Discovering the hidden structure of financial markets through Bayesian modelling

a thesis presented for the degree of

Doctor of Philosophy of Imperial College London

and the

Diploma of Imperial College

by

Théophile Griveau-Billion

Department of Mathematics

Imperial College

180 Queen's Gate, London SW7 2BZ

October 2020

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Signed: _____

Copyright

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Thesis advisor: Dr. Ben Calderhead, Dr. Mikko Pakkanen

Discovering the hidden structure of financial markets through Bayesian modelling

Abstract

Understanding what is driving the price of a financial asset is a question that is currently mostly unanswered. In this work we go beyond the classic one step ahead prediction and instead construct models that create new information on the behaviour of these time series. Our aim is to get a better understanding of the hidden structures that drive the moves of each financial time series and thus the market as a whole.

We propose a tool to decompose multiple time series into economically-meaningful variables to explain the endogenous and exogenous factors driving their underlying variability. The methodology we introduce goes beyond the direct model forecast. Indeed, since our model continuously adapts its variables and coefficients, we can study the time series of coefficients and selected variables. We also present a model to construct the causal graph of relations between these time series and include them in the exogenous factors.

Hence, we obtain a model able to explain what is driving the move of both each specific time series and the market as a whole. In addition, the obtained graph of the time series provides new information on the underlying risk structure of this environment. With this deeper understanding of the hidden structure we propose novel ways to detect and forecast risks in the market. We investigate our results with inferences up to one month into the future using stocks, FX futures and ETF futures, demonstrating its superior performance according to accuracy of large moves, longer-term prediction and consistency over time. We also go in more details on the economic interpretation of the new variables and discuss the created graph structure of the market.

Acknowledgments

I would like to thank my supervisors Dr. Ben Calderhead and Dr. Mikko Pakkanen for their valuable feedbacks, support and help at every stage of my PhD journey.

I am also greatful to the great community of the Mathematics department at Imperial College London for creating such an enriching and stimulating environment. Our intellectual and entertaining conversations, inside and outside of mathematics, had an invaluable impact on this journey.

Lastly, special thanks to Gabrielle and my family for the unconditional love and support they have shown me over the years which helped me throughout the ups and downs of the PhD.

Contents

1 Introduction **1**
- 1.1 The motivation behind this thesis . . . 1
- 1.2 An overview of the construction of our models and their results 4
- 1.3 Thesis outline . . . 9
- 1.4 List of available Publications . . . 12

2 Financial Time series **13**
- 2.1 Introduction . . . 13
- 2.2 Stylised facts on financial time series . . . 14
  - 2.2.1 Different asset classes and markets . . . 14
  - 2.2.2 Different trading behaviours for different frequencies . . 16
  - 2.2.3 Stationarity . . . 17
- 2.3 Main time series model . . . 17
  - 2.3.1 AR + MA = ARMA . . . 17
  - 2.3.2 Dynamic state space model . . . 18
  - 2.3.3 Multivariate approach . . . 19
- 2.4 How to obtain the covariance matrix . . . 20
  - 2.4.1 Empirical computation . . . 20
  - 2.4.2 Inverse Wishart distribution . . . 20
- 2.5 Kalman filtering . . . 21
  - 2.5.1 Theory . . . 21
  - 2.5.2 Documented issues . . . 23
  - 2.5.3 Extension to non-linear environment . . . 23

3 Volatility modelling **25**
- 3.1 Introduction . . . 25
- 3.2 Prerequisites on Volatility modelling . . . 27
  - 3.2.1 Stylised facts of volatility . . . 28
  - 3.2.2 Base models . . . 30
- 3.3 Volatility vs. time lag . . . 34
  - 3.3.1 Highlighting the market heterogeneity . . . 34
  - 3.3.2 The variogram . . . 36
- 3.4 Markov Switching Multifractal . . . 39
  - 3.4.1 Description of the model . . . 40

3.4.2 Algorithms to estimate the parameters of MSM 42
3.4.3 MSM vs. variogram 42
3.5 Statistical feedback model 43
3.5.1 Multi time-scale statistical feedback volatility model . . 43
3.5.2 Presentation of the model and q-Gaussian distribution 46
3.5.3 The existing methods to estimate the parameter q . . . 50
3.5.4 Estimating the parameter q with σ unknown 52
3.5.5 Results 56
3.6 The limit of the classic volatility models 59
3.6.1 The limitations of GARCH models 59
3.6.2 The link between physics and volatility 60
3.6.3 The HAR-RV model of Corsi 61
3.6.4 How can we model cross-series relationships? 61
3.6.5 The SGDLM model 62

4 The Heterogenous-SGDLM model 64
4.1 Introduction 64
4.2 Techniques to approximate a distribution 65
4.2.1 Importance Sampling 65
4.2.2 Variational Bayes 66
4.2.3 Kullback-Leibler divergence 66
4.3 SGDLM model 69
4.3.1 Description of the model 69
4.3.2 Parents update 71
4.3.3 Posterior update 72
4.3.4 Recoupling with importance sampling 73
4.3.5 Decoupling with variational Bayes 74
4.3.6 Inference 75
4.3.7 Parameters sensitivity 75
4.4 The Heterogeneous AR model explained 76
4.4.1 The motivations for the model 76
4.4.2 The original HAR-RV 77
4.4.3 The extended version for leverage and volatility-of-volatility 79
4.5 A complete model of the market 79
4.5.1 The motivations for the model 80
4.5.2 Extending the standard DLM 81
4.5.3 Add OHLC data 82
4.5.4 The implementation on a GPU 83
4.6 Results 85
4.6.1 On a simulated environment 85
4.6.2 Description of the dataset 85

4.6.3 Direct inference . . . 86
4.6.4 Evolution of the coefficients . . . 88
4.6.5 Change point algorithm . . . 91
4.6.6 Risk metric application . . . 95
4.7 Application of the algorithm on different environments . . . 98
4.7.1 Application on the stocks from the S&P500 . . . 98
4.7.2 Applying the model to predict prices of futures . . . 101
4.8 Conclusion . . . 104

5 Mean reverting portfolio **106**
5.1 Introduction . . . 106
5.2 Description of the sparse MRP optimisation problem . . . 109
5.2.1 The MRP as a constraint minimisation problem . . . 110
5.2.2 Previous techniques used to solve this optimisation . . . 112
5.3 New formulation of the MRP with H-SGDLM . . . 113
5.3.1 Description of the H-SGDLM model . . . 113
5.3.2 Sparsity with the H-SGDLM . . . 117
5.3.3 MRP reformulated with H-SGDLM data . . . 118
5.4 Solving the minimisation problem with CCD . . . 119
5.4.1 The CCD algorithm for non-convex function . . . 119
5.4.2 The CCD algorithm applied to sparse mean reverting portfolios . . . 121
5.5 Trading a Mean Reverting Portfolio . . . 123
5.5.1 Jurek and Yang trading strategy . . . 123
5.5.2 Ornstein-Ulhenbeck based trading strategy . . . 124
5.5.3 Mixed mean reversion trading strategy . . . 124
5.6 Results . . . 125
5.6.1 US stocks from the S&P500 . . . 125
5.6.2 FX Futures . . . 126
5.6.3 ETF Futures . . . 129
5.7 Conclusion . . . 130

6 Causal Graph Process **132**
6.1 Introduction . . . 132
6.2 Prerequisite on graph and causality . . . 134
6.2.1 Granger causality . . . 135
6.2.2 Structure learning . . . 136
6.2.3 Signal processing on graphs . . . 138
6.3 Estimating the adjacency matrix with coordinate descent . . . 140
6.3.1 Update equation for $i > 1$ . . . 140
6.3.2 CCD for $i = 1$ . . . 141

6.3.3 Retrieving the polynomial coefficients c . . . 143
6.4 Selecting the L_1 coefficient . . . 144
6.4.1 Two new distance measures for selecting the L_1 coefficient . . . 145
6.5 Applications . . . 146
6.5.1 Computation time complexity . . . 150
6.6 Application to financial time series . . . 152
6.7 How to learn the covariance structure . . . 156
6.7.1 Graphical-LASSO approach . . . 157
6.7.2 Extend the CGP-CCD to simultaneous relations . . . 158
6.7.3 Results of the simultaneous CGP-CCD . . . 160
6.8 Summary of CGP-CCD . . . 160

7 Causal-Heterogenous-SGDLM **162**
7.1 Introduction . . . 162
7.2 Capturing both simultaneous and lagged causality . . . 163
7.2.1 Simultaneous parents . . . 164
7.2.2 Granger causality . . . 164
7.3 Adding Granger Causality to HSGDLM . . . 165
7.3.1 The extended model . . . 165
7.3.2 Implementation issues . . . 167
7.4 Results . . . 168
7.4.1 The datasets . . . 168
7.4.2 Direct predictions . . . 169
7.4.3 Evolution of the coefficients . . . 169
7.4.4 Change point algorithm . . . 171
7.4.5 Results on stocks from the S&P500 . . . 175
7.5 Conclusion . . . 177

8 Implementation on GPU **181**
8.1 Introduction . . . 181
8.2 How to optimise computation on a GPU . . . 182
8.2.1 The GPU architecture . . . 182
8.2.2 Example with the cholesky decomposition . . . 183
8.2.3 Example of batch computation on GPU for a multivariate Normal . . . 185
8.3 The main libraries . . . 187
8.3.1 PyTorch . . . 187
8.3.2 Tensorflow . . . 188
8.4 Computational issues . . . 188
8.4.1 How to fit dynamic vectors into fix-size tensors . . . 188

8.4.2 Positive semi-definiteness 189
8.5 Sparse computation in TensorFlow 191
8.5.1 Why use sparse computations 191
8.5.2 Constructing the indices 192

Appendix A Appendix for HSGDLM **194**
A.1 SGDLM equations 194
A.1.1 Posterior and Recoupling 194
A.1.2 Decoupling with Variational Bayes 195
A.1.3 Evolution to $t+1$ 195
A.1.4 Prior inference 196

References **202**

Acronyms

AIC Akaike Information Criterion.

AR Autoregressive.

ARCH Autoregressive Conditional Heteroscedasticity.

ARMA Autoregressive Moving Average.

BIC Bayesian Information Criterion.

CCD Cyclical Coordinate Descent.

CGP Causal Graphical process.

CPU Central Processing Unit.

DAG Directed Acyclic Graph.

DLM Dynamic Linear Model.

ETF Exchange Traded Funds.

EW Equally Weighted.

FFN Feed Forward Neural Network.

FX Foreign Exchange.

GARCH Generalised Autoregressive Conditional Heteroscedasticity.

GC Granger Causal.

GPU Graphics Processing Unit.

HAR Heterogeneous Autoregressive.

HARCH Heterogeneous Autoregressive Conditional Heteroscedasticity.

HSGDLM Heterogeneous Simultaneous Graphical Dynamic Linear Model.

IS Importance Sampling.

KF Kalman Filter.

KKT Karush–Kuhn–Tucker.

KL Kullback-Leibler.

LASSO Least Absolute Shrinkage and Selection Operator.

M-Z Mincer Zarnowitz.

MA Moving Average.

MAD Median Absolute Deviation.

MCMC Markov Chain Monte Carlo.

MLE Maximum Likelihood Estimator.

MRP Mean Reverting Portfolio.

MSE Mean Squared error.

MSM Markov Switching Multifractal.

O-U Ornstein–Uhlenbeck.

OHLC Open High Low Close data.

PCA Principal Component Analysis.

RMSE Root Mean Squared error.

RV Realised Variance.

SA Simulated Annealing.

SBM Stochastic Block Model.

SDP Semidefinite Programming.

SGDLM Simultaneous Graphical Dynamic Linear Model.

SGP Sparse Gradient Projection.

SV Stochastic Volatility.

TPU Tensor Processing Unit.

TVAR Time Varying Autoregressive.

VAR Vector Autoregressive.

VB Variational Bayes.

Glossary

endogenous the endonegous factors of a variable represent information specific to that variable. E.g. endogenous factor of a company represent information internal to that company..

exogenous the exonegous factors of a variable represent information exterior to that variable. E.g. exogenous factors on a company would be information external to that company, such as the stock market movement or the weather..

1

Introduction

1.1 The motivation behind this thesis

According to the efficient market hypothesis, see e.g. Malkiel (2003); Jensen (1978), the price of an asset represents all the available information on that asset at the time of trading. This information impacting the behaviour of each asset is composed of both endogenous factors representing the information specific to that asset and exogenous factors representing the influence of the market. However, if all traders have the same objective and access to the same information they should rationally decide on the same price and position to take on each asset which would not allow for any exchange. Thus, for an exchange to take place traders must have either different information, different understanding of that information or different objectives.

The heterogeneous market hypothesis considers that agents in the market trade with different objectives. While these objectives can be related to many characteristics, Müller et al. (1997) argue that most of them are reflected in the time horizon and highlight this fact by studying the impact of heterogeneous investment horizons with a volatility model using the returns

computed at different frequencies. Following this reasoning, the study of a time series at different frequencies should reflect the impact of endogenous factors on that time series.

On the other hand, for each asset the exogenous variables correspond to time series that have an impact on the behaviour of that asset. The classic approach involves creating a graph of cross-series relationships between the time series in a market based on the covariance matrix. However, the covariance is a symmetric object, while in reality some assets might influence many and others none, in a non-symmetric manner. Hence, a model that selects the exogenous factors without assuming such symmetry in the relationship may be more appropriate.

Combining these two sources of driving factors, our aim is to construct a model that decomposes each time series into its endogenous and exogenous parts. Having such a decomposition gives us a better understanding of what is driving each time series' behaviour and thus the market as a whole. A better structural understanding both for a single asset and the overall market allows us to produce more accurate inferences, stress indicators and provide new modelling of the risk structure of the market.

In order to extract as much economic information from the time series of stock prices as possible our model needs to include all possible factors but also be able to select between them, evolve with time and adapt to any environment without any hard parameters to tune. The endogenous factors represent the influence of past information of a time series on the future behaviour of that time series only. For example, the means of previous returns computed at different frequencies, the number of past negative returns during a specific time window or the evolution of the variance during the past week, month or years are variables that would fall into the endogenous category. However, putting too many variables which are potentially highly correlated between each others in the same regression could have a detrimental effect on the quality of that regression. Thus, because we want factors with a clear economic interpretation that aim will drive our selection of endogenous vari-

ables: we will select the variables that would give us the highest amount of information on what is driving the underlying time series. For example, having two variables one representing the past weekly returns and the other the past monthly returns would tell us which frequency is currently influencing the evolution of the time series.

The exogenous group consist of factors external to the time series under study but that impact its future behaviour. We will focus on the impact of the environment in which the stock is evolving, i.e. on the impact of the other time series. The connections between two time series are either simultaneous or lagged. A simultaneous relationship is commonly represented in a multivariate distribution by the covariance matrix where two variables have a non-zero covariance if the observation of one of them would give an information on the current state of the other. However, a non-zero covariance does not give any information on a possible lagged causal link. In time-series analysis we commonly study lagged causal relations through the Granger causality. A variable is consider to Granger cause another one if the observation of its past value improves the predictions of that variable. Because these two types of causal relations are complementary of each others we need to include both of them into our group of exogenous factors.

Both endogenous and exogenous factors are time series dependent, i.e. that each time series might have different factors. However, it is not possible to model each time series independently if we want to model simultaneous relationships. Thus, we need a multivariate model of the environment that is able to include different factors for each time series. Furthermore, due to the high number of possible factors our model should be able to dynamically select them. In other terms, the connections between the stocks might evolve with time and thus the factors in the corresponding model should evolve too. In addition to evolving factors, the coefficients of these factors should also be able to evolve with time. Indeed, the distribution of stock-prices evolve with time, the mean usually has a drifting factor and the variance shows different clusters of high and low states. In addition, we do not want to fix a certain time window on which to compute the coefficient at every time t

because the choice of that time window too might evolve with the stock-market conditions.

1.2 An overview of the construction of our models and their results

While the exogenous factors should contain both simultaneous and lagged coefficients, finding Granger causal relations between many time series in a large environment is a much harder problem then computing simultaneous connections. Thus, we start by looking for a multivariate model able to model simultaneous relations and different endogenous variables for each time series. After testing different volatility models we propose a model that extends the Simultaneous Graphical Dynamic Linear Model (SGDLM) of Gruber and West (2016a,b); Zhao et al. (2016); McAlinn and West (2016) to incorporate the heterogeneous autoregressive realised volatility (HAR-RV) model of Corsi (2004); Corsi et al. (2008); Corsi and Reno (2009). The HAR-RV model brings the selection of endogenous factors and the SGDLM model the simultaneous connections, flexibility of its framework and ability to sequentially update all the parameters with the Kalman filter. We call this model the heterogeneous simultaneous graphical dynamic linear model or HSGDLM in short.

The decomposition performed by the HSGDLM model can explain at any time which economic variables are likely to be driving the variability; for example, it may be due to the sector, the market or internal information, or at a particular frequency. This decomposition of the move into economically-meaningful variables and the capacity of the algorithm to follow their evolution creates new signals to study. In order to get the most out of these different signals we used a simple scale space change point algorithm, such that the correlations between these signals and the underlying time series allows us to perform more reliable inference for days and weeks ahead for each stock, and indeed the whole market. In addition, this model has proven to be an efficient market stress indicator and forecaster. While Gruber and West

(2016a) observed striking similarities between some metric of the SGDLM and the market stress index St. Louis Fed Financial Stress Index, our model appears to offer insight into the likely moves of this index weeks ahead.

In order to assess the performance of the HSGDLM model we look at the percentage of measured points that lie within the inferred confidence interval. We are especially interested by the figures obtained for large moves in the variance, in particular positive moves, since more than 68% of them are negative. We compare the percentage of correct predictions between algorithms for a confidence interval smaller than the move. If our novel approach for decomposing the variance into different groups of variables representing complementary information at different scales works, it should give us new insights on what is driving the market and thus be much better at predicting large moves. Indeed, when considering the realized variance of stocks in a group of 487 European stocks over 18 years from 2001 to 2019, the HAR-RV and SGDLM models correctly predicted only respectively 53.24% and 34.69% of the changes in variance bigger than 9.28%. In contrast, our model inferred 63.89% of them correctly and in a different environment with stocks from the S&P500 over the same period, with the same model and parameters, that figure is 65.93%.

In addition, the benefits of these added factors are not related to any specific market situation, since we show with a backtest that the performance of the new signals is constant through time. Instead of relying only on the direct one-step ahead inference from the multivariate model we leverage the fact that the HSGDLM model sequentially updates the coefficients with the Kalman Filter equation to study the evolution of these coefficients with time. More precisely, we use the multivariate HSGLDM model as a feature extractor to create time series of these coefficients. Then, we group the coefficients depending on the economic information they represent, e.g. endogenous or exogenous, and study this newly created time series. By studying the evolution of the time series of coefficients with a change point algorithm we can go beyond the one step ahead prediction and perform much longer term predictions such as one month ahead. While the performance of long term

prediction decreases compared to shorter term ones they observe the same consistency through time.

In addition, since each group of coefficient represents a specific economic factor, our model can answer more interesting structural questions such as: does the variance increase when the spread between the exogenous and endogenous variables increases or decreases? Furthermore, the flexibility of the statistical tools used by the SGDLM framework and the economic decomposition into endogenous and exogenous groups allows our approach to be applied to other financial metrics. We will demonstrate applications on modelling the prices of the ETF and FX futures with the same performance qualities: accuracy of large moves, longer-term prediction and consistency through time.

The HSGDLM model creates new information that can be leveraged in many different ways. In this work we present an application of these new information that extends the approach of d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) by characterising a sparse mean reverting portfolio according to its predictability but without assuming a VAR(p) process. Instead, we will leverage the results of the HSGDLM to set up the MRP optimisation problem for assets following time varying autoregressive TVAR(p) processes. The HSGDLM builds a sparse covariance matrix which is not assumed stationary and our new formulation of the MRP minimisation problem with normalisation constraint allows us to derive a CCD algorithm to solve it. The algorithm we propose can be applied to compute a MRP from any predicted covariance matrix. However, we focus in this paper on the predictions obtained from the HSGDLM model.

Using this new MRP solution we can then consider a number of different trading strategies. Hence we will numerically compare the approaches proposed by Jurek and Yang (2007) and Sipos and Levendovszky (2013); Fogarasi and Levendovszky (2013), as well as a combination of the two. In order to assess the performance and robustness of our solution we computed the sparse MRP and implemented the trading strategy on three different datasets: stocks from

the S&P500, futures on FXs and futures on ETFs. All of the backtests we obtained exhibit a linear growth throughout the time under study, which includes the financial crisis of 2008. While in some environments the backtests are smoother than others, the fact that none has a particularly large drawdown, and they are all roughly linear through time, highlights the robustness of our proposed approach.

While the HSGDLM model includes simultaneous connections it lacks lagged causal relations. As detailed before, finding the Granger causal relations between a large set of time series is a hard problem. In this work we follow the idea of the graph-LASSO of Friedman et al. (2008) and the individual LASSO regularization of Meinshausen and Bühlmann (2006) to propose a new CCD algorithm that solves the causal graph process (CGP) problem of Mei and Moura (2015, 2017). The CCD approach allows us to leverage the knowledge of the specific structure of the adjacency matrix to optimise the computational steps. In addition, we propose a new metric that uses the prediction quality of each node separately to select the LASSO coefficient. Our solution does not require additional parameters and produces better results than relying solely on the prediction error. Thus, our algorithm computes the directed adjacency matrix with an appropriate number of edges, as well as the polynomial coefficients of the CGP. Furthermore, the quality of the results and speed are not affected by the sparsity level of the underlying problem. Indeed, while the algorithm proposed by Mei and Moura (2015, 2017) scales cubically with the number of nodes, the algorithm we propose scales quadratically while automatically selecting the LASSO coefficient. We show the performance of our solution on simulated CGPs following a stochastic block model with different sizes, levels of sparsity and history lengths. We assess the quality of the estimated adjacency matrix by considering the difference in number of edges, the percentage of correct positives and the percentage of false positives.

We want to include in the DLM of each time series factors representing the lagged causal impact of other stocks in the dataset. Since the CGP-CCD algorithm we propose is able to learn the Granger causal relations in

a large set of time series it will allow us to select for each time series which causal variables to include. Thus, we follow the two steps approach of the SGDLM model to select the simultaneous coefficients of each DLM. Hence, combining these different models creates a multivariate model of the market that includes both endogenous and exogenous factors with both lagged and simultaneous connections. Figure 7.1 shows the evolution of the different groups averaged over all the time series in that dataset. The rv and r groups correspond to endogeneous coefficients while *cgp* and *core* are the groups of respectively lagged and simultaneous causal relations. Figure 7.1 plots on the right axis the evolution of the variance of the market, hence we can observe on this figure how each group evolve with time and compare to the market. We assess the importance of these added factors through the one-step-ahead direct prediction quality and the long term predictions using the group of CGP coefficients. On a dataset of European stocks we observe an increase in forecast quality of adding the causal factors both for the direct one step ahead prediction and backtests with the change point algorithm.

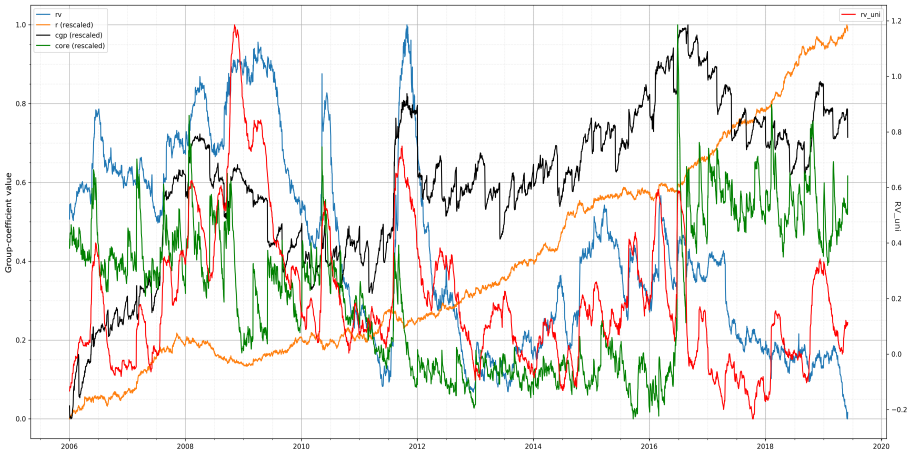


Figure 1.1: Plots computed with data from 500 European stocks. The graphic shows the evolution of the different groups (endogenous, with RV and R, and exogenous, denoted Core) rescaled to be on the same $y-axis$, which is left on the figure. The right axis corresponds to the approximated $log(RV)$ of the market, i.e. an equal weighted average of the individual $log(RV)$. This graphic highlights the connections between the behaviour of the market, represented here by $log(RV)$, and the different groups: rv, r, cgp and $core$. For example, looking at the volatility jump of 2011 the rv and $core$ group seem positively correlated with the market while the leverage group seems negatively correlated.

1.3 Thesis outline

In Chapter 2 we start by detailing the stylised facts of financial time series that we would like our model to capture. Then, we detail the base time series model that will form the basis of our more complex model and different methods to compute the covariance matrix from noisy observations. The last section of this chapter details the Kalman Filter which we use in our main model to sequentially update the moments of the distributions with each new observation.

Chapter 3 focuses on volatility modelling by detailing the stylised facts of the volatility time series of stocks, then the base model developed to capture these facts. The more complex models we propose later in this work will combine different extensions from these base models. Then, we study the evolution of the volatility according to different time scales and how these observation can be linked to economic behaviours. While this first study produces interesting results it does not provide a complete model of the volatility. Thus, we study two more complex models of volatility for which we present new methodologies to estimate them: the Markov Switching Multifractal and Statistical Feedback models. However, since these models do not meet our criteria we do not select them for further study. We detail in the last section of this chapter the limits of the current volatility models which led us to develop the HSGDLM model we propose in the following chapter.

In Chapter 4 we propose a heterogeneous simultaneous graphical dynamic linear model (H-SGDLM), which extends the standard SGDLM framework to incorporate a heterogeneous autoregressive realised volatility (HAR-RV) model. This novel approach creates a GPU-scalable multivariate volatility estimator, which decomposes multiple time series into economically-meaningful variables to explain the endogenous and exogenous factors driving the underlying variability. This unique decomposition goes beyond the classic one step ahead prediction; indeed, we investigate inferences up to one month into the future using stocks, FX futures and ETF futures, demonstrating its superior performance according to accuracy of large moves, longer-term prediction

and consistency over time.

The computational implementation is critical to leverage the full capability of the HSGDLM model we propose. Thus, in Chapter 8 we detail why GPUs are perfectly suited for our problem and how to use them. We detail how to implement a Cholesky decomposition to run on a GPU since it is the most computationally expensive step of the HSGDLM mode. Then, we explain the different libraries we use in this thesis that enable GPU computations. We further detail some computational issues we encounter during the implementation and the different fixes we developed.

As presented in Chapter 4 the HSGDLM model creates new information that have many applications. In Chapter 5 we detail the use of the HSGDLM results to solve the well known econometric challenge of finding sparse mean reverting portfolios based on a subset of a large number of assets. Many current state-of-the-art approaches fall into the field of co-integration theory, where the problem is phrased in terms of an eigenvector problem with sparsity constraint. Although a number of approximate solutions have been proposed to solve this NP-hard problem, all are based on relatively simple models and are limited in their scalability. In this chapter we leverage information obtained from the heterogeneous simultaneous graphical dynamic linear model and propose a novel formulation of the mean reversion problem, which is phrased in terms of a quasi-convex minimisation with a normalisation constraint. This new formulation allows us to employ a cyclical coordinate descent algorithm for efficiently computing an exact sparse solution, even in a large universe of assets, while the use of HSGDLM data allows us to easily control the required level of sparsity. We demonstrate the flexibility, speed and scalability of the proposed approach on S&P500, FX and ETF futures data.

The HSGDLM model is able to decompose the evolution of the time series into endogenous and exogenous factors. However, exogenous factors only contain simultaneous connections. Thus Chapter 6 proposes a novel algorithm for efficiently computing a sparse directed adjacency matrix from a

group of time series following a causal graph process. Our solution is scalable for both dense and sparse graphs and automatically selects the LASSO coefficient to obtain an appropriate number of edges in the adjacency matrix. Current state-of-the-art approaches rely on sparse-matrix-computation libraries to scale, and either avoid automatic selection of the LASSO penalty coefficient or rely on the prediction mean squared error, which is not directly related to the correct number of edges. Instead, we propose a cyclical coordinate descent algorithm that employs two new non-parametric error metrics to automatically select the LASSO coefficient. We demonstrate state-of-the-art performance of our algorithm on simulated stochastic block models and a real dataset of stocks from the S&P500 and European markets.

In Chapter 7 we combine the HSGDLM model introduced in Chapter 4 and the results of the CGP-CCD algorithm presented in Chapter 6. Combining these two models creates a multivariate model of the market for which each individual DLM includes both endogenous and exogenous group of factors with both simultaneous and lagged causal relations. We follow the methodology introduced to assess the HSGLDM model to study the performance of this causal-HSGDLM (CHSGDLM) model. We can osberve that the addition of the new causal factors improves the direct one step ahead predictions for any size of movements. Furthermore, the forecasts using the difference between the exogenous and endogenous factors is better both in terms of absolute performance and consistency of that performance through time compare to the model without causal factors.

1.4 List of available Publications

Griveau-Billion, T., & Calderhead, B. (2020). Efficient computation of mean reverting portfolios using cyclical coordinate descent. *Quantitative Finance*, 1–12.
http://doi.org/10.1080/14697688.2020.1803497

Griveau-Billion, T., & Calderhead, B. (2019). Efficient Structure Learning with Automatic Sparsity Selection for Causal Graph Processes (pp. 1–13). *Presented at the CausalML Workshop at 33rd Conference on Neural Information Processing Systems.*
https://arxiv.org/abs/1906.04479v2

Griveau-Billion, T., & Calderhead, B. (2019). Learning the causal graph structure of financial markets. *Presented at the DS3 Data Science Summer School.*,
https://2019.ds3-datascience-polytechnique.fr/wp-content/uploads/2019/06/DS3-634_2019.pdf

Griveau-Billion, T., & Calderhead, B. (2019). A Dynamic Bayesian Model for Interpretable Decompositions of Market Behaviour.
http://arxiv.org/abs/1904.08153v3

2

Financial Time series

2.1 Introduction

According to the efficient market hypothesis, see e.g. Malkiel (2003); Jensen (1978), the price of an asset represents all the available information on that asset at the time of trading. Hence, the patterns of a time series should contain all the information related to that asset. However, as theorised in the Heterogeneous Market Hypothesis of Müller et al. (1997), different investors have different objectives and will thus react differently to the same information. The time-series patterns of the different assets and markets depend on the type of investors but all these patterns follow the influence of either or both endogenous and exogenous factors. Section 2.2 details different stylised facts of financial time-series we are interested by in this work.

The two starting blocks to model a time-series are the Moving-Average (MA) and Autoregressive (AR) models which try to capture the exogenous and endogenous patterns respectively. The AR approach regresses the time-series at time t on its previous values while the MA approach models the value at time t as an average of past white noise values. Since a model will only

be able to detect effects it was built to, more complex combination of these two models have been developed to capture specific patterns of the trading behaviours. Section 2.3 details the models we will use in this work.

The exogenous effects of the environment on a time series are either simultaneous are lagged, Chapter 6 studies the lagged relations through Granger causality. The simultaneous connections between the different time series of an environment are commonly modelled in the covariance matrix. However, the computation of this covariances has to take into account the noise from the observations of the time-series, hence Section 2.4 details different approaches to compute the matrix of covariance.

Different techniques to delete the noise from noisy observations of a time series exist but they depend on the pattern the model wants to capture and more precisely the frequency of that pattern. To capture the targeted pattern we can use a filter that only let the frequency of interest through, e.g. when we are interested by the low-frequency movement of the process, the Kalman filter is widely used to study time series since it is the optimal filter in the case of linear Normal model with a quadratic loss function. Since we will use the Kalman filter to sequentially update the DLM equations we detail its implementation in Section 2.5.

2.2 Stylised facts on financial time series

2.2.1 Different asset classes and markets

The two most common types of non-exotic assets traded in the market are stocks and futures. A stock corresponds to a percentage ownership of a company while the future is a contract to deliver an underlying asset at a specified future date. Stocks are widely electronically traded through different stock-exchanges around the world. However, each stock-exchange has specific opening hours and trading day during which the exchanges take place. Hence, the trading is not continuous and different documented patterns linked to the trading and closing time exist. More precisely, a common

summary of a trading day uses not only the close price, but also the min, max and open prices. The close price corresponds to the last traded price at the end of the day during the closing auction, the open price to the first traded price at the opening of the day and min, max prices are the highest and lowest traded price observed during that day. Hence, if a news comes out during the night its impact will appear on the opening price of the following day.

The efficient market hypothesis, see e.g. Malkiel (2003); Jensen (1978), states that the price of an asset reflects all the available information on that asset at the time of trading. Hence, when a new information arrives outside the trading hours it is reflected in the opening price. Thus, the opening price is often different from the closing price and this difference is called the night effect. Since different investors may interpret the same information differently, they all compete in the opening auction during which investors specify the price and quantity they are willing to buy or sell that asset for and then the algorithm of the stock-exchange determines the optimal price matching the highest number of buyers and sellers. While we have described how the opening price is determined, the same system works for the closing auction to determine the closing price of the day, and the same heterogeneity of behaviour to an information impacts the trading price during the day. These phenomenons create well documented daily patterns, see for example Cartea et al. (2015). In the rest of this thesis when we speak about a stock price it refers to the closing price if not specified otherwise.

In addition to the patterns linked to the trading hours, stocks traded in different geographical regions behaves differently. This is mainly due to the heterogeneity of investors strategies and regulations of the stock-exchanges. For example, stocks for the S&P500 are known for their mean-reverting behaviour while European stocks from the EuroStoxx500 do not observe similar mean-reverting behaviour, e.g. see Fama and French (1988); Serletis and Rosenberg (2009). Another example, since the Italian stock-exchange has a higher trading fee, a trading strategy that rely on small average gains for thousands of trades might work in the UK but not in Italy. In addition to the

observable patterns due to the stock-exchange electronic processes and geographic particularities, we can observe many patterns linked to the frequency at which the financial time series is observed.

2.2.2 Different trading behaviours for different frequencies

For each asset, the order book corresponds to the number of stocks available to buy or sell at different prices specified by the traders. Each investor proposes to buy or sell a certain quantity of the asset at a specific price and an exchange happens when both quantity and price matches. The behaviour of the order book and its impact on the evolution of the price observed at high frequency are called microstructure effects. While a study of these effects is out of the scope of this thesis, it is interesting to note that some microstructure patterns are also observable at lower frequencies, Section 0.3 details such links.

If we compare the empirical distribution of the log-returns of the stock prices to the Gaussian distribution we observe a higher Kurtosis with a shape depending on the time scale and an empirical distribution that converges toward a Gaussian when the time scale increases, i.e. the frequency of the observations decreases. Interestingly, some low-frequency patterns related to the impact of human investors are observable; for example, many traders prefer to leave for the weekend with a book perfectly hedged, hence we can observe weekly price patterns linked to this behaviour. At a lower frequency, every year or quarter the book and its performance are assessed. Hence, at the end of the year or quarter we can observe a "window dressing" behaviour where investors adapt their book to improve its short-term performance. Window dressing can be done in different ways such as selling the worst performing stocks and buy high yield ones instead. Since the quaters and year end are the same for every investors, this window dressing behaviour is observable at those specific dates. New information arrives throughout the year and policies evolve with time thus, in addition to the evolution of the distribution according to the time scale the distribution for a fix time scale but at different dates might be different i.e. non-stationary.

2.2.3 Stationarity

The stationary property of a time-series process characterises the evolution of its probability distribution with respect to time. A stochastic process is said to be strong-stationary, or strict-stationary, if the unconditional cumulative distribution function at time t is the same at time $t+\tau$ for any τ. On the other hand, the weak-stationary condition requires that the first moment and the autocovariance stay constant with respect to time while the second moment is finite for all times. The following section will show why this property is often required by models of time series process but, empirically, for financial time series this property rarely holds. For example, a stochastic process with constant variance but trending mean is a simple model to represent stock-prices but is not stationary due to its first moment evolving with time.

2.3 Main time series model

2.3.1 AR + MA = ARMA

Let y_t be a stochastic process, an auto-regressive model of order p, $AR(p)$ is defined by:

$$y_t = c + \sum_{j=1}^{p} \phi_j y_{t-j} + \epsilon_t \ . \tag{2.1}$$

With c a constant and white noise $\epsilon_t \sim \mathcal{N}(0, \nu)$. The $AR(p)$ process is causal when the roots of the polynom $\phi(u) = 1 - \sum_{j=1}^{p} \phi_j u^j$ are outside the unit circle. When that condition is satisfied the distribution is also stationary but the contrary does not always holds. The auto-regressive model describes a process that has an endogenous behaviour, i.e. that can be accurately predicted by looking at its past behaviour.

A moving-average model of order q, $MA(q)$ is defined by:

$$y_t = \mu + \sum_{i=1}^{q} \psi_i \epsilon_{t-i} + \epsilon_t \ . \tag{2.2}$$

With μ the expectation of the process y_t. Hence, this process follows its average plus a deviation equals to a weighted average of white noise terms. Contrary to the AR process, moving-average processes are by construction stationary and while AR processes describe an endogenous behaviour, the moving-average approach models an exogenous behaviour, i.e. a process that is driven by exterior noise or a series of unobserved shocks that we model by white noise. Since these two models $AR(p)$ and $MA(q)$ are complementary Box et al. (2008) thought about combining them into the $ARMA(p, q)$ model:

$$y_t = c + \sum_{j=1}^{p} \phi_j y_{t-j} + \sum_{i=1}^{q} \psi_i \epsilon_{t-i} + \epsilon_t \ . \tag{2.3}$$

Hence this process is able to handle both long and short memory through different combinations of the parameters p and q. However, this model limits the evolution of the process y_t to a combination of its own passed value and white noise processes when for example state-space models allow for more freedom to include different exogenous and endogenous variables.

2.3.2 Dynamic state space model

A state-space model corresponds to a model with a state-equation describing the evolution of hidden states and an observation-equation describing the link between the observations and the hidden states. A dynamic-linear-model (DLM) is a form of time-varying state-space models which is defined by the quadruple $\{F_t, G_t, \nu_t, W_t\}$ that represents respectively the vector of variables, the state evolution matrix, the observation noise and the evolution noise. The DLM links the observations y_t with the states θ_t by the distributions:

$$y_t \sim \mathcal{N}(F_t\theta_t, \nu_t) \ , \tag{2.4}$$

$$\theta_t \sim \mathcal{N}(G_t\theta_{t-1}, W_t) \ . \tag{2.5}$$

With the noise terms being i.i.d and independent from each others. This model is more general than the $ARMA$ since the vector of variables F_t on which the process y_t will be regressed on can include any variables depending

on information available before time t. Furthermore, the vector of variables and the state evolution matrix depend on the time t which allow this model to evolve with time. Thus, a DLM does not assume the observation process y_t to be stationary and can update both moments of the observation and state distributions at each time t. Also, the quadruple $\{F_t, G_t, \nu_t, W_t\}$ only depends on the information available up to and including time $t-1$ which we define by $\mathcal{D}_{t-1}$. Hence, conditionally on $\mathcal{D}_{t-1}$ the probability distributions are defined and we can perform inference on the next time t. Furthermore, in this Gaussian setting if we use a quadratic loss function the different parameters of the DLM can be sequentially updated with the Kalman filter. However, the Equations 2.4 and 2.5 model a univariate process y_t when often multivariate relationships between time-series are important to model exogenous relationships.

2.3.3 Multivariate approach

Different time series evolving in the same environment, e.g. a group of sensors on the same machine, or a group of stocks from the same market, are often *connected* to each-others. However this word *connected* can have many different interpretations. For example, Chapter 6 studies lagged connections between time series with the Granger causality. In this section we will focus on simultaneous connections. In the case of multivariate distributions, the second moment, or covariance matrix, models the simultaneous relation between the different variables. For example, in a multivariate environment of dimension 2, if the covariance between these two variables is different from zero the observation of the first variable gives information on the behaviour of the second one and vice-versa. Hence, the importance of using a multivariate model instead of a group of univariate ones to model the time series from the same environment. Both the $ARMA$ or DLM model can be extended into a multivariate model with a covariance matrix modelling the simultaneous connections between the different time series. However, the computation of that covariance matrix is not always straightforward.

2.4 How to obtain the covariance matrix

2.4.1 Empirical computation

The second moment, or covariance matrix cov, of a multivariate distribution $Y = Y_i \; ; \forall i \in [1, N]$ with N the number variables, is defined by:

$$cov(Y_i, Y_j) = \mathbb{E}\left[\left(Y_i - \mathbb{E}\left[Y_i\right]\right)\left(Y_j - \mathbb{E}\left[Y_j\right]\right)\right] \; ; \forall i, j \in [1, N] \; .$$

Hence, in practice the covariance matrix for a group of time-series is often estimated by using a specific time-window of historical measures and estimating the expectation by the empirical mean over that time-window. However, the observed values are often noisy measures of the underlying process, thus the empirical matrix computed with estimated means is a noisy estimate of the true one. Different techniques exist to compute a covariance estimate taking the observed noise into account, for example see Section 0.7.1 which details the sparse projection with graphical-LASSO algorithm. Another approach is to consider a Bayesian setting with a prior-distribution that models the covariance matrix.

2.4.2 Inverse Wishart distribution

A prior is said to be the conjugate for the likelihood function if both the prior and posterior probability distributions belong to the same family of distributions. For example, if the likelihood is an univariate normal distribution with known mean but unknown variance, the conjugate prior distribution of the variance is the inverse-gamma for which both the prior and posterior distributions belong to the family of exponential distributions.

Following the same reasoning, for a multivariate normal distribution with known mean, the conjugate prior of the unknown covariance matrix is the inverse-Wishart distribution. Hence, when we model an environment of time series by a multivariate normal with unknown covariance, we assume an inverse-wishart distribution on the covariance. This is the reason why the

multivariate-DLM model assumes the mean to be a function of the states and the covariance to follow an inverse-Wishart distribution. Let us define the multivariate Normal-inverse-Wishart DLM by:

$$Y_t = F_t\Theta_t + \nu_t \ , \tag{2.6}$$
$$\Theta_t = G_t\Theta_{t-1} + \Omega_t \ . \tag{2.7}$$

Where the noise process ν_t follows a multivariate normal distribution $\mathcal{N}(0, \Sigma_t)$ and Ω_t follows a matrix normal distribution $\mathcal{N}(0, W_t, \Sigma_t)$. The distribution of the covariance matrix Σ_t conditional on the information available at $t-1$, D_{t-1}, is inverse-Wishart. This multivariate DLM system can be sequentially updated with the classic Kalman filter equations.

2.5 Kalman filtering

2.5.1 Theory

The Kalman filter (KF) minimises any quadratic loss function of errors in the predictions. Theoretically, the Kalman filter is the optimal filter if the following hypothesis are verified: the covariance of the noise is exactly known, the model perfectly matches the real system and the entering noise is white. Hence, the first step to check that the filter works correctly is to look at the distribution of the predictions' errors which should behave as a white noise if the filter works optimally. The filter sequentially updates its belief modelled by the parameters of the system with each new information. Let us define a state-space system with θ_t the hidden-states at time t and y_t the observations at time t by:

$$y_t = F_t\theta_t + \nu_t \ , \tag{2.8}$$
$$\theta_t = G_t\theta_{t-1} + \Omega_t \ . \tag{2.9}$$

Where the noises $\Omega_t \sim \mathcal{N}(0, W_t)$ and $\nu_t \sim \mathcal{N}(0, R_t)$ are independent of each others. The KF updates the parameters of the state-space model according to the performance of the previous predictions. At every time step t the

KF computes the prediction-errors and update the first two moments of both distributions with the values that minimise the residual error. Hence, at any time t the KF filter only stores in memory the previous first and second moments because all the knowledge of the previous time points are summarised in the current mean and covariance of each distribution. Let us define the a posteriori distributions of the state estimate and the covariance matrix by $\hat{\theta}_{t|t}$ and $P_{t|t}$ respectively, then the predictions correspond to the a priori distributions:

$$\hat{\theta}_{t|t-1} = G_t \hat{\theta}_{t-1|t-1} \ , \tag{2.10}$$

$$P_{t|t-1} = G_t P_{t-1|t-1} G_t^T + \Omega_t \ , \tag{2.11}$$

we can then compute the prediction-errors of the mean and the covariance:

$$e_t = y_t - F_t \hat{\theta}_{t|t-1} \ , \tag{2.12}$$

$$q_t = R_t + F_t P_{t|t-1} F_t^T \ , \tag{2.13}$$

The filter will update the a-posteriori distributions the state and covariance with the latest prediction-errors. In the linear Gaussian environment, no algorithm can do better than the Kalman filter since the optimal Kalman gain corresponds to the gain that minimises the residual error for an L_2 loss function. Hence, the Kalman gain corresponds to $K_t = P_{t|t-1} F_t^T q_t^{-1}$. The a-posteriori state and covariance updates use the Kalman gain combined to the measured errors as follow:

$$\hat{\theta}_{t|t} = \hat{\theta}_{t|t-1} + K_t e_t \ , \tag{2.14}$$

$$P_{t|t} = \left(I - K_t F_t\right) P_{t|t-1} \ . \tag{2.15}$$

Hence, the KF updates the previous knowledge of the mean and variance with the observed prediction-errors multiplied by the Kalman gain to minimise the expected quadratic loss. While these equations are exact analytic expression many numerical issues can arise in practice.

2.5.2 Documented issues

Many numerical issues of the KF are known and documented in the literature. For example, round-offs due to numerical precision can cause the a-posteriori covariance estimate $P_{t|t-1}$ to have negative diagonal values which can create a diverging state expectation since the Kalman gain will have the wrong sign. Thus, in practice we have to check the computation of the a-posteriori covariance for negative variances to ensure correct behaviour of the filter.

Furthermore, the covariance update of Equation 2.15 does not guarantee its symmetry or its positive-semi-definiteness. Hence, the implementation of the KF needs to incorporate non-singularity tests and repairs to guarantee the correct behaviour of the filter. In addition, one of the condition for the KF to perform optimally is for the model to perfectly match the underlying system and the KF update assumes a linear Gaussian environment which might not suit every system. Hence, if an important variable is missing from the system or the connections are not Gaussian the KF will not work optimally.

2.5.3 Extension to non-linear environment

The Kalman filter can be extended to avoid the restriction of a linear Gaussian system by replacing the linear relations with non-linear differentiable functions. Hence, with f and h two differentiable functions we can express the state space system by:

$$y_t = h(\theta_t) + \nu_t \,, \tag{2.16}$$

$$\theta_t = f(G_t, \theta_{t-1}) + W_t \,. \tag{2.17}$$

Due to these non-linearities, the computation of the Kalman update equations require an approximation of the derivation of these functions with respect to the states. In practice, this approximation often relies on the first order Taylor expansion to obtain the Jacobi matrix of the derivation of these functions as a function of the different states. Hence, while it is possible to extend the KF beyond a simple linear Gaussian state-space system, the

added burden of the extension loses the simplicity of the original solution and its optimality property which renders this option less attractive.

3

Volatility modelling

3.1 Introduction

The aim of this chapter is to detail the research process that brought our focus on HAR and SGDLM models and describes the pros and cons of mant classic volatility models. In addition, we detail the practical limits of the Markov Chain Multifractal and statistical feedback models, with parametrisation issues that are often overlooked in the literature.

While the log-normal distribution is often chosen to model the time-series of returns for its simplicity and ability to obtain a closed-form solution for many option-pricing formula it is too restrictive to model certain stylised facts. For example, empirical distribution of price-returns can be highly peaked, with fat-tails or both and the underlying process may not be stationary. In addition, many papers such as Müller et al. (1997); Muzy et al. (2000); Lynch and Zumbach (2003) study the multifractal behaviour and time asymmetry of financial returns which are two stylized facts that the widely used basic stochastic volatility (SV) or GARCH models do not reproduce. However, a simple model such as the Realised Variance (RV) can highlight interesting

volatility patterns. We show in Section 3.3.2 how studying the RV at different time scales through a variogram can capture some trading behaviour.

In Calvet and Fisher (2001, 2004) they extend the multifractal model of asset returns introduced by Mandelbrot to propose a stationary volatility model that can be estimated with maximum likelihood. Their markov switching volatility model is based on a multiplicative cascade of volatility components each representing a different frequency. However, while this model has a closed form likelihood its maximisation requires the use of an expensive optimisation algorithm and, most importantly, the parameters lack econometric interpretation.

The volatility cascade represents the propagation of the low-frequency volatility movements on to the higher-frequency volatility. The empirical observation of this cascade can be explained by the heterogeneous market hypothesis of Müller et al. (1997) which details that different market participants trade with different objectives that impact participants asymmetrically, one of them being the time scale. The economic interpretation is that short term investors are impacted by trades from long term investors more than the other way around. This asymmetric impact can be described by as a statistical feedback which led Borland (2002, 2004); Borland and Bouchaud (2005) to propose a return process with a noise component following the non-linear Fokker-Planck equation. This equation is solved by a q-Gaussian or Tsallis distribution that can model the evolution of the distribution for different time-scales and is time asymmetric.

The q-Gaussian distribution is interesting for many reasons, one of which being its ability to reproduce a wide array of distributions by simply varying the parameter q. For example, depending on q the distribution can have fat or bounded tails, peaks or none and with a specific value of q we can recover the student or normal distribution. In addition, the extension of the statistical feedback model to multiple time-scales can empirically reproduce multifractal effects even though it is an additive model.

The empirical distribution of financial time-series such as returns can evolve with time. Since in one calendar year there are about 256 trading days, to use a q-Gaussian distribution for the returns we need to be able to select the parameter q with a few hundreds data-points. In their papers Borland (2004); Borland and Bouchaud (2005) find the q that best fit the empirical observation of a time-series of returns over thousands of points by looking at specific plots. While this produces good results in simulation, in practice it is not viable to use time series of daily data with thousands of points, that would represent more than 4 years of history over which the distribution might have changed. If q is considered to be evolving with time we should fit q with as little amount of data as possible. In this work we present an algorithm that does not require any expensive optimisation step and can fit q to a time series with only hundreds of points but only with a precision of 0.1. In Borland (2004); Borland and Bouchaud (2005) they assume time-series of stock-price-returns to have a $q = 1.4$. However, in the environment of stocks from the S&P500 that we study the empirical distributions have a $q = 1.4$ on average but some distributions have much lower or higher q. Interestingly, the value of q is linked to the different patterns we observe in the variogram study and thus the different trading behaviours.

This chapter starts by detailing stylised facts of volatility processes and base volatility models in Section 6.2. Then, Section 3.3 introduces the variogram and uses it to study the connection between volatilities computed at different time scales and economic factors. After the observations obtained by this variogram study we implement the Markov Switching Multifractal model in Section 3.4 and compare its results with the variogram. Since the performance of the MSM model does not statisfy our criteria, in Section 3.5 we implement and propose a new algorithm to estimate the Statistical Feedback model. After studying the performance of these different approaches we summarise in Section 3.6 the limits of the current volatility models.

3.2 Prerequisites on Volatility modelling

3.2.1 Stylised facts of volatility

Time asymmetry

A time series is defined as a list of values observed at a specific time. However, time-series of stock-prices have the additional particularity of being asymmetric, i.e. not time reversal invariant. Zumbach (2009) provides an interesting study on the time reversal invariance and defines it by a system in which the transformation $t \to -t$ provides the exact symmetry of that system. In other words, for a time reversal invariant system, inverting the arrow of time should not create any measurable difference. However, stock prices are not time reversal invariant at any time t the past and future of the time series contain important information but not all models take that stylised fact into account. For example some ARCH or GARCH models are not time reversal invariant while most Stochastic volatility ones are.

Leverage effect

Time series of stock prices behave differently after a positive or negative returns: they tend to go down much faster than they go up. This phenomenon reflects itself into the volatility patterns with the volatility being negatively correlated with the returns, e.g. after a negative return the volatility is more likely to increase. This connection is called leverage effect and can be integrated into a volatility model in different ways. For example, the HAR model of Corsi et al. (2001) includes the leverage effect by adding the previous positive and negative returns as two different variables. Section 3.2.2 details another approach that adds to the SV and GARCH models a variable connecting directly the past return to the current volatility.

Volatility clustering

The volatility clustering of financial time series is a well documented phenomenon which describes the strong autocorrelaion of the volatility process. More precisely, a period of high volatility is more likely to stay high and a

period of low volatility to stay low. This is why the GARCH model became so popular. In the GARCH model, In addition to the heteroskedasticity, because the volatility process is assumed to follow an ARMA model it can also reproduce the volatility clustering effect.

Long memory

The long memory behaviour is another visualisation of the volatility clustering which describes the slow decay of the autocorrelation of squared and absolute returns. Since the squared and absolute returns are rough approximations of the variance of the time series this slow decay is a proxy to the long memory of the underlying volatility process. However, this stylised fact is not easy to model, e.g. a GARCH model has an autocorrelation declining exponentially. Long memory describes the distribution of a time series observed at one specified frequency, however observed at different frequencies the time series may have different behaviours hence the interest to study multifractality.

Multifractality

For a time series, multifractalility characterises the evolution of the absolute moment with respect to the time scale. Let us consider x to be a time series and x_{-l} to be the same time series shifted by l time steps, then the return at scale l is defined by $r_x(l) = x - x_{-1}$. A process is considered multifractal when the different moments scale as a power of the time scale with different scaling exponent. In other words, we can define the absolute moment of order n at time scale l by:

$$M_n(l) = \mathbb{E}\left[|r_x(l)|^n\right] = A_n l^{c_n} \ .$$

where A_n is a coefficient depending on the order n and c_n is the scaling exponent. If the scaling exponent c_n follows a linear expression, the process is considered monofractal, while in the non-linear case it is multifractal. Multifractality characterises how the distribution of the time series evolves for

different time scales. For example, the self-similar property of the brownian motion translates here to the monofractal case where the moments stay the same up to a rescaling. In practice, the time series of stock prices have a multifractal behaviour where the distribution becomes closer to a Gaussian when the time scale increases while it has much fatter tails for higher frequencies.

3.2.2 Base models

GARCH

As explained in the previous section describing the stylised facts, the H of both ARCH and GARCH models refers to *heteroskedasticity*. These models aim to capture the variation in the variance of a process and the GARCH is also able to model volatility clustering thanks to its memory term on the past values of the variance. The ARCH model considers the error term, i.e. the variance of the process to follow an autoregressive process $AR(q)$. Let us consider the error term $\epsilon_t^2 = \sigma_t^2 z_t^2$ where σ_t^2 corresponds to the variance and z_t is a strong white noise. A white noise process follows a distribution with zero mean and fixed variance, if the samples are considered uncorrelated we speak of *weak* white noise while if the samples are considered independent we call them *strong* white noise. Then the variance process of an ARCH model follows:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 \ ,$$

where all the coefficients α_i must be positive and additionally α_0 must be strictly greater than zero. This process links the variance to the size of the previous error terms but not to the previous values of the variance. Hence the GARCH extension which adds a moving average term on the variance to model the impact of the previous variances on the future one. With the addition of this memory the model is able to recreate the volatility clustering effect observable on financial time series. A $GARCH(p, q)$ models corresponds to volatility process following an $ARMA(p, q)$, so with an $MA(q)$

process on the error terms and a $AR(p)$ process on the variance:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 \ ,$$

While the ARCH and GARCH model the error term by a product of the variance with a Stochastic noise it does not fall into the class of Stochastic volatility model because at any time t the volatility it exactly determined with the previous observations up to $t-1$.

Stochastic volatility

Contrary to GARCH models the stochastic-volatility (SV) ones allows for separate error processes for both the mean and the error terms. Let us assume the underlying time series x_t to follow a geometric Brownian motion $dx_t/x_t = \mu dt + \sigma_t dW_t$ where W_t is a Brownian motion, μ the mean and σ_t the variance. Instead of a constant variance, a SV process considers the variance σ_t to follow its own stochastic diffusion with mean μ_σ and variance ν. Hence a basic SV model would be:

$$\begin{aligned} dx_t/x_t &= \mu dt + \sigma_t dW_t \ , \\ d\sigma_t &= \mu_\sigma dt + \nu dB_t \ . \end{aligned}$$

Where B_t corresponds to a Brownian motion independent to W_t. Many different extensions to this based model exists. But one of the key aspects of SV model is the correlation between the two Brownian motions W_t and B_t. The GARCH model is not part of the class of stochastic-volatility model per se because at any time t the conditional distribution of the volatility is an exact function of the previous values. While both SV and GARH models can be extended to model specific stylised facts of the underlying time series, estimating their parameters is not straightforward and can become very expensive for the extended models. To illustrate that point, the following section compares two extended SV and GARCH models.

Comparaison of GARCH and SV with leverage and jumps

In Nakajima (2009) they perform a Bayesian comparison of GARCH and SV models with jumps, leverage and heavy tails. That paper focuses on the modelling of leverage, jumps and heavy tailed error. They tried to implement a GARCH process with jumps that do not affect the volatility equation but it required too much time to compute. Hence, they work with an EGARCHJt(1, 1) process that includes leverage, heavy tails and jumps that do affect the volatility:

$$\begin{aligned}
y_t &= \mathbb{E}_{t-1}(y_t) + e_{t-1} \ , \\
e_t &= k_t\gamma_t + \sqrt{\lambda_t}\sigma_t\epsilon_t \ ; \ \epsilon_t \sim \mathcal{N}(0,1) \ ; \ t = 1, \ldots, n \ , \\
\ln(\sigma_t^2) &= \omega + \beta \ln(\sigma_{t-1}^2) + \theta\frac{e_{t-1}}{\sigma_{t-1}} + \alpha\left(\left|\frac{e_{t-1}}{\sigma_{t-1}}\right| - \zeta\right) \ ; \ t = 2, \ldots, n \ , \\
\text{with:} \quad & 0 < \beta < 1 \ ; \ \omega > 0 \ ; \ \alpha \geq 0 \ , \\
& \zeta = \mathbb{E}(|z|) \ ; \ z \sim \text{Student-t}(\nu) \ , \\
& \lambda_t^{-1} \sim \text{Gamma}(\nu/2, \nu/2) \ .
\end{aligned}$$

Where the jumps variable γ_t is defined as Bernoulli(κ), the jump size $k_t \sim \mathcal{N}(0, \delta^2)$ and $\ln(\sigma_1^2) = \omega/(1-\beta)$. They estimate the parameters with a MCMC chain instead of MLE for the model comparison with SV class. The discrete time stochastic volatility process with leverage, heavy tails and uncorrelated jumps is:

$$\begin{aligned}
y_t &= \mathbb{E}_{t-1}[\, y_t] \ + e_t \ , \\
e_t &= k_t\gamma_t + \sqrt{\lambda_t}\epsilon_t \exp(h_t/2) \ ; \ t = 1, \ldots, n \ , \\
h_{t+1} &= \mu + \phi(h_t - \mu) + \eta_t \ ; \ t = 1, \ldots, n-1 \ , \\
\text{with:} \quad & |\phi| < 1 \ , \\
& \begin{pmatrix} \epsilon_t \\ \eta_t \end{pmatrix} \sim \mathcal{N}(0, \Sigma) \ ; \ \Sigma = \begin{pmatrix} 1 & \rho\sigma \\ \rho\sigma & \sigma^2 \end{pmatrix} \ .
\end{aligned}$$

Where ρ measures the leverage effect and the jump size is defined by the relation: $\psi_t = \ln(1 + k_t) \sim \mathcal{N}(-0.5\delta^2, \delta^2)$. To obtain a simpler MCMC procedure they rewrite both models so the equations can be approximated as

a linear Gaussian state-space model with mixture component. Even though, the MCMC procedure for both models is very expensive to compute and limits the applications of these models. Using the marginal likelihood and Bayes factor Nakajima (2009) compares both models with and without the leverage effect and jump components. Overall, the models with leverage effect outperform those without but the inclusion of the jump component does not always increase the performance. Between the two approaches the SV model outperform the GARCH model because it is limited to including the jump component in the volatility term. However, when both models have the jumps affecting the volatility the GARCH version performs best. Their main conclusion from this comparison is an argument in favour of the leverage effect coefficients.

Realised Variance

The realised variance (RV) of a time series is defined as the sum of square returns computed at a certain frequency over a certain time window and is considered as a proxy of the underlying, unobservable quadratic variation of a stochastic diffusion process. The RV converges to the quadratic variation when the time scale of the returns tends to zero. Let Δ_i^n be a differentiation operator such that for a process X: $\Delta_i^l X = X_i - X_{i-l}$ where l corresponds to a time scale. Hence, we define the RV at time t and time scale l by:

$$RV_t^l = \sum_{i=t-(n-1)l}^{t} \left(\Delta_i^l X\right)^2 .$$

Where n corresponds to the size of the time window. Thus, for small time scale l the RV_t^l tends in probability to the quadratic variation of X when $l \to 0$. Hence, this metrics should theoretically be used on high frequency data with frequencies smaller than the minute scale. With high frequency data, the computation of the RV over a time window of one day gives a daily RV with which we can create a times series of daily-RV. In practice high-frequency data is very expensive to get, thus many practitioners use the RV metric with time series using a time scale l of one day or lower. Hence, they

obtain a monthly RV as being the RV computed using daily return over a time window of one month.

3.3 Volatility vs. time lag

3.3.1 Highlighting the market heterogeneity

The HARCH model

The HARCH model from Dacorogna et al. (1997); Müller et al. (1997) was the first to study the time-asymmetry in volatility measured with a model using different time scales. To reveal the impact on the volatility of the market components at different time horizons they study the interrelations of the volatility observed at different time scales. The HARCH model of Müller et al. (1997) used a linear combination of the squared aggregated returns but was too complex to estimate due to the high number of coefficients. Hence, Dacorogna et al. (1997) modified the HARCH to keep a smaller number of representative interval and use the exponential moving average (EMA) on each interval to obtain the EMA-HARCH. With r_t the log returns the EMA-HARCH defines the variance σ_t^2 as a weighted sum of variances computed at different scale:

$$\begin{aligned} r_t &= \sigma_t \epsilon_t \ , \\ \sigma_t^2 &= c_0 + \sum_{j=1}^{n} c_j \sigma_j^2(t) \ , \end{aligned}$$

With the variables σ_j^2 representing the contribution of the j-th component to the total market variance σ_t^2:

$$\sigma_j^2(t) = \mu_j \sigma_j^2(t - \triangle t) + (1 - \mu_j) \left(\sum_{i=1}^{k_j} r(t - i \triangle t) \right)^2 .$$

Where n corresponds to the number of time components and the EMA coefficient is defined as $\mu_j = \exp(-2/(k_{j+1} - k_j))$ with the aggregation factor

of the returns following $k_j = p^{j-2} + 1$. The necessary stationary condition of the model is $\sum_{j=1}^{n} k_j c_j < 1$. Computing the maximum likelihood (MLE) estimator of the model to obtain the coefficient c_j is not straightforward. They compute the MLE with a combination of a genetic algorithm and BHHH algorithm (a gradient descent variant). The genetic algorithm creates a first generation of parameters which is then used as the starting point for the gradient descend. Then, the algorithm iterates using the combined results from the BHHH and previous genetic step. Interestingly, even though this model was originally developed to show the time asymmetry in volatility it does also produce long-memory behaviour. It was the first model to decompose the variance factor into a weighted average of variance components computed at different time scales and thus able to highlight the variance asymmetry: the low frequency volatility impact the high frequency volatility more than the other way around.

Mug plot

In their paper Lynch and Zumbach (2003) study 50 million high-frequency quotes from 11 years of different FX-rates. To study the market heterogeneity they define two types of volatilities: the historical and the realised or future one. The historical volatility corresponds to the sum of squared return over a selected historical δ_σ time window. On the other hand, the realised volatility corresponds to a forward translation of the historical ones; it computes the sum of squared returns over the next δ_σ time points. To study the relation between the historical and future volatility they compute a mug-shot which represents the correlation between these two volatilities for different time windows.

These mug-shots highlight the heterogeneity of the underlying time series. However, for the different FX-rates they study they observe similar patterns: an asymmetry around the diagonal, a volatility cascade and local maxima. They explain the asymmetry by the fact that short-term realised volatility depends on all time horizons while the long-term realised volatility depends mainly on the long-term historical volatility. This observation is directly

connected to the volatility cascade where long-term historical volatility impacts short term realised volatility more than the other way around. More precisely, they observe on these mug-shots local maxima at specific frequencies. A maxima corresponds to a high correlation between the historical and realised volatility, hence showing a strong connection at a specific frequency. The frequencies at which appear these maximum highlight the dependence of the market on intra-day, day, week and month components. Those mug-shot observations highlight the limits of the widely used GARCH$(1,1)$ model to reproduce such heterogeneity and thus Lynch and Zumbach (2003) proposed different extensions to the GARCH model that include long-memory components but are all univariate and hard to estimate.

3.3.2 The variogram

Some insight from variograms

The motivation for this work started with a simple observation. In an article written for Risk, Anderson et al. (2000) used a volatility signature plot to determine the appropriate frequency to use for computing the realized variance (RV) over high frequency returns. This plot corresponded to the realized variance averaged at different time scales, which they then used to determine the time scale on which the microstructure effects stop interfering on the realized variance. On this graphic they also observed distinct patterns for liquid and illiquid stocks. In the first case the realized variance was increasing with the scale while it was decreasing for the illiquid stocks. A similar approach was used in Borland (2004) and Borland and Bouchaud (2005) where they used the variogram computed with a stock's returns to exhibit the properties of multifractality and correlation of volatilities across time scales. Their variogram corresponded to the averaged squared difference between squared returns at different scales. Both of these approaches used high frequency data, but what would we see on lower frequency scales, such as daily returns? Following this logic we computed the averaged realized variance for different frequencies of returns ranging from days to weeks, rescaled by frequency. The common stochastic diffusion process of log-prices assumes

a property of self-similarity. If that was correct, as explained in Borland (2004), the variance should be scale-invariant and thus exhibit a straight line on the variogram. In practice we observe three different clusters of stocks on this plot. Stocks with variances that increase, stay constant or decrease w.r.t. the time scale, i.e. going from daily to weekly returns. Can we explain this behaviour? As detailed in the following section, we can indeed formulate different economic interpretations which lead us to follow this logic further.

An economic phenomenon we want to capture

We expect to detect with a variogram what practitioners often describe as the gamma effect. Gamma corresponds here to the Greek letter Γ which represents for option hedgers the derivative of the delta. The delta Δ corresponds to the first derivative of the price of the option with respect to the price of the underlying asset and aims to measure the sensitivity of the option to the price of the asset. Hence the gamma represents the second order sensitivity to variations in the price of the asset.

Let us consider stocks; traders have to delta-hedge their positions, which signifies buying or selling a certain quantity of stocks in order for their Δ to be zero, and to do so they follow the results given by the equations of Black-Scholes. However, as they are not alone in the market a stock's price might not move in their direction. When Γ is positive, if the price increases then the Δ will increase too; while for a negative Γ the Δ will decrease. Thus, depending on the Γ traders will not react symmetrically to the move of the stock. Furthermore, when a trader wants to delta-hedge his position this second order effect might have an unexpected impact if the absolute value of Γ is high, since it means a high sensitivity on the variation of the stocks. This is a simple description, for more details on Delta and Gamma hedging refer to Hull (2017).

If we consider a stock on which a trader has a big option position its hedging might have a detectable impact on the time series of prices. Indeed, in the case where the trader tries to keep the price in a specific range, we might not

see any impact on the daily scale, but on a lower one, such as weekly returns, we might observe a variance lower than usual due to this bounding effect.

Description of the approach

The proof of concept for our intuition uses the following computation. For each stock we select the last $L = 180$ close-prices. Then for each specific time frequency we compute the realized variance, RV, of the exponentially weighted returns. Let l be the time lag, p the stock price, w the exponential weights and ρ the selected exponential decay speed, then for $l > 1$:

$$\begin{aligned} \rho_l &= \rho^{(L-1)/(L-l)}, \\ w_i &= (1-\rho_l)(\rho_l)^i, \\ RV_l &= \frac{1}{l}\sum_{i=l}^{L} w_i \left(\log\left(\frac{p_i}{p_{i-l}}\right)\right)^2. \end{aligned}$$

Where $\rho_l = \rho^{(L-1)/(L-l)}$ ensures that the sum of the exponential average weights is the same at each frequency l. For each stock we compute RV_l for $l \in [1, 5]$ to compare the daily variance with the weekly one, where l corresponds to the returns' frequency used for the computation. By re-weighting the variance by the time-scale we want to observe the spread with the commonly used theoretical variance rescaling. For example to obtain a weekly variance from a daily one it suffices to multiply it by 5. This comes from the assumption that prices are i.i.d, see Diebold et al. (1997) for a critique of this practice. Hence the expected behaviour would be for the function RV_l to be constant with respect to the time scale l.

Results for different stocks

In Figure 3.1 we show the result of the previously described computation for some selected stocks in the European market. This graphic exhibits three clusters: stocks with an increasing, constant, or decreasing variogram. Naturally the group of constant RV_l are not strictly constant and should be

better described as oscillating. The other two groups increase or decrease at different speeds. Indeed, while some seem to have an exponential variation, others are more linear. While the graphic shows only a few sampled stocks, we observed this behaviour across many different markets and stocks. Another interesting aspect of this phenomenon is that stocks with an increasing or decreasing variogram tend to keep this behaviour for some period of time.

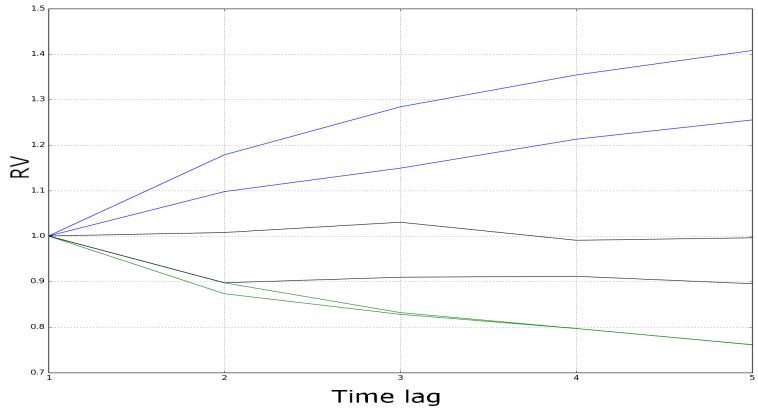


Figure 3.1: Variogram for 6 different European stocks. The X-axis represents the time lag used to compute the realised variance while the Y-axis corresponds to the obtained RV values re-scaled to obtained a lag one value of 1. The curves in black represent the expected behaviour. In blue are two selected stocks with increasing RV, while those in green are selected examples with decreasing RV.

Following the description above, a stock with a bounding effect on its time series would have a decreasing variogram. On the other hand, stocks which tend to diverge will have a weekly variance higher than the daily one due to these sparse explosions, which creates an increasing variogram. While this explanation makes sense, it is hard to prove if this is the dominating effect. The interesting point here is that by simply observing the time series of prices at different frequencies we could potentially detect such behaviour. That leads us to go further and try to decompose the moves into different and interpretable economic factors that will explain the stock's behaviour.

3.4 Markov Switching Multifractal

3.4.1 Description of the model

Let us recall that a process is defined as multifractal when the moments of the absolute returns scale as a power-law of the time frequency used to compute the returns. If the scaling exponent is linear the process is considered monofractal else multifractal when non-linear. The multifractal behaviour of financial time series can be observed by looking at the evolution of the distribution with respect to the time scale. For example Borland et al. (2005) shows that log-returns series tend toward Gaussian distribution at lower frequency while they have fatter tails at higher frequencies.

In Calvet and Fisher (2004, 2001) they propose a regime switching Markov model to estimate the multifractal property of the volatility process and forecast long-run volatility. Instead of a weighted sum of variances computed at different frequencies used by the HARCH models, the markov switching multifractal (MSM) approach of Calvet and Fisher (2004, 2001) uses a product of random variables representing different time scales. The MSM decomposes the volatility in hidden states, $M_t = (M_{1,t}, \ldots, M_{\bar{k},t})$, that are statistically independent of each others, first-order Markov and follow the same distribution M. The hidden states M follow a binomial distribution taking values m_0 or $2 - m_0$ with equal probability and $m_0 \in (1, 2)$. The states $M_{i,t}$ only differ in the time-scale at which they are updated. Hence, at each time t a state is updated with probability γ_k or remains the same with probability $1 - \gamma_k$. Thus, we define the vector of parameters by $\psi = (m_0, \sigma, b, \gamma_k)$. Let us define by r_t a log-return with a mean assumed to be zero and the noise $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, then the MSM model is:

$$\begin{aligned} r_t &= \sqrt{M_{1,t} M_{2,t} \ldots M_{\bar{k},t}}\epsilon_t \ , \\ \gamma_k &= 1 - (1 - \gamma_1)^{b^{k-1}} \ . \end{aligned}$$

Where $\bar{k}$ is the number of volatility frequencies. Hence, conditional on the state vector M_t the return r_t is Gaussian. For the transition probability to increase with the frequency we consider $\gamma_1 \in (0, 1)$ and $b \in (1, \infty)$. The only constraint on the states variables is $\forall k \ M_{k,t} \geq 0$ and $\mathbb{E}[M_{k,t}] = 1$. This regime

switching mechanism allows the MSM to model both long-term volatility regime as well as abrupt changes. And, when the number of lags $\bar{k}$ tends to infinity, the autocovariance of the returns tends toward the hyperbolic decline we observe empirically in returns of financial time series of prices.

In Calvet and Fisher (2004, 2001) they limit each state $M_{k,t}$ to a finite number of values b_m and denote by $m^1, \ldots, m^d$ the $d = b_m^{\bar{k}}$ values that the state vector M_t can take. To derive the estimation procedure for this model we first need to estimate the log-likelihood. Let us denote the transition matrix $A = \{a_{i,j}\}$ with $a_{i,j} = \mathbb{P}\left(M_{t+1} = m^j | M_t = m^i\right)$, the vector of conditional probabilities $\Pi_t = (\Pi_t^1, \ldots, \Pi_t^d)$ with $\Pi_t^i = \mathbb{P}(M_t = m_i | r_1, \ldots, r_t)$ and $\omega(r_t)$ by:

$$\omega(r_t) = \left[\frac{\mathcal{N}(r_t/\sigma g(m_1))}{\sigma g(m_1)}, \ldots, \frac{\mathcal{N}(r_t/\sigma g(m_d))}{\sigma g(m_d)}\right] .$$

where $\mathcal{N}(.)$ denotes the standard normal density. The recursive equation that defines the conditional state probability is:

$$\Pi_t = \frac{\omega(r_t) * (\Pi_{t-1} A)}{(\omega(t) * (\Pi_{t-1} A)).1^T} .$$

Where $*$ denotes the element by element product. Hence, the log-likelihood:

$$\ln(L(r_1, \ldots, r_T | \psi)) = \sum_{t=1}^{T} \ln\left(\omega(r_t) * (\Pi_{t-1} A)\right) .$$

In Calvet and Fisher (2004, 2001) they divide the problem of estimating the parameters in two parts: first, find the number of hidden states $\bar{k}$ and then, for a given one solve the maximum likelihood to obtain the vector of parameters ψ. They compute the maximum likelihood estimation (MLE) with numerical optimization but without detailing which algorithm they use and determine $\bar{k}$ with a comparison of the likelihoods for different values.

3.4.2 Algorithms to estimate the parameters of MSM

While Calvet and Fisher (2004, 2001) tested their algorithm on time series with thousands of data points, in our case we need the algorithm to produce accurate results with only hundreds of data points. Hence, we tried two algorithms to compute the MLE of the vector of parameters ψ on simulated datasets with 260 time points, the equivalence of 1 year of trading days for the US or EU markets. The first algorithm is a genetic algorithm with a population of 100, 10 generations, crossing and mutation probabilities of 0.5. The second is a Bayesian optimisation algorithm with 8 initial points, i.e. twice the number of parameters in ψ, a maximum number of iteration of 30 and using the Expected Improvement acquisition function with a parameter value to balance the exploration-exploitation trade-off of $\xi = 0.001$.

Following the variogram study we want to capture the weekly pattern of the volatility hence we restrict our study to the case $\bar{k} = 5$, however our observations are valid even for different numbers of volatility components. We consider uniform priors for all the parameters within the interval $\psi \in [\psi^{min}, \psi^{max}]$ with $\psi^{min} = (m_0^{min}, \sigma^{min}, b^{min}, \gamma_k^{min}) = (1.0, 0.1, 1.0, 0.1)$ and $\psi^{max} = (m_0^{max}, \sigma^{max}, b^{max}, \gamma_k^{max}) = (1.9, 1.5, 10, 1.0)$. On simulated datasets with $\bar{k} = 5$ and $T = 260$, both algorithms perform well with a parameter precision in the scale of 10^{-2} which we consider good enough for such a small number of points. However, for the same level of accuracy the genetic algorithm is much more expensive to run than the Bayesian optimisation. Thus we will use the Bayesian optimisation to run the MLE on real financial dataset.

3.4.3 MSM vs. variogram

We use a dataset of 378 stocks from the S&P500 with a price history available between 2000 and 2019. For each stock in this dataset we compute the MLE of the vector of parameters ψ with the last 180 quoted price with the Bayesian optimisation described in the previous section. We are looking for a connection between the different parameters of the MSM model and

volatility patterns. Since we want a model that will help understand the volatility behaviour the different parameters of that model should have economic interpretation; i.e. that we want to link the evolution of a parameter to a volatility pattern or economic effect. However, except for the volatility scaling factor σ, the four parameters of the MSM model $(m_0, \sigma, b, \gamma_k)$ lack economic interpretation. Hence, to find links between known volatility patterns and these parameters we look at the impact on the variogram of each of these parameters and vice-versa.

We simulate time series from the MSM model for different parameters value and compute the corresponding variogram for each of them. Unfortunately we do not observe any connections between the parameters value and the observe variogram, for a constant set of parameters from the MSM we can obtain all kinds of variograms: constant, increasing or decreasing. Then, we rank the time series using the variogram obtained with the last 180 trading days according to their increasing or decreasing rate. We choose the convention that a positive rate correspond to an increasing variogram and vice-versa. From this ranking, we look for patterns between the different parameters of the MSM model, $(m_0, \sigma, b, \gamma_k)$, and the rate of the variogram. Unfortunately, no clear pattern appears from this comparison. Hence, due to the lack of clear economic interpretation of the different parameters of this model and its inability to reproduce the results from the variogram study we try another model built from factors with economic interpretations.

3.5 Statistical feedback model

3.5.1 Multi time-scale statistical feedback volatility model

Statistical feedback evidence

In Borland (2002) they introduce a process to model log-returns with a noise following a non-linear Fokker Planck equation. This choice of distribution was motivated by the observation that time-series of log-returns exhibit peaks and fat-tails which the usual log-normal distribution struggles to model while

the Tsallis distribution can. By assuming that the noise process follows a non-linear Fokker-Planck equation of order $2 - q$, Borland (2002) show that the resulting process follows a Tsallis distribution. In her paper, Borland (2002) uses this model to develop new option pricing formulae but its application was limited due to the dependence of the model on the state at t_0 and its single time-scale.

In her following paper, Borland (2004) tried to solve those two shortcomings. The distribution of financial returns exhibits fat-tails with a cumulative distribution decaying as a power law and converging slowly toward a Gaussian as the time scale increases. In addition, the volatility process observes bursts and significant time correlation. In Borland (2004) they observe that Stochastic Volatility models capture the volatility clustering and fat-tails but their tails decay too quickly toward a Gaussian; on the other hand the multifractal approach is better but adds a source of noise. To solve these issues she extends her previous model to multiple time-scales with a discrete-time increment approach to free it from the t_0 dependency.

In Borland and Bouchaud (2005) they study the statistical property of the multi time-scale statistical feedback model of Borland (2004). Their study shows the model's ability to capture the fat-tails, long-memory, multi- fractality, volatility relaxation speed after a shock and time reversal asymmetry. Furthermore, they describe another reasoning, based on modelling traders' behaviour, that results in the same statistical feedback model. The statistical feedback corresponds to a volatility increase following abnormal past price moves. Hence, a volatility model such as GARCH can be seen as updating previous belief by how close to its expected value was the last move. Borland and Bouchaud (2005) justify the statistical feedback equation by modelling the impact on the price of traders stop-gain and stop-loss thresholds. Thresholds appear because traders usually enter a trade with a target gain and loss limit beyond which they will exit the trade. In addition, as theorised in the heterogeneous market hypothesis of Dacorogna et al. (1997), long-term traders under-react to high frequency moves while short-term traders take into account the low frequency behaviour.

Ideas behind the statistical feedback model

Let us define by y_i the log-price, $r_i = y_i - y_{i-1}$ the log-return, τ the time scale of the dataset, σ_i the volatility and ω_i a Gaussian noise; we consider the following discrete-time model of returns:

$$r_i = \mu\tau + \sigma_i\omega_i\sqrt{\tau}$$

From this process they build the statistical feedback model by extending a simple $ARCH(1)$ model. The volatility process is extended to add a coefficient representing the difference between the expected and measured volatility move. It compares the last squared return r_i^2 to its expected value $\sigma^2\tau$ as follows:

$$\sigma_i^2 = \sigma_0^2\left(1 + g\frac{r_i^2}{\sigma_0^2\tau}\right) \tag{3.1}$$

where g represents the strength of the feedback effect. Following the heterogeneous market hypothesis it is important to consider the different time horizons of the different traders; hence Equation 3.1 can be extended to include different time scales l. This leads to the symmetric general multi-scale feedback model:

$$\sigma_i^2 = \sigma_0^2\left(1 + \sum_{l=1}^{\infty} g_l\frac{(y_i - y_{i-l})^2}{\sigma_0^2 l\tau}\right)$$

Where the feedback strength is modelled by a variable $g_l = g/l^\alpha$ inversely proportional to the time scale to model the volatility cascade effect of long-term traders trading less frequently and with an impact on short term ones.

While being an additive model, the statistical feedback model empirically reproduces multi-fractal effects. Let us consider the moments as a function of the time scale define in Equation 3.2.1 and recall that the monofractal property corresponds to the linear exponent case, e.g. $\zeta_n = \zeta_2 n/2$. A multifractal behaviour corresponds to moments scaling with respect to a unique power-law. Since the moments can be computed as a sum of power laws with different exponents theoretically the statistical feedback model is not multifractal but Borland (2002, 2004); Borland and Bouchaud (2005) proved numerically that it does exhibit multifractal behaviour.

They test the time reversal asymmetry property of the model with the mug-shot methodology introduced by Lynch and Zumbach (2003) and described in Section 3.3.1. They used FX-rates to test the model and observed that it captures the time asymmetry; which is not surprising since the feedback model is build around the idea of time asymmetry with the past influencing the future behaviour. On this dataset of FX-rates Borland and Bouchaud (2005) find that the Student-Tsallis distribution with a time dependent degree of freedom as defined in the first paper of Borland (2002) fits the observed distribution well.

In Borland (2004); Borland and Bouchaud (2005) they calibrate the model by studying different stylised facts of volatility processes and instead of looking for exact values they select the range of acceptable values of the parameters that reproduces the stylised facts. Let us define the second moment of the volatility process by $z_2 = \sum_{l=1}^{\inf} g_l$, this process is considered stationary if $z_2 < 1$. Thus, studying the different volatility moments and distributions of returns will restrict the range of acceptable values for z_2. The observation of the temporal volatility correlation restricts the range of acceptable values for α, while the mean absolute deviation and kurtosis are sensitive to z_2, τ and α. Since the exponent ζ_n depends on z_2, reproducing the multifractal behaviour reduces the acceptable range for z_2. While this study provide useful information on the range of acceptable values for the parameters it does not provide a usable methodology to fit the model to each different financial time-series.

3.5.2 Presentation of the model and q-Gaussian distribution

The multi-time-scale statistical feedback model

In Borland (1998) she studies the properties of the solution to the non-linear Fokker-Planck equation and applied those results to create a non-Gaussian model of stock-returns which Borland (2002) uses to price options and then Borland (2004) extends into a multi-scale framework. Let us detail the multi time-scale statistical feedback model of Borland (2004) where the log price y

follows the diffusion process:

$$dy = \sigma d\Omega \ ,$$

with the noise following a statistical feedback process defined by:

$$d\Omega = P\left(\Omega|\Omega_0\right)^{\frac{1-q}{2}} d\omega \ ,$$

where ω is a standard Brownian noise, σ a volatility scale parameter and the conditional probability $P\left(\Omega|\Omega_0\right)$ satisfies the non-linear Fokker-Planck equation of order $2-q$ defined by:

$$\frac{\delta P}{\delta t} = \frac{1}{2}\frac{\delta^2}{\delta \Omega^2} P^{2-q} \ .$$

The solution of this equation is a q-Gaussian distribution, also known as Tsallis distribution, of entropic index q which can be expressed by:

$$P\left(\Omega_t|\Omega_0\right) = \left(a_q t^{\gamma} + \frac{b_q}{t}\left(\Omega_t - \Omega_0\right)^2\right)^{\frac{1}{1-q}} \ ,$$

Section 3.5.2 details the properties of this distribution. Let us, consider a discrete time approach with $\Delta t = t_{i+1} - t_i$ the time increment; the diffusion process becomes:

$$y_{i+1} - y_i = \bar{\sigma}_i \Delta\omega_i \ . \tag{3.2}$$

We define $\Omega_i = y_i/\sigma$ to obtain the following expression of the volatility process $\bar{\sigma}_i^2$:

$$\begin{aligned} \bar{\sigma}_i^2 &= \sigma^2 P\left(\Omega_i|\Omega_0\right)^{1-q} \ , \\ &= a_{i,0} + b_{i,0}\left(y_i - y_0\right)^2 \ , \end{aligned}$$

where $a_{i,0} = a_q \sigma^2 i^{\gamma}$ and $b_{i,0} = b_q i^{-1}$. This model corresponds to the single time scale statistical feedback model, to generalise the volatility into multi-scale we decompose the process into a sum of volatility processes computed

at different time scales:

$$\bar{\sigma}_i^2 = \frac{1}{N} \sum_{j=i-N}^{i-1} \bar{\sigma}_{i,j}^2 \ , \tag{3.3}$$

where N is the size the time window and $\bar{\sigma}_{i,j}$ corresponds to the volatility factor at time scale $i - j$ and defined by:

$$\begin{aligned} \bar{\sigma}_{i,j}^2 &= P_q \left(i, j\right)^{1-q} \ , \\ &= a_{i,j} + b_{i,j} \left(y_i - y_j\right)^2 \ , \end{aligned}$$

with $a_{i,j} = a_q \sigma^2 (i-j)^\gamma$ and $b_{i,j} = b_q (i-j)^{-1}$ the coefficients of the q-Gaussian distribution. We merge Equation 3.3 and Equation 3.2 to obtain the expression of the returns:

$$y_{i+1} - y_i = \sqrt{\frac{1}{N} \sum_{j=i-N}^{i-1} \bar{\sigma}_{i,j}^2} \Delta\omega_i \ .$$

This new formulation does not depend on t_0 and the volatility at time i is composed of volatility processes computed at different time scales each modelled as a statistical feedback process at that scale. Furthermore, the above framework can be extended to include an averaging kernel $w_{i,j}$ such as exponential average:

$$\Delta y_i = \sqrt{\frac{\sum_{j=i-N}^{i-1} w_{i,j} \left(a_{i,j} + b_{i,j} \left(y_i - y_j\right)^2\right)}{\sum_{j=i-N}^{i-1} w_{i,j}}} \Delta w_i \ ,$$

The q-Gaussian distribution

The non-linear Fokker-Planck equation of order $2 - q$ is defined by:

$$\frac{\delta P}{\delta t} = \frac{1}{2} \frac{\delta^2}{\delta \Omega^2} P^{2-q} \ ,$$

The solution of this equation is a q-Gaussian distribution, also known as Tsallis distribution of entropic index q which can be expressed by:

$$\begin{aligned} P\left(\Omega_t|\Omega_0\right) &= \left(a_q t^{\gamma} + \frac{b_q}{t}\left(\Omega_t - \Omega_0\right)^2\right)^{\frac{1}{1-q}} , \\ a_q &= \left(\left(2-q\right)\left(3-q\right)c_q\right)^{\gamma} , \\ b_q &= \frac{q-1}{\left(2-q\right)\left(3-q\right)} , \end{aligned}$$

with $\gamma = \frac{q-1}{3-q}$ and the coefficient c_q depends on the parameter q as follows:

$$\begin{aligned} \text{If } -\infty < q < 1 \text{ then: } c_q &= \frac{4\pi}{\left(3-q\right)^2\left(1-q\right)} \frac{\Gamma^2\left(\frac{1}{1-q}\right)}{\Gamma^2\left(\frac{1}{1-q} - \frac{1}{2}\right)} , \\ \text{If } q = 1 \text{ then: } c_q &= \pi , \\ \text{If } 1 < q < 3 \text{ then: } c_q &= \frac{\pi}{q-1} \frac{\Gamma^2\left(\frac{1}{1-q} - \frac{1}{2}\right)}{\Gamma^2\left(\frac{1}{1-q}\right)} . \end{aligned}$$

While this q-Gaussian distribution accepts any real value of q it is the solution of the non-linear Fokker Planck equation of order $2-q$ only for $q < 2$. This distribution has also the interesting property of maximising the Tsallis entropy. Its variance is given by:

$$E[\Omega_t^2] = \frac{1}{(5-3q)\beta_t} ,$$

with:

$$\beta_t = c_q^{\frac{1-q}{3-q}} \left(\left(2-q\right)\left(3-q\right)t\right)^{-\frac{2}{3-q}} .$$

Hence for $q \geq 5/3$ the variance diverges. Furthermore, for $q < 1$ the q-Gaussian distribution can exhibits cut-offs. In consequence of which Borland restricted her study to $q \in [1, 5/3]$. For financial time series we disagree on this limit since the distribution of some log-returns may observe cut-off or diverging variances. For example, cut-offs can appear in stocks with strong mean reverting patterns, and on the other hand studying jumps in volatility process is an active area of research. For different values of q this q-Gaussian

distribution will be very different, hence the interest for this distribution to model financial returns but also the importance of an accurate selection of the parameter q.

3.5.3 The existing methods to estimate the parameter q

For the continuous time version of the statistical feedback model, Borland (2004) computed the second moment as:

$$E[dy_t^2] = \frac{a_q \sigma^2 T^\gamma}{(\gamma+1)(1-b_q)} dt \ , \tag{3.4}$$

where T corresponds to the continuous time limit of N. With an exponential moving average $w_{i,j} = \exp(-\lambda(i-j))$ the equation becomes:

$$E[dy_t^2] = \frac{a_q \sigma^2 \Gamma(\gamma+1)}{(1-b_q)\lambda^\gamma} dt \ . \tag{3.5}$$

Thus, Equation 3.4 and Equation 3.5 highlight a direct relation between the variance of the time series and the variable q if the volatility level coefficient σ is known. In other words, for a discrete version of these equations and with log-returns r computed on time scale l we have the relation:

$$\frac{E[r_t(l)^2]}{l\sigma^2} = \frac{a_q \Gamma(\gamma+1)}{(1-b_q)\lambda^\gamma} \ . \tag{3.6}$$

Hence, if q or σ is known with Equation 3.6 we can obtain the value of the other parameter. Since we have two unknown variables, q and σ, we need another equation linking the time series process with q or σ. Borland (2004) looks at the volatility-volatility correlation and kurtosis behaviour but observes the kurtosis decaying rate for different time-scale l to be determined mostly by the exponential weighting average parameter λ, i.e. the memory decay. Similarly, the volatility-volatility correlation decaying speed is mainly influenced by the averaging weight w more than the parameter q. Following a different approach, Borland and Bouchaud (2005) find the appropriate q

by studying the rescaled mean absolute deviation (MAD) defined by:

$$MAD(l) = \frac{E[|y_i - y_{i-l}|]}{\sqrt{E[\sigma_i^2]l}} ,$$

Which they relate to q by:

$$MAD(l) = \sqrt{\frac{2\,(\mu-2)}{\pi\,(\mu-1)^2}} \frac{\Gamma\left(\frac{\mu+1}{2}\right)}{\Gamma\left(\frac{\mu}{2}\right)} .$$

with $\mu = (3-q)\,/\,(q-1)$. In addition, they define the excess kurtosis $\kappa(l)$ which is expected to decay as a power law of the time scale $l^{-\nu}$ and is linked to the parameter q by:

$$\kappa = \frac{6}{\mu - 4} ,$$

In Borland and Bouchaud (2005) they use plots of $MAD(l)$ and $\kappa(l)$ over a grid of values of q for a large number of time scales l in the thousands. In addition, since the choice of q also influences the multifractal spectrum, $M_n(l) = E[r_t(l)|^n] = A_n l^{\zeta_n}$, Borland and Bouchaud (2005) study the exponent ζ_n through the different moments to obtain information on the acceptable range for q.

In de Santa Helena et al. (2015) they propose to find q with a robust measurement of the kurtosis. They ignore the variable σ and use the tail distribution medcouple as a robust measure of the tails heaviness which they link to q. Medcouple is a robust statistic initially made to identify skewness, with sample data $X = \{x_1 < \cdots < x_n\}$ and $med(x)$ its median, they define the kernel:

$$h(x_i, x_j) = \frac{(x_j - med(x)) - (med(x) - x_i)}{x_j - x_i} ,$$

Then the medcouple metric is defined by:

$$\text{medcouple}(X) = \text{med}\left(h(x_i, x_j); \forall\{x_i, x_j\}/x_i \leq \tilde{x} \leq x_j\right) ,$$

They apply the medcouple to each side of the distribution $X^- = \{x_1 < \cdots < \tilde{x}\}$ and $X^+ = \{\tilde{x} < \cdots < x_n\}$ to measure the weight of the tails

and thus obtain left $LMC(X) = -\text{medcouple}(X^-)$ and right $RMC(X) = \text{medcouple}(X^+)$ side medouples. As the q-Gaussian distribution is symmetric they decided to study only the right side: RMC. Let us denote by $X(q)$ simulated data with parameter q which they sample with the modified box-muller method. They empirically find that the relation between the medcouple of the tails and q is well fitted by $RMC(X(q)) = \tanh(a + b\gamma + c\gamma^2)$. However, their study was done on one time scale with 2^8 different time series of length $n > 2^{14}$ which is unrealistic for our dataset of low frequency timeseries of prices.

3.5.4 Estimating the parameter q with σ unknown

We are looking for an estimator of q that can continuously update the q distribution without requiring thousands of data points. For example, by looking at the daily log-returns over the past year, i.e. with ~ 260 points. Since the multi-scale statistical feedback process can reproduce the behaviour of the log-returns at different time-scales we could try to find q by comparing simulated distributions with the observed ones at these different time scales. Let us denote the observed log-returns by r and the simulated one by r^s. Since the two parameters q and σ are linked, we start by studying different methods to fit q while considering a fix $\sigma = 1$, then in section 3.5.4 we solve the more difficult problem of fitting both q and σ simultaneously.

For a fixed σ the Equation 3.6 gives a bijective relation between the returns' volatility and the parameter q. Thus we can estimate q by studying the rescaled realised variance at different time scales. We use an exponential weighting average for the weights w and denote the observed and sampled RV by RV_r^l and RV_s^l respectively. We measure the distance between two RVs with an L_2 norm rescaled by the variance of the sampled dataset:

$$d_2\left(RV_r^l, RV_s^l\right) = \frac{\sum_{i=1}^{K}\left(RV_r^l(i) - RV_s^l(i)\right)^2}{\sum_{i=1}^{K}\left(RV_s^l(i)\right)^2},$$

Where K corresponds to the number of datapoints. Hence we define the

distance between the sampled set and observed one $d(r, r_s)$ by the sum of the d_2 distances for the RVs computed at different scales:

$$d(r, r_s) = \sum_{l=1}^{L} d_2 \left(RV_r^l, RV_s^l \right) \ , \tag{3.7}$$

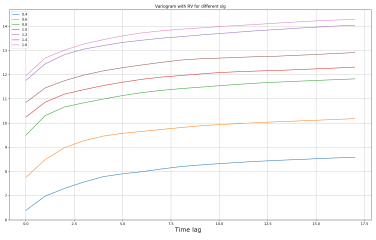
Where L represents the highest scale used, which is limited by the length of the time series under study. Unfortunately, due to the high variance in the scales of the log-return variances in financial data we need to take σ into account. Equation 3.6 shows that the combination of a high value for q and a low value for σ would produce a process with the same variance as a low q with a high σ values. Therefore, the distance defined in Equation 3.7 is not enough to estimate both parameters simultaneously and we need to find other equations. Looking at higher moments, the kurtosis decay between time-scales is linked to the parameter q. An other solution would be to use the medcouple as a robust tail measurement of the kurtosis, study the multifractal spectrum decaying exponent or look at the autocorrelation decay between different time scales. These methods are studied in Borland and Bouchaud (2005); de Santa Helena et al. (2015) to fit q on simulated time series with thousands of points while we want to use as few points as possible. On simulated datasets with only hundreds of points the results we obtain for the $MAD(l)$, multifractal decay and medcouple methods were too noisy to be usable.

Let us recall that in the statistical feedback model q represents the strength of the feedback effect while σ the scale. And, modifying these parameters impacts how the distribution of the returns evolve with the time scale, for example $q = 1$ corresponds to the Gaussian case where the distribution is self-similar. As presented in Section 3.3.2 the variogram is an interesting tool to study the evolution of the volatility for different time scales and Borland (2004); Borland and Bouchaud (2005) use the variogram to exhibit the correlation in volatilities across time scales. Let us recall that the variogram, $V(l, RV)$ represents the average squared spread between the RV at scale 1

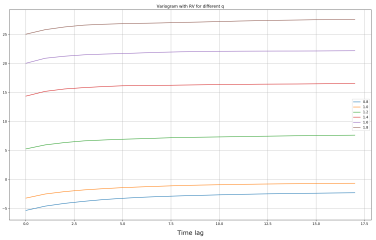
and $l > 1$:

$$V(l, RV) = mean\left([RV(1) - RV(l)]^2\right) \ .$$

To study the impact of q we study the variogram obtained for RV computed with different time windows. When using the RV computed over the last day, i.e. the previous absolute return we call this the rough variogram which reflects the rapid changes in time series behaviour. The evolution of the variogram with respect to the time scale l depends on both parameters σ and q where q impacts the slope while σ impacts the scale of the variogram. Since we are interested by q we want to minimise the impact of σ on the variogram thus we rescale each component of the variogram by the square root of the average squared daily-returns $\sqrt{mean(r_1^2)}$. Figure 3.2 shows on the impact of varying q or σ on the variogram on simulated data. Looking at the scale of the y-axis we see how minimised is the impact of σ compared to q. The same observation goes for the rough variogram with $RV(l) = |r_l|/l$ in figure 3.3.



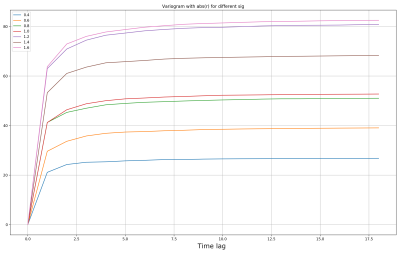
(a) Rescaled variogram evolution for varying σ values and fix $q = 1.3$



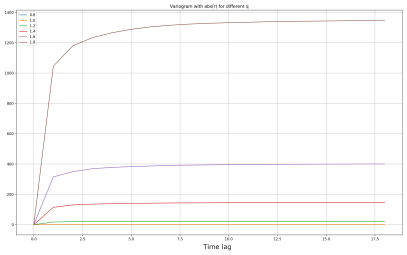
(b) Rescaled variogram evolution for varying q values and fix $\sigma = 1$

Figure 3.2: Impact of σ and q on the variogram with RV_r^l

In Borland (2004); Borland and Bouchaud (2005) they study the impact of q on the kurtosis decay for different time-scales and concluded that the kurtosis decaying rate was mainly influenced by the exponential moving average coefficient. In our simulations the choice of σ and q impacts the decaying rate in a similar fashion to the variogram. Hence, we use the log-kurtosis, $log(mean(|r_l|^4))$, to study the impact of q and σ on a time series. We might observe different results than Borland (2004); Borland and Bouchaud (2005) due to our choice of equation for the kurtosis. Because the impact of the



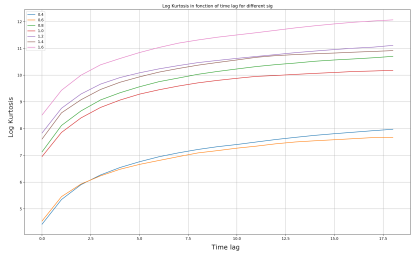
(a) Rescaled variogram evolution for varying σ values and fix $q = 1.3$



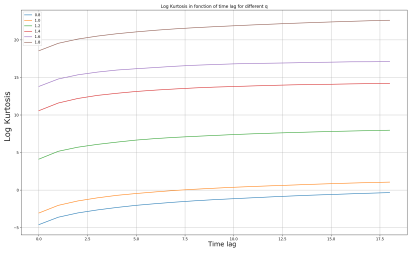
(b) Rescaled variogram evolution for varying q values and fix $\sigma = 1$

Figure 3.3: Impact of σ and q on the variogram with $|r_l|/l$

parameters q and σ on the kurtosis is more noisy with a rescaled-kurtosis $|(r_l - mean(r_l))/std(r_l - mean(r_l))|^4$ which they use in Borland (2004); Borland and Bouchaud (2005). Figure 3.4 exhibits the impact of σ and q on the evolution of the log-kurtosis as a function of the time scale. As with the variogram, the impact of σ is not completely cancelled but diminished compared to q.



(a) Rescaled log-kurtosis evolution for varying σ values and fix $q = 1.3$



(b) Rescaled log-kurtosis evolution for varying q values and fix $\sigma = 1$

Figure 3.4: Impact of σ and q on the log-kurotsis evolution with time lag

As explained with Equation 3.6, q and σ are connected to the second moment. Thus, we can compute a volatility map connecting the values of the two parameters to the variance of the time series in order to obtain a bounded interval of values compatible with the observed time series. Hence, we can compute a map of RV values obtained for a range of values of q and σ. Then, with the RV of the observed time series we use this maps to select

the acceptable range of values for q and σ. From this smaller range we run an extensive search over the acceptable values to measure which set fits the observed data best. However, due to the variance of the RV relying on only one map produces noisy results. Thus, we follow the same methodology but computing two values corresponding to the selected q and σ for the observed RV plus and minus one standard deviation of the RV. Then we compute the average of those two values to select q and σ. In our simulation, this approach gives results with smaller variance.

For the volatility map we use the RV computed over daily returns and we measure the distance between the simulated time series and the observed one with the L_2 norm of the difference for the rescaled variogram, the kurtosis decay and the distance $d(r, r_s)$ defined in Equation 3.7. These distances are combined into one by rescaling them between 0 and 1 and then summing them.

3.5.5 Results

On simulated data

We assess the performance of our proposed algorithm on simulated time series following a multi-scale statistical feedback model. Figure 3.5 represents the results of an extensive search distance measured between a simulated path with $(q, \sigma) = (1.1, 1.0)$ and the sampled one for each q value, where for a better visualisation the distance is caped at 8. Hence, we can clearly observe the window of possible values allowed by the filter. As expected, for the selected values the distance has a clear minimum at $q = 1.1$.

To assess the capacity of our algorithm to recover any value of q we simulated data from all possible combinations of q and σ within a range of values. The simulation requires the selection of additional parameters: the number of time-scales in the feedback model L, the size of the time window N, the exponential weighting parameter ρ and the number of Monte Carlo samples N_{spl}. We fix the values of these parameters to $(L, N, \rho, N_{spl}) = (20, 120, 0.99, 10)$ and use the grid $q \in [0.1, 1.9]$ and $\sigma \in [0.1, 5]$ with a step size of 0.1. We sim-

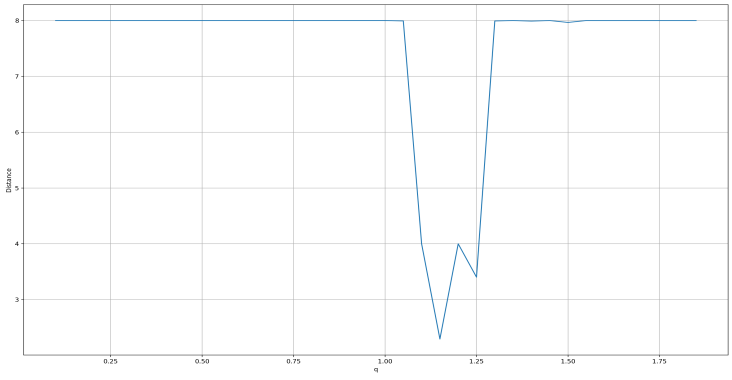


Figure 3.5: Extensive search result for a simulated path with $(q, \sigma) = (1.1, 1.0)$.

ulated time series for all the combination of q and σ parameters and estimate them with the algorithm proposed in Section 3.5.4. Figure 3.6 compares the estimated q with the true one used to simulate the data. Since the simulation used a range of values for σ the estimated q value corresponds to the average (in blue) and median (in orange) of the estimated q obtained over all the values of σ. Thus, if the results were perfect we should observe a straight line with coefficient one and passing through the origin. Interestingly, the estimations are more accurate for values of $q \geq 1$. We can perform the symmetric exercise to estimate σ averaging over the values of q. However, the estimation of σ over the range of different q is too far from the truth. While it has the right increasing trend it is offset by some margin making the estimated σ unusable. This simulation highlights the issue of the multi time-scale statistical feedback model, many combinations of σ and q result in distributions hard to distinguish over a small time window.

The limits of this approach

The results for the estimation of q in simulations were promising however, on real time-series data they are too noisy. We selected the 376 time series of log-prices for stocks from the S&P500 with quotes available between 2000 and 2019. Then we randomly picked one stock, Bristol-Myers Squibb Co

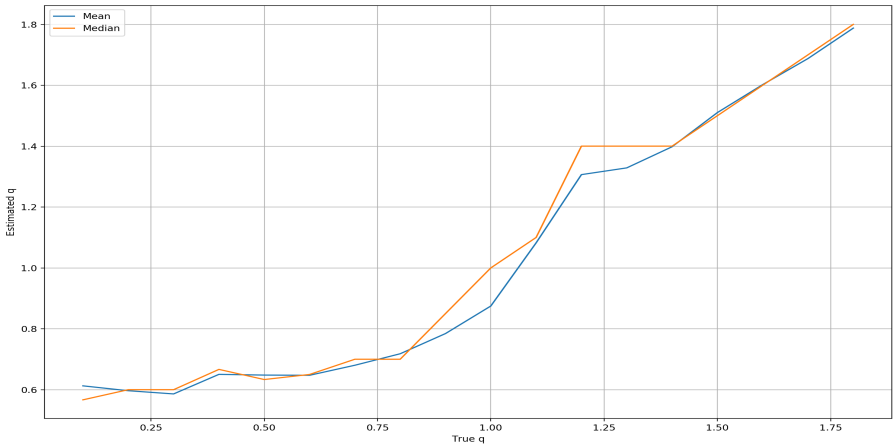


Figure 3.6: Extensive search result for a simulated data for $q \in [0.1, 1.9]$ and $\sigma \in [0.1, 5]$ with a step size of 0.1

(BMY), and estimated q between 09/2016 and 06/2019 with the parameters fixed at $(L, N, \rho, N_{spl}) = (20, 120, 0.99, 10)$. Figure 3.7 shows the evolution of the estimated q and the stock price during that time window on the same graph. While the algorithm only used the last 120 quotes to estimate the parameters, the evolution of q is nearly constant through time even when the underlying time series of prices show very different behaviours. The estimation of q is not precise enough to give accurate information on the behaviour of the underlying time series.

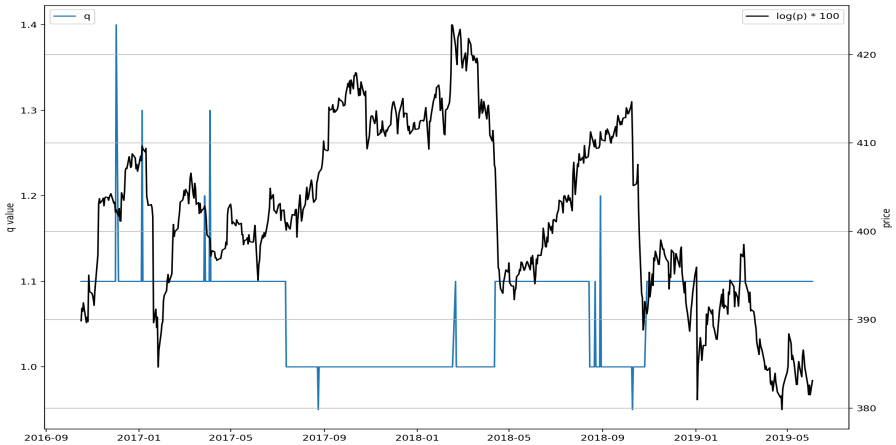


Figure 3.7: Evolution of the estimated parameter q for the stock BMY between $09/2016$ and $06/2019$ using parametrisation: $(L, N, \rho, N_{spl}) = (20, 120, 0.99, 10)$.

We then compute for one time t the estimation of the parameter q for all the stocks and look at the histogram of the estimated values. From this histogram we select stocks with q-values higher than 1.55 and those below 1 and compare their variograms. While we can observe a clear difference in the variogram between stocks with high q and those with low q it is not clear where the limit is since many stocks with the same q value close the limit have very different variograms. In other words, the estimation of q is not precise enough, we should have values with a 0.01 precision to clearly distinguish stocks as efficiently as the variogram but our algorithm only allows for a 0.1 precision below which the results become too noisy.

Thus, while the variable q describes well how the distribution of a time series evolve at different time scales its estimation at a precision small enough to cluster the stocks in different categories is complex. The algorithm we proposed in this section is able to accurately estimate the value of q on simulation for a range of σ values but at a precision to high to be usable on financial time series with only hundreds of data points.

3.6 The limit of the classic volatility models

3.6.1 The limitations of GARCH models

Müller et al. (1997) proposed the HARCH variance model to build on the idea of heterogeneous investment horizons. Following their approach, we start by working with variances before extending the model to other metrics. The variance of a time series of asset prices has many interesting properties that many models have previously tried to capture. Some stylized facts of variance-processes are of particular interest to us. The first is the time asymmetry observed in financial time series, i.e. the importance of distinguishing past from future. From this we can conclude that a model of variance should not be time reversible. This assumption then leads to a leverage effect, which is the response asymmetry between the magnitude of previous stock returns and future variance. Furthermore, we may consider multifractality, which defines how the distribution of returns should change when studied at differ-

ent time scales. These stylized facts are not reproduced by the widely used stochastic volatility or GARCH models. Further complicating matters, the distribution of each returns time series may have different characteristics and these can evolve through time. Papers such as Müller et al. (1997), Muzy et al. (2000), Lynch and Zumbach (2003) studied the multifractal behaviour and time asymmetry of financial returns. In Calvet and Fisher (2001, 2004) the authors extended the multifractal model of asset returns introduced by Mandelbrot to propose a stationary volatility model that can be estimated with maximum likelihood. Their Markov switching volatility model is based on a multiplicative cascade of volatility components, each representing a different frequency. While this model has a closed form likelihood it still requires an optimization step and the parameters lack clear interpretation. In contrast, we seek a model that helps us understand the time series, such that all parameters and variables have a clear interpretation.

3.6.2 The link between physics and volatility

As explained by the heterogeneous market hypothesis in Müller et al. (1997), different market participants trade with different objectives. One of them is the time scale and it impacts participants asymmetrically; the proposed HARCH model is a modified GARCH model using returns at different frequencies. They used this decomposition to show that lower time scales influence higher scales more than the other way around. In other words, short term traders are more greatly impacted by trades from long term traders than the other way around. This information asymmetry between scales has been further detailed in many papers by studying the volatility cascade, i.e. the flow of information between scales, see Zumbach and Lynch (2001) for a study of this cascade and its parallel with ideas from physics. Indeed, the name came from a similar concept in physics where the fluid vorticity cascade observed in turbulent states has a comparable behaviour to the distribution of financial returns at different scales, thus allowing the same mathematics to be used for volatility modelling. See also for example Muzy et al. (2000) for a multifractal random walk model that produces a scale-invariant stochastic volatility model. This form of statistical feedback model led Borland (1998,

2002, 2004); Borland et al. (2005); Borland and Bouchaud (2005) to propose a process with a noise component following the non-linear Fokker-Planck equation, the solution of which results in a q-Gaussian or Tsallis distribution. With this approach the model could theoretically give a distribution that models the behaviour of the price process for different frequencies and thus characterize each stock into different categories depending on the value of q. However, although this complicated distribution has interpretable parameters, it is hard to fit with few data points.

3.6.3 The HAR-RV model of Corsi

The models described previously are able to reproduce most stylized facts, especially multifractality, however they are hard to estimate and can lack clear economic interpretation. While many papers make use of multiplicative models, such as Calvet and Fisher (2004), for reproducing multifractality, LeBaron (2001) proved for the first time that a simple three-factor additive model could also display this behaviour. Based on these results, Corsi (2004); Corsi et al. (2008); Corsi and Reno (2009) build the HAR-RV model which uses the heterogeneous market idea to create a simple additive volatility model combining $AR(1)$ processes at different scales. This model is able to reproduce multifractal scaling while conserving simple economic interpretations and being easily extendible to incorporate additional stylized facts such as leverage and jumps Corsi and Reno (2009). They used this decomposition into different time-scales to learn which economic factor is influencing the time series. But this was done for each stock individually and lacks multivariate connections.

3.6.4 How can we model cross-series relationships?

In order to obtain a model of the market as a whole it is important to take into account the cross-series relationships between volatilities. Many studies have shown the clear performance increase of integrating the multivariate effect in their model, such as multivariate- GARCH and -HEAVY models, see e.g. Noureldin et al. (2011), however such models use heavy MCMC

parametrization techniques which make them expensive to scale. Other papers proposed methods to minimize this cost; for example Nakajima (2014) proposed a multivariate SV model using a covariance matrix Cholesky decomposition to allow for parallel evaluation of independent univariate SV models but still fit the model with an expensive MCMC procedure. Another limitation of most multivariate models is their dependence on a historical window to perform the parametrization compared to employing a sequential learning approach. One reason for this is the complexity of online learning algorithms for non-linear and non-Gaussian models, see e.g. Johannes et al. (2005), which presents an auxiliary particle filter to sequentially update a stochastic volatility model with jumps. In contrast, the model we proposed in the Chapter 4 is multivariate and due to a Gaussian setup it can sequentially adapt its parameters sequentially to the evolving market.

3.6.5 The SGDLM model

The novel model we present in Chapter 4 aims at explaining what drives the time series. This approach will give both new structural information about the market, as well as better longer-term forecasts. We build each module of the model to capture the different observations described above, and then combine each time series into a multivariate model of the market under study. This is possible using the SGDLM framework developed by Gruber and West (2016a,b); Zhao et al. (2016); McAlinn and West (2016), which considers an environment of independent time series modelled by a simple Normal DLM, using a sparse covariance matrix representation to incorporate cross-series relationships. Also, the SGDLM does not follow the classic multivariate variance approach of assuming symmetric connections between assets by using a covariance matrix to model the cross-series relationship. Instead, the SGDLM has a multivariate Wishart model running in parallel to sequentially select for each time series which other assets have influence on it. After this selection phase, the candidates are tested as a regression variable. This techniques creates non-symmetric cross-series relationships which produce interesting information on the underlying market risks.

In addition, compared to the previously described multivariate approaches, the combination of de-coupling and re-coupling steps allows for parallel updates of each DLM with Kalman filter equations. This highly parallel architecture to sequentially update the series in a large scale environment is perfectly suited for a GPU which allows us to study bigger environments. Since the DLMs are independent they do not need to follow the same model; their only requirement is for the distributions of the states and observations to be Normally distributed. Since this sequential approach continuously updates the coefficients of the DLMs, their distribution will adapt to changes in the environment.

4

The Heterogenous-SGDLM model

4.1 Introduction

In this chapter we propose a model that extends the Simultaneous Graphical Dynamic Linear Model (SGDLM) of Gruber and West (2016a,b); Zhao et al. (2016); McAlinn and West (2016) to incorporate the heterogeneous autoregressive realised volatility (HAR-RV) model of Corsi (2004); Corsi et al. (2008); Corsi and Reno (2009). Combining the HAR-RV model with the SGDLM framework creates an easy-to-scale multivariate volatility estimator. Each time series of daily log-volatility is a DLM with idiosyncratic factors from the HAR-RV model, cross-series relationship factors from the multivariate Wishart and any additional variables specific to each time series. The variables can be clustered into two groups: endogenous and exogenous. The endogenous group represents the information specific to the stock, while the exogenous one represents the influence of the environment. As a result of the flexibility of the SGDLM framework, as long as the normality condition of each DLM is respected they can easily be extended to include additional variables, and these do not have to be the same for every series. Hence, we will present different extensions that move further away from the standard

HAR-RV formulation .

Section 4.2 details the different techniques to approximate a probability distributions that we will use in the SGDLM algorithm. The SGDLM model is described in section 4.3 and the HAR-RV model in section 4.4. Section 4.5 explains the combined model and a variety of implementation issues. Section 7.4 shows the results on stocks from European markets and Section 4.7 on other environments and assets.

4.2 Techniques to approximate a distribution

4.2.1 Importance Sampling

Importance sampling (IS) is an estimation techniques used to reduce the variance of a sampling procedure or when it is hard to sample from the target distribution. The idea of IS is based on the observation that some values of the simulated variables have more impact on the parameters being estimated than others. Thus, sampling these values more frequently to emphasize them would reduce the variance of the estimator. This selection phase is done by choosing a distribution which will emphasize these values. However, using such distribution would add a bias to the results. Hence, the results are weighted by the likelihood ratio of the desired distribution over the sampled one, i.e. the Radon-Nikodyn derivative.

For example let us consider the case where we want to estimate a multivariate Gaussian distribution by sampling from a product of univariate Gaussian distributions. The IS weights consist of the ratio of the multivariate Gaussian over the product of univariate Gaussians. Thus, the weights are obtained by decomposing the multivariate distribution into a product of univariate Gaussians. Thus, the IS scheme is useful when we want to compute samples of a complex multivariate distribution from simpler univariate distributions. However, to do the opposite i.e. estimating the parameters of univariate distributions that best fit samples from a complex multivariate distribution we shall use the Variational Bayes technique.

4.2.2 Variational Bayes

Variational Bayes is similar to averaging. This technique corresponds to approximating an intractable conditional distribution $P(X|D)$ of unobserved variables X given some data D by a tractable distribution $G(D)$. We need a measure of dissimilarity between two distributions in order to approximate an intractable distribution by a distribution which we can use. One of the most common distance measure between two probability distributions is the Kullback-Leibler (KL) divergence. Thus the Variational Bayes methodologies decomposes into two steps: the first step is to select the distribution $G(D, \theta)$, then find the parameters θ that minimise the KL distance to the target distribution.

For example, we can consider the inverse problem than the example used to illustrate the IS approach: we have samples from a multivariate Gaussian distribution and we want to approximate this distribution by a product of univariate Gaussians. To implement the Variational Bayes technique we need the KL distance between a multivariate Gaussian and the product of independent univariate Gaussian. From this KL distance, we can derive the equation according to each parameter of the univariate Gaussian distribution to obtain the values that minimises that distance. We will detail these equations after the definition of the KL distance.

4.2.3 Kullback-Leibler divergence

The Kullback-Leibler divergence measures the discrepancy between two distribution with the particularity to be asymmetric. In the discrete space, the KL distance between the distributions G and P is defined by:

$$D_{KL}\left(G||P\right) = \sum_{X} G(X) \log\left(\frac{G(X)}{P(X)}\right) \ .$$

With X a set of samples from G. The KL distance corresponds to the expectation with respect to P of the logarithmic difference between the two distributions. As an example on how KL divergence and VB work together

we will estimate a multivariate Normal-Gamma distribution by a product of univariate Normal-Gamma distribution since we will use the results of this computation for the SGDLM model. The first step is to compute the KL distance, next to find the equation for each parameter that minimises that distance. The inputs correspond to the samples from the multivariate distribution, the outputs are the parameters of each univariate Normal-Gamma distribution. The KL distance corresponds to:

$$KL\left[\mathcal{NG}(\Theta,\Lambda_t)||\prod_i \mathcal{NG}(\theta_i,\lambda_i)\right] = \mathbb{E}\left[\log\left(\frac{\mathcal{NG}(\Theta,\Lambda)}{\prod_i \mathcal{NG}(\theta_i,\lambda_i)}\right)\right] .$$

In order to obtain the equation for each parameters of the distribution i we will compute the derivative of the KL distance with respect to that parameter. Hence the problem simplifies into finding the parameters of the Normal-Gamma distribution:

$$\mathcal{NG}\left(\theta_i,\lambda_i;m_i,C_i,n_i,s_i\right) = \mathcal{N}\left(\theta_i|\lambda_i;m_i,\frac{C_i}{s_i,\lambda_i}\right)\mathcal{G}\left(\lambda_i;\frac{n_i}{2},\frac{s_i n_i}{2}\right) .$$

With the Gamma distribution defined for $\lambda_i > 0$ as:

$$\mathcal{G}\left(\lambda_i;\frac{n_i}{2},\frac{s_i n_i}{2}\right) = \Gamma\left(\frac{n_i}{2}\right)^{-1}\lambda_i^{n_j/2-1}\left(\frac{s_i n_i}{2}\right)^{n_i/2}\exp\left(-\frac{s_i n_i}{2}\lambda_i\right) ,$$

where $\Gamma()$ corresponds to the function Gamma. Hence, the KL distance becomes:

$$\begin{aligned} KL = c \quad &- \quad \mathbb{E}\left[\log\mathcal{N}\left(\theta_i|\lambda_i;m_i,\frac{C_i}{s_i,\lambda_i}\right)\right] && (4.1)\\ &- \quad \mathbb{E}\left[\log\mathcal{G}\left(\lambda_i;\frac{n_i}{2},\frac{s_i n_i}{2}\right)\right] , && (4.2)\end{aligned}$$

with c a constant. Then developing the expectations leads to:

$$\begin{aligned} KL \quad &= \quad c+\frac{1}{2}\log\left|\frac{C_i}{S_i}\right| + \mathbb{E}\left[\lambda_i\left(\theta_i-m_i\right)^T\left(\frac{C_i}{s_i}\right)^{-1}\left(\theta_i-m_i\right)\right]\\ &+ \quad \log\Gamma\left(\frac{n_i}{2}\right) - \frac{n_i}{2}\log\left(\frac{s_i n_i}{2}\right) + \frac{1-n_i}{2}\mathbb{E}\left[\log\lambda_i\right] + \frac{s_i n_i}{2}\mathbb{E}\left[\lambda_i\right] ,\end{aligned}$$

where $|.|$ represents the determinant. We obtain the equation for each parameter by computing the derivative of the KL divergence with respect to that variable and putting it to zero. For the mean m_i of the parameters θ_i:

$$\begin{aligned} \partial_{m_i} KL &= \mathbb{E}\left[\lambda_i \left(\theta_i - m_i\right)\right] , & (4.3) \\ &= 0 \iff m_i = \frac{\mathbb{E}\left[\lambda_i \theta_i\right]}{\mathbb{E}\left[\lambda_i\right]} . & (4.4) \end{aligned}$$

Then for the variance $V(\theta) = |C_i/s_i|$ of the parameters θ_i:

$$\begin{aligned} \partial_{V(\theta)} KL &= V(\theta)^{-1} - V(\theta)^{-2} \mathbb{E}\left[\lambda_i \left(\theta_i - m_i\right)^T \left(\theta_i - m_i\right)\right] , & (4.5) \\ &= 0 \iff V(\theta) = \mathbb{E}\left[\lambda_i \left(\theta_i - m_i\right)^T \left(\theta_i - m_i\right)\right] . & (4.6) \end{aligned}$$

For the intensity s_i of the gamma distribution:

$$\begin{aligned} \partial_{s_i} KL &= \frac{n_i}{s_i} - n_i \mathbb{E}\left[\lambda_i\right] , & (4.7) \\ &= 0 \iff s_i = \frac{n_i}{n_i \mathbb{E}\left[\lambda_i\right]} . & (4.8) \end{aligned}$$

For the scale n_i of the gamma distribution:

$$\partial_{n_i} KL = 1 + \log\left(\frac{s_i n_i}{2}\right) - 2\gamma\left(\frac{n_i}{2}\right) + \mathbb{E}\left[\log \lambda_i\right] - s_i \mathbb{E}\left[\lambda_i\right] , \quad (4.9)$$

where $\gamma\left(\frac{n_i}{2}\right) = \Gamma'\left(\frac{n_i}{2}\right) / \Gamma\left(\frac{n_i}{2}\right)$ with Γ' the derivative of the function Gamma. Then, let us replace s_i with it's expression to obtain:

$$\begin{aligned} \partial_{n_i} KL &= 0 \iff & (4.10) \\ 0 &= \log(n_i) - 2\gamma\left(\frac{n_i}{2}\right) + \mathbb{E}\left[\log \lambda_i\right] - \log\left(2\mathbb{E}\left[\lambda_i\right]\right) . & (4.11) \end{aligned}$$

Contrary to the other variables the computation of n_i requires the use of an optimisation algorithm. Hence, following this system of equations, from a sample set of variables from the multivariate distribution we obtain the parameters for each univariate distribution that minimise the KL divergence between those two distributions.

4.3 SGDLM model

4.3.1 Description of the model

The SGDLM model developed by Gruber and West (2016a,b); Zhao et al. (2016); McAlinn and West (2016) brings together many different statistical techniques. Before it, the classic approach to working with DLMs in a multivariate setting was the inverse Wishart model; see Prado and West (2010) for a detailed description of this model and most tools used in the SGDLM. The motivation for this set-up was to study each time series of a multivariate model independently in parallel while still modelling the combined multivariate distribution. They used the variational Bayes approximation to approximate the multivariate model into independent individual distributions and then recouple them with importance sampling. This allows each time series' DLM to be different from the others. Let us denote by $y_{j,t}$ the observations, $\theta_{j,t} = (\phi_{j,t}, \gamma_{j,t})$ the state vector, $G_{j,t}$ the state evolution matrix, $F_{j,t}$ the vector of variables, $\nu_{j,t}$ the observations' noise independent of $\omega_{j,t}$ the states' noise. Therefore, $\lambda_{j,t}$ corresponds to the observations' precision and $W_{j,t}$ to the states' covariance matrix. Hence the DLM of series j at time t reads :

$$\begin{aligned} y_{j,t} &= F_{jt}\theta_{j,t} + \nu_{j,t} = x_{j,t}\phi_{j,t} + y_{sp_t(j),t}\gamma_{j,t} + \nu_{j,t}\,, \\ \theta_{j,t} &= G_{j,t}\theta_{j,t-1} + \omega_{j,t}\,, \\ \text{with:} \quad & \nu_{j,t} \sim \mathcal{N}(0, \lambda_{j,t}^{-1})\,, \\ & \omega_{j,t} \sim \mathcal{N}(0, W_{j,t})\,. \end{aligned}$$

Where $x_{j,t}$ and $\phi_{j,t}$ represent variables and coefficients of the internal predictors, while $y_{sp_t(j),t}$ and $\gamma_{j,t}$ correspond to the variables and coefficients of the cross-series time varying conditional dependencies. In other words $(x_{j,t}, \phi_{j,t})$ represents the endogenous information and $(y_{sp_t(j),t}, \gamma_{j,t})$ the exogenous one. $sp_t(j)$ denotes the group of selected simultaneous parents which is adaptively revised over time to capture the changes in the market. Let us define $\mu_{j,t} = x_{j,t}\phi_{j,t}$ as the mean, Λ_t as the precision-matrix with only $\lambda_{j,t}$ on the diagonal, and the matrix with the coefficients over the related parents by Γ_t,

then we can write the multivariate model in matrix form:

$$\begin{aligned} y_t &\sim \mathcal{N}\left(A_t \mu_t, \Sigma_t\right) , \\ \text{With:} \quad & A_t = \left(I - \Gamma_t\right)^{-1} , \\ & \Omega_t = \Sigma^{-1} = \left(I - \Gamma_t\right)^T \Lambda_t \left(I - \Gamma_t\right) . \end{aligned}$$

Once this decomposition between endogenous and exogenous variables has been defined, the parents need to be chosen. In the first version of the model a fixed matrix defines the relationships between the different time series to select the exogenous variables. This proves particularly limiting as the performance of the whole model heavily depends on the choice of this matrix. In the latest evolution the authors added the two letters SG for Simultaneous Graphical. This means that a multivariate Wishart DLM runs in parallel to the main algorithm to build a relational graph and select which exogenous variables to include in the DLM. The selection of this group of parents will be detailed in the following section.

It is interesting to emphasize how this set-up differs from the classic multivariate modelling approaches. Usually the multivariate distribution of the market will represent the cross-series relationships by non-zero elements in the covariance matrix, denoted Σ here. This implicitly assumes the relationships to be symmetric. While in the SGDLM model the cross-series coefficients are present in each individual DLM as variables of the regression, $(y_{sp_t(j),t}, \gamma_{j,t})$, hence in the mean part of the individual DLM. Furthermore, in this model the step selecting the parents variable is separated from the one computing its value. It first fits a multivariate inverse-Wishart distribution to the environment under study and use it to create a short-list of candidates. Members of this list are then included in the DLM formulation and the model will compute with its sequential update their coefficients. Therefore, it creates an asymmetrical relationship between the stocks in the environment; for each individual DLM the parents are selected because they impact the behaviour of the stock.

By restricting each DLM to follow a log-normal-gamma distribution we can

sequentially update the parameters of the DLM. This uses a Kalman filter with a Gamma distribution for the noise. Using this sequential update is possible thanks to the decomposition of the multivariate distribution into simple DLMs for which an extensive theory exists. Having independent equations for each time series will later allow us to include as endogenous factors an entire economic model. Another quality of this model is its capability to learn sequentially. Thanks to the Bayesian formulation and Kalman filter the model will continuously update the parameters of the distributions without depending on a specific look-back window. Indeed in the Kalman filter the past information is only present through the current value of the mean and variance of the distribution. That will allow us to follow the evolution of the coefficients through time and for the model to quickly adapt in case of rapid changes in the market.

4.3.2 Parents update

The group of parents for stock j, $sp_t(j)$, includes three subsets: the core $sp_{core,t}(j)$, up $sp_{up,t}(j)$ and down $sp_{down,t}(j)$ sets. The core parental set $sp_{core,t}(j)$ corresponds to the selected group of series used as predictors for j. In the up set are core-group candidates for entry into the parental set. In the down set are all the series previously in the core or up group.

A multivariate Wishart DLM runs in parallel to the main algorithm to update the parental sets. With this model we obtain a dense covariance precision matrix which is used to select the candidates to the up set. For each series j the n_{max} series with the highest precision element become candidates for inclusion in the up set. As long as the size limit of the up set is not reached the n_{max} series not currently in the up or core group are added to the up set. The candidates stay in this set during ΔT time before being considered for promotion to the core set. During that time they will be part of the parental set $y_{sp_t(i),t}$ used in the SGDLM. When the core group is full all the members of the up- and core-sets are ranked according to their signal to noise ratio, a_t/R_t, and the smallest values retired to the down set. Down set members will have their coefficients gradually put to zero over ΔT time periods.

Issues arise in the choice for n_{max}, ΔT and the number of parents allowed. For example, if n_{max} equals the number of parents allowed in the core and up sets, then the parents set can only be updated every ΔT times since it is full in-between. If the up-set does not have any size limit and the number of series considered for the up-set, n_{max}, is high then the number of parents could reach the total number of series m. These parameters will also influence the frequency at which the core-set can change all its members. A large n_{max} will result in big and noisy up-sets and the bigger the up-set the higher the potential turnover rate of the core-set. A smaller n_{max} will limit the maximum number of changes in the core-set, thus reducing the noise in those sets. If the number of allowed parents in the up-set is small, ΔT determines the turnover-frequency of the core-set. In addition, a short ΔT will put more importance in the choice of prior for the new members of the up-set, while, with a longer ΔT the prior choice becomes irrelevant in the signal to noise ratio used in the acceptance decision.

These choices of parameters will influence the size of the matrices used in the following steps, since they increase as a square of the number of parents. Hence, while the choice of parameters does not influence the computation of the parents-update it does have a significant impact on the memory requirement of the other steps. The standard multivariate Wishart DLM and parent updates do not have any computationally intensive steps.

4.3.3 Posterior update

We update the posterior with the data measured at $\mathcal{D}_t$, and parameters computed at $t-1$. Since the time series are independent we can update them separately. Each independent DLM posterior distribution follows a Normal Gamma distribution $p(\theta_{j,t}, \lambda_{j,t}|\mathcal{D}_t) \sim \mathcal{NG}(m_{j,t}, C_{j,t}, n_{j,t}, s_{j,t})$:

$$\begin{aligned} \theta_{j,t}|\lambda_{j,t}, \mathcal{D}_t &\sim \mathcal{N}\left(m_{j,t}, C_{j,t}/\left(s_{j,t-1}\lambda_{j,t}\right)\right) , \\ \lambda_{j,t}|\mathcal{D}_t &\sim \mathcal{G}\left(n_{j,t}/2, n_{j,t}s_{j,t-1}/2\right) . \end{aligned}$$

Where $m_{j,t}$ corresponds to the mean, $n_{j,t}$ the shape of the Gamma distribution and $n_{j,t}s_{j,t-1}$ its scale. The notations were motivated by the observation that following those equations the distribution of $\theta_{j,t}$ is multivariate student-t distribution with $n_{j,t}$ degrees of freedom and scale matrix $C_{j,t}$. As a result of this simple formulation the posterior update follows the Kalman filter's equations. The multivariate posterior $P(\Theta_t, \Lambda_t|\mathcal{D}_t)$ is obtained by re-coupling those independent posterior distributions with importance sampling.

The multivariate normal distribution sampling for $(\theta_{j,t}|\lambda_{j,t}, \mathcal{D}_t)$ will require a Cholesky decomposition of the covariance matrix. But since in practice the covariance matrix $C_{j,t}$ positive definiteness is not guaranteed we can run into numerical issues when sampling this variable. We therefore added in our implementation a shrinkage method to guarantee the positive definiteness of this covariance matrix estimation.

This step quickly becomes computationally expensive with a large number of series, samples and allowed parents. Let N_{MC} represent the number of Monte-Carlo samples, m the number of series and K the number of parents. Then the posterior sampling requires $m \times N_{MC}$ K-dimension multivariate normal-gamma computations.

4.3.4 Recoupling with importance sampling

The independent DLMs are recoupled to form the multivariate posterior distribution with importance sampling:

$$p\left(\Theta_t, \Lambda_t|\mathcal{D}_t\right) \propto |I - \Gamma_t| \prod_{i=1}^{m} p(\theta_{j,t}, \lambda_{j,t}|\mathcal{D}_t)\,.$$

Where I is the identity matrix and $|-|$ corresponds to the determinant. Hence the importance sampling weights are defined by $\alpha \propto |I - \Gamma_t|$. The weights are computed with Monte Carlo samples from the independent Normal-Gamma posterior distribution. Since the number of parents will stay low compare to m the coefficient matrix Γ_t will be sparse. Following previous sec-

tion reasoning, the exact posterior requires the computation of N_{MC} $m \times m$-sparse-matrices determinants to obtain the importance sampling weights.

In Gruber and West (2016b) they used the importance sampling entropy as a proxy of the Kullback-Leibler distance and thus of the quality of the model-fit. They define the entropy H by:

$$H_N = \sum_{n=1}^{N} \alpha_n \log(N\alpha_n) \,.$$

We will study this variable as a proxy of the environment stress. The higher the entropy the lower the approximation quality, which may be interpreted as a sign of increased environment-variance as a result of divergence between the model and the observations.

4.3.5 Decoupling with variational Bayes

The one-step-ahead inference corresponds to the prior update $p(\Theta_{t+1}, \Lambda_{t+1} | \mathcal{I}_{t+1} \mathcal{D}_t)$. The authors used Variational Bayes to decouple the multivariate distribution into independent series and thus process the time series in parallel. The Variational Bayes decomposition approximates the multivariate posterior $p\left(\Theta_t, \Lambda_t | \mathcal{D}_t\right)$ by a product of Normal-Gammas $p(\theta_{j,t}, \lambda_{j,t} | \mathcal{D}_t) \sim \mathcal{NG}(m_{j,t}, C_{j,t}, n_{j,t}, s_{j,t})$ by minimizing the Kullback-Leibler (KL) distance in the usual manner.

The resulting KL distance at each time t gives information on the quality of the fit. Significant changes in the environment under study would decrease the approximation quality and thus increase the KL distance. Gruber and West (2016a) applied this model on the S&P500 time series and observed similarities between a rescaled KL measure and the St. Louis Federal Reserve Bank Financial Stress Index. That observation led them to study further the strength of this relation, see Section 4.6.6 for a more detailed discussion.

Once the multivariate distribution is decomposed into independent DLMs

the parameters are updated independently in parallel to $t+1$. The updated parental set updates the filter's parameters $(a_{j,t+1}, R_{j,t+1}, r_{j,t+1}, W_{j,t+1})$. The innovations follow a Gamma-Beta Stochastic volatility update $\lambda_{j,t+1} = \lambda_{j,t} b_{j,t}/\beta_\lambda$ where $b_{j,t} \sim \text{Beta}\left(\beta_\lambda n_{j,t}/2, (1-\beta_\lambda) n_{j,t}/2\right)$. Where the parameter β_λ influences the smoothness of the predicted λ. The Gamma-Beta Stochastic volatilities appears through the gamma parameter update $r_{j,t+1} = \beta_\lambda n_{j,t}$. According to the DLM equations, the states follow the update $\theta_{j,t+1} = G_{j,t+1}\theta_{j,t} + \omega_{j,t+1}$ with $\omega_{j,t+1} \sim \mathcal{N}(0, W_{j,t+1}/(s_{j,t}\lambda_{j,t+1}))$.

4.3.6 Inference

Each DLM has an updated prior distribution $p(\theta_{j,t+1}, \lambda_{j,t+1}|\mathcal{D}_t) \sim \mathcal{NG}(a_{j,t+1}, R_{j,t+1}, r_{j,t+1}, s_{j,t})$. Samples from these independent priors are combined to obtain the multivariate distribution $y_{t+1} \sim \mathcal{N}(A_{t+1}\mu_{t+1}, \Sigma_{t+1})$, with first moments, $A_{t+1}\mu_{t+1}$, and second moments, Σ_{t+1}. This step is computationally expensive and requires the computation of $N_{MC} \times m$ multivariate normal gamma distribution to obtain $p(\theta_{j,t+1}, \lambda_{j,t+1}|\mathcal{D}_t)$, N_{MC} inverses and matrix-matrix products of (m, m) matrices to compute the first two moments $A_{t+1}\mu_{t+1} = (I - \Gamma_{t+1})^{-1}\mu_{t+1}$ and $\Sigma_{t+1} = \left((I-\Gamma_{t+1})^T \Lambda_{t+1} (I-\Gamma_{t+1})\right)^{-1}$.

Inference more than one day ahead can be computed recursively with the previous one-step-ahead prediction. Before the next step prediction, the filter's parameters $a_{j,t+1}$ and $R_{j,t+1}$ are updated with the previously inferred value. As before we produce Monte-Carlo samples of the first two moments to infer time $t+k$ with the computed parameters at t and inferred values at $t+k-1$. The difficulty comes from the internal predictors variables $x_{j,t}$ which requires to have the values for the next step update. Hence it is only possible to recursively produce inference more than one step ahead if the predictors' variables can be updated with the previously inferred values.

4.3.7 Parameters sensitivity

Due to its Bayesian structure the model has a few hyper-parameters. For the parent step the size of the different groups is fixed and must be decided with

the update frequency. Due to the impact of these variables on the memory requirement, the hardware limits the possible range. Regarding the parent set, the newly selected variables will need a mean and variance to initialise their distribution. We choose a zero mean and small variance for them not to impact the rest of the regression knowing that this value will quickly be updated with the Kalman filter in the subsequent steps.

The states $\theta_{j,t}$ follow a random walk update $\theta_{j,t+1} = G_{j,t+1}\theta_{j,t} + \omega_{j,t}$, where the covariance matrix for the noise terms $W_{j,t}$ follow a block discounting update; we refer to the appendix for the detailed equations. This step uses discounting factors $\delta_{j,\phi}$ for the endogenous variables and $\delta_{j,\gamma}$ for the parental ones. The multivariate Wishart also uses this discounted covariance update technique with parameter δ_w. These parameters influence the variance-of-variance of the states coefficients. Similarly, the precision λ_j follows a Gamma-Beta Stochastic volatility update which needs a parameter β_λ that influences the variance of this update. The same parameter, β_w is also present with the same role in the multivariate Wishart. These are the five hyper-parameters of the SGDLM framework. Gruber and West (2016b) studied the influence of the discounting parameters and gave theoretical bounds for the betas.

4.4 The Heterogeneous AR model explained

4.4.1 The motivations for the model

Corsi (2004); Corsi and Reno (2009); Corsi et al. (2008) developed the HAR-RV, Heterogeneous Autoregressive Realised Volatility model for two main reasons: first the model must reproduce the fat-tails, long-memory, scaling and multifractal volatility behaviours; second it must be easy to estimate and keep clear economic interpretations. They used the easy to compute Realised Volatility as a proxy of the unobservable latent volatility. Additive models were known for their clear economic interpretations and easy calibration but not known to reproduce multifractality. That was before LeBaron (2001) reproduced multifractality and long-memory with a simple three factor model where the variables represented three different and specifically chosen fre-

quencies. Hence, a simple additive model with correctly chosen frequencies can reproduce the expected stylised facts. This led to the HAR model with the three, short, medium and long frequencies defined as: day, week and month.

Different papers studied the asymmetric propagation of the volatilities between different time scales. The HARCH model introduced by Muller, Müller et al. (1997); Dacorogna et al. (1997) followed the heterogeneous market hypothesis to decompose the volatility into different time-scale dependent components to prove this asymmetry. An interpretation behind this effect is the heterogeneous objectives of markets participants, one of them being the investment horizon. In other words, the daily volatilities impact short-term traders more than long-term ones while low frequency movements impact both. A mathematical representation of this idea creates an additive cascade of the different market components. Corsi (2004) build the cascade with an $AR(1)$ models of the RV at each frequency with a coefficient to relate it to the closest lower frequency component.

This model reproduces the long-memory and multifractality behaviours without being in the long-memory model class and keeps an economic interpretation. In their paper, they studied the coefficients to learn which frequency is driving the volatility move. They use the observation that the low frequency coefficients are non-negligible to explain the daily volatility moves as an argument for the volatility cascade.

4.4.2 The original HAR-RV

The model recreates the volatility cascade by incorporating dependencies on lower scale volatility components at each frequency. Thus the return process is a function of the cascade's highest frequency component, which is the daily volatility σ_t^d here. Let us consider the returns r_t to follow:

$$r_t = \sigma_t^d \epsilon_t \,,$$

with Gaussian noise $\epsilon_t \sim \mathcal{N}(0,1)$. Let RV_t^s represents the observed Realised Volatility at t for time scale s. They define the daily RV by:

$$RV_t^d = \sqrt{\sum_{j=0}^{M-1} r_{t-j\Delta}^2}\,,$$

with Δ the sampling frequency and M the resulting number of points. By doing so they obtain a daily volatility computed from high frequency returns. The lower-frequency RVs are built by averaging the daily ones. e.g. the weekly RV is:

$$RV_t^w = \left(RV_{t-1d}^d + RV_{t-2d}^d + \cdots + RV_{t-5d}^d\right)/5\,,$$

Following the volatility cascade idea, at each time scale the unobserved partial volatility process $\tilde{\sigma}_t^s$ has an $AR(1)$ structure with a coefficient for the expected volatility at the next lower scale. If we consider three frequencies, d daily, w weekly, m monthly the cascade model reads:

$$\begin{aligned}
\tilde{\sigma}_{t+1m}^m &= c^m + \phi^m RV_t^m + \tilde{\omega}_{t+1m}^m\,,\\
\tilde{\sigma}_{t+1w}^w &= c^w + \phi^w RV_t^w + \gamma^w E_t[\tilde{\sigma}_{t+1m}^m] + \tilde{\omega}_{t+1w}^w\,,\\
\tilde{\sigma}_{t+1d}^d &= c^d + \phi^d RV_t^d + \gamma^d E_t[\tilde{\sigma}_{t+1d}^d] + \tilde{\omega}_{t+1d}^d\,,
\end{aligned}$$

where c^s corresponds to a constant, ϕ^s the coefficient of the RV variable and γ^s the coefficient representing the dependence on the closest lower scale. The noises $\tilde{\omega}_t^s$ are independent in time and between each others. By recursion we obtain:

$$\tilde{\sigma}_{t+1d}^d = c + \beta^d RV_t^d + \beta^w RV_t^w + \beta^m RV_t^m + \tilde{\omega}_{t+1d}^d\,,$$

Where β^s represent the different coefficients. With the relation between the daily latent volatility measure $\tilde{\sigma}_{t+1d}^d$ and its estimate RV_{t+1d}^d written as $\tilde{\sigma}_{t+1d}^d = RV_{t+1d}^d + \omega_{t+1d}^d$, we obtain the following time series representation of the cascade model:

$$RV_{t+1d}^d = c + \beta^d RV_t^d + \beta^w RV_t^w + \beta^m RV_t^m + \omega_{t+1d}\,.$$

Where $\omega_{t+1d} = \tilde{\omega}^d_{t+1d} - \omega^d_{t+1d}$. Hence the variance is modelled as a combination of different frequency components. Due to its simple additive form the authors suggested the use of classic least square regression to compute the parameters. In their paper they used all the available history as an increasing window on which to fit the model. Hence the first point uses only 30 days while the last one uses the whole dataset. In their original paper they also discussed using a moving window regression to obtain a time series evolution of the weight but did not show any results.

4.4.3 The extended version for leverage and volatility-of-volatility

In order to avoid negativity issues they used the logarithm of the realized-variance $log(RV^l_t)$. They also extended the original HAR-RV to model the leverage effect by adding variables at each frequency representing past positive r^+_s and negative r^-_s returns, with s the time scale. The cascade model with leverage coefficients γ for each time scale becomes:

$$\begin{aligned} log\left(RV^d_{t+1d}\right) = c \quad &+ \quad \beta^d log\left(RV^d_t\right) + \gamma^{(d)+} r^{(d)+}_t + \gamma^{(d)-} r^{(d)-}_t \,, \\ &+ \quad \beta^w log\left(RV^w_t\right) + \gamma^{(w)+} r^{(w)+}_t + \gamma^{(w)-} r^{(w)-}_t \,, \\ &+ \quad \beta^m log\left(RV^m_t\right) + \gamma^{(m)+} r^{(m)+}_t + \gamma^{(m)-} r^{(m)-}_t \,, \\ &+ \quad \omega_{t+1d} \,. \end{aligned}$$

In Corsi et al. (2008) they presented a HAR-RV model with Normal Inverse Gamma variance to model the variance-of-variance. While this model did not out-perform alternative models for all of the tests, it did prove to be the best regarding forecasts of the distribution of the realized-variance. Moreover they showed that incorporating time variation of the variance of RV, without necessarily using an inverse gamma distribution, produced better results for inference and for the fit of the RV distribution.

4.5 A complete model of the market

4.5.1 The motivations for the model

The HAR-RV model of Corsi (2004); Corsi and Reno (2009); Corsi et al. (2008) allowed them to identify which frequency, and thus which traders were influencing the stock. On the other-hand the SGDLM framework of Gruber and West (2016a,b); Zhao et al. (2016); McAlinn and West (2016) allowed them to dynamically create a graph of connections between stocks in a market and incorporate this information into a multivariate distribution of the environment. Combining those two models creates a time varying DLM for each asset with factors representing the influence of different frequencies to identify which trader is driving the moves and others for the influence of selected stocks in the market. Furthermore, Corsi et al. (2008) studied the importance of including a time varying noise component and more specifically a Normal inverse gamma distribution to efficiently model the variance-of-variance distribution. The SGDLM model uses exactly that distribution for the noise terms since it is the conjugate distribution of a Normal with unknown variance. In addition, both models follow a simple additive set-up, hence the extension of the classic HAR-RV to incorporate leverage and jumps is straightforward to include into the DLM.

Let us recall the initial variogram study. The study of the evolution of the RV while changing the returns' scale from daily to weekly allowed us to detect abnormal patterns in the time series, which were potentially related to option hedging. The combination of the HAR-RV and SGDLM models allow us to go further into this decomposition. Each time series is decomposed into two groups of factors. The exogenous factors consist of the influence of the environment on the series, and this group combines the variables selected during the parent selection phase. Secondly, the endogenous factors represent the information internal to the time series. This is composed of the previous RV and leverage variables for different frequencies.

With this decomposition we aim to understand whether the moves are being driven by external or internal factors, and also at which frequency. In addition, it is interesting to study the evolution of the spread between en-

dogenous and exogenous groups. The intuition here is that a company with a variance being pushed mainly by external factors should behave differently from one which mainly follows endogenous ones. Looking at a different scale, can we relate different market situations to phases where the stocks are being mainly driven by exogenous factors, endogenous ones or that the spread between these two is increasing. The results presented in sections 7.4 & 4.7 provide an answer of these questions.

4.5.2 Extending the standard DLM

While Corsi (2004); Corsi and Reno (2009); Corsi et al. (2008) used high frequency intra-day data to obtain a daily-variance, in this study we only make use of end-of-day prices. At this low frequency we define the s-scale RV as the sum of squared s-scale log-returns, $r^s_{j,k} = \sum_{k=t-i}^{t-i-s} r_{j,k}$, over a time window L. While using a de-meaned version of the returns is possible and won't change the results presented in this Thesis, we chose not to de-mean them because this is a variable often used in practice. With a weighting kernel w, the s-scale-RV is defined by:

$$RV^s_{j,t} = \frac{1}{s}\sum_{i=0}^{L-1} w_i \left(\sum_{k=t-i}^{t-i-s} r_{j,k}\right)^2 .$$

We will use an exponential moving average weighting kernel for w. To reproduce the HAR-RV three scales framework we will use daily ($s = 1$), weekly ($s = 5$) and monthly ($s = 20$) frequencies. If we suppose unknown RV precision $\eta_{j,t}$ as in the original SGDLM model, the resulting $RV^d_{j,t}$ is Normal-inverse-Gamma. Following Corsi et al. (2008) we use the logarithm of the RV to avoid negativity issues, hence obtaining a log-Normal-Inverse-Gamma distribution for each time series.

A DLM is defined by a distribution on the output variable and another on the state variables. Following the notations introduced for the SGDLM, we

can write the HAR-RV cascade model of one stock j as:

$$\begin{aligned} RV^d_{j,t} &= F_{jt}\theta_{j,t} + \nu_{j,t}\,, \\ \theta_{j,t} &= G_{j,t}\theta_{j,t-1} + \omega_{j,t}\,, \\ \text{with:} \quad & \nu_{j,t} \sim \mathcal{N}(0, \lambda^{-1}_{j,t})\,, \\ & \omega_{j,t} \sim \mathcal{N}(0, W_{j,t})\,. \end{aligned}$$

Where $\theta_{j,t}$ is the state vector and $F_{j,t}$ the vector of variables. The multivariate state-noise is $W_t \sim \mathcal{N}(0, V_t)$ with V_t the covariance matrix. The evolution matrix $G_{j,t}$ is diagonal with non-zero elements for the selected parents. Following the SGDLM approach we can extend this DLM to include K parents from the m stocks in the environment under study. The state vector becomes:

$$\theta_{j,t} = \left(c_{j,t}, \beta^d_{j,t}, \beta^w_{j,t}, \beta^m_{j,t}, \gamma_{1,t}, \ldots, \gamma_{K,t}\right)^T\,,$$

Where the β represent the coefficients of the endogenous variables, here the offset and three frequencies from the HAR-RV model, and the γ the coefficients of the exogeneous ones. Hence the corresponding vector of variables is:

$$F_{j,t} = \left(1, RV^d_{j,t}, RV^w_{j,t}, RV^m_{j,t}, RV^d_{sp_t(1),t}, \ldots, RV^d_{sp_t(K),t}\right)\,.$$

Extending this set-up to include the leverage effect is straightforward. The complete model will therefore be composed of internal coefficients, the realised variance and leverage effect coefficients at different frequencies, and external ones, the group of daily realised variance of the selected parents. While for the HAR-RV model they considered the variance at different scales to be the average of the daily one over different time windows we considered it to be the RV over the same time window but with different frequencies of returns. We will refer to this combined model as H-SGDLM.

4.5.3 Add OHLC data

In this section we will a present an example of a possible extension of the H-SGDLM to incorporate new endogenous variables. As a result of the simple additive structure of the H-SGDLM it is easy to extend as long as the

constraint of having Normal distributions is respected. A day of trading is often summarized with the Open, High, Low, Close (OHLC) data. Open and Close correspond to the first and last traded price of the day while, High and Low are the highest and lowest traded price of the day. We studied the correlation of different metrics combining those variables and the underlying time series of close prices. Let us note a stock price as S then the variables with the highest correlation are:

$$\begin{aligned} r_t^{low} &= log(S_t^{low}) - log(S_{t-1}^{low}), \\ CH_t &= \frac{S_t^{high} - S_t^{close}}{S_t^{high} - S_t^{low}} - 0.5, \\ COHL_t &= \frac{S_t^{close} - S_t^{open}}{S_t^{high} - S_t^{low}}. \end{aligned}$$

Following the idea of incorporating the OHLC data to make the model more responsive to recent moves we can add them to the vector of endogenous variable. Instead of increasing its side we deleted the less informative variables from the previous formulation which corresponded to the RV variables computed of a time window of a week. Thus we obtain:

$$\begin{aligned} H_{j,t} = (&1, RV_{j,t}^d, RV_{j,t}^w, RV_{j,t}^m, \\ &(r_{j,t}^d)^2, (r_{j,t}^w)^2, (r_{j,t}^m)^2, \\ &r_{j,t}^{low}, CH_{j,t}, COHL_{j,t}, \\ &r_{j,t}^{(d)+}, r_{j,t}^{(d)-}, r_{j,t}^{(w)+}, r_{j,t}^{(w)-}, r_{j,t}^{(m)+}, r_{j,t}^{(m)-})^T. \end{aligned}$$

In our simulations, incorporating OHLC variables did not improve the performance of the H-SGDLM when applied to model the log-RV. But when in section 4.7.2 we adapted the H-SGDLM to predict the logarithm of the prices on different environment the OHLC data improved the results.

4.5.4 The implementation on a GPU

Since the SGDLM framework decomposes the multivariate distribution into independent DLMs they can be updated in parallel. Gruber and West

(2016b) used a GPU implementation developed in C++ with the CUDA library from Nvidia and ran it on a cluster of GPUs. We used a simpler approach in the form of the TensorFlow library. This library developed by Google is primarily made for neural networks and deep learning architecture. Nevertheless, the wide array of functions available are all optimised to process large tensors on GPU; hence, we used TensorFlow as a library for tensor computation on GPU and implemented the H-SGDLM with it.

The two main implementation issues are firstly, the matrix inversion for the Kalman update, and secondly the size of the matrix of the full multivariate distribution. For the matrix inverse we added a regularization step to guaranty its positive definiteness before inversion and monitor its values to avoid any divergence. In addition, the scale of the variables used in the model also has an impact, which is why we rescale them. For the memory issue we used the fact that the coefficient matrix Θ_t, the update matrix G_t and the covariance matrices are all sparse. Unfortunately, at the time of writing the TensorFlow library did not have the necessary functions to perform the sparse matrix manipulations we needed. Hence, we stored a dense version of those matrices and a matrix of indices to rebuild them when needed for computation. While this is not optimal, it suffices for the purpose of testing our model.

Due to the memory consumption of these matrices and the limited memory available on a single GPU some parameters had to be bounded. More specifically, the number of allowed parents into the core, up and down sets and the number of time step between the update of those groups. The GPU memory places limits on the feasible number of Monte-Carlo samples too. One option would be to store a set of samples on the permanent memory and continue sampling on the GPU, however the added memory transfer counterbalances the gain in GPU processing. Since we only had one GPU at our disposal we could not reproduce the approach taken in Gruber and West (2016b) of distributing the sampling on multiple GPUs to allow for larger sets.

4.6 Results

4.6.1 On a simulated environment

Before studying real data we tested the algorithm in a simulated environment. Each stock prices is assumed to follow a random walk with a Brownian motion with different means and variances which are distributed randomly but constant through time. With this process we created a dataset of 300 time series. The parent selection using the multivariate Wishart does not show any clear clear pattern; the connections are noisy and seem random. This behaviour is also reflected in the exogenous coefficients, their absolute value stays low throughout with no clear patterns. All of these observations meet the key objective; namely that given a random environment with no connections between the time series, indeed the model does not find any. For the endogenous variables, the autoregressive coefficient of the RV at the previous time point stands out from the rest. Again, this was expected since the simulated time series follow a random walk.

4.6.2 Description of the dataset

In this section we use data from European stocks. The data corresponds to the 487 most liquid stocks selected from thousands of stocks in different European markets. We start the backtest from 2000 but since not all stocks have such a long trading history we assume $log(RV)$ to be zero when the price is missing. With that number of time series the available GPU memory only allows us to work 500 Monte Carlo samples. For the exponential averages we used a parameter $\rho = 0.98$ and a time window of $rv_L = 40$ days. Since we initiate the variables randomly the model needs some time to converge, so all the graphics and data shown below are between 01/06/2001 and 04/06/2019.

Regarding the update of the parental sets, the sizes are for the core set $nCore = 5$, for the up and down sets $nUp = nDown = 5$. With a time interval of $\Delta T = 10$ the core set will need a minimum of 10 days to be completely updated. These numbers were also chosen to obtain a number of

parents' variables less than 15. The objective is to have a balance between the number of endogenous and exogenous variables, as well as a low percentage of parents relative to the environment. In this section we compare the performance of our proposed algorithm with the previously published HAR-RV and SGDLM models. Since they did not use OHLC variables we won't either. In the following section 4.7 we apply the model on different investment environments.

4.6.3 Direct inference

This section compares the one step ahead inference computed from the proposed model to those from the HAR-RV and SGDLM models. Our model was developed to study the evolution of the coefficients to learn which one is driving the market, hence the quality of the one step ahead inference is used to assess the accuracy of this decomposition. We ran the algorithm on daily $log(RV)$ which produced one day ahead predictions. To put the metrics in perspective it is interesting to realize that the average daily return of $log(RV)$ over the whole dataset is 0.0, while its standard deviation is 0.28. Therefore, simply predicting a constant value for the next day already produces a good mean absolute deviation (MAD), hence we consider this case as model $t-1$ in table 7.1a. Corsi (2004); Corsi and Reno (2009); Corsi et al. (2008) assessed the out of sample performance with the R^2 of Minor-Zarnowitz, Root Mean Squared Error (RMSE) and Mean Absolute Deviation (MAD). West et al. used the MAD to assess the one point ahead forecast and the performance of a portfolio optimisation to assess the quality of the multivariate model and especially the resulting sparse covariance matrix. We ran on our dataset the HAR-RV model with leverage effect and the SGDLM model with only a constant and the previous value as endogenous variables. All the metrics in table 7.1a correspond to averages over the individual ones obtained for each time series. The similarity between the HAR-RV model and $t-1$ model can be easily explained: when we look at the values of the different coefficients in the HAR decomposition, the previous $log(RV_{t-1})$ variable has a weight close to 1.

When assessing inference of a Bayesian model, the median absolute deviation between the predicted mean of the distribution and the target time series does not take into account the quality of the inferred distribution. A more interesting metric can be computed by using the predicted confidence interval and the percentage of measured points which occurred inside it. The better the forecast, the smaller the confidence interval and the higher the percentage of points in the predicted interval should be. Since the HAR-RV employed a classic least-square parametrization, we used a multiple of the squared standard deviation of the predictions over the past 30 points to compute the confidence interval. In order to compare the performance of the three models, HAR-RV, SGDLM and ours, we used a multiple of the standard deviations to obtain similar sizes of confidence interval and thus compare the percentage of correct predictions for that interval. Regarding the choice of the interval, we selected a size smaller than the twice the average move size, i.e. for an average move size of 5% we select a confidence interval lower but close to 10%. To compare the three models we select an average confidence interval of similar size so we can compare the predicted move direction and potential confidence interval increase at the jumps. The perfect model should have very low confidence interval on average with large increase when a jump is predicted.

Since the motivation behind combining endogenous information representing the volatility cascade and exogenous information representing the influence of the market was to predict the underlying risk, we expect our model to perform well in predicting large moves. Hence, Table 7.3a shows the percentage metric for moves with an absolute return larger than a certain threshold. But, since $68,7\%$ of the moves of $log(RV)$ in our dataset are negative the quality of the model is assessed by the prediction of increases in variance. Therefore we also show in Table 7.3b the percentage of correct predictions when focusing on volatility increases above a certain threshold. In every scenarios our model correctly predicted more than 62.0% of the moves, positive or negative. While it correctly predicted 63.89% of moves larger than 9.28% and 63.95% of the increases higher than 10.04%. On the same data, for the same thresholds, the classic HAR-RV and SGDLM models never achieves more than 54.14%

and 40.41% respectively. Our combined model behaves as expected and outperforms both previous approaches for positive, negative, small and large moves. Interestingly, the bigger the moves the better the performance of our model and the bigger the gap with the other two models.

4.6.4 Evolution of the coefficients

The algorithm was built to explain what is driving the time series by decomposing it into different economic factors. We assessed the accuracy of this decomposition in the previous section by measuring the quality of the direct one step ahead inference. Going into more details, Figure 4.1 shows the evolution of the endogenous variables through time for the company Actividades de Construccion y Servicios SA (ACS), which we chose randomly out of the 487 stocks. Hence, at any time t this graphic shows which variable and frequency is mainly influencing the stocks' behaviour. We did not put the corresponding graphic obtained by the HAR-RV model since in that case only the coefficient of the previous RV value was meaningful throughout the dataset.

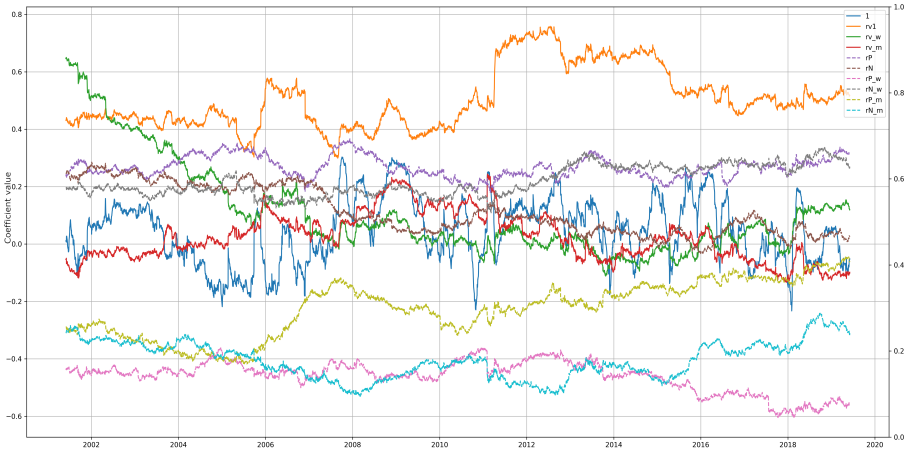


Figure 4.1: This graphic shows the evolution of the endogenous coefficients for the company ACS. The solid lines correspond to the RV at different frequencies while the dashed ones are for the leverage effect coefficients. At each time t this figure shows which variable is mainly driving the volatility. In addition, the sequential update of these coefficients allows us to observe on this figure how they evolve through time.

Table 4.1: Performance metrics of the one-day-ahead inference obtained for 487 European stocks.

Model	t - 1	HAR-RV	SGDLM	Full
Median ADV	0.010	0.012	0.015	0.013
Median RMSE	0.038	0.040	0.050	0.043
Median R^2	0.990	0.989	0.985	0.988
M-Z coefficient	N/A	0.977	1.006	0.991

(a) One day ahead inference comparison between different models. Median ADV corresponds to the Median Absolute deviation. Median RMSE is the median of the Root Mean Squared Errors. M-Z coefficient corresponds to the Minor-Zarnowitz regression coefficient, i.e. the regression of the observed values on the predicted ones.

Table 4.2: Performance metrics of one-day-ahead inference obtained for 500 European stocks.

Model		HAR-RV	SGDLM	Full
without threshold	Interval size	3.96%	3.97%	3.95%
	% in	64.55%	49.43%	62.52%
with mean$(\|r\|) = 5.73\%$	Interval size	10.75%	10.74%	10.71%
	% in	55.41%	37, 43%	62.38%
with mean$(\|r\|) = 7.55\%$	Interval size	14.22%	14.20%	14.20%
	% in	53.43%	34.5%	62.33%
with mean$(\|r\|) = 9.28\%$	Interval size	18.02%	18.04%	18.06
	% in	53.24%	34.69%	63.89%

(a) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for the selected absolute-returns above a certain threshold. The first columns shows the average absolute-return, mean$(|r|)$, of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean$(|r|)$. This table shows the ability of our model to predict large, positive and negative, moves of any size.

Model		HAR-RV	SGDLM	Full
with mean$(r) = 6.92\%$	Interval size	13.06%	13.11%	13.10%
	% in	56.30%	43.75%	62.69%
with mean$(r) = 8.48\%$	Interval size	16.63%	16.71%	16.63%
	% in	55.80%	43.30%	64.12%
with mean$(r) = 10.04\%$	Interval size	19.71%	19.71%	19.73%
	% in	54.14%	40.41%	63.95%

(b) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for selected returns above a threshold. The selection corresponds to only positive moves. The first columns shows the average return, mean(r), of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean(r). This table highlights the ability of our model to predict large upward moves of any size.

Figure 4.2 shows the evolution of the exogenous coefficients through time for the company ACS. The variables correspond to the parents which are selected dynamically with the two step process described in Section 4.3.2. Hence, as the coefficients evolve with the selected parents, the new ones will start close to zero. This effect is responsible for the observable mean reversion on the graphic. As with the endogenous variables, at any time t this graphic is showing which stock in the market is mainly influencing the move. An interesting aspect we can see on this Figure 4.2 is the decrease in activity of these coefficients between 2010 and 2016 after the financial crisis. This observation motivated us to study how the mean absolute values of these coefficients evolve through time.

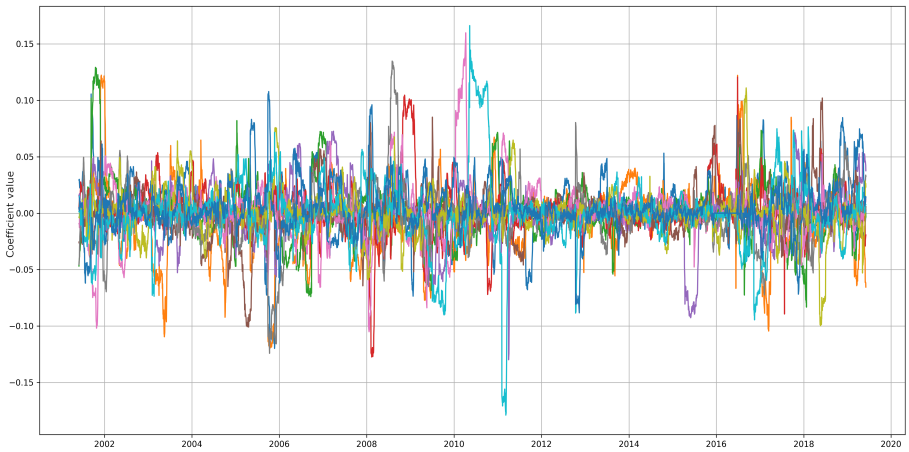


Figure 4.2: This figure shows the evolution of the exogenous coefficients for the company ACS. At each time t the lines correspond to the value of the coefficients of the selected core-parent group variables. It shows at each time t which parent is mainly influencing the volatility of ACS. Also, it shows how the coefficients of the different exogenous variables evolve through time.

To understand how these patterns are related to the underlying time series we clustered the coefficients into three groups: a first group including all the RV variables, rv, a leverage one for the leverage effect coefficients, r, and an exogenous group, $core$. These groups are constructed by summing the absolute values of the individual coefficients. We then sum the values of the individual groups to obtain a market view. The log-realised-variance, $log(RV)$, of the market is modelled by an average over the individual ones.

Figure 7.1 shows the evolution of the different groups and the market on the same graphic. Even from this low scale view we can observe correlations between the behaviour of these different variables; not only between the market and each group separately, but also between the market and the spread between the different groups.

Before commenting on Figure 7.1, it is interesting to think about what we should observe. We expect a stock to behave differently when its move is driven more by exogenous factors then endogenous ones and vice-versa. More interestingly, when the behaviour of a stock is changing, e.g. going from endogenously to exogenously driven, that should impact its variance. One interpretation is that when the variance of the stock is driven by endogenous factors its behaviour is mainly influenced by internal information and thus less sensitive to the market. Hence, it should be more predictable which should result in a lower variance. On the other hand, when the contribution of the exogenous factors increases that may be correlated with an increasing influence of the market and thus uncertainty on that company which should result in higher variance.

In Figure 4.2 we could observe a recent increase in activity of the exogenous variables from the end of 2016. In the market view of Figure 7.1 we can clearly see the spike of the exogenous group at the end of 2016, followed by a spike in the variance of the market. While Figure 7.1 is interesting for having a global view and defining the relations between the different groups and the market, the approach described in the next section will give a clearer view of the strength of this relation.

4.6.5 Change point algorithm

We want to leverage these correlations between the different groups and the underlying time series to make predictions. To do so we will focus on predicting either an increase, decrease or neutral move. Since we previously observed that each group seems to be correlated to the time series at different frequencies we used a simple discrete derivative approximation to detect a

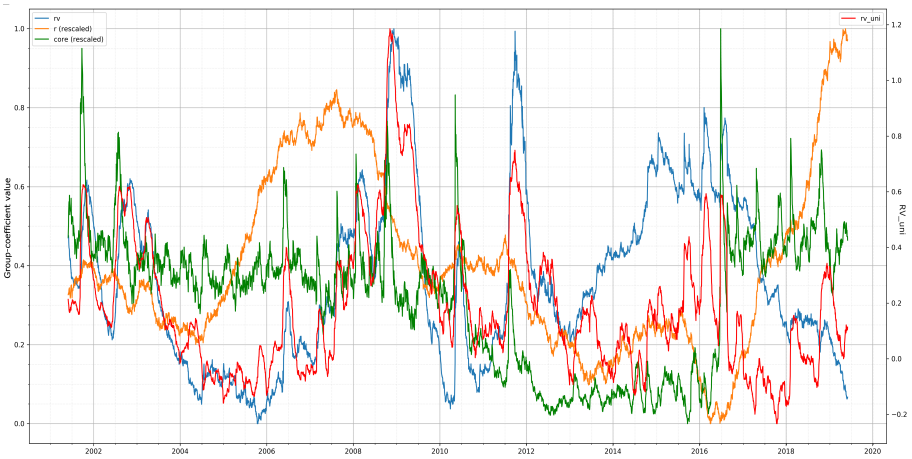


Figure 4.3: Plots computed with data from 500 European stocks. The graphic shows the evolution of the different groups (endogenous, with RV and R, and exogenous, denoted Core) rescaled to be on the same $y-axis$, which is left on the figure. The right axis corresponds to the approximated $log(RV)$ of the market, i.e. an equal weighted average of the individual $log(RV)$. This graphic highlights the connections between the behaviour of the market, represented here by $log(RV)$, and the different groups: RV, R and core. For example, looking at the volatility jump of the financial crisis of 2008 the RV and Core group seem positively correlated with the market while the leverage group seems negatively correlated. What this graphic cannot show is if these connections have any predictive power over the volatility of the market or if they simply react to it with a time lag, Figure 7.2 shows this predictive relation.

change of trend for each of those frequencies. The derivative is approximated by the difference between the value at t and with a lag at $t - l$. We only considered multiples of 5 days for the lag l. For example, for daily moves we used the data up to year 2010 to find time lag l, for which the endogenous group was most correlated to $log(RV)$; this turned out to be one month $l = 20$. At each time t, we look at the sign of the difference between the value of the endogenous group at time t and $t - 20$ to infer whether the $log(RV)$ will increase, decrease or stay neutral for the next day.

With this set-up we looked at the correlation between the moves of the groups and $log(RV)$ at different frequencies: daily, two days in advance, weekly, by-weekly and monthly; i.e. lags=$\{1, 2, 5, 10, 20\}$. We used this change point approach to produce inferences on the variance at those different frequencies. Each signal produces a value in the set $\{-1, 0, 1\}$ which corresponds respectively to a predicted decrease, constant or increase in variance at the frequency under study. Hence when we use a monthly lag, i.e. 20 we take one

points of the data set every 20 and use the difference of the signals to predict the move for the next time point, i.e. 20 days later. In other words, we make one month ahead prediction. Interestingly, each group produces their most accurate prediction at different frequencies. Since each signal is evolving at a different frequency and related to a different economic information, they provide different insight on the behaviour of the underlying time series. We sum all of them to obtain the final inference. To reduce the signals' volatility we can add a threshold representing the confidence on the predicted value. For example, by putting a threshold of 2, the absolute sum of the signals must be higher than 2 for a position to be entered i.e. more than 2 different signals need to agree on the same direction.

Using the previously described logic, we performed a backtesting on the underlying $log(RV)$ to demonstrate the quality of the prediction. we use the first 70% of the datasets to select the change point thresholds, but only limit the range of selectable values to $1, 2, 5, 10, 20$. The graphics and metrics shown represent the whole dataset, both in and out of sample combined. In order to assess the performance for the whole environment we constructed an equally weighted portfolio and compared it to the market. The linearity observable in Figure 7.2 shows that the quality of the prediction is constant through time. Also, if we allow for only two states be it either short-only, $\{-1, 0\}$, or long-only, $\{0, 1\}$, they both perform as well, hence the prediction does not show any bias toward a certain move direction. Another interesting aspect resides in the prediction for different time scales. Running the change point for different time scales on the group of RV coefficients and computing the EW portfolio for each of them produces Figure 4.5. It represents the performance of the EW portfolio using signals created with the coefficients studied at the different lags (in days) $\{1, 2, 5, 10, 20\}$. E.g. a lag $s = 20$ corresponds to an update of the prediction every 20 days. The usual issue with long term predictions is the potential size of the drawdowns due to the re-sampling frequency. While more volatile than the lower frequencies the portfolio with monthly update has small draw-downs.

Let us recall the question raised in the previous section: is the variance in-

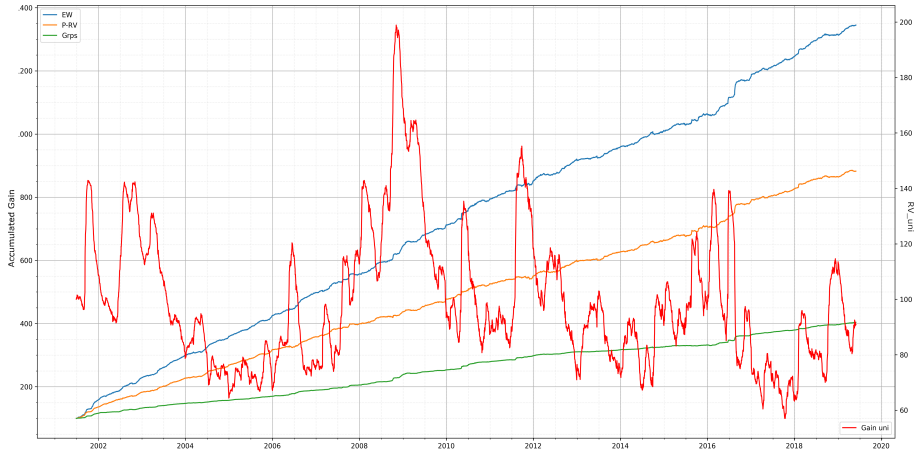


Figure 4.4: Plots computed with data from 500 European stocks. The orange line shows the performance of an EW portfolio built from the one-day-ahead inference. The green line corresponds to the performance of the EW portfolio using the signals from the change point algorithm applied on the different groups. The blue line shows the performance of the EW portfolio built by combining the one-day-ahead direct inference with the signals from the change points at different frequencies. The simulated portfolios correspond to equal weighted portfolios over the individual time series. In order to highlight the constance through time of the quality of the inference we added in red and on the right Y-axis the evolution of the underlying $log(RV)$ of the market.

creasing when the spread between the exogenous and endogenous variables increases or decreases? Following this change point set-up Figure 7.4 shows the difference between the exogenous group minus the endogenous ones for the European stocks we are studying. For the different frequencies we looked at, $\{1, 2, 5, 10, 20\}$, the variance increases when the exogenous groups become more important than the endogenous one, and vice versa. In other word the behaviour of the variances is positively correlated with the spread exogenous minus endogenous groups. This corresponds to the results we expected, i.e. the more a variance is being driven by external factor the more uncertain its behaviour is and thus its variance increases, while if the endogenous coefficient are dominants the stock moves due to information proper to itself and thus in a more predictable manner which decreases the variance. However, as we can see on Figure 7.4 the quality of that connection is not constant with time and depends on the market conditions.

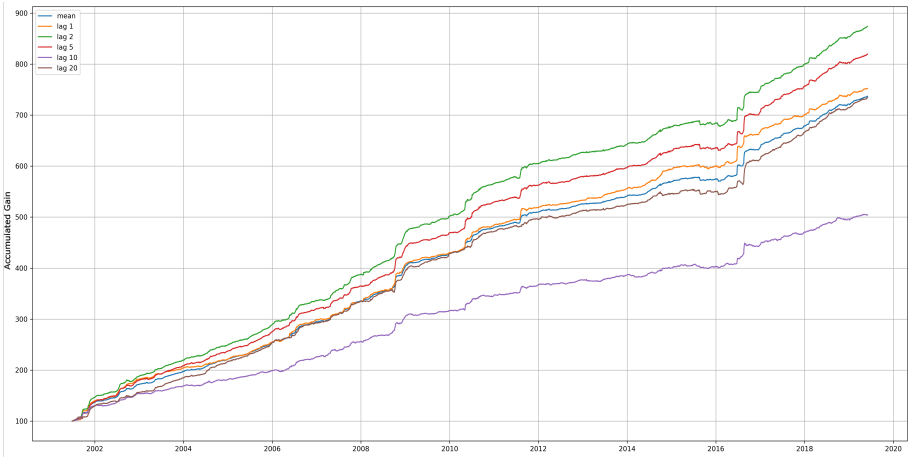


Figure 4.5: Plots computed with data from 500 European stocks. This figure plots the performance of the EW portfolio computed with the signals from the group of external coefficients (core group) created with the change points at different frequencies. This graphics shows that while the portfolios using signals updated at lower frequencies than daily are more volatile they still have a linear performance through time and small draw-downs.

4.6.6 Risk metric application

While the previous section focused on the direct inference of $log(RV)$, this section will discuss the application on market stress. The SGDLM model uses a multivariate Wishart distribution to compute a graph of the market and select the parents for each DLM. With this selection each DLM only has a few parents representing the stocks with the highest influence. We expect to see stocks that are driving a sector and others that are followers. In Figure 4.7 we plotted the evolution of the matrix of parents' relation through time. At each time t we counted the number of core parent sets each stock is part of. The higher the number of sets the blacker and vice-versa. By doing so, we want to identify the stocks that are in many core-sets and thus influencing the environment. Indeed, by looking at this matrix we can identify few stocks that have a clear influence in the market. Another way to see this would be to consider the market risk to be concentrated on a few stocks which moves could have systemic impacts. We can also observe a clear change occurring after the financial crisis and after 2016.

The Figure 4.7 is an interesting argument in favour of an asymmetric sparse

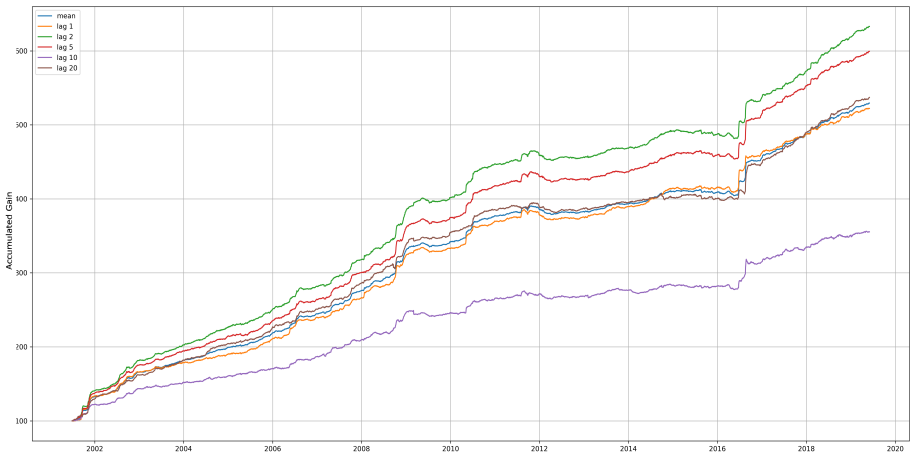


Figure 4.6: Plots computed with data from 500 European stocks. The figure shows the performance of the EW portfolio computed with the signals from the difference between the external and the RV groups using the change points algorithm at different frequencies. Hence those plots show the positive correlation between the difference *(exogenous coefficients - endogenous coefficients)* variables and the evolution of the volatility. In other words, when the volatility becomes more driven by exogenous factors than endogenous ones the volatility increases and vice-versa.

covariance matrix. Indeed with the classic multivariate approach modelling the cross-relationships with a symmetric covariance this matrix would not exist. This assymetry exists because the SGDLM decomposes the parent update into two phases, the first using the multivariate Wishart to select the potential candidates from the precision matrix; and the second computing the variables' coefficients as a regression in each individual DLM to predict the move of the stock and selecting the parents according to their signal-to-noise ration. And for a stock to stay in the core group, which Figure 4.7 represents, it needs to pass the two phases i.e. have a high precision element in the multivariate Wishart, and a high signal-to-noise ratio when included in the up set of the SGDLM and part of the DLM's variables. Thus, the coefficients of the core-group variables represent the strength of the relationship between the stock and its selected parents; and the parents selected in the core group the stocks with the highest signal-to-noise ratio.

In Corsi et al. (2008) they modified the original HAR-RV to model the vol-of-vol, or variance of the variance. This vol-of-vol represents the variance of the DLM and hence the confidence in its prediction and decomposition.

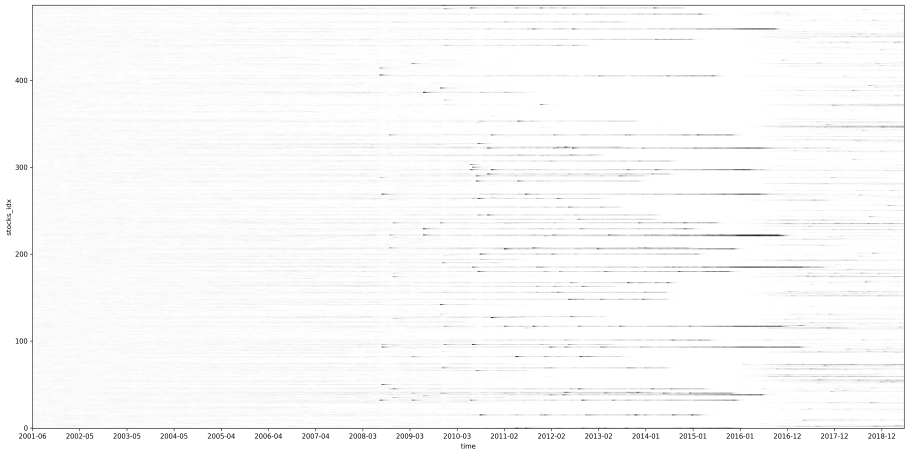


Figure 4.7: Plots computed with data from 500 European stocks. The $y - axis$ corresponds to the index of each of the 487 stocks while the time is on the $x - axis$. This graphic is a 2-dimensions representation of the evolution of the matrix of core-parental-set through time. For each stock, at each time t the darker the point the more core-parent sets it is part of. Hence, with this graphic we can observe at each time t which stocks are influencing many others, that corresponds to the ones with dark points. While, the stocks in white are followers; i.e. they are not influencing the move of any other stock in this environment.

If that variance increases, either it is due to a bad fit from the DLM or it represents an increase in uncertainty for the variance of that stock. The inferred variance of variance is on average much lower than the computed one although it does spike during jumps of variances. In addition, the percentage of correct predictions shown in table 7.3b when restricting the analysis to large moves is another measure of the quality of the inferred variance of variance.

Gruber and West (2016a) used the KL divergence as a market stress indicator and compared it to the St. Louis Fed Financial Stress Index (STLFSI). They observed that both metrics increased at the same time to the point of considering their metric to be better than the STLFSI index since it reacted faster. It turns out this index is quite similar to the computed estimate of the market $log(RV)$. Thus, we used the change point algorithm on the different group variables described previously to produce an inference of this index. As explained previously, we can add a threshold on the value of the sum of the added signal, to show how that can impact the EW portfolio; Figure 4.8 used a threshold of 4, which means the absolute sum of the signals must be

higher than 4 for a position to be taken. Enforcing such a high threshold diminishes the number of trades. But, Figure 4.8 shows the quality of the prediction after 2007. It is not clear why the prediction between 2004 and 2007 stagnate, could be due to the low variability of the STLFSI itself.

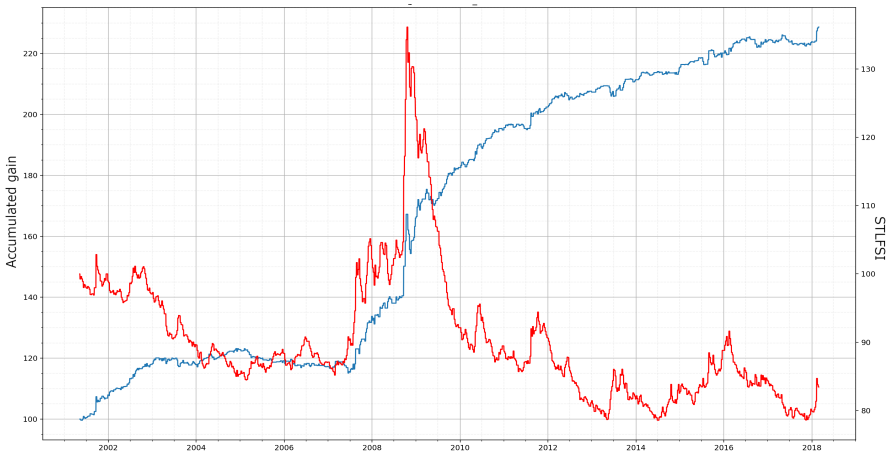


Figure 4.8: Plots computed with 390 stocks from the S&P500. The left axis is linked to the blue line that represents the performance of the portfolio computed with the signals from the change point algorithms using the evolution of the different groups to predict the evolution of the STLFSI index. In red and linked to the right axis is the STLFSI stress index re-based to 100 at the start of the backtest. This graphic shows the quality of the prediction of the stress index using the signals obtained from the different groups from 390 stocks from the S&P500.

4.7 Application of the algorithm on different environments

4.7.1 Application on the stocks from the S&P500

With exactly the same set-up as in section 7.4 for EU stocks, i.e. without OHLC data, we ran the algorithm on US stocks from the S&P500. While we used different stock exchanges and countries to select the most liquid European stocks for the US market we use a different methodology. We select the stocks currently present in the S&P500 index but only select the 376 stocks that have a history since 2000.

Because all the stocks in this environment belong to the same stock index we expect to see more connections between the time series in the index and

thus the SGDLM model to perform much better than in the European environment which mixed many stock exchanges and countries. In terms of metrics the direct one step ahead prediction does not perform as well as for the European environment which is probably due to the lower amount of information contained in the HAR variables. We followed the same methodology as described in section 4.6.3 for European stocks. As for the European stocks, in the case of large moves our model correctly predicted more than 60.0% of the moves, positive or negative. While it correctly predicted 65.93% of moves bigger than 11.25% and 63.83% of the increases in variance larger than 20.27%. On the same thresholded-data, the classic HAR-RV never achieves more than 55.86% accuracy and, while the SGDLM performance is close to our model for small moves the gap increases proportionally to the size of the moves considered. Table 7.5a shows the percentage metric for moves with an absolute return larger than a certain threshold. And, Table 7.5b the percentage of correct predictions when focusing on volatility increases alone.

Figure 4.9 shows the evolution of the different groups rescaled to be on the same graph and compared to the averaged $log(RV)$ of the S&P500. Recall that on EU data as shown in Figure 7.1 the evolution of the different groups seemed to be positively correlated to the variance of the environment. On the US data this correlation seems just as strong as we can see in Figure 4.9. We also perform the same backtesting strategy using the signals from the change point algorithm and show in Figure 4.10 the EW portfolio versus the $log(RV)$ of the market. As previously the performance is constant through time even during the financial crisis, highlighting further the robustness of our model.

As previously, we wanted to answer the question: is the variance increasing when the spread between the exogenous and endogenous variables increases or decreases? To do so we computed the EW portfolio from the signal build for the different frequencies $\{1, 2, 5, 10, 20\}$ (in days) using the difference between the exogenous and endogenous groups. Figure 4.11 shows the results for those different frequencies. As in the EU market the variance is positively correlated to the difference between these two groups. And the link

Table 4.4: Performance metrics of the one-day-ahead inference obtained for stocks from the S&P500.

Model	t - 1	HAR-RV	SGDLM	Full
Median ADV	0.011	0.014	0.018	0.015
Median RMSE	0.040	0.041	0.047	0.044
Median R^2	0.991	0.991	0.988	0.990
M-Z coefficient	N/A	0.995	1.007	0.996

(a) One day ahead inference comparison between different models. Median ADV corresponds to the Median Absolute deviation. Median RMSE is the median of the Root Mean Squared Errors. M-Z coefficient corresponds to the Minor-Zarnowitz regression coefficient, i.e. the regression of the observed values on the predicted ones.

Model		HAR-RV	SGDLM	Full
without threshold	Interval size	4.37%	4.37%	4.39%
	% in	59.02%	62.47%	62.57%
with mean$(\|r\|) = 5.74\%$	Interval size	10.18%	10.16%	10.19%
	% in	53.72%	60.32%	61.06%
with mean$(\|r\|) = 11.25\%$	Interval size	22.07%	22.00%	22.04%
	% in	54.21%	64.56%	65.93%
with mean$(\|r\|) = 19.96\%$	Interval size	39.35%	39.34	39.40
	% in	50.23%	61.00%	63.26%

(b) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for the selected absolute-returns above a certain threshold. The first columns shows the average absolute-return, mean$(|r|)$, of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean$(|r|)$. This table shows the ability of our model to predict large, positive and negative, moves of any size.

Model		HAR-RV	SGDLM	Full
with mean$(r) = 6.97\%$	Interval size	12.18%	12.18%8	12.19%
	% in	55.86%	59.80%	60.49%
with mean$(r) = 11.92\%$	Interval size	22.02%	22.05%	22.06%
	% in	52.07%	61.08%	62.40%
with mean$(r) = 20.27\%$	Interval size	40.02%	40.08%	40.08%
	% in	51.13%	61.96%	63.83%

(c) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for selected returns above a threshold. The selection corresponds to only positive moves. The first columns shows the average return, mean(r), of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean(r). This table highlights the ability of our model to predict large upward moves of any size.

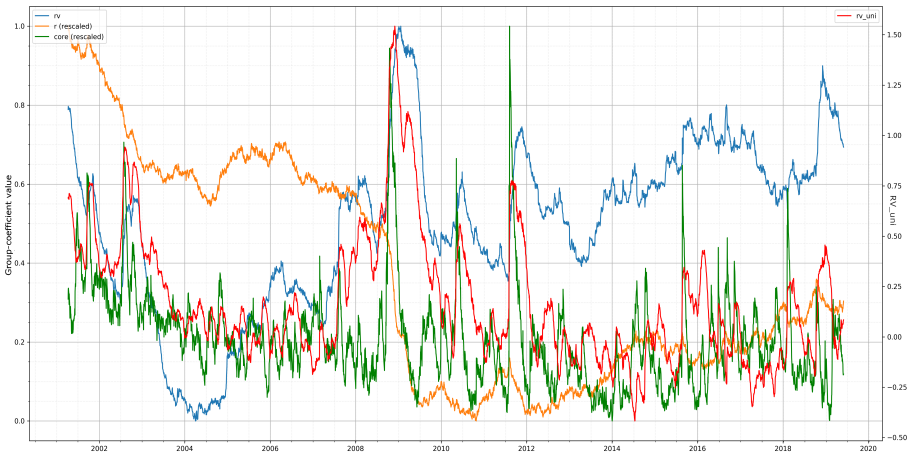


Figure 4.9: Plots computed with stocks from the S&P500. The graphic shows the evolution of the different groups (endogenous, with RV and R, and exogenous, denoted Core) rescaled to be on the same $y - axis$, which is left on the figure. The right axis corresponds to the approximated $log(RV)$ of the market, i.e. an equal weighted average of the individual $log(RV)$. This graphic highlights the connections between the behaviour of the market, represented here by $log(RV)$, and the different groups: RV, R and core. For example, looking at the volatility jump of the financial crisis of 2008 the RV and Core group seem positively correlated with the market while the leverage group seems negatively correlated. What this graphic cannot show is if these connections have any predictive power over the volatility of the market or if they simply react to it with a time lag, Figure 4.10 shows this predictive capacity.

seems even stronger than the EU market since the backtest is more linear and smoother than in Figure 7.4. Naturally, on the lower monthly-frequency the connection is not as robust as for the higher ones.

4.7.2 Applying the model to predict prices of futures

In the previous sections we focused predicting the log-realised-variance of US and EU stocks without using OHLC data, in this section we will show the results on different environments of futures with OHLC data. It is important to study the scale of the different input-variables when applying this algorithm to different datasets. To guarantee the numerical-stability of the computation the variables should have similar scales. Also, it is important to modify the number of allowed parents to retain sparsity relative to the size of the new environment.

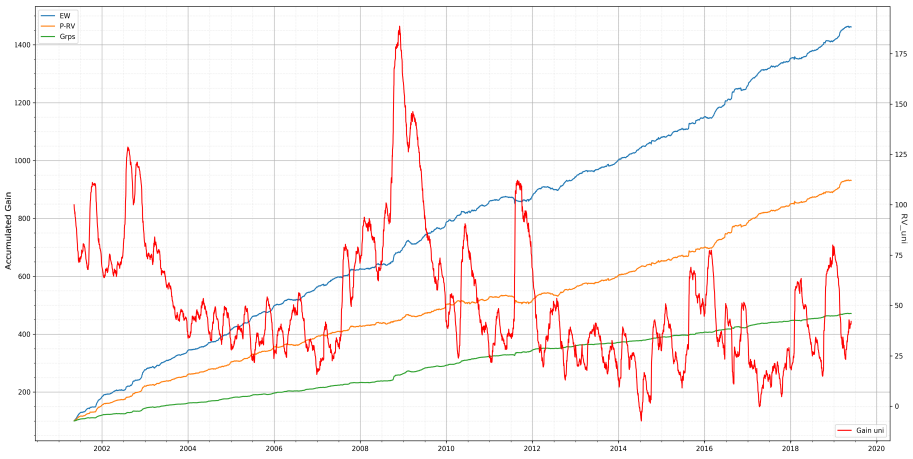


Figure 4.10: Plots computed with stocks from the S&P500. The orange line shows the performance of an EW portfolio built from the one-day-ahead inference. The green line corresponds to the performance of the EW portfolio using the signals from the change point algorithm applied on the different groups. The blue line shows the performance of the EW portfolio built by combining the one-day-ahead direct inference with the signals from the change points at different frequencies. The simulated portfolios correspond to equal weighted portfolios over the individual time series. To highlight the consistency through time of the quality of the inference we added in red and on the right Y-axis the evolution of the underlying $log(RV)$ of the market.

While we previously worked with the variance, we can adapt the framework to work directly with prices. Following the logic of the information cascade we replace the different realised variance variables by the mean of the prices at these same different frequencies: daily, weekly and monthly. In addition, instead of incorporating the previous squared returns, we use an exponentially weighted average of the returns over the same frequencies. On the other hand, the leverage effect coefficients are not modified. Thus we keep the decomposition into groups with different and complementary economic information.

In order to demonstrate the robustness of our approach we now apply the model on ETFs and FX futures. We dispose of 75 futures on ETF and 22 on FX-rates. Compared to the previous examples we only had to modify the number of allowed parents to adapt to the smaller dataset. We chose a number of parents in the core set of 5 for both the ETFs and FXs environment, but a up-set size of 5 for the ETFs and 3 for the FXs. ETF and FX

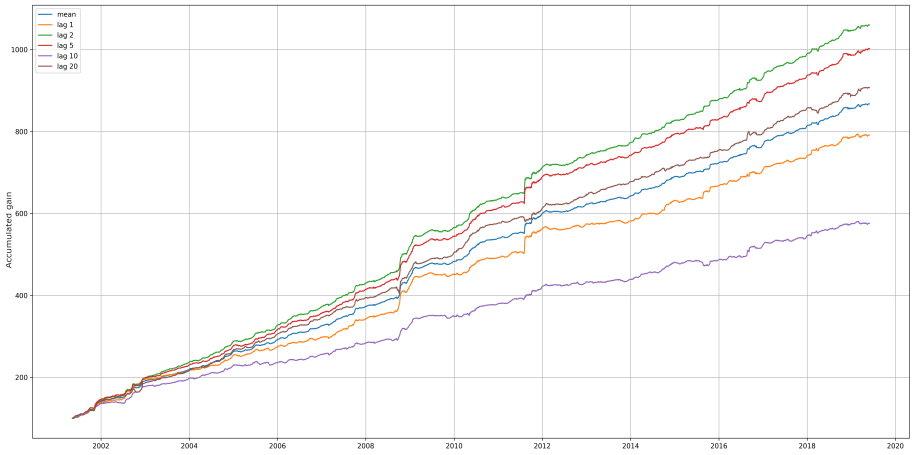


Figure 4.11: Plots computed with stocks from the S&P500. The figure shows the performance of the EW portfolio computed with the signals from the difference between the external and the RV groups using the change points algorithm at different frequencies. Hence those plots show the positive correlation between the difference *(exogenous coefficients - endogenous coefficients)* variables and the evolution of the volatility. In other words, when the volatility becomes more driven by exogenous factors than endogenous ones the volatility increases and vice-versa.

futures have different behaviours from stocks but still follow this influence of endogenous and exogenous information. Hence the model is still able to capture those different connections and create meaningful signals. To show the flexibility and robustness of our proposed model we ran the algorithm on weekly data instead of the daily one. I.e. we selected one point every five days and considered this new dataset as the input for the algorithm. Thus the one step ahead inference now corresponds to one week ahead. We used the EW portfolio to assess the performance on these datasets. The combination of signals built from endogenous and exogenous groups at different frequencies produces a portfolio with low volatility especially on large downward moves, as can be seen in Figures 4.12 for the ETF and 4.13 for the FX. As we used weekly data for this example the median length of trades was 2 weeks for both environments.

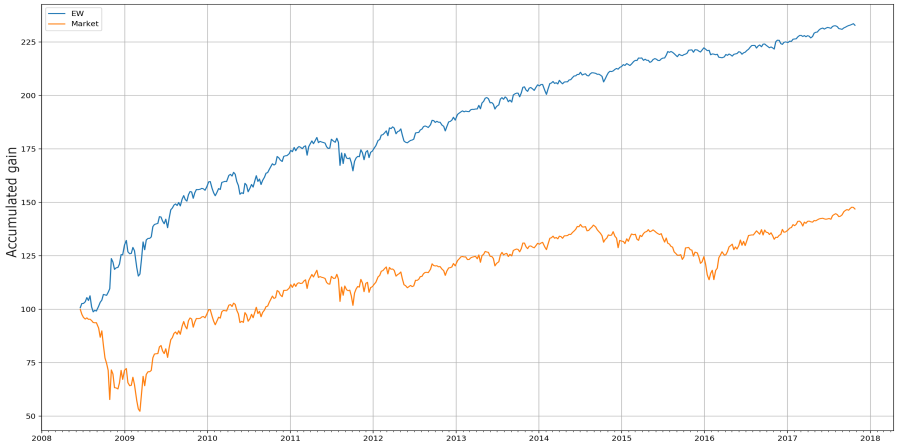


Figure 4.12: This graphics used data from 75 futures on ETFs. The yellow line corresponds to the performance of an EW portfolio on all the ETF of the environment. The Blue line corresponds to the performance of our EW portfolio computed using the signals from the change point algorithm on the different groups at different frequencies. The performance of our portfolio shows the quality of the inference especially during large decreases such as the one during the 2008 crisis.

4.8 Conclusion

In this paper we proposed a new model that gives a better understanding of the underlying cause of price fluctuations and what it reveals about a specific asset and the market as a whole. In particular, we proposed a heterogeneous autoregressive time varying simultaneous graphical multivariate dynamic linear model, H-SGDLM. This model can efficiently model large-scale environments including the cross-series relationships. It is easy to fit, does not assume stationarity of the underlying time series, and updates itself sequentially. The proposed approach decomposes the price movement into different known underlying economically-meaningful variables to produce a better understanding of the behaviour of the collection of time series; it can explain at any time t which ones are likely to be driving the moves, for example, at which trading frequency and if changes are due to endogenous or exogenous factors.

We assessed the quality of our decomposition by looking at the forecasting performance of our model. We showed that it outperformed previous models

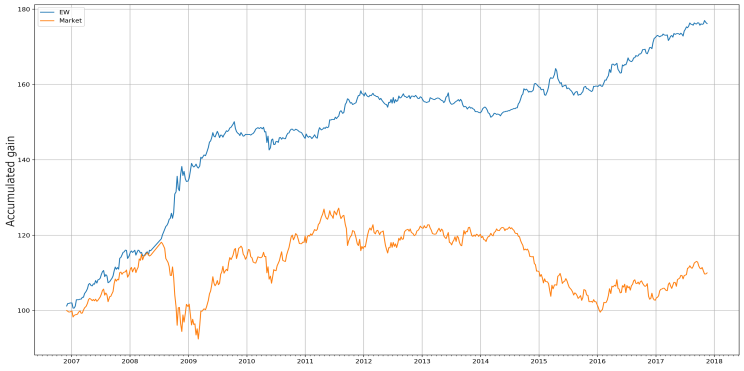


Figure 4.13: This graphics used data from 22 futures on ETFs. The yellow line corresponds to the performance of an EW portfolio on all the ETF of the environment. The Blue line corresponds to the performance of our EW portfolio computed using the signals from the change point algorithm on the different groups at different frequencies. The performance of our portfolio shows the quality of the inference especially during large decreases such as the one during the 2008 crisis.

in predicting large moves; positive and negative ones alike. In addition, the quality of these predictions are relatively constant through time, within different markets and for multiple assets types. This decomposition creates new insights into the behaviour of a specific asset and the rest of the market. In order to leverage those new signals, we clustered them into endogenous and exogenous groups. By studying the link between those groups and the underlying time series we could obtain accurate long-term predictions that worked well both for individual assets as well as the market as a whole. This allowed us to produce more than a month ahead inference regarding the evolution of the variance of the asset and the market. Furthermore, these signals were shown to be reliable enough to predict the STLFSI stress index while having learned only on a few hundred stocks.

In this paper we focused on using the signals from the endogenous and exogenous groups, although our multivariate model can also produce many other interesting outputs such as an simultaneous directed cross-series graph, illustrating relationships between assets. This provides many interesting directions for future research using our proposed model.

5

Mean reverting portfolio

5.1 Introduction

Finding a linear combination of stocks that mean revert is an econometric problem that has been captivating researchers for years, since such portfolios with reasonably predictable short term behaviour can be used for developing systematic trading strategies. When applied to two assets, this is often referred to as looking for trading pairs. Such combinations however may be hard to find due to non-stationarity and combinatorial considerations, especially for baskets of more than two assets. Furthermore, finding a mean reverting portfolio is not sufficient in itself; indeed, to be viable the effect must persist over a long enough time horizon and be strong enough to overcome trading costs.

Currently, the classic approach to this problem relies on cointegration theory (see Johansen (2005) for an overview), which tries to find a linear combination of non-stationary processes that is stationary. These methods usually rely on statistical tests to check that the time series are cointegrated. However, as stated in Johansen (2005) such tests strongly depend on the modelling

assumptions of the data. In this paper we are interested in the case of choosing a sparse subset of more than two assets from a much larger universe of assets. Recent advances in this area have proposed a new formulation of the problem (d'Aspremont, 2010; Cuturi and d'Aspremont, 2013, 2015), which relies on a characterisation of portfolio predictability (Box and Tiao, 1977). With this definition, a mean reverting portfolio (MRP) is characterised by a low predictability, in contrast to a momentum portfolio which has high predictability. In these three papers, the authors reformulated the MRP definition as an optimisation problem with constraints and compared it to other methods, such as minimising the portmanteau test or minimising the crossing statistic. They emphasised the importance of a good MRP to be sparse and argued in favour of an additional minimum variance constraint to select a portfolio with a variance high enough for the effect to be tradable.

The approach of d'Aspremont (2010) and Cuturi and d'Aspremont (2013, 2015) relies on the hypothesis of Box and Tiao (1977), which requires a model for the price process s_t conditional on all previous information. The authors assumed that the assets follow a vector autoregressive model, $VAR(p)$, of order p and that the covariance matrix of the portfolio be stationary. In d'Aspremont (2010), and subsequently Sipos and Levendovszky (2013) and Fogarasi and Levendovszky (2013), they also assumed that the resulting portfolio follows an Orstein-Uhlenbeck process and made the parallel between the mean reversion parameter and the predictability variable. In order to solve this optimisation, Box and Tiao (1977) rewrote the problem as a generalised eigenvalue problem, to which d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) then added a sparsity constraint, making this combinatorial problem NP-hard, and proposed to solve this minimisation with a semi-definite relaxation approach. As an alternative, Sipos and Levendovszky (2013); Fogarasi and Levendovszky (2013) proposed a simulated annealing algorithm to find the solution. Both methods produce approximations of the solution, are expensive to compute and lack scalability. Finally, given the resulting MRP, Cuturi and d'Aspremont (2013, 2015) followed a strategy introduced by Jurek and Yang (2007) to trade a stationary autoregressive processes, whereas Sipos and Levendovszky (2013); Fogarasi and Levendovszky

(2013) assumed the portfolio follows an Ornstein-Uhlenbeck process and thus used decision theory to implement their trading strategy.

In this paper we extend the approach of d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) by characterising a sparse mean reverting portfolio according to its predictability but without assuming a VAR(p) process. Instead we will leverage the results of a heterogeneous simultaneous graphical dynamic linear model (H-SGDLM) to set up the MRP optimisation problem for assets following time varying autoregressive TVAR(p) processes. The H-SGDLM builds a sparse covariance matrix which is not assumed stationary and our new formulation of the MRP minimisation problem with normalisation constraint allows us to use an efficient optimisation to compute the global minimum.

The H-SGDLM of Griveau-Billion and Calderhead (2019); Gruber and West (2016a); Corsi (2004) models the entire investment environment by decomposing each time series, and thus the whole market, into economically meaningful variables. This decomposition creates different groups of variables representing the endogenous and exogenous influences on the behaviour of each asset. Many different signals are created by this model; for this paper we are mainly interested in the graph of connections between the different assets and the resulting sparse covariance matrix and autoregressive coefficients. These variables will allow us to propose a new formulation of the sparse mean reverting portfolio optimisation problem.

While the previous approaches of d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) and Sipos and Levendovszky (2013); Fogarasi and Levendovszky (2013) obtained approximations of the solution to a constrained optimisation problem, we propose a new algorithm that finds the global minimum. We employ a cyclical coordinate descent (CCD) algorithm which is fast and efficient in large-scale problems. Furthermore, Tseng (2001) proved the convergence to the global minimum of cyclical coordinate descent algorithms when the the utility function can be written as the combination of a convex differentiable function and a non-differentiable but additively separable one.

Building on this result, Yen and Yen (2014) used the CCD algorithm to solve the mean variance portfolio with an elastic-net constraint. In addition, Griveau-Billion et al. (2013) used this CCD method to solve the non-convex equal risk contribution portfolio optimisation problem, where it was observed to be faster than the Newton algorithm in large environments. We follow the direction of these approaches and derive a CCD algorithm to solve the sparse MRP minimisation problem with a normalisation constraint. The algorithm we propose can be applied to compute a MRP from any predicted covariance matrix. However, we focus in this paper on the predictions obtained from the H-SGDLM model.

Using this new MRP solution we can then consider a number of different trading strategies. Hence we will numerically compare the approaches proposed by Jurek and Yang (2007) and Sipos and Levendovszky (2013); Fogarasi and Levendovszky (2013), as well as a combination of the two. In order to assess the performance and robustness of our solution we computed the sparse MRP and implemented the trading strategy on three different datasets: stocks from the S&P500, futures on FXs and futures on ETFs. All of the backtests we obtained exhibit a linear growth throughout the time under study, which includes the financial crisis of 2008. While in some environments the backtests are smoother than others, the fact that none has a particularly large drawdown, and they are all roughly linear through time, highlights the robustness of our proposed approach.

Section 5.2 will recall the sparse mean reversion optimisation problem and briefly review the current approaches for solving it. Section 5.3 will then detail the H-SGDLM model and the novel formulation of the MRP problem it allows. The CCD algorithm and its application to solve the new MRP optimisation are detailed in section 5.4. Once the optimal solution is found we consider possible trading strategies, as detailed in section 5.5. Finally, in Section 7.4 we present results for different assets classes.

5.2 Description of the sparse MRP optimisation problem

5.2.1 The MRP as a constraint minimisation problem

In this paper we build on the approach introduced by d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) for characterising a mean reverting portfolio. The authors used the predictability ratio defined for stationary processes by Box and Tiao (1977) and extended it by adding a cardinality constraint and a minimum variance constraint, in d'Aspremont (2010) and Cuturi and d'Aspremont (2013, 2015), respectively. Let us assume a univariate model such that the asset s_t follows the process $s_t = \hat{s}_t + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$ represents i.i.d. noise, and $\hat{s}_t$ is an estimator for x_t, which can use all the information available up to $t-1$. With this formulation, Box and Tiao (1977) used the variance $\sigma^2 = \hat{\sigma}^2 + \sigma_\epsilon^2$ to define the predictability as the ratio of the variance of the estimator over the variance of the asset:

$$\nu = \frac{\hat{\sigma}^2}{\sigma^2}\,.$$

The interpretation motivating the use of this ratio is as follows: if the variance of the estimator $\hat{\sigma}^2$ is higher than that of the asset σ^2, then the dependence on the past dominates the noise and thus the predictability is high, while if ν is small then the noise dominates the prediction. It is straightforward to extend this definition to a multivariate model where ν becomes the ratio of the variances of two portfolios. The Box and Tiao (1977) procedure requires us to define a model for the price process; in d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) the authors considered the prices to follow a VAR(1) process:

$$S_t = AS_{t-1} + \epsilon_t\,.$$

where S_t is a vector of prices, A the autoregressive coefficient and $\epsilon_t \sim \mathcal{N}(0, \Sigma)$ the noise with covariance matrix Σ. For the multivariate case, we use a weight vector x_t to form the portfolio $x_t^T S_t$ and obtain the predictability ratio:

$$\nu = \frac{x_t^T A^T \Sigma A x_t}{x_t^T \Sigma x_t}\,.$$

Box and Tiao (1977) showed that minimising the predictability is equivalent to solving the generalised eigenvalue problem, $\det(\nu\Sigma - A^T\Sigma A) = 0$.

d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) highlighted the fact that we only need to find the extremal generalised eigenvalues to rewrite the problem as a minimisation. They used this new formulation as an optimisation problem to introduce a sparse constraint on the weight vector, however the resulting combinatorial problem has been proven to be a NP-hard. We can write this minimisation problem with normalisation and sparse constraint as:

$$\min_{x_t} \frac{x_t^T A^T \Sigma A x_t}{x_t^T \Sigma x_t} \quad \text{subject to:} \qquad \|x\|_2 = 1\,, \tag{5.1}$$

$$\|x\|_0 = k\,. \tag{5.2}$$

In Cuturi and d'Aspremont (2013, 2015) the authors reformulated this optimisation to include a constraint on the minimum variance of the portfolio and proposed the use of two other utility functions, which have fewer constraining assumptions: the Portmanteau statistic, which tests if a process is white noise, and the crossing statistic, which represents the frequency at which a process crosses its mean. In this paper however we will focus solely on the predictability ratio, since its formulation is suited for the H-SGDLM model and the CCD algorithm.

In Sipos and Levendovszky (2013); Fogarasi and Levendovszky (2013) they followed the idea of d'Aspremont (2010) but considered the inverse problem. Indeed, they assumed the portfolio follows an Ornstein-Uhlenbeck process and considered the coefficient representing the mean reversion speed of the portfolio to be directly proportional to the predictability variable, as defined by Box and Tiao (1977). Hence they maximised the predictability instead of minimising it. In a second paper Sipos and Levendovszky (2013) proposed replacing the generalised eigenvalue maximisation problem introduced by d'Aspremont (2010) with a maximisation of the average return. This seems to highlight the fact that they compute the inverse of a mean reverting portfolio and, as explained by d'Aspremont (2010), by maximising the predictability they obtained a momentum portfolio. For this reason we follow the approach of Box and Tiao (1977) and d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) in the rest of this paper.

Since d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) modelled the price process as a VAR(1) process, the authors had to estimate A, the autoregressive coefficient, and Σ, the covariance matrix, in order to use the Box and Tiao (1977) formulation of predictability. Box and Tiao (1977) considered the least squares estimate for A and plugged this into their general eigenvalue problem. On the other hand, d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) added an L_1 penalty to obtain more stable estimates and highlight the dependencies between S_t and S_{t-1}. Hence, they used a likelihood with L_1 penalty to compute the covariance matrix and replaced the least squares estimate of A with a LASSO-based estimate. In Cuturi and d'Aspremont (2013, 2015) the $lag - k$ covariance matrices were computed according to their empirical estimates and the minimisation problem was reformulated with those variables.

5.2.2 Previous techniques used to solve this optimisation

Since the MRP optimisation problem is non-convex and includes a cardinality constraint, the classic minimisation algorithms will not work. In their papers, d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) solve the constraint optimisation problem using semi-definite programming. In particular they relax the problem into a convex semi-definite optimisation programme (SDP), which they then solve with a minimum eigenvalue solver. However, this relaxation only provides a sub-optimal solution. Indeed, while they proved that in special cases the relaxed problem is tight, i.e. gives the exact solution, in general the equivalence is not guaranteed. Nonetheless, when the correspondence is not exact, the solution of this semi definite relaxation can be deflated to produce a good approximation. In order to do so, the authors used a sparse PCA method which takes the SDP solution and converts it into a sparse vector of weights by recovering the leading sparse eigenvectors.

While we do not fully agree on the utility function proposed by Sipos and Levendovszky (2013); Fogarasi and Levendovszky (2013), the methods the authors use to solve the resulting optimisation problem are nonetheless in-

teresting. Indeed, they proposed two techniques: a simulated annealing (SA) approach and a feed forward neural network (FFNN) algorithm. In their paper the authors also compared them with exhaustive search methods on small enough problems, before using them to tackle bigger ones where those greedy approaches are too expensive to run. For the SA method they converted the problem into a combinatorial one by considering only integer values for the weights x. Thus, they run the SA algorithm on a discrete grid whose dimension corresponds to the number of assets. As a stochastic optimisation procedure this is not guaranteed to find the optimal solution, however, and its performance depends heavily on the choice of the initial vector. For the FFNN they first considered the problem in a smaller dimension and computed some optimal portfolio via exhaustive search to create a training dataset. Once the network has been trained with classic back-propagation on this sub-problem they run it on the large one to obtain a weight vector, again with no guarantees of convergence. The results of the FFNN algorithm were also used as an initialisation procedure to select the weight vector for the simulated annealing steps.

These different approaches incorporate the cardinality constraint at different steps of the algorithm to obtain the closest approximation to the optimal solution of this non-convex problem. Unfortunately all of these methods provide only an approximation of the optimal weight vector and are expensive to compute. Furthermore, only the SDP technique can prove an optimal convergence in some special cases. In the coming section we will introduce a novel formulation of the MRP optimisation problem and a procedure to compute the weight vector which is guaranteed to converge and works efficiently even in a large environment.

5.3 New formulation of the MRP with H-SGDLM

5.3.1 Description of the H-SGDLM model

The Bayesian multivariate modelling of the market, H-SGDLM model of Griveau-Billion and Calderhead (2019); Gruber and West (2016a); Corsi

(2004) gives interesting insights on the behaviour of each asset and the rest of the market. The letters of H-SGDLM stands for heterogeneous simultaneous graphical dynamic linear model. This model extended the previously introduced SGDLM model of Gruber and West (2016a) and HAR-RV model of Corsi (2004) to decompose the moves of the time series into different economically-meaningful variables which can be grouped into endogenous and exogenous ones. In addition, it does not assume stationarity of the time series and sequentially updates its parameters with each newly available data point thus producing new signals on the evolution of the different factors. Furthermore, the flexibility of this model makes it straightforward to extend it to fit our particular problem.

The H-SGDLM models each time series independently and then combines them to form a complete multivariate model of the market hence its matrix formulation:

$$\begin{aligned} S_t &\sim \mathcal{N}\left(H_t\mu_t, \Sigma_t\right) , \\ \text{With:} \quad & H_t = \left(I - \Gamma_t\right)^{-1} , \\ & \Omega_t = \Sigma^{-1} = \left(I - \Gamma_t\right)^T \Lambda_t \left(I - \Gamma_t\right) . \end{aligned}$$

Where Λ_t is the diagonal precision matrix with $\lambda_{j,t}$ on its diagonal, Γ_t is the matrix of the exogenous state coefficients, and $\mu_{j,t} = e_{j,t}\phi_{j,t}$ the mean vector. Thus, Ω_t corresponds to the sparse precision matrix and Σ_t the corresponding sparse covariance matrix.

The endogenous group of variables represents the influence of the previous information of this time series while the exogenous one represents the influence of the rest of the market. Hence, the endogenous variables include, among others, the leverage effect and the influence of lower frequencies on the current evolution. On the other hand, the exogenous variables combine an evolving set of assets from the market which have been selected for their influence on the time series. While in the paper Griveau-Billion and Calderhead (2019) they mainly used the H-SGDLM to model and predict the variance of individual time series and the market, in this paper we will apply

it directly on the logarithm of stock prices to obtain the variables needed to solve the mean reversion portfolio optimisation.

For asset j the time series of prices $s_{j,t}$ is modelled as a function of the state variables; both assuming a Normal Inverse-Gamma model. Let us denote by $\theta_{j,t}$ the state vector, $F_{j,t}$ the vector of variables and $G_{j,t}$ the state evolution matrix; the model reads:

$$\begin{aligned} s_{j,t} &= F_{jt}\theta_{j,t} + v_{j,t}\,, \\ \theta_{j,t} &= G_{j,t}\theta_{j,t-1} + \omega_{j,t}\,, \\ \text{with:} \quad & v_{j,t} \sim \mathcal{N}(0, \lambda_{j,t}^{-1})\,, \\ & \omega_{j,t} \sim \mathcal{N}(0, W_{j,t})\,. \end{aligned}$$

Where the variance of the states $W_{j,t}$ and of the observations $\lambda_{j,t}^{-1}$, are both assumed to follow an Inverse-Gamma distribution. The vectors of variables $F_{j,t} = \left(e_{j,t}, r_{sp_t(j),t}\right)$ and states $\theta_{j,t} = (\phi_{j,t}, \gamma_{j,t})$ are composed of two subsets representing the endogenous and exogenous variables and coefficients, respectively. The endogenous variables represent the influence of the past information of asset j on its future behaviour. In order to model the influence of the past prices on the future evolution of the time series we include the past returns and the average of the past prices over three different frequencies: day, week and month. In addition, because we work at the daily frequency we want to include information from higher frequency behaviours. When available, we use the open, high, low and close prices as summaries of the day's movement. More specifically, we use three variables constructed from these prices. Let us denote the different versions of the daily price of asset j at time t by $s_{j,t}^{test}$ where *test* specifies the metric, for example *open* for the price at the open. Thus, the selected variables representing the intra

day moves are:

$$\begin{aligned} r_t^{low} &= log(s_t^{low}) - log(s_{t-1}^{low})\,, \\ CH_t &= \frac{s_t^{high} - s_t^{close}}{s_t^{high} - s_t^{low}} - 0.5\,, \\ COHL_t &= \frac{s_t^{close} - s_t^{open}}{s_t^{high} - s_t^{low}}\,. \end{aligned}$$

where r represents the log-return. In addition to those variables we will include the leverage effect representing the asymmetric influence of previous positive or negative daily, weekly and monthly return on the future move. Thus the complete vector of endogenous variable is:

$$\begin{aligned} e_{j,t} = (&1, as_{j,t}^d, as_{j,t}^w, as_{j,t}^m, \\ &r_{j,t}^d, r_{j,t}^w, r_{j,t}^m, \\ &r_{j,t}^{low}, CH_{j,t}, COHL_{j,t}, \\ &r_{j,t}^{(d)+}, r_{j,t}^{(d)-}, r_{j,t}^{(w)+}, r_{j,t}^{(w)-}, r_{j,t}^{(m)+}, r_{j,t}^{(m)-})^T\,. \end{aligned}$$

Where for each variable the exponent represents the different frequencies: daily, weekly and monthly; as^f represents the averaged price over the frequency f; r^f is the past log-return at frequency f. Hence the variables of the endogenous vector $e_{j,t}$ represent different economic factors influencing the next price movement.

The exogenous variables are composed of a selected group, $sp_t(j)$, of assets which move simultaneously to stock j. This part is composed of two steps. The first step proposes candidates for inclusion in $sp_t(j)$, which are then included for a certain time period in the vector of variables. After this time period we look at their signal to noise ratio to decide whether to include it in the group of parents $sp_t(j)$ or not. Once the parents with the highest influence are selected, they are included in the vector of exogenous variables $r_{sp_t(j),t}$. The coefficients $\theta_{j,t}$ of the variables $F_{j,t}$ are then learned sequentially with each new data point following the classic state-space update.

5.3.2 Sparsity with the H-SGDLM

As explained by d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) sparsity is an important factor to take into account when building mean reverting portfolios. The previously introduced H-SGDLM model constructs a sparse representation of the whole market. Indeed, instead of the classic covariance matrix the cross series relationships are modelled by the exogenous variables; in each individual DLM the exogenous variables inform the mean of the distribution and are sequentially selected and updated for each asset. When combining these individual DLMs to obtain a multivariate distribution, the exogenous coefficients inform both the mean and variance of the multivariate normal through the matrix Γ. As described previously, the selection of these variables follows a two-step process: first filtering with the Wishart covariance matrix, then testing for its capacity to explain the evolution of the current price given its signal to noise ratio. The number of parents allowed for each DLM is a parameter to be chosen; the smaller this number the greater the sparsity of the resulting multivariate distribution.

Thanks to this two steps process, the cross-series relationships are modelled as asymmetric. While some stocks might not be present in any core-group other might be in many. In other words, some assets influence the behaviour of many others in the market while others are mostly followers. These cross-series relationships are represented in the sparse covariance matrix Γ. Because the number of allowed parents is chosen small, the resulting covariance matrix Σ is sparse. And, thanks to its construction the non-zeros elements represent stocks with simultaneous behaviours. Thus, any mean reverting portfolio should be constructed within one of these subsets build from the stocks with the highest non-zero coefficient in the sparse covariance matrix Γ. Hence, any addition or removal of asset must be done directly in the environmnet used to learn the HSGDLM model.

5.3.3 MRP reformulated with H-SGDLM data

We follow the approach of d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) to minimise the predictability as defined in optimisation problem 5.1, but instead of using the ratio we will work with the difference:

$$\min_{x_t} \; x_t^T A^T \Sigma A x_t - x_t^T \Sigma x_t \quad \text{subject to:} \qquad \begin{aligned} \|x\|_1 &= 1\,, \\ \|x\|_0 &= k\,. \end{aligned}$$

While d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) modelled the price process by a VAR(1) model (i.e. with stationary autoregressive coefficient A and stationary covariance matrix Σ), here we use the outputs from the H-SGDLM thereby relaxing this condition of stationarity. By considering the price process s_t to follow a time varying autoregressive process TVAR(1) we can express the portfolio by: $P_t = x_t^T S_t = x_t^T A_t S_{t-1} + x_t^T \epsilon_t$, where the prediction of S_t assuming all the data up to $t-1$ known is denoted by $\hat{S}_t = A_t S_{t-1}$. We recall that the motivation for the Box and Tiao (1977) predictability ratio was to compare the variance of the prediction to the measured one and see if the prediction dominates the noise, implying that the process is predictable. Thus the ratio of the predicted portfolio variance over the measured one is:

$$\nu_t = \frac{\hat{\sigma}_t^2}{\sigma_t^2} = \frac{\mathbb{E}[x_t^T A_t S_{t-1} S_{t-1}^T A_t^T x_t | D_t]}{\mathbb{E}[x_t^T S_t S_t^T x_t | D_t]} = \frac{x_t^T A_t \tilde{\Sigma}_{t-1} A_t^T x_t}{x_t^T \tilde{\Sigma}_t x_t}\,.$$

Where $\tilde{\Sigma}_t$ corresponds to the empirical covariance matrix measured with prices up to t. Therefore, this ratio corresponds to the the predicted covariance matrix using the autoregressive coefficient A_t over the measured portfolio variance at t. But since our model can actually infer the next covariance matrix Σ_t we could use instead:

$$\nu_t = \frac{x_t^T \Sigma_t x_t}{x_t^T \tilde{\Sigma}_t x_t}\,.$$

The H-SGDLM model gives us the autoregressive coefficient A_t, and sparse predicted covariance matrix Σ_t. Therefore the predictability ratio could be

computed in different ways using either one of these outputs. Since for the optimisation problem we work with the difference instead of the ratio, possible utility functions include:

$$\begin{aligned} U_1(x) &= x^T \left(A_t \tilde{\Sigma}_{t-1} A_t^T - \tilde{\Sigma}_t \right) x \,, \\ U_2(x) &= x^T \left(\Sigma_t - \tilde{\Sigma}_t \right) x \,, \\ U_3(x) &= x^T \left(A_t \tilde{\Sigma}_{t-1} A_t^T - \Sigma_t \right) x \,. \end{aligned}$$

Let us detail the different options. U_1 uses the measured covariance matrices at $t-1$ and t and computed autoregressive coefficient A_t. U_2 compares the sparse covariance matrix at t inferred with data up to $t-1$ to the measured covariance matrix using prices up to t. U_3 compares a prediction of the variance using the covariance matrix measured with data up to $t-1$ and the computed autoregressive coefficients to the predicted covariance matrix Σ_t. Now since we assumed all the data up to t is known for computing the predictability ratio at t, we could do the same computation as U_3 but using the data available at t, i.e. compare the $t+1$ predictions. However, this option (as with U_3) does not respect the predictability as defined by Box and Tiao (1977) since it would compare two predictions instead of a predicted versus a measured value; for this reason we work with U_1 and U_2. With this set-up we can either use the predicted covariance matrix or the predicted autoregressive coefficient. Let us denote the minimisation of the covariance differences as problem PC and the other one as problem PA. Further, let us denote the utility function by $x^T \left(D_t - \tilde{\Sigma}_t \right) x$, where D_t is the predicted covariance matrix. Then, for PC we have $D_t = \Sigma_t$, while for PA it is $D_t = A_t \tilde{\Sigma}_{t-1} A_t^T$.

5.4 Solving the minimisation problem with CCD

5.4.1 The CCD algorithm for non-convex function

Coordinate descent algorithms work by optimising one direction at a time instead of all of them simultaneously as classical descent algorithms do. The

main drawback for this type of algorithm is their requirement for the utility function to be strictly convex and differentiable. Tseng (2001) proved that the convergence can be extended to a specific type of non-convex and non-differentiable function. In particular they considered a non-convex objective function that can be decomposed into a quasi-convex and differentiable function f_0 and a non-differentiable but additively separable one f_k:

$$f(w) = f_0(w) + \sum_{k=1}^{m} f_k(w_k)\,.$$

If a function can be written in such a way, then a block coordinate descent algorithm applied on this function will converge to the global minimum of f with respect to w. More specifically, the convergence is guaranteed according to theorem 5.1 of Tseng (2001) if the functions respect the following conditions: f_0 must be continuous, f is quasi-convex and hemivariate in each coordinate block and the functions f_k are left side continuous $\forall k$. In addition, Tseng (2001) made assumptions on the domain of f_0, for which we refer the reader to the original paper.

The coordinate descent algorithm works by iteratively updating the weight vector w^r. At each step it updates the value of the weights w_i^r for all $i \in [1, N]$ by minimising f assuming all the other weights $w_{j\neq i}^r$ fixed: $w_i^r = min_{w_i^r} f(w^r)$. Then, it minimises the next weights using the updated values of the other weights. Once the entire vector w^r has been updated the iteration starts again until convergence: $\|w^r - w^{r-1}\| \leq \epsilon$.

Subsequently, Yen and Yen (2014) used this algorithm to solve the minimum variance portfolio optimisation with elastic net constraints since it naturally fits in the formulation of Tseng (2001). The Lagrangian of this problem is:

$$L\left(W, \gamma, \Sigma, \beta, \alpha\right) = W^T \Sigma W + \beta\left(\alpha \|W\|_{l_1} + (1-\alpha)\|W\|_{l_2}^2\right) - \gamma\left(W^T 1 - 1\right),$$

where β is the penalty parameter, α the weight balancing the importance of the L_1 and L_2 terms, γ the Lagrange multiplier and W the weights vector. Fixing γ, we can solve the KKT conditions with respect to the weight w_i to

obtain:

$$\begin{aligned} w_i &= \frac{ST\left(\gamma - z_i, \beta\alpha\right)}{2\left(\sigma_i^2 + \beta\left(1-\alpha\right)\right)}, \\ z_i &= 2\sum_{j\neq i}^{N} w_j \sigma_{i,j}\,, \end{aligned}$$

where $\sigma_{i,j}$ corresponds to the covariance between weights i and j, and $ST(x,y) = sign(x)max(|x|-y,0)$ is the soft-threshold function. The CCD algorithm recursively updates each weight w_i with this equation until the vector w converges. The following section will detail the application of the CCD algorithm to solve the sparse mean reverting portfolio optimisation problem.

5.4.2 The CCD algorithm applied to sparse mean reverting portfolios

Instead of the cadinality constraint used by d'Aspremont (2010); Cuturi and d'Aspremont (2013, 2015) to fix the number of non-zeros weights to a pre-selected value, we use the graphical structure learned by the H-SGDLM model to select a group of assets of size k. While in this paper we use the predicted covariance matrix D_t obtained from the H-SGDLM model, the CCD algorithm we propose in this section could be applied using any other covariance matrix prediction, however we note that without the H-SGDLM sparsity selection, an L_1 regularisation coefficient would be needed to impose sparsity. Since we work with the difference instead of the ratio we add a L_2 regularisation factor to the utility function. The minimisation problem becomes:

$$\min_{x_t} \; x_t^T \left(D_t - \tilde{\Sigma}_t\right) x_t + \beta \|x_t\|_2^2\,, \quad \text{subject to:} \quad \|x_t\|_1 = 1\,,$$

where β is the weight of the L_2 norm. With this formulation only the normalisation constraint remains. We use the L_1 norm of the weight as the normalisation constraint because we want to take into account the full exposure of the portfolio. Hence, the normalisation takes into account the short positions in the total weight. In addition to its smoothness property, the

L_2 regularisation term here guarantees the positive semi-definiteness of the problem. Indeed, although D_t and Σ_t are positive semi-definite covariance matrices, their difference is not guaranteed to be. We circumvent this issue with the addition of an L_2 weight, which acts as a matrix regularisation coefficient to keep the difference within the positive semi-definite domain.

This approach is not perfect however, since the L_2 constraint adds a bias to the solution. Therefore we want to select the smallest possible coefficient β that guarantees the positive semi-definiteness of the matrix $M = D_t - \tilde{\Sigma}_t + \beta I$, where I corresponds to the identity matrix. We select a value $\beta = \beta_0$ and at each time t compute the optimal solution, then we check for the positivity of the resulting utility function. In case of negativity, we increase the value of β by 10^{-5} until the function becomes positive, but only for a maximum of 10 steps. If the maximum number of iterations is reached we increase the step-size to 10^{-1} and start again. This incremental approach is computationally feasible due to the speed of the CCD algorithm, although we have to take care with choice of β_0; if it is chosen too small it will end up in the loop step too often, while if chosen too big it adds an unnecessary bias. In addition, if the value of the MRP part becomes negligible compared to the value of the L_2 constraint then the optimisation algorithm will just minimise the L_2 norm of the weight and thus produce the equal weighted portfolio. We detail in section 7.4 a pragmatic selection of values for the problems considered.

For the sake of clarity, since we consider all the data up to t known and assume all the variables in the optimisation problem are taken at time t, we drop the subscript t in the following expressions. The Lagrangian of our minimisation problem (5.4.2) follows as:

$$\mathcal{L}(x, \gamma) = x^T \left(D - \tilde{\Sigma} + \beta I \right) x + \gamma \left(\|x\|_1 - 1 \right) , \tag{5.3}$$

This function satisfies the convergence requirements defined by Tseng (2001) since we can write the problem as the combination of a differentiable, quasi-convex term and a non-differentiable but additively separable one. Thus, we can follow a coordinate descent to find the global optimum. Without loss

of generality we will fix the Lagrange coefficient to zero and renormalise the weight vector at the end of each loop. Let us denote the elements of matrix D by $d_{i,j}$ and of Σ by $\sigma_{i,j}$. Then, the line i of the matrix $\left(D - \tilde{\Sigma} + \beta I\right) x$ can be written $\left(D - \tilde{\Sigma} + \beta I\right) x|_i = \left(d_i^2 - \sigma_i^2 + \beta\right) x_i + \Sigma_{j \neq i} \left(d_{i,j} - \sigma_{i,j}\right) x_j$. Where $\sigma_i^2 = \sigma_{i,i}$ and $d_i^2 = d_{i,i}$ are the diagonal elements of D and Σ respectively. Thus, the gradient of the Lagrangian in Equation 5.3 with respect to x_i is:

$$\nabla_{x_i} \mathcal{L}(x, \gamma) = 2\left(d_i^2 - \sigma_i^2 + \beta\right) x_i + 2\Sigma_{j \neq i} \left(d_{i,j} - \sigma_{i,j}\right) x_j \,,$$

Therefore, for the gradient to be zero each weight x_i must follow:

$$x_i = -\frac{2\Sigma_{j \neq i} \left(d_{i,j} - \sigma_{i,j}\right) x_j}{2\left(d_i^2 - \sigma_i^2 + \beta\right)} \,. \tag{5.4}$$

The coordinate descent algorithm iteratively updates the weights with equation 5.4 until convergence. While we did not include an L_1 norm on the weights because we enforce sparsity directly using the results from the H-SGDLM model, it is straightforward to include one. Indeed, with an L_1 regularisation term and coefficient λ_1, the numerator of Equation 5.4 simply becomes ST $\left(2\Sigma_{j \neq i} \left(d_{i,j} - \sigma_{i,j}\right) x_j, \lambda_1\right)$, where $ST(.,.)$ is the soft-thresholding function.

5.5 Trading a Mean Reverting Portfolio

5.5.1 Jurek and Yang trading strategy

In the papers Cuturi and d'Aspremont (2013, 2015) used the Jurek and Yang (2007) strategy to trade the mean reversion of the obtained MRP portfolios. This strategy considers the portfolio to follow a stationary VAR(1) process $P_t = \rho P_{t-1} + \sigma \epsilon_t$, with $|\rho| < 1$ and mean μ, where the parameters were obtained using a least squares estimate. Let us denote by w_t the investment weight of portfolio P_t, at t, which the authors define by $w_t = (\mu - p_t)\rho/\sigma^2$. In their paper from 2013 the authors considered trading swaps and concluded that the gains are too small to create a profitable strategy in practice, since the transaction costs cancelled all arbitrage opportunities. On the

other hand, in the 2015 paper they built baskets of options to trade implied volatility and with added constraints they obtained performances robust to costs.

5.5.2 Ornstein-Ulhenbeck based trading strategy

In their papers, Fogarasi and Levendovszky (2013); Sipos and Levendovszky (2013) assumed the portfolio to follow an Ornstein-Ulhenbeck process $P_t \sim \mathcal{N}\left(\mu, \sqrt{\frac{\sigma^2}{2\rho}}\right)$ and considered three scenarios. Either the portfolio is in a stationary state and no investment is made, or it is outside a threshold and a position is entered. The authors define a threshold ϵ such that:

- H1: $P\left(p_t < \mu - \alpha\right) = \epsilon/2$, the portfolio's value is bellow its long term mean and they buy it.
- H2: $P\left(p_t \in [\mu - \alpha, \mu + \alpha]\right) = \epsilon$, the portfolio is in its stationary state and thus they stay neutral.
- H3: $P\left(p_t > \mu + \alpha\right) = \epsilon/2$, the portfolio's value is above its long term mean and they sell it.

Once a position is entered, they keep it until the portfolio comes back to its stationary state. In this strategy based on O-U thresholds the maximum amount is always invested, in contrast to the original strategy of Jurek and Yang (2007) where the portfolio's weight can take any real number.

5.5.3 Mixed mean reversion trading strategy

We could also use a combination of those two ideas. Instead of always investing an amount corresponding to the distance from the mean (as in Jurek and Yang (2007)), we could instead adopt this approach only when the move is larger than a threshold, resulting in the strategy:

- H1: $W_t = \frac{\rho}{\sigma^2}\left((\mu - \alpha) - P_t\right)$,

- H2: $W_t = 0$,
- H3: $W_t = \frac{\rho}{\sigma 2}\left((\mu + \alpha) - P_t\right)$,

For the rest of this paper we will refer to this strategy as the mixed $JY - OU$ strategy.

5.6 Results

5.6.1 US stocks from the S&P500

In order to test our algorithm on stocks, we selected the 371 stocks from the S&P500 with data available since 2001. This dataset consists of the OHLCV data for each stock at the end of each trading day. However, instead of using daily time points we worked on weekly time points, i.e. we selected one data every 5 time points. Hence, the one-step ahead inference on which the H-SGDLM learns corresponds to a one week ahead forecast. Since the H-SGDLM takes some time to converge we study the results between 2003 and 2018. The vector of variables includes the OHLC data as described in section 5.3.1. The size of the core parent group was fixed to 10 with an update every 10 time steps and due to the memory limit of the GPU card the number of Monte-Carlo samples was fixed to 500.

For the parametrisation step we selected half the dataset, from 2011 to 2018. We used the second half of the dataset to avoid fitting the parameters over the 2008 crisis, as we wanted our out-of-sample to test the robustness of the parameters over this crisis period. The MRP algorithm has two parameters, β the L_2 weight and λ the coefficient of the exponential weighting we use to compute the empirical covariance matrix $\tilde{\Sigma}$. Since we use two different optimisation problems, PA and PC, we allow the parameters to be different for each one. To find the acceptable range of values for those parameters we compared the computed MRP portfolios to the equally weighted one. Once this range was found, we selected the value that created the most mean-reverting portfolio over this time window. As an example figure 5.1 shows the resulting

portfolio obtained over the whole dataset with $\beta = 10^{-5}$ and $\lambda = 0.9$. Since we want a sparse portfolio, we fixed the size of the portfolio to 50 different assets following the methodology described in 5.3.2 that uses the matrix of parents' coefficients Γ. We then selected $\beta = \{10^{-3}, 10^{-4}, 10^{-5}, 10^{6}\}$, $\lambda_{PA} = 0.98$ and $\lambda_{PC} = 0.85$. For the trading part we have to select the length, T_{tr} of

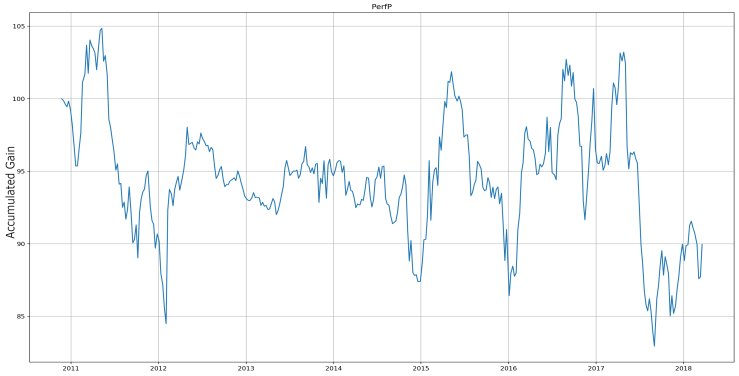


Figure 5.1: This figure shows the performance of the MRP portfolio computed for 371 stocks from the S&P500. On this figure we can observe the mean reversion of the obtained portfolio.

the time window on which the different trading parameters will be computed at each time t. We looked at the performances of the backtests and chose $T_{tr} = 50$. We then compared the different trading strategies described in 5.5. For each strategy we averaged the results obtained for the different β values and the different PA and PC problems. The figures 5.2 and 5.3 show the results for the Jurek and Yang (2007) and O-U based strategies respectively. Interestingly on this dataset the mixed $JY - OU$ one had worse performance then the other two. Even though the backtests do not include any costs it is interesting to observe the steady performance even during the 2008 crisis.

5.6.2 FX Futures

In this section we test our algorithm on exchange rates data with a dataset consisting of 22 different FX-futures between 2006 and 2018. For this environment, the H-SGDLM used the average of the past 40 daily-, weekly-, monthly-log-returns. This average used an exponential weighting with a co-

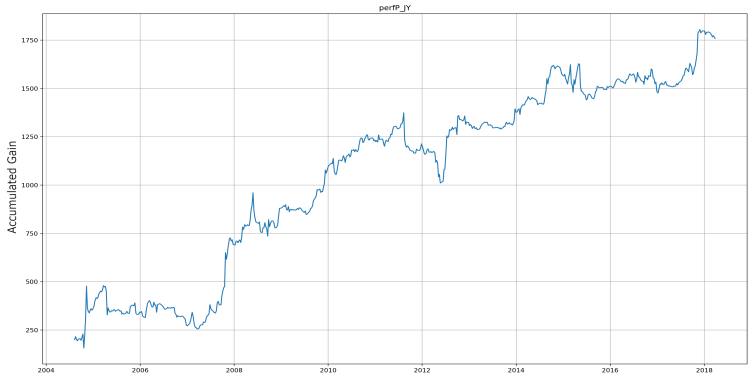


Figure 5.2: This plot shows the performance of the Jurek and Yang (2007) trading strategy applied on the MRP portfolio computed on an evolving set of 50 stocks from the S&P500.

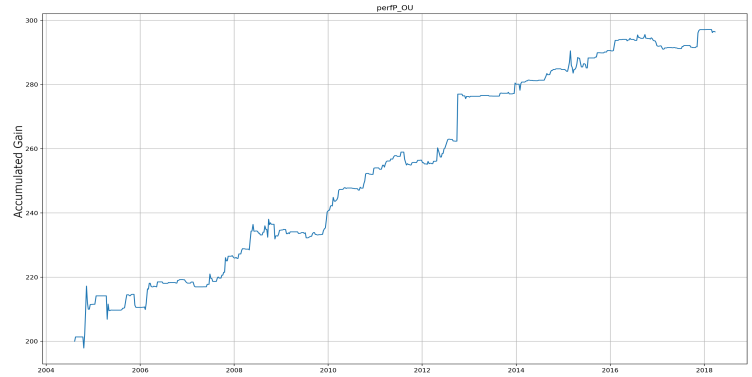


Figure 5.3: This plot shows the performance of the O-U based trading strategy applied on the MRP portfolio computed on an evolving set of 50 stocks from the S&P500.

efficient of 0.98. The size of the parent group was fixed at 5 and the updates happened every 10 time steps. Just like in the previous example, we used a weekly time point dataset, such that the parent selection corresponded to an update every 50 days. Furthermore, we used 2000 Monte-Carlo samples.

We followed the same methodology to find the parameters as the one described for S&P500 data. Hence, the parametrisation step used half the dataset, from 2012 to 2018 and the number of asset of the portfolio, or sparsity constraint, is fixed to 10. The exponential weighting coefficient is chosen to be the same for both problems PA and PC, i.e. $\lambda_{PA} = \lambda_{PC} = 0.8$. But, the L_2 coefficients are different: $\beta_{PA} = \{10^{-1}, 10^{-2}, 10^{-3}\}$ and $\beta_{PC} =$

$\{10^{-2}, 10^{-4}, 10^{-5}\}$. For the trading strategies the time window was fixed to $T_{tr} = 10$ time points equal to 50 trading days. Figure 5.4 shows the backtest performance using the Jurek and Yang (2007) trading strategy while figure 5.5 corresponds to the backtest of the mixed $JY - OU$ strategy. As with the previous backtest on stocks it is interesting to observe the algorithm's steady performance. In this case however, contrary to the backtest on stocks, the mixed $JY - OU$ strategy performed slightly better than the O-U based one, hence we only show the performance of the mixed one.

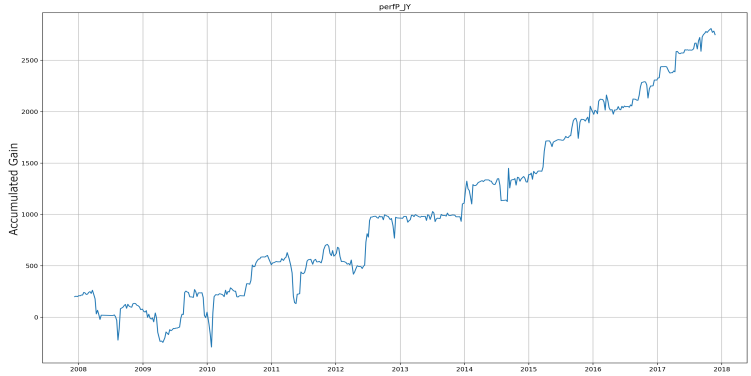


Figure 5.4: This plot shows the performance of the Jurek and Yang (2007) trading strategy applied on the MRP portfolio computed on an evolving set of 10 FX-futures.

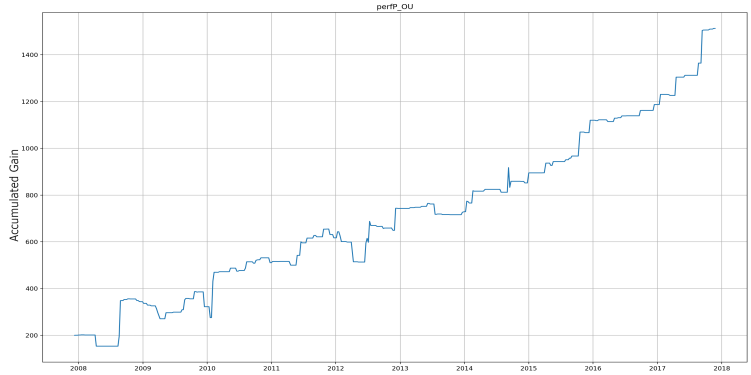


Figure 5.5: This plot shows the performance of the mixed $JY - OU$ trading strategy applied on the MRP portfolio computed on an evolving set of 10 FX-futures.

5.6.3 ETF Futures

Finally we considered the dataset of futures on ETFs, in particular 75 different futures with OHLCV data from 2008 to 2018. Hence, the H-SGLDM learns on log-prices with the variables described in section 5.3.1. As for the previous datasets we used weekly time points instead of daily ones. The size of the core parent group is fixed to 10 with an update every 10 time steps and we computed 2000 Monte-Carlo samples.

We followed the same methodology as before to find the MRP parameters. The sparsity variable was fixed to 20 futures, while the time window used for parametrisation was 2013 to 2018. We selected exponential weighting coefficients of $\lambda_{PA} = 0.85$ and $\lambda_{PC} = 0.98$, and L_2 coefficients: $\beta_{PA} = \{10^{-5}, 10^{-6}\}$ and $\beta_{PC} = \{10^{-3}, 10^{-4}\}$. As with FX-futures the time window for the trading strategies was fixed to $T_{tr} = 10$ time points. As for the dataset on stocks, the mixed $JY - OU$ strategy performed worse than the other two. Figure 5.6 shows the backtest of the Jurek and Yang (2007) trading strategy and figure 5.7 corresponds to the performance of the O-U based approach. While these backtests are more volatile than the those on FX-futures and US-stocks they still have a relatively steady growth highlighting the robustness of this portfolio construction.

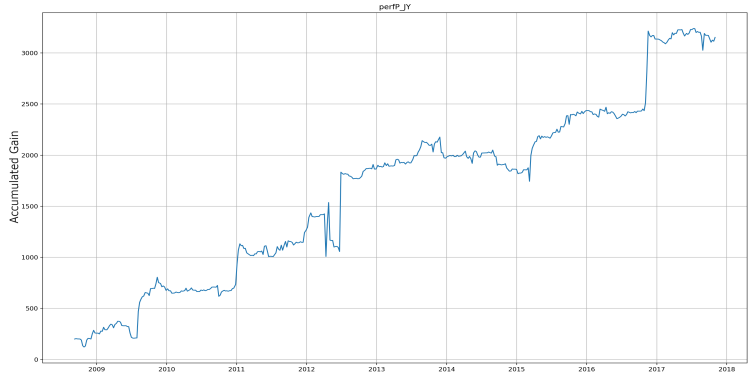


Figure 5.6: This plot shows the performance of the Jurek and Yang (2007) trading strategy applied on the MRP portfolio computed on an evolving set of 20 ETF-futures.

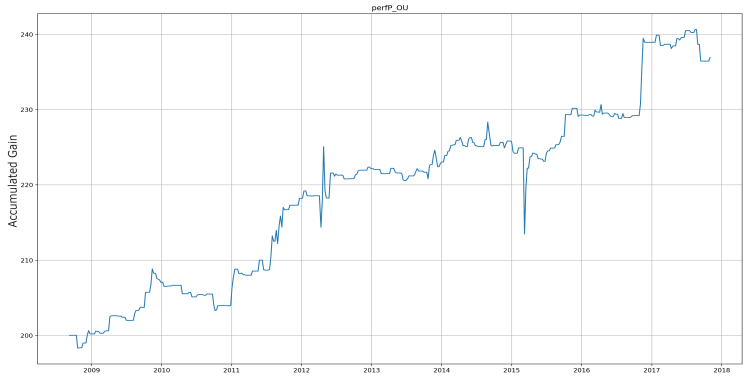


Figure 5.7: This plot shows the performance of the O-U based trading strategy applied on the MRP portfolio computed on an evolving set of 20 ETF-futures.

5.7 Conclusion

In this paper we presented a new approach for computing a sparse mean reverting portfolio within a large environment. The goal was to find an efficient procedure that is able to find a sparse subset of series from a large environment of time series, for which we could compute a linear combination exhibiting the mean-reverting property. Furthermore, the solution had to work on the more complicated problem of subsets larger than two assets. Given such a mean-reverting portfolio the performance of our solution was numerically evaluated by performing backtests in different investment environments.

The solution we proposed in this paper leveraged the data obtained by the newly introduced H-SGDLM model. Indeed, we were able to reformulate the MRP problem as a quasi-convex optimisation problem with a normalisation constraint. In addition, the sparsity constraint was moved away from the minimisation formulation and instead we used the matrix of core parents from the H-SGDLM to select the subset of time series on which to compute the MRP. Using different variables from the H-SGDLM model we obtained two different formulations of the problem, PA and PC, using the matrix of the regression coefficients and the predicted sparse covariance matrix, respectively. This new formulation is perfectly suited for a cyclical coordinate

descent (CCD) algorithm. Hence, while the existance of a MRP portfolio isn't garanteed, once the data from the H-SGDLM is available the CCD optimisation step finds the global solution of the MRP minimisation problem.

As a result of this formulation it is possible to compute the optimal MRP at each time t even on environments involving hundreds of time series. We then evaluated different trading strategies to take advantage of this mean-reversion. We performed backtests on three different datasets: stocks from the S&P500, FX-futures and ETF-futures. All the resulting backtests exhibited steady performances through time even on the out-of-sample part of the backtest that included the financial crisis, suggesting a certain robustness of our proposed solution.

6

Causal Graph Process

6.1 Introduction

Graph structures can be represented with an adjacency matrix containing the edge weights between different nodes. Since the adjacency matrix is usually sparse, common estimation techniques involve a quadratic optimization with a LASSO penalty to impose this sparsity. While the coefficient of the L_1 regularization term is easy to choose when we have prior knowledge of the appropriate sparsity level, in most cases we do not and the parametrization becomes non-trivial. In this paper we focus on causal relationships between time series, following the causal graph process (CGP) approach (Mei and Moura, 2015, 2017) which assumes an autoregressive model, where the coefficients of each time lag is a polynomial function of the adjacency matrix. Using the adjacency matrix instead of the graph Laplacian allows this set-up to consider directed graphs with positive and negative edge weights. Mei and Moura (2017) simplify the problem to a combination of quadratic minimisations with L_1 regularization terms which they then solve with a sparse gradient projection algorithm. They choose this algorithm for its use of sparse-matrix-computation libraries, which is efficient only for highly sparse

graphs; its performance deteriorates significantly with more dense graphs.

The cyclical coordinate descent algorithm is a widely used optimisation method due to its speed. Its popularity increased after Tseng (2001) proved its convergence for functions that decompose into a convex part and a non-differentiable but separable term, which makes it perfectly suited for solving quadratic minimizations involving an L_1 regularization term. In Wu and Lange (2008) the efficiency of both a cyclical and a greedy coordinate descent algorithm to solve the LASSO was shown. Subsequently, Wang (2013) demonstrated that the cyclical version is more stable and applied it to solve the graphical LASSO problem. Meinshausen and Bühlmann (2006) extended the LASSO to high-dimensional graphs by performing individual LASSO regularization on each node. This approach inspired Friedman et al. (2008) to create the graphical-LASSO with the block coordinate descent algorithm. This algorithm computes a sparse estimate of the precision matrix assuming that the data follows a multivariate Normal distribution. The use of CCD to solve this graph-LASSO problem allows for efficiently computing the sparse precision matrix in large environments. However, this algorithm focuses on simultaneous connections not causal relationships. In contrast, Shojaie and Michailidis (2010a,b) proposed different approaches to compute what they define as a graph-Granger causality effect, however their models assumed prior knowledge of the structure of the underlying adjacency matrix and therefore focused on estimating the weights.

In the literature there are many proposed methods for selecting the LASSO coefficient. Banerjee et al. (2007); Meinshausen and Bühlmann (2006); Shojaie and Michailidis (2010a,b) use a function of the probability of error, which depends on selection of a probability. Wu and Lange (2008); Mei and Moura (2015, 2017) perform an expensive grid search and use the in- and out-of-sample error to select the coefficient. Friedman et al. (2008); Wang (2013) avoid the problem by simply fixing the coefficient to a selected value. These strategies either require the selection of free parameter or rely on the prediction error to find the correct LASSO coefficient. In this work we follow the idea of the graph-LASSO of Friedman et al. (2008) and the individual

LASSO regularization of Meinshausen and Bühlmann (2006) to propose a new CCD algorithm that solves the causal graph process problem of Mei and Moura (2015, 2017).

The CCD approach allows us to leverage the knowledge of the specific structure of the adjacency matrix to optimise the computational steps. In addition, we propose a new metric that uses the prediction quality of each node separately to select the LASSO coefficient. Our solution does not require additional parameters and produces better results than relying solely on the prediction error. Thus, our algorithm computes the directed adjacency matrix with an appropriate number of edges, as well as the polynomial coefficients of the CGP. Furthermore, the quality of the results and speed are not affected by the sparsity level of the underlying problem. Indeed, while the algorithm proposed by Mei and Moura (2015, 2017) scales cubically with the number of nodes, the algorithm we propose scales quadratically while automatically selecting the LASSO coefficient. We show the performance of our solution on simulated CGPs following a stochastic block model with different sizes, levels of sparsity and history lengths. We assess the quality of the estimated adjacency matrix by considering the difference in number of edges, the percentage of correct positives and the percentage of false positives. We highlight the performance of our approach and its limits. We then run the algorithm on a financial dataset and interpret the results.

Section 6.2 introduces the Granger causality principles as well as different approach of structure learning and define the signal processing on graphs model that we will use. We then introduce our novel algorithm based on a coordinate descent algorithm to efficiently estimate the adjacency matrix of the causal graph process in Section 6.3. In Section 6.4 we present a new non-parametric metric to automatically select the LASSO sparsity coefficient. Finally, in Section 6.5 we present the results on simulated and real datasets.

6.2 Prerequisite on graph and causality

6.2.1 Granger causality

In this section we summarise the chapter on Granger causality from Peters et al. (2017) book. Causality in time series is specific since different variables can refer to the same time point and we usually have only one sample of the time series. The two main types of relations are instantaneous and lagged. Let us consider a bivariate environment, in the instantaneous case a variable X_t^1 will influence the other variable at the same type point X_t^2. In the lagged case, the past values of the variable $X_{s<t}^1$ have a causal effect on the current value the other variable X_t^2. When studying the causal links between variables we assume that all the relevant variables are observed and included in the model. This is called the causal sufficiency hypothesis. This hypothesis is equivalent to assuming the absence of hidden cofounders. A cofounder is a variable Z that causes two other variables X and Y. If Z is hidden the model will consider X and Y to be connected. This problem becomes even more important in time series where a cofounder Z can have an impact on two variables, X and Y, with a different time lag, at t for X and $t+1$ for Y. In this case, if the cofounder is hidden we could falsely conclude that X Granger-causes Y with a lag 1.

Since a full time graph will have as many nodes as time points for each variable we usually represent the causal relations with a summary graph. In a summary graph, each node represents a variable and an arrow between the nodes a causal relation instantaneous or lagged. A popular model to represent causal relation between time series is the Vector Autoregressive (VAR) model $X_t^j = \sum_{i=0}^{l} A_t^j X_{t-i} + \omega_t^j$, where ω is a noise term.

Granger causality is the most popular method to analyse causality in time series. For two variables X and Y, X is said to Granger-cause Y if the past values of X improve the prediction of Y from its own past values. In other words, the past values of X bring informations about Y that were not contained in its own history. When considering linear prediction it is

equivalent to comparing the two models:

$$Y_t = \sum_{i=0}^{l} A_t Y_{t-i} + \omega_t \ , \tag{6.1}$$

$$Y_t = \sum_{i=0}^{l} A_t Y_{t-i} + \sum_{i=0}^{l} B_t X_{t-i} + \hat{\omega}_t \ . \tag{6.2}$$

Where ω and $\hat{\omega}$ are noise terms. Then, X is said to Granger-cause Y it the model of Equation 6.2 produces better predictions than the model of Equation 6.1. In other terms, the variance of the noise $\hat{\omega}$ is lower than the variance of ω. These definitions are straightforward to extend to the multivariate case. However, in the multivariate case the causal sufficiency hypothesis is unrealistic in practice. Hence, the importance of including all possible variables in order to avoid a hidden cofounders that would lead to false Granger-Causal links.

6.2.2 Structure learning

The utility of a graph model between variables is to indicate which ones do not interact directly because the knowledge of having two variables independent of each others simplifies the problem. Having a graph modelling the interactions between the different variables creates a model with fewer parameters and thus easier to train and study. However, contrary to a directed graph sampling from an undirected graph is a complex and expensive problem. There are different ways to build graphs that connect non-independent variables with directed edges. The first step consists into determining if two variables are connected to each other, i.e. are non-independent. The second, consists in determining the direction of that edge. While some algorithm such as the one we will present later in this chapter merge these two steps, it is interesting to understand the complexity of each of them.

Sparse projection vs. sparse construction

A graph skeleton consists of an undirected graph with edges between non-independent variables. While in some cases the dataset directly indicates the edges, e.g. in social media where an edge could be the number of common friends between two profiles; in most cases this skeleton has to be learned. Two approaches are possible: either starting from a fully connected graph and then deleting edges or starting from a sparse graph and adding the edges.

When starting from a fully connected graph, the greedy solution would be to perform a greedy path assessing the quality of each edge. However that quickly becomes too exhaustive even for graphs with small sizes. A more common approach consists in using a regularisation parameter and performing a sparse projection. For example, the Graph-LASSO algorithm of Friedman et al. (2008) starts from a dense covariance matrix and regularise it to obtain a sparse matrix.

On the other hand, finding edges in a set of non-connected nodes requires an exploration strategy. While as before the greedy approach is theoretically doable it is too expensive for any real world scenario. We could think about implementing a Bayesian optimisation or stochastic-search algorithm but they would both be very expensive to run and only obtain approximations. In this chapter we present a strategy to build the graph with a LASSO regularisation on each node, hence building the graph by looping over all the nodes. But, contrary to the greedy or stochastic search approaches we leverage the structure of the matrices to optimise the computations.

Skeleton vs. direction

Let us assume we dispose of a graph skeleton $\mathcal{G}$. An edge between two nodes signifies that they are not independent. The next step is to learn the causal relation between the two. The common approaches rely on hypothesis testing or scoring. For example in the paper from Facebook AI reseacrh, Goudet et al. (2018), they assume a known skeleton graph and learn the

causal structure with a scoring method. The score measure the distance between the generated distribution created by a graph having that edge and the observed distribution. They loop through each edge and compute a score for each possible orientation, the higher score indicates the direction of the edge. Even though they focus on directed acyclic graph (DAG), constructing the directed graph directly with a greedy exploration of possible DAG would be super-exponential in the number of nodes. However, if a skeleton is known, a greedy procedure to select the direction of the edges becomes tractable.

6.2.3 Signal processing on graphs

There exist many approaches for modelling a graph; in this paper we use a directed adjacency matrix A. An element $A_{i,j} \in \mathbb{R}$ of the adjacency matrix corresponds to the weight of the edge from node j to node i. We consider a time dependent graph with N nodes evolving over K time samples. Let $x(k) \in \mathbb{R}^N$ be the vector with the value of each node at time k. With this formulation, the graph signal over K time samples is denoted by the matrix $X(K) = [x(0) \dots x(K-1)] \in \mathbb{R}^{N \times K}$. We assume the graphs to follow a causal graph process (CGP) as defined in Mei and Moura (2015, 2017), that extends the Discrete Signal Processing on graph framework (DSP) to graph of time series. Hence, the CGP is built to capture dependencies in times series network data. The polynomial coefficients for each time lag correspond to causal graph filters that model delays across the network. As described by Mei and Moura (2015, 2017), with the graph filters being a polynomial of the adjacency matrix A, they allow a signal on a node to be affected through network effects by signals on other nodes at past time. The order of the polynome being limited to the current time lag i it cannot be influenced by more than the network effects from i time steps ago.

Hence, CGP assumes the graph at time k to follow an autoregressive process over M time lags. The current state of the graph, $x(k)$, is related to the lag l through a graph filter $P_l(A)$. The graph filter is considered to be a polynomial function over the adjacency matrix with coefficients $C = \{c_{l,j}\}$

defined by: $P_l(A) = \sum_{j=0}^{l} c_{l,j} A^j$. Without loss of generality we can fix the coefficients of the first time lag to be $(c_{1,0}, c_{1,1}) = (0, 1)$. Thus, the CGP at k can be expressed as:

$$\begin{aligned} x(k) &= w(k) + \sum_{l=1}^{M} P_l(A) x(k-l) \ , & (6.3) \\ &= w(k) + \sum_{l=1}^{M} \left(\sum_{j=0}^{l} c_{l,j} A^j \right) x(k-l) \ , & (6.4) \\ &= w(k) + Ax(k-1) + \cdots + \left(c_{M,0} I + \cdots + c_{M,M} A^M \right) x(k-M) & (6.5) \end{aligned}$$

Where $w(k)$ corresponds to Gaussian noise. Hence, the problem of reconstructing the CGP from a group of time series consists of estimating the adjacency matrix A and the polynomial coefficients C. Mei and Moura (2015, 2017) consider the optimisation problem:

$$(A, c) = \min_{A,c} \frac{1}{2} \sum_{k=M}^{K-1} \left\| x(k) - \sum_{l=1}^{M} P_l(A) x(k-l) \right\|_2^2 + \lambda_1 \|A\|_1 + \lambda_1^c \|C\|_1 \ , \quad (6.6)$$

where they include a LASSO penalty for both the adjacency matrix and the polynomial coefficients to enforce sparsity. They decompose this optimisation into different steps: first estimate the coefficients $R_i = P_i(A)$, then from those coefficients retrieve the adjacency matrix A, which allows the polynomial coefficients C to be obtained via another minimisation. Due to the L_1 penalty, the optimisation problem is not convex for R_1 and C.

For the first step, they perform a block coordinate descent over the matrix coefficients R_i. Each step of the descent is quadratic except for R_1 which has the L_1 regularisation term. From the obtained matrix coefficients R_i they perform an additional step in the block descent to obtain the adjacency matrix A. With the estimated adjacency matrix, we can reformulate the minimisation of Equation 6.6 in a function of the vector C with an L_1 regularisation term. In their paper, they do not go into details on how they perform these minimisations and just specify that they use a sparse gradient projection algorithm. The authors argue in favour of this algorithm because

it is particularly efficient, however only in the case of highly sparse problems; otherwise, for a dense graph this algorithm will scale as the cube of the number of nodes which renders it impractical. This motivates our interest in developing a cyclical coordinate descent (CCD) algorithm for this problem.

6.3 Estimating the adjacency matrix with coordinate descent

The sparse gradient projection (SGP) algorithm Mei and Moura (2015, 2017) used to solve each step of the block coordinate descent algorithm does not take full advantage of the structure of the problem. We therefore propose an efficient CCD algorithm for estimating the adjacency matrix and the CGP coefficients, and since the L_1 regularisation constraint in the optimisation problem for the matrix coefficients only applies to R_1, we consider the two cases $i = 1$ and $i > 1$ separately.

6.3.1 Update equation for $i > 1$

In the case of $i > 1$, the minimisation of Equation 6.6 on the matrix coefficient R_i simplifies to a quadratic problem, which is well suited for a CCD algorithm looping over the different lags i. Furthermore, since the minimisation is over a matrix, CCD allows us to avoid computing a gradient over a matrix; instead we compute the update equation for matrix R_i directly. We reformulate the loss function to isolate R_i with $S_k^i = x(k) - \sum_{l \neq i}^{M} R_l x(k-l)$, thus the Lagrangian is:

$$\mathcal{L}(R_i) = \frac{1}{2} \sum_{k=M}^{K-1} \left(S_k^i - R_i x(k-i)\right)^T \left(S_k^i - R_i x(k-i)\right) \ , \tag{6.7}$$

where S_k^i, x_k are vectors of size N. The derivative of $\mathcal{L}(R_i)$ with respect to the matrix R_i is:

$$\frac{\partial \mathcal{L}(R_i)}{\partial R_i} = R_i \sum_{k=M}^{K-1} x(k-i)x(k-i)^T - \sum_{k=M}^{K-1} S_k x(k-i)^T \ .$$

This derivative of $\mathcal{L}(R_i)$ with respect to R_i is equal to zero if:

$$R_i = \left(\sum_{k=M}^{K-1} S_k^i x(k-i)^T \right) \left(\sum_{k=M}^{K-1} x(k-i)x(k-i)^T \right)^{-1} . \tag{6.8}$$

which gives us an update equation for CCD. However, the matrix denominator $\sum_{k=M}^{K-1} x(k-i)x(k-i)^T$ is not guaranteed to be positive semi-definite. In case of singularity we can add an L_2 regularisation term, $\lambda_2 \|R_i\|_2^2$, to the utility function of Equation 6.7. With this coefficient the updating equation of matrix R_i becomes:

$$R_i = \left(\sum_{k=M}^{K-1} S_k^i x(k-i)^T \right) \left(\sum_{k=M}^{K-1} x(k-i)x(k-i)^T + 2\lambda_2 \mathbb{I}_N \right)^{-1} .$$

With $\mathbb{I}_N$ the identity matrix of size N. Thus, with this coefficient we add noise to the diagonal of the matrix to make it non-singular. In practice, Since this coefficient adds a bias, we select the lowest value of the coefficient λ_2 that makes this matrix non-singular. While this inverse step is expensive it does not depend on the matrices R_i. Thus, it can be computed in advance outside of the CCD loop. Hence, the update consists of a vector-vector multiplication followed by a matrix-matrix multiplication of size $N \times N$.

6.3.2 CCD FOR $i = 1$

For $i = 1$, the optimisation corresponds to Equation 6.6 with the L_1 regularisation term. We can follow the methodology of the previous section and derive a matrix update to compute the CCD step. However, in practice this solution produces matrices that are too dense. Hence, we instead constrain the sparsity on each node. This corresponds to running a CCD over the columns of the matrix R_1. Indeed, a column j of the adjacency matrix corresponds to the weight of the edges going from node j to the nodes it influences. To do so, we have to reformulate the loss function as a problem over the column j of R_1. Let us denote by R_1^{-j} the matrix R_1 without the column j, R_1^j the column j of the matrix R_1, $x^{-j}(k-1)$ the vector $x(k-1)$

without the term at index j, and $x^j(k-1)$ the value of the j-th term of $x(k-1)$. Thus, we can reformulate the error term to isolate the column j: $x(k) - \sum_{i=1}^{M} R_i x(k-i) = S_k^1 - R_1^{-j} x^{-j}(k-1) - R_1^j x^j(k-1)$. Therefore, the utility function of the minimisation over the column j of R_1 with a LASSO regularisation term follows as:

$$\mathcal{L}(R_1^j) = \frac{1}{2} \sum_{k=M}^{K-1} \left\| S_k^1 - R_1^{-j} x^{-j}(k-1) - R_1^j x^j(k-1) \right\|_2^2 + \lambda_1 |R_1^j| \ . \tag{6.9}$$

Due to the non-differentiable L_1 term we compute sub-gradient of the Lagrangian with respect to the vector of the column R_1^j:

$$\frac{\partial \mathcal{L}(R_1^j)}{\partial R_1^j} = R_1^j \sum_{k=M}^{K-1} \left(x^j(k-1) \right)^2 - \sum_{k=M}^{K-1} \left[S_k^1 - R_1^{-j} x^{-j}(k-1) \right] x^j(k-1)^T + \lambda_1 \Gamma_j \ .$$

Where $\Gamma_j(i) = sign(R_1^j(i))$ if $R_1^j(i) \neq 0$, else $\Gamma_j(i) \in [-1, 1]$ if $R_1^j(i) = 0$. Thus, we need to introduce the soft-thresholding function to obtain the CCD updating equation. We define the soft-threshold function as $S(a, b) = sign(a)(|a| - b)_+$, where $sign(a)$ is the sign of a and $(y)_+ = max(0, y)$. Then, the derivative of the utility function 6.9 is zero if:

$$R_1^j = \frac{S\left(\sum_{k=M}^{K-1} \left(S_k^1 - R_1^{-j} x^{-j}(k-1) \right) x^j(k-1), \lambda_1 \right)}{\sum_{k=M}^{K-1} (x^j(k-1))^2} \ . \tag{6.10}$$

The CCD algorithm for R_1 will loop by updating each column using Equation 6.10. As for the update of R_i, the denominator of Equation 6.10 can be computed outside the loop. Thus the complete algorithm consist of a CCD for each lag i with an inner CCD loop over the columns of R_1. Algorithm 1 shows the complete CCD algorithm to compute the matrix coefficients R_i of the CGP process. We stop the descent when the first of four criterion is reached: the maximum number of iterations is reached, the L_1 norm of the difference between the matrix coefficients R and its previous value is below a threshold ϵ, the L_1 norm of the difference between the new and previous in-sample MSE is below ϵ or the in-sample MSE increases. We then obtain the adjacency matrix by running an extra step of the columns CCD on R_1.

Algorithm 1 CCD algorithm to compute the matrix coefficients R

procedure COMPUTE-R(x, M, N, K)
  Compute the denominators of Equations 6.10 and 6.8 outside the loop
  $R = [zeros(N, N)\ ,\ \forall i \in [1, M]]$ ▷ Initialise coefficients at zero
  while Convergence criterion not met **do**
    Run CCD over the columns of R_1 with Equation 6.10
    Run CCD for each lag $i > 1$ with matrix update Equation 6.8
  end while
  Run the CCD over the columns of R_1 to obtain the adjacency matrix A
end procedure

6.3.3 Retrieving the polynomial coefficients c

Once we have retrieved the adjacency matrix A, the next step of the block coordinate descent is to estimate the polynomial coefficients C. To obtain these coefficients we minimise the MSE over the training set with both L_1 and L_2 regularisation terms. Hence, we can derive the CCD update for each coefficient $C_{i,j}$. We denote by $\hat{C}$ and $\hat{A}$ the estimated values of C and A respectively. Then, let us define $y_k = x_k - \hat{A}x_{k-1}$ and $w_k = \sum_{i',j' \neq i,j} C_{i',j'} \hat{A}^{j'} x_{k-i'}$. The error term of Equation 6.6 as a function of $C_{i,j}$ becomes $\left\| y_k - w_k - C_{i,j}\hat{A}^j x_{k-i} \right\|_2^2$. Thus, taking into account the L_1 and L_2 regularisation terms on C, the derivative of the Lagrangian with respect to $C_{i,j}$ is zero if:

$$C_{i,j} = \frac{S\left(\sum_{k=M}^{K} \left(\hat{A}^j x(k-i)\right)^T (y_k - w_k)\,, (K-M)\lambda_1^c\right)}{\sum_{k=M}^{K} \left(\hat{A}^j x(k-i)\right)^T \left(\hat{A}^j x(k-i)\right) + 2(K-M)\lambda_2^c}\,. \tag{6.11}$$

Hence, the CCD algorithm for C will loop over each coefficient $C_{i,j}$ and update it with Equation 6.11. This step completes the block coordinate descent to obtain the CGP process from the observed time series x. However, this CCD algorithm has three parameters: λ_1 for LASSO penalty used to obtain R_1, and λ_1^c and λ_2^c for the L_1 and L_2 regularisation terms used to compute the polynomial coefficients C. In this paper we focus on the estimation of

the adjacency matrix A and the choice of the LASSO parameter, while fixing the regularisation parameters of equation 6.11 to $\lambda_1^c = 0.05$ and $\lambda_2^c = 10^3$, for which the algorithm appears to be reasonably robust, as is evident from the results.

6.4 Selecting the L_1 coefficient

We now introduce two new non-parametric metrics to efficiently and automatically select the LASSO coefficient, λ_1, which directly influences the sparsity of the adjacency matrix A. A classic approach for selecting this involves computing the cross-validation error. When applied to time series this corresponds to computing a prediction error over an out-of-sample time window, see for example Wu and Lange (2008); Mei and Moura (2015, 2017), in which the authors use the estimated CGP variables $\hat{A}$ and $\hat{C}$ to make a prediction and use the MSE to assess its quality. This technique implies a direct relationship between the sparsity level of the adjacency matrix and the MSE of the prediction, which is questionable in practice. Figure 6.1 plots the evolution of the prediction MSE over increasing values of λ_1, alongside the percentage difference in the number of edges between the estimated adjacency matrix and the real one computed for a simulated CGP following a Stochastic Block Model (SBM) graph, i.e. the difference in number of edges divided by the total possible number of edges.

It is clear that current techniques do not produce good results and we do not want to follow Friedman et al. (2008); Wang (2013) in fixing different values of λ_1 producing a sparse and a dense result without knowing which one is correct. Thus, we need a metric that weights the improvement in prediction error against an increased number of edges. In statistical modelling the AIC and BIC criteria aim to avoid over-fitting by including a penalty on the number of input-variables, which works when the number of output-variables to be predicted is not related to the selected set of input-variables. However, in the case of an adjacency matrix the sparsity of each node also impacts the number of nodes predicted; we want sparsity in the adjacency matrix

to encourage each node to more accurately predict a small subset of other nodes. Following this idea further we derive two new error metrics.

6.4.1 Two new distance measures for selecting the L_1 coefficient

We want a distance metric that has no parameters and a maximum around the exact number of edges of the adjacency matrix. An appropriately sparse adjacency matrix has few edges but enough to accurately reproduce the whole CGP process. What is important is not the prediction quality obtained by the complete adjacency matrix but rather the prediction quality of each node *independently*; the adjacency matrix should have few edges connecting nodes with each connection having a low in- and out-of-sample error. For each node we compute the errors of each connection with other nodes. We sum these errors over all edges and compute the average over time divided by the number of edges. From this error, we compute the sum over all nodes to obtain an error metric of the whole graph:

$$err = \sum_{j=1}^{N} \frac{1}{\sum_{i=1}^{N} \mathbb{I}_{\{A_j \neq 0\}}(i)} \frac{1}{K-M} \sum_{k=M}^{K} \left\| x(k)\mathbb{I}_{\{A_j \neq 0\}} - Ax_j(k-1) \right\|_2^2 , \tag{6.12}$$

where $1_{A_j \neq 0}$ corresponds to a vector of zeros with ones only where the lines of the column A_j of the adjacency matrix A are non-zero. Compared to the in-sample MSE of node j the error metric of Equation 6.12 focuses on the error in the nodes it is connected to. Since we divide by the number of edges, the error should increase as sparsity increases, but it should start decreasing once the gain in the prediction quality of each edge offsets the decrease in number. Intuitively this peak should correspond to the underlying sparsity level of the graph under study; before the peak, the model has too many parameters with poor individual prediction quality, whereas after the peak, the model has too few edges with very low individual error.

The error metric of Equation 6.12 averages over the number of edges of each node j. Another approach would be to work with the degree of the graph

instead, by which we mean the sum of the absolute values of the weights of the edges of node j. Thus we define the error with degree by:

$$err^d = \sum_{j=1}^{N} \frac{1}{\sum_{i=1}^{N} |A_{i,j}|} \frac{1}{K-M} \sum_{k=M}^{K} \left\| x(k)\mathbb{I}_{\{A_j \neq 0\}} - Ax_j(k-1) \right\|_2^2 \qquad (6.13)$$

We simulate a CGP on a stochastic block model following the methodology and parameters in Mei and Moura (2015, 2017), which we describe in Section 6.5. On this simulated graph we test our intuition by comparing the evolution of our error metrics as a function of the sparsity parameter λ_1. Figure 6.1 shows the evolution of these metrics as well as the commonly used in- and out-of-sample MSE and the AIC and BIC criteria. On this figure both metrics, err and err^d, perform as expected while the others do not show any relationship to the number of edges in the adjacency matrix except for the BIC criterion. While this graph represents only one sample of a simulated scenario, this behaviour is consistent throughout the different simulations in Section 6.5. Interestingly, our two metrics complement each other; on average, for dense graphics err often produces better results while for sparser ones err^d is often better. In practice we can use the following pragmatic approach for robust results: if both metrics have a peak we take the mean value of the two resulting λ_1, if only one has a peak we take its value for λ_1. The Table 6.1 compares the performance of these different metrics for different SBMs. It is interesting to observe that the performance of the two error metrics we propose is on par with the BIC criterion and actually complement is. Indeed, we observe the best results when averaging the selected λ_1 of err, err^d and BIC. Since the BIC criterion is widely known, for the rest of this paper we will study the performance of the two new error distances err and err^d that we propose knowing that the resulting adjacency matrix would be better if the BIC criterion was included.

6.5 Applications

We are interested in two performance metrics: the accuracy of the LASSO coefficient selection, assessed by measuring the difference in number of edges

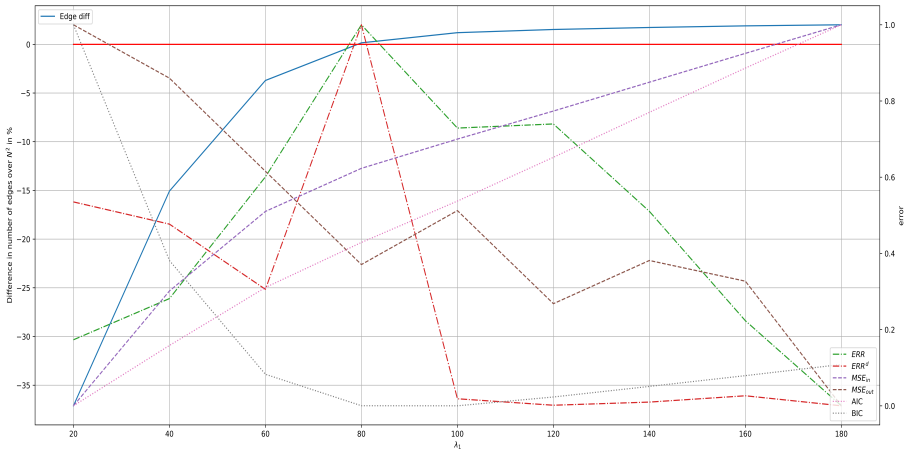


Figure 6.1: Comparison of the evolution of different metrics as a function of the value of the LASSO coefficient λ_1 for a simulated CGP-SBM graph with 200 nodes, 5 clusters, 3 lags and 1040 time points. The left axis indicates the number of different edges between the estimate and the true adjacency matrix. The blue line shows the evolution of that difference as a function of the coefficient λ_1, while the red line highlights the zero mark. The different error metrics are rescaled to be between 0 and 1 on the right y-axis. We compare the two metrics proposed in this paper, err and err^d, with the in- and out-of-sample error, MSE_{in} and MSE_{out}, as well as the AIC and BIC criteria, AIC and BIC

between the real and estimated adjacency matrix; and the quality of the CCD algorithm, assessed by measuring the percentage of true positives and false positives. For the computations we fix two parameters of our algorithm: the maximum iterations $maxIt = 50$ and the convergence limit $\epsilon = 0.1$. For the initialisation, we observe that the algorithm performs better when starting from zeros, i.e. $Rh = [0]$, than from random matrices. When starting with matrices of zeros, due to the block coordinate structure of the algorithm, the first step corresponds to solving the mean square problem of a CGP with only one lag, then the second step considers a CGP with two lags, whereby we learn on the errors left by the first lag, and so on. Thus, each step of the descent complements the previous lags by iteratively refining previous predictions.With this approach the solutions never skip any time link but naturally stops after certain number of lags once the error is null.

We use the same causal graph stochastic block model (CGP-SBM) structure to assess the performance of our algorithm as Mei and Moura (2017), and hence our results are directly comparable. However, they focus on minimising

Table 6.1: Differences between A and its estimate $\hat{A}$ on CGP-SBM graph for different distance metrics.

Distance measure	N	Nc	M	K	NBDE	NBDE (%)	True positive	False positive
Err & Err^d	200	5	3	1040	242	0.39% (0.35)	62.7% (6.0)	15.4% (9.3)
BIC	200	5	3	1040	175	0.48% (0.20)	62.8% (4.1)	21.8% (5.8)
MSE_{out}	200	5	3	1040	615	1.54% (0.35)	40.3% (11.6)	0.2% (3.2)
AIC	200	5	3	1040	8124	20.31% (5.36)	82.2% (4.2)	91.0% (3.5)
MSE_{in}	200	5	3	1040	8124	20.31% (5.36)	82.2% (4.2)	91.0% (3.5)
Err & Err^d	500	5	3	2080	785	0.31% (0.33)	63.0% (7.0)	23.9% (10.4)
BIC	500	5	3	2080	1552	0.62% (0.33)	74.3% (4.5)	40.8% (9.0)
Err & Err^d & BIC	500	5	3	2080	942	0.38% (0.28)	66.8% (5.5)	27.1% (10.9)
Err & Err^d	1000	10	3	2080	4059	0.33% (0.34)	64.9% (8.9)	18.8% (9.3)
BIC	1000	10	3	2080	3106	0.31% (0.21)	71.3% (6.9)	35.8% (9.2)
Err & Err^d & BIC	1000	10	3	2080	1965	0.16% (0.33)	67.8% (8.1)	24.8% (9.9)

the MSE and choose the LASSO coefficient that produces the best results. Although our approach results in a higher MSE, we have shown in Section 6.4 that the MSE is not linked to the sparsity of the graph. Thus, our algorithm obtains a more accurate estimate of the adjacency matrix and its approximate sparsity.

The SBM consists of a graph with a set of clusters where the probability of a connection is higher within a cluster than between them. This structure is interesting because it appears in a wide variety of real applications. The parameters to simulate a CGP-SBM are the number of nodes, N, the number of clusters, Nc, the number of lags, M and the number of time points, K. For each simulation we use a burn in of 500 points. For a visual assessment of our proposed algorithm's performance we show in Figure 6.2 the absolute value of the adjacency matrix used to simulate the CGP and the estimation we obtain with our algorithm. We note that it is hard to visually detect the discrepancies. Hence we added in appendix Figure 6.4 to show the matrix of the differences between real and estimated adjacency matrices and, Figure 6.3 to show the non-zero elements of each matrix and the matrix of differences with a black square at each non-zero edge.

We also wish to quantitatively assess the accuracy of the estimated adjacency matrix. Hence, we measure the quality of the results by considering different metrics: the difference in the number of edges between $\hat{A}$ and A as a abso-

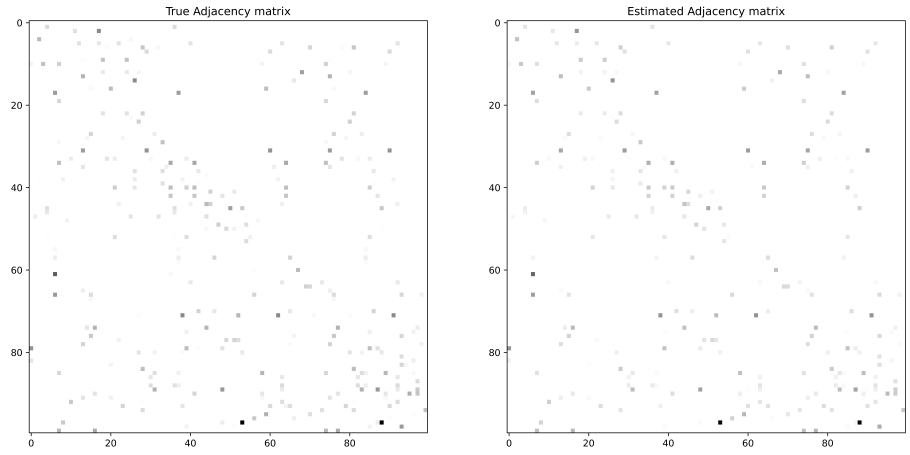


Figure 6.2: Estimated adjacency matrix $\hat{A}$, on the right; the true one A, on the left. Absolute values of the weights are shown for better visualisation, with blacker points representing bigger weights. The estimation was performed on a CGP-SBM graph with $N = 100$, $Nc = 5$, $M = 3$ and $K = 1560$.

lute value and as percentage of the total number of possible edges, N^2; the percentage of true positive, i.e. the number of edges in $\hat{A}$ that are also edges in A over the total number of edges in A; the percentage of false positive, i.e. the number of edges in $\hat{A}$ that are not in A over the total number of edges in $\hat{A}$; the mean squared error $MSE = \|\hat{A} - A\|_2^2/N^2$. The two metrics measuring the difference between the real and the selected number of edges assess the performance of the selection of the sparsity coefficient λ_1. The true and false positive rates assess the performance of the CGP-CCD algorithm 1 to compute the adjacency matrix. For assessing the consistency of the performance we simulate different environments and compute the median of the measures obtained over 10 samples. In the simulated environments, the sparsity level, i.e. the number of non-zeros elements in the adjacency matrix, varies between 1.4% and 3.0% with an average at 2.1%. Table 6.2 shows that the results are consistent for different graph sizes, sparsity levels, lags and numbers of time points with a median over all the environments of percentage-difference in number of edges of 0.41%, true positive rate of 65.9% and false positive rate of 20.8%. Interestingly, even when the number of time points was too small to obtain accurate results, i.e. high true positive rate, the percentage of different edges stays small, below 0.5%. The results in

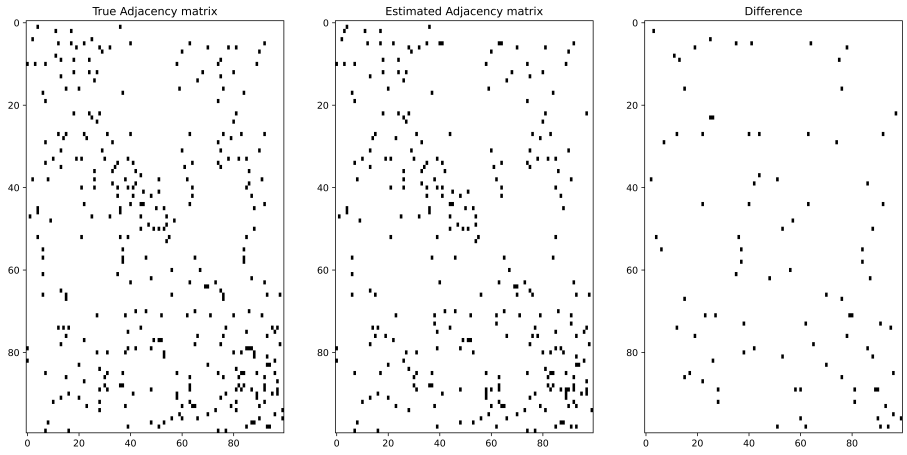


Figure 6.3: This figure compares the selected edges of the estimated adjacency matrix $\hat{A}$, in the middle, to the true one A, on the left. The right matrix correspond to the difference these two matrices. Each black square corresponds to an edge with non-zero weight. The estimation was performed on a CGP-SBM graph with $N = 100$, $Nc = 5$, $M = 3$ and $K = 1300$.

Table 6.2 assume we know the number of time lags of the underlying CGP we are looking for. Since this assumption is unlikely to hold on real datasets we tested the reliability of the performances by using wrong input parameters. We therefore simulated a graph with $M = 5$ lags and ran the learning algorithm for $M = 3$ lags and vice-versa, in both scenarios the average results were approximately unchanged. Section 6.6 further studies the performance of the algorithm on real datasets with unknown parameters.

6.5.1 Computation time complexity

All the computations employed Python 2.7 using Numpy and Scipy-sparse libraries, which can use up to 4 threads. The Stochastic Gradient projection (SGP) method used by Mei and Moura (2015, 2017) is especially efficient for highly sparse environments, which can leverage sparse matrix-vector computation. In our environment, when we have a graph with more than 1% of non-zero weights a matrix-matrix or matrix-vector multiplication using the sparse function of Scipy-sparse is slower than using the dense functions of Numpy. Thus, for graphs with an adjacency matrix with more than 1% of non-zero edges we use the dense library. For example, on a CGP-SBM

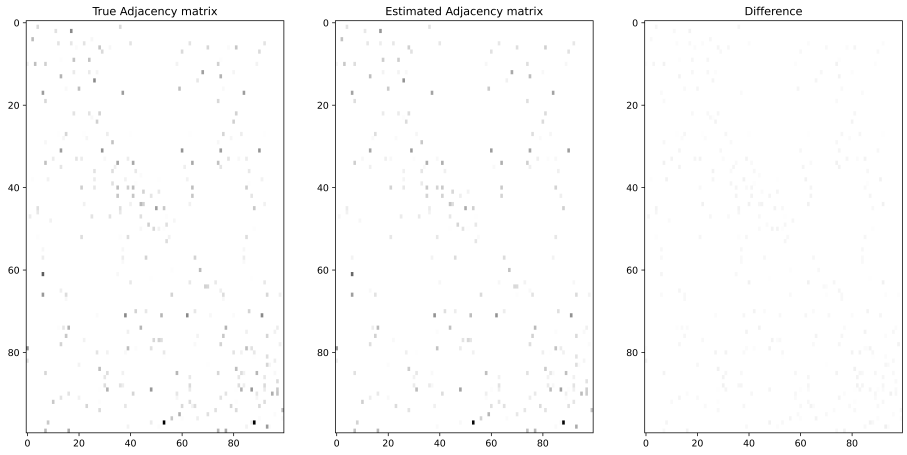


Figure 6.4: This figure compares the estimated adjacency matrix $\hat{A}$, on the right, to the true one A, on the left. The right matrix correspond to the difference these two matrices. To obtain a better visualisation the plots represent the absolute values of the weights, hence the blacker the bigger the weight. The estimation was performed on a CGP-SBM graph with $N = 100$, $Nc = 5$, $M = 3$ and $K = 1300$.

graph with parameters $(N, Nc, M, K) = (200, 5, 3, 2080)$ with $2,4\%$ of non-zeros edges, our CCD algorithm solves the optimisation of Equation 6.9 to obtain the matrix R_1 faster than the SGP algorithm by more than 100-fold. When the graph size increases to $N = 500$, CCD is faster than by more than 350-fold.

We perform an empirical time complexity estimation of the complete block coordinate descent algorithm by measuring the evolution of the execution time as a function of each parameter (N, Nc, M, K) individually. Increasing the number of time lags M has a negligible effect on the execution time of the CCD to obtain the adjacency matrix, although the computation of the polynomial coefficients C scales quadratically with the lags M. We observe that the execution time is not affected by the sparsity level Nc. However, it scales linearly as function of the number of time points K and quadratically as a function of the number of nodes N. This quadratic complexity in N can be tempered in different ways however. In the case of highly sparse graphs we can leverage sparse libraries, and an even faster solution for both dense and sparse graph is to perform the computations on a GPU. Indeed, Algorithm 1 does not use much memory and can thus be computed entirely in

Table 6.2: Differences between the adjacency matrix A and its estimate $\hat{A}$ for different CGP-SBM environments. *NBDE*: absolute difference in the number of edges between $\hat{A}$ and A, and as a percentage of the total number of possible edges N^2 with *NBDE (%)*. *True positive*: number of edges in $\hat{A}$ that are edges in A over the total number of edges in A. *False positive*: number of edges in $\hat{A}$ that are not in A over the total number of edges in $\hat{A}$. *Mean squared error*: $MSE = \|\hat{A} - A\|_2^2/N^2$.

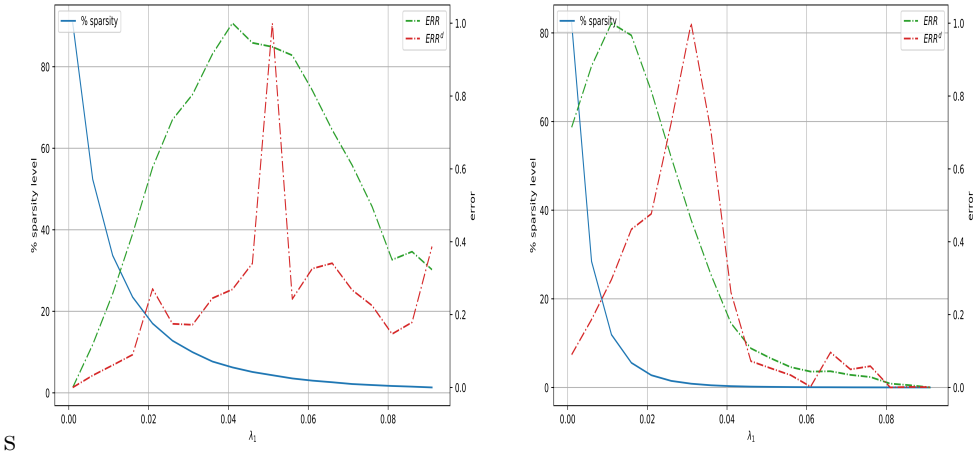
N	Nc	M	K	MSE	NBDE	NBDE (%)	True positive	False positive
100	5	3	1040	2.0×10^{-4}	40.5	0.41% (0.28)	72.4% (7.6)	20.8% (11.4)
200	5	3	1040	2.5×10^{-4}	115.0	0.29% (0.29)	65.9% (4.1)	25.4% (9.3)
200	10	3	1040	1.6×10^{-4}	198.5	0.50% (0.46)	67.0% (8.6)	26.1% (18.9)
200	5	5	1040	2.3×10^{-4}	208.0	0.52% (0.27)	63.9% (4.7)	26.2% (13.7)
200	5	3	2080	1.2×10^{-4}	135.5	0.34% (0.27)	73.7% (6.3)	21.1% (10.0)
500	5	3	2080	1.7×10^{-4}	1722.5	0.69% (0.26)	61.3% (4.6)	17.5% (4.9)
1000	10	3	2080	9.3×10^{-5}	4835.5	0.48% (0.26)	56.8% (6.7)	17.0% (9.7)
1000	10	3	4160	6.6×10^{-5}	3989.5	0.40% (0.27)	66.7% (8.5)	15.1% (9.7)
5000	50	3	5000	1.3×10^{-5}	87709.5	0.35% (0.05)	64.3% (3.5)	14.0% (8.6)
Median				1.7×10^{-4}	208.0	0.41% (0.12)	65.9% (5.2)	20.8% (4.7)

the GPU memory. With a GPU implementation using the library PyTorch, the algorithm has a 20-fold speed-up compared to our CPU implementation with matrix computations parallelised over 4 threads.

6.6 Application to financial time series

We now apply our algorithm to a real dataset of stock prices consisting of the 371 stocks from the S&P500 that have quotes between 2000/01/03 and 2018/03/27. Since we do not know the exact adjacency matrix of this environment, we test the accuracy of the obtained graph by studying how it changed following a known market shock. More specifically we compute two graphs, one before and one after the financial crisis of 2008/2009. For both graphs to use the same number of time points, the first uses prices from 2004/11/15 to 2009/01/01 and the second from 2009/11/12 to 2014/01/01. We chose to build the graphs with a 4 year time window, $K = 1040$, since in simulations with the same number of nodes it produces good results. The lag was fixed to one week, $M = 5$, since there are documented trading patterns at a weekly frequency. We note that in the time windows studied modifying the lags to $M = 3$ or $M = 10$ has negligible impact on the results. As we can see in Figure 6.5, for both time windows the error metrics peaked at slightly

different values, thus we took the mean of the two for estimating the LASSO coefficient.



S

(a) From 2004/11/15 to 2009/01/01 **(b)** From 2009/11/12 to 2014/01/01

Figure 6.5: This figures shows the obtained sparsity and error metrics for an environment of 371 stocks from the S&P500 using the close prices for two different time windows specified bellow each sub-figure and considering a weekly lag $M = 5$. On the left axis the blue line shows the evolution of the sparsity level of the adjacency matrix in function of the LASSO coefficient λ_1. The right axis correspond to the evolution of the two errors metrics err and err^d we propose in this paper.

Interestingly, the algorithm selects a much sparser matrix after the crisis with a sparsity level decreasing from 5.1% to 2.8%. This evolution can be visualised in Figure 6.5 in the appendix, which shows the evolution of the sparsity level and error metrics as a function of λ_1. This points to a more inter-connected market leading up to the crisis. Since the crisis was due to sub-prime issues, one might expect real-estate and financial firms to have many edges in the graph and influence the market in the pre-crisis period, with the importance of these firms decreasing after the crash. Indeed, this aspect is reflected in the estimated adjacency matrix; before the crisis, financial firms represent more than 60% of the top ten nodes with the highest number of connections, while it decreases to less than 40% afterwards, including insurance firms. Furthermore, before the crisis the firm with the highest number of connection was GGP Inc., a real estate firm which went on to file for bankruptcy in 2009. While financial and oil firms represented more than 70% of the top 20 most connected nodes before the crisis, the graph of 2014 is much more diversified with more sectors in the top 20 and none represent-

ing more than 30%. Figure 6.6 shows the evolution of the adjacency matrix before and after the crisis with the stocks grouped by financial sectors using the BCIS (Bloomberg Industry Classification Standard) sector classification. For a better visualisation we do not show the weight of the edge but instead a black square for every non-zero edge., and we can see the shift in importance and the increase in sparsity. Overall, the post-crisis market is sparser and less concentrated than before 2009, with fewer edges linked to financial firms.

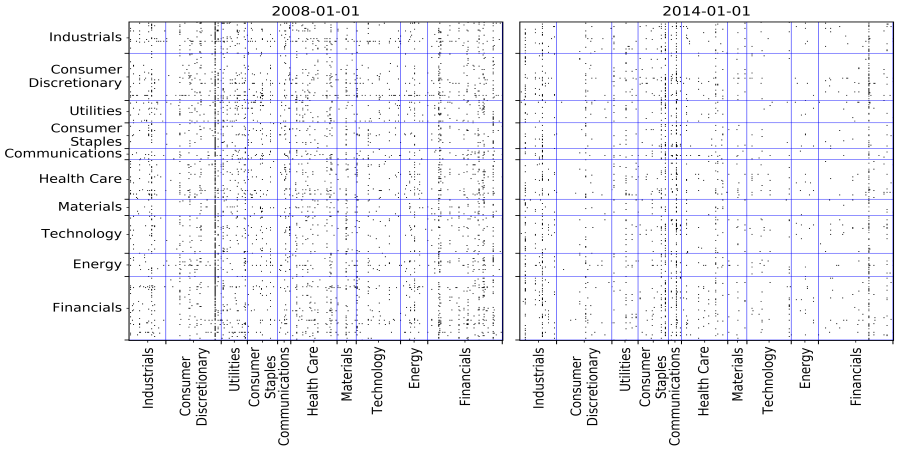


Figure 6.6: This figures shows the obtained adjacency matrix for a weekly lag $M = 5$ on 467 European stocks computed using data between $06/11/2003$ and $01/01/2008$ on the left, and $03/11/2009$ to $01/01/2014$ on the right. The stocks are grouped by financial sectors using the BCIS (Bloomberg Industry Classification Standard) sector classification. For a better visualisation we do not show the weight of the edge but instead a black square for every non-zero edge. The sparsity level was selected by taking the average λ_1 value obtained with the error metrics err and err^d.

Since the CGP-CCD algorithm automatically selects the sparsity level, in addition to studying the different connections we can also study the evolution of the sparsity level. Let us define the sparsity level as the percentage of non-zeros edges, hence 100.0 corresponds to a fully connected graph. We estimate the variance of the market by the average of the log-Realised-Variance, $log(RV)$, of each stock. We computed the $log(RV)$ with exponential weighting on the daily log-returns, using a coefficient of 0.99 and time window of 40 days. Figure 6.7 shows the evolution of the sparsity level of the adjacency matrix and of the $log(RV)$ of the market through time. We computed the

adjacency matrix every 6 months and the corresponding $log(RV)$ at the last date of that time window. We can observe an interesting correlation between the increase in density of the causal graph and the increase in the realised variance.

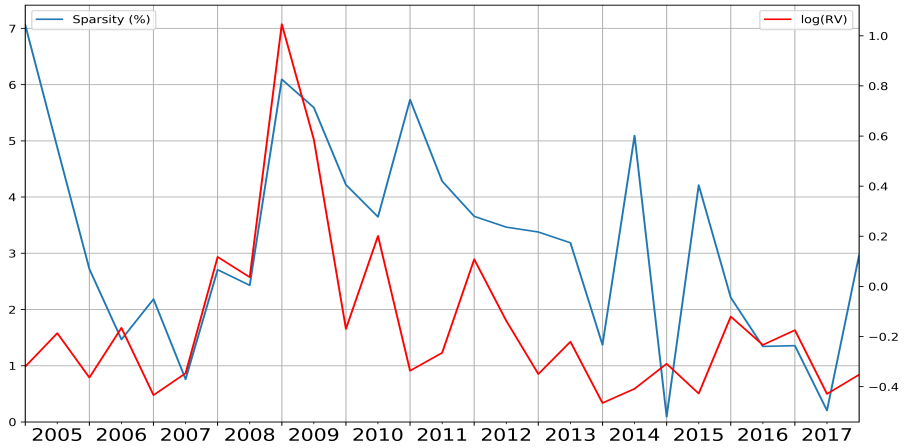


Figure 6.7: For 371 stocks from the S&P500, evolution of the sparsity level of the adjacency matrix, left y-axis, and the $log(RV)$ on the right y-axis. The coefficient λ_1 value obtained with the error metrics err and err^d. We define the sparsity level by the percentage of non-zeros edges, hence 100.0 corresponds to a fully connected graph. We estimate the variance of the market by the average of the log-Realised-Variance, $log(RV)$, of each stock. We computed the $log(RV)$ with exponential weighting on the daily log-returns, using a coefficient of 0.99 and time window of 40 days.

To test our algorithm on a different environment we apply the CGP-CCD algorithm on stocks from the EU markets. We select the 467 most liquid stocks from different European markets with data between 01/01/2000 to 01/01/2018. Since we do not select stocks with a price history from 2000, when the quote is missing for a day we carry over the previous price. We run the CGP-CCD algorithm on the daily log-returns with a time windows of 4 years, $K = 1040$, and a weekly time lag, $M = 5$. Following the same methodology as for the US stocks, Figure 6.8 shows the evolution of the adjacency matrix before and after the crisis with the stocks grouped by financial sectors using the BCIS (Bloomberg Industry Classification Standard) sector classification. We can do the same observation as for the US datasets on this European environment, the graph is much denser before the crisis with more weight on the financial sector.

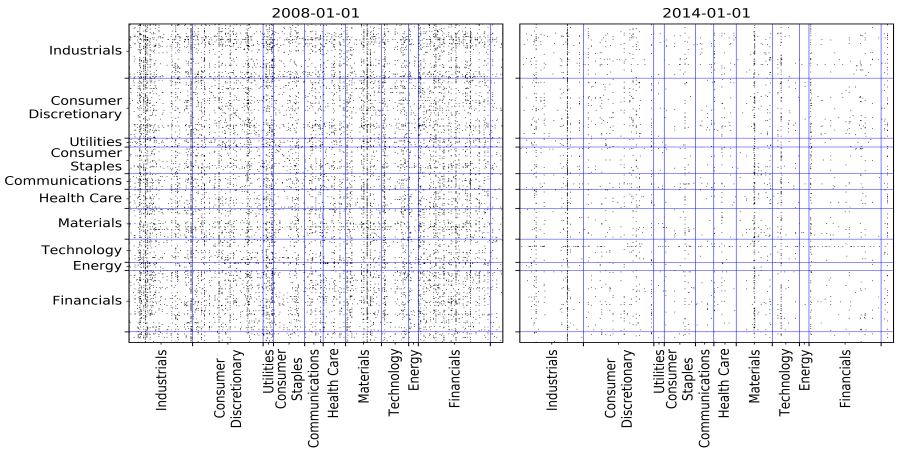


Figure 6.8: This figures shows the obtained adjacency matrix for a weekly lag $M = 5$ on 467 European stocks computed using data between $06/11/2003$ and $01/01/2008$ on the left, and $03/11/2009$ to $01/01/2014$ on the right. The stocks are grouped by sector. The sparsity level was selected by taking the average λ_1 value obtained with the error metrics err and err^d.

Furthermore, the network of dependencies stay relatively stable through time, to illustrate that point Figure 6.9 shows the number of outward connections for each node at different time. The dependencies tend to evolve with large market changes but during each cycle the main connections of each node are stable.

6.7 How to learn the covariance structure

The CGP-CCD algorithm we introduced in this chapter learns the lagged causal relation between different time series. However, the methodology used to developed this CGP-CCD algorithm can be applied to the simultaneous connections problem. In this section we will described the previous approach at estimating the sparse covariance matrix with a CCD algorithm and implement a CGP-CCD variant that learns the simultaneous relations between a set of time series.

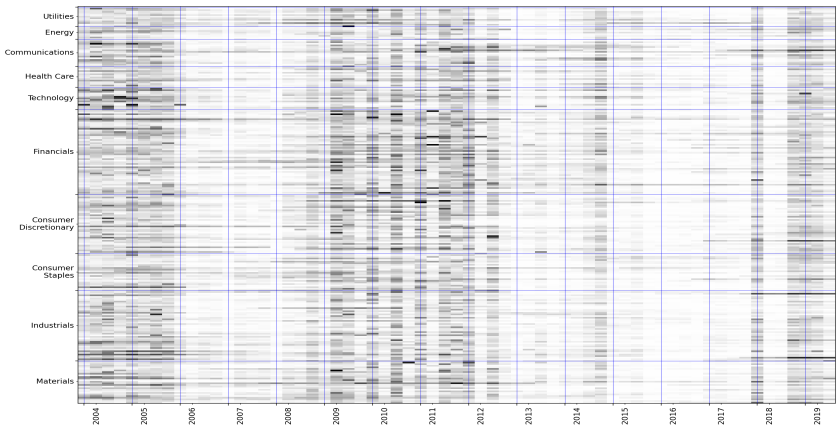


Figure 6.9: This Figure represent the evolution of the number of outward connections per node on 467 European stocks computed using data between 06/11/2003 and 01/06/2019. On the Y-axis, each line corresponds to a stock, grouped by sectors. The X-axis corresponds to the dates at which the adjacency matrix was computed. At a specific time t, the darker the square the higher number of outward connections that node has.

6.7.1 Graphical-LASSO approach

This section describes the graphical LASSO presented by Friedman et al. (2008). Let us consider a graph with p nodes, N observations that follows a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$, the precision matrix $\Theta = \Sigma^{-1}$ and the empirical covariance matrix S. In Friedman et al. (2008) they estimate the sparse covariance matrix by maximising the log-likelihood of the multivariate Gaussian distribution of the graph:

$$\log\det\Theta - tr(S\Theta) - \rho\|\Theta\|_1 \ .$$

In Banerjee et al. (2007) they show that this MLE problem is convex and use that result to propose a block coordinate descent algorithm to solve it. Let W be the estimate of the covariance matrix, then we can decompose it into different blocks:

$$W = \begin{pmatrix} W_{1,1} & w_{1,2} \\ w_{1,2}^T & w_{2,2} \end{pmatrix} \quad , \quad S = \begin{pmatrix} S_{1,1} & s_{1,2} \\ s_{1,2}^T & s_{2,2} \end{pmatrix} \ .$$

In Banerjee et al. (2007) they prove that $w_{1,2}$ is the solution to the following minimisation problem:

$$w_{1,2} = \min_{y} y^T W_{1,1}^T y \ \ , s.t. \ \ \|y - s_{1,2}\|_\infty \leq \rho \ ,$$

and that this optimisation is equivalent to the dual:

$$\min_{\beta} \frac{1}{2} \|W_{1,1}^{1/2} \beta - b\|^2 + \rho \|\beta\|_1 \ ,$$

with $b = W_{1,1}^{-1/2} s_{1,2}$. Once the optimal β is found, the solution of the primal problem is $w_{1,2} = W_{1,1}\beta$. In Friedman et al. (2008) they leverage the fact that the dual problem is similar to a LASSO and use a CCD algorithm to solve it. They find the CCD update to be:

$$\hat{\beta}_j = S\left(u_j - \sum_{k \neq j} V_{k-j} \hat{\beta}_k, \rho\right) / V_{j,j} \ ,$$

with $V = W_{1,1}$ and $u = s_{1,1}$. With this CCD algorithm they can compute a sparse projection of the dense covariance matrix even in large environments with thousands of nodes. However, this method requires a dense covariance matrix as input and the sparsity of the resulting matrix depends on the choice of the LASSO parameter λ_1. Thus the results are highly dependent on the quality of the computation of the empirical covariance matrix used as input and the choice of the regularisation parameter. In other words, the graph-LASSO algorithm constructs a sparse estimate of an empirical covariance estimation. Following the logic of the CGP-CCD algorithm we presented in the previous section, we will construct a sparse covariance matrix directly from the data without going through the dense computation and doing a projection.

6.7.2 Extend the CGP-CCD to simultaneous relations

We assume a basic SGDLM model, with a fixed parental set and common $AR(1)$ process for all the time series. Let us denote by X_t^{-i} a vector composed

of all the time series at time t except for asset i. Then, each time series i follows $x_t^i = A_i x_{t-1}^i + B_i X_t^{-i} \epsilon_i$ where B_i represents the simultaneous and A_i the autoregressive coefficients. In other words, as with the SGDLM model, we assume that the value of a time series at time t is influenced by the simultaneous value of other time series in the environment. Hence, let X_t be a vector of m time series values at t we have:

$$X_t = AX_{t-1} + BX_t + \epsilon \ ,$$

where A is a diagonal matrix, B is a matrix with zeros on the diagonal and ϵ is a Gaussian noise with variance σ. With this formulation the time series follow a multivariate Gaussian distribution:

$$X_t \sim \mathcal{N}\left((I-B)^{-1} AX_{t-1}, \Sigma\right) \ , \tag{6.14}$$

$$\Sigma = \left((I-B)^{-1}\right)^T \sigma \left(I-B\right)^{-1} \ . \tag{6.15}$$

where σ is a diagonal matrix with the variance of each time series on its diagonal. Our aim is to estimate the matrices A and B. To do so we perform a block descent where we first estimate B then A and cycle until convergence. We initialise the algorithm by assuming no simultaneous coefficients, thus the least square estimate of the lag parameter A is $A = \left(X_{t-1}^T X_{t-1}\right) X_{t-1}^T X_t$. Following the methodology of the CGP-CCD algorithm we estimate B by cycling through the time series to perform the CCD update. The step j of the CCD minimises the loss function:

$$L(B^j) = \sum_{t=0}^{T} \|X_t - AX_{t-1} - B^{-j} X_t^{-j} - B^j x_t^j\|_2^2 + \lambda_1 \|B^j\|_1 \ .$$

where we added a LASSO regularisation term to enforce sparsity on the matrix of simultaneous coefficients. With $E_t^j = X_t - AX_{t-1} - B^{-j} X_t^{-j}$ the CCD update for step j is:

$$B^j = \frac{S\left(\sum_{t=0}^{T} E_t^j x_t^j, \lambda_1\right)}{\sum_{t=0}^{T} (x_t^j)^2} \ .$$

Once we have updated the matrix B we update the coefficients A with the least square estimation:

$$A = \left(X_{t-1}^T X_{t-1}\right) X_{t-1}^T [(I - B)\, X_t] \ .$$

The algorithm cycle though those two updates until convergence. As for the CGP-CCD algorithm the LASSO parameter λ_1 plays an important role. Hence, we follow the same methodology as described previously to select this parameter according to err, err_d and BIC distances.

6.7.3 Results of the simultaneous CGP-CCD

We simulated an environment of time series with simultaneous connections by sampling from the multivariate distribution of Equation 6.14. For the simulation we sampled the coefficients A, B and σ from uniform distributions. Thus, we can run the simultaneous CGP-CCD algorithm on this simulated environment and compare the estimated matrix $\hat{B}$ to the real one B with the same metrics as Table 6.2. We simulated environment of different sizes with different level of sparsity but our observations were constant throughout those situations.

While the sparsity selection, i.e. the number of edges, was as accurate as the results obtained for the lagged CGP-CCD algorithm and detailed in Table 6.2, the quality of the estimated sparse covariance matrix was not. The percentage of correct edges, i.e. percentage of true positives, stayed well below 50% in every scenario. Unfortunately, we are still trying to understand why this simultaneous version of the CGP-CCD algorithm does not work as expected.

6.8 Summary of CGP-CCD

We have proposed a cyclical coordinate descent algorithm to efficiently infer the directed adjacency matrix and polynomial coefficients of a causal graph process. Compared to the previous state-of-the-art our solution has lower

complexity and does not depend on the sparsity level of the graph for scalability. Furthermore, we propose two new error metrics to automatically select the coefficient of the LASSO constraint. Our solution is able to recover approximately the correct number of edges in the directed adjacency matrix of the CGP. The performance of our algorithm is consistent across the different simulated stochastic block model graphs we tested. In addition, we provided an example application to a real-world dataset consisting of stocks from the S&P500, demonstrating results that are in line with economic theory.

7

Causal-Heterogenous-SGDLM

7.1 Introduction

We want to construct for all the stocks in a market a multivariate model that adapts to the datasets and can evolve with time. That model must bring new economic information on the behaviour of both each time series individually and the market as a whole. We can cluster the factors that impact the time series of stock-prices into an endogenous and an exogenous group. The endogenous part represents the information of that particular time series such as its past behaviour and trends. The exogenous factors represent the impact of its environment on the time series under study. The HSGDLM model we presented in Chapter 4 uses for the exogenous factors the simultaneous connections between the different time series. However, as we detailed in Chapter 6 the causality between different time series can be either simultaneous or lagged. Thus, a more accurate representation of the exogenous factors impacting a time series should be able to model both possible causal links. Different time series and environment could have different type of connections, such for a model to adapt to these different environments it should be able to have the options between those different causal

links and not enforce one. In other words, the model must be able to select no lagged factors or no simultaneous factors or no exogenous factors.

An important part of the HSGDLM model is its computational speed allowing us to run large models but learning lagged causal links between a large group of time series can become tedious. Thanks to the CGP-CCD algorithm presented in Chapter 6 we can efficiently-compute the adjacency matrix of lagged connections in large environment. In this chapter we will combine the results from the CGP-CCD algorithm into the HSGDLM framework to create a morel of the market taking into account both causal and simultaneous exogenous effects. Thus creating the Heterogenous Causal and Simultaneous Dynamic Linear Model (HCSGDLM).

Compared to the HSGDLM model from Chapter 4, adding GC causal variables improves both the direct one-step ahead predictions and longer prediction through the change point method. Interestingly, while the performance increase on the US dataset is marginal, on European stocks it has a much bigger impact. This may be due to the higher heterogeneity of markets present in that dataset. In addition, adding causal coefficient drastically improves the reliability of the information created with the variable ($exogenouscoefficients - endogenouscoefficients$). For both dataset the backtest with this variable becomes much more linear through time. Overall, adding causal coefficient improves the decomposition of the time series in exogenous and endogenous variables and the reliability of the different information created.

This chapter starts by detailing the different types of causality we would like our model to capture in Section 7.2. Then in Section 7.3 we detail how these new factors are integrated into the HSDGLM model. Finally, Section 7.4 shows the results obtained for a dataset of European stocks and compare them to the the HSGDLM model without causal factors.

7.2 Capturing both simultaneous and lagged causality

7.2.1 Simultaneous parents

Let us recall how the simultaneous parents are selected by the HSGDLM model presented in Chapter 4. The simulatenous parents are divided into three groups: the core-set corresponds to the selected parents, in the up-set are potential future parents and in the down-set are previous candidates or member of the core-set which have been excluded. We first consider a multivariate-Wishart-DLM where the covariance following an inverse-Wishart distribution and the observations following a simple $AR(1)$ process. Then, for each time series the algorithm selects the n_{up} variables with the highest covariance coefficient from the multivariate-Wishart-DLM and add them to the up-set. Those selected variables will then be included into the individual DLM of the respective time series as an exogenous factor. Then, after Δ_T time steps the members of the up-set compete against the members of the core-set for a place in the core-set. Members of both sets, up and core, are ranked according to their signal-to-noise-ratio and the n_{pa} variables with the highest value are selected as members of the core-set. After that competition, the variables not selected for the core-set are retired into a down-set that will gradually put their coefficients to zero. This two steps approach with gradual inclusion and exclusion is not too sensible to the noise and the choice of the priors. However, this methods heavily depends on the degree of freedom of the Wishart distribution and is limited to simultaneous links. Indeed, a covariance connection between two variables does not infer any lagged causal link between those same variables which is why we want to include Granger causal variables in each individual DLM.

7.2.2 Granger causality

Similarly to the simultaneous parents selection, we use an additional model to select the causal parents before including them into the individual DLM of each time series. Let us recall the method to select the causal relations used by the CGP-CCD model presented in Chapter 6. We fit to the environment of time series a causal graph model which regressed each time series on all the variables in the environment, including itself, with more than one time

lagged difference. In other words, it regresses the value of a time series at t on all the other variables at time $t-1$ up to $t-l$ with l the highest lagged considered. Thus, it tries to find lagged causal relations between the different time series in the environment under study and add a sparsity constraint on the number of relations to avoid noisy variables. Let us recall the expression of the causal graph process (CGP) model:

$$X(t) = w(t) + AX(t-1) + \cdots + \left(C_{M,0}I + \cdots + C_{M,M}A^M\right) X(t-M) \; .$$

Where A is the adjacency matrix, C coefficients, $X(t)$ the vector of time series value at t and M the number of lagged considered. For each time series we use the obtained sparse adjacency matrix A to select the n_{cgp} highest coefficients and include them in the individual DLM of each time series as lagged-one factors. We restrain the selection to one lag and a maximum of n_{cgp} variables due to the memory constraints in the implementation of the algorithm on GPUs. The CGP model used in the CGP-CCD algorithm to compute the adjacency matrix does not assume any endogenous variables or simultaneous connections, thus for each time series the model tries to explain all the changes with Granger causal relations.

The restriction to one-lag causality does not create a multivariate HCSDGLM model that includes all the endogenous variables, simultaneous and lagged exogenous connections. However, it is sufficient to test if the adding causal factors in addition to the simultaneous ones bring significant new information to the model. Another possible extension would be to look for causal relations assuming the endogenous and simultaneous connections of the HSGDLM into the CGP model.

7.3 Adding Granger Causality to HSGDLM

7.3.1 The extended model

The HSGDLM model presented in Chapter 4 assumes that each time series follow a DLM with endogenous and simultaneous exogenous coefficients. Let

us recall the expression of the DLM for the time series j at time t:

$$\begin{aligned} y_{j,t} &= F_{jt}\theta_{j,t} + \nu_{j,t} = H_{j,t}\phi_{j,t} + y_{sp_t(j),t}\gamma_{j,t} + \nu_{j,t}\,, \\ \theta_{j,t} &= G_{j,t}\theta_{j,t-1} + \omega_{j,t}\,, \\ \text{with:} \quad & \nu_{j,t} \sim \mathcal{N}(0, \lambda_{j,t}^{-1})\,, \\ & \omega_{j,t} \sim \mathcal{N}(0, W_{j,t})\,. \end{aligned}$$

Where $x_{j,t}$ represents the endogenous factors and $y_{sp_t(j),t}$ the selected exogenous factors. Recall that the vector of endogenous factors consists of previous RV and returns' values:

$$\begin{aligned} H_{j,t} &= (1, RV_{j,t}^{d}, RV_{j,t}^{w}, RV_{j,t}^{m}, \\ & \quad r_{j,t}^{(d)+}, r_{j,t}^{(d)-}, r_{j,t}^{(w)+}, \\ & \quad r_{j,t}^{(w)-}, r_{j,t}^{(m)+}, r_{j,t}^{(m)-})^T\,. \end{aligned}$$

We want to include Granger causal coefficients of each time series into this DLM. let us denote by $gc_t(j)$ the parents selected for time series j by the CGP-CCD algorithm. With these parents we extend the multivariate DLM model of time series j to:

$$\begin{aligned} y_{j,t} &= F_{jt}\theta_{j,t} + \nu_{j,t} = H_{j,t}\phi_{j,t} + y_{sp_t(j),t}\gamma_{j,t} + y_{gc_t(j),t-1}\psi_{j,t} + \nu_{j,t}\,, \\ \theta_{j,t} &= G_{j,t}\theta_{j,t-1} + \omega_{j,t}\,, \\ \text{with:} \quad & \nu_{j,t} \sim \mathcal{N}(0, \lambda_{j,t}^{-1})\,, \\ & \omega_{j,t} \sim \mathcal{N}(0, W_{j,t})\,. \end{aligned}$$

Where now the states vector includes $\theta_{j,t} = (\phi_{j,t}, \gamma_{j,t}, \psi_{j,t})$. Following the study done in Chapter 6 of the CGP-CCD model applied on financial time-series we do not update the adjacency matrix at every time t. Instead, because we do not expect the structure of the market to change daily we only update the adjacency matrix every three months. Thus, the Granger causal (GC) parents at time t, $gc_t(j)$, represents the parents selected by the latest computed adjacency matrix. Furthermore, as explained in Section **??**, putting too many parameters in the regression may be detrimental to the

performance of the predictions. Hence, instead of including all the parents selected by the CGP-CCD algorithm, which could theoretically include all the time series in the environment, we restrict the maximum number of GC-parents to n_{cgp}. In practice, the value of the limit on the maximum number of simultenous and GC parents is determined by the available GPU memory.

7.3.2 Implementation issues

Compared to the HSGDLM model, adding the causal coefficients only represents one additional term in the DLM equation of each time series but, due to its the sparsity of the adjacency matrix it creates new implementation difficulties. First, considering the computation of the adjacency matrix with the CGP-CCD algorithm we have to take into account the missing data. Since we model causal relations between log realised variances, $log(RV)$, we run the CGP-CCD algorithm on $log(RV)$. When the underlying time series of prices is missing we put the RV value to zero. While putting the RV to zero is perfectly suited for the HSGDLM coefficients, for the CGP-CCD the computation of the denominator of the matrix update for $i > 1$ in Equation 8 becomes singular. Hence we add noise on the diagonal values of variables at zero to avoid the singularity and be able to inverse the matrix.

In the HSGDLM model we update the parents in the core-, up- and down-sets every Δ_T time steps and while any of these sets could theoretically be empty, in practice they rarely have any null values. Thus, the dense version of the state variables $\theta_{j,t}$ with dimension $m \times KnE$ that we use for the computations is dense with few null values. On the other hand, the GC coefficients are only updated every 3 months and the sparsity constraint on the adjacency matrix can push any time series to have zero causal coefficients. Thus, the dense version of the state variable $\theta_{j,t}$ has dimension $m \times KnE + \times n_{cgp}$ with potentially many zero values for the causal coefficients. Those zero values appear since each time series may have between zero and n_{cgp} causal relations. These added zeros makes the covariance matrix of the state variable singular. Thus we follow the methodology describe in Chapter 8 to avoid this singularity. Before running the computation we add noise on the diagonal in

places with a zero value and put the values of the resulting matrix back to zero after the computation.

The range of acceptable values for the parameter d_j has a big impact on the quality of the results. Its value is directly linked to the number of parents, hence when starting with no CGP relation but simultaneous parents the value of d_j should reflect that and be around $nE + K$. While when CGP relations exist it increases to around $nE + K + n_{cgp}$. Thus compared to the HSDLM implementation we let this variable take a wider range of values inside the bound $nE + K + n_{cgp}$ plus or minus 0.2.

7.4 Results

7.4.1 The datasets

In this section we use the same datasets as in Chapter 4. Let us recall that the dataset of European stocks consists of the 487 most liquid stocks selected from thousands of stocks in different European markets. We use data from 2000 but since not all stocks have such a long trading history we assume $log(RV)$ to be zero when the price is missing. On the other hand, for the US dataset we started from the 500 stocks in the S&P500 in June 2019 and selected the 376 members that have a price history since 2000. A key difference between these two datasets is that the European dataset mixes stocks from different stock exchanges from different countries in europe trading in different currencies while the US dataset only contains stocks trading in the sape stock exchange in the same currency.

The memory available on the GPU only allows us to perform 500 Monte Carlo samples. The realised variance is computed with an exponential moving average on the return for which we use a parameter $\rho = 0.98$ and a time window of $rv_L = 40$ days. Since the CGP-CCD algorithm needs a minimum of 3 years of data to compute the adjacency matrix we study the results for the time window between 02/01/2006 and 04/06/2019.

We fix the number of allowed parents for the core set and causal parents to $n_{cgp} = nCore = 5$. While the cgp parents do not have any down set, for the simultaneous parents we use an update interval of $\Delta T = 10$ time steps so the core set will need a minimum of 10 days to be fully updated. Hence, the complete model we run has 10 endogenous factors and 10 exogenous factors, when only considering the core-set.

7.4.2 Direct predictions

We follow the same methodology as presented in Chapter 4 to assess the performance of our algorithm with causal factors. While we are interested by the prediction performance of the change point algorithm applied on the time-series of the coefficients of the DLM we assess the quality of the fit of the multivariate DLM by looking at the direct one step ahead inference. Table 7.1 shows the results of the one-step ahead prediction errors for both the HSGDLM model of Chapter 4 and the new CHSGDLM model with causal coefficient proposed in this chapter. The metrics in this table represent the average prediction performance over all the time series in the environment under study. Tables 7.3a and 7.3b show the percentage of observed value of $log(RV)$ that were in the predicted confidence interval. To assess the consistency of the predictions we look at the prediction quality for moves of different sizes. We can observe in Tables 7.3a and 7.3b that while the new model with causal coefficient shows slightly worse performance for very small moves it outperforms the HSGDLM model consistently for any move size with thresholds, both positive and negatives.

7.4.3 Evolution of the coefficients

The results in Table 7.1 shows that the new model with causal inference fits the data better than the HSGDLM model. However, as detailed in Chapter 4 we use the time series of the coefficients as a new feature to predict the evolution of the underlying time series of realised variance for up to a month in advance. Let us recall the change point algorithm methodology. The CHSGDLM sequentially updates the parameters of each individual DLM

Table 7.1: Performance metrics of the one-day-ahead inference obtained for 487 European stocks.

Model	HSGDLM	C-HSGDLM
Median ADV	0.014	0.014
Median RMSE	0.044	0.044
Median R^2	0.987	0.987
M-Z coefficient	0.990	0.990

(a) One day ahead inference comparison between different models. Median ADV corresponds to the Median Absolute deviation. Median RMSE is the median of the Root Mean Squared Errors. M-Z coefficient corresponds to the Minor-Zarnowitz regression coefficient, i.e. the regression of the observed values on the predicted ones.

Table 7.2: Performance metrics of one-day-ahead inference obtained for 500 European stocks.

Model		HSGDLM	C-HSGDLM
without threshold	Interval size	3.94%	3.94%
	% in	62.41%	59.30%
with mean$(\|r\|) = 5.74\%$	Interval size	10.56%	10.57%
	% in	61.84%	62.02%
with mean$(\|r\|) = 7.57\%$	Interval size	14.25%	14.25%
	% in	62.62%	63.16%
with mean$(\|r\|) = 9.32\%$	Interval size	18.03%	18.02%
	% in	63.81%	64.57%

(a) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for the selected absolute-returns above a certain threshold. The first columns shows the average absolute-return, mean$(|r|)$, of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean$(|r|)$. This table shows the ability of our model to predict large, positive and negative, moves of any size.

Model		HSGDLM	C-HSGDLM
with mean$(r) = 6.93\%$	Interval size	12.44%	12.44%
	% in	60.36%	60.31%
with mean$(r) = 8.50\%$	Interval size	16.16%	16.16%
	% in	62.76%	63.13%
with mean$(r) = 10.06\%$	Interval size	19.57%	19.58%
	% in	63.68%	64.42%

(b) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for selected returns above a threshold. The selection corresponds to only positive moves. The first columns shows the average return, mean(r), of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean(r). This table highlights the ability of our model to predict large upward moves of any size.

with every new data point which creates a time series of the coefficients value for each factor. Instead of studying the evolution of each individual coefficient separately we group them in different clusters representing a specific economic factor. The endogenous group is divided into two subgroups one including all the coefficients of the RV variables, rv, and the second the leverage effect coefficients, r. The exogenous group is divided into two subgroups one representing the coefficients of the causal connections, cgp, and the second the simultaneous coefficients, $core$.

To observe how the market evolve we estimate the variance of the market as the average of all the individual $log(RV)$ at each time t. Similarly, we estimate the value of each group as the average of that group value for each stock at time t. To visualise how these groups evolve with time compared to the market we rescale them and plot them on the same x-axis in Figure 7.1. In Figure 7.1 the groups are on the left axis while the variance of the market is in red and on the right axis. The cyclical pattern of the cgp group is due to the update process of the causal variables. Let us recall that the causal variables are updated every 3 months, do not go through the same up and down sets at the simultaneous parents and due to the sparsity enforced by the CGP-CCD algorithm presented in Chapter 6 the number of parents varies between 0 and n_{cgp}. Interestingly, we can observe that the group the most impacted by the addition of the causal variables in the leverage effect one r and in second place the realised variance one rv. Hence, from this figure it seems that the addition of the causal variables did not impact the values of the other exogenous factors but impacted the endogenous ones. To study these relationships in more details we will use the change point algorithms as presented in Chapter 4.

7.4.4 Change point algorithm

Let us recall the change point algorithm we presented in Chapter 4. We want to use the time series of each group of coefficients rv, r, cgp and $core$ to predict the evolution of the underlying variance for the day, week or month ahead. We do so by looking at the connection between the evolution of

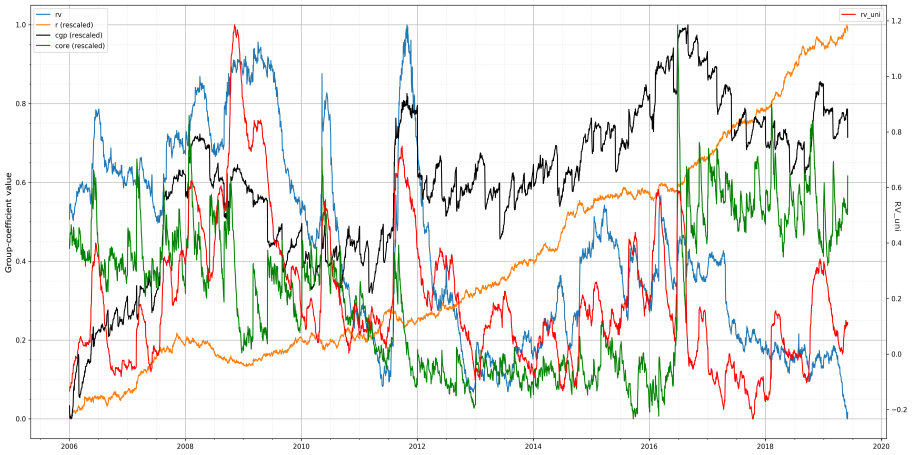


Figure 7.1: Plots computed with data from 500 European stocks. The graphic shows the evolution of the different groups (endogenous, with RV and R, and exogenous, denoted Core) rescaled to be on the same $y-axis$, which is left on the figure. The right axis corresponds to the approximated $log(RV)$ of the market, i.e. an equal weighted average of the individual $log(RV)$. This graphic highlights the connections between the behaviour of the market, represented here by $log(RV)$, and the different groups: RV, R and core. For example, looking at the volatility jump of the financial crisis of 2008 the RV and Core group seem positively correlated with the market while the leverage group seems negatively correlated. What this graphic cannot show is if these connections have any predictive power over the volatility of the market or if they simply react to it with a time lag, Figure 7.2 shows this predictive relation.

the time series of the group and of the underlying $log(RV)$. To study the evolution of a time series we approximate the derivative by the difference between two different measures separated by a specified number of time steps. More precisely, to predict a weekly move, i.e 5 time steps ahead, at each time t we look at the sign of the derivative between the value of the group at time $t-1$ and the value of that group at time $t-11$ and use the sign of that derivative to predict if the variance is going to increase or decrease between t and $t+5$. In that previous example the derivative used a parameter of 10 time steps but we will actually find the best value for that parameter for each group by studying the first 80% dates of the dataset. To avoid overfitting and keep values that have a clear economic interpretation we only allow the parameter of the derivative to be a multiple of 5 days between 1 day and 20 days. Thus, we predict the evolution of the variance for the next day, week or month by looking at the previous daily, weekly, bi-weekly or monthly evolution of the group of coefficients.

Using this methodology, each group of coefficient performs a prediction on the evolution of the variance for five different time scales: 1, 2, 5, 10 and 20 days. We will study the prediction of each group separately but we start by looking at the prediction quality of all the groups combined. We perform a backtest in which we select the position at each time t by summing the signals of all the groups and of the direct one-step ahead prediction. We can take three different positions: buy, sell or no trade. Then, to assess the quality of the predictions for all the stocks in our environment we compute an equally weighted portfolio (EW). Figure 7.2 shows the performance of the EW using both the signals from the groups and the direct inference in blue, using only the direct inference in yellow and using only the groups in green. It is interesting to observe on this graph the linearity of the performance. That linearity shows the capacity of our algorithm to adapt to the different market conditions which is only made possible thanks to the sequential updates by the Kalman filter.

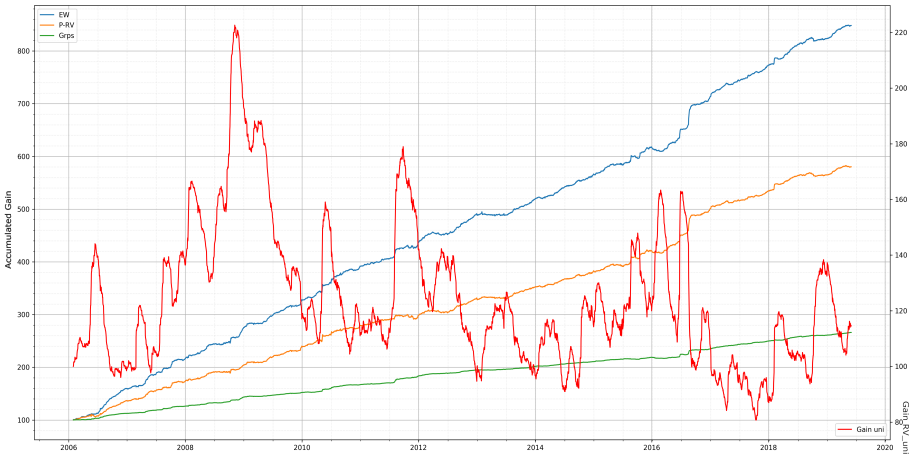


Figure 7.2: Plots computed with data from 500 European stocks. The orange line shows the performance of an EW portfolio built from the one-day-ahead inference. The green line corresponds to the performance of the EW portfolio using the signals from the change point algorithm applied on the different groups. The blue line shows the performance of the EW portfolio built by combining the one-day-ahead direct inference with the signals from the change points at different frequencies. The simulated portfolios correspond to equal weighted portfolios over the individual time series. In order to highlight the constance through time of the quality of the inference we added in red and on the right Y-axis the evolution of the underlying $log(RV)$ of the market.

Let us now study the evolution of the group of causal coefficients. In Figure 7.3 we plot the results of the EW portfolios obtained for each time scale. Each time scale corresponds to the length of the time window the algorithm forecasts, e.g. $lag20$ means that at each time t we forecast the evolution of the variance for the next 20 trading days. Thus, this figure is interesting to show the performance consistency in the prediction throughout the different time lags. Except for $lag1$, i.e. one day ahead prediction, the forecasts quality decreases for all the time lags in two market conditions: year 2008 and year 2016 which are known for their unusual volatility moves. However, when combined with the other groups the addition of the causal coefficient actually improves the predictions of all groups combined even during these years.

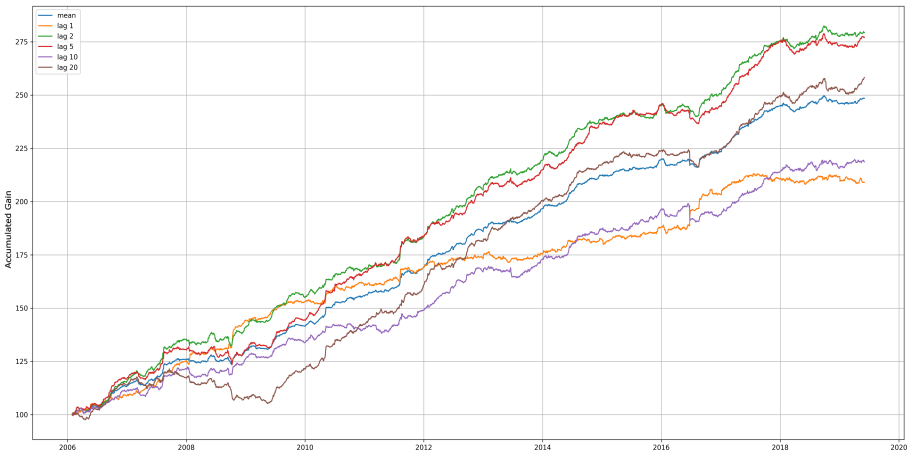


Figure 7.3: Plots computed with data from 500 European stocks. This figure plots the performance of the EW portfolio computed with the signals from the group of CGP coefficients created with the change points at different frequencies. This graphics shows that while the portfolios using signals updated at lower frequencies than daily are more volatile they still have a linear performance through time and small draw-downs.

We plot the performance of the EW portfolio of the difference between exogenous and endogenous groups for the different time lags for the CHSGDLM in Figure 7.5 and HSGDLM models in Figure 7.4. Recall that in Chapter 4 we answered the question: is the variance increasing when the spread between the exogenous and endogenous variables increases or decreases? To answer that question we computed the EW portfolio for the different lags using

for the signal the difference between the exogenous and endogenous groups. Figure 7.4 shows the results for the HSGDLM model. In this figure we can observe that an increase in the spread *(exogenous coefficients - endogenous coefficients)* does predict an increase in the underlying volatility. However, that connections does not seem as strong between 2011 and 2017 than the other years. On the other hand, Figure 7.5 shows the results for the same EW portfolios but this time using the groups from the CHSGDLM model that include causal coefficient. With these new variables the predictions perform much better during that time period between 2011 and 2017, hence including the causal coefficients improves the quality of the signal *(exogenous coefficients - endogenous coefficients)*.

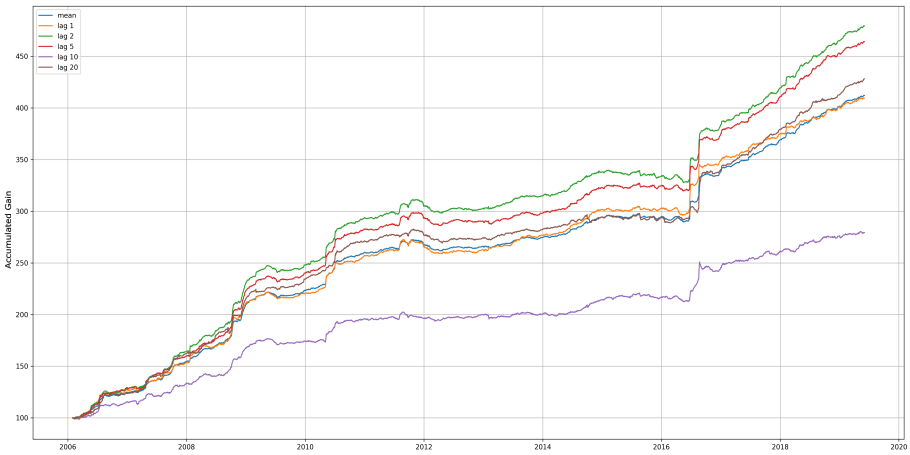


Figure 7.4: Plots computed with data from 500 European stocks with the HSGDLM model, i.e. without causal coefficients.. The figure shows the performance of the EW portfolio computed with the signals from the difference *(exogenous coefficients - endogenous coefficients)* using the change points algorithm at different frequencies.

7.4.5 Results on stocks from the S&P500

In this section we study the results obtain on the 376 stocks from the S&P500. Recall from Chapter 4 that in this environment with all the stocks from the same index the SGDLM model performs much better than the HAR-RV model. This observation confirmed our intuitions that the high number of connections between stocks from the same index makes the simultaneous con-

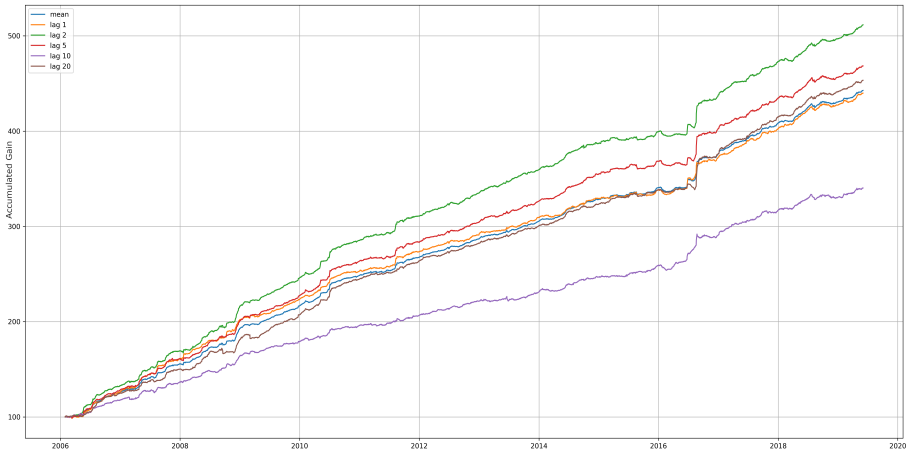


Figure 7.5: Plots computed with data from 500 European stocks. The figure shows the performance of the EW portfolio computed with the signals from the difference *(exogenous coefficients - endogenous coefficients)* using the change points algorithm at different frequencies. Hence those plots show the positive correlation between the difference *(exogenous coefficients - endogenous coefficients)* variables and the evolution of the volatility. In other words, when the volatility becomes more driven by exogenous factors than endogenous ones the volatility increases and vice-versa.

nections seem to be the main driver of the moves at the daily-frequency or lower. However, in the HSGDLM, adding the HAR coefficients did improve the performance for every move size, especially the small ones, and the endogenous coefficients improve the performance of the predictions using the change point algorithm.

In Tables 7.4 and 7.5 we compare the one step ahead prediction performance of the HSGDLM and the CHSGDLM models. Surprisingly, adding the causal variables decreases the performance of the HSGDLM model for small moves, but it does improve the results for large moves in both directions. Looking in more details on the performance of each group, the addition of the CGP variables seems to have impacted the group of RV variables the most. In the HSGDLM these coefficients from the HAR models were responsible for the performance of the model on small moves, hence that decrease in importance had a detrimental effect on the small moves in the CHSGDLM.

Let us recall results from Chapter 4. The performance of the EW portfolio using change point algorithm on the difference between exogenous and

endogenous variables was constant through time with no draw-downs or non-performing periods. Hence, while the addition of the causal coefficients did not improve the performance of the EW portfolio, it did lower its variance.

The difference in performance between the EU and US datasets of adding causal variables may be linked to the different trading behaviours. Indeed, since all the stocks come from the same stock exchange and we have a survival bias in the selection of the 376 stocks those are highly traded stocks. Hence, in a market with so many traders causal relations are more likely to appears in intraday behaviour. While in the more complex European dataset, causal relations between stocks in different stocks exchange trading in different currencies with different regulations are more likely to appear at lower frequencies.

7.5 Conclusion

In this chapter we added causal coefficients computed with the CGP-CCD algorithm of Chapter 6 to the HSGDLM model of Chapter 4. While the addition of causal coefficients in the multivariate DLM is straightforward the difficulty arises in the added instability of the computations due to thie new variables.

We show the impact of these coefficients on both US and EU datasets used to assess the performance of the HSGDLM model. While the impact of the causal coefficients is limited on stocks from the S&P500 on the European environment it improves on the performance of the previous model without causal variables. The one day ahead performance is better for any move size. The backtest with the change point algorithm is better both in terms of higher returns and lower variance. Furthermore, while the connection between the difference exogenous minus endogenous coefficients and the underlying volatility does not perform well between 2011 and 2016 for the HSGDLM model, the addition of the causal coefficient improves it. Indeed, with the causal coefficients the backtest using the difference *(exogenous coefficients - endogenous coefficients)* has a performance constant through time

with no non-performing years. Thus, adding causal variables increase the quality of the decomposition into endogenous and exogenous variables.

Table 7.4: Performance metrics of the one-day-ahead inference obtained for 376 stocks from the S&P500.

Model	HSGDLM	C-HSGDLM
Median ADV	0.015	0.016
Median RMSE	0.043	0.045
Median R^2	0.989	0.988
M-Z coefficient	0.996	0.994

(a) One day ahead inference comparison between different models. Median ADV corresponds to the Median Absolute deviation. Median RMSE is the median of the Root Mean Squared Errors. M-Z coefficient corresponds to the Minor-Zarnowitz regression coefficient, i.e. the regression of the observed values on the predicted ones.

Table 7.5: Performance metrics of one-day-ahead inference with thresholds obtained for 376 stocks from the S&P500.

Model		HSGDLM	C-HSGDLM
without threshold	Interval size	4.48%	4.52%
	% in	63.22%	59.31%
with mean($\|r\|$) = 5.80%	Interval size	10.22%	10.22%
	% in	61.27%	59.70%
with mean($\|r\|$) = 11.33%	Interval size	22.10%	22.10%
	% in	65.90%	65.55%
with mean($\|r\|$) = 20.02%	Interval size	39.28%	39.28%
	% in	62.66%	63.75%

(a) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for the selected absolute-returns above a certain threshold. The first columns shows the average absolute-return, mean($|r|$), of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean($|r|$). This table shows the ability of our model to predict large, positive and negative, moves of any size.

Model		HSGDLM	C-HSGDLM
with mean(r) = 7.04%	Interval size	13.13%	13.13%
	% in	63.89%	63.36%
with mean(r) = 12.01%	Interval size	23.04%	23.05%
	% in	64.70%	65.05%
with mean(r) = 20.34%	Interval size	40.00%	40.00%
	% in	63.31%	64.60%

(b) In order to compare the different models, we fixed a confidence interval size but allowed for different means in the predicted distribution. This table shows the percentage of measured $log(RV)$ which were in the predicted confidence interval for selected returns above a threshold. The selection corresponds to only positive moves. The first columns shows the average return, mean(r), of those selected points. The confidence intervals are selected to be smaller than twice the average-return of the selected points, here mean(r). This table highlights the ability of our model to predict large upward moves of any size.

8

Implementation on GPU

8.1 Introduction

We are interested by Graphical Processing Units (GPU) for their capacity to run thousands of threads in parallel. When a CPU has less than 50 different cores but running at very high frequency, a GPU will have thousands of cores but running at a much lower frequency. This hardware difference makes each processing unit better suited for different types of computations and thus an algorithm has to be optimised for each type of processing unit.

The HSGDLM model introduced in Chapter 4 models both endogenous and exogenous factors of all the time series in an environment with individual DLMs. Thus, the size of the environment we can study is limited by the computational-time required to study each time-series. More precisely, since each time series is modelled by a DLM with multiple hidden states, the update of the global multivariate distribution requires the update of parameters for each multivariate DLM, which requires Monte-Carlo samples. Thus, the ability to compute thousands of threads in parallel allows us to implement more complex multivariate systems with the aim of discovering new hidden

connections between the time series.

While GPUs are very attractive for time series analysis they require each algorithm developed for a CPU to be adapted for the GPU architecture. Because CPU and GPU have different level of memories and group of threads, without any optimisation of the algorithm the GPU would not bring any speed-up. Section 8.2 details the architecture differences, how to adapt the Cholesky decomposition and the sampling of a multivariate-Normal to a GPU. The two main GPU brands are AMD and Nvidia, and they both have their own math-computation libraries called OpenCL and Cuda respectively. OpenCL adapts to the available hardware and thus a code in OpenCL can run on both CPU and GPU. On the other hand, Cuda only works for Nvidia GPU but since it was launched first it is supported by a higher number of open-source ML Python libraries including the two most common ones, PyTorch and TensorFlow, which we use in this thesis.

We implemented the HSGDLM algorithm to run on a GPU with the library TensorFlow. However, the different steps to compute the multivariate DLM updates and predictions are not all well-suited for the fixed computational-graph structure used by TensorFlow. For example, some of the variables have dimensions that evolve with time when TensorFlow requires the size of the tensors to be fixed at the creation of the graph. Furthermore, the sparse-functions available on TensorFlow 1.14 do not have all the functionalities needed to run the HSGDLM algorithm. Hence, Sections 8.4 and 8.5 details the solutions we developed to implement the HSGDLM algorithm with TensorFlow.

8.2 How to optimise computation on a GPU

8.2.1 The GPU architecture

A GPU card consists of a fast memory and thousands of computation units called work-group each able to process K threads simultaneously. Since the

GPU memory is expensive it is often limited, and thus much of the GPU optimisation consists at minimising the amount of memory transfer between the GPU card and the RAM. When a CPU chip consists of a few cores running at very high frequency and able to process thousands of threads simultaneously; a GPU card has thousands of cores running at a lower frequency and each able to process tens or hundreds of threads simultaneously. Hence, the GPU is best suited for highly parallelisable processes which use simple computations.

Since a GPU card consists of thousands of work-groups able to process small vectors, to get the best performance out of the card the algorithm should use batched computation with each batch size matching the work-group size of the card and use as many work-groups as possible. If a large computation is divided in batch processes to run on the GPU, once the computations are done all the threads must be synchronised and combined on the GPU memory before starting the next step. This synchronisation is important to avoid one core being useless while waiting for the other core to finish its computation. Hence, an algorithm that optimises the computation on a GPU runs as many threads as work-groups and optimise the synchronisations steps. Due to this synchronisation, while the GPU card is very efficient at running thousands of processes in parallel it is not well-suited for conditional evaluation in the middle of a thread that will stop synchronise and re-distribute the processes.

8.2.2 Example with the cholesky decomposition

This section will detail how we can optimise the Cholesky decomposition algorithm for a GPU card. For a square positive-definite matrix A the Cholesky decomposition produces a lower triangular matrix L such that $A = LL^T$. The Cholesky-Banachiewicz or Cholesky-Crout algorithm starts from the upper-left corner of the matrix and proceeds row by row or column by column with

the following steps:

$$\forall j \quad L_{j,j} = \sqrt{A_{j,j} - \sum_{k=1}^{j-1} L_{j,k}^2}\,, \tag{8.1}$$

$$\forall i > j \quad L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} L_{j,k} \right)\,. \tag{8.2}$$

Following Dong et al. (2014), we can compute the Cholesky decomposition steps described in Equations 8.1 and 8.2 for a matrix A with dimensions (K, K) with the procedure detailed in Algorithm 2. However, this procedure only uses one thread. There are different ways to parallelize the procedure

Algorithm 2 Cholesky algorithm

```
1: procedure CHOLESKY(A)
2:     for k = 0 : K do                          ▷ loop on the K columns
3:         A_{k,k} = sqrt(A_{k,k})
4:         for i = k + 1; i < K; i++ do          ▷ loop on the K-k lines under the diagonal
5:             A_{i,k} = A_{i,k} / A_{k,k}
6:         end for
7:         for j = k + 1; j < K; j++ do          ▷ loop on the lower triangular matrix
8:             for i = j; i < K; i++ do
9:                 A_{i,j} = A_{i,j} − A_{i,k} A^T_{j,k}
10:            end for
11:        end for
12:    end for
13:    Select the lower triangular matrix
14: end procedure
```

of Algorithm 2: for step 3 we can compute the square root of the diagonal elements in parallel and for steps 5 and 9 we can process all the lines in the column under the diagonal in parallel. As described previously, the main difficulty in GPU computing is the memory management and work-group decomposition. Let us suppose that the size K of the matrix is less or equal than the maximum number of available processing units on the GPU and that the matrix A can fit on the local memory of the GPU. Then, we can adapt

the Cholesky decomposition by distributing the updates of steps 3, 5 and 9 over all the available threads. This implementation is detailed in Algorithm 3. If the algorithm does not have to compute a very large matrix but many smaller ones we can use batch computation to optimise the available tensor computation capabilities of each work-group.

Algorithm 3 Parallel Cholesky decomposition

1: **procedure** CHOLESKYGPU(A)
2: Load A to local memory
3: Each thread computes $A_{k,k} = \sqrt{A_{k,k}}$
4: **for** $k = 0 : K$ **do** ▷ loop on the N columns
5: Each thread computes $A_{i,k} = A_{i,k}/A_{k,k}$
6: **for** $j = k + 1; j < K; j + +$ **do**
7: Each thread computes $A_{i,j} = A_{i,j} - A_{i,k}A_{j,k}*$
8: **end for**
9: **end for**
10: Select the lower triangular matrix
11: **end procedure**

8.2.3 Example of batch computation on GPU for a multivariate Normal

In the recoupling step of the HSGDLM algorithm we draw multivariate random normal values for each DLM. Hence, we use this computation as an example on how to adapt an algorithm to run batch computations over the different work-groups of a GPU. To sample from a multivariate normal distribution we need random values sampled from a standard normal and the Cholesky decomposition of the covariance matrix. In our case we want to compute N_{MC} Monte-Carlo samples of a small multivariate normal distribution. Hence, if the size of each multivariate distribution is small enough to fit in one work-group we can compute it with the different threads of one work-group of the GPU and use each work-group for a different Monte-Carlo sample.

The recoupling step computes for each of the m time series N_{MC} Monte-

Carlo samples of a multivariate-normal distribution of size K. We want to distribute the computation of this step through $m \times N_{MC}$ different work-group processes each computing a Cholesky decomposition and the corresponding multivariate sample of size K. The GPU memory has to store the m different covariance matrices A^j of size $K \times K$, the m different mean-vectors μ^j of size K and the standard normal random values vectors $\mathcal{N}^j$ of size K. This batch computation works if the size K is smaller or equal than the maximum work-group size.

With these conditions on the dimensions, we can sample the standard normal values on the CPU before loading them and the added parameters on the GPU memory. Then, each work-group computes the Cholesky decomposition of the covariance matrix over its K threads followed by the matrix-vector multiplication and the vector-vector addition needed to obtain the sample from the multivariate normal. The implementation of the different steps is detailed in Algorithm 4. If the computation requires more memory to load

Algorithm 4 Batch computation for a multivariate normal algorithm

1: **procedure** MVNBATCH($A, \mu, \mathcal{N}$)
2: Load A to local memory
3: Each thread computes $A^j_{k,k} = \sqrt{A^j_{k,k}}$ ▷ Start Cholesky decomposition for A^j
4: **for** $k = 0 : K$ **do**
5: Each thread computes $A^j_{i,k} = A^j_{i,k}/A^j_{k,k}$
6: **for** $j = k + 1; j < K; j++$ **do**
7: Each thread computes $A^j_{i,j} = A^j_{i,j} - A^j_{i,k}A^j_{j,k}*$
8: **end for**
9: **end for**
10: Each thread computes a line of the calculus: $L^j\mathcal{N}^j + \mu^j$
11: **end procedure**

A, μ and $\mathcal{N}$ than the available GPU memory, then we need to divide the process in different groups. Furthermore, the split of K threads per work-group is possible as long as K does not exceed the maximum number of threads allowed per work-group. When K does not fit into one work-group

we have to use a block approach that consists into dividing the different computation into blocks of the corresponding sizes. The difficulty of this block approach is in the synchronisation of the computation of each block to reconstruct the full-size matrix. We will not detail the block approach since it is similar to the batch computation we just described. We will now describe the different libraries and functions we used. In practice we implemented the batch algorithm described in 4 but dealing with the handling of all the numerical errors and synchronisation steps to optimise the computations for any dimensions proved tedious. Instead we decided to use the TensorFlow and PyTorch libraries that have many functions already optimised for GPU computations.

8.3 The main libraries

8.3.1 PyTorch

PyTorch is a machine learning framework maintained by Facebook that can run computations on both CPU and GPU but only for Nvidia cards with the CUDAS library. This library was developed to be completely integrated into Python and thus adopts a very pythonic syntax that resemble the libraries such as Numpy or Scipy. Thus, for a person used to Numpy the learning curve to use this library is fast. The main characteristic of this library compared to TensorFlow is its use of dynamic computational graph. A computational graph is a graph where each node corresponds to a tensor or a computational step to evaluate. Using a dynamic computational graph means that the library build the computational graph dynamically on the go with each new statement and evaluation. This dynamic approach makes the code easier to debug and is very well suited to code dynamic model with dimensions evolving with the iterations. However, due to this dynamic approach issues in the graph might appear only when that corresponding part is ran, we have less control over the use of the memory since it evolves with the graph and as we have detailed previously, memory management is a key aspect of GPU computation.

8.3.2 Tensorflow

TensorFlow is a machine learning framework created and maintained by Google, which can run processes on either CPU, TPU or GPU but only for Nvidia cards with the CUDAS library. Contrary to PyTorch, at runtime TensorFlow will evaluate a static computational graph. A static computational graph means that the whole graph needs to be defined before any evaluation can take place. Thus, any error in the graph can be detected at the creation before any evaluation takes place, and the exact memory requirement is also know when the graph is defined. However, due to this static approach coding a model with dynamic dimensions takes more time and can waste resources since the highest amount of memory that might be needed during computation will have to be allocated at the creation of the graph. Furthermore, having the whole graph defined before the evaluation makes it more stable but harder to debug because its harder to evaluate specific expressions at runtime. At the same time, TensorFlow also comes with a visualisation library called TensorBoard which allows to visualise the static computational graph.

8.4 Computational issues

8.4.1 How to fit dynamic vectors into fix-size tensors

The advantage of the ML libraries that use tensors is to compute operations in parallel over the different dimensions of the tensor. For example, if we have two tensors with 3 dimensions consisting of a list of matrices we could compute the matrix-matrix multiplication directly multiplying the tensors together without having to loop over the list of matrices. The ML libraries will load these tensors on the GPU memory and optimise the computations according to the specifications of the available GPU card, i.e. size of the work-groups, number of cores, etc.

In addition, ML libraries such as PyTorch and TensorFlow can handle complex computations such as Cholesky decomposition on multi-dimensional ten-

sors such as a list of a list of matrices. However, to optimise the parallel computations on the tensors, all the matrices in the list need to have the same dimensions in order to fit in a single Tensor otherwise a list of matrices with different dimensions would require the use of a loop on the list on which the libraries cannot run parallel computations. Thus, in order to allow for variable dimensions we fix the dimensions of the tensor to the maximum one that can arise during the computation and fill the extra space with zeros. While this solution works for simple computations such as matrix-matrix multiplication, a Cholesky decomposition will fail due to the added zeros since that can make the matrix singular.

The HSGDLM algorithm requires many Cholesky decompositions to sample from a multivariate normal distribution or compute the inverse of a matrix. Since a matrix can be non-singular if any combination of sub-matrices which includes the diagonal is non-singular we replace the diagonal elements that have a null value with a very low non-zero value. Adding non-zero values on the diagonal makes the matrix non-singular and choosing a low value minimises the added bias on the result. In addition, since we know where we added the noise we can enforce zero values in the corresponding indices of the tensor resulting from the computation. The added zero allows us to implement vectors with evolving size into fixed-size tensors.

8.4.2 Positive semi-definiteness

In the HSGDLM algorithm the computation of the Cholesky decomposition to sample from the multivariate-Normal distribution or perform a matrix inverse is the step the most prone to numerical errors. If the computational time was not an issue, for each matrix we would study the eigenvalues and repair the matrices that have negative eigenvalues. In practice, this solution is too expensive and since we work with tensors of more than 4 dimensions combining thousands of matrices, running a test on each matrix individually would be too expensive. Hence, we developed a series of verification steps to guarantee the positive semi-definiteness of the matrix and compensate for the added bias. The first verification is to look for any negative diagonal

elements and use its absolute value instead.

To find fast-to-compute measures of singularity we performed a study on a sample of singular and non-singular matrices obtained from the HSGDLM algorithm computed on a real dataset. On this sample we observed that the Cholesky decomposition would not converge on matrices where the absolute sum of the covariance elements outside the diagonal was higher than the sum of the variances or where the ratio between these two sums was higher than a threshold. We then used different samples to test this observation and realised that the ratio of the two sum was more reliable than the difference. Thus, we implemented the following detection and repair procedure: compute the ratio of the sum of the absolute value of the covariance elements, i.e. outside the diagonal, over the sum of the variances; if the ratio is above 3.4 perform a matrix shrinkage that divides the covariance elements by 5 and increase the variances by 25%. While this procedure is not as precise as studying the eigen-decomposition it is very fast to compute and in practice significantly decreased the number of singular matrices.

However, due to the added noise and numerical errors, even with the previous correction-step some matrix will still be singular. Since running a computation on the tensor that fails is cheaper than running tests on each matrices separately we try to compute the Cholesky decomposition over the tensor and catch the exception in case it does not converge. If the decomposition does not converge we perform a matrix shrinkage step that does a weighted sum of that matrix and a diagonal matrix with only the variance on the diagonal. For example, with a weight of 0.2 the resulting matrix consists of the addition of 20% of the complete matrix with 80% of the diagonal matrix. In addition, since the Cholesky decomposition often fails when the diagonal elements are too small compare to the rest of the matrix we add noise to the diagonal. In practice we used a noise value of 0.001. Compare to the correction-step described in the previous paragraph, this matrix shrinkage is performed over the whole tensor, hence also adding bias to the non-singular matrices that are part of that tensor. If after this second matrix shrinkage step the Cholesky decomposition does still not converge we perform a third

transformation. The last transformation step selects the variances and approximates the covariance matrices to only a diagonal matrix with non-zero variances on the diagonal plus the added noise of 0.001. Since the resulting tensor is diagonal with strictly positive values on the diagonal it is non-singular however this method deletes any covariance elements and thus adds a significant bias on the results.

8.5 Sparse computation in TensorFlow

8.5.1 Why use sparse computations

Instead of saving the value of every elements, for a sparse matrix we can store only the non-zero values and their corresponding indices. For a dense matrix this storage method adds a significant amount of overheads and is thus less efficient then the classic one. However, the added overhead of computing such sparse computation brings measurable benefits when the sparsity of the matrix is above a threshold depending on the sparse library used. For example, with the Scipy-sparse library the tipping point for a matrix-matrix or matrix-vector multiplication is a density level below 1%, i.e. with less then 1% of non-zeros elements. Thus, when a matrix is guaranteed to have a sparsity level which will always stay higher than a specific threshold it is worth using a dedicated sparse-library.

In addition, if the available memory of the system is restricted, such as on a GPU card, even if the sparse-computations are not faster, storing the matrix in a sparse format reduces the required memory. Unfortunately, at the time of writing our code the sparse libraries of PyTorch and TensorFlow did not have all the sparse-functions we needed to implement the HSGDLM model.

We coded the HSGDLM in Python using the library TensorFlow 1.14 which has sparse functions to store tensors in a sparse format but few sparse computations are available. The library has functions to compute matrix-matrix multiplications optimised if only one of the two matrices is sparse. According

to the official documentation, the breakeven for using the sparse matrix multiplication function versus the dense version is more than 30% of zero values in the sparse matrix. At the time of coding, the only operation available for two sparse matrices was the addition. Thus, to compute the multiplication between two sparse matrices the library converts those matrices to dense and uses the dense matrix multiply to perform the operation. Due to this added step, using the sparse library for matrix-matrix multiplications has a benefit in terms of memory requirements more than computation-time.

8.5.2 Constructing the indices

Since the available ML libraries did not have the needed sparse-functions available at the time of coding we developed specific sparse-functions to implement the HSGDLM algorithm. A sparse matrix consists of a tensor of indices and a tensor of values for each of those indices. However, due to the specific format of some of the tensors instead of saving the full-size tensor of indices we can compute the indices from a dense generative vector. Thus, we only have to save the non-zero values and a dense version of the indices from which we can compute the full-size tensor. When a computation requires the dense version of the tensor we compute the indices of the non-zero elements and reconstruct the dense-tensor for that computation.

The TensorFlow library has two functions that allows use to do this conversion between sparse and dense formats: *scatter_nd*() and *gather_nd*(). *gather_nd*() selects from a dense tensor the elements at the specified indices and returns a vector of values. *scatter_nd*() does the opposite, it uses a tensor of indices and a vector of values to reconstruct a dense tensor with the values at the specified locations. For example, if we have a dense matrix of size 10×10 with a maximum of 3 non-zeros elements per line; we could store a smaller matrix of size 10×3 with a tensor of indices of the non-zero elements. The function *gather_nd*() will perform the transformation from the 10 matrix to the 10×3 one, while *scatter_nd*() will do the opposite.

Since the number of selected parents is dynamic it can vary from 0 to n_{pa}

which includes core, up and down sets. Thus, the number of non-zeros elements in the state vector θ_j of stock j can vary between nE and $nE + n_{pa}$. Most of the computations of the HSGDLM requires the sampling and update of the state vectors θ and of the first two moments of its distribution. If some elements are zeros, they will not affect the sampling or update of the parameters, thus we could use a version of the tensor with only the non-zeros elements for the computations. More precisely, θ is a tensor with dimension $m \times (nE + m)$, i.e. the number of time series times the number of time series plus the number of endogenous variables. For each time series, the non-zero elements of the state vector correspond to the endogenous variables and the selected parents. However, in practice there will never be more than n_{pa} selected parents and thus instead of storing and handling an $m \times (nE + m)$ tensor we can use an $m \times (nE + n_{pa})$ tensor for most of the computations. The only steps that require the full-size state-tensor is for the computation of the importance sampling weights α to compute the determinant and when performing the random walk update from θ_{t-1} to θ_t. For these two steps, with the indices of the selected parents we compute the indices of the non-zero elements of the dense tensor and thus reconstruct the full-size version only for the needed computations and store the smaller tensor and indices in-between.

A

Appendix for HSGDLM

A.1 SGDLM equations

A.1.1 Posterior and Recoupling

Each posterior follow a Normal-Gamma distribution $p(\theta_{j,t}, \lambda_{j,t}|\mathcal{D}_t) \sim \mathcal{NG}(m_{j,t}, C_{j,t}, n_{j,t}, s_{j,t})$

$$\begin{aligned} \theta_{j,t}|\lambda_{j,t}, \mathcal{D}_t &\sim \mathcal{N}\left(m_{j,t}, C_{j,t}/\left(s_{j,t-1}\lambda_{j,t}\right)\right) , \\ \lambda_{j,t}|\mathcal{D}_t &\sim \mathcal{G}\left(n_{j,t}/2, n_{j,t}s_{j,t-1}/2\right) . \end{aligned}$$

Hence with $y_{j,t}$ the observed data at t the parameters follow the classic Kalman filter update:

$$\begin{aligned}
e_{j,t} &= y_{j,t} - F_{j,t}^T a_{j,t}\,, \\
q_{j,t} &= s_{j,t-1} + F_{j,t}^T R_{j,t} F_{j,t}\,, \\
A_{j,t} &= R_{j,t} F_{j,t}/q_{j,t}\,, \\
z_{j,t} &= \left(r_{j,t} + e_{j,t}^2/q_{j,t}\right)/\left(r_{j,t}+1\right)\,, \\
m_{j,t} &= a_{j,t} + A_{j,t} e_{j,t}\,, \\
C_{j,t} &= \left(R_{j,t} - A_{j,t} A_{j,t}^T q_{j,t}\right) z_{j,t}\,, \\
n_{j,t} &= r_{j,t} + 1\,, \\
s_{j,t} &= z_{j,t} s_{j,t-1}\,.
\end{aligned}$$

A.1.2 Decoupling with Variational Bayes

The decoupling step approximate the multivariate posterior distribution by a product of Normal-Gamma distributions $p(\theta_{j,t}, \lambda_{j,t}|\mathcal{D}_t) \sim \mathcal{NG}(m_{j,t}, C_{j,t}, n_{j,t}, s_{j,t})$ whose parameters are obtain by minimising the Kullback-Leibler distance.

$$\begin{aligned}
m_{j,t} &= E[\lambda_{j,t}\theta_{j,t}]/E[\lambda_{j,t}]\,, \\
V_{j,t} &= E[\lambda_{j,t}\left(\theta_{j,t} - m_{j,t}\right)\left(\theta_{j,t} - m_{j,t}\right)^T]\,, \\
d_{j,t} &= E[\lambda_{j,t}\left(\theta_{j,t} - m_{j,t}\right)^T V_{j,t}^{-1}\left(\theta_{j,t} - m_{j,t}\right)]\,, \\
s_{j,t} &= \left(n_{j,t} + p_{j,t} - d_{j,t}\right)/\left(n_{j,t}E[\lambda_{j,t}]\right)\,, \\
C_{j,t} &= s_{j,t} V_{j,t}\,.
\end{aligned}$$

$n_{j,t}$ is updated through an optimisation step solving:

$$log(n_{j,t}+p_{j,t}-d_{j,t}) - \psi(n_{j,t}/2) - (p_{j,t}-d_{j,t})/n_{j,t} - log(2E[\lambda_{j,t}]) + E[log(\lambda_{j,t})] = 0\,.$$

A.1.3 Evolution to $t+1$

For the evolution from t to $t+1$ the states $\theta_{j,t}$ follow a random walk $\theta_{j,t+1} = G_{j,t+1}\theta_{j,t} + \omega_{j,t}$. The coefficient matrix $G_{j,t}$ is updated with the parents

down set values following $1-(\Delta_T+1-l)^{-1}$ for $l=1:\Delta_T$. $m_{j,t}$ is updated to include a value for the new members of the up-set. With $B_{j,t+1}=G_{j,t+1}C_{j,t}G_{j,t+1}^T$ the covariance matrix follow the following block discounting update:

$$W_{j,t+1}=\begin{pmatrix}\left(\frac{1}{\delta_{j,\phi}}-1\right)B_{j,t+1}[:n_E,:n_E] & \left(\frac{1}{\sqrt{\delta_{j,\phi}\delta j,\gamma}}-1\right)B_{j,t+1}[:n_E,n_E:] \\ 0 & \left(\frac{1}{\delta_{j,\gamma}}-1\right)B_{j,t+1}[n_E:,n_E:]\end{pmatrix}.$$

where n_E corresponds to the number of external variables, vs. parents ones, δ_ϕ the external variables factor and δ_γ the parental one. The filters' parameters follow:

$$\begin{aligned}a_{j,t+1} &= G_{j,t+1}m_{j,t+1}\,,\\ R_{j,t+1} &= B_{j,t+1}+W_{j,t+1}\,,\\ r_{j,t+1} &= \beta_j n_{j,t}\,.\end{aligned}$$

where this last equation represent the beta-stochastic-volatility model.

A.1.4 Prior inference

The prior for $t+1$ samples follow $p(\theta_{j,t+1},\lambda_{j,t+1}|\mathcal{D}_t)\sim\mathcal{NG}(a_{j,t+1},R_{j,t+1},r_{j,t+1},s_{j,t})$. Which are used to compute $\Gamma_{j,t+1}=[\theta_{j,t+1}]_{\forall j}$, $\Lambda_{j,t+1}=diag\left(\lambda_{0,t+1},\ldots,\lambda_{N,t+1}\right)$ and $\mu_{t+1}=x_{j,t}\phi_{j,t+1}$. Recall $\theta_{j,t}=(\phi_{j,t},\gamma_{j,t})^T$. Hence we obtain the mean and covariance matrix to sample y_{t+1}:

$$\begin{aligned}A_{t+1} &= \left(I-\Gamma_{t+1}\right)^{-1}\,,\\ \Sigma_{t+1} &= \left(\left(I-\Gamma_{t+1}\right)^T\Lambda_{t+1}\left(I-\Gamma_{t+1}\right)\right)^{-1}\,,\\ y_{t+1} &\sim \mathcal{N}\left(A_{t+1}\mu_{t+1},\Sigma_{t+1}\right)\,.\end{aligned}$$

References

Anderson, T., Bollerslev, T., Diebold, F., and Labys, P. (2000). Great realisations. *Risk*.

Banerjee, O., El Ghaoui, L., and d'Aspremont, A. (2007). Model Selection Through Sparse Maximum Likelihood Estimation. *arXiv.org*.

Borland, L. (1998). Microscopic dynamics of the nonlinear Fokker-Planck equation: A phenomenological model . *physical review*, 57.

Borland, L. (2002). A theory of non-Gaussian option pricing. *Quantitative Finance*.

Borland, L. (2004). A multi-time scale non-Gaussian model of stock returns. *arXiv.org*.

Borland, L. and Bouchaud, J.-P. (2005). On a multi-timescale statistical feedback model for volatility fluctuations. *arXiv.org*.

Borland, L., Bouchaud, J.-P., Muzy, J.-F., and Zumbach, G. (2005). The Dynamics of Financial Markets–Mandelbrot's multifractal cascades, and beyond. *arXiv.org*.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2008). Time series analysis: forecasting and control. John Wiley & Sons.

Box, G. E. P. and Tiao, G. C. (1977). A canonical analysis of multiple time series. *Biometrika*, 64(2):355–365.

Calvet, L. E. and Fisher, A. J. (2001). Forecasting multifractal volatility. *Journal of Econometrics*, 105(1):27–58.

Calvet, L. E. and Fisher, A. J. (2004). How to Forecast Long-Run Volatility: Regime Switching and the Estimation of Multifractal Processes. *Journal of Financial Econometrics*, 2(1):49–83.

Cartea, Á., Jaimungal, S., and Penalva, J. (2015). Algorithmic and high-frequency trading. Cambridge University Press.

Corsi, F. (2004). A Simple Long Memory Model of Realized Volatility. *SSRN Electronic Journal*.

Corsi, F., Mittnik, S., Pigorsch, C., and Pigorsch, U. (2008). The Volatility of Realized Volatility. *Econometric Reviews*, 27(1-3):46–78.

Corsi, F. and Reno, R. (2009). HAR volatility modelling with heterogeneous leverage and jumps. *Available at SSRN 1316953*.

Corsi, F., Zumbach, G., Müller, U. A., and Dacorogna, M. M. (2001). Consistent High-precision Volatility from High-frequency Data. *Economic Notes*, 30(2):183–204.

Cuturi, M. and d'Aspremont, A. (2013). Mean reversion with a variance threshold. In *International Conference on Machine Learning*.

Cuturi, M. and d'Aspremont, A. (2015). Mean-Reverting Portfolios: Tradeoffs Between Sparsity and Volatility. *arXiv.org*.

Dacorogna, M. M., Müller, U. A., Pictet, O. V., and Olsen, R. B. (1997). Modelling Short-Term Volatility with GARCH and HARCH Models. *SSRN Electronic Journal*.

d'Aspremont, A. (2010). Identifying small mean-reverting portfolios. *Quantitative Finance*, 11(3):351–364.

de Santa Helena, E. L., Nascimento, C. M., and Gerhardt, G. J. L. (2015). Alternative way to characterize a q-Gaussian distribution by a robust heavy tail measurement. *Physica A*, 435:44–50.

Diebold, F. X., Hickman, A., Inoue, A., and Schuermann, T. (1997). *Converting 1-day volatility to h-day volatility: Scaling by sqrt(n) is worse than you think*. F. Diebold.

Dong, T., Haidar, A., Tomov, S., and Dongarra, J. (2014). A Fast Batched Cholesky Factorization on a GPU. In *2014 43rd International Conference on Parallel Processing*, pages 432–440. IEEE.

Fama, E. F. and French, K. R. (1988). Permanent and temporary components of stock prices. *journals.uchicago.edu*, 96(2):246–273.

Fogarasi, N. and Levendovszky, J. (2013). Sparse, mean reverting portfolio selection using simulated annealing. *content.iospress.com*.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*.

Goudet, O., Kalainathan, D., Caillou, P., Guyon, I., Lopez-Paz, D., and Sebag, M. (2018). Causal Generative Neural Networks. *arxiv.org*.

Griveau-Billion, T. and Calderhead, B. (2019). A Dynamic Bayesian Model for Interpretable Decompositions of Market Behaviour. *arXiv.org*.

Griveau-Billion, T., Richard, J.-C., and Roncalli, T. (2013). A fast algorithm for computing high-dimensional risk parity portfolios. *ssrn*.

Gruber, L. F. and West, M. (2016a). Bayesian forecasting and scalable multivariate volatility analysis using simultaneous graphical dynamic models. *arXiv.org*.

Gruber, L. F. and West, M. (2016b). GPU-Accelerated Bayesian Learning and Forecasting in Simultaneous Graphical Dynamic Linear Models. *Bayesian Analysis*, 11(1):125–149.

Hull, J. C. (2017). The Greek Letters. In Limited, P. E., editor, *Options, Futures, and Other Derivatives*, pages 421–452.

Jensen, M. C. (1978). Some Anomalous Evidence Regarding Market Efficiency. *SSRN Electronic Journal*.

Johannes, M., Polson, N., and Stroud, J. (2005). Sequential parameter estimation in stochastic volatility models with jumps. *columbia.edu*.

Johansen, S. (2005). Cointegration: an overview. *ku.dk*, pages 1–40.

Jurek, J. W. and Yang, H. (2007). Dynamic Portfolio Selection in Arbitrage. *SSRN Electronic Journal*.

LeBaron, B. (2001). Stochastic volatility as a simple generator of apparent financial power laws and long memory. *Quantitative Finance*, 1(6):621–631.

Lynch, P. E. and Zumbach, G. (2003). Market heterogeneities and the causal structure of volatility. *Quantitative Finance*, 3(4):320–331.

Malkiel, B. G. (2003). The Efficient Market Hypothesis and Its Critics. *Journal of Economic Perspectives*, 17(1):59–82.

McAlinn, K. and West, M. (2016). Dynamic Bayesian Predictive Synthesis in Time Series Forecasting. *arXiv.org*.

Mei, J. and Moura, J. e. M. F. (2017). Signal processing on graphs: causal modeling of unstructured data. *IEEE Transactions on Signal Processing*, 65(8):2077–2092.

Mei, J. and Moura, J. M. F. (2015). Signal processing on graphs: Estimating the structure of a graph. In *ICASSP 2015 - 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5495–5499. IEEE.

Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics*, 34(3):1436–1462.

Müller, U. A., Dacorogna, M. M., Davé, R. D., Olsen, R. B., Pictet, O. V., and von Weizsäcker, J. E. (1997). Volatilities of different time resolutions — Analyzing the dynamics of market components. *Journal of Empirical Finance*, 4(2-3):213–239.

Muzy, J.-F., Delour, J., and Bacry, E. (2000). Modelling fluctuations of financial time series: from cascade process to stochastic volatility model. *arXiv.org*, (3):537–548.

Nakajima, J. (2009). Bayesian analysis of GARCH and stochastic volatility: modeling leverage, jumps and heavy-tails for financial time series. *duke.edu*.

Nakajima, J. (2014). Bayesian Analysis of Multivariate Stochastic Volatility with Skew Return Distribution. *Econometric Reviews*, 8:1–23.

Noureldin, D., Shephard, N., and Sheppard, K. (2011). Multivariate high-frequency-based volatility (HEAVY) models. *Journal of Applied Econometrics*, 27(6):907–933.

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference*. Foundations and learning algorithms.

Prado, R. and West, M. (2010). *Time series: modeling, computation, and inference*. Chapman & Hall Book CRC Press.

Serletis, A. and Rosenberg, A. A. (2009). Mean reversion in the US stock market. *Chaos, Solitons and Fractals*, 40(4):2007–2015.

Shojaie, A. and Michailidis, G. (2010a). Discovering graphical Granger causality using the truncating lasso penalty. *Bioinformatics*, 26(18):i517–i523.

Shojaie, A. and Michailidis, G. (2010b). Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. *Biometrika*, 97(3):519–538.

Sipos, I. R. and Levendovszky, J. (2013). Optimizing sparse mean reverting portfolios. *Algorithmic Finance*, 2(2):127–139.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494.

Wang, H. (2013). Coordinate descent algorithm for covariance graphical lasso. *Statistics and Computing*, 24(4):521–529.

Wu, T. T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2(1):224–244.

Yen, Y.-M. and Yen, T.-J. (2014). Solving norm constrained portfolio optimization via coordinate-wise descent algorithms. *Computational Statistics and Data Analysis*, 76:737–759.

Zhao, Z. Y., Xie, M., and West, M. (2016). Dynamic dependence networks: Financial time series forecasting and portfolio decisions. *Applied Stochastic Models in Business and Industry*, 32(3):311–332.

Zumbach, G. (2009). Time reversal invariance in finance. *Quantitative Finance*, 9(5):505–515.

Zumbach, G. and Lynch, P. (2001). Heterogeneous volatility cascade in financial markets. *Physica A: Statistical Mechanics and its Applications*, 298(3-4):521–529.