April 27, 2019

PHYS 6006 Project Summary Report

Speeding up Stochastic Volatility Simulations for Financial Markets

Daniel J. Rose 27568326

University of Southampton
Andreas Jüttner

1 Introduction

Financial institutions are responsible for carefully managing large amounts of assets from different sources, and constantly face the risk that some unexpected event could suddenly cause the value of these assets to change. Companies try and work out how likely this is to happen, in order to predict the optimal time to invest in such assets. The return on an investment is a measure of how beneficial the investment was (how much money was made or lost), and a measure of the varying returns over time is known as the volatility.

To analyse volatility various scientific models can be used. A model is a simplified version of a real system, for example with no external influences. Traditional models (such as 'Black-Scholes options') assume that the volatility does not change over time, and while important for modern financial theory, this is not true in the real world. In this project then we assume that the volatility of an asset's price changes randomly, and so choose a model which simulates this more accurately.

2 Stochastic Volatility (SV) Model

The word 'stochastic' is used to define a random process that is unable to be predicted precisely, and therefore it is this model that we choose for this project. The stochastic volatility model itself is given by a pair of SV equations, and is governed by three parameters (a parameter is just a number, the value of which characterises the behaviour of a system).

The initial aim of our project then, is to write an algorithm (computer program) capable of extracting values for the three parameters from volatility data. In order to determine that our algorithm works we choose some values for the three parameters, and use these values along with the SV equations mentioned before to generate a set of artificial volatility data (the artificial data we mainly use for this project is shown in Figure 1).

When we apply our algorithm to this artificial data it calculates the value of each of the three parameters. If the calculated parameters match the values that we chose to generate the artificial volatility data then the algorithm has functioned correctly.

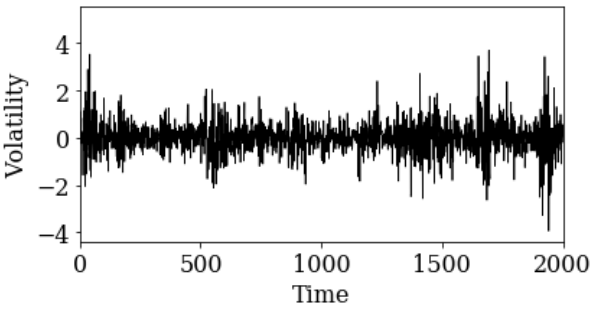

Figure 1: Our artificial data, where a larger spread means higher volatility at that time.

3 Hybrid Monte Carlo Algorithm

The type of algorithm chosen is known as a 'Hybrid Monte Carlo', as it consists of several parts.

First we need to determine the probability distribution of our stochastic volatility model, and for this we use Bayesian statistics. A probability distribution tells you how likely different values are, and Bayesian statistics are a mathematical approach where you constantly update probabilities as more information is collected. We use a 'Monte Carlo' method to do this, which is a type of algorithm which estimates an unknown quantity by repeatedly generating random values based on a probability distribution.

The second part is a 'Markov chain', a sequence of random events where each event depends only on the previous event (and not the

rest of the sequence). These two parts are combined to form a 'Markov chain Monte Carlo' algorithm using an acceptance check, where the chance of a new event being accepted or rejected is calculated from our model's probability distribution.

However, it proves difficult to apply this directly to part of our stochastic volatility model as the corresponding probability distribution is so complicated. Finally then we must also combine what are essentially quantum mechanical energy calculations in order to correctly model our probability distribution, and now have our Hybrid Monte Carlo algorithm.

4 Testing and Results

We apply our algorithm to the artificial data shown in Figure 1, and confirm that it correctly extracts the chosen parameters to an acceptable degree of accuracy. We also test our algorithm on other artificial data generated using different values for the three parameters, and confirm that our algorithm works for a range of input values.

We find that there is a limit to this range, and our algorithm fails when the chosen parameters are either too large or too small. However when this is the case the artificial data is not representative of real volatility data, and so this is not concerning. We have now achieved our initial aim, and move onto our ultimate goal of speeding up our algorithm.

To do this we look at how the calculated parameters vary during each repeated step of our algorithm. If subsequent steps show similar values, we say that they have a certain amount of correlation. We measure this using 'autocorrelation time', and our algorithm essentially works better the lower the parameter autocorrelation time.

5 Improving the Algorithm

We make two improvements to our algorithm. The first enables us to modify how random each next step of the algorithm is from the last, using a control value β (smaller β is less random). The second attempts to predict what the next several values should be, up to a maximum number of predictions k.

Again using the same artificial data in Figure 1, we apply our algorithm while varying both β and k. The autocorrelation times for one of the three parameters are shown in the contour plot in Figure 2, and we find an optimum setting for our algorithm using $\beta = 0.7$ and $k = 5$, giving roughly a 60% reduction in the autocorrelation time. For the other two parameters, one has similar results to Figure 2, while the other does not change by much.

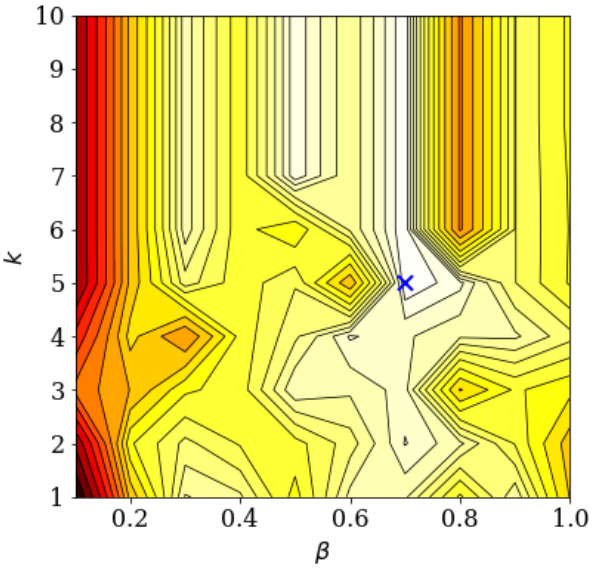

Figure 2: Autocorrelation times for one parameter, where a lighter colour signifies a lower time. The blue cross marks the maximum reduction in autocorrelation time.

6 Conclusion

In this project we have chosen a stochastic model which closely represents real volatility data, and have built an algorithm capable of extracting the three controlling parameters from artificial data. If desired we could now apply our algorithm to real volatility data, and use the extracted parameters to predict future behaviour.

We have also improved our algorithm, and shown that we can optimise these improvements to decrease the parameter autocorrelation times by 60% (increasing the algorithm's performance). Because of this performance increase the improved algorithm can accurately extract the parameters faster, resulting in a shorter program run time.