical time",
     xlab = "Value of starting parameter omega", main = "Different starting values of omega")
plot(time6,RSS3,type = 'l', ylab = "Value of RSS", xlab = "Value of starting parameter omega", main = "")



#plotting shit with good parameters
parStart = list(a=700,b=-55,c=2018.8,d=0.5,e=1,f=10,g=5.0)
nls.out <- nls.lm(par=parStart, fn = residFun, observed = aex,
                  xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=1))
summary(nls.out)
#use the obtained parameters as input for nls()
par <- nls.out$par
df=data.frame(Price=aex,t=time)
nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
summary(nls.final)

#plot real prices with fit
plot(time,aex,type='l',xlab = 'Time in years', ylab='Value of the AEX index',main = 'AEX fit')
lines(time,price(nls.out$par, time), col=2)
```

A.4. Simulating data

A.4.1. Means and standard deviations

```
ibrary(minpack.lm)

#Defining functions and parameters
price <- function(parS, xx) parS$a+(parS$b*(parS$c-xx)^parS$d)*(1+parS$e*cos(parS$f*log((parS$c-xx))))+
                            ((parS$c -xx)^parS$d)*(parS$g*sin(parS$f*log((parS$c-xx))))
residFun <- function(p, observed, xx) observed - price(p,xx)
time3=c()
vector=c()
pars=list(a=4000,b=-50,c=530,d=0.5,e=0.1,f=10,g=0.1)
Dataa =c()
meanvector1=c()
sdvector1=c()
meanvector2=c()
sdvector2=c()
meanvector3=c()
sdvector3=c()

###       Double loop generating data and testing sensitivity of non linear parameters   ###
for(k in 0:100){
  time3=c()
  Dataa =c()
#Generating Data with the model
for (j in 0:500){
  Dataa<-c(Dataa,price(pars,j))
  time3<-c(time3,j)
}
#Adding noise
noise <- rnorm(length(Dataa),0,0.0365*pars$a) # generate the noise to add
noise2<- rt(length(Dataa),4)*(0.0365*pars$a)  # generate the t-student distributed noise
Dataa <- Dataa + noise #or Dataa <- Dataa + noise2 if you want t-student distributed noise

#looking at the sensitivity of tc regarding tc (parameter c)
vector1=c()
for (i in 0:10){
  parStart = list(a=4000,b=-50,c=505+5*i,d=0.5,e=0.1,f=10,g=0.1)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = Dataa,
                      xx = time3,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=-1))
    par <- nls.out$par
    df=data.frame(Price=Dataa,t=time3)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    vector1<-c(vector1,par$c)
  }, error=function(e){})
}
#hist(vector1,breaks=50,main='sensitivity of the critical time to starting parameter tc',xlab='Estimation of critical time')
meanvector1<-c(meanvector1,mean(vector1))
sdvector1<-c(sdvector1,sd(vector1))


#looking at the sensitivity of tc regarding m (parameter d)
vector2=c()
for (i in 0:10){
  parStart = list(a=4000,b=-50,c=520,d=0.1+i*0.08,e=0.1,f=10,g=0.1)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = Dataa,
                      xx = time3,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=-1))
    par <- nls.out$par
    df=data.frame(Price=Dataa,t=time3)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    vector2<-c(vector2,par$c)
  }, error=function(e){})

}
#hist(vector2,breaks=50,main='sensitivity of the critical time to starting parameter m',xlab='Estimation of critical time')
meanvector2<-c(meanvector2,mean(vector2))
sdvector2<-c(sdvector2,sd(vector2))


#looking at the sensitivity of tc regarding omega (parameter f)
vector3=c()
for (i in 0:10){
  parStart = list(a=4000,b=-50,c=520,d=0.5,e=0.1,f=5+i,g=0.1)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = Dataa,
                      xx = time3,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=-1))
    par <- nls.out$par
    df=data.frame(Price=Dataa,t=time3)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    vector3<-c(vector3,par$c)
  }, error=function(e){})
```

```
}
#hist(vector3,breaks=50,main='sensitivity of the critical time to starting parameter omega',xlab='Estimation of critical time')
meanvector3<-c(meanvector3,mean(vector3))
sdvector3<-c(sdvector3,sd(vector3))
}

###     Plotting histograms to look at the sensitivity    ###
par(mfrow=c(1,2))
hist(meanvector1,main="Varying critical time as starting parameter",xlab = "Mean")
hist(sdvector1,main="with 3.65% deviation in noise ",xlab = "Standard deviation ")
hist(meanvector2,main="Varying m as starting parameter",xlab = "Mean")
hist(sdvector2,xlab = "Standard deviation ",main='with 3.65% deviation in noise')
hist(meanvector3,main="Varying omega as starting parameter",xlab = "Mean")
hist(sdvector3,xlab = "Standard deviation",main='with 3.65% deviation in noise')

###   plotting all in one window   ###
par(mfrow=c(1,1))
hist(vector1,breaks=50,main='sensitivity of the critical time to starting parameter tc',xlab='Estimation of critical time')
hist(vector2,breaks=50,main='sensitivity of the critical time to starting parameter m',xlab='Estimation of critical time')
hist(vector3,breaks=50,main='sensitivity of the critical time to starting parameter omega',xlab='Estimation of critical time')
```

A.4.2. Best fit

```
library(minpack.lm)

#Defining functions and parameters
price <- function(parS, xx) parS$a+(parS$b*(parS$c -xx)^parS$d)*(1+parS$e*cos(parS$f*log((parS$c-xx))))
                            +((parS$c -xx)^parS$d)*(parS$g*sin(parS$f*log((parS$c-xx))))
residFun <- function(p, observed, xx) observed - price(p,xx)
time=c()
vector=c()
pars=list(a=4000,b=-50,c=530,d=0.5,e=0.1,f=10,g=0.1)
Dataa =c()
MIN1=c()
MIN2=c()
MIN3=c()


#Generating Data with the model
for (m in 0:500){
time = c()
Dataa = c()
for (j in 0:500){
  Dataa<-c(Dataa,price(pars,j))
  time<-c(time,j)
}
#Adding noise
#noise2<- rt(length(Dataa),4)*(0.005*pars$a)
noise <- rnorm(length(Dataa),0,0.005*pars$a) # generate the noise to add
Dataa <- Dataa + noise

#looking at the sensitivity of tc regarding tc (parameter c)
time1=c()
vector1=c()
RSS1=c()
for (i in 0:10){
  parStart = list(a=4000,b=-50,c=505+runif(1, 0, 50),d=0.5,e=0.1,f=10,g=0.1)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = Dataa,
                      xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=-1))
    par <- nls.out$par
    df=data.frame(Price=Dataa,t=time)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    RSS1<-c(RSS1,nls.out$deviance)
    vector1<-c(vector1,par$c)
    time1=c(time1,parStart$c)
  }, error=function(e){})
}
MINRSS1<-min(RSS1)
MIN1<-c(MIN1,vector1[match(MINRSS1,RSS1)])

#looking at the sensitivity of tc regarding m (parameter d)
vector2=c()
time2=c()
RSS2=c()
for (i in 0:10){
  parStart = list(a=4000,b=-50,c=520,d=0.1+runif(1, 0, 0.8),e=0.1,f=10,g=0.1)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = Dataa,
                      xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=-1))
    par <- nls.out$par
    df=data.frame(Price=Dataa,t=time)
```

```
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    RSS2<-c(RSS2,nls.out$deviance)
    vector2<-c(vector2,par$c)
    time2=c(time2,parStart$d)
  }, error=function(e){})
}
MINRSS2<-min(RSS2)
MIN2<-c(MIN2,vector2[match(MINRSS2,RSS2)])
#looking at the sensitivity of tc regarding omega (parameter f)
vector3=c()
time3=c()
RSS3=c()
for (i in 0:10){
  parStart = list(a=4000,b=-50,c=520,d=0.5,e=0.1,f=5+runif(1, 0, 10),g=0.1)
  tryCatch({
    nls.out <- nls.lm(par=parStart, fn = residFun, observed = Dataa,
                      xx = time,control = nls.lm.control(maxiter=1000,ftol=1e-6, maxfev=1e6,nprint=-1))
    par <- nls.out$par
    df=data.frame(Price=Dataa,t=time)
    nls.final <- nls(Price~a+(b*(c -t)^d)*(1+e*cos(f*log((c-t))))+((c-t)^d)*g*sin(f*log(c-t)),data=df,start=par)
    RSS3<-c(RSS3,nls.out$deviance)
    vector3<-c(vector3,par$c)
    time3=c(time3,parStart$f)
  }, error=function(e){})
}
MINRSS3<-min(RSS3)
MIN3<-c(MIN3,vector3[match(MINRSS3,RSS3)])
}
hist(MIN1, ylab = "density", xlab="Estimation␣of␣the␣critical␣time", main = "Choosing␣tc␣random",breaks = 20)
hist(MIN2, ylab = "density", xlab="Estimation␣of␣the␣critical␣time", main = "Choosing␣m␣random",breaks = 20)
hist(MIN3, ylab = "density", xlab="Estimation␣of␣the␣critical␣time", main = "Choosing␣omega␣random",breaks = 20 )
par(mfrow=c(1,3))

shapiro.test(MIN1)
shapiro.test(MIN2)
shapiro.test(MIN3)
mean(MIN1)
mean(MIN2)
mean(MIN3)
sd(MIN1)
sd(MIN2)
sd(MIN3)
plot(time1,vector1,type = 'l',xlab = "Value␣of␣starting␣parameter␣tc"     ,ylab="Value␣of␣critical␣time",main="Estimation␣of␣critical␣time")
plot(time1,RSS1,type = 'l',xlab= "Value␣of␣starting␣parameter␣tc",ylab="Value␣of␣RSS",main="")

plot(time2,vector2,type = 'l',xlab = "Value␣of␣starting␣parameter␣m"      ,ylab="Value␣of␣critical␣time",main="Estimation␣of␣critical␣time")
plot(time2,RSS2,type = 'l',xlab= "Value␣of␣starting␣parameter␣m",ylab="Value␣of␣RSS",main="")

plot(time3,vector3,type = 'l',xlab = "Value␣of␣starting␣parameter␣omega"  ,ylab="Value␣of␣critical␣time",main="Estimation␣of␣critical␣time")
plot(time3,RSS3,type = 'l',xlab= "Value␣of␣starting␣parameter␣omega",ylab="Value␣of␣RSS",main="")
```

Bibliography

[1] Wanfeng Yan Wei-Xing Zhou Didier Sornette, Ryan Woodard. Clarifications to questions and criticisms on the johansen-ledoit-sornette financial bubble model. *Physica A*, 392:4417–4428, September 2013. doi: https://doi.org/10.1016/j.physa.2013.05.011.

[2] D.Sornette. Critical market crashes. *Physics Reports*, 378(1):1–98, April 2003. doi: https://doi.org/10.1016/S0370-1573(02)00634-8.

[3] Petr Geraskin & Dean Fantazzini. Everything you always wanted to know about log-periodic power laws for bubble modeling but were afraid to ask,. *The European Journal of Finance*, pages 366–391, 2013. doi: https://doi.org/10.1080/1351847X.2011.601657.

[4] Yahoo finance. Aex, 12:03, June 18, 2018. URL `https://finance.yahoo.com/quote/%5EAEX/history?p=%5EAEX`.

[5] Yahoo finance. Bitcoin, 12:03, May 24, 2018. URL `https://finance.yahoo.com/quote/BTC-USD/history/`.

[6] Yahoo finance. S&p 500, 15:20, May 23, 2018. URL `https://finance.yahoo.com/quote/%5EGSPC/history?p=^GSPC&.tsrc=fin-srch-v1`.

[7] David Gurney. Student t distribution vs normal distribution, 10:48, July 17, 2018. URL `https://www.geogebra.org/m/zxUzwgkH`.

[8] Jorge J. Moré. *The Levenberg-Marquardt algorithm: Implementation and theory*. Springer, 1978.

[9] Richard Shaw. S & p 500: 60 years of monthly and daily percentage price changes, 12:22, July 16, 2018. URL `https://seekingalpha.com/article/167991-s-and-p-500-60-years-of-monthly-and-daily-percentage-price-changes`.

[10] Didier Sornette. *Why Stock Markets Crash: Critical Events in Complex Financial Systems*. Princeton University Press, Princeton science library, 2002.

[11] D.Sornette V. Filiminov. A stable and robust calibration scheme of the log-periodic power law model. *Physica A*, 392:3698–3707, September 2003. doi: https://doi.org/10.1016/j.physa.2013.04.012.