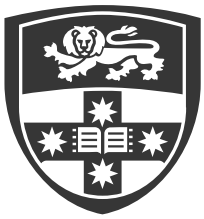

THE UNIVERSITY OF
SYDNEY

# Deep Learning Based Statistical Models for Business and Financial Data

by

**Trong Nghia Nguyen**

*A thesis submitted to fulfil requirements for the degree of*

**Doctor of Philosophy**

in

Discipline of Business Analytics, Business School
The University of Sydney

Nov 2021

**Supervisors:**
Associate Professor Minh-Ngoc Tran ,
Discipline of Business Analytics,
The University of Sydney Business School

Scientia Professor Robert Kohn,
School of Economics,
The University of New South Wales Business School

# Contents

# Abstract

Deep learning refers to classes of machine learning models using highly sophisticated neural network-based structures to efficiently learn non-linear effects normally exhibited in many types of data such as images, text, video, etc. Deep learning models have been widely used in industry level and empirically proven to work amazingly well in a wide range of machine learning and artificially intelligent applications such as object detection, face recognition, text translation, etc. In recent years, a significant amount of academic research have used advanced machine learning techniques, including deep learning models, as attractive alternatives to traditional statistical approaches in the financial econometrics literature. While there are some interesting works showing promising results to support this line of research, some recent studies extensively comparing the predictive performance of machine learning and financial econometrics models suggest an opposite but unsurprising conclusion: purely advanced machine learning techniques do not outperform traditional statistical models in time series forecasting but their computational requirements are considerably greater than those of statistical methods. Additionally, these advanced models are often called black-box techniques because of their low level of interpretability. This reasonably explains the overall low usage of the stage-of-the-art machine learning methods in real business and financial applications, despite of their undeniably impressive predictive performance in many highly sophisticated learning tasks.

The lack of applicability of deep learning models in financial econometrics applications is mainly due to the fact that they are not originally designed to learn stylized facts naturally existing in business and financial data. To compensate for this shortage of the model development, these engineer-oriented models in general require a substantial amount of representative training data to be able to learn distinct patterns about the data, and inherently ignore the interpretation of model outcomes. These advanced models can make highly accurate prediction when training data is sufficient but when the interpretation of the models is also one of the main interests, they are still not the first choice for econometricians, who are often interested in how to interpret not only the forecast quantities and but also the in-sample fitting. The statistical and financial econometrics models, on the other hand, are developed to

learn the characterized stylized facts directly and hence well-explain the underlying data generating process. Consequently, these models are highly interpretable and have good in-sample performance. However, due to the simplicity of model designs, these statistical models normally perform badly when the underlying data processes exhibit non-linear effects, which cannot be captured efficiently by simple linear structures mainly developed to explain distinct behaviors of business and financial data.

In this dissertation, we investigate a wide range of statistical models commonly used in many business and financial econometrics applications, and propose flexible ways to combine these highly interpretable models with powerful predictive models in the deep learning literature to leverage the advantages and compensate the disadvantages of each of the modeling approaches. Our methodologies of utilizing deep learning techniques for financial data are different from the recently proposed deep learning based models in the financial econometrics literature in several perspectives. First, we do not overlook well-established structures that have been successfully used in statistical modeling. More specifically, we flexibly incorporate deep learning techniques to the statistical model to capture the data effects that cannot be explained by the simple linear components of those models. Our proposed modeling frameworks therefore normally include two components: a linear component to explain linear dependencies and a deep learning based component to capture data effects rather than linearity possibly exhibited in the underlying process. Second, we do not use the neural network structures in the same fashion as they are implemented in the deep learning literature, but modify those black-box methods to make them more explainable and hence improve the interpretability of the proposed models. As the results, our hybrid models not only perform better than the pure deep learning techniques in term of interpretation but also often produce more accurate out-of-sample forecasts than the counterpart statistical frameworks. Third, we propose advanced *Bayesian* inference methodologies to efficiently quantify the uncertainty about the model estimation and prediction, which are of high interest to statisticians and econometricians. For the proposed high-dimensional deep learning based models, performing efficient Bayesian inference is extremely challenging and is often ignored in the engineer-oriented papers, which generally prefer the *frequentist* estimation approaches, e.g. maximum likelihood, mainly due to the simplicity.

Finally, an important contribution of the dissertation is to provide an unified software package implementing all the proposed models and methodologies, which are

normally missing in statistical modeling research. We believe that an user-friendly software package will make our research works more accessible to researchers and practitioners in the fields of statistical and financial econometrics modeling. The software provided is completely written in Matlab with some parts having been converted to R and Python.

# Statement of Originality

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Sydney and where applicable, any partner institution responsible for the joint award of this degree.

The author acknowledges that copyright of published works contained within this dissertation resides with the copyright holder(s) of those works.

I give permission for the digital version of my dissertation to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University of Sydney to restrict access for a period of time.

# Acknowledgements

I would like to thank my supervisors, Associate Professor Minh-Ngoc Tran and Scientia Professor Robert Kohn, for their unconditional support, guidance and inspiration. It is my absolute pleasure to work with them over the last three and a half years and this dissertation would not be possible without them. The knowledge that I got from my supervisors is tremendous, and probably one of the most valuable assets I could have. Especially, I am sincerely grateful to be a student of A/Prof. Minh-Ngoc, who encouraged and inspired me to become an independent researcher and completely transformed me from inexperienced graduate student to be a passionate and dedicated researcher. His support goes far beyond teaching me the first concepts of Variational Bayes, correcting my erroneous paper drafts and giving me better advice than most people I have ever met. I will never forget our weekly coffee sessions on the fifth floor of the Business School when we discuss about research ideas, paper rejection and life. I have been incredibly lucky to have you as a supervisor and will always appreciate you allowing me to enjoy the freedom of working on my research ideas and always being there when I need some help, which is something not many PhD students seem to benefit from. Your lessons will definitely go with me to whatever career path I will follow in the future.

My gratitude is extended to my collaborators, including Professor David Nott, Dr. David Gunawan, Dr. Hoang Nguyen, Dr. TN-Lan Le and Mr. Viet-Hung Dao for theirs hard-works during our joined projects. It is my privilege to have opportunity to work with them. Their wisdom, dedication and responsibility undeniably play an important role in this dissertation.

My appreciation also goes to Australian Center of Excellence in Mathematical and Statistical Frontiers (ACEMS) and the University of Sydney Business School for awarding me with the prestigious post graduation award under an International Postgraduate Research Scholarship (IPRS) scheme. I will never forget the moment when my supervisors announced that I were being awarded the Postgraduate scholarship for pursuing the PhD at the University of Sydney Business School.

I would like also to thank all of my Vietnamese friends in Sydney, especially for our traditional Friday meetings which really enriched my research life and helped me get through the difficulties of academic life.

My special thanks go my dear parents, Trong-Man and Thi-Suong, my fiancé, Hannah, for always loving and encouraging me to be a better person. Their infinite and unconditional support undoubtedly helped me to complete my doctoral research.

# Dissertation Conventions

The following conventions have been adopted in this dissertation:

## Typesetting

This document was compiled using LaTeX2e. Texstudio 2.12.22 was used as a text editor interfaced to LaTeX2e. Inkscape 0.92.3 was used to produce schematic diagrams and other drawings.

## Spelling

Australian English spelling conventions have been used, as defined in the Macquarie English Dictionary—A. Delbridge (Ed.), Macquarie Library, North Ryde, NSW, Australia, 2001.

## Referencing

The Harvard reference style is used for referencing and citation in this dissertation.

## System of Units

The units comply with the international system of units recommended in an Australian Standard: AS ISO 1000-1998 (Standards Australia Committee ME/71, Quantities, Units and Conversions 1998).

# Acronyms

| | |
|---|---|
| **ARCH** | Autoregressive Conditional Heteroskedasticity |
| **BART** | Bayesian Additive Regression Trees |
| **GARCH** | Generalized Autoregressive Conditional Heteroskedasticity |
| **ARFIMA** | Autoregressive Fractionally Integrated Moving Average |
| **CPM** | Correlated Pseudo Marginal |
| **DT-SMC** | Density Tempered Sequential Monte Carlo |
| **DFNN** | Deep Feedforward Neural Network |
| **FNN** | Feedforward Neural Network |
| **FIGARCH** | Fractional Integrated GARCH |
| **GJR** | GJR GARCH |
| **GLM** | Generalized Linear Model |
| **GLMM** | Generalized Linear Mixed Model |
| **LMSV** | Long Memory Stochastic Volatility |
| **LSTM** | Long-Short Term Memory |
| **MCMC** | Markov Chain Monte Carlo |
| **NAGVAC** | NAtural gradient Gaussian Variational Approximation with factor Covariance |
| **PMMH** | Pseudo Marginal Metropolis Hasting |
| **SMC** | Sequential Monte Carlo |
| **SRU** | Statistical Recurrent Unit |
| **SR-SV** | Statistical Recurrent Stochastic Volatility |
| **SV** | Stochastic Volatility |
| **RECH** | REcurrent Conditional Heteroskedasticity |
| **RNN** | Recurrent Neural Network |
| **VAFC** | Variational Approximation with Factor Covariance |
| **VB** | Variational Bayes |

# Authorship attribution statement

Chapter 3 of this thesis is published as

M.-N. Tran, **T.-N. Nguyen**, D. Nott, and R. Kohn (2019). Bayesian Deep Net GLM and GLMM. *Journal of Computational and Graphical Statistics.* 29(1):97-113.

I implemented the algorithm, analysed the results, wrote some chapters and made the final software package. I contributed around 60% amount of work presented in the publication.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Trong-Nghia Nguyen

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Minh-Ngoc Tran

# Publications

## Journal Articles

[1] M.-N. Tran, **T.-N. Nguyen**, D. Nott, and R. Kohn (2019). Bayesian Deep Net GLM and GLMM. *Journal of Computational and Graphical Statistics.* 29(1):97-113 (Rank A*).

[2] H. Nguyen, **T.-N Nguyen**, and M.-N. Tran (2021). A dynamic leverage stochastic volatility model. *Applied Economics Letters.* 0(0):1-6 (Rank B).

## Conditionally Accepted

[1] **T.-N. Nguyen**, M.-N. Tran, and R. Kohn (2020). **Recurrent Conditional Heteroskedasticity**. *Journal of Applied Econometrics* (Rank A*).

## Under Revision

[1] **T.-N. Nguyen**, M.-N. Tran, D. Gunawan and R. Kohn (2020).**A Statistical Recurrent Stochastic Volatility Model for Stock Markets**. *Journal of Business & Economic Statistics* (Rank A*).

## Under Submission

[1] V.-H. Dao, **T.-N. Nguyen**, and M.-N. Tran (2021). **A practical tutorial on Variational Bayes**. *International Statistical Review* (Rank A).

[3] TN-Lan Le, **T.-N. Nguyen**, and M.-N. Tran (2020). **Recurrent transformation stochastic volatility model for currency exchange rates**. *Quantitative Finance* (Rank A).

# List of Figures

# List of Tables

# Chapter 1

# Introduction

---

This chapter introduces the motivation for the problems considered in this dissertation and discusses the challenging nature of these problems. A summary of the contributions made in the following chapters and an overview of the structural organisation of the dissertation are provided.

---

# 1.1 Introduction

The modern machine learning (ML) techniques significantly involve the use of neural network-based models, which is a class of ML methods imitating human neural functionalities to learn complex structures exhibited in many types of data such as image, text or video. A neural network in general is a collection of neurons separated by layers. More specifically, a network contains many layers, and each layer can have many neurons. Neurons are the main components of the networks to perform computational tasks; they receive the signal from the previous layers as input, then do specific calculations based on the tasks with the inputs and send the output to the next layers, or even feedback to the previous layers. The overall length of the chain of layers is called the *depth* of the models and the term *deep learning* is normally used to describe neural networks with high depth, e.g. have many layers. In this dissertation, we use the terms deep learning and neural network based models exchangeably as most of discussions are applicable to neural networks with arbitrary depth.

Neural network models in general and deep learning models in particular have great predictive performance, but they always come with several costs that significantly limit their applicability in many business and financial econometric applications: data consuming, computationally expensive, and poor interpretability. First, deep learning models normally have a substantial amount of parameters and hence proportionally require a large amount of data to estimate the model accurately. In many business applications where not much data is available, this is a serious issue as the fitted neural network models tend to be overfitting; these models do not forecast well the out-of-sample data. Second, when there is enough data to train the deep learning models, the highly computational cost comes as a by-product. For models with many parameters and there are a lot of data to learn, it requires sufficient power resource to quickly train and deploy the models, especially in real-time applications. The computationally expensive property of these high dimensional models is even more problematic when performing Bayesian inference is necessary to quantity the uncertainty about model parameters and forecast values, which is one of the main interests to statisticians and econometricians. Third, the deep learning models use sophisticated structures of neurons and layers to learn the complex properties of the data, e.g. non-linearity or long range dependency, and hence sacrifice the interpretation of the output estimates as a trade-off. For business applications where

the interpretation of predictions is important, the use of these powerful models is still very limited, regardless the impressive predictive performance they can achieve.

A large body of academic research in recent years has been proposed to use ML methods, including deep learning models, as attractive alternatives to statistical ones to enhance the forecastability of traditional models in financial econometrics applications, e.g. financial time series forecasting. However, an extensive study on a large subset of 1045 monthly time series, used in the M3 Competition [1], of Makridakis et al. (2018a) suggests an interesting result: the accuracy of sophisticated ML models is below that of simple statistical ones such as Simple Exponential Smoothing (SES) (Gardner Jr., 1985), Holt and Damped ES (Gardner, 2006) and ARIMA (Box and Jenkins, 1976). The same observation is drawn from the recent M4 competition [2] of developing best forecast models for a data set of 100,000 time series. The overall results again support the hypothesis that pure ML and neural network methods perform worse than standard models like ARIMA or ES, and are also outperformed by various combinations of these benchmark statistical methods (Smyl, 2020). Makridakis et al. (2018a) also point out the disadvantage of the highly computational cost of ML models compared to statistical alternatives.

This dissertation concentrates on mitigating the aforementioned limitations of deep learning models and develops new classes of statistical models that have both good in-sample and out-of-sample performance and hence improve the applicability of the neural network based models in business and financial applications. The main philosophy of our proposed approaches is that we do not overlook the financial econometrics models but leverage the advantages from both worlds of deep learning, for high predictability, and traditionally statistical, for high interpretability, modeling in flexible ways. Our methodologies are consistent with the recent work of Smyl (2020), proposing hybrid Exponential Smoothing-Recurrent Neural Networks (ES-RNN) method and is the winner of the highly competitive M4 competition. The ES-RNN model combines ES and RNN in a flexible way to compensate each other the weaknesses of two models: the linear trend assumption and the incapability of cross-series learning of the ES model, and the lacking of capability of modeling seasonality of RNN models. Consequently, the ES-RNN is able to not only model time series with non-linear trend, seasonality but also capture the cross effects in multivariate time series. The success of the ES-RNN suggests that any

[1] https://forecasters.org/resources/time-series-data/m3-competition/
[2] https://github.com/Mcompetitions/M4-methods

models incorporating advanced ML techniques should not overlook well-established statistical models which were carefully designed to well explain stylized facts about business and financial data.

The dissertation also focuses on developing efficient Bayesian inference methods for the proposed models. This task is challenging due to the fact that the traditional Bayesian methodologies do not perform well on high-dimensional models, which is an inherited property of our neural network based models. It is even more problematic when combining deep learning methods and statistical models whose likelihood functions are intractable, e.g. the likelihood functions have no analytical forms. Designing computationally efficient Bayesian methods for these high-dimensional and intractable models are essential to enhance their applicability in real applications. Importantly, the dissertation comes with a software package so that practitioners can easily adopt the proposed methods on their own data. Most engineering-oriented papers in financial econometric research do not offer open-source software, making it difficult for econometricians to use and test their methods. Providing software in an openly accessible form is a crucial aspect for acceptance of the new methodologies in the community.

## 1.2 Summary of Original Contributions

This dissertation comprises of several original contributions to the fields of financial econometrics, statistical modeling and computational statistics. These are described as follows:

1. We introduce four new classes of deep learning based models to improve the predictive performance of traditionally statistical models, which are widely used for cross-sectional, panel and time series data available in many business and financial applications. Our approaches of leveraging the predictive power of deep learning based structures in financial econometric literature are different from many engineering-oriented methodologies in the sense that we emphasize not only on enhancing the out-of-sample performance of the traditional models, which are normally low, but also focus on retaining the interpretation of the proposed models, which is one of the main interests of econometricians and is normally overlooked in engineering-oriented models. By combining deep learning and statistical models in flexible ways, the proposed models are

often able to obtain satisfactory levels of both in-sample and out-of-sample performance through a wide range of simulation studies and applications on real datasets.

2. The proposed deep learning based models are highly sophisticated in two aspects. First, due to the natural design of neural network based structures, the deep learning models normally require a significant amount of parameters to be learned. Even though our proposed methodologies still work efficiently with small-size neural networks, the number of model parameters are still significantly higher than those of traditionally statistical models. For these high-dimensional models, designing scalable and computationally effective method for Bayesian computation is important. In chapter 3, we describe a Variational Bayes algorithm for approximating Bayesian inference in high-dimensional models and show that the proposed Bayesian approximation framework performs very well on a wide range of simulation data and real applications. Second, combining deep learning models with statistical models whose likelihood functions are intractable, e.g. stochastic volatility model, will inherently result in the intractability of combined models. In chapters 4 and 5, we show that by flexibly using the recent advances in Bayesian computational statistics, we can still efficiently sample from the those highly sophisticated posterior distributions.

3. We develop an unified user-friendly software package that can be easily used to replicate the results in our papers and to implement our methodologies on custom data. The software package is developed on multiple programming languages such as Matlab, Python and R to increase the accessibility of the proposed models in the community.

## 1.3 Dissertation Structure

The dissertation structure is outlined in Figure 1.2, and is described as follows:

1. Chapter 1 provides an introduction on the motivation and main contributions of the dissertation. The chapter also describes the overall structure of the dissertation.

2. Chapter 2 discusses background in deep learning models that will be used in later chapters of the dissertation. Fundamentals of useful Bayesian inference methods are also included in the chapter.

3. Chapter 3 introduces two new classes of deep learning based models, namely deepGLMs and deepGLMMs, to improve the predictive performance of the GLMs and GLMMS models, respectively. The chapter also discusses how to efficiently perform variable selection for neural network based models in a principle way to improve the low interpretability of traditional neural networks. An efficient variational Bayes framework called NAGVAC to perform Bayesian approximation for the proposed models is also described. This work has been published in the *Journal of Computational and Graphical Statistics* (Rank A*) under title of "Bayesian Deep Net GLM and GLMM" (Tran et al., 2020).

4. Chapter 4 introduces a new class of conditional heteroskedasticity models called Recurrent Conditional heteroskedasticity (RECH) to improve both in-sample and out-of-sample performance of the GARCH-type models. We show that the flexible design of the RECH models provide a convenient way to incorporate advances from both deep learning and statistical modeling literature, and significantly simplifies the model selection process, which is less trivial for the counterpart GARCH-type models. This work has been revised and re-submitted to the *Journal Applied Econometrics* (Rank A*).

5. Chapter 5 discusses a new parametric model, called Statistical Recurrent Stochastic Volatility (SR-SV), to model financial volatility of stock market index. The SR-SV model improves the predictive performance of the traditional SV-type models and is able to capture long-range dependencies and non-linear effects normally exhibited in financial volatility. This work is under revision for the second submission to the *Journal of Business & Economic Statistics* (Rank A*).

6. Chapter 6 gives a conclusive summary for all of the completed work in this dissertation, and discuss the potential work in the future.

| | | |
|---|---|---|
| Introduction | Chapter 1 | • Motivation and Contribution.<br>• Dissertation structure. |
| | Chapter 2 | • Background on deep learning models.<br>• Background on Bayesian inference methods. |
| Cross-sectional & Panel data | Chapter 3 | • Using DFFN as a representation learning technique to improve the predictive performance of GLM models for cross-sectional data.<br>• Using DFFN as a representation learning technique to improve the predictive performance of GLMM models for panel data.<br>• Introducing NAGVAC, an efficient VB framework to perform Bayesian approximation for high dimensional models.<br>• Evaluate model performance via simulation studies and applications to real datasets. |
| Time series data | Chapter 4 | • Improving the in-sample and out-of-sample performance of SV-type models by using a RNN structure to capture long-range dependence and non-linear effects exhibited in financial volatility.<br>• Evaluate model performance via simulation studies and applications to four internationally major stock indexes. |
| | Chapter 5 | • Propose a new class of Conditional Heteroskedastic models by incorporating auxiliary deterministic processes, governed by RNNs, into the conditional variance of GARCH-type models.<br>• Evaluate model performance via simulation studies and applications to four internationally major stock indexes. |
| Conclusion | Chapter 6 | • Conclusion<br>• Research outlook – future work |

**Table 1.1.** Outline of the dissertation.

# Chapter 2

# Fundamentals of deep learning models and Bayesian Inference methods

---

This chapter presents background on different deep learning models that will be used in the dissertation. We also revisit fundamentals of Bayesian inference frameworks which are useful to construct Bayesian methodologies in later chapters.

---

## 2.1 Notations

We use $D$ to denote a generic dataset of arbitrary data type. Table 2.1 provides notation details of different types of data that will be discussed in the dissertation.

| Data | Notation | Description |
|---|---|---|
| Cross-sectional | $D = \{(y_i, x_i), i = 1, ..., n\}$ | $y_i$ is the response, $x_i = (x_{i1}, ..., x_{ip})^\top$ is the vector of $p$ covariates |
| Panel | $D = \{(y_{it}, x_{it}), t = 1, ..., T_i, i = 1, ..., n\}$ | $y_{it}$ the response, $x_{it}$ the vector of covariates of subject $i$ at time $t$ |
| Time series | $D = y_{1:T} = \{y_t, t = 1, ..., T\}$ | $y_t$ is the observation of the time series at time $t$ |

**Table 2.1.** Notation for different data type discussed in the dissertation.

We use the notation $a := b$ to denote that $a$ is defined by $b$. For any random variable or random vector $X$ and any function $g(X)$, we denote by $\mathbb{E}_f\big(g(X)\big)$ (or $\mathbb{E}_{X\sim f}\big(g(X)\big)$, or simply $\mathbb{E}_X\big(g(X)\big)$) the expectation of $g(X)$ where $X$ follows a probability distribution with density function $f(x)$.

## 2.2 Deep learning models

This section provides background on Deep feedforward neural networks and Recurrent neural networks, two of the most commonly used neural network structures in the deep learning literature. For the purpose of the dissertation, we describe the deep learning models from a statistician's perspective by formalizing the neural network structures in the same way that statistical models are normally presented.

### 2.2.1 Deep Feedforward Neural Networks

For the purpose of this section, we use $y$ and $x$ to denote a generic response and a covariate vector, respectively. Deep feedforward neural network (DFNN), also called feedforward neural network (FNN) or multi-layer perceptrons (MLP), modeling

provides a powerful technique for approximating regression functions with multiple regressors. More specifically, the primary goal of a feedforward neural network is to define a mapping, or a regression function $y = f(x, w)$, from a set of input covariates $x = \{x_1, ..., x_p\}$ to an output $y$ and parameterized by a set of parameters $w$, which are normally called weights. These models are called feedforward because the mapping function $f(\cdot)$ is constructed by feeding input information $x$ forwardly through a sequence of computational layers, e.g input, hidden and output layers. Figure 2.1 shows the graphical presentation of a DFNN model with an input layer of 9 covariates, $L = 4$ hidden layers and a single-unit output layer representing the scalar output $y$. When the number of layers $L$ goes to large numbers, the neural network gets *deeper* and the name *deep learning* from this intuition.

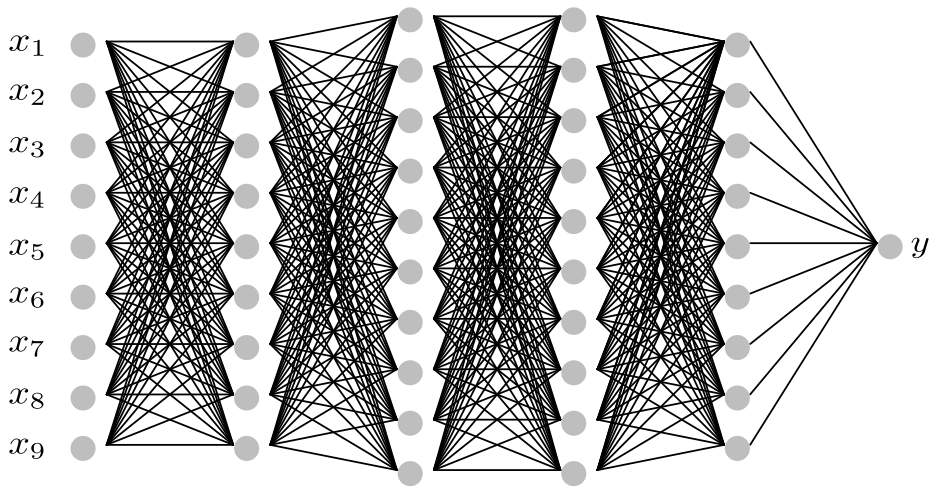


**Figure 2.1.** Graphical representation of a layered composition function with $L = 4$ hidden layers. The input layer represents 9 raw covariates $\{x_1, ..., x_9\}$. The last hidden layer (hidden layer 4) represents the predictors $Z_L = \{z_{L1}, ..., z_{Ln_L}\}$. The output layer has only one neuron representing scalar value $y$.

The final form of the mapping function $f(\cdot)$ then depends on how each individual layer is designed. Mathematically, we can think of each hidden layer defines a vector-valued function $Z_l = f_l(W_l, Z_{l-1})$, $l = 1, 2, \ldots, L$, with $L$ the number of hidden layers in the network, $w = (W_1, ..., W_L)$ the set of weights, and we define $Z_0 = f_0(x) = x$. Then the output of the FNN model in Figure 2.1 can be represented as composition function

$$y = f(x, w) = f_5(W_5, f_4(W_4, f_3(W_3, f_2(W_2, f_1(W_1, x))))) \tag{2.1}$$

which can be rewritten in a hierarchical form as

$$Z_1 = f_1(W_1, x) \tag{2.2}$$
$$Z_2 = f_2(W_2, Z_1) \tag{2.3}$$
$$Z_3 = f_3(W_3, Z_2) \tag{2.4}$$
$$Z_4 = f_4(W_4, Z_3) \tag{2.5}$$
$$y = f_5(W_5, Z_4) \tag{2.6}$$

In the deep learning literature, the function $f_l(W_l,Z_{l-1})$ is normally assumed to be of the form $\Phi_l(W_l Z_{l-1})$, where $W_l$ is a matrix of weights that connect layer $l$ to layer $l+1$ and $\Phi_l(\cdot)$ is a scalar function, called the activation function. In modern neural network modelling, the default recommendation for $\Phi_l(\cdot)$ is to use the rectified linear unit (Nair and Hinton, 2010; Le et al., 2015), or ReLU, having the form $\Phi_l(z) = max\{0,z\}$. Applying $\Phi_l$ to a vector should be understood component-wise.

Depending on the task and the characteristic of the input data, neural networks are distinguished based on how the layers and neurons are connected. For example, FNN indicate a class of neural networks in which neurons between layers are fully connected with no feedback connections. This type of structure is design to work efficiently with cross-sectional or tabular data, which are normally used with linear regression or logistic regression models in statistical modeling.

See Schmidhuber (2015) for a historical survey and Goodfellow et al. (2016) for a more comprehensive recent discussion of DFNNs and other types of neural networks, collectively known as deep learning models.

### 2.2.2 Recurrent Neural Networks

This section denotes the time series data as $\{D_t = (x_t,z_t),t = 1,2,..\}$, where $x_t = (x_{t,1},...,x_{t,K})^\top$ is the vector of inputs and $z_t$ the scalar output. For the sequence $\{x_t\}$, $x_{i:j}$ denotes $(x_i,...,x_j)$ for $i \leq j$. The goal of recurrent neural network models is to model the conditional distribution $p(z_t|x_t,D_{1:t-1})$.

There are several standard time series models. One approach is to represent time effects *explicitly* via some simple functions, often a linear function, of the lagged values of the time series. This is the mainstream time series data analysis approach with the well-known ARIMA method (Box and Jenkins, 1976). This section considers an

alternative approach representing time effects *implicitly* via latent variables that are designed to store the memory of the dynamics in the data. These latent variables, also called hidden states, are updated recurrently using the information carried over by their values from the past and the information from the data at the current time. Recurrent neural networks (RNNs), belong to the second category, were first developed in cognitive science (Elman, 1990) and successfully used in machine learning.

If the serial dependence structure is ignored, then a FNN discussed in the previous section can be used to transform the raw input data $x_t$ into a set of hidden units $h_t$, often called *learned features*, for the purpose of explaining or predicting $z_t$. Figure 2.2 is a graphical representation of a FNN model with one hidden layer containing $L$ hidden units.

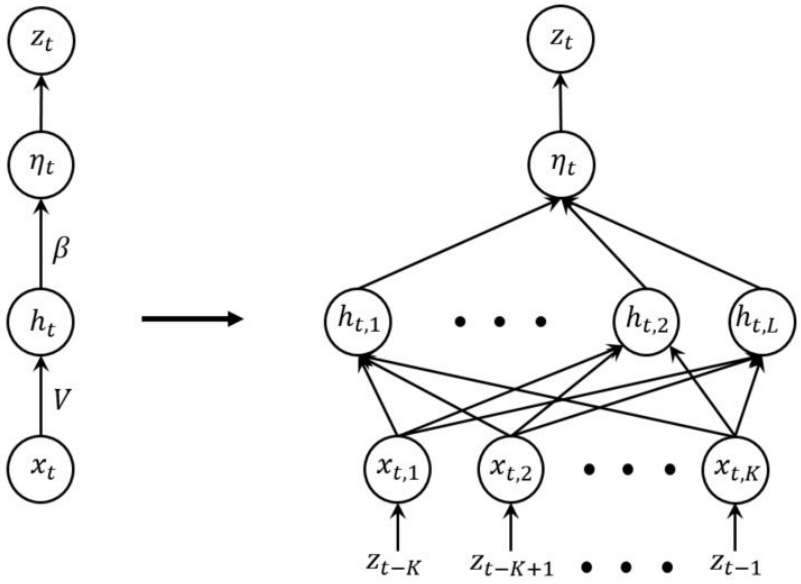


**Figure 2.2.** A FNN model with one hidden layer in compact *(left)* and explicit *(right)* styles.

.

Given the FNN model in Figure 2.2, the output $z_t$ is calculated as:

$$h_t = \Phi(Vx_t + b), \tag{2.7a}$$

$$\eta_t = \beta^\top h_t + \beta_0, \tag{2.7b}$$

$$z_t|\eta_t \sim p(z_t|\eta_t); \tag{2.7c}$$

$V$ is a $L \times K$ matrix of weights connecting the input layer to the hidden layer; $\beta = (\beta_1,...,\beta_L)^\top$ is a vector of weights connecting the hidden layer to the output layer; $\beta_0$ is

a scalar; $b=(b_1,...,b_L)^\top$ is a bias vector and $\Phi(\cdot)$ is the activation function as discussed previously. The density $p(z_t|\eta_t)$ depends on the learning task. For example, if $z_t$ is continuous, then typically $p(z_t|\eta_t)$ is a normal distribution with mean $\eta_t$ and variance $\sigma^2$; if $z_t$ is binary, then $z_t|\eta_t$ follows a Bernoulli distribution with probability $\text{logit}^{-1}(\eta_t)$.

FNNs provide a powerful way to approximate the true function that maps the input $x_t$ to the mean $\mathbb{E}(z_t|x_t)$ or to transform the raw data $x_t$ into summary statistics $h_t$ having some desirable properties. However, FNNs are unsuitable for time series data analysis as the time effects and the serial correlations are ignored. The main idea behind RNNs is to let the set of hidden units $h_t$ feed itself on its lagged value $h_{t-1}$. Hence, an RNN can be best thought of as a FNN that allows a connection of the hidden units to their value from the previous time step, enabling the network to possess memory. This basic RNN model (Elman, 1990) can be written as:

$$h_t=\Phi(Vx_t+Wh_{t-1}+b), \tag{2.8a}$$

$$\eta_t=\beta^\top h_t+\beta_0, \tag{2.8b}$$

$$z_t|\eta_t\sim p(z_t|\eta_t); \tag{2.8c}$$

the parameters are the bias vector $b$, the bias scalar $\beta_0$, the weight matrices $V$, $W$, and $\beta$ for input-to-hidden, hidden-to-hidden and hidden-to-output connections, respectively. Similarly to FNNs, $\Phi(\cdot)$ is a non-linear activation function; common choices are the ReLU or the sigmoid $\phi(z)=1/(1+e^{-z})$. Usually, we can set $h_1=0$, i.e. the neural network initially does not have any memory.

Figure 2.3 graphically illustrates the RNN model (2.8a)-(2.8c). We follow Goodfellow et al. (2016) and use a black square to indicate the delay of a single time step in the circuit diagram (*left*). The circuit diagram (*Left*) can be interpreted as an unfolded computational graph (*Right*), where each node is associated with a particular time step. The calculation of $h_t$ can be represented as a Simple Recurrent Neuron (SRN) unit, as Figure 2.4 (*Left*) shows, and we refer to (2.8a) as $h_t=\text{SRN}(x_t,h_{t-1})$, taking data $x_t$ at time $t$ and the previous state $h_{t-1}$ as the inputs. Using the SRN structure, the unfolded graph of the RNN model of Elman (1990), which is normally referred to as the Simple RNN model, can be reinterpreted as the unfolded graph in Figure 2.4 (*Right*).

The unfolded graph in Figure 2.3 suggests that the hidden state at time $t$ is the output of a composite function

$$h_t=f\Big(x_t,f(x_{t-1},...,f(x_1,h_0))\Big), \text{ where } f(x_t,h_{t-1}):=\Phi(Vx_t+Wh_{t-1}+b), \tag{2.9}$$

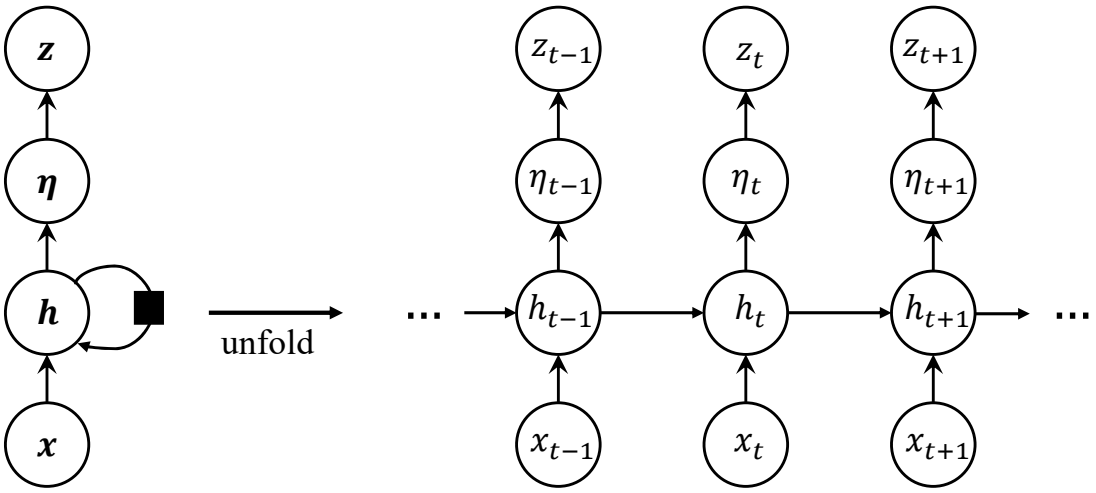


**Figure 2.3.** Graphical representation of the RNN model in (2.8a)-(2.8c).

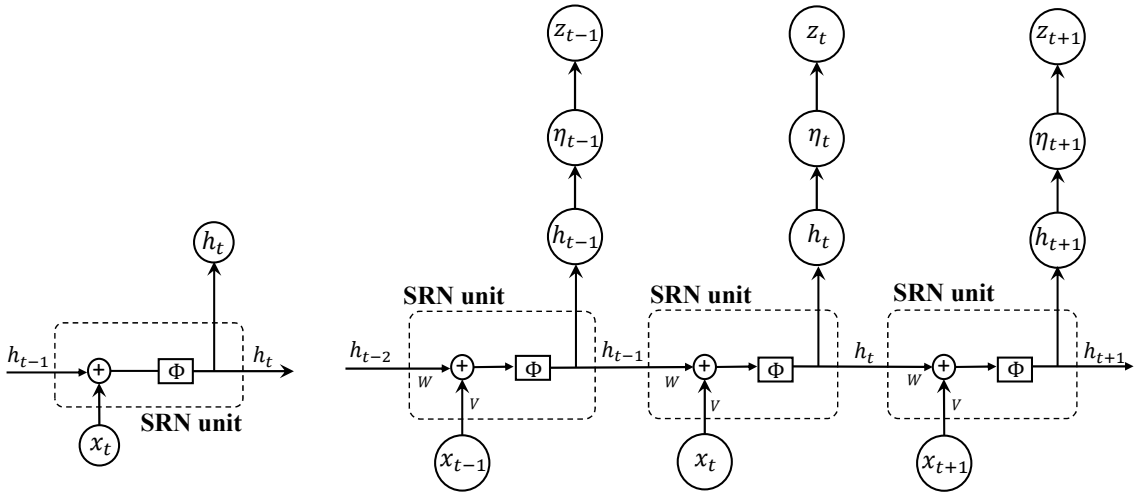


**Figure 2.4.** The structures of the SRN unit (*left*) and the graphical representation of the Simple RNN model (*right*), which uses the SRN unit to compute the latent state $h_t$. The $\oplus$ symbols represents the addition operation.

which somewhat resembles a multiplication structure in terms of the weight $W$. Consequently, the gradient of $h_t$ with respect to the model parameters might either explode or vanish if $t$ is sufficiently large and $W$ is not equal to 1, and hence making it

inefficient for the Simple RNN model to learn in long time series. See Goodfellow et al. (2016) for further explanation.

Many sophisticated RNN structures have been proposed to overcome the aforementioned problem in the Simple RNN model; for example, the Long Short-term Memory model of Hochreiter and Schmidhuber (1997), the Gated Recurrent Unit of Cho et al. (2014) and the Statistical Recurrent Unit (SRU) of Oliva et al. (2017). The SRU allows the vector of summary statistics $h_t$ to traverse through the network using a moving average. Section 5 will discuss a SRU based statistical model as its structure and some of its main parameters carry statistical meaning. A general SRU structure is mathematically written as

$$r_t = \Phi(W_h h_{t-1} + b_r), \tag{2.10a}$$

$$\varphi_t = \Phi(W_r r_t + W_x x_t + b_\varphi), \tag{2.10b}$$

$$h_t^{(\alpha_j)} = \alpha_j h_{t-1}^{(\alpha_j)} + (1-\alpha_j)\varphi_t,\ j=1,...,m;\ h_t = \left(h_t^{(\alpha_1)},\cdots,h_t^{(\alpha_m)}\right)^\top, \tag{2.10c}$$

where $\alpha = (\alpha_1,...,\alpha_m) \in (0,1)$ is a vector of moving average weights, and $W_h$, $b_r$, $W_r$, $W_x$ and $b_\varphi$ are the model parameters. We denote the functional learning structure in (2.10a)-(2.10c) as $h_t = \text{SRU}(x_t, h_{t-1})$, which takes $x_t$ - the input data at current time $t$ - and $h_{t-1}$ - the previous output of the SRU - as the input arguments. See Figure 2.5(a) for the graphical representation of this SRU structure. The moving average structure of the state $h_t$ allows the RNN network with SRU units to enjoy some advantages compared to other RNN models. The current state $h_t$ is related to the previous state $h_{t-1}$ both *directly* and *indirectly* and hence mitigate the problem of multiplying the same quantities multiple times as in the Simple RNN model. The novel architecture of the SRU allows the model to capture long term dependencies in data via simple moving averages.

## 2.3 Bayesian inference

Let $D$ denote the data and $p(D|\theta)$ the likelihood function based on a postulated model, with $\theta \in \Theta$ the vector of model parameters to be estimated. Let $p(\theta)$ be the prior. Bayesian inference encodes all the available information about the model parameter $\theta$ in its posterior distribution with density

$$\pi(\theta) = p(\theta|D) = \frac{p(D,\theta)}{p(D)} = \frac{p(\theta)p(D|\theta)}{p(y)} \propto p(\theta)p(D|\theta), \tag{2.11}$$

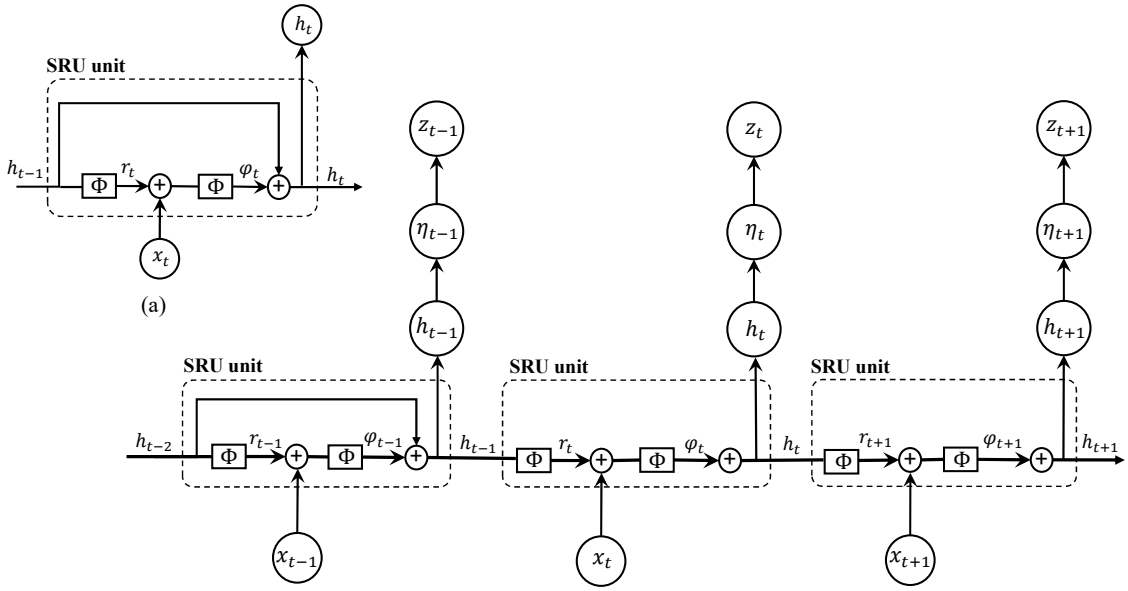


**Figure 2.5.** The structure of the SRU unit (*a*) and the graphical representation of the SRU model (*b*), which uses the SRU unit to compute the latent state $h_t$.

where $p(D) = \int_\Theta p(\theta)p(D|\theta)d\theta$, called the *marginal likelihood* or *evidence*. Here, the notation '$\propto$' means proportional up to the normalizing constant that is independent of the parameter ($\theta$). In most Bayesian derivations, such a constant can be safely ignored.

Bayesian inference typically requires computing expectations with respect to the posterior distribution. For example, the posterior mean, which is often used for point estimation, is an expectation of $\theta$ with respect to the posterior distribution $p(\theta|D)$. However, it is often difficult to compute such expectations, partly because the density $p(\theta|y)$ itself is intractable as the normalizing constant $p(D)$ is often unknown. For many applications, exact Bayesian inference is performed using Markov Chain Monte Carlo (MCMC) or Sequential Monte Carlo (SMC), which estimates expectations w.r.t. $p(\theta|D)$ unbiasedly by sampling from it. For other applications where $\theta$ is high dimensional or fast computation is of primary interest, VB is an attractive alternative to MCMC/SMC; VB approximates the posterior distribution by a probability distribution with density $q(\theta)$ belonging to some tractable family of distributions $\mathcal{Q}$ such as Gaussians. The next sections will discuss fundamental background on MCMC, SMC ans VB.

### 2.3.1 Markov Chain Monte Carlo

When directly sampling from the posterior distribution $\pi(\theta)$ is infeasible, the Markov chain Monte Carlo (MCMC) method (Metropolis et al., 1953; Hastings, 1970) is the most widely used Bayesian inference method. One standard MCMC algorithm is the Metropolis-Hastings (MH) algorithm. Under the MH algorithm, an iterative process of proposing new sample $\theta'$ from a proposal distribution $p(\theta'|\theta)$ and accepting that proposed sample with a certain probability is carried out. In practice, the random walk proposal, e.g. multivariate normal distributions, for $p(\theta'|\theta)$ is the most widely used proposal distribution. Algorithm 2.1 summarizes the MH sampler for sampling from the posterior distribution of the model parameters $\theta$ in (2.11).

**Algorithm 2.1** Metropolis-Hastings Algorithm

For each MCMC iteration:

1. Sample $\theta'$ from the proposal density $q(\theta'|\theta)$.
2. Accept the proposal $\theta'$ with the probability

$$\min\left\{1, \frac{p(D|\theta')}{p(D|\theta)}\frac{p(\theta')}{p(\theta)}\frac{q(\theta|\theta')}{q(\theta'|\theta)}\right\}. \tag{2.12}$$

For modles with intractable likelihood function, e.g. there is no analytical form for $p(D|\theta)$, Andrieu and Roberts (2009) shows that the Algorithm 2.1 is still valid when unbiased estimate of the likelihood function $\widehat{p}(D|\theta')$ is available and the acceptance probability then becomes

$$\min\left\{1, \frac{\widehat{p}(D|\theta')}{\widehat{p}(D|\theta)}\frac{p(\theta')}{p(\theta)}\frac{q(\theta|\theta')}{q(\theta'|\theta)}\right\}$$

and call this variant as the Pseudo Marginal Metropolis Hasting (PMMH) sampler. Chapter 5 will discuss a special case of the PMMH algorithm to perform MCMC iterations for non-linear state space models whose likelihood functions are intractable.

### 2.3.2 Sequential Monte Carlo

MCMC is the most popular Bayesian inference technique due to its implementation simplicity and computational efficiency. However, when the target distribution $\pi(\theta)$

is far from univariate normal distributions, it can be costly and difficult to tune model parameters when using conventional MCMC techniques (Roberts and Rosenthal, 2009) due to the nature of the algorithm. Sequential Monte Carlo (SMC), on the other hand, can sample efficiently from non-standard posteriors, provides the marginal likelihood estimate as a by-product, and is a convenient way for computing one-step-ahead forecasts. In order to sample from the posterior $\pi(\theta)$, the SMC method (Neal, 2001; Chopin, 2002; Del Moral et al., 2006) first samples a set of $M$ weighted particles $\{W_0^j, \theta_0^j\}_{j=1}^M$ from an easy-to-sample distribution $\pi_0(\theta)$, such as the prior $p(\theta)$, and then traverses these particles through intermediate distributions $\pi_t(\theta)$, $t = 1,...,K$, which become the posterior distribution $\pi(\theta)$, i.e. $\pi_K(\theta) = \pi(\theta)$. A common choice is to set $\pi_0(\theta) = p(\theta)$ as the prior $p(\theta)$ if it is possible to sample from $p(\theta)$. There are two common ways to design such a sequence of intermediate distributions: likelihood annealing (Neal, 2001) and data annealing (Chopin, 2002). The SMC with likelihood annealing uses the following intermediate distributions

$$\pi_t(\theta) := \pi_t(\theta|y_{1:T}) \propto p(y_{1:T}|\theta)^{\gamma_t} p(\theta), \tag{2.13}$$

where $\gamma_t$ is referred to as the temperature level and $0 = \gamma_0 < \gamma_1 < \gamma_2 < ... < \gamma_K = 1$.

We denote the data as $y_{1:T}$ to imply that the dissertation uses SMC to sample statistical models working with time series data discussed in Chapter 4 and 5. The SMC method consists of three main steps: reweighting, resampling and a Markov move. There are various ways to implement SMC in practice; here we briefly present one of these. At the begining of iteration $t$, the set of weighted particles $\{W_{t-1}^j, \theta_{t-1}^j\}_{j=1}^M$ that approximate the intermediate distribution $\pi_{t-1}(\theta)$ is reweighted to approximate the target $\pi_t(\theta)$. The efficiency of these weighted particles as a representation of $\pi_t(\theta)$ is often measured by the effective sample size (ESS) (Kass et al., 1998; Liu and Chen, 1998) defined in (2.16). If the ESS is below a prespecified threshold, the particles are resampled; the resulting equally-weighted samples are then refreshed by a Markov kernel whose invariant distribution is $\pi_t(\theta)$. Algorithm 2.2 summarizes this SMC using the likelihood annealing method. We follow Gunawan et al. (2018) and choose the tempering sequence $\gamma_t$ adaptively to ensure a sufficient level of particle efficiency by selecting the next value of $\gamma_t$ such that ESS stays above a threshold.

SMC with likelihood annealing sampler is suitable for in-sample analysis, as it uses the sequence of distributions in (2.13) which requires the full training data $y_{1:T}$ to be available. For out-of-sample rolling forecasts where the model parameters $\theta$ are updated once new data arrive, it is more appealing to use the SMC with the data

**Algorithm 2.2** SMC with likelihood annealing

1. Sample $\theta_0^j \sim p(\theta)$ and set $W_0^j = 1/M$ for $j = 1...M$

2. **For** $t = 1,...,K$,

**Step 1: Resampling**: Compute the unnormalized weights

$$w_t^j = W_{t-1}^j \frac{p(y_{1:T}|\theta_{t-1}^j)^{\gamma_t} p(\theta_{t-1}^j)}{p(y_{1:T}|\theta_{t-1}^j)^{\gamma_{t-1}} p(\theta_{t-1}^j)} = W_{t-1}^j p(y_{1:T}|\theta_{t-1}^j)^{\gamma_t - \gamma_{t-1}}, \quad j = 1,...,M \tag{2.14}$$

and set the new normalized weights

$$W_t^j = \frac{w_t^j}{\sum_{s=1}^M w_t^s}, \quad j = 1,...,M. \tag{2.15}$$

**Step 2:** Compute the effective sample size (ESS):

$$\text{ESS} = \frac{1}{\sum_{j=1}^M \left(W_t^j\right)^2}. \tag{2.16}$$

**if** ESS $< cM$ for some $0 < c < 1$, **then**

(i) **Resampling**: Resampling from $\{\theta_{t-1}^j\}_{j=1}^M$ using the weights $\{W_t^j\}_{j=1}^M$, and then set $W_t^j = 1/M$ for $j = 1...M$, to obtain the new equally-weighted particles $\{\theta_t^j, W_t^j\}_{j=1}^M$.

(ii) **Markov move**: For each $j = 1,...,M$, move the sample $\theta_t^j$ according to $N_{\text{lik}}$ random walk Metropolis-Hasting steps:

(a) Generate a proposal $\theta_t^{j\prime}$ from a multivariate normal distribution $\mathcal{N}(\theta_t^j, \Sigma_t)$ with $\Sigma_t$ the covariance matrix.

(b) Set $\theta_t^j = \theta_t^{j\prime}$ with the probability

$$\min\left(1, \frac{p(y_{1:T}|\theta_t^{j\prime})^{\gamma_t} p(\theta_t^{j\prime})}{p(y_{1:T}|\theta_t^j)^{\gamma_t} p(\theta_t^j)}\right); \tag{2.17}$$

otherwise keep $\theta_t^j$ unchanged.

**end**

3. The log of the estimated marginal likelihood is:

$$\widehat{\log p(y_{1:T})} = \sum_{t=1}^K \log\left(\sum_{j=1}^M w_t^j\right). \tag{2.18}$$

annealing sampler of Chopin (2002). This SMC sampler generates weighted particles from the following sequence of distributions

$$\pi_t(\theta) := \pi_t(\theta|y_{1:t}) \propto p(y_{1:t}|\theta)p(\theta) \propto \pi_{t-1}(\theta)p(y_t|\theta, y_{1:t-1}), \tag{2.19}$$

with $y_{1:t}$ the data available up to time $t$. The unnormalized weights at step $t$ in (2.14) become

$$w_t^j = W_{t-1}^j \frac{p(y_{1:t}|\theta_{t-1}^j)p(\theta_{t-1}^j)}{p(y_{1:t-1}|\theta_{t-1}^j)p(\theta_{t-1}^j)} = W_{t-1}^j p(y_t|y_{1:t-1}, \theta_{t-1}^j), j = 1,...,M. \tag{2.20}$$

Algorithm 2.3 summarizes SMC with data annealing. For the RECH models in chapter 4, we use SMC with likelihood annealing for in-sample Bayesian analysis, and SMC with data annealing for out-of-sample analysis and forecasting.

It is noted that the formula in (2.20) only requires the partial likelihood $p(y_t|y_{1:t-1}, \theta_{t-1}^j)$ to update the weights of particle set when a new observation $y_t$ is observed. In this manner, the reweighting step does not involve the historical data to update the weights and hence the data annealing strategy is computationally efficient, especially when the time series is long. However, similar to the SMC with likelihood anneal approach, the SMC with data anneal method also requires all the data up to time $t$, i.e. $y_{1:t}$, to perform the Markov moving steps. Fortunately, as the particles are moved separately, it is possible to parallelize these computationally expensive steps. It is worth noting that, similar to the SMC with likelihood annealing approach, the resampling and Markov move steps are only implemented when the ESS is below a threshold value.

### 2.3.3 Variational Bayes

Bayesian inference has been long called for Bayesian computation techniques that are scalable to large data sets and applicable in big and complex models with a huge number of unknown parameters to infer. Sampling methods, such as MCMC and SMC, in their current development do not meet this need. Sampling methods have not been successfully used in some modern areas such as deep neural networks. Even in more traditional areas such as graphical modelling and mixture modelling, it is very challenging to use MCMC and SMC. Variational Bayes (VB) is an optimization-based technique for approximate Bayesian inference, and provides a computationally efficient alternative to sampling methods. VB belongs to the bigger class of Variational

**Algorithm 2.3** SMC with data annealing

1. Sample $\theta_0^j \sim p(\theta)$ and set $W_0^j = 1/M$ for $j = 1...M$
2. **For** $t = 1,...,T$,

**Step 1, reweighting:** Compute the unnormalized weights

$$w_t^j = W_{t-1}^j p(y_t|y_{1:t-1}, \theta_{t-1}^j), \quad j = 1, ..., M, \tag{2.21}$$

and set the new normalized weights as

$$W_t^j = \frac{w_t^j}{\sum_{s=1}^M w_t^s}, \quad j = 1, ..., M. \tag{2.22}$$

**Step 2:** Compute the effective sample size (ESS):

$$\text{ESS} = \frac{1}{\sum_{j=1}^M \left(W_t^j\right)^2}. \tag{2.23}$$

**if** ESS $< cM$ for some $0 < c < 1$, **then**

(i) **Resampling**: Resample from $\{\theta_{t-1}^j, W_t^j\}_{j=1}^M$, and then set $W_t^j = 1/M$ for $j = 1...M$, to obtain the new equally-weighted particles $\{\theta_t^j, W_t^j\}_{j=1}^M$.

(ii) **Markov move**: for each $j = 1,...,M$, move the sample $\theta_t^j$ according to $N_{\text{data}}$ random walk Metropolis-Hasting steps:

(a) Generate a proposal $\theta_t^{j\prime}$ from multivariate normal distribution $\mathcal{N}(\theta_t^j, \Sigma_t)$ with $\Sigma_t$ the covariance matrix.

(b) Set $\theta_t^j = \theta_t^{j\prime}$ with the probability

$$\min\left(1, \frac{p(y_{1:t}|\theta_t^{j\prime})p(\theta_t^{j\prime})}{p(y_{1:t}|\theta_t^j)p(\theta_t^j)}\right) \tag{2.24}$$

otherwise keep $\theta_t^j$.

**end**

Inference methods, which can also be used in the frequentist context for maximum likelihood estimation when there are missing data. The names Variational Bayes and Variational Inference are often used exchangeably in the literature, however, we prefer the former in this dissertation as we are solely interested in approximating the posterior distributions for Bayesian inference.

The best VB approximation $q^* \in \mathcal{Q}$ is found by minimizing the Kullback-Leibler (KL) divergence *from* $q(\theta)$ *to* $p(\theta|y)$

$$q^* = \arg\min_{q\in\mathcal{Q}} \left\{ \mathrm{KL}(q\|p(\cdot|y)) := \int q(\theta) \log \frac{q(\theta)}{p(\theta|y)} d\theta \right\}. \tag{2.25}$$

Then, Bayesian inference is performed with the intractable posterior $p(\theta|y)$ replaced by the tractable VB approximation $q^*(\theta)$. It is easy to see that

$$\mathrm{KL}(q\|p(\cdot|y)) = -\int q(\theta) \log \frac{p(\theta)p(y|\theta)}{q(\theta)} d\theta + \log p(y),$$

thus minimizing KL is equivalent to maximizing the lower bound on $\log p(y)$

$$\mathrm{LB}(q) := \int q(\theta) \log \frac{p(\theta)p(y|\theta)}{q(\theta)} d\theta = \mathbb{E}_q\left(\log \frac{p(\theta)p(y|\theta)}{q(\theta)}\right). \tag{2.26}$$

Without any constraint on $\mathcal{Q}$, the solution to (2.25) is $q^*(\theta) = p(\theta|y)$. Chapter 3 will discuss a VB framework called NAGVAC which provides a computationally efficient way to find the variational distribution $q^*(\theta)$ for statistical models with many parameters. For a comprehensive discussion on common Variational Bayese methods, see Tran et al. (2021).

### 2.3.4 Model choice by marginal likelihood

The marginal likelihood is often used to choose between models via the Bayes factor (Jeffreys, 1935; Kass and Raftery, 1995). In order to compare the relative performance between two models $M_1$ and $M_2$ on a given data $y_{1:T}$, we can use the Bayes factor defined by

$$F_{M_1,M_2} = \frac{p(y_{1:T}|M_1)}{p(y_{1:T}|M_2)}, \tag{2.27}$$

providing a Bayesian alternative to hypothesis testing. The larger the Bayes factor $F_{M_1,M_2}$, the stronger $M_1$ is supported by the data than $M_2$. Jeffreys (1961) suggests a scale of interpretation of the Bayes factor $F_{M_1,M_2}$ as listed in Table 2.2. We note that the SMC sampler in the previous section provides an efficient way to compute the marginal likelihood, e.g. Equation 2.18.

| Grade | $F_{M_1,M_2}$ | $\log_{10} F_{M_1,M_2}$ | $\ln F_{M_1,M_2}$ | Strength of evidence |
|---|---|---|---|---|
| 0 | $<10^0$ | $<0$ | $<0$ | Negative (supports $M_2$) |
| 1 | $10^0-10^{1/2}$ | $0.0-0.5$ | $0.0-1.2$ | Barely worth mentioning |
| 2 | $10^{1/2}-10^1$ | $0.5-1.0$ | $1.2-2.3$ | Substantial |
| 3 | $10^1-10^{3/2}$ | $1.0-1.5$ | $2.3-3.5$ | Strong |
| 4 | $10^{3/2}-10^2$ | $1.5-2.0$ | $3.5-4.6$ | Very strong |
| 5 | $>10^2$ | $>2.0$ | $>4.6$ | Decisive |

**Table 2.2.** Jeffreys' scale of interpretation of the Bayes Factor $F_{M_1,M_2}$.

# Chapter 3

# Bayesian Deep Net GLM and GLMM

Deep feedforward neural networks (DFNNs) are a powerful tool for functional approximation. We describe flexible versions of generalized linear and generalized linear mixed models incorporating basis functions formed by a DFNN. The consideration of neural networks with random effects is not widely used in the literature, perhaps because of the computational challenges of incorporating subject specific parameters into already complex models. Efficient computational methods for high-dimensional Bayesian inference are developed using Gaussian variational approximation, with a parsimonious but flexible factor parametrization of the covariance matrix. We implement natural gradient methods for the optimization, exploiting the factor structure of the variational covariance matrix in computation of the natural gradient. Our flexible DFNN models and Bayesian inference approach lead to a regression and classification method that has a high prediction accuracy, and is able to quantify the prediction uncertainty in a principled and convenient way. We also describe how to perform variable selection in our deep learning method, and improve the interpretability of our DFNN based models. The proposed methods are illustrated in a wide range of simulated and real-data examples, and compared favourably to a state of the art flexible regression and classification method in the statistical literature, the Bayesian additive regression trees (BART) method.

## 3.1 Motivation and Contribution

This chapter considers variants of generalized linear models (GLMs) and generalized linear mixed model (GLMMs) using DFNNs as a way to efficiently transform a vector of $p$ raw covariates $X = (X_1,...,X_p)^\top$ into a new vector of $m$ predictors $Z$ in the model. We refer to these DFNN-based versions of GLM and GLMM as DeepGLM and DeepGLMM, respectively. A conventional GLM uses a link function that links the conditional mean of the response variable $Y$ to a linear combination of the predictors $Z = \phi(X) = (\phi_1(X),...,\phi_m(X))^\top$, with each $\phi_j(X)$ a function of $X$. We refer to the original raw input variables $X_j$ as covariates, and the transformations $\phi_j(X)$ as predictors. In conventional GLMs, the $\phi_j(X)$ are chosen *a priori* in some way before any model selection, but here we are concerned with learning an appropriate $Z$ from data through a flexible smooth transformation. In the machine learning literature the predictors $Z$ are commonly referred to as learned features. If a DFNN is used for transforming the covariates, then $Z$ has the form

$$Z = f_L\Big(W_L, f_{L-1}\big(W_{L-1}, \cdots f_1(W_1, X)\cdots\big)\Big), \tag{3.1}$$

which is discussed in Section 2.2.1.

The architecture (3.1) provides a powerful way to transform the raw covariate data $X$ into summary statistics $Z$ that have some desirable properties. In the DeepGLM, we link the conditional mean of the response $\xi = \mathbb{E}(Y|X)$ to a linear combination of $Z$

$$g(\xi) = \beta_0 + Z^\top \widetilde{\beta},$$

and the model parameters consist of $w = (W_1,...,W_L)$, $\beta$ and other possible parameters such as dispersion parameters. Section 3.3.2 defines DeepGLMM models by adding random effects to DeepGLM models, which are designed to model within-subject dependence. The deepGLM and deepGLMM models are fully parametric models in the sense that model structure (number of layers, number of hidden units in each layer, activation function) and model parameters (weights), and hence the closed-form presentation, must be specified prior to the estimation and prediction steps. The use of neural network basis functions in the context of mixed effects models seems not considered much in the literature. Lai et al. (2006) is the only work that we know of that deals with a neural network basis and random effects, and they use it for modeling pharmacokinetic data. They consider neural networks with only one hidden layer, however.

Estimation and variable selection in complex and high-dimensional models like DeepGLM and DeepGLMM are challenging. This article develops Bayesian inference based on variational approximation, which provides an approach to approximate Bayesian inference that is useful for many modern applications involving complex models and large datasets. We also consider variable selection, which is often of primary interest in statistics but is somewhat overlooked in the deep learning literature. We describe a Bayesian adaptive group lasso method (Leng et al., 2014; Kyung et al., 2010) for variable selection in DeepGLM and DeepGLMM, in which the adaptive shrinkage parameters are estimated using marginal likelihood maximization with the optimization updating procedure conveniently embeded within the variational approximation. Our variational approximation scheme assumes a multivariate Gaussian approximating family. With such a family, parsimonious but flexible methods for parametrizing the covariance matrix are necessary if the approach is to be useful for problems with a high-dimensional parameter. Here we consider factor parametrizations (Bartholomew et al., 2011) which are often effective for describing dependence in high dimensional settings. We discuss efficient methods for performing the variational optimization in this context using the natural gradient (Amari, 1998) by leveraging the factor structure and using iterative conjugate gradient methods for solving large linear systems. In the case with a one-factor decomposition, we show that the natural gradient can be computed efficiently without iterative conjugate gradient methods, which leads to a particularly simple Gaussian variational method for fitting high-dimensional models that can often be adequate for predictive inference. We will refer to this estimation method as the NAtural gradient Gaussian Variational Approximation with factor Covariance (NAGVAC).

We illustrate the DeepGLM and DeepGLMM and the training method NAGVAC in a range of experimental studies and applications, with a focus on datasets of only moderate size where quantification of prediction uncertainty is important. The Bayesian approach we follow is attractive in these applications as it allows a principled and automatic way for selecting the (*many*) shrinkage parameters; see Section 3.5. The Bayesian approach also leads to a principled and convenient way for quantifying prediction uncertainty through prediction intervals on test data, which would be challenging to do so with non-Bayesian approaches for large models like DeepGLM. Many successful applications of deep learning are in image processing and speech recognition, where the datasets are large and have some special domain-application characteristics such as association between the local pixels in an image. Skepticism

has sometimes been expressed about whether deep learning methods are useful for applications involving limited data where both prediction accuracy and quantifying prediction uncertainty are the focus. The main conclusion in our examples is that the DFNN-based regression models DeepGLM and DeepGLMM perform very well in terms of prediction accuracy and prediction uncertainty, provided appropriate attention is paid to regularization methods, prior distributions and computational algorithms. We obtain results which compare favourably to the Bayesian Additive Regression Trees method (BART) of Chipman et al. (2010), which is a commonly-used flexible regression and classification method in the statistics literature. Software packages in Matlab, R and Python implementing the proposed methods are available at `https://github.com/vbayeslab`.

## 3.2 Related work

Polson and Sokolov (2017) provide a Bayesian perspective on DFNN methodology and explain its interesting connection to statistical techniques such as principal component analysis and reduced rank regression. However, they do not provide any training method, which is a challenging problem in Bayesian inference with DFNN.

The NAGVAC method is closely related to Ong et al. (2017a), who consider Gaussian variational approximation with a factor covariance structure using stochastic gradient approaches for the optimization. Their approach uses the so-called reparametrization trick (Kingma and Welling, 2013; Rezende et al., 2014) and its modification by Roeder et al. (2017) for estimating gradients of the variational objective. However, for certain models where it is very challenging to optimize the variational objective, first-order optimization methods such as those considered in Ong et al. (2017a) may be very slow to converge. Ong et al. (2017b) consider natural gradient methods for Gaussian variational approximations using a factor covariance structure. However, their work was in the context of likelihood-free inference methods with a parameter of dimension at most a few hundred, and the approach they develop does not scale to larger problems. The current work shows how the natural gradient Gaussian variational approximation with factor covariance can be implemented in very high dimensions. Tran et al. (2017) consider variational approximation for intractable likelihood problems and demonstrate the importance of natural gradient in this context.

Much of the recent literature relevant to our training method occurs in the field of deep learning. Martens (2010) considers second-order optimization methods in deep learning and describes Hessian-free optimization methods, adapting a long history of related methods in the numerical analysis literature to that context. A detailed discussion of the connections between natural gradient methods and second order optimization methods is given in Martens (2014). Pascanu and Bengio (2014) consider the use of the natural gradient in deep learning problems and its connections with Hessian-free optimization, but like Martens (2010) their work is not specifically concerned with variational objectives. Fan et al. (2015) consider how to implement Hessian-free methods for the case of a variational objective function and Gaussian approximating family. Similarly to our development here, they consider using conjugate gradient linear solvers as an efficient solution to the difficult matrix calculations that occur in a naive formulation of second-order methods. By a reparametrization approach they are able to obtain estimates of Hessian vector products. They do not consider factor parametrizations of the covariance structure, however. Here we leverage the factor covariance structure to calculate the matrix vector products we need directly, without the need to store large matrices. Recently Regier et al. (2017) consider a second order trust region method for black box variational inference. They show that while their approach may be more expensive per iteration than common first-order methods for variational Gaussian approximation such as those implemented in Titsias and Lázaro-Gredilla (2014) and Kucukelbir et al. (2017), it reduces total computation time and provably converges to a stationary point. It may be useful to consider how to exploit a factor parametrization of the covariance structure in the framework they develop, but this is not considered here.

## 3.3 Flexible regression models with DFNN

This section presents the DeepGLM and DeepGLMM models. Deep feedforward neural network models, or multi-layer perceptrons, for classification and regression with a continuous response have been widely used in the machine learning literature. We use statistical terminologies and unify these models under the popular GLM framework, and propose the DFNN-based version of GLMM for analyzing panel data.

### 3.3.1 DeepGLM

Consider a dataset $D=\{(y_i,x_i),i=1,...,n\}$ with $y_i$ the response and $x_i=(x_{i1},...,x_{ip})^\top$ the vector of $p$ covariates. We also use $y$ and $x$ to denote a generic response and a covariate vector, respectively. Consider a neural net with the input vector $x$ and a scalar output as represented in Figure 2.1. Denote by $z_j=\phi_j(x,w)$, $j=1,...,m$, the units in the last hidden layer, where $w$ is the weights up to the last hidden layer, and $\beta=(\beta_0,\beta_1,...,\beta_m)^\top$ are the weights that connect the $z_j$ to the output which we write as

$$\mathfrak{N}(x,w,\beta)=\beta_0+\beta_1 z_1+...+\beta_m z_m=\beta_0+\widetilde{\beta}^\top z$$

with $\widetilde{\beta}=(\beta_1,...,\beta_m)^\top$ and $z=(z_1,...,z_m)^\top$.

We assume that the conditional density $p(y|x)$ has an exponential family form (McCullagh and Nelder, 1989)

$$p(y|x)=\exp\left(\frac{y\varpi-b(\varpi)}{\phi}+c(\phi,y)\right) \tag{3.2}$$

with the canonical parameter $\varpi$ and the dispersion parameter $\phi$, and $c(\phi,y)$ is a function independent of $\varpi$. Let $g(\cdot)$ be the link function that links the conditional mean $\xi=\xi(x)=\mathbb{E}(y|x)$ to a function of the covariates $x$. In the conventional GLM, $g(\xi)$ is assumed to be a linear combination of $x$. In order to achieve flexibility and capture possible non-linear effects that $x$ has on $\xi$, we propose the more flexible model

$$g(\xi)=\beta_0+\widetilde{\beta}^\top z=\mathfrak{N}(x,w,\beta). \tag{3.3}$$

In our later examples we use the canonical link function where $\varpi=g(\xi)$, which is the log link for Poisson responses and the logit link for binomial responses. The predictors (learned features) $z_j$ efficiently capture the important non-linear effects of the original raw covariates $x$ on $g(\xi)$.

The model (3.3) is flexible, but can be hard to interpret. It may be useful in this respect to introduce additional structure. One possibility is to partition the covariates as $x=({x^{(1)}}^\top,{x^{(2)}}^\top)^\top$ with $x^{(1)}$ and $x^{(2)}$ expected to have nonlinear and linear effects respectively on $g(\xi)$, so that

$$g(\xi) \;=\; \mathfrak{N}(x^{(1)},w,\beta^{(1)})+{\beta^{(2)}}^\top x^{(2)} \tag{3.4}$$

where $\beta^{(1)}$ parametrizes the nonlinear effects w.r.t. the covariates $x^{(1)}$ and $\beta^{(2)}$ parametrizes the linear effects w.r.t. the covariates $x^{(2)}$. Write $\beta=(\beta^{(1)},\beta^{(2)})$. We refer

to the general regression model with the exponential family for response distribution (3.2) and the mean model (3.3) or (3.4) as DeepGLM.

The vector of model parameters $\theta$ consists of $w$, $\beta$ and possibly dispersion parameters $\phi$ if $\phi$ is unknown. The density $p(y|x)$ in (3.2) is now a function of $\theta$, $p(y|x) = p(y|x,\theta)$. Given a dataset $D$, the likelihood function is

$$L(\theta) = \prod_{i=1}^{n} L_i(\theta), \quad L_i(\theta) = p(y_i|x_i,\theta), \tag{3.5}$$

and likelihood-based inference methods, including Bayesian methods, can be applied.

### 3.3.2 DeepGLMM

Consider a panel dataset $D = \{(y_{it},x_{it}),t = 1,...,T_i,i = 1,...,n\}$ with $y_{it}$ the response and $x_{it}$ the vector of covariates of subject $i$ at time $t$. Generalized linear mixed models (GLMM) use random effects to account for within-subject dependence; see, e.g., Stroup (2012). Let $z_{it,j} = \phi_j(x_{it},w)$, $j = 1,...,m$, be the units in the last hidden layer of a neural net, $z_{it} = (z_{it,1},...,z_{it,m})^\top$. Similarly to GLMM, to account for within-subject dependence, we propose to link the conditional mean of $y_{it}$ given $x_{it}$, $\xi_{it} = \mathbb{E}(y_{it}|x_{it})$, to the predictors $z_{it,j}$ as follows

$$g(\xi_{it}) = \beta_0 + \alpha_{i0} + (\beta_1 + \alpha_{i1})z_{it,1} + ... + (\beta_m + \alpha_{im})z_{it,m} = \mathfrak{N}(x_{it}, w, \beta + \alpha_i), \tag{3.6}$$

where $\alpha_i = (\alpha_{i0},...,\alpha_{im})^\top$ are random effects that reflect the characteristics of subject $i$. The variation between the subjects is captured in the distribution of $\alpha_i$. Our paper assumes $\alpha_i \sim \mathcal{N}(0,\Gamma)$ but more flexible distributional specifications for the random effects can also be considered.

Similarly to the previous section, a more interpretable model can be developed if some additional structure is assumed. Partition the covariates $x_{it}$ as $({x_{it}^{(1)}}^\top,{x_{it}^{(2)}}^\top)^\top$ where $x_{it}^{(1)}$ and $x_{it}^{(2)}$ are expected to have non-linear and linear effects respectively on $g(\xi_{it})$:

$$g(\xi_{it}) \;=\; \mathfrak{N}(x_{it}^{(1)}, w, \beta^{(1)}) + (\alpha_i + \beta^{(2)})^\top x_{it}^{(2)}, \tag{3.7}$$

where $\beta^{(1)}$ parametrizes fixed non-linear effects for $x_{it}^{(1)}$, and $\beta^{(2)}$ and $\alpha_i$ are fixed and random linear effects w.r.t. the covariates $x_{it}^{(2)}$. Write $\beta = (\beta^{(1)},\beta^{(2)})$. The model parameters $\theta$ include $w$, $\beta$, $\Gamma$ and any dispersion parameters $\phi$ if unknown. We note that this partially linear approximation have been wide used in classical

non-parametric methods, e.g. see Härdle and Turlach (1992) for a comprehensive discussion on this topic. We refer to the panel data model with the distribution (3.2) and the link (3.6) or (3.7) as DeepGLMM.

The likelihood for the DeepGLMM is equation (3.5) with the $i$th likelihood contribution

$$\begin{aligned} L_i(\theta) = p(y_i|x_i,\theta) &= \int p(y_i|x_i,w,\beta,\phi,\alpha_i)p(\alpha_i|\Gamma)d\alpha_i \\ &= \int \prod_{t=1}^{T_i} p(y_{it}|z_{it},\beta,\phi,\alpha_i)p(\alpha_i|\Gamma)d\alpha_i. \end{aligned} \tag{3.8}$$

Section 3.4 describes a variational approximation algorithm for fitting the DeepGLMM.

The likelihood for the DeepGLMM described in (3.5) and (3.8) is intractable, because the integral in (3.8) cannot be computed analytically, except for the case where the conditional distribution of $y_{it}$ given $z_{it}$ is normal. However, we can estimate each likelihood contribution $L_i(\theta)$ unbiasedly using importance sampling, and this allows estimation methods for intractable likelihoods, such as the block pseudo-marginal MCMC of Tran et al. (2016a) or the variational approximation method of Tran et al. (2017), to be used. We consider here an alternative method based on the reparametrization trick in Section 3.4.1, which utilizes the information of the gradient of the log-likelihood computed by the back-propagation algorithm (Goodfellow et al., 2016). By Fisher's identity (Douc, 2013), the gradient of the log-likelihood contribution $\nabla_\theta \ell_i(\theta)$, $\ell_i(\theta) = \log L_i(\theta)$ with $L_i(\theta)$ in (3.8), is

$$\nabla_\theta \ell_i(\theta) = \int \nabla_\theta \left( \log \prod_{t=i}^{T_i} p(y_{it}|z_{it},\beta,\phi,\alpha_i)p(\alpha_i|\Gamma) \right) p(\alpha_i|\theta,y_i,x_i)d\alpha_i, \tag{3.9}$$

where $p(\alpha_i|\theta,y_i,x_i)$ is the conditional distribution of the random effects $\alpha_i$ given data $(y_i,x_i)$ and $\theta$. The gradient inside the integral (3.9) can be computed by back-propagation, and then the integral can be estimated easily by importance sampling.

## 3.4 Gaussian variational approximation with factor covariance structure

This section describes the NAtural gradient Gaussian Variational Approximation with factor Covariance method (NAGVAC) for approximate Bayesian inference in high-dimensional models. We note that this estimation method can be used for training

other high-dimensional models rather than the DeepGLM and DeepGLMM described in Section 3.3. Let $D$ be the data and $\theta \in \Theta$ the vector of unknown parameters. Bayesian inference about $\theta$ is based on the posterior distribution with density function $\pi(\theta) = p(\theta|D) = p(\theta)L(\theta)/p(D)$, with $p(\theta)$ the prior, $L(\theta) = p(D|\theta)$ the likelihood function and $p(D) = \int p(\theta)L(\theta)d\theta$ the marginal likelihood. In all but a few simple cases the posterior $\pi(\theta)$ is unknown, partly because $p(D)$ is unknown, which makes it challenging to carry out Bayesian inference.

In this work, we are interested in variational approximation (sometimes called variational Bayes (VB)) methods, which are widely used as a scalable and computationally effective method for Bayesian computation (Bishop, 2006; Blei et al., 2017). We will approximate the posterior $\pi(\theta)$ by a Gaussian distribution with density $q_\lambda(\theta) = \mathcal{N}(\theta;\mu,\Sigma)$, the density of a multivariate normal distribution with mean vector $\mu$ and covariance matrix $\Sigma$. The optimal variational parameter $\lambda = (\mu,\Sigma)$ is chosen by minimizing the Kullback-Leibler divergence between $q_\lambda(\theta)$ and $\pi(\theta)$

$$\begin{aligned} \mathrm{KL}(\lambda) &= \int q_\lambda(\theta) \log \frac{q_\lambda(\theta)}{\pi(\theta)} d\theta = \int q_\lambda(\theta) \log \frac{q_\lambda(\theta)}{p(\theta)L(\theta)} d\theta + \log p(D) \\ &= -\mathrm{LB}(\lambda) + \log p(D), \end{aligned} \tag{3.10}$$

where

$$\mathrm{LB}(\lambda) = \int q_\lambda(\theta) \log \frac{p(\theta)L(\theta)}{q_\lambda(\theta)} d\theta \tag{3.11}$$

is a lower bound on $\log p(D)$. Minimizing $\mathrm{KL}(\lambda)$ is therefore equivalent to maximizing the lower bound $\mathrm{LB}(\lambda)$. If we can obtain an unbiased estimator $\widehat{\nabla_\lambda \mathrm{LB}(\lambda)}$ of the gradient of the lower bound, then we can use stochastic optimization to maximize $\mathrm{LB}(\lambda)$, as in Algorithm 3.4 below. The learning rate sequence $\{a_t\}$ in Algorithm 3.4

**Algorithm 3.1** Algorithm 1

- Initialize $\lambda^{(0)}$ and stop the following iteration if the stopping criterion is met.
- For $t = 0,1,...$, compute $\lambda^{(t+1)} = \lambda^{(t)} + a_t \widehat{\nabla_\lambda \mathrm{LB}}(\lambda^{(t)})$.

should satisfy the Robbins-Monro conditions, $a_t > 0$, $\sum_t a_t = \infty$ and $\sum_t a_t^2 < \infty$ (Robbins and Monro, 1951). The choice of $a_t$ is discussed later on in some detail.

### 3.4.1 Reparametrization trick

As is typical of stochastic optimization algorithms, the performance of Algorithm 3.4 depends greatly on the variance of the noisy gradient so that variance reduction methods are needed. We will use the so-called reparametrization trick (Kingma and Welling, 2013; Rezende et al., 2014) in this paper, and its modification by Roeder et al. (2017), who generalized ideas considered in Han et al. (2016) and Tan and Nott (2017).

Suppose that for $\theta \sim q_\lambda(\cdot)$, there exists a deterministic function $g(\lambda,\epsilon)$ such that $\theta = g(\lambda,\epsilon) \sim q_\lambda(\theta)$ where $\epsilon \sim p_\epsilon(\cdot)$, which is independent of $\lambda$. For example, if $q_\lambda(\theta) = \mathcal{N}(\theta;\mu,\Sigma)$ then $\theta = \mu + \Sigma^{1/2}\epsilon$ with $\epsilon \sim \mathcal{N}(0,I)$ and $I$ is the identity matrix. Writing $\mathrm{LB}(\lambda)$ as an expectation with respect to $p_\epsilon(\cdot)$ gives

$$\mathrm{LB}(\lambda) = \mathbb{E}_\epsilon\Big(h(g(\epsilon,\lambda)) - \log q_\lambda(g(\epsilon,\lambda))\Big),$$

where $\mathbb{E}_\epsilon(\cdot)$ denotes expectation with respect to $p_\epsilon(\cdot)$, $h(\theta) := \log(p(\theta)L(\theta))$. Differentiating under the integral sign and simplifying as in Roeder et al. (2017) gives

$$\nabla_\lambda \mathrm{LB}(\lambda) = \mathbb{E}_\epsilon\Big(\nabla_\lambda g(\lambda,\epsilon)\nabla_\theta\{h(g(\epsilon,\lambda)) - \log q_\lambda(g(\epsilon,\lambda))\}\Big). \tag{3.12}$$

The gradient (3.12) can be estimated unbiasedly using i.i.d samples $\epsilon_s \sim p_\epsilon(\cdot)$, $s = 1,\ldots,S$, as

$$\widehat{\nabla_\lambda \mathrm{LB}}(\lambda) = \frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda g(\lambda,\epsilon_s)\nabla_\theta\big\{h(g(\lambda,\epsilon_s)) - \log q_\lambda(g(\lambda,\epsilon_s))\big\}. \tag{3.13}$$

The gradient estimator (3.13) has the advantage that if the variational family is rich enough to contain the exact posterior, so that $\exp(h(\theta)) \propto q_\lambda(\theta)$ at the optimal $\lambda$, then the estimator (3.13) is exactly zero at this optimal value even for $S = 1$ where we use just a single Monte Carlo sample from $p_\epsilon(\epsilon)$. Reparametrized gradient estimators are often more efficient than alternative approaches to estimating the lower bound gradient, partly because they take into account information from $\nabla_\theta h(\theta)$. For further discussion we refer the reader to Roeder et al. (2017).

### 3.4.2 Natural gradient

It is well-known that the ordinary gradient $\nabla_\lambda \mathrm{LB}(\lambda)$ does not adequately capture the geometry of the approximating family $q_\lambda(\theta)$ (Amari, 1998). A small Euclidean distance between $\lambda$ and $\lambda'$ does not necessarily mean a small Kullback-Leibler divergence

between $q_\lambda(\theta)$ and $q_{\lambda'}(\theta)$. Rao (1945) was the first to point out the importance of information on the geometry of the manifold of a statistical model and introduced the Riemannian metric on this manifold induced by the Fisher information matrix. Amari (1998) shows that the steepest direction for optimizing the objective function $\text{LB}(\lambda)$ on the manifold formed by the family $q_\lambda(\theta)$ is directed by the so-called natural gradient which is defined by pre-multiplying the ordinary gradient with the inverse of the Fisher information matrix

$$\nabla_\lambda \text{LB}(\lambda)^{\text{nat}} = I_F^{-1}(\lambda)\nabla_\lambda \text{LB}(\lambda), \tag{3.14}$$

with $I_F(\lambda) = \text{cov}_{q_\lambda}(\nabla_\lambda \log q_\lambda(\theta))$.

The use of the natural gradient in VB algorithms is considered, among others, by Sato (2001), Honkela et al. (2010), Hoffman et al. (2013), Salimans and Knowles (2013) and Tran et al. (2017). A simple demonstration of the importance of the natural gradient can be found in Tran et al. (2017). The use of the natural gradient in deep learning problems is considered in Pascanu and Bengio (2014), who show the connection between natural gradient descent and other second-order optimization methods such as Hessian-free optimization.

The main difficulty of using the natural gradient is the computation of $I_F(\lambda)$, and the solution of linear systems involving this matrix, which is required to compute (3.14). The problem is more severe in high dimensional models because this matrix often has a large size. Some approximation methods, such as the truncated Newton approach, are needed (Pascanu and Bengio, 2014). We consider in the next section an efficient method for computing $I_F(\lambda)^{-1}\nabla_\lambda \text{LB}(\lambda)$ based on the use of iterative conjugate gradient methods for solving linear systems when the covariance matrix of the Gaussian variational approximation is parametrized by a factor model. We compute (3.14) by solving the linear system $I_F(\lambda)x = \nabla_\lambda \text{LB}(\lambda)$ for $x$ using only matrix-vector products involving $I_F(\lambda)$, where the matrix vector products can be done efficiently both in terms of computation time and memory requirements by using the factor structure of the variational covariance matrix. In the special cases of one factor the natural gradient in (3.14) can be computed analytically and efficiently.

### 3.4.3 Gaussian variational approximation with factor covariance

We now describe in detail the Gaussian variational approximation with factor covariance (VAFC) method of Ong et al. (2017a). The VAFC method considers the

multivariate normal variational family $q_\lambda(\theta) = \mathcal{N}(\mu, \Sigma)$, where $\Sigma$ is parametrized as

$$\Sigma = BB^\top + D^2. \tag{3.15}$$

The factor loading matrix $B$ is of size $d \times f$, where $d$ is the dimension of $\theta$ and $f$ the number of factors, $f \ll d$, and $D$ is diagonal with diagonal entries $c = (c_1,...,c_d)^\top$. $c$ is a vector of idiosyncratic noise standard deviations. Factor structures are well known to provide useful parsimonious representations of dependence in high-dimensional settings (Bartholomew et al., 2011). We assume $B$ is lower triangular, i.e., $B_{ij} = 0$ for $j > i$. Although imposing the constraint $B_{ii} > 0$ makes the factor representation identifiable (Geweke and Zhou, 1996), we do not impose this constraint to simplify the optimization. The variational optimization simply locks onto one of the equivalent modes. An intuitive generative representation of the factor structure that is the basis of our application of the reparametrization trick is the following: if we consider $\theta \sim q_\lambda(\theta) = \mathcal{N}(\mu, BB^\top + D^2)$, then we can represent $\theta$ as $\theta = \mu + B\epsilon_1 + c \circ \epsilon_2$ where $\epsilon = (\epsilon_1^\top, \epsilon_2^\top)^\top \sim \mathcal{N}(0, I)$, $\epsilon_1$ and $\epsilon_2$ have dimensions $f$ and $d$ respectively, and $\circ$ denotes the Hadamard (element by element) product for two matrices of the same size. We can see from this representation that the latent variables $\epsilon_1$ (the "factors", which are low-dimensional) explain all the correlation between the components, whereas component-specific idiosyncratic variance is being captured through $\epsilon_2$.

The variational parameters are $\lambda = (\mu^\top, \mathrm{vec}(B)^\top, c^\top)^\top$, where we have written $\mathrm{vec}(B)$ for the vectorization of $B$ obtained by stacking its columns from left to right. Ong et al. (2017a) show that the gradient of the lower bound takes the form

$$\nabla_\mu \mathrm{LB}(\lambda) = \mathbb{E}_\epsilon\Big(\nabla_\theta h(\mu + B\epsilon_1 + c \circ \epsilon_2) + (BB^\top + D^2)^{-1}(B\epsilon_1 + c \circ \epsilon_2)\Big), \tag{3.16}$$

$$\nabla_B \mathrm{LB}(\lambda) = \mathbb{E}_\epsilon\Big(\nabla_\theta h(\mu + B\epsilon_1 + c \circ \epsilon_2)\epsilon_1^\top + (BB^\top + D^2)^{-1}(B\epsilon_1 + c \circ \epsilon_2)\epsilon_1^\top\Big), \tag{3.17}$$

$$\nabla_c \mathrm{LB}(\lambda) = \mathbb{E}_\epsilon\Big(\mathrm{diag}\big(\nabla_\theta h(\mu + B\epsilon_1 + c \circ \epsilon_2)\epsilon_2^\top + (BB^\top + D^2)^{-1}(B\epsilon_1 + c \circ \epsilon_2)\epsilon_2^\top\big)\Big), \tag{3.18}$$

where we have written $\mathrm{diag}(A)$ for the diagonal elements of a square matrix $A$. Here, the inverse matrix $(BB^\top + D^2)^{-1}$ can be computed efficiently; see (3.19). We note that in the expression for the gradient of $B$ above, we should set to zero the upper triangular components which correspond to elements of $B$ which are fixed at zero. Unbiased estimation of gradients for stochastic gradient ascent can proceed based on these expressions by drawing one or more samples from $p_\epsilon(\cdot)$ to estimate the expectations.

### 3.4.4 Efficient natural gradient VAFC method

We now describe how to efficiently compute the natural gradient (3.14) by leveraging the factor structure (3.15). Ong et al. (2017b) also considered a natural gradient method for Gaussian variational approximation with factor covariance. However, this was in the context of likelihood-free inference methods where the dimension of $\theta$ is low compared to the models of interest here, and they simply used naive methods for solving the linear systems involving $I_F(\lambda)$ required to compute the natural gradient. This is impractical in high-dimensional problems, and here we demonstrate how to implement natural gradient VAFC when $\theta$ is high-dimensional using conjugate gradient methods (see, for example, Stoer (1983)).

Write $I_F(\lambda)$ in partitioned form as

$$I_F(\lambda) = \begin{bmatrix} I_{11} & I_{21}^\top & I_{31}^\top \\ I_{21} & I_{22} & I_{32}^\top \\ I_{31} & I_{32} & I_{33} \end{bmatrix},$$

where the blocks in the partition follow the partition of $\lambda$ as $\lambda = (\mu^\top, \mathrm{vec}(B)^\top, c^\top)^\top$. Because the upper triangle of $B$ is fixed at zero the corresponding rows and columns of $I_F(\lambda)$ should be omitted. Ong et al. (2017b) show that $I_{11} = \Sigma^{-1}$, $I_{21} = I_{31} = 0$, $I_{22} = 2(B^\top \Sigma^{-1} B \otimes \Sigma^{-1})$ (where $\otimes$ denotes the Kronecker product), $I_{33} = 2(D\Sigma^{-1}) \circ (\Sigma^{-1} D)$ and $I_{32} = 2(B^\top \Sigma^{-1} D \otimes \Sigma^{-1}) E_d^\top$, where $E_d$ is the $d \times d^2$ matrix that picks out the diagonal elements of the $d \times d$ matrix $A$ from its vectorization, so that $E_d \mathrm{vec}(A) = \mathrm{diag}(A)$. To use a conjugate gradient linear solver to compute $I_F(\lambda)^{-1} \nabla_\lambda \mathrm{LB}(\lambda)$ we simply need to be able to compute matrix vector products of the form $I_F(\lambda)x$ for any vector $x$ quickly without needing to store the elements of $I_F(\lambda)$.

This can be done provided we can do matrix vector products for the matrices $I_{11}$, $I_{22}$, $I_{33}$ and $I_{32}$. Except for the one-factor case described below, this is still difficult so we further approximate $I_F(\lambda)$ by setting $I_{32} = 0$ and replacing $I_{33}$ with $\tilde{I}_{33} = 2(D\tilde{\Sigma}^{-1}) \circ (\tilde{\Sigma}^{-1} D)$, where $\tilde{\Sigma}$ is the diagonal approximation to $\Sigma$ obtained by setting its off-diagonal elements to zero. Using this approximation and $I_{32} = 0$ we obtain a positive definite approximation $\tilde{I}_F(\lambda)$ to $I_F(\lambda)$ which we use instead of $I_F(\lambda)$ in the natural gradient. We note that these approximations do not affect the factor structure of $\Sigma$ in (3.15).

Multiplications involving $\tilde{I}_{33}$ are simple since this matrix is diagonal, but we still need efficient methods to compute matrix vector products for $I_{11}$ and $I_{22}$. Considering $I_{11}$

first, we note that by using the Woodbury formula

$$I_{11} = \Sigma^{-1} = D^{-2} - D^{-2}B(I + B^\top D^{-1}B)^{-1}B^\top D^{-2}, \tag{3.19}$$

and then noting that $D$ is diagonal and $(I+B^\top D^{-2}B)$ is $f \times f$, $f \ll d$, we can calculate $I_{11}x = \Sigma^{-1}x$ without needing to store any $d \times d$ matrix or do any dense $d \times d$ matrix multiplications. Next, consider $I_{22}x$ for some vector $x$. We note that

$$I_{22} = 2(B^\top \Sigma^{-1} B \otimes \Sigma^{-1})x = 2\text{vec}(\Sigma^{-1}x^* B^\top \Sigma^{-1} B),$$

where $x^*$ denotes the $d \times f$ matrix such that $x = \text{vec}(x^*)$ and where we have used the identity that for conformable matrices $X$,$Y$,$Z$, $\text{vec}(XYZ) = (Z^\top \otimes X)\text{vec}(Y)$. Then $\Sigma^{-1}x^*$ is computed efficiently by the Woodbury formula, and similarly for $B^\top \Sigma^{-1} B$. We refer to our natural gradient estimation method with $f$ factors as NAGVAC-$f$.

We now consider the special case of the NAGVAC-1 method where the covariance matrix $\Sigma$ is parameterized as in (3.15) with $B$ a vector. In this case, the natural gradient (3.14) can be computed efficiently and the computational complexity is $O(d)$. See Algorithm 3.2, whose detailed derivation is in the Appendix A. This estimation method is computationally attractive, especially when the dimension $d$ is extremely large. The experimental studies in Section 3.7 suggest that in some applications this method is able to produce a prediction accuracy comparable to the accuracy obtained by methods that use more flexible factor decomposition structures of $\Sigma$. Trippe and Turner (2018) discuss the phenomenon where richer variational families produce inferior performance in terms of predictive loss for neural networks models, and provide some theoretical insights into this phenomenon.

## 3.5 Variable selection and regularization priors

This section first presents a method for variable selection in DeepGLM and DeepGLMM. The method is based on the Bayesian adaptive group Lasso method which is developed for GLM in Leng et al. (2014) and normal regression in Kyung et al. (2010). Consider a neural network as in Figure 2.1. Denote by $w_{X_j}$ the vector of weights that connect the covariate $X_j$ to the $m$, say, units in the first hidden layer. We use the following priors

$$w_{X_j}|\tau_j \sim \mathcal{N}(0, \tau_j I_m), \quad \tau_j|\gamma_j \sim \text{Gamma}\left(\frac{m+1}{2}, \frac{\gamma_j^2}{2}\right), \quad j = 1, ..., p,$$

**Algorithm 3.2** Computing the natural gradient

Input: Vector $B$, $c$ and ordinary gradient vector $g = (g_1^\top, g_2^\top, g_3^\top)^\top$ with $g_1$ the vector formed by the first $d$ elements of $g$, $g_2$ formed by the next $d$ elements, and $g_3$ the last $d$ elements. Output: The natural gradient $g^{\text{nat}} = I_F^{-1} g$.

- Compute the vectors $v_1 = c^2 - 2B^2 \circ c^{-4}$, $v_2 = B^2 \circ c^{-3}$, and the scalars $\kappa_1 = \sum_{i=1}^d b_i^2 / c_i^2$, $\kappa_2 = \frac{1}{2}(1 + \sum_{i=1}^d v_{2i}^2 / v_{1i})^{-1}$.
- Compute

$$g^{\text{nat}} = \begin{pmatrix} (g_1^\top B)B + c^2 \circ g_1 \\ \frac{1+\kappa_1}{2\kappa_1}\left((g_2^\top B)B + c^2 \circ g_2\right) \\ \frac{1}{2} v_1^{-1} \circ g_3 + \kappa_2 \left[(v_1^{-1} \circ v_2)^\top g_3\right](v_1^{-1} \circ v_2) \end{pmatrix}.$$

with the $\gamma_j > 0$ the shrinkage parameters. By the normal-Gamma mixture (Andrews and Mallows, 1974; Kyung et al., 2010), we have that

$$p(w_{X_j}|\gamma_j) = \int p(w_{X_j}|\tau_j) p(\tau_j|\gamma_j) d\tau_j \propto \exp\left(-\gamma_j \|w_{X_j}\|_2\right)$$

with $\|w_{X_j}\|_2$ the $l_2$-norm of $w_{X_j}$. Hence, the posterior mode of the $w_{X_j}$ induced from the above hierarchical prior with the same $\gamma_j = \gamma$ is equivalent to the group Lasso estimator of Yuan and Lin (2006).

Selecting the adaptive shrinkage parameters $\gamma_j$ is challenging. We develop an empirical Bayes method for estimating these tuning parameters based on an iterative scheme within the variational approximation procedure. Writing the Bayesian hierarchical model in the generic form

$$y|\psi, \theta \sim p(y|\theta, \psi), \quad \theta|\psi \sim p(\theta|\psi),$$

with $y$ the data, $\theta$ the model parameters and $\psi$ the hyperparameters to be selected. The marginal likelihood for $\psi$ is $p(y|\psi) = \int p(\theta|\psi) p(y|\theta, \psi) d\theta$, which can be maximized using an EM-type algorithm (Casella, 2001). Given an initial value $\psi^{(0)}$, we iteratively update $\psi$ by

$$\psi^{(k+1)} = \text{argmax}_\psi \left\{ \mathbb{E}_{\theta|y,\psi^{(k)}} \log p(y, \theta|\psi) \right\},$$

where $\mathbb{E}_{\theta|y,\psi^{(k)}}(\cdot)$ is the expectation with respect to the posterior distribution $p(\theta|y, \psi^{(k)})$. It can be shown that the updating rule for $\gamma_j$ is

$$\gamma_j^{(k+1)} = \sqrt{\frac{m+1}{\mathbb{E}_{\theta|y,\gamma_j^{(k)}}[\tau_j]}}. \tag{3.20}$$

In our variational approximation framework, the expectation $\mathbb{E}_{\theta|y,\psi^{(k)}}(\cdot)$ can be naturally approximated by the expectation w.r.t. the current variational approximation $q_{\lambda^{(k)}}(\theta)$, and the updates in (3.20) can be computed in closed form. One can merge the auxiliary parameters $\tau_1,...,\tau_p$ into the model parameters and learn them jointly by the Gaussian variational posterior $q_\lambda(\theta)$. Alternatively, one can conveniently update the variational posterior for each $\tau_j$ separately in a fixed-form within mean-field variational approximation procedure; see, e.g. Tran et al. (2016b). We use the latter in this paper. The optimal VB posterior for $1/\tau_j$ is inverse-Gaussian with mean and shape parameter

$$\alpha_{\tau_j} \leftarrow \frac{\gamma_j}{\sqrt{\mathbb{E}_{q_\lambda}[w'_{X_j} w_{X_j}]}}, \quad \beta_{\tau_j} \leftarrow \gamma_j^2. \tag{3.21}$$

Because $\mathbb{E}_{\theta|y,\gamma_j^{(k)}}[\tau_j]$ can be approximated by $\mathbb{E}_{q_\lambda}[\tau_j] = 1/\alpha_{\tau_j} + 1/\beta_{\tau_j}$, the update in (3.20) becomes

$$\gamma_j \leftarrow \sqrt{\frac{m+1}{1/\alpha_{\tau_j} + 1/\beta_{\tau_j}}}. \tag{3.22}$$

The updates in (3.21) and (3.22) are then embedded in the main variational iterate procedure Algorithm 1. This leads to a convenient and principled way for selecting the shrinkage parameters $\gamma_j$, which would be very challenging for alternative methods.

In order to control overfitting, we use a ridge-type regularization for the rest of the weights in the neural network. Let $\widetilde{w}$ be the vector of all the weights except those that connect the input layer to the first hidden layer and the intercepts (also called bias terms). For $\widetilde{w}$, we use the prior $p(\widetilde{w}) \propto \exp\left(-\frac{\gamma_w}{2}\widetilde{w}^\top \widetilde{w}\right)$, with $\gamma_w$ the shrinkage parameter. Similarly to above, $\gamma_w$ is selected by maximizing the marginal likelihood and the update rule is

$$\gamma_w \leftarrow \frac{d_{\widetilde{w}}}{\mathbb{E}_{q_\lambda}[\widetilde{w}^\top \widetilde{w}]}, \tag{3.23}$$

where $d_{\widetilde{w}}$ denotes the dimension of $\widetilde{w}$.

## 3.6 Practical recommendations

The NAGVAC estimation method can be used as a general estimation method for any model. However, this section focuses on estimating DFNN-based models, and discusses some implementation recommendations in training DeepGLM and DeepGLMM that we found useful in practice.

### 3.6.1 Stopping rule and lower bound for model choice

It is common in deep learning applications of neural network methods to implement early stopping in the optimization to avoid overfitting, because often the training loss decreases steadily over the optimization updates, but the validation loss starts increasing at some point. In our VB framework, the lower bound (3.11) can be monitored to check the convergence and decide when to stop training. There are two advantages to using the lower bound. The first advantage is that a validation set is unnecessary, which leaves more information for the training phrase. Also, a stopping rule based on a validation set might depend on how the validation set is selected. The second advantage is that, given that appropriate regularization priors on the weights as the ones in Section 3.5 have been used to control overfitting, the maximized lower bound can be used for model choice; see the examples in Section 3.7. The lower bound (3.11) can be estimated efficiently using the same sample of $\theta$ generated for computing the gradient vector. To reduce the noise level in estimating the lower bound, we follow Tran et al. (2017) and take the average of the lower bound over a moving window of $K$ iterations. We stop training if this moving averaged lower bound does not improve after $P$ iterations.

### 3.6.2 Learning rate and the momentum method

We employ the following fixed learning rate which is widely used in the deep learning literature

$$a_t = \epsilon_0 \frac{\tau}{t}, \tag{3.24}$$

where $\epsilon_0$ is a small value, e.g. 0.01 and $\tau$ is some threshold from which the learning rate is reduced, e.g. $\tau = 1000$. It might also be useful to use some adaptive learning rate methods, and later we consider one method adapted from Ranganath et al. (2013) which is described in Ong et al. (2018). This learning rate is suitable for use with the natural gradient.

As a method of accelerating the stochastic gradient optimization we also consider using the momentum method (Polyak, 1964). The update rule is

$$\begin{aligned} \overline{\nabla_\lambda \mathrm{LB}} &= \alpha_\mathrm{m} \overline{\nabla_\lambda \mathrm{LB}} + (1-\alpha_\mathrm{m}) \widehat{\nabla_\lambda \mathrm{KL}}(\lambda^{(t)})^{\mathrm{nat}}, \\ \lambda^{(t+1)} &= \lambda^{(t)} + a_t \overline{\nabla_\lambda \mathrm{LB}}, \end{aligned}$$

where $\alpha_{\mathrm{m}} \in [0,1]$ is the momentum weight. The use of the moving average gradient $\overline{\nabla_\lambda \mathrm{LB}}$ helps remove some of the noise inherent in the estimated gradients of the lower bound. See Goodfellow et al. (2016), Chapter 8, for a detailed discussion on the usefulness of the momentum method.

### 3.6.3 Activation function and initialization

For initialization of the variational parameters $\lambda^{(0)} = (\mu^{(0)}, B^{(0)}, c^{(0)})$, we follow Glorot and Bengio (2010) and initialize each weight in $\mu^{(0)}$ by the uniform distribution $\mathcal{U}(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}})$, where the weight connects a layer with $m$ units to a layer with $n$ units. The elements in $B^{(0)}$ are initialized by $\mathcal{N}(0, 0.01^2)$ and the elements in $c^{(0)}$ are initialized by 0.01. It is advisable to first standardize the input data so that each column has a zero sample mean and a standard deviation of one. Finally, we use the rectified activation function $h(x) = \max(0, x)$ in all examples, unless otherwise stated. This activation function has a strong connection with biological neuroscience and has been widely used in the deep learning literature; see, e.g., Goodfellow et al. (2016).

## 3.7 Experimental studies and applications

To illustrate the performance of variable selection, the prediction accuracy of DeepGLM and DeepGLMM, and the efficiency of the NAGVAC training algorithm, we consider a range of experimental studies and applications. All the examples are implemented in Matlab and run on a desktop computer with i5 3.3 Ghz Intel Quad Core. All the DeepGLM and DeepGLMM models are trained using the NAGVAC-1 method, unless otherwise stated.

We use two predictive measurements. The first is the partial predictive score (PPS)

$$PPS = -\frac{1}{n_{\text{test}}} \sum_{(x_i, y_i) \in \text{test data}} \log p(y_i | x_i, \widehat{\theta}),$$

with $\widehat{\theta}$ a point estimate of the model parameters. The second is the mean squared error (MSE),

$$MSE = \frac{1}{n_{\text{test}}} \sum_{(x_i, y_i) \in \text{test data}} (y_i - \widehat{y}_i)^2,$$

with $\widehat{y}_i$ a prediction of $y_i$, which is called the misclassification rate (MCR) for binary response $y_i$.

### 3.7.1 Experimental studies

#### Efficiency of the NAGVAC algorithm

We first study the efficiency of the NAGVAC algorithm as a genenal training method. We use the German credit dataset from the UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml`, with 1000 observations, of which 750 are used as the training data and the rest as the test data. The data consist of a binary response variable, credit status, which is good credit (1) or bad credit (0), together with 30 covariate variables such as education, credit amount, employment status, etc. We consider a simple logistic regression model for predicting the credit status, based on the covariates. We use an improper flat prior for the coefficients $\theta$, i.e. $p(\theta) \propto 1$.

We study the performance of the natural gradient method compared to the ordinary gradient method. Figure 3.1 shows the convergence of the lower bound of the ordinary gradient method and the NAGVAC-1 method. For the ordinary gradient, we use the adaptive learning rate method ADADELTA of Zeiler (2012). For the natural gradient, we use both the fixed learning rate in (3.24) and the adaptive learning rate of Ong et al. (2018) (which is based on the method in Ranganath et al. (2013)). The figure shows that both the natural gradient method speed up the convergence significantly. Although incorporating an adaptive learning rate into NAGVAC helps to speed up the convergence, the improvement is not always significant compared to a fixed learning rate. We used the fixed learning rate (3.24) in all the examples reported below.

#### Variable selection and prediction accuracy of DeepGLM: Binary response

Data are generated from the following deterministic model

$$\begin{aligned} a &= 5-2(x_1+2x_2)^2+4x_3x_4+3x_5+\sum_{i=6}^{20}0x_i, \\ y &= 1 \text{ if } a \geq 0, \text{ 0 otherwise,} \end{aligned} \tag{3.25}$$

where the $x_i$ are generated from the uniform distribution $\mathcal{U}(-1,1)$. Note that the last 15 variables are irrelevant variables. The training data consist of $n_{\text{train}} = 100{,}000$ observations and the test data of $n_{\text{test}} = 100{,}000$ observations.

We use a neural net with the structure (20,20,20,1), i.e. the input layer has 20 variables, two hidden layers each has 20 units and one scalar output. Figure 3.2 plots the update

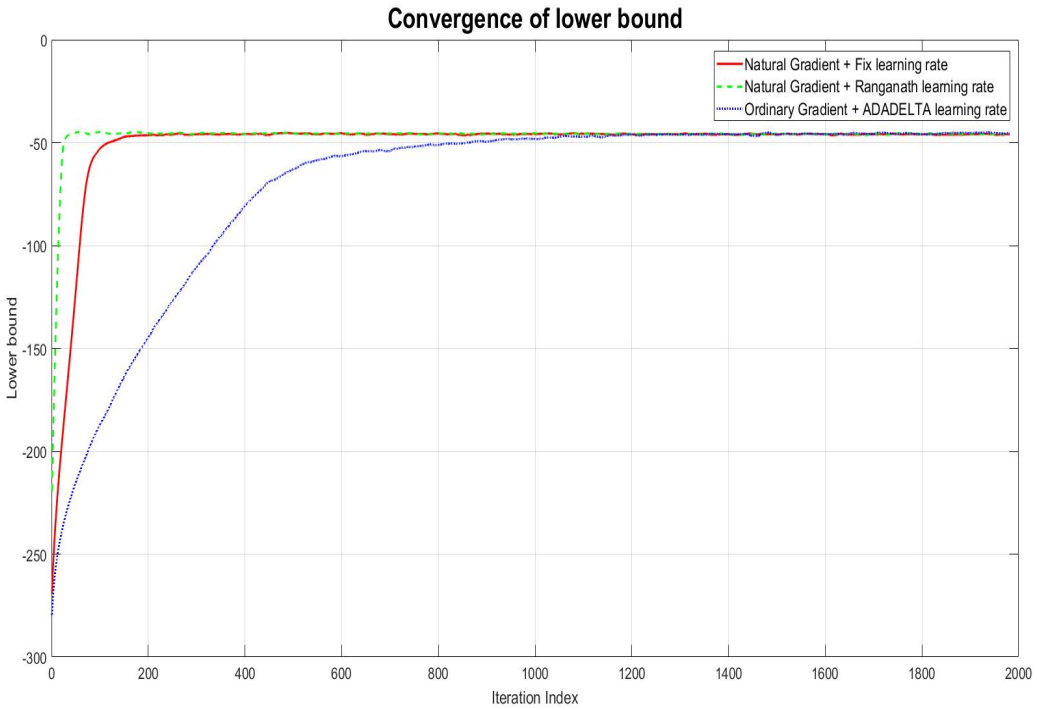


**Figure 3.1.** Plot of the lower bound over iterations for the ordinary gradient and NAGVAC-1 methods.

of the shrinkage parameters $\gamma_j$ as in (3.22). The shrinkage parameters $\gamma_6$,...,$\gamma_{20}$ w.r.t. the irrelevant variables keep increasing over iterations, while the ones w.r.t. the relevant variables keep decreasing. This shows that the Bayesian adaptive group Lasso with the empirical Bayes method for updating the shrinkage parameters is able to scan out correctly irrelevant covariates.

We now compare the predictive performance of DeepGLM to the Bayesian Additive Regression Tree method (BART) of Chipman et al. (2010). BART is a commonly used *nonparametric* regression method in the statistics literature and is well known for its prediction accuracy and its ability to capture nonlinearity effects. Table 3.1 summarizes the prediction performance of DeepGLM compared to that of the conventional GLM and BART. GLM, which is logistic regression in this example, does just slightly better than a random guess with a MCR of 40.84%. As shown, DeepGLM works very well in this example and outperforms both GLM and BART. All the comparisons with BART in this paper are done using the R package `BART` (the lattest version 1.6) with default settings for the tuning parameters.

| Method | PPS | MCR (%) |
|---|---|---|
| GLM | 0.67 | 40.84 |
| BART | 0.08 | 3.09 |
| DeepGLM | 0.03 | 1.03 |

**Table 3.1.** Binary response simulation: Performance of DeepGLM v.s. GLM and BART in term of the partial predictive score (PPS) and the misclassification rate (MCR). Both are evaluated on the test data.

To investigate the performance of DeepGLM in "big data" settings, we generate data from model (3.25) with the last 995 covariates being irrelevant. The training and testing dataset each has 100,000 observations. The results in Table 3.2 show that DeepGLM significantly outperforms its competitors by a large margin. This also shows that DeepGLM is an attractive alternative to BART in big data settings.

| Method | PPS | MCR (%) |
|---|---|---|
| GLM | 0.71 | 41.54 |
| BART | 0.34 | 14.3 |
| DeepGLM | 0.08 | 2.8 |

**Table 3.2.** Binary response simulation with high-dimensional data: Performance of DeepGLM v.s. GLM and BART in term of PPS and MCR. Both are evaluated on the test data and DeepGLM's neural net structure is (1000,30,1).

.

### Variable selection and prediction accuracy of DeepGLM: Continuous response

We generate data from the following highly nonlinear model

$$y = 5 + 10x_1 + \frac{10}{x_2^2+1} + 5x_3x_4 + 2x_4 + 5x_4^2 + 5x_5 + 2x_6 + \frac{10}{x_7^2+1} + 5x_8x_9 + 5x_9^2 + 5x_{10} + \epsilon, \tag{3.26}$$

where $\epsilon \sim \mathcal{N}(0,1)$, $(x_1,...,x_{20})^\top$ are generated from a multivariate normal distribution with mean zero and covariance matrix $(0.5^{|i-j|})_{i,j}$. Note that the last 10 variables are irrelevant variables. The training data has 100,000 observations and the test data has 20,000.

### 3.7.1 Experimental studies

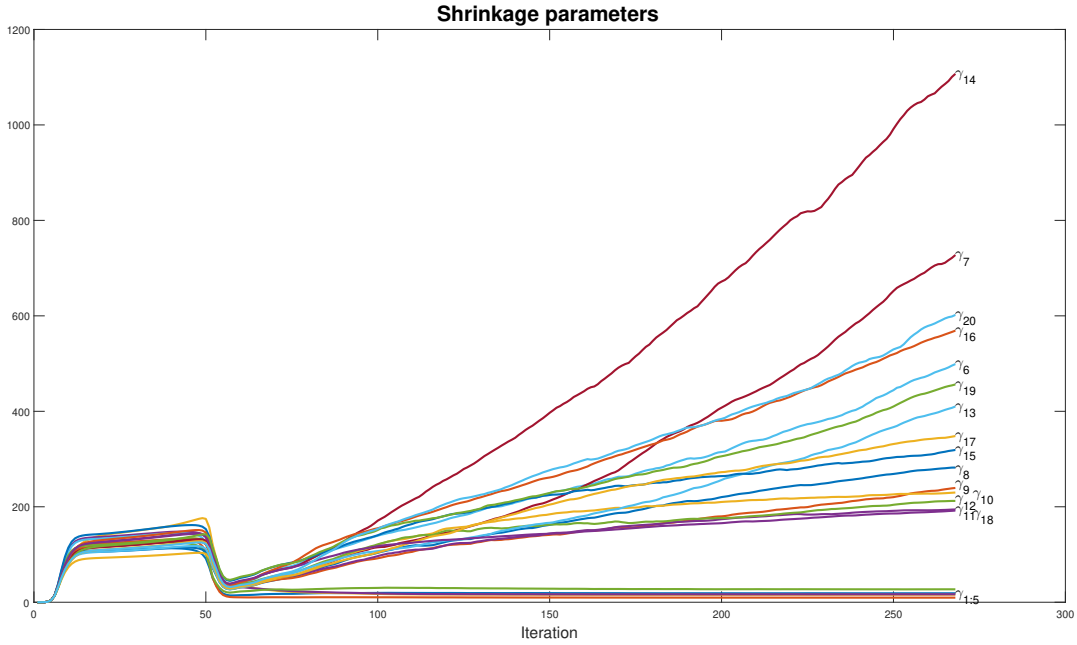


**Figure 3.2.** Plots of the shrinkage parameters $\gamma_j$ over iterations. The shrinkage parameters w.r.t the irrelevant variables keep increasing, while the ones w.r.t the relevant variables keep decreasing.

| Method | PPS | MSE |
|---|---|---|
| GLM | 3.31 | 275.5 |
| BART | 1.49 | 4.89 |
| DeepGLM | 1.20 | 4.07 |

**Table 3.3.** Continuous response simulation: Performance of DeepGLM v.s. conventional GLM and BART in term of the partial predictive score (PPS) and the mean squared error (MSE). Both are evaluated on the test data. The structure of the neural net is (20,20,20,1).

Figure 3.3 plots the update of the shrinkage parameters $\gamma_j$ as in (3.22). All the shrinkage parameters w.r.t. the irrelevant variables keep increasing over iterations, while the ones w.r.t. the relevant variables, except $\gamma_8$, keep decreasing. The reason is that the main effect of $x_8$ is not included into model (3.26), hence the signal of $x_8$ might not be strong enough to be identifiable. Table 3.3 summarizes the prediction performance of DeepGLM in comparison to the conventional GLM and BART. DeepGLM works well in this example and ourperforms both GLM and BART.

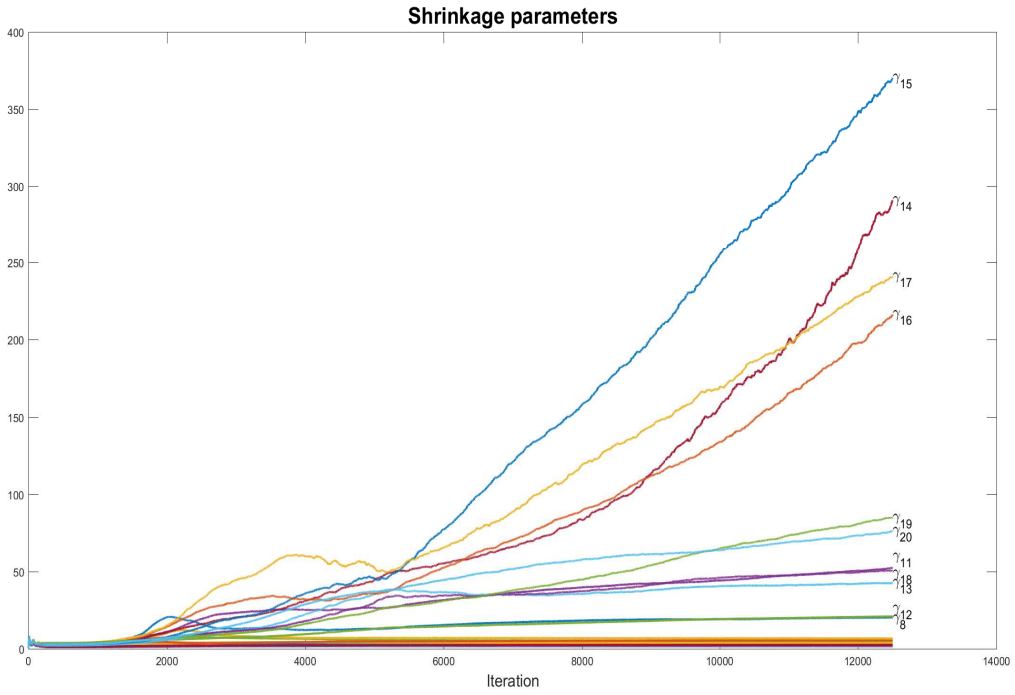


**Figure 3.3.** Plots of the shrinkage parameters $\gamma_j$ over iterations. The shrinkage parameters w.r.t. the irrelevant variables keep increasing, while the ones w.r.t. the relevant variables keep decreasing.

### Prediction accuracy of DeepGLMM: binary panel data simulation

We study the DeepGLMM on a simulation binary panel dataset $D = \{(x_{it},y_{it});\ t = 1,...,20;\ i = 1,...,1000\}$ with $x_{it}$ the vector of covariates and $y_{it}$ the response of subject $i$ at time $t$. The response $y_{it}$ is generated from the following model:

$$\begin{aligned} a_{it} &= 2 + 3(x_{it,1} - 2x_{it,2})^2 - 5\frac{x_{it,3}}{(1+x_{it,4})^2} - 5x_{it,5} + b_i + \epsilon_{it}, \\ y_{it} &= 1 \text{ if } a_{it} \geq 0, 0 \text{ otherwise,} \end{aligned}$$

where $b_i \sim \mathcal{N}(0,0.1)$ is a random intercept representing charactersistics of subject $i$ and $\epsilon_{it} \sim \mathcal{N}(0,1)$ is random noise associated with reponses $y_{it}$. The $x_{it,j}, j = 1,...,5$, are generated from a uniform distribution $\mathcal{U}(-1,1)$.

We fit the following DeepGLMM to this dataset

$$y_{it}|x_{it} \sim \text{Binomial}(1,\mu_{it}), \quad \text{logit}(\mu_{it}) = \mathfrak{N}(x_{it},w,\beta+\alpha_i),$$

where $\mathfrak{N}(x_{it},w,\beta+\alpha_i)$ is the scalar output of a neural net with input $x_{it}$, inner weights $w$ and output weights $\beta+\alpha_i$. We assume that the random effects $\alpha_i \sim \mathcal{N}(0,\Gamma)$ with $\Gamma = \text{diag}(\Gamma_0,...,\Gamma_m)$. The model parameters are $\theta = (w,\beta,\Gamma_0,...,\Gamma_m)$. We use Gamma

priors on the $\Gamma_j$, $\Gamma_j \sim \text{Gamma}(a_0, b_0)$, and set the hyperparameters $a_0 = 1$ and $b_0 = 0.1$ in this example. The Appendix A gives further details on training this model.

For each subject, we use the first 17 observations for training and the last 3 observations for testing. That is, we are interested in the within-subject prediction. The neural network has one hidden layer with 10 nodes, and we compare the DeepGLMM model with the conventional logistic regression model with a random intercept.

For binary panel data, the misclassification rate is defined as follows. Let $\bar{\theta}$ be the mean of the VB approximation $q_\lambda(\theta)$ after convergence. Let $\widehat{\mu}_{\alpha_i}$ in (A.1) be the mode of $p(\alpha_i | y_i, x_i, \bar{\theta})$, which can be used as the prediction of the random effects $\alpha_i$ of subject $i$. Let $(y_{it_0}, x_{it_0})$ be a future data pair. We set the prediction of $y_{it_0}$ as 1 if $\mathfrak{N}(x_{it_0}, w, \beta + \widehat{\mu}_{\alpha_i}) \geq 0$, 0 otherwise. The classification error is

$$\text{MCR} = \sum I_{\widehat{y}_{it_0} \neq y_{it_0}} \,/\, \text{total number of future observations}$$

with the sum over the test data points $(y_{it_0}, x_{it_0})$.

| Model | PPS | MCR |
|---|---|---|
| GLMM | 1.24 | 17.57% |
| DeepGLMM | 0.13 | 5.27% |

**Table 3.4.** Simulation binary panel dataset: Performance of DeepGLMM v.s. GLMM in term of the partial predictive score (PPS) and the misclassification rate (MCR). Both are evaluated on the test data.

Table 3.4 summarizes the performance of DeepGLMM and GLMM. The results show that modelling covariate effects in a flexible way using the neural network basis functions is helpful here in terms of improving both PPS and MCR.

## 3.7.2 Applications

### Direct Marketing data

We consider the Direct Marketing dataset used in the statistics textbook of Jank (2011). This dataset consists of 1000 observations, of which 900 were used for training and the rest for testing. The response is the amount (in \$1000) a customer spends on the company's products per year. There are 11 covariates including gender, income,

the number of ads catalogs, married status, young, old, etc. The careful analysis in Jank (2011) shows that the ordinary linear regression model fits well to this dataset. We first wish to explore the significance of the covariates in terms of explaining the response $y$. We tried many DeepGLM models with one hidden layer neural network and the number of units varying, the plots of the shrinkage parameters over iterations have a consistent pattern and all show that the shrinkage parameters with respect to the covariates married, gender, home owner, old and young increase over iterations. The left panel of Figure 3.4 plots the shrinkage parameters from a DeepGLM with 6 hidden units, with an MSE of 0.2325. The plot suggests that these five covariates can be removed from the model. The right panel of Figure 3.4 plots the shrinkage parameters from a DeepGLM with a neural net structure (6,6,6,1), after the five insignificant covariates have been removed. The MSE for this model is 0.1718. Table 3.5 shows that DeepGLM gives a better predictive performance than its competitors. We note, however, that without considering variable selection, we could not successfully train a DeepGLM model that has a better predictive performance than BART.

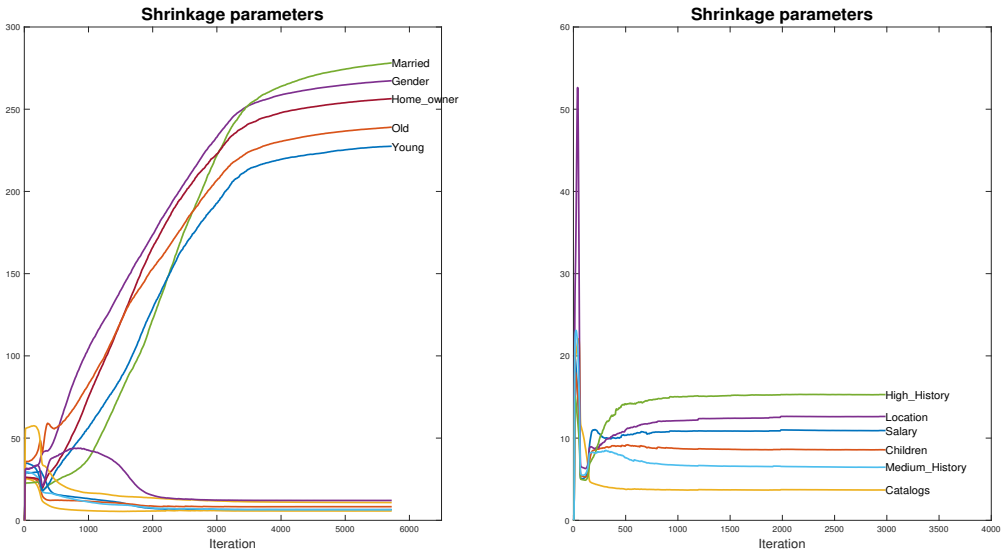


**Figure 3.4.** Plots of the shrinkage parameters over iterations. Left panel: The shrinkage parameters from a neural net with 6 hidden units. Right panel: The shrinkage parameters from a (6,6,6,1) neural net, after the five insignificant covariates have been removed.

| Model | PPS | MSE |
|---|---|---|
| GLM | $-0.11$ | 0.29 |
| BART | $-0.35$ | 0.18 |
| DeepGLM | $-0.38$ | 0.17 |

**Table 3.5.** Direct marketing data after the five insignificant covariates have been removed: Performance of GLM, BART and DeepGLM. Neural net structure is (6,6,6,1).

### Abalone data

The abalone data, available on the UCI Machine Learning Repository, is a benchmark dataset that has been used in many regression analysis papers. The data has 4177 observations of which 85% were used for training and the rest for testing. We first explore the use of the lower bound for model selection. Table 3.6 summarizes the lower bound and MSE (computed on the test data), averaged over 10 different runs, for various neural network structures. Here, we use the MSE on the test data in order to assess the usefulness of the lower bound as a model selection criterion. The results suggest that, in general, a small lower bound leads to a worse MSE and that a DeepGLM with either neural net (9,5,5,1) or (9,10,10,1) can be selected. We conducted the same experiment for other structures (9,6,6,1), (9,7,7,1), (9,8,8,1) and (9,9,9,1) and observed a little change in both LB and MSE. This result is consistent with the findings that have been long realized in the deep learning literature (Bengio, 2012) that a small change around a neural net structure that works well does not affect the predictive performance appreciably. In Table 3.6, the neural nets (9,5,5,1) and (9,10,10,1) have similar LB, but the former should be selected as it has a simpler structure. This experimental exploration illustrates the attractiveness of using the lower bound as a model selection tool in our NAGVAC method.

| Structure | [9,2,2,1] | [9,5,5,1] | [9,10,10,1] | [9,20,20,1] |
|---|---|---|---|---|
| LB | $-2.212(0.019)$ | $-2.193(0.012)$ | $-2.190(0.010)$ | $-2.446(0.060)$ |
| MSE | 5.17 (0.32) | 4.71 (0.12) | 4.74 (0.13) | 8.61 (1.52) |

**Table 3.6.** Abalone data: Lower bound LB and MSE (on the test data), averaged over 10 runs, for various neural network structures. The numbers in brackets are standard errors.

One of the attractive features of DeepGLM is that, as a Bayesian method, it offers an easy and principled way to construct the prediction intervals for test data. The predictive distribution of the response $y$ given covariate vector $x$ and data $D$ is

$$p(y|x,D) = \int p(y|x,\theta)p(\theta|D)d\theta. \tag{3.27}$$

As we approximate the posterior $p(\theta|D)$ by the VB distribution $q_\lambda(\theta)$ which we can sample from, it is possible to sample from $p(y|x,D)$ (assuming that it is easy to sample from $p(y|x,\theta)$, which is the case all of our applications). Based on this sample from the predictive distribution, we can compute prediction intervals for the mean $\mathbb{E}(y|x,D)$. Figure 3.5 shows the one standard deviation prediction intervals for the test data.

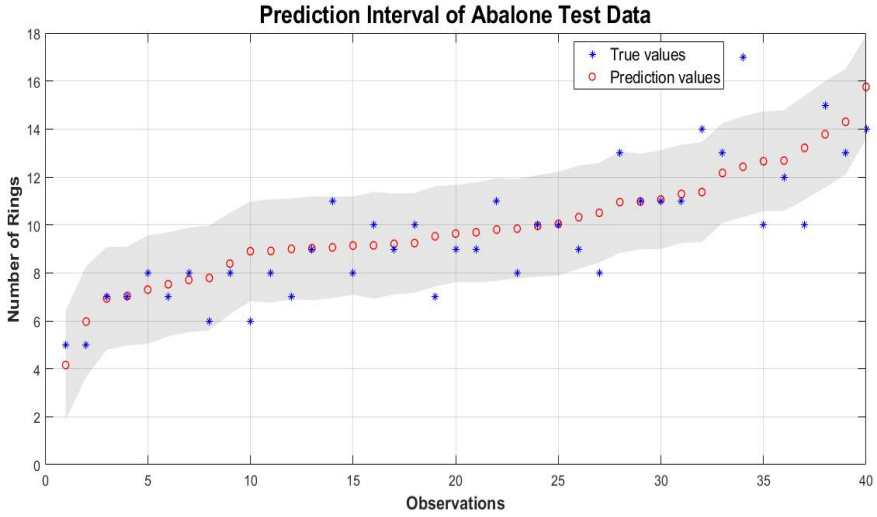


**Figure 3.5.** Abalone data: Point prediction and one standard deviation prediction intervals (the shaded area) calculated from the predictive distribution in (3.27).

| Model | PPS | MSE |
|---|---|---|
| BART | 1.30 (0.01) | 4.88 (0.06) |
| DeepGLM | 1.28 (0.01) | 4.71 (0.12) |

**Table 3.7.** Abalone data: Performance of BART and DeepGLM. The neural net structure is (9,5,5,1). The values are averaged over 10 runs with the standard errors in brackets.

### Census income data

This census dataset was extracted from the U.S. Census Bureau database and is available on the UCI Machine Learning Repository. The prediction task is to determine

whether a person's income is over \$50K per year, based on 14 attributes including age, workclass, race, etc, of which many are categorical variables. After using dummy variables to represent the categorical variables, there are 103 input variables. There are 45221 observations without missing data, of which 33.3% are kept for testing, the rest are used for training. Table 3.8 summarizes the predictive performance and Figure 3.6 plots the ROC curves of DeepGLM and BART, which show that DeepGLM gives slightly better prediction accuracy than BART. Both DeepGLM and BART are run once with a fixed random seed.

| Model | PPS | MCR (%) |
|---|---|---|
| BART | 0.68 | 18.6 |
| DeepGLM | 0.34 | 16.09 |

**Table 3.8.** Census data: Performance of DeepGLM v.s. BART. We use a one hidden layer neural net with 40 units.

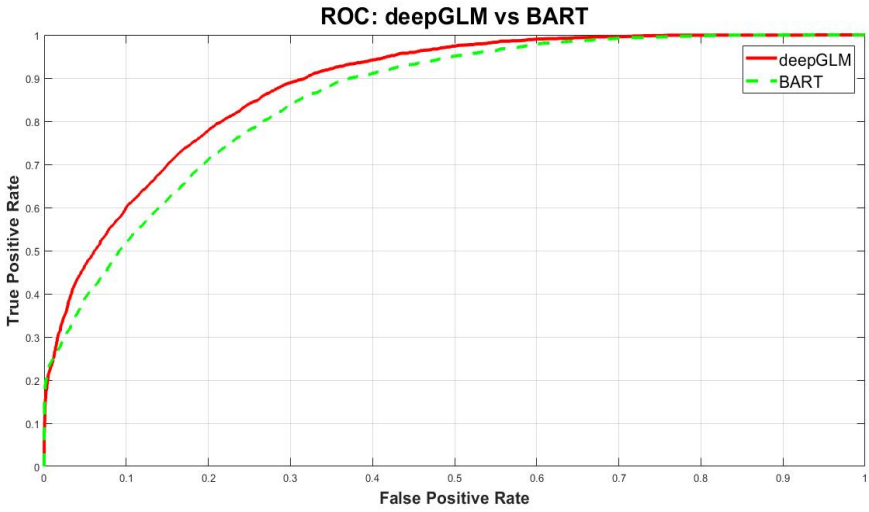


**Figure 3.6.** Census data: The ROC curves of DeepGLM v.s. BART. The area under the curve of DeepGLM is larger than that of BART, which suggests that DeepGLM has a better predictive performance in this example.

### A continuous panel data set: Cornwell and Rupert data

This section analyzes a continuous panel data set originally analyzed in Cornwell and Rupert (1988). This is a balanced panel dataset with 595 individuals and 4165

observations, each individual was observed for 7 years. The dataset is available from the website of the textbook Baltagi (2013). The variables are listed in Table 3.9.

| Variable | Meaning |
|---|---|
| EXP | Years of full-time work experience |
| WKS | Weeks worked |
| OCC | blue-collar occupation = 1; otherwise = 0 |
| IND | manufacturing industry=1; otherwise = 0 |
| SOUTH | south residence =1; otherwise = 0 |
| SMSA | metropolitan residence=1, otherwise = 0 |
| MS | married = 1; otherwise = 0 |
| FEM | female = 1; otherwise = 0 |
| UNION | Union contract wage = 1; otherwise = 0 |
| ED | Years of education |
| BLK | black = 1; otherwise = 0 |
| LWAGE | log of wage |

**Table 3.9.** Cornwell and Rupert data: variables and their meaning

We are interested in predicting the wage (on the log scale) of each individual, given the covariates. Let $y_{it}$ be a continuous variable indicating the log of wage of person $i$ with the vector of covariates $x_{it}$ in year $t$, $t = 1,...,7$. We use the following DeepGLMM model

$$y_{it}|x_{it} \sim \mathcal{N}(\mu_{it}, \sigma^2), \quad \mu_{it} = \mathfrak{N}(x_{it}, w, \beta + \alpha_i),$$

where $\mathfrak{N}(x_{it},w,\beta+\alpha_i)$, $\beta$ and $\alpha_i$ have the same interpretation as in Section 3.7.1, and $\sigma^2$ is the noise variance.

Since we are interested in within-subject prediction, for each individual, we use the first 5 observations as training data, and the last 2 observations for test data. We use a neural network with 2 hidden layers with 5 nodes each; this structure was selected after some experiments using the lower bound as the model selection criterion. We compare the performance of DeepGLMM to a linear regression model with a random effect, using PPS and MSE as evaluation metrics. Table 3.10 summarizes the results, which show that using the neural network basis functions to model covariate effects in a flexible way can significantly improve both PPS and MSE.

| Model | PPS | MSE |
|---|---|---|
| GLMM | 0.05 | 0.18 |
| DeepGLMM | -0.87 | 0.05 |

**Table 3.10.** Cornwell and Rupert data: Performance of DeepGLMM v.s. conventional GLMM in term of the partial predictive score (PPS) and the mean square error (MSE). Both are evaluated on the test dataset.

## 3.8 Chapter summary

This paper is concerned with flexible versions of generalized linear and generalized linear mixed models where DFNN methodology is used to automatically choose efficient transformations of the raw covariates. The challenges of Bayesian computation are addressed using variational approximation methods with a parsimonious factor covariance structure. We have demonstrated that a natural gradient approach to the variational optimization with this family of approximations is feasible even in high dimensions. Our Bayesian treatment offers a principled and convenient way for selecting the tuning parameters, quantifying uncertainty and doing model selection. Using simulated and real datasets and several different models we show that the improvement that these methods can bring in terms of speed of convergence and computation time are substantial, and that the use of neural network basis functions with random effects is a class of models that deserve more consideration in the literature.

Similar to the GLM and GLMM models, their DeepGLM and DeepGLMM variants are designed to learn the underlying effects of cross-sectional and panel data, respectively. However, the same way of design cannot be applied to the financial applications where time series data are required, e.g. financial volatility modeling, as the DFNN structure are not be able to capture the temporal dependencies exhibited in the time series data. In the next chapters, we will discuss other approaches to flexibly combine Recurrent Neural Network (RNN) and two widely used classes of financial econometrics models in financial volatility modeling literature.

# Chapter 4

# RECH: REcurrent Conditional Heteroskedastic Models

We propose a new class of financial volatility models, which we call the REcurrent Conditional Heteroskedastic (RECH) models, to improve both the in-sample analysis and out-of-sample forecast performance of the traditional conditional heteroskedastic models. In particular, we incorporate auxiliary deterministic processes, governed by recurrent neural networks, into the conditional variance of the traditional conditional heteroskedastic models, e.g. the GARCH-type models, to flexibly capture the dynamics of the underlying volatility. The RECH models can detect interesting effects in financial volatility overlooked by the existing conditional heteroskedastic models such as the GARCH (Bollerslev, 1986), GJR (Glosten et al., 1993) and EGARCH (Nelson, 1991). The new models often have good out-of-sample forecasts while still explain well the stylized facts of financial volatility by retaining the well-established structures of the econometric GARCH-type models. These properties are illustrated through simulation studies and applications to four real stock index datasets. An user-friendly software package together with the examples reported in the paper are available at https://github.com/vbayeslab.

## 4.1 Motivation and Contribution

Financial time series, e.g. currency exchange rates or stock returns, exhibit stylized facts such as volatility clustering and leverage effects. The volatility clustering phenomenon of financial time series refers to the observation that "large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes" (Mandelbrot, 1967). This behavior implies that the volatilities, i.e. the conditional standard deviations, of financial returns are observed to be highly autocorrelated in certain time periods and exhibit periods of both low and high volatility. The leverage effects exhibited in financial time series, on the other hand, relate to the observation that the negative and positive past returns have asymmetric effects on the volatility (Black, 1976). More specifically, the current volatility tends to be larger following a previous negative shock, i.e. a return below its expected value, than a positive one of the same absolute value. The volatility clustering and leverage effects stylized facts arguably imply that the volatility of financial assets changes over time, i.e. is heteroskedastic.

The time-varying volatility is the key assumption in the volatility modeling literature. A large number of volatility models have been developed since Engle (1982) proposed the Autoregressive Conditional Heteroskedastic (ARCH) model, which allows the conditional variance, i.e, the squared volatility, to change over time as a *deterministic* function of the historical shocks while leaving the unconditional variance unchanged. The most important extension of the ARCH model is the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model of Bollerslev (1986); it models the current conditional variance as a function of the past conditional variance and shocks. The GARCH model together with the ARCH model and their variants define a class of models, often referred to as the conditional heteroskedastic or GARCH-type models, which use deterministic functions of historical information, e.g. the past returns and past conditional variances, to model the current conditional variance and are able to capture the stylized facts of financial time series. Another notable line of research in the volatility modeling literature focuses on the stochastic volatility (SV) model (Taylor, 1986) and its variants, which formulate the conditional variance using latent *stochastic* processes that do not directly involve the past returns. This paper is, however, interested in the GARCH-type models which are probably more popular in the volatility modeling literature because it is much easier to estimate a GARCH-type

model than a SV-type alternative. See Koopman et al. (2016) for a comprehensive comparison between the GARCH-type and SV-type models.

Hansen and Lunde (2005) compare empirically the out-of-sample performance of the GARCH model and its many more sophisticated variants. Their results on the DM/USD exchange rate show no evidence that the GARCH(1,1) is outperformed by more sophisticated models, whereas the GARCH(1,1) is clearly inferior to the models equipped with leverage effects in the analysis of the IBM stock returns. The conclusions of Hansen and Lunde (2005) suggest two important implications. First, using complicated models can be useful to explain the underlying volatility movement but it does not necessarily mean that the ability of out-of-sample forecast is improved. In their exchange rate example, the conventional GARCH(1,1) model is simple, yet still has comparable forecast performance to its more complicated counterparts. Second, there are arguably significant differences in the underlying dynamics between the volatility process of stock returns and that of the exchange rate data. This is consistent with the analysis in Nguyen et al. (2019) that reveals a significant difference in the dynamics of the underlying volatility processes between the stock index SP500, ASX200 datasets and the AUD/USD exchange rate data. These differences in the volatility dynamics of different financial products imply that it is important to select the right GARCH-type model for a particular financial time series. The two implications above suggest that it is important to design more flexible volatility models which have both good in-sample and out-of-sample performance.

Recurrent neural networks (RNNs) in the Deep Learning literature are successfully used in a large number of industrial-level applications; e.g. language translation, image captioning, speech synthesis. RNNs are well-known for their ability to efficiently capture the long-range memory and non-linear serial dependence existing within various types of sequential data, and are considered as the state-of-the-art models for many sequence learning problems (Lipton et al., 2015). See Goodfellow et al. (2016) for a comprehensive discussion of various types of neural network models (NNs) and their broad range of applications. The recent success of RNNs has motivated econometricians to incorporate the RNNs and other deep learning models into their econometric models. There is a large amount of research along this line, with a focus on modelling the mean rather than the variance of financial asset returns; see, e.g., Zhang et al. (1998); Zhang (2003a). Leveraging the power of deep learning in volatility modelling is still somewhat overlooked in the econometrics

literature. Donaldson and Kamstra (1997) are one of the first to propose a NN-GARCH model that adds a feedforward NN component into the conditional variance of the GJR model (Glosten et al., 1993). Their in-sample and out-of-sample results on four stock markets suggest that the NN-GARCH model is preferred to several benchmark GARCH-type models. Roh (2007) proposes NN-based volatility models that first estimate the conditional variance by an econometric volatility model, then use these estimates as inputs to a feedforward neural network, which then non-linearly transforms these inputs to output the final estimate of the conditional variance. Kim and Won (2018) extend this idea by using the outputs from several GARCH-type models rather than a single one as the inputs to a recurrent neural network; see also Luo et al. (2018). Liu (2019) uses the Long Short-term Memmory, a sophisticated RNN tehnique, for volatility modelling and reports its improved prediction over GARCH in two datasets. In general, these hybrid models that combine neural networks and financial econometric models are empirically superior to several econometric models and plain neural network models, in term of predictive performance. However, these models are often engineering-oriented and ignore the interpretational aspects of the volatility estimates and the important stylized facts of financial time series. Some of these existing models use feedforward NN models rather than RNNs and thus might ignore the time effects in time series data. Their design is also rather inflexible in the sense that they combine a particular econometric model with a particular NN model. It is important to design a more flexible framework that is easy to adapt to advances in both the deep learning and volatility modelling literature.

The GARCH-type models are generally simple yet highly interpretable in the sense that they are well designed to explain the distinct behaviors of financial time series. Any new volatility models should not overlook this interpretability of the traditional econometric models. This paper proposes a new class of models, called the REcurrent Conditional Heteroskedastic models (RECH), that not only improve the forecast performance of the GARCH-type models, by leveraging the capability of learning non-linearity and long-range dependence of RNNs, but also place significant emphasis on the interpretation of the estimated volatility, by inheriting the well-established structures of the GARCH-type models. We now explain why the RECH models fit well within the volatility modeling literature. First, similarly to the GARCH-type models, the conditional variances in the RECH models are a deterministic function of the past values; hence it is easy to estimate the RECH models as the likelihood functions can be evaluated analytically. Second, the RECH models still explain well the stylized

facts of the underlying volatility dynamics. Third, by inheriting the predictive power from deep learning techniques, the RECH models often forecast better. Fourth, the highly flexible design of RECH models makes it easy to adopt advances in both the deep learning and volatility modeling literatures, allowing it to be used in a wide range of applications in financial time series analysis. A Matlab software package implementing Bayesian estimation and inference for the RECH models together with the examples reported in this paper are available at https://github.com/vbayeslab.

## 4.2 Conditional heteroscedastic models

Let $y = \{y_t, t = 1,...,T\}$ be a time series of demeaned returns and $\mathcal{F}_t$ be the $\sigma$-field of the information up to time $t$. Conditional heteroskedastic models represent the conditional variance $\sigma_t^2 := \text{Var}(y_t|\mathcal{F}_{t-1})$ of the observation $y_t$ as a deterministic function of the observations and the conditional variances in the previous time steps. Mathematically, these models are expressed as:

$$y_t = \sigma_t \epsilon_t, \epsilon_t \sim i.i.d \text{ with } t = 1,2,...,T, \tag{4.1a}$$

$$\sigma_t^2 = \omega + f(\sigma_{t-1}^2,...,\sigma_{t-p}^2, y_{t-1},...,y_{t-q}, \theta); \tag{4.1b}$$

$f(\cdot)$ is a positive deterministic function parameterized by the vector of unknown parameters $\theta$; $p,q \geq 0$ are the number of lags of $\sigma_t^2$ and $y_t$ respectively; and $\omega$ is a non-negative constant ensuring the positivity of the conditional variance $\sigma_t^2$. The shocks $\epsilon_t$ are i.i.d. with zero mean and unit variance.

The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model (Bollerslev, 1986) formulates the conditional variance $\sigma_t^2$ as a linear combination of the previous returns and conditional variances in a ARMA($p$,$q$) form as:

$$y_t = \sigma_t \epsilon_t, t = 1,2,...,T, \tag{4.2a}$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, t = 2,...,T; \tag{4.2b}$$

$\omega > 0, \alpha_i, \beta_j \geq 0$, $i = 1,...,p$, $j = 1,...,q$ and $\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j < 1$ to ensure the stationarity of the GARCH process.

The structure of the conditional variance in (4.2b) is symmetric in the sense that the conditional variance $\sigma_t^2$ does not depend on the sign of the $y_t$, implying that the conventional GARCH($p$,$q$) model cannot capture the important leverage effect, i.e., $\sigma_t^2$

depends asymmetrically on the previous returns, in financial time series. Glosten et al. (1993) propose a variant of the GARCH model, often called the GJR model, of the form

$$y_t = \sigma_t \epsilon_t,\ t = 1,2,...,T, \tag{4.3a}$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i y_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 + \sum_{i=1}^{p} \gamma_i \mathbb{I}[y_{t-i} < 0] y_{t-i}^2; \tag{4.3b}$$

$\omega > 0, \alpha_i, \beta_j \geq 0, \alpha_i + \gamma_i \geq 0$, $i = 1,...,p$, $j = 1,...,q$ and $\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j + \sum_{i=1}^{p} \gamma_i < 1$ to ensure the stationarity of the $y_t$ process and the positivity of the conditional variance. The indicator function $\mathbb{I}[y_t < 0]$ in (4.3b) equals 1 if $y_t < 0$ and 0 otherwise. In the GJR model, negative returns have more influence than positive returns.

Another popular GARCH-type model is the Exponential Garch (EGARCH) model of Nelson (1991)

$$y_t = \sigma_t \epsilon_t,\ t = 1,2,...,T, \tag{4.4a}$$

$$\log \sigma_t^2 = \omega + \sum_{i=1}^{p} \beta_i \log \sigma_{t-i}^2 + \sum_{j=1}^{q} \alpha_j \left[ \frac{|y_{t-j}|}{\sigma_{t-j}} - \mathbb{E}\left\{ \frac{|y_{t-j}|}{\sigma_{t-j}} \right\} \right] + \sum_{j=1}^{q} \gamma_j \frac{y_{t-j}}{\sigma_{t-j}}, \tag{4.4b}$$

where the roots of the polynomial $(1 - \beta_1 L - ... - \beta_p L^p)$ must lie outside the unit circle to ensure the stationarity of the $y_t$ process. By working on the log-scale, the EGARCH model removes the positivity constraints on the model parameters and state the leverage terms in Eq. (4.4b) to capture the asymmetry in volatility clustering. See Poon and Granger (2003) and Bollerslev (2008) for a comprehensive discussion of the family of GARCH models and their properties. We will use GARCH, GJR and EGARCH as the benchmark econometric models to compare against the RECH models, because they are widely used in the volatility modelling literature. See

## 4.3 Recurrent conditional heteroskedastic models

### 4.3.1 The general formulation

The key motivation of the RECH models is to allow the constant term $\omega$ in the general formulation of the GARCH-type models in (4.1a)-(4.1b) to be driven by an auxiliary deterministic process governed by an RNN, in order to capture complex dynamics such as non-linearity and long-term dependence that might not be captured efficiently

by the GARCH-type component $f(\sigma_{t-1}^2,....,y_{t-q})$. The general RECH model is written as:

$$y_t = \sigma_t \epsilon_t, \epsilon_t \sim i.i.d, t = 1,2,...,T \tag{4.5a}$$

$$\sigma_t^2 = g(\omega_t) + f(\sigma_{t-1}^2,...,\sigma_{t-p}^2,y_{t-1},...,y_{t-q}), t = 2,...,T, \sigma_1^2 = \sigma_0^2 \tag{4.5b}$$

$$\omega_t = \beta^\top h_t + \beta_0, t = 2,...,T, \tag{4.5c}$$

$$h_t = \text{RNN}(x_t,h_{t-1}), t = 2,...,T, \text{with} h_1 \equiv 0; \tag{4.5d}$$

$g(\cdot)$ is a non-negative activation function; $p$ and $q$ are lag orders of $\sigma_t^2$ and $y_t$ respectively; $\beta_0$ is a scalar; $\beta = (\beta_1,...,\beta_L)$ is the weight vector with $L$ the number of hidden states. The reason an activation function is applied to $\omega_t$ is to ensure the conditional variance $\sigma_t^2$ is positive. We refer to $\omega_t$ in (4.5b) as the *recurrent* component, as it is driven by a RNN, and $f(\cdot)$ as the *garch* component as this is formed based on the GARCH-type structures without the constant term. Hence, we shall refer to the parameters of the recurrent component, e.g. $\beta_0,\beta,V,W,b$ if we use the SRN unit, as the recurrent parameters and refer to the parameters of the garch component $f(\cdot)$ as the *garch* parameters. The recurrent state $h_t = RNN(x_t,h_{t-1})$ takes as its inputs the previous state $h_{t-1}$ and a vector of additional information $x_t$ whose choices are discussed shortly.

The conditional variance in (4.5b) is a sum of the recurrent and the garch components. This flexible design allows the RECH models to enjoy many advances from both worlds of deep learning and volatility modelling. Similarly to deep learning models, the RECH models can use highly sophisticated neural network structures to capture the complicated dynamics, e.g. long-range dependence and non-linearity, of the volatility dynamics and hence improve the predictive performance of the traditional GARCH-type models in the applications where the underlying volatility dynamics exhibits long memory and nonlinearity. Similarly to GARCH-type models, the RECH models use simple yet highly interpretable structures to simulate important stylized facts in financial time series such as volatility clustering and leverage effects. RECH models are well suited to modeling volatility because they inherit many properties from the GARCH-type models and distinguish themselves from the existing NN-based volatility models that often overlook the interpretability of the mainstream econometric models. As the recurrent and garch components are separated, increasing the complexity of the recurrent component $\omega_t$ will not decrease the interpretability of the garch component, and hence of the RECH models. The general formulation in (4.5a)-(4.5d) implies that most (if not all) variants of the GARCH models are nested

in the RECH framework, because the RECH models reduce to the corresponding GARCH-type models if $\beta = 0$.

In previous work that combined an NN-based component with a GARCH-type model, Donaldson and Kamstra (1997) added a FNN-based component to a GJR model and reported some improvement of their FNN-GJR model compared to several benchmark GARCH-type models. However, as discussed above, it might be inefficient to use a FNN to analyze time series data as FNNs are typically designed for cross-sectional data. Also, their estimation method uses randomized weights for the FNN component rather than optimizing them; this technique is not recommended in the modern deep learning literature (Goodfellow et al., 2016). Nguyen et al. (2019) incorporate LSTM into the stochastic volatility models and name their model LSTM-SV. LSTM-SV belongs to the class of parameter-driven models while RECH is an observation-driven model; see Koopman et al. (2016) for a comprehensive comparison between these two classes of models. More specifically, the volatility dynamics in RECH is a *deterministic* process rather than a stochastic latent process as in LSTM-SV, making it much easier to estimate RECH models than the LSTM-SV model as the likelihood of the RECH models can be calculated analytically. The RNN component of the LSTM-SV model takes only its past values as inputs while the recurrent component $\omega_t$ of the RECH models allows any information including past observations as inputs. We show below that this flexibility enables RECH models with SRN to capture complicated dynamics and the leverage effect in financial time series without needing to use complicated RNN structures such as LSTM. Finally, like GARCH and SV, RECH and LSTM-SV complement each other and offer different perspectives towards the volatility modelling problem. We note that the RECH models are fully *parametric* models as the mathematical representation of the RNN component must be specified before the estimation and forecast steps. See Engle and Rangel (2008); Koo and Linton (2015) For the discussion on other line of non-parametric models allowing GARCH parameters to be time-varying. For the discussion on the difference between parametric and non-parametric volatility models, see Linton (2009); Andersen et al. (2010).

### 4.3.2 Specifications for the RECH models

The RECH framework is highly flexible because it can easily incorporate the advances from both the deep learning and volatility modeling literatures to design the recurrent and garch components, respectively. For example, by using the SRN structure for the

recurrent component $\omega_t$ and the conditional variance structure of the GARCH(1,1) model for the garch component, we obtain the SRN-GARCH specification of the RECH model as:

$$y_t = \sigma_t \epsilon_t, \epsilon_t \overset{iid}{\sim} \mathcal{N}(0,1), t = 1,2,...,T \tag{4.6a}$$

$$\sigma_t^2 = \omega_t + \alpha y_{t-1}^2 + \beta \sigma_{t-1}^2, t = 2,...,T, \sigma_1^2 = \sigma_0^2 \tag{4.6b}$$

$$\omega_t = \beta_0 + \beta_1 h_t, t = 2,...,T, \tag{4.6c}$$

$$h_t = \text{SRN}(x_t, h_{t-1}), t = 2,...,T, \text{ with } h_1 \equiv 0; \tag{4.6d}$$

$x_t = (\omega_{t-1}, y_{t-1}, \sigma_{t-1}^2)^\top$ is the vector of input of the RNN at time $t$. For simplicity, we consider the standard normal distribution $\mathcal{N}(0,1)$ for the errors $\epsilon_t$. To ensure the positivity of the conditional variance, we specify the activation function $g(\cdot)$ in Eq. (4.5b) as a linear function of $\omega_t$, i.e. $g(\omega_t) = \omega_t$ and set $\beta_0, \beta_1 \geq 0$. Alternatively, one can use a positive activation function for $g(\cdot)$, such as the ReLu or sigmoid, and relax the positivity constraints for $\beta_0$ and $\beta_1$. We follow the GARCH literature and put the stationarity and positivity constraints on the garch parameters $\alpha$ and $\beta$, i.e., $\alpha, \beta \geq 0$ and $\alpha + \beta < 1$. These constraints do not necessarily imply that the SRN-GARCH model is stationary, but might improve numerical stability in estimation. Figure 4.1 is a graphical representation of the SRN-GARCH model.

The recurrent function SRN is:

$$\text{SRN}(x_t, h_{t-1}) = \phi(v^\top x_t + w h_{t-1} + b),$$

with $v = (v_0, v_1, v_2)^\top$ the vector of weights and $\phi(\cdot)$ an activation function. We use the ReLU activation for $\phi(\cdot)$ as it is easier to train and performs better than other alternatives in the deep learning literature (Nair and Hinton, 2010; Le et al., 2015). The weights $\beta_1$ and $w$ are scalars, as by default we use only one recurrent state, i.e. $h_t$ is a scalar, for the SRN-GARCH specification. However, it is possible to extend the specification in (4.6a)-(4.6d) to the case where $h_t$ is a vector of hidden states.

For the RECH model, there is no restriction on the choice of the input vector $x_t$. A convenient choice for $x_t$ is the vector of the past recurrent component $\omega_{t-1}$, past return $y_{t-1}$ and past conditional variance $\sigma_{t-1}^2$. While a non-zero $\beta$ in (4.6b) quantifies the linear dependence of the current conditional variance $\sigma_t^2$ on its past value $\sigma_{t-1}^2$, a non-zero weight $v_2$ quantifies the non-linear dependence of $\sigma_t^2$ on $\sigma_{t-1}^2$. The choice for $x_t$ is somewhat simulated the structure of GARCH, e.g. using $\sigma_{t-1}^2$ component, and EGARCH, e.g. using the $y_{t-1}$ component, but in a non-linear manner, e.g. output

is a non-linear function for $x_t$. We note that the SRN-GARCH specification uses the conditional variance structure of the GARCH model, but is still able to capture the leverage effects of the volatility dynamics as the input vector $x_t$ includes the asymmetric leverage term $y_{t-1}$ itself, not $y_{t-1}^2$. Section 4.4 shows that this choice of the input vector $x_t$ makes the volatility estimated by the RECH models less sensitive to the choice of the structure for the garch component. In terms of model specification, the SRN-GARCH model is simpler than the corresponding GARCH($p$,$q$) model in the sense that it does not require choosing the lag orders $p$ and $q$ as the recurrent component $\omega_t$ is designed to capture the potential long range dependence and non-linear effects that cannot be detected by the garch component $f(\cdot)$. That SRN-GARCH($p$,$q$) The RECH model with the SRN-GARCH specification simplifies to the GARCH(1,1) model if $\beta_0 > 0$ and $\beta_1 = 0$.

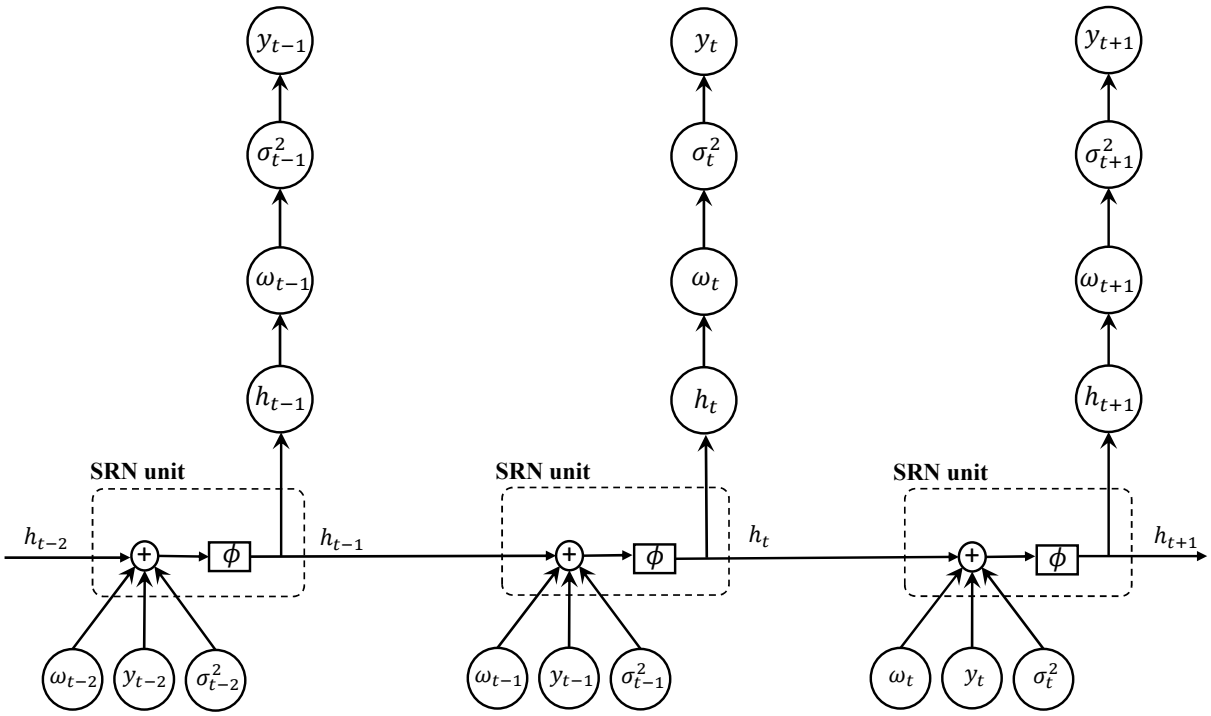


**Figure 4.1.** Graphical representation of the SRN-GARCH specification of the RECH models.

Many other specifications of the RECH model can be constructed. For example, the SRN-GJR specification is obtained by using the SRN structure for the recurrent component and the conditional variance structure of the GJR(1,1) for the garch component. Table 4.1 presents several RECH model specifications. It is also possible to use other RNN structures such as LSTM (Hochreiter and Schmidhuber, 1997) or SRU for the recurrent component. It is worth noting that the GJR and EGARCH models

accommodate the leverage effects as linear terms in the conditional variance equation and hence can only capture the linear dependence of the leverage effects. The RECH models, on the other hand, are able to capture other leverage dependence rather than the linearity, e.g. non-linearity or temporal dependence, of the leverage effects by allowing the leverage term $y_{t-1}$ to be an input of the RNN.

| Models | Conditional variance | Constraints |
|---|---|---|
| SRN-GJR | $\sigma_t^2 = \omega_t + \alpha y_{t-1}^2 + \beta\sigma_{t-1}^2 + \gamma \mathrm{I}_{[y_{t-1}<0]} y_{t-1}^2$<br>$\omega_t = \beta_0 + \beta_1 h_t$<br>$h_t = \mathrm{SRN}(x_t, h_{t-1})$, with $x_t = (\omega_{t-1}, y_{t-1}, \sigma_{t-1}^2)$ | $\alpha,\beta \geq 0$<br>$\alpha+\gamma \geq 0$<br>$\alpha+\beta+\gamma < 1$<br>$\beta_0,\beta_1 \geq 0$ |
| SRN-EGARCH | $\sigma_t^2 = \omega_t + \exp\left\{\omega + \beta \log\sigma_{t-1}^2 + \alpha\left[\frac{\lvert y_{t-1}\rvert}{\sigma_{t-1}} - \sqrt{\frac{2}{\pi}}\right] + \gamma\frac{y_{t-1}}{\sigma_{t-1}}\right\}$<br>$\omega_t = \beta_0 + \beta_1 h_t$<br>$h_t = \mathrm{SRN}(x_t, h_{t-1})$ with $x_t = (\omega_{t-1}, \mathrm{e}^{y_{t-1}}, \sigma_{t-1}^2)$ | $0 \leq \beta < 1$ |

**Table 4.1.** Several specifications of the RECH model.

Given the general formulation of the RECH models in (4.5a)-(4.5d), the $\sigma_t^2$ process, and thus the $y_t$ process of the RECH models, is not guaranteed to be stationary unless $\beta_1 = 0$ and the garch parameters satisfy the stationary constraints of the corresponding garch components $f(\cdot)$. For example, the SRN-GARCH specification in (4.6a)-(4.6d) is stationary if $\beta_0,\alpha,\beta > 0, \beta_1 = 0, \alpha+\beta < 1$. Although non-stationarity for volatility may be mathematically less appealing, it is often argued to be more realistic in practice, e.g. van Bellegem (2012).

**Theorem 1.** *Consider the SRN-GARCH specification in* (4.6a)-(4.6d)*. Assume that*

- *the garch component is stationary, i.e.* $\alpha,\beta > 0$, $\alpha+\beta < 1$;
- *the recurrent component is bounded, i.e.* $\omega_t \leq M$ ***almost surely*** *for some* $M < \infty$.

*Then the unconditional variance of* $y_t$ *is bounded,*

$$\mathrm{Var}(y_t) \leq \frac{M}{1-\alpha-\beta} + \sigma_0^2, \textit{for all } t.$$

The condition that $\alpha > 0, \beta > 0$, $\alpha+\beta < 1$ is standard in the GARCH literature and easy to impose. The second condition, i.e. the finite recurrent component condition, is

imposed by using the bounded ReLU activation function (to bound $h_t$), and assuming a bounded support for $\beta_0$ and $\beta_1$ (we used uniform U(0,0.5) for these two parameters).

*Proof.* Recall the $\sigma$-fields $\mathcal{F}_t = \sigma(y_s, s \leq t)$ and let us define $\mathcal{F}_0 = \{\emptyset, \Omega\}$. As the recurrent component is bounded,

$$\mathbb{E}(y_t^2|\mathcal{F}_{t-1}) = \sigma_t^2 \leq M + \alpha\sigma_{t-1}^2 + \beta y_{t-1}^2, \quad t > 1. \tag{4.7}$$

We have that

$$\begin{aligned} \mathbb{E}(y_t^2|\mathcal{F}_{t-2}) &= \mathbb{E}\big(\mathbb{E}(y_t^2|\mathcal{F}_{t-1})|\mathcal{F}_{t-2}\big) \\ &\leq M + \alpha\sigma_{t-1}^2 + \beta\mathbb{E}(y_{t-1}^2|\mathcal{F}_{t-2}) \\ &= M + \alpha\sigma_{t-1}^2 + \beta\sigma_{t-1}^2, \end{aligned}$$

hence, by (4.7),

$$\mathbb{E}(y_t^2|\mathcal{F}_{t-2}) \leq \begin{cases} M + (\alpha+\beta)\sigma_1^2 < M + \sigma_0^2, & t = 2 \\ M + (\alpha+\beta)\big(M + \alpha\sigma_{t-2}^2 + \beta y_{t-2}^2\big), & t > 2. \end{cases}$$

Similarly,

$$\begin{aligned} \mathbb{E}(y_t^2|\mathcal{F}_{t-3}) &= \mathbb{E}\big(\mathbb{E}(y_t^2|\mathcal{F}_{t-2})|\mathcal{F}_{t-3}\big) \\ &\leq M + (\alpha+\beta)\big(M + \alpha\sigma_{t-2}^2 + \beta\mathbb{E}(y_{t-2}^2|\mathcal{F}_{t-3})\big) \\ &= M\big(1 + (\alpha+\beta)\big) + (\alpha+\beta)^2\sigma_{t-2}^2, \quad t \geq 3. \end{aligned}$$

For $t = 3$,

$$\mathbb{E}(y_t^2|\mathcal{F}_{t-3}) = M\big(1+(\alpha+\beta)\big) + (\alpha+\beta)^2\sigma_1^2 < M\big(1+(\alpha+\beta)\big) + \sigma_0^2,$$

and for $t > 3$, by (4.7),

$$\begin{aligned} \mathbb{E}(y_t^2|\mathcal{F}_{t-3}) &\leq M\big(1 + (\alpha+\beta)\big) + (\alpha+\beta)^2\big(M + \alpha\sigma_{t-3}^2 + \beta y_{t-3}^2\big) \\ &\leq M\big(1 + (\alpha+\beta) + (\alpha+\beta)^2\big) + (\alpha+\beta)^2\big(\alpha\sigma_{t-3}^2 + \beta y_{t-3}^2\big). \end{aligned}$$

Hence

$$\mathbb{E}(y_t^2|\mathcal{F}_{t-3}) \leq \begin{cases} M\big(1 + (\alpha+\beta)\big) + \sigma_0^2, & t = 3 \\ M\big(1 + (\alpha+\beta) + (\alpha+\beta)^2\big) + (\alpha+\beta)^2\big(\alpha\sigma_{t-3}^2 + \beta y_{t-3}^2\big), & t > 3. \end{cases}$$

By deduction we have that, for $k = 2, \ldots, t-1$,

$$\mathbb{E}(y_t^2|\mathcal{F}_{t-k}) \leq \begin{cases} M\big(1 + \sum_{i=1}^{k-2}(\alpha+\beta)^i\big) + \sigma_0^2, & t = k \\ M\big(1 + \sum_{i=1}^{k-1}(\alpha+\beta)^i\big) + (\alpha+\beta)^{k-1}\big(\alpha\sigma_{t-k}^2 + \beta y_{t-k}^2\big), & t > k. \end{cases} \tag{4.8}$$

Therefore,

$$\begin{aligned}\mathbb{E}(y_t^2) = \mathbb{E}(y_t^2|\mathcal{F}_0) &\leq M\big(1+\sum_{i=1}^{k-2}(\alpha+\beta)^i\big)+\sigma_0^2 \\ &< M\big(1+\sum_{i=1}^{\infty}(\alpha+\beta)^i\big)+\sigma_0^2 = \frac{M}{1-\alpha-\beta}+\sigma_0^2.\end{aligned}$$

□

## 4.4 Simulation study and applications

This section evaluates the in-sample and out-of-sample performance of the RECH models on simulation and stock return datasets. We use the SMC with likelihood annealing sampler to perform in-sample Bayesian inference and the SMC with data annealing to obtain one-step-ahead forecasts. See Li et al. (2021) for a comprehensive study on the efficiency of the SMC methods for GARCH-type models. Table 4.2 lists our implementation settings of the SMC samplers.

| Variable | Description | Value |
|---|---|---|
| $K$ | Number of annealing levels | 10000 |
| $M$ | Number of particles | 10000 |
| $c$ | Constant of the ESS threshold | 0.8 |
| $N_{\text{lik}}$ | Number of Markov moves in the SMC with likelihood annealing | 30 |
| $N_{\text{data}}$ | Number of Markov moves in the SMC with data annealing | 30 |

**Table 4.2.** Implementation settings of the SMC samplers.

We now discuss the selection of the prior distributions for the RECH models. We use a normal prior with a zero mean and variance 0.1 for all the recurrence parameters $(v_0,v_1,v_2,w,b)$, except $\beta_0$ and $\beta_1$; as empirical results from the deep learning literature show that the values of the weights of neural networks are often small. As a linear activation function for $g(\cdot)$ is used, we put the uniform prior $U(0,0.5)$ on $\beta_0$ and $\beta_1$ to impose the positivity of $\omega_t$. For the garch parameters, we choose the same priors as we use for the corresponding GARCH-type models, as summarized in Table 4.3.

Next, we discuss the score metrics used to evaluate the out-of-sample performance. Denote by $D_{\text{test}}$ a test dataset, $T_{\text{test}}$ the number of observations in $D_{\text{test}}$ and $\widehat{\theta}$ the

| GARCH(1,1) | | GJR(1,1) | | EGARCH(1,1) | |
|---|---|---|---|---|---|
| Parameter | Prior | Parameter | Prior | Parameter | Prior |
| $\omega$ | $U(0,10)$ | $\omega$ | $U(0,10)$ | $\omega$ | $\mathcal{N}(0,1)$ |
| $\alpha$ | $U(0,1)$ | $\alpha$ | $U(0,1)$ | $\alpha$ | $\mathcal{N}(0,1)$ |
| $\beta$ | $U(0,1)$ | $\beta$ | $U(0,1)$ | $\beta$ | $U(0,1)$ |
| | | $\gamma$ | $\mathcal{N}(0,0.1)$ | $\gamma$ | $\mathcal{N}(0,0.1)$ |

**Table 4.3.** Prior distributions for the parameters in the GARCH(1,1), GJR(1,1) and EGARCH(1,1) models. The notations $U$ and $\mathcal{N}$ denote the Uniform and Gaussian distributions, respectively.

posterior mean estimate of $\theta$; we use four predictive scores to measure out of sample performance: the partial predictive score (PPS), the number of violations (#Vio.), the quantile score (QS) and the hit percentage (%Hit) to measure the out-of-sample performance. The PPS (Gneiting and Raftery, 2007) is evaluated on the test dataset $D_{test}$ as

$$PPS = -\frac{1}{T_{test}} \sum_{D_{test}} \log p(y_t | y_{1:t-1}, \widehat{\theta}).$$

The model with smallest PPS is preferred. The #Vio. is defined as the number of times over the test data $D_{test}$ that the observation $y_t$ is outside its 99% one-step-ahead forecast interval. One of the main applications of volatility modelling is to forecast the Value at Risk (VaR). The $\alpha$-VaR is defined as the $\alpha$-quantile of the one-step-ahead forecast distribution $p(y_t | y_{1:t-1}, \widehat{\theta})$. The performance of a method producing VaR forecasts is often measured by the quantile score (Taylor, 2019) defined as:

$$QS = \frac{1}{T_{test}} \sum_{D_{test}} (\alpha - I_{y_t \leq q_{t,\alpha}})(y_t - q_{t,\alpha}),$$

where $q_{t,\alpha}$ is the $\alpha$-VaR forecast of $y_t$, conditional on $y_{1:t-1}$. The smaller the quantile score, the better the VaR forecast. The %Hit (Taylor, 2019) is defined as the percentage of the $y_t$ in the test data that is below its $\alpha$-VaR forecast. The %Hit is expected to be close to $\alpha$, if the model predicts well. We note that these predictive performance measures complement each other. For example, it is possible to make the number of violations small by increasing the forecast volatility, but the PPS and QS scores then increase. A volatility model minimizing all three predictive scores, and having a hit percentage close to $\alpha$, is arguably the preferred one.

For the simulation data, given the true volatility $\sigma_t$, we follow Hansen and Lunde (2005) and use six additional predictive scores as summarized in Table 4.4.

| Score | Definition |
|---|---|
| $\text{MSE}_1$ | $T_{test}^{-1}\sum_{D_{test}}(\sigma_t-\widehat{\sigma}_t)^2$ |
| $\text{MSE}_2$ | $T_{test}^{-1}\sum_{D_{test}}(\sigma_t^2-\widehat{\sigma}_t^2)^2$ |
| $\text{MAE}_1$ | $T_{test}^{-1}\sum_{D_{test}}\lvert\sigma_t-\widehat{\sigma}_t\rvert$ |
| $\text{MAE}_2$ | $T_{test}^{-1}\sum_{D_{test}}\lvert\sigma_t^2-\widehat{\sigma}_t^2\rvert$ |
| QLIKE | $T_{test}^{-1}\sum_{D_{test}}\left(\log(\widehat{\sigma}_t^2)+\sigma_t^2\widehat{\sigma}_t^{-2}\right)$ |
| $\text{R}^2\text{LOG}$ | $T_{test}^{-1}\sum_{D_{test}}\left[\log(\sigma_t^2\widehat{\sigma}_t^{-2})\right]^2$ |

**Table 4.4.** Definition of the predictive scores to measure the out-of-sample performance on simulation and index data. Here, $\widehat{\sigma}_t$ is an estimate of the volatility $\sigma_t$.

It is also possible to test for statistical differences between the forecast results, e.g. using DM test Diebold (2015). However, these tests often rely on assumptions made directly on the forecast error loss differential and hence significantly sensitive to the true values of volatility, which is not directly observed. We therefore do not follow this approach in our comparison of predictive performance between models.

### 4.4.1 Simulation studies

#### Simulation study I (SIM I)

We generated a time series of 2000 observations from the GARCH(1,1) model

$$y_t=\sigma_t\epsilon_t,\ \epsilon_t\sim\mathcal{N}(0,1),\ t=1,...,T, \tag{4.9a}$$

$$\sigma_t^2=0.05+0.18y_{t-1}^2+0.8\sigma_{t-1}^2,\ t=2,...,T,\ \sigma_1^2=0.1. \tag{4.9b}$$

The first 1000 observations are used for model estimation and the last 1000 observations for out-of-sample analysis. Table 4.5 shows the posterior means and the posterior standard deviations of the GARCH(1,1) and SRN-GARCH model parameters, obtained from the SMC using likelihood annealing. Figure 4.2 plots the

estimated volatility together with the true volatility of the simulated data, i.e. the true values $\sigma_t^2$ generated from equation (4.9b).

| | $\omega$ | $\alpha$ | $\beta$ | $\beta_0$ | $\beta_1$ | $v_1$ | $v_2$ | log ML |
|---|---|---|---|---|---|---|---|---|
| GARCH | **0.048** | **0.178** | **0.791** | | | | | $-1507.2$ |
| | (0.022) | (0.029) | (0.035) | | | | | (0.118) |
| SRN-GARCH | | **0.150** | **0.806** | **0.056** | **0.232** | **0.154** | $\mathbf{-0.248}$ | $-1507.3$ |
| | | (0.028) | (0.029) | (0.022) | (0.152) | (0.314) | (0.378) | (0.152) |

**Table 4.5.** SIM I: Posterior means (in bold) of the GARCH and SRN-GARCH model parameters with the posterior standard deviations in brackets. The last column shows the natural logarithms of the estimated marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the SMC with likelihood annealing sampler.

The estimation results in Table 4.5 and the volatility plots in Figure 4.2 suggest some important implications. First, the garch parameters $\alpha$ and $\beta$ of the SRN-GARCH model in Table 4.5 are close to those of the GARCH model and the volatility estimated from the RECH model in Figure 4.2 are close to the true volatility, implying that the estimated RECH model is close to the GARCH(1,1) model if the GARCH(1,1) is the true model. The almost identical estimates of the marginal likelihood in the last column of Table 4.5 also indicate that the GARCH(1,1) and RECH models provide an equally good fit to the data. Second, the coefficient $\beta_1$ is statistically insignificant, i.e. the posterior mean is less than two standard deviations from zero, and the values of the recurrent component $\omega_t$ are consistently small at all time points as Figure 4.2 shows, implying that there are no volatility effects rather than linearity that are captured by the recurrent component $\omega_t$ and that the recurrent component $\omega_t$ contributes very little to the conditional variance at all time steps. Additionally, the weights $v_1$ and $v_2$ of the inputs of the recurrent component are statistically insignificant, suggesting that there is no evidence of non-linearity, long range dependence and leverage effects within the data generating process GARCH(1,1).

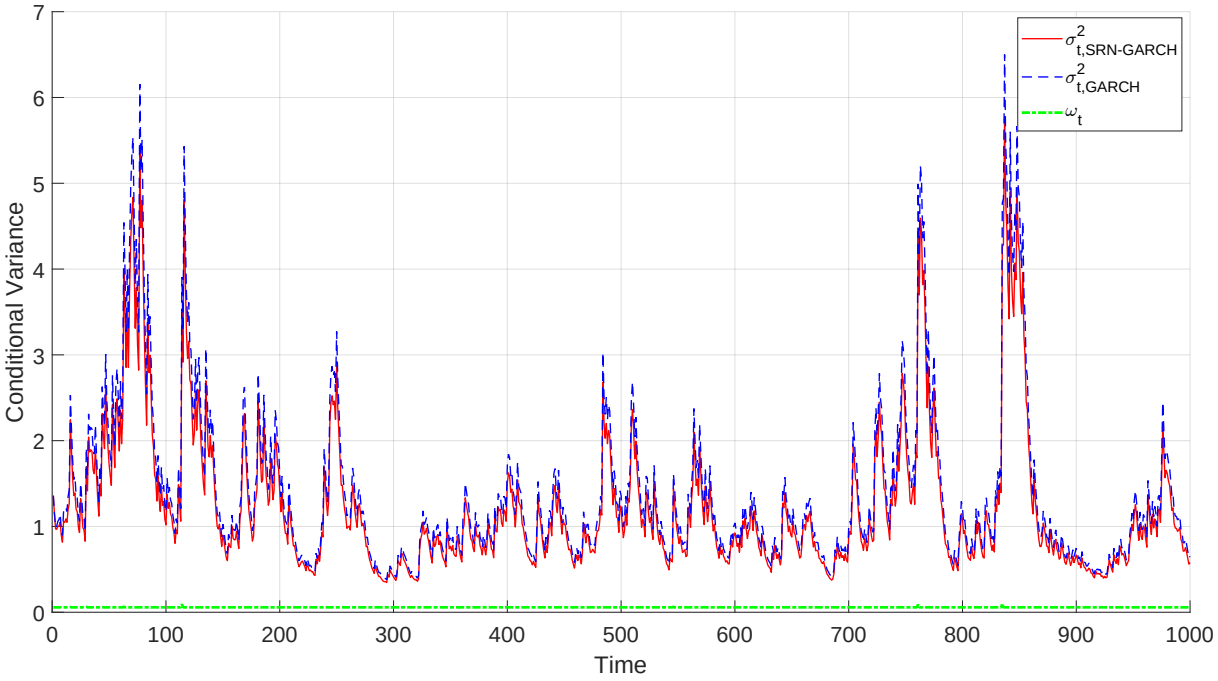


**Figure 4.2.** SIM I: The true conditional variance (dashed line) and the estimated conditional variance (solid line) using the SRN-GARCH model. The bottom line shows the values of the recurrent component $\omega_t$ of the SRN-GARCH specification at all time points. (The figure better viewed in colour).

### Simulation study II (SIM II)

We generated a time series of 2000 observations from the following non-linear GARCH-type model

$$y_t = \sigma_t \epsilon_t,\ \epsilon_t \sim \mathcal{N}(0,1),\ t = 1,2,...,T, \tag{4.10a}$$

$$\sigma_t^2 = 0.05 + 0.10 y_{t-1}^2 + 0.21 \frac{y_{t-1}^2}{1+y_{t-1}^2} + 0.8\sigma_{t-1}^2 + 0.11 \frac{\sigma_{t-1}^2}{1+\sigma_{t-1}^2} + 0.21 I_{[y_{t-1}<0]} y_{t-1}^2 + 0.1 \frac{I_{[y_{t-1}<0]}}{1+e^{-y_{t-1}^2}}. \tag{4.10b}$$

The model in (4.10a)-(4.10b) modifies the GJR(1,1) model by adding non-linear transformations of the past observation, conditional variance and leverage term to the equation of the conditional variance. The volatility evolution in (4.10b) suggests that the simulated volatility exhibits highly non-linear effects. The parameters of the model in (4.10a)-(4.10b) are set so that the simulated data somewhat resembles real financial time series data exhibiting both volatility clustering and leverage effects. The first $T = 1000$ observations are used for model estimation and the last 1000 for out-of-sample analysis. Table 4.6 shows the posterior means and standard deviations

of the parameters from the GARCH(1,1), GJR(1,1), EGARCH(1,1) and three RECH counterparts.

| | $\omega$ | $\alpha$ | $\beta$ | $\gamma$ | $\beta_0$ | $\beta_1$ | $v_1$ | $v_2$ | Log ML |
|---|---|---|---|---|---|---|---|---|---|
| GARCH | **0.854** | **0.188** | **0.807** | | | | | | $-3617.9$ |
| | (0.124) | (0.018) | (0.018) | | | | | | (0.164) |
| SRN-GARCH | | **0.246** | **0.556** | | **0.328** | **0.382** | **-0.525** | **0.396** | $-3614.9^*$ |
| | | (0.048) | (0.158) | | (0.120) | (0.100) | (0.253) | (0.232) | (0.232) |
| GJR | **0.846** | **0.204** | **0.801** | **-0.012** | | | | | $-3620.1$ |
| | (0.130) | (0.035) | (0.021) | (0.024) | | | | | (0.141) |
| SRN-GJR | | **0.141** | **0.557** | **0.179** | **0.354** | **0.373** | **-0.180** | **-0.418** | $-3611.6^*$ |
| | | (0.045) | (0.138) | (0.064) | (0.110) | (0.092) | (0.294) | (0.172) | (0.211) |
| EGARCH | **0.106** | **0.373** | **0.975** | **-0.101** | | | | | $-3616.3$ |
| | (0.027) | (0.041) | (0.005) | (0.026) | | | | | (0.147) |
| SRN-EGARCH | **-0.058** | **0.450** | **0.976** | **-0.114** | **0.227** | **0.270** | **-0.028** | **0.211** | $-3613.1^*$ |
| | (0.173) | (0.145) | (0.017) | (0.037) | (0.140) | (0.139) | (0.361) | (0.360) | (0.202) |

**Table 4.6.** SIM II: Posterior means (in bold) of the GARCH and RECH model parameters with the posterior standard deviations in brackets. The last column shows the natural logarithms of the estimated marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the SMC with likelihood annealing sampler. The asterisks indicate when the Bayes factors strongly support RECH models over their GARCH-type counterparts.

The estimation results from Table 4.6 suggest that: first, the posterior means of $\alpha$ and $\beta$ from the GARCH and GJR models are close to their true values, suggesting that the GARCH and GJR models can capture the linear serial dependence within the volatility dynamics of the data generating process. The constants $\omega$ of both the GARCH and GJR model are significantly inflated compared to the true value, possibly caused by the non-linear effects that cannot be captured by the GARCH and GJR models. The leverage parameter $\gamma$ of the GJR model is close to zero and statistically insignificant, implying that the GJR model cannot capture the leverage effect in the data generating process. The leverage parameter $\gamma$ of the EGARCH model is more than three standard deviations from zero, implying that the EGARCH model is the only benchmark GARCH-type model that can capture the simulated leverage effect.

Second, the coefficients $\beta_1$ of the SRN-GARCH and SRN-GJR models are more than three standard deviations from zero, implying that there is strong evidence

of non-linearity in the volatility dynamics, and that the RNN structure within the recurrent component $\omega_t$ of the SRN-GARCH and SRN-GJR model is able to capture such dependence. The weight $v_1$ with respect to the leverage input $y_{t-1}$ of the RNN in the SRN-GARCH model is more than two standard deviations from zero and the leverage parameters $\gamma$ of the SRN-GJR and SRN-EGARCH are more than three standard deviation from zero, all suggesting that the three RECH specifications can capture the leverage effects exhibited within the simulated volatility. The recurrent component $\omega_t$ of the RECH models is useful in capturing the leverage effects overlooked by the GARCH and GJR models. Interestingly, the coefficient $v_2$ with respect to the input $\sigma_{t-1}^2$ in the SRN-GJR is more than two standard deviations from zero, indicating that the SRN-GJR is able to detect the non-linearity dependence of the past conditional variance on the current conditional variance $\sigma_t^2$, exhibited within the simulated data generating process. The analysis above suggests that observing the recurrent parameters of the RECH models, e.g. $\beta_1$,$v_1$ and $v_2$, helps to detect the possible non-linearity effects within the underlying volatility.

Third, the estimates of marginal likelihood in the last column of Table 4.6 show that the RECH models consistently have higher marginal likelihood than their GARCH-type counterparts and that the SRN-GJR model provides the best fit to the data. The difference between the log marginal likelihood estimates is equivalent to the Bayes factors of the SRN-GARCH, SRN-GJR and SRN-EGARCH models compared to the GARCH, GJR and EGARCH models of roughly $e^3$, $e^8$ and $e^3$, respectively, strongly supporting the RECH models. We note that among the benchmark GARCH-type models, the EGARCH model est fits to the SIM II data.

Figure 4.3 plots the values of the recurrent component and estimated volatility of the SRN-GJR model, together with the true volatility, at all time points. Figure B.1 in the Appendix B plots the volatility estimated by the GJR model and the true volatility. During the low volatility periods, i.e. time $t$ is between 0-100 or 900-1000, the volatility of the SRN-GJR model are very close to the true volatility, which is also the case for the GJR model in Figure B.1. During the periods when the volatility changed dramatically and oscillated highly, i.e. $t$ is between 200-250, 350-400 or 500-700, the volatility produced by the SRN-GJR model still tracks well the true volatility, but this is not the case for the GJR model. The GJR model often produces overly-large and overly-small volatility during these periods of abruptly changed volatility. The SRN-GJR model, on the other hand, appears to be able to capture well these changes.

### 4.4.1 Simulation studies

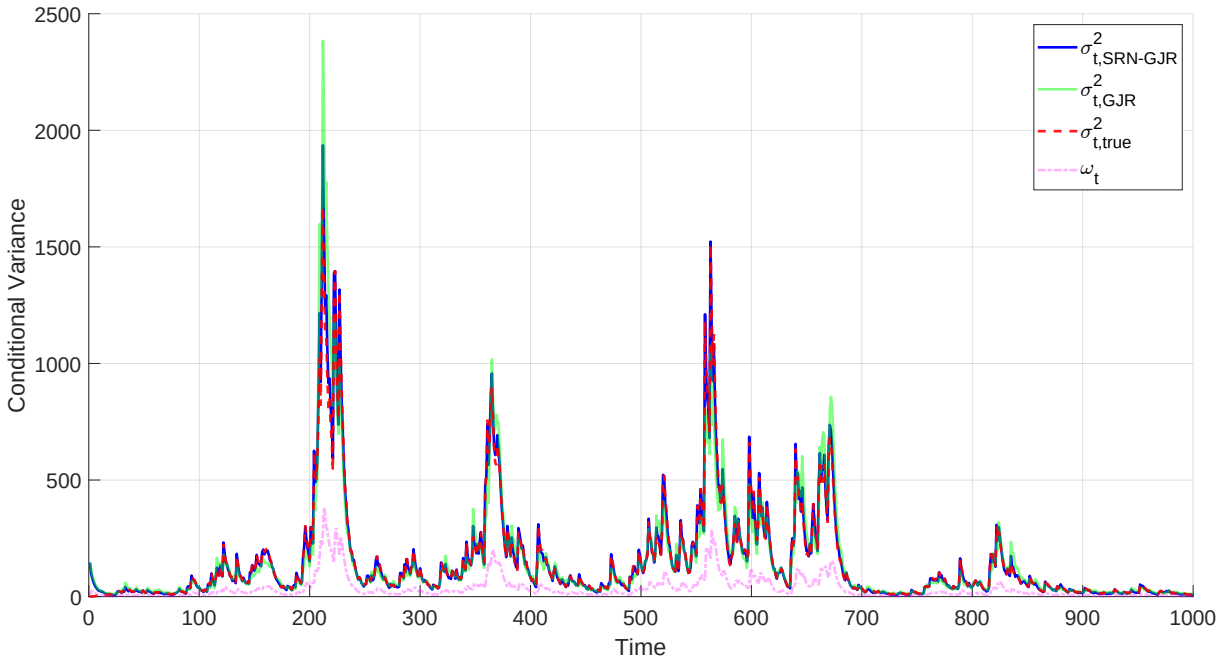


**Figure 4.3.** SIM II: The true conditional variance (dashed line) and the estimated conditional variance (solid lines) by the GJR and SRN-GJR models. The bottom line shows the values of the recurrent component $\omega_t$ of the SRN-GJR specification. The SRN-GJR plot appears to trace the true volatility plot better than the GJR plot. (The figure is better viewed in colour).

The plot of the recurrent component $\omega_t$ at all time points in Figure 4.3 shows that it is highly responsive to the changes in the true volatility.

Table 4.7 reports the forecast performance of the benchmark GARCH-type and the RECH models. For the SIM I data, the forecast performance of the SRN-GARCH model is very close to that of the GARCH model, in all predictive scores, which again strongly supports the earlier in-sample conclusion that the SRN-GARCH model closely approximates the GARCH model if the GARCH is the true model. For the SIM II data, Table 4.7 suggests some important results. First, the RECH models outperform their GARCH-type counterparts in most of the predictive scores, which is consistent with the in-sample analysis showing that the RECH models fit better than their benchmark GARCH-type counterparts. Second, the SRN-GJR model has the best forecast performance for all predictive measures and the SRN-EGARCH model also performs well on the SIM II data, which is consistent with the in-sample analysis showing that these two RECH specifications have the highest marginal likelihood estimates. Third, amongst the benchmark GARCH-type models, the EGARCH model has the best predictive performance, compared to the GARCH and GJR models.

| | PPS | #Vio | QS | %Hit | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLike | $R^2$Log |
|---|---|---|---|---|---|---|---|---|---|---|
| **SIM I** | | | | | | | | | | |
| GARCH* | 1.794 | 09 | 0.044 | 0.011 | 0.001 | 0.020 | 0.016 | 0.069 | 1.689 | 0.001 |
| SRN-GARCH | 1.795 | 09 | 0.044 | 0.011 | 0.001 | 0.057 | 0.022 | 0.101 | 1.690 | 0.001 |
| **SIM II** | | | | | | | | | | |
| GARCH | 3.013 | 14 | 0.145 | 0.008 | 0.390 | 88.970 | 0.485 | 6.018 | 4.207 | 0.046 |
| SRN-GARCH* | 3.009 | 12 | 0.142 | 0.008 | 0.256 | 85.151 | 0.349 | 4.739 | 4.195 | 0.022 |
| GJR | 3.011 | 16 | 0.144 | 0.008 | 0.331 | 77.430 | 0.445 | 5.503 | 4.204 | 0.039 |
| SRN-GJR* | **3.003** | **10** | **0.139** | **0.009** | **0.023** | **5.071** | **0.114** | **1.339** | **4.186** | **0.003** |
| EGARCH | 3.005 | **10** | 0.142 | 0.006 | 0.109 | 22.250 | 0.251 | 3.078 | 4.191 | 0.013 |
| SRN-EGARCH* | 3.004 | **10** | 0.140 | 0.008 | 0.054 | 12.876 | 0.173 | 2.153 | 4.188 | 0.007 |

**Table 4.7.** Simulation: one-step-ahead forecast comparison. For the QS and %Hit measures, the results are calculated at the 1%-quantile. For the SIM II data, the bold numbers denote the best scores. For each pair of the RECH and GARCH-type models, the asterisk indicates the models having better forecast performance.

### Simulation study III (SIM III)

This simulation study examines if the RECH models are able to simulate the long-memory volatility, by fitting the FIGARCH$(1,d,1)$ model of Baillie et al. (1996) to data generated from the RECH models. The FIGARCH$(1,d,1)$ model is defined as:

$$y_t = \sigma_t \epsilon_t, \epsilon_t \sim \mathcal{N}(0,1), t = 1,2,...,T,$$
$$\sigma_t^2 = \omega + \beta\sigma_{t-1}^2 + \left[1 - \beta L - (1-\psi L)(1-L)^d\right] y_t^2,$$

where $d \in (0,1)$ is the fractional integrated parameter and $L$ is the backshift operator. The parameter $\Theta = (\omega,\psi,d,\beta)$. When $d = 0$, the FIGARCH becomes a GARCH model. When $d > 0$ and is close to 1, the persistence of the past shocks in the FIGARCH process decays at a slow hyperbolic rate (Baillie et al., 1996); hence the FIGARCH process exhibits the long-memory effects in its volatility dynamics.

We use the SRN-GARCH model as the true data generating process (DGP) with the four different parameter sets $\theta_i, i = 1,..,4$, listed in Table 4.8. These are the estimated parameters obtained in Section 4.4.2 when the SRN-GARCH model is fitted to four real datasets. For each parameter set $\theta_i$, 500 datasets of $T = 3000$ observations are generated from each of two different specifications of the SRN-GARCH model: $\beta_1 = 0$ and $\beta_1$ equals to the true values in Table 4.8, i.e. $\beta_1 \neq 0$. We note that if $\beta_1 = 0$, the DGP is the

### 4.4.1 Simulation studies

| | $\alpha$ | $\beta$ | $\beta_0$ | $\beta_1$ | $v_0$ | $v_1$ | $v_2$ | $w$ | $b$ |
|---|---|---|---|---|---|---|---|---|---|
| $\theta_1$ | 0.058 | 0.681 | 0.068 | 0.418 | $-0.018$ | $-0.430$ | 0.524 | 0.161 | $-0.173$ |
| $\theta_2$ | 0.071 | 0.690 | 0.075 | 0.362 | 0.062 | $-0.422$ | 0.538 | 0.087 | $-0.130$ |
| $\theta_3$ | 0.076 | 0.744 | 0.016 | 0.388 | $-0.075$ | $-0.574$ | 0.400 | $-0.040$ | $-0.023$ |
| $\theta_4$ | 0.057 | 0.562 | 0.101 | 0.413 | 0.015 | $-0.380$ | 0.652 | 0.270 | $-0.170$ |

**Table 4.8.** SIM III: The parameters used in the DGP.

GARCH(1,1) model. To generate each time series, we generate 10,000 observations and use the last 3,000 for the simulation data. We then use the Matlab MFE toolbox [3], with the default settings, to produce the Quasi-Maximum Likelihood Estimate (QMLE) of the parameter $\widehat{\Theta}_i, i = 1,..,4$, of the FIGARCH(1,$d$,1) model. Table 4.9 shows the means and standard deviations averaged over 500 QMLE estimates of the FIGARCH(1,$d$,1) parameters.

| | DGP is GARCH(1,1) ($\beta_1 = 0$) | | | | DGP is SRN-GARCH ($\beta_1 \neq 0$) | | | |
|---|---|---|---|---|---|---|---|---|
| | $\widehat{\omega}$ | $\widehat{\psi}$ | $\widehat{d}$ | $\widehat{\beta}$ | $\widehat{\omega}$ | $\widehat{\psi}$ | $\widehat{d}$ | $\widehat{\beta}$ |
| $\widehat{\Theta}_1$ | 0.211 | 0.423 | 0.017 | 0.373 | 0.088 | 0.107 | 0.728 | 0.708 |
| | (0.043) | (0.159) | (0.039) | (0.127) | (0.014) | (0.052) | (0.109) | (0.078) |
| $\widehat{\Theta}_2$ | 0.225 | 0.366 | 0.036 | 0.323 | 0.126 | 0.152 | 0.608 | 0.647 |
| | (0.064) | (0.206) | (0.053) | (0.155) | (0.020) | (0.058) | (0.12) | (0.100) |
| $\widehat{\Theta}_3$ | 0.045 | 0.210 | 0.093 | 0.222 | 0.095 | 0.105 | 0.711 | 0.653 |
| | (0.018) | (0.214) | (0.059) | (0.162) | (0.016) | (0.053) | (0.110) | (0.089) |
| $\widehat{\Theta}_4$ | 0.235 | 0.437 | 0.002 | 0.376 | 0.096 | 0.124 | 0.706 | 0.711 |
| | (0.019) | (0.127) | (0.014) | (0.125) | (0.017) | (0.054) | (0.116) | (0.084) |

**Table 4.9.** SIM III: Means and standard deviations (in brackets) of 500 QMLE estimates of the FIGARCH(1,$d$,1) model parameters.

The important conclusion from Table 4.9 is that the short-memory and long-memory properties of the GARCH(1,1) and SRN-GARCH models, respectively, are distinguishable. When the DGP is the GARCH(1,1), the estimates of the fractional integrated parameter $\widehat{d}$ are insignificant in all cases, suggesting that there is no evidence of the long-memory effects in the volatility of the GARCH(1,1) model. When the DGP is the SRN-GARCH model, i.e. $\beta_1 \neq 0$, the estimates of the

[3] https://github.com/bashtage/mfe-toolbox/

fractional integrated parameter $\widehat{d}$ are close to 1 in all cases, implying the existence of long-memory in the volatility dynamics of simulated time series, which are generated from SRN-GARCH models. The difference in the QMLE estimates of the parameter $d$ between the two DGPs in Table 4.9 implies that the SRN-GARCH model is able to simulate long-memory volatility effects. We observe similar results for the SRN-GJR and SRN-EGARCH models.

### 4.4.2 Applications to stock market returns

We demonstrate the performance of the RECH models using four stock index datasets: the Standard and Poor's 500 Index (SP500), the Japanese Nikkei 225 Index (N225), the Russell 2000 Index (RUT) and the German stock index (DAX). The datasets were downloaded from the Realized Library of The Oxford-Man Institute[4]. We used the daily closing prices $\{P_t, t=1,...,T_P\}$ and calculated the demeaned return process as:

$$y_t = 100\left(\log\frac{P_{t+1}}{P_t} - \frac{1}{T_P-1}\sum_{i=1}^{T_P-1}\log\frac{P_{i+1}}{P_i}\right), \quad t=1,2,...,T_P-1. \tag{4.11}$$

The length of the four return series is fixed to be $T=4000$, with $T=T_P-1$, and each series is divided into an in-sample period of the first $T_{\text{in}}=2000$ observations and an out-of-sample period of the last $T_{\text{out}}=2000$ observations. Table 4.10 summarizes the datasets.

| | In-sample Period | Out-of-sample Period | $T_{\text{in}}$ | $T_{\text{out}}$ |
|---|---|---|---|---|
| SP500 | 27 Feb 2004 – 06 Feb 2012 | 06 Feb 2012 – 24 Jan 2020 | 2000 | 2000 |
| N225 | 16 Sep 2003 – 16 Nov 2011 | 17 Nov 2011 – 24 Jan 2020 | 2000 | 2000 |
| RUT | 24 Feb 2004 – 01 Feb 2012 | 02 Feb 2012 – 24 Jan 2020 | 2000 | 2000 |
| DAX | 06 Aug 2003 – 02 Nov 2011 | 02 Nov 2011 – 24 Jan 2020 | 2000 | 2000 |

**Table 4.10.** Description of the four index datasets.

Table 4.11 reports some descriptive statistics for these four datasets together with the modified R/S test (Lo, 1991) for long-range memory in the logarithm of the squared returns. Lo's modified R/S test is widely used in the financial time series literature; see, e.g., Lo (1991), Giraitis et al. (2003), Breidt et al. (1998). All the index data exhibit

[4]https://realized.oxford-man.ox.ac.uk/

some negative skewness, a high excess kurtosis and high variation. The N225 returns are more skewed and leptokurtic than those of the SP500, RUT and DAX data. The result of Lo's modified R/S test for long-memory dependence with several different lags $q$ indicates that there is significant evidence of long-memory dependence in the SP500, RUT and DAX stock indices. For the N225 data, however, the evidence of long memory is less clear as the null hypothesis of short memory for the squared returns is not rejected at the 5% level of significance when $q=20$ and $q=30$.

| | Min | Max | Std | Skew | Kurtosis | $V_n(10)$ | $V_n(20)$ | $V_n(30)$ |
|---|---|---|---|---|---|---|---|---|
| SP500 | −9.351 | 10.220 | 1.307 | −0.256 | 12.502 | 3.188* | 2.412* | 2.047* |
| | | | | | | 2.664* | 2.040* | 1.748* |
| N225 | −10.563 | 11.658 | 1.224 | −0.585 | 18.171 | 2.768* | 2.171* | 1.905* |
| | | | | | | 1.956* | 1.566 | 1.415 |
| RUT | −8.391 | 8.056 | 1.364 | −0.130 | 8.764 | 3.065* | 2.385* | 2.055* |
| | | | | | | 2.459* | 1.943* | 1.691 |
| DAX | −7.437 | 9.993 | 1.267 | 0.115 | 10.960 | 3.226* | 2.501* | 2.146* |
| | | | | | | 2.456* | 1.926* | 1.670 |

**Table 4.11.** Descriptive statistics for the demeaned returns of the SP500, N225, RUT and DAX datasets. $V_n(q)$, $q=10$, 20 and 30, are the test statistics of Lo's modified R/S test of long memory with lag $q$. Upper and lower values of the 3 last columns are Lo's test statistics for absolute and squared returns, respectively. The asterisks indicate significance at the 5% level.

The Realized Library provides different realized measures[5] that can be used in financial econometrics as a proxy to the latent $\sigma_t^2$. We use the following six common realized measures including Realized Variance (RV) (Andersen and Bollerslev, 1998), Bipower Variation (BV) (Barndorff-Nielsen and Shephard, 2004), Median Realized Volatility (MedRV) (Andersen et al., 2012), Realized Kernel Variance (Barndorff-Nielsen et al., 2008) with the Non-Flat Parzen kernal ($RKV_1$) and the Tukey-Hanning kernal ($RKV_2$) , to evaluate the forecast performance of the volatility models using the predictive scores in Table 4.4. Shephard and Sheppard (2010) give more details about the Realized Library.

[5]See https://realized.oxford-man.ox.ac.uk/documentation/estimators for the list of the available realized measures

Denote by $RV_t$ the realized measure of $\sigma_t^2$ at time $t$. As the realized measures ignore the variation of the prices overnight and sometimes the variation in the first few minutes of the trading day when recorded prices may contain large errors (Shephard and Sheppard, 2010), we follow Hansen and Lunde (2005) to scale the realized measure $RV_t$ as

$$\widetilde{\sigma}_t^2 = \widehat{c} \cdot RV_t \quad \text{where} \quad \widehat{c} = \frac{T_{\text{out}}^{-1} \sum_{t=T_{\text{in}}+1}^{T} [y_t - \mathbb{E}(y_t|\mathcal{F}_{t-1})]^2}{T_{\text{out}}^{-1} \sum_{t=T_{in}+1}^{T} RV_t}, \quad t = T_{\text{in}} + 1, 2, ..., T, \tag{4.12}$$

with $\mathbb{E}(y_t|\mathcal{F}_{t-1}) = 0$, and use $\widetilde{\sigma}_t^2$ as the estimate of the latent conditional variance $\sigma_t^2$; see Table 4.10 for a definition of $T_{\text{in}}$ and $T_{\text{out}}$ used in our datasets. See Martens (2002) and Fleming et al. (2003) for a similar scaling estimator of the daily volatility.

### In-sample analysis

Table 4.12 and 4.13 summarize the estimation results of fitting the benchmark GARCH-type models and their RECH counterparts to the SP500, N225, RUT and DAX datasets. The posterior mean estimates and posterior standard deviation estimates are obtained using the SMC with likelihood annealing sampler. We draw the following conclusions from the estimation results.

First, the marginal likelihood estimates show that the RECH models fit the index datasets better than the GARCH-type models, except for the SRN-EGARCH model for the N225 data. For example, for the SP500 data, the Bayes factors of the SRN-GARCH, SRN-GJR and SRN-EGARCH models compared to the GARCH, GJR and EGARCH models are roughly $e^{36}$, $e^{28}$ and $e^{10}$, respectively, which decisively support the RECH models. Among the benchmark GARCH-type models, the EGARCH model constantly has the highest marginal likelihood.

Second, the estimated posterior means of the parameter $\beta_1$ of the RECH models are more than two standard deviations from zero in all cases, providing evidence of the volatility effects rather than linearity, e.g. probably non-linearity and long-memory effects, in the volatility dynamics and also suggesting that the recurrent component of the RECH models is able to effectively detect these effects. Additionally, the coefficients $v_2$ of the RECH models are statistically significant, indicating that the RECH models are able to detect the serial dependence rather than linearity that the previous conditional variance $\sigma_{t-1}^2$ has on $\sigma_t^2$.

### 4.4.2 Applications to stock market returns

| | $\omega$ | $\alpha$ | $\beta$ | $\gamma$ | $\beta_0$ | $\beta_1$ | $v_1$ | $v_2$ | Mar.llh |
|---|---|---|---|---|---|---|---|---|---|
| **SP500** | | | | | | | | | |
| GARCH | **0.016** | **0.093** | **0.894** | | | | | | $-2778.3$ |
| | (0.004) | (0.011) | (0.012) | | | | | | (0.113) |
| SRN-GARCH | | **0.057** | **0.562** | | **0.101** | **0.413** | **-0.380** | **0.652** | $-2742.3$* |
| | | (0.011) | (0.082) | | (0.026) | (0.063) | (0.076) | (0.167) | (0.284) |
| GJR | **0.024** | **0.040** | **0.891** | **0.065** | | | | | $-2764.4$ |
| | (0.004) | (0.011) | (0.009) | (0.011) | | | | | (0.121) |
| SRN-GJR | | **0.011** | **0.685** | **0.109** | **0.078** | **0.349** | **-0.222** | **0.581** | $-2736.6$* |
| | | (0.008) | (0.109) | (0.026) | (0.036) | (0.102) | (0.093) | (0.196) | (0.323) |
| EGARCH | **0.004** | **0.136** | **0.978** | **-0.126** | | | | | $-2754.8$ |
| | (0.002) | (0.016) | (0.003) | (0.013) | | | | | (0.133) |
| SRN-EGARCH | **-0.196** | **0.106** | **0.972** | **-0.236** | **0.066** | **0.343** | **0.104** | **0.538** | $-2744.0$* |
| | (0.092) | (0.030) | (0.016) | (0.044) | (0.027) | (0.099) | (0.103) | (0.198) | (0.225) |
| **N225** | | | | | | | | | |
| GARCH | **0.033** | **0.138** | **0.839** | | | | | | $-2800.1$ |
| | (0.008) | (0.016) | (0.018) | | | | | | (0.119) |
| SRN-GARCH | | **0.076** | **0.744** | | **0.016** | **0.388** | **-0.574** | **0.400** | $-2766.8$* |
| | | (0.016) | (0.051) | | (0.013) | (0.076) | (0.156) | (0.147) | (0.279) |
| GJR | **0.048** | **0.060** | **0.835** | **0.100** | | | | | $-2787.5$ |
| | (0.009) | (0.015) | (0.020) | (0.018) | | | | | (0.121) |
| SRN-GJR | | **0.066** | **0.734** | **0.029** | **0.027** | **0.386** | **-0.531** | **0.429** | $-2769.2$* |
| | | (0.021) | (0.061) | (0.036) | (0.015) | (0.081) | (0.146) | (0.183) | (0.313) |
| EGARCH | **-0.003** | **0.215** | **0.967** | **-0.134** | | | | | $-2772.0$ |
| | (0.004) | (0.022) | (0.006) | (0.015) | | | | | (0.133) |
| SRN-EGARCH | **-0.088** | **0.255** | **0.990** | **-0.164** | **0.033** | **0.344** | **-0.355** | **0.434** | $-2772.5$ |
| | (0.048) | (0.046) | (0.009) | (0.019) | (0.020) | (0.117) | (0.148) | (0.325) | (0.225) |

**Table 4.12.** SP500 and N225 data: Posterior means (in bold) of the parameters with the posterior standard deviations (in brackets). The last column shows the estimated log marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the SMC using the likelihood annealing algorithm. The asterisks indicate the cases when the Bayes Factors strongly support the RECH models over their corresponding GARCH-type models.

| | $\omega$ | $\alpha$ | $\beta$ | $\gamma$ | $\beta_0$ | $\beta_1$ | $v_1$ | $v_2$ | Mar.llh |
|---|---|---|---|---|---|---|---|---|---|
| **RUT** | | | | | | | | | |
| GARCH | **0.036** | **0.112** | **0.863** | | | | | | −3037.6 |
| | (0.009) | (0.015) | (0.019) | | | | | | (0.117) |
| SRN-GARCH | | **0.071** | **0.690** | | **0.075** | **0.362** | **-0.422** | **0.538** | −3014.3* |
| | | (0.017) | (0.078) | | (0.040) | (0.082) | (0.118) | (0.203) | (0.284) |
| GJR | **0.049** | **0.048** | **0.864** | **0.084** | | | | | −3025.4 |
| | (0.009) | (0.013) | (0.017) | (0.015) | | | | | (0.121) |
| SRN-GJR | | **0.019** | **0.780** | **0.109** | **0.078** | **0.311** | **-0.255** | **0.428** | −3010.6* |
| | | (0.008) | (0.109) | (0.026) | (0.036) | (0.102) | (0.093) | (0.196) | (0.323) |
| EGARCH | **0.013** | **0.162** | **0.969** | **-0.111** | | | | | −3023.1 |
| | (0.004) | (0.021) | (0.006) | (0.015) | | | | | (0.130) |
| SRN-EGARCH | **-0.090** | **0.156** | **0.990** | **-0.161** | **0.050** | **0.307** | **-0.282** | **0.238** | −3015.5* |
| | (0.049) | (0.027) | (0.010) | (0.019) | (0.018) | (0.115) | (0.205) | (0.254) | (0.225) |
| **DAX** | | | | | | | | | |
| GARCH | **0.019** | **0.096** | **0.890** | | | | | | −2902.0 |
| | (0.005) | (0.013) | (0.014) | | | | | | (0.119) |
| SRN-GARCH | | **0.058** | **0.681** | | **0.068** | **0.418** | **-0.430** | **0.524** | −2867.2* |
| | | (0.019) | (0.126) | | (0.037) | (0.059) | (0.131) | (0.281) | (0.279) |
| GJR | **0.031** | **0.046** | **0.884** | **0.066** | | | | | −2889.1 |
| | (0.006) | (0.010) | (0.013) | (0.013) | | | | | (0.120) |
| SRN-GJR | | **0.035** | **0.711** | **0.072** | **0.067** | **0.369** | **-0.301** | **0.575** | −2866.4* |
| | | (0.016) | (0.088) | (0.032) | (0.038) | (0.095) | (0.117) | (0.213) | (0.313) |
| EGARCH | **0.006** | **0.158** | **0.975** | **-0.116** | | | | | −2872.5 |
| | (0.002) | (0.018) | (0.004) | (0.014) | | | | | (0.128) |
| SRN-EGARCH | **-0.063** | **0.169** | **0.989** | **-0.127** | **0.034** | **0.265** | **-0.185** | **0.193** | −2869.0* |
| | (0.063) | (0.026) | (0.011) | (0.020) | (0.024) | (0.132) | (0.160) | (0.394) | (0.232) |

**Table 4.13.** RUT and DAX data: Posterior means (in bold) of the parameters with the posterior standard deviations (in brackets). The last column shows the estimated log marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the SMC using the likelihood annealing algorithm. The asterisks indicate the cases when the Bayes Factors strongly support the RECH models over their corresponding GARCH-type models.

Third, the existence of the leverage effects in the volatility is clear across all four stock markets. The leverage parameters $\gamma$ of the GJR and EGARCH models are statistically significant, implying that these models can detect the asymmetric volatility. All the leverage effect-related parameters $\gamma$ and $v_1$ in the RECH models are statistically significant, except the parameter $v_1$ of the SRN-EGARCH model. In particular, the *linear* leverage coefficient $\gamma$ of the SRN-GRJ and SRN-EGARCH models are significant, similarly to those of the GRJ and EGARCH models. Interestingly, the *non-linear* leverage coefficient $v_1$ of the RECH models is significant in almost all cases, suggesting that the spillover effect of asymmetric volatility can be in other form rather than just linearity. In particular, the leverage coefficient $v_1$ of the SRN-GARCH model is also statistically significant across all markets; i.e., unlike the conventional GARCH model, SRN-GARCH can detect the leverage effects in volatility.

Lastly, as pointed out by a reviewer, in the EGARCH case the $\alpha$ and $\beta$ appear to be lessed affected by adding the recurrent component. This is because the EGARCH structure is very different from GARCH and GJR (see Table 4.1), the EGARCH part in SRN-EGARCH is already non-linear in $\sigma_{t-1}$ hence it can accommodate non-linearity. This doesn't mean that SRN-EGARCH cannot improve EGARCH - $\beta_1$ is still far away from zero and the improvement is evident in better marginal likelihood and better out-of-sample prediction as confirmed in Section 4.4.2.

Figure 4.4 shows the volatility estimated by the GARCH and SRN-GARCH models for the SP500 index data, together with the values of the recurrent component $\omega_t$ at all time points. Figure B.2 and B.3 in the Appendix B are similar plots for the SRN-GJR and SRN-EGARCH models. Clearly, the recurrent component $\omega_t$ is responsive to the changes in the volatility dynamics: it is small during the low volatility periods and large in the high volatility periods. This distinct behavior of financial volatility is well-captured by the recurrent neural network structure of the recurrent component $\omega_t$.

Figure 4.5 plots the standardized residuals $\widehat{\epsilon}_t$ from the GARCH and SRN-GARCH models together with their QQ-plots. The Appendix B gives similar plots for the SRN-GRJ and GRJ models, the SRN-EGARCH and EGARCH models. Generally the RECH residuals appear to lie closer to the expected straight line than those of the counterpart GARCH-type models.

Table 4.14 provides the skewness and kurtosis statistics together with the $p$-values of the Ljung-Box (LB) autocorrelation test of the residuals and squared residuals

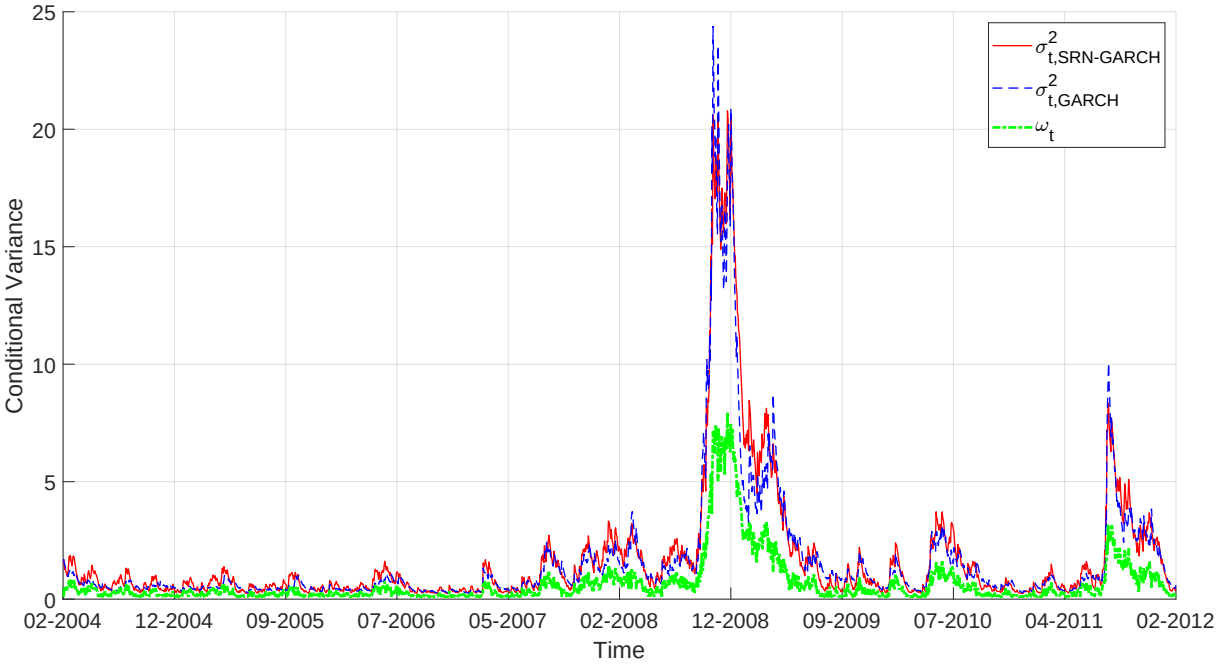


**Figure 4.4.** SP500: The in-sample conditional variance of the GARCH (dashed line) and SRN-GARCH (solid line) at all time points. The bottom line shows the values of the recurrent component $\omega_t$ of the SRN-GARCH specification. (The figure is better viewed in colour).

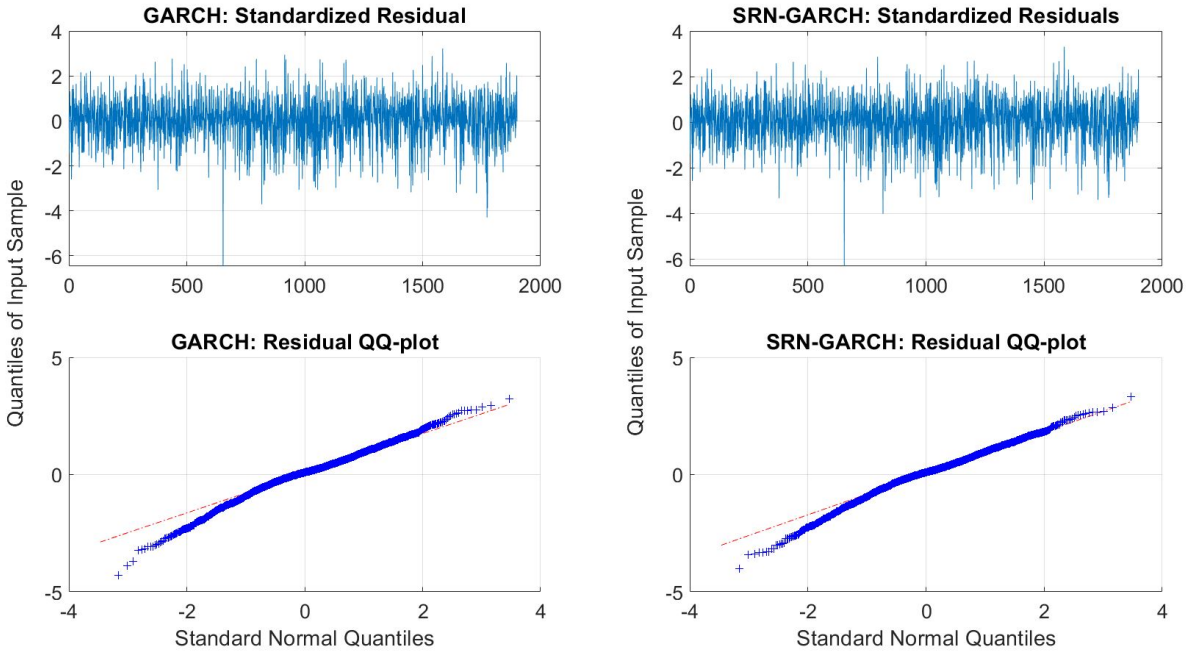


**Figure 4.5.** SP500: Estimated residuals $\widehat{\epsilon}_t$ of the SRN-GARCH and GARCH models and their Q-Q plots.

estimated by the RECH and the benchmark GARCH-type models. The $p$-value of the LB test, together with the sample ACF plots, of the standardized and squared

| | Fitted Conditional Variance | | | | Residual $\widehat{\epsilon}_t$ | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Skew | Kurtosis | Std | Skew | Kurtosis | LB-$\widehat{\epsilon}_t$ |
| **S&P500** | | | | | | | | |
| GARCH | 1.650 | 2.845 | 4.543 | 26.522 | 1.002 | −0.505 | 4.535 | 0.054 |
| SRN-GARCH | 1.755 | 2.910 | 4.045 | 21.251 | 1.002 | −0.509 | 4.219 | 0.119 |
| GJR | 1.423 | 2.342 | 4.585 | 27.169 | 1.001 | −0.485 | 4.202 | 0.058 |
| SRN-GJR | 1.600 | 2.877 | 4.445 | 25.585 | 1.001 | −0.515 | 4.200 | 0.110 |
| EGARCH | 1.534 | 2.318 | 4.291 | 25.389 | 0.999 | −0.587 | 4.616 | 0.051 |
| SRN-EGARCH | 1.603 | 2.887 | 4.661 | 29.080 | 0.999 | −0.524 | 4.125 | 0.107 |
| **N225** | | | | | | | | |
| GARCH | 1.493 | 3.133 | 7.792 | 72.005 | 0.999 | −0.528 | 5.037 | 0.138 |
| SRN-GARCH | 1.390 | 2.521 | 6.950 | 58.482 | 0.999 | −0.407 | 4.275 | 0.108 |
| GJR | 1.352 | 2.692 | 7.780 | 72.782 | 1.003 | −0.455 | 4.509 | 0.104 |
| SRN-GJR | 1.434 | 2.634 | 7.054 | 60.510 | 1.003 | −0.404 | 4.235 | 0.085 |
| EGARCH | 1.370 | 2.282 | 7.252 | 68.428 | 1.001 | −0.404 | 4.197 | 0.073 |
| SRN-EGARCH | 1.242 | 2.430 | 7.605 | 73.471 | 1.001 | −0.378 | 4.158 | 0.083 |
| **RUT** | | | | | | | | |
| GARCH | 1.803 | 2.524 | 4.177 | 23.108 | 1.000 | −0.318 | 3.717 | 0.046 |
| SRN-GARCH | 1.826 | 2.407 | 3.746 | 18.897 | 1.000 | −0.379 | 3.728 | 0.084 |
| GJR | 1.619 | 2.122 | 4.343 | 25.208 | 1.002 | −0.322 | 3.663 | 0.034 |
| SRN-GJR | 1.831 | 2.635 | 4.000 | 21.316 | 1.002 | −0.364 | 3.725 | 0.085 |
| EGARCH | 1.681 | 1.973 | 3.775 | 20.933 | 0.999 | −0.397 | 3.843 | 0.061 |
| SRN-EGARCH | 1.634 | 2.246 | 4.245 | 24.624 | 0.999 | −0.373 | 3.690 | 0.057 |
| **DAX** | | | | | | | | |
| GARCH | 1.598 | 2.203 | 4.332 | 26.122 | 1.002 | −0.462 | 4.870 | 0.957 |
| SRN-GARCH | 1.656 | 2.378 | 3.484 | 16.904 | 1.002 | −0.468 | 4.323 | 0.893 |
| GJR | 1.400 | 1.739 | 3.815 | 19.772 | 1.001 | −0.399 | 4.478 | 0.957 |
| SRN-GJR | 1.510 | 2.013 | 3.517 | 17.005 | 1.001 | −0.420 | 4.228 | 0.911 |
| EGARCH | 1.504 | 1.766 | 3.438 | 17.766 | 1.000 | −0.414 | 4.116 | 0.905 |
| SRN-EGARCH | 1.262 | 1.730 | 4.132 | 23.497 | 1.000 | −0.385 | 4.066 | 0.916 |

**Table 4.14.** SP500: Model diagnostics of the fitted conditional variance and residual $\widehat{\epsilon}_t$. The LB p-values denote the p-value from the Ljung-Box test with 10 lags.

standardized residuals suggest that there is no evidence of autocorrelation. The residuals produced by all models in Table 4.14 exhibit some negative skewness and have kurtosis values higher than 3 (the kurtosis of the standard normal distribution). In general, the residuals of the RECH models seem closer to normality than those of the corresponding GARCH-type models. Similarly to the GARCH-type models, it is straightforward to use Student's $t$ distribution for the innovation in the RECH models to improve the residual diagnostics; however, this extension is not considered here.

### Out-of-sample analysis

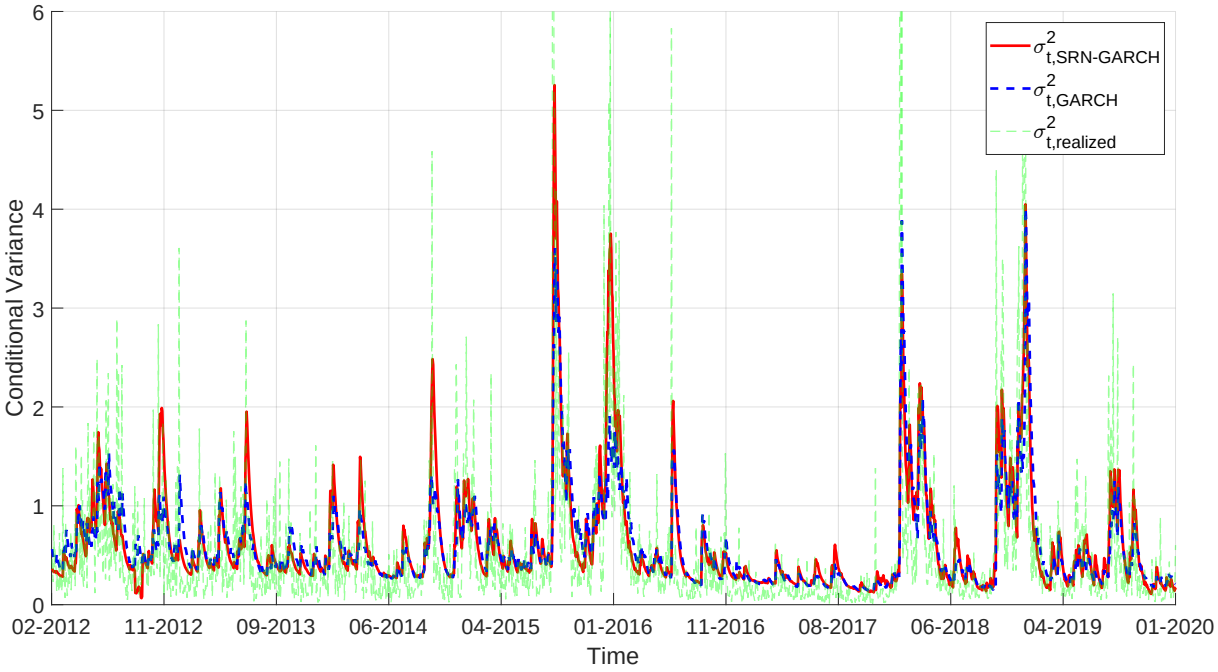


**Figure 4.6.** SP500 data: Forecast conditional variance by the GARCH (dashed) and SRN-GARCH (solid) models, together with the realized variance (dotted). (The figure is better viewed in colour).

Figure 4.6 plots the one-step-ahead forecast conditional variance of the GARCH and SRN-GARCH models, together with the out-of-sample realized variance of the S&P500 data, obtained by Data annealing SMC. Figure B.6 and B.7 in the Appendix B are similar plots for the case of the SRN-GJR and SRN-EGARCH models, respectively. To save space, we do not report the plots for the N225, RUT and DAX datasets as similar behaviors of the forecast variance are observed for these datasets. We note from Figure 4.6, B.6 and B.7 that in general, the RECH models and their GARCH-type counterparts produce forecast variances that adequately track the movement of the

realized variance. The forecast variance of the RECH and benchmark models are similar in the low volatility regions while the RECH models have higher variance forecasts during high volatility periods. The variance forecasts of the RECH models seem to track the realized variance better than those of the benchmark GARCH-type models.

The in-sample analysis suggests that the RECH models fit the in-sample data of the four index datasets better than the counterpart GARCH-type models. However, it is possible that the superior in-sample performance is the result of overfitting (Pagan and Schwert, 1990; Donaldson and Kamstra, 1997). Table 4.15 provides summary statistics on the one-step-ahead forecasts of conditional variance and standardized residuals. The most important conclusion from Table 4.15 is that the RECH models do not overfit the data, as the forecast conditional variance of the RECH models are not excessively variable and the forecast residuals of the RECH models are very close to those of the GARCH-type benchmark models. The RECH models occasionally produce one-step-ahead forecast residuals with lower kurtosis than the counterpart GARCH-type models.

Table 4.16 shows the forecast performance of the models using the four predictive scores PPS, #Vio, QS and %Hit. As these four predictive scores complement each other, for each pair of the RECH and GARCH-type models, we compare their forecast performance by counting the number of times one model has a better predictive score than the other and report this count in the last column of Table 4.16. The model with the higher count is preferred. Table 4.16 shows that the RECH models consistently outperform their counterpart GARCH-type models for the S&P500, RUT and DAX data. For the N225 index, the predictive improvement of the RECH models over the benchmark counterparts is less clear, especially for the SRN-GARCH and GARCH models.

Tables 4.17 to 4.20 summarize the forecast performance measured by the predictive scores defined in Table 4.4. Each table contains six panels, corresponding to the six realized measures mentioned earlier. For each pair of the RECH and GARCH-type models, their forecast performance are also compared in the same way as in Table 4.16. Additionally, in each panel, bold numbers are used to indicate the lowest forecast errors. For each type of realized measure, the model with the highest number of lowest forecast errors is preferred. The table shows that the RECH models in general outperform their counterpart GARCH-type models.

In particular, for the SP500, N225 and RUT datasets, the RECH models consistently perform the best across all panels. For example, the forecast results in Table 4.17 show that the SRN-GARCH model has highest numbers of lowest forecast errors in all panels, implying that the SRN-GARCH model forecasts volatility the best for the SP500 data. For the N225 and RUT datasets, the SRN-EGARCH model clearly outperforms the other RECH and benchmark models in all realized measures. The superior predictive performance of the RECH models over the GARCH-type counterparts provides further evidence to support the conclusion that the RECH models do not overfit the index datasets.

As mentioned in Section 4.3.2, we now discuss an useful feature of the RECH models; that is, the volatility estimates and volatility forecasts of the RECH specifications are less sensitive to the choice of the structure for the garch component than a single GARCH-type model. For example, for each dataset in Table 4.12 and 4.13, we compute the difference between the highest and lowest marginal likelihood estimates among the RECH specifications and calculate the same value for the GARCH-type benchmark models. Table 4.21 shows that these discrepancies of in-sample performance among the RECH models are much smaller than those of the GARCH-type models, across all datasets. For each panel in Tables 4.17 to 4.20, we compute the difference between the highest and lowest forecast scores among the RECH models and do the same for the benchmark models; Figure 4.7 plots the results.

| | SP500 | N225 | RUT | DAX |
|---|---|---|---|---|
| GARCH-type models | 21.7 | 30.3 | 12.6 | 29.8 |
| RECH models | 6.9 | 6.8 | 5.5 | 3.3 |

**Table 4.21.** Applications: The difference between the highest and lowest marginal likelihood estimates of the RECH and the benchmark models across all in-sample data. The numbers are in the natural log scale.

The comparison results in Figure 4.7 indicate that the discrepancies of out-of-sample performance among the RECH models are consistently lower than those of the GARCH-type models, except for the $MSE_2$ score for the N225 data. The result that the model performance is less sensitive to the RECH specification is useful in practice as users do not need to worry about which specification should be used for their financial dataset.

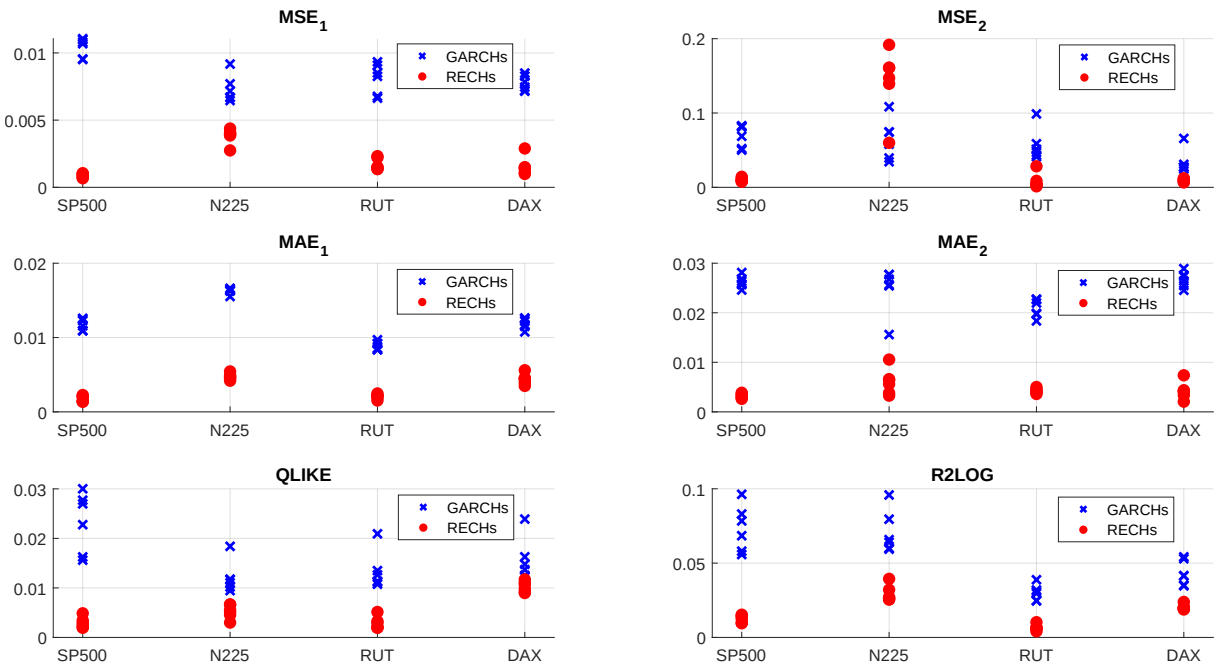


**Figure 4.7.** Applications: The difference between the highest and lowest forecast scores of the RECH and the benchmark models. In each subplot, each column contains 6 values of the RECH models and 6 values of the benchmark models, corresponding to 6 realized measures.

## 4.5 Chapter summary

We propose a new class of conditional heteroskedastic models, which we call the RECH models, by incorporating a RNN structure into the conditional variance of the GARCH-type models, and study in detail three RECH specifications: SRN-GARCh, SRN-GJR and SRN-EGARCH. We use Sequential Monte Carlo with likelihood annealing and data annealing for in-sample Bayesian inference and out-of-sample forecasting, respectively. We also use the estimate of marginal likelihood as a by-product of the SMC for model choice. The extensive simulation and empirical studies suggest that the RECH models not only have both attractive in-sample performance and accurate out-of-sample forecasts, but can also explain the volatility movement.

The GARCH-type and SV-type models are two main classes of volatility models that are widely used in the financial volatility literature. The most important difference between the two class is the the SV models interpret the volatility as a separated stochastic process, instead of a deterministic function of the past observations and conditional variance. This distinct design of SV model makes it and its variants less popular in the volatility modeling due to the fact that the likelihood functions

of SV-type models are intractable, and hence make it more challenging to efficiently estimate the models, especially when Bayesian inference are preferred over frequentist approaches. Consequently, it is possibly more challenging to efficiently combing the SV-type models with deep learning based structures because the proposed hybrid models will be inherently intractable and the volatility dynamic is captured in a completely different way compared to the GARCH-type models. The next chapter will discuss an enhanced version of the SV model by using the SRU model, with some important modifications, to improve the capability of learning non-linear effects and long memory dependency of the SV-type models.

| | Forecast Conditional Variance | | | | Forecast Residual $\widehat{\epsilon}_t$ | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Skew | Kurtosis | Std | Skew | Kurtosis | LB-$\widehat{\epsilon}_t$ |
| **SP500** | | | | | | | | |
| GARCH | 0.597 | 0.476 | 2.805 | 14.391 | 0.932 | −0.818 | 6.438 | 0.471 |
| SRN-GARCH | 0.620 | 0.546 | 2.309 | 9.009 | 0.921 | −0.697 | 5.243 | 0.592 |
| GJR | 0.566 | 0.417 | 2.856 | 14.351 | 0.924 | −0.738 | 5.775 | 0.484 |
| SRN-GJR | 0.626 | 0.565 | 2.443 | 9.868 | 0.932 | −0.730 | 5.477 | 0.537 |
| EGARCH | 0.632 | 0.566 | 2.404 | 10.518 | 0.914 | −0.815 | 5.939 | 0.438 |
| SRN-EGARCH | 0.634 | 0.615 | 2.945 | 13.554 | 0.910 | −0.749 | 5.592 | 0.598 |
| **N225** | | | | | | | | |
| GARCH | 0.930 | 1.061 | 4.553 | 30.093 | 0.977 | −1.327 | 13.908 | 0.184 |
| SRN-GARCH | 0.884 | 0.921 | 3.752 | 21.485 | 1.004 | −1.640 | 14.399 | 0.089 |
| GJR | 0.903 | 1.046 | 5.145 | 38.668 | 0.986 | −1.495 | 15.828 | 0.128 |
| SRN-GJR | 0.889 | 0.944 | 3.761 | 21.200 | 1.006 | −1.661 | 18.781 | 0.094 |
| EGARCH | 0.917 | 1.245 | 6.677 | 69.000 | 1.007 | −1.599 | 17.672 | 0.146 |
| SRN-EGARCH | 0.897 | 1.231 | 6.858 | 74.418 | 1.017 | −1.668 | 18.652 | 0.116 |
| **RUT** | | | | | | | | |
| GARCH | 0.928 | 0.459 | 1.961 | 9.011 | 0.959 | −0.471 | 4.101 | 0.721 |
| SRN-GARCH | 0.953 | 0.531 | 2.139 | 9.850 | 0.947 | −0.547 | 4.378 | 0.562 |
| GJR | 0.919 | 0.448 | 2.034 | 8.473 | 0.953 | −0.462 | 4.084 | 0.689 |
| SRN-GJR | 0.958 | 0.548 | 2.258 | 10.044 | 0.940 | −0.465 | 4.080 | 0.572 |
| EGARCH | 0.990 | 0.545 | 1.828 | 8.450 | 0.932 | −0.492 | 4.273 | 0.631 |
| SRN-EGARCH | 0.964 | 0.594 | 3.273 | 12.528 | 0.937 | −0.462 | 4.106 | 0.618 |
| **DAX** | | | | | | | | |
| GARCH | 0.867 | 0.474 | 1.238 | 4.394 | 0.973 | −0.245 | 4.923 | 0.152 |
| SRN-GARCH | 0.904 | 0.585 | 1.566 | 5.807 | 0.962 | −0.235 | 4.992 | 0.125 |
| GJR | 0.829 | 0.440 | 1.661 | 6.735 | 0.977 | −0.235 | 4.733 | 0.097 |
| SRN-GJR | 0.907 | 0.594 | 1.646 | 6.226 | 0.964 | −0.250 | 4.969 | 0.107 |
| EGARCH | 0.924 | 0.617 | 1.584 | 6.046 | 0.967 | −0.296 | 5.205 | 0.124 |
| SRN-EGARCH | 0.902 | 0.634 | 1.987 | 7.982 | 0.972 | −0.228 | 4.956 | 0.105 |

**Table 4.15.** Application: Summary statistics on the one-step-ahead out-of-sample forecast conditional variances $\widehat{\sigma}_t^2$ and residual $\widehat{\epsilon}_t$. The LB p-values denote the p-value from the Ljung-Box test with 10 lags.

| | PPS | # Violation | QS | Hit Per. | Count |
|---|---|---|---|---|---|
| **SP500** | | | | | |
| GARCH | 0.993 | 32 | 0.026 | 0.018 | 0 |
| SRN-GARCH* | 0.955 | 25 | 0.024 | 0.017 | 4 |
| GJR | 0.981 | 27 | 0.025 | 0.018 | 0 |
| SRN-GJR* | 0.959 | 25 | 0.024 | 0.017 | 4 |
| EGARCH | 0.963 | 29 | 0.025 | 0.018 | 0 |
| SRN-EGARCH* | 0.963 | 23 | 0.025 | 0.015 | 2 |
| **N225** | | | | | |
| GARCH* | 1.214 | 32 | 0.036 | 0.016 | 2 |
| SRN-GARCH | 1.216 | 33 | 0.035 | 0.016 | 1 |
| GJR | 1.217 | 31 | 0.036 | 0.017 | 1 |
| SRN-GJR* | 1.216 | 32 | 0.035 | 0.017 | 2 |
| EGARCH | 1.212 | 32 | 0.035 | 0.017 | 0 |
| SRN-EGARCH* | 1.212 | 30 | 0.035 | 0.017 | 1 |
| **RUT** | | | | | |
| GARCH | 1.298 | 34 | 0.032 | 0.018 | 0 |
| SRN-GARCH* | 1.286 | 26 | 0.030 | 0.016 | 4 |
| GJR | 1.290 | 31 | 0.031 | 0.017 | 0 |
| SRN-GJR* | 1.283 | 24 | 0.030 | 0.015 | 4 |
| EGARCH | 1.285 | 23 | 0.030 | 0.014 | 0 |
| SRN-EGARCH* | 1.281 | 23 | 0.030 | 0.014 | 1 |
| **DAX** | | | | | |
| GARCH | 1.257 | 47 | 0.031 | 0.020 | 0 |
| SRN-GARCH* | 1.247 | 38 | 0.029 | 0.020 | 3 |
| GJR | 1.249 | 50 | 0.030 | 0.022 | 0 |
| SRN-GJR* | 1.246 | 41 | 0.029 | 0.022 | 3 |
| EGARCH | 1.246 | 39 | 0.028 | 0.018 | 1 |
| SRN-EGARCH* | 1.243 | 37 | 0.029 | 0.020 | 3 |

**Table 4.16.** Applications: Forecast performance of the RECH and benchmark GARCH-type models. For the QS and %Hit measures, the results are calculated at the 1%-quantile. For each pair of the RECH and GARCH-type models, the asterisks indicate the model with the higher count.

| Estimator | | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | $QLIKE_2$ | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|
| | GARCH | 0.088 | 0.780 | 0.220 | 0.351 | 0.115 | 0.715 | 0 |
| | SRN-GARCH* | **0.075** | 0.703 | **0.203** | 0.325 | **0.078** | **0.631** | 6 |
| BV | GJR | 0.077 | 0.721 | 0.206 | **0.322** | 0.097 | 0.678 | 1 |
| | SRN-GJR* | 0.076 | **0.691** | 0.205 | 0.328 | 0.080 | 0.642 | 5 |
| | EGARCH | 0.078 | 0.695 | 0.207 | 0.332 | 0.091 | 0.644 | 1 |
| | SRN-EGARCH* | **0.075** | 0.694 | 0.204 | 0.326 | 0.081 | 0.646 | 5 |
| | GARCH | 0.097 | 0.600 | 0.237 | 0.367 | 0.127 | 0.911 | 0 |
| | SRN-GARCH* | **0.084** | 0.538 | **0.222** | 0.346 | **0.093** | 0.815 | 6 |
| $RKV_1$ | GJR | 0.085 | 0.545 | 0.225 | **0.339** | 0.111 | 0.873 | 1 |
| | SRN-GJR* | 0.084 | 0.531 | 0.224 | 0.348 | 0.094 | 0.827 | 5 |
| | EGARCH | 0.086 | **0.528** | 0.224 | 0.349 | 0.098 | **0.810** | 2 |
| | SRN-EGARCH | 0.085 | 0.538 | 0.224 | 0.349 | 0.097 | 0.834 | 2 |
| | GARCH | 0.071 | 0.422 | 0.200 | 0.315 | 0.135 | 0.575 | 0 |
| | SRN-GARCH* | **0.061** | 0.371 | **0.187** | 0.296 | **0.109** | **0.507** | 6 |
| $RKV_2$ | GJR | 0.062 | 0.374 | 0.188 | **0.288** | 0.121 | 0.542 | 2 |
| | SRN-GJR* | **0.061** | **0.366** | 0.189 | 0.299 | 0.110 | 0.517 | 3 |
| | EGARCH | 0.063 | 0.368 | 0.190 | 0.301 | 0.117 | 0.514 | 2 |
| | SRN-EGARCH* | 0.062 | 0.376 | 0.188 | 0.298 | 0.111 | 0.521 | 4 |
| | GARCH | 0.102 | 0.637 | 0.246 | 0.386 | 0.083 | 0.946 | 0 |
| | SRN-GARCH* | **0.087** | 0.555 | **0.227** | **0.357** | **0.039** | **0.849** | 6 |
| MedRV | GJR | 0.091 | 0.586 | 0.234 | 0.360 | 0.067 | 0.909 | 1 |
| | SRN-GJR* | 0.088 | 0.544 | 0.230 | 0.361 | 0.043 | 0.863 | 5 |
| | EGARCH | 0.091 | 0.550 | 0.232 | 0.366 | 0.051 | 0.863 | 1 |
| | SRN-EGARCH* | **0.087** | **0.540** | 0.229 | 0.359 | 0.045 | 0.867 | 5 |
| | GARCH | 0.096 | 1.124 | 0.226 | 0.361 | 0.105 | 0.782 | 0 |
| | SRN-GARCH* | **0.083** | 1.054 | **0.212** | 0.340 | **0.073** | 0.701 | 6 |
| RV | GJR | 0.084 | 1.061 | 0.214 | **0.333** | 0.091 | 0.751 | 1 |
| | SRN-GJR* | 0.084 | 1.039 | 0.214 | 0.343 | 0.075 | 0.713 | 3 |
| | EGARCH* | 0.085 | **1.037** | 0.214 | 0.345 | 0.075 | **0.695** | 3 |
| | SRN-EGARCH | 0.084 | 1.045 | 0.214 | 0.344 | 0.078 | 0.720 | 2 |

**Table 4.17.** SP500 data: Forecast performance of the RECH and benchmark GARCH-type models using different realized measures. For each pair of the RECH and GARCH-type models, the asterisks indicate the model with the higher count. In each panel, the bold numbers indicate the best predictive scores.

| Estimator | | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | $QLIKE_2$ | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|
| | GARCH | 0.138 | 1.924 | 0.242 | 0.526 | 0.562 | 0.537 | 0 |
| | SRN-GARCH* | **0.124** | 1.795 | 0.227 | **0.487** | 0.557 | 0.491 | 6 |
| BV | GJR | 0.133 | 1.896 | 0.234 | 0.505 | 0.561 | 0.526 | 0 |
| | SRN-GJR* | 0.125 | **1.792** | 0.227 | 0.491 | 0.553 | 0.485 | 6 |
| | EGARCH | 0.130 | 1.870 | 0.226 | 0.499 | 0.553 | 0.478 | 0 |
| | SRN-EGARCH* | **0.124** | 1.794 | **0.224** | **0.487** | **0.552** | **0.472** | 6 |
| | GARCH | 0.216 | 3.792 | 0.311 | 0.661 | 0.573 | 1.028 | 0 |
| | SRN-GARCH* | **0.200** | **3.635** | 0.297 | 0.624 | 0.568 | 0.956 | 6 |
| $RKV_1$ | GJR | 0.210 | 3.758 | 0.304 | 0.641 | 0.574 | 1.012 | 0 |
| | SRN-GJR* | 0.201 | 3.648 | 0.298 | 0.628 | 0.565 | 0.952 | 6 |
| | EGARCH | 0.206 | 3.761 | 0.295 | 0.634 | 0.567 | 0.939 | 0 |
| | SRN-EGARCH* | **0.200** | 3.668 | **0.293** | **0.623** | **0.564** | **0.930** | 6 |
| | GARCH | 0.121 | 1.421 | 0.228 | 0.492 | 0.580 | 0.460 | 0 |
| | SRN-GARCH* | **0.108** | **1.268** | 0.215 | 0.455 | 0.575 | 0.420 | 6 |
| $RKV_2$ | GJR | 0.116 | 1.404 | 0.221 | 0.473 | 0.577 | 0.449 | 0 |
| | SRN-GJR* | **0.108** | 1.277 | 0.214 | 0.457 | **0.569** | 0.411 | 6 |
| | EGARCH | 0.114 | 1.438 | 0.213 | 0.466 | **0.569** | 0.405 | 0 |
| | SRN-EGARCH* | **0.108** | 1.304 | **0.211** | **0.454** | **0.569** | **0.400** | 5 |
| | GARCH | 0.132 | 1.575 | 0.252 | 0.518 | 0.541 | 0.655 | 0 |
| | SRN-GARCH* | **0.118** | **1.415** | 0.238 | 0.484 | 0.530 | 0.600 | 6 |
| MedRV | GJR | 0.129 | 1.574 | 0.248 | 0.507 | 0.540 | 0.651 | 0 |
| | SRN-GJR* | **0.118** | 1.421 | 0.238 | 0.486 | 0.525 | 0.593 | 6 |
| | EGARCH | 0.125 | 1.636 | 0.238 | 0.502 | 0.524 | 0.580 | 0 |
| | SRN-EGARCH* | **0.118** | 1.458 | **0.234** | **0.482** | **0.523** | **0.576** | 6 |
| | GARCH | 0.145 | 2.068 | 0.246 | 0.535 | 0.577 | 0.557 | 0 |
| | SRN-GARCH* | **0.132** | **1.918** | 0.233 | 0.500 | 0.572 | 0.510 | 6 |
| RV | GJR | 0.141 | 2.046 | 0.239 | 0.515 | 0.576 | 0.546 | 0 |
| | SRN-GJR* | 0.133 | 1.927 | 0.232 | 0.501 | **0.567** | 0.503 | 6 |
| | EGARCH | 0.138 | 2.067 | 0.231 | 0.508 | 0.569 | 0.496 | 0 |
| | SRN-EGARCH* | 0.133 | 1.948 | **0.229** | **0.498** | **0.567** | **0.490** | 6 |

**Table 4.18.** N225 data: Forecast performance of the RECH and benchmark GARCH-type models using different realized measures. For each pair of the RECH and GARCH-type models, the asterisks indicate the model with the higher count. In each panel, the bold numbers indicate the best predictive scores.

| Estimator | | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | $QLIKE_2$ | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|
| | GARCH | 0.099 | 0.593 | 0.246 | 0.468 | 0.708 | 0.534 | 0 |
| | SRN-GARCH* | 0.087 | 0.509 | 0.232 | 0.440 | 0.687 | 0.482 | 6 |
| BV | GJR | 0.089 | 0.538 | 0.236 | 0.443 | 0.694 | 0.499 | 0 |
| | SRN-GJR* | 0.087 | 0.509 | 0.233 | 0.442 | 0.686 | 0.686 | 6 |
| | EGARCH | 0.092 | 0.527 | 0.242 | 0.460 | 0.693 | 0.502 | 0 |
| | SRN-EGARCH* | **0.085** | **0.505** | **0.231** | **0.436** | **0.684** | **0.475** | 6 |
| BV | GJR | 0.089 | 0.538 | 0.236 | 0.443 | 0.694 | 0.499 | 0 |
| | GARCH | 0.121 | 0.620 | 0.278 | 0.518 | 0.740 | 0.713 | 0 |
| | SRN-GARCH* | 0.108 | 0.543 | 0.266 | 0.493 | 0.720 | 0.651 | 6 |
| $RKV_1$ | GJR | 0.110 | 0.569 | 0.268 | 0.497 | 0.725 | 0.669 | 0 |
| | SRN-GJR* | 0.109 | 0.549 | 0.266 | 0.497 | 0.718 | 0.652 | 5 |
| | EGARCH | 0.113 | 0.562 | 0.273 | 0.510 | 0.725 | 0.670 | 0 |
| | SRN-EGARCH* | **0.106** | **0.540** | **0.263** | **0.491** | **0.714** | **0.640** | 6 |
| | GARCH | 0.097 | 0.524 | 0.247 | 0.464 | 0.708 | 0.565 | 0 |
| | SRN-GARCH* | **0.087** | **0.459** | 0.234 | 0.437 | 0.690 | 0.514 | 6 |
| $RKV_2$ | GJR | 0.090 | 0.483 | 0.238 | 0.441 | 0.697 | 0.537 | 0 |
| | SRN-GJR* | 0.088 | 0.464 | 0.235 | 0.440 | 0.690 | 0.520 | 6 |
| | EGARCH | 0.092 | 0.474 | 0.241 | 0.451 | 0.694 | 0.534 | 0 |
| | SRN-EGARCH* | **0.087** | 0.462 | **0.233** | **0.435** | **0.688** | **0.512** | 6 |
| | GARCH | 0.137 | 0.950 | 0.289 | 0.553 | 0.669 | 0.819 | 0 |
| | SRN-GARCH* | 0.123 | 0.824 | 0.275 | 0.524 | 0.642 | 0.763 | 6 |
| MedRV | GJR | 0.127 | 0.879 | 0.278 | 0.527 | 0.654 | 0.784 | 0 |
| | SRN-GJR* | 0.123 | 0.817 | 0.276 | 0.525 | 0.641 | 0.764 | 6 |
| | EGARCH | 0.128 | 0.839 | 0.284 | 0.543 | 0.645 | 0.784 | 0 |
| | SRN-EGARCH* | **0.119** | **0.793** | **0.274** | **0.519** | **0.638** | **0.758** | 6 |
| | GARCH | 0.100 | 0.562 | 0.250 | 0.475 | 0.716 | 0.546 | 0 |
| | SRN-GARCH* | 0.089 | **0.490** | 0.235 | 0.443 | 0.698 | 0.492 | 6 |
| RV | GJR | 0.092 | 0.513 | 0.239 | 0.449 | 0.704 | 0.510 | 0 |
| | SRN-GJR* | 0.090 | 0.492 | 0.236 | 0.445 | 0.697 | 0.494 | 6 |
| | EGARCH | 0.094 | 0.507 | 0.244 | 0.462 | 0.704 | 0.513 | 0 |
| | SRN-EGARCH* | **0.088** | 0.492 | **0.234** | **0.441** | **0.695** | **0.486** | 6 |

**Table 4.19.** RUT data: Forecast performance of the RECH and benchmark GARCH-type models using different realized measures. For each pair of the RECH and GARCH-type models, the asterisks indicate the model with the higher count. In each panel, the bold numbers indicate the best predictive scores.

| Estimator | | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | $QLIKE_2$ | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|
| | GARCH | 0.083 | 0.572 | 0.216 | 0.404 | 0.618 | 0.449 | 0 |
| | SRN-GARCH* | 0.078 | **0.542** | 0.211 | 0.399 | 0.611 | 0.419 | 6 |
| BV | GJR* | **0.075** | **0.542** | **0.205** | **0.379** | 0.609 | 0.417 | 4 |
| | SRN-GJR | 0.078 | 0.545 | 0.210 | 0.399 | 0.606 | 0.412 | 2 |
| | EGARCH | 0.078 | 0.547 | 0.210 | 0.401 | 0.602 | 0.408 | 1 |
| | SRN-EGARCH* | 0.077 | 0.548 | 0.207 | 0.395 | **0.601** | **0.401** | 5 |
| | GARCH | 0.103 | 0.691 | 0.245 | 0.452 | 0.628 | 0.609 | 0 |
| | SRN-GARCH* | 0.098 | 0.672 | 0.240 | 0.448 | 0.621 | 0.567 | 6 |
| $RKV_1$ | GJR* | **0.094** | **0.664** | **0.234** | **0.426** | 0.619 | 0.570 | 4 |
| | SRN-GJR | 0.098 | 0.676 | 0.239 | 0.448 | 0.616 | 0.560 | 2 |
| | EGARCH | 0.099 | 0.678 | 0.240 | 0.452 | 0.613 | 0.556 | 1 |
| | SRN-EGARCH* | 0.097 | 0.681 | 0.237 | 0.447 | **0.612** | **0.549** | 5 |
| | GARCH | 0.067 | 0.396 | 0.197 | 0.368 | 0.623 | 0.354 | 0 |
| | SRN-GARCH* | 0.064 | 0.375 | 0.192 | 0.363 | 0.621 | 0.333 | 6 |
| $RKV_2$ | GJR* | **0.060** | **0.369** | **0.185** | **0.341** | 0.614 | 0.324 | 4 |
| | SRN-GJR | 0.064 | 0.379 | 0.191 | 0.363 | 0.615 | 0.326 | 2 |
| | EGARCH | 0.064 | 0.380 | 0.191 | 0.366 | **0.610** | 0.320 | 1 |
| | SRN-EGARCH* | 0.063 | 0.382 | 0.188 | 0.359 | **0.610** | **0.314** | 4 |
| | GARCH | 0.084 | 0.525 | 0.223 | 0.416 | 0.586 | 0.490 | 0 |
| | SRN-GARCH* | 0.077 | 0.460 | 0.215 | 0.402 | 0.574 | 0.454 | 6 |
| MedRV | GJR | 0.077 | 0.490 | 0.213 | **0.392** | 0.578 | 0.462 | 2 |
| | SRN-GJR* | 0.076 | 0.462 | 0.214 | 0.402 | 0.569 | 0.448 | 4 |
| | EGARCH | 0.076 | 0.458 | 0.213 | 0.403 | **0.562** | 0.436 | 1 |
| | SRN-EGARCH* | **0.074** | **0.450** | **0.211** | 0.396 | 0.563 | **0.433** | 5 |
| | GARCH | 0.087 | 0.637 | 0.218 | 0.410 | 0.621 | 0.456 | 0 |
| | SRN-GARCH* | 0.083 | 0.619 | 0.213 | 0.407 | 0.616 | 0.426 | 6 |
| RV | GJR* | **0.079** | **0.613** | **0.206** | **0.384** | 0.613 | 0.423 | 4 |
| | SRN-GJR | 0.083 | 0.624 | 0.214 | 0.408 | 0.612 | 0.421 | 2 |
| | EGARCH | 0.083 | 0.624 | 0.214 | 0.412 | 0.608 | 0.415 | 1 |
| | SRN-EGARCH* | 0.082 | 0.627 | 0.210 | 0.405 | **0.607** | **0.409** | 5 |

**Table 4.20.** DAX data: Forecast performance of the RECH and benchmark GARCH-type models using different realized measures. For each pair of the RECH and GARCH-type models, the asterisks indicate the model with the higher count. In each panel, the bold numbers indicate the best predictive scores.

# Chapter 5

# SRSV: A Statistical Recurrent Stochastic Volatility Model

The Stochastic Volatility (SV) model and its variants are widely used in the financial sector while recurrent neural network (RNN) models are successfully used in many large-scale industrial applications of Deep Learning. Our article combines these two methods in a non-trivial way and proposes a model, which we call the Statistical Recurrent Stochastic Volatility (SR-SV) model, to capture the dynamics of stochastic volatility. The proposed model is able to capture complex volatility effects (e.g., non-linearity and long-memory auto-dependence) overlooked by the conventional SV models, is statistically interpretable and has an impressive out-of-sample forecast performance. These properties are carefully discussed and illustrated through extensive simulation studies and applications to five international stock index datasets: The German stock index DAX30, the Hong Kong stock index HSI50, the France market index CAC40, the US stock market index SP500 and the Canada market index TSX250.

## 5.1 Motivation and Contribution

The volatility of a financial time series, such as stock returns, is defined as the variance of the returns and serves as a measure of the uncertainty about the returns. The volatility, which is of great interest to financial econometricians, is unobserved and therefore often modelled statistically in order to estimate it. The two model classes most frequently used in volatility modelling are the GARCH models and the Stochastic Volatility (SV) models. The GARCH model expresses the current volatility, conditional on the previous returns and volatilities, as a *deterministic* and linear function of the squared returns and the conditional volatilities in the previous time period. The SV model (Taylor, 1982, 1986), on the other hand, uses a latent stochastic process to model the volatility, which is usually taken as a first order autoregressive process. It is well documented that the GARCH and SV models are able to capture important effects exhibited in the variance of financial returns. For example, the volatilities in financial returns are observed to be highly autocorrelated in certain time periods and exhibit periods of both low and high volatility. This so-called volatility clustering phenomenon can be modeled by the volatility processes introduced in the GARCH and SV models, making these volatility models widely employed in financial time series modelling.

Although the GARCH and SV models were independently and almost concurrently introduced, the GARCH models were initially more widely adopted as it is much easier to estimate GARCH models than SV models. This is because the likelihood of a GARCH model can be obtained explicitly, while the likelihood of a SV model is intractable as it is an integral over the latent process. However, the conditional variance process of GARCH models is deterministic and hence GARCH models might not capture efficiently the random oscillatory behavior of financial volatility (Nelson, 1991). SV models are considered as an attractive alternative to GARCH models because they overcome this limitation (Kim et al., 1998; Yu, 2002). Recent advances in Bayesian computation such as particle Markov chain Monte Carlo (PMCMC) (Andrieu et al., 2010) allow straightforward estimation and inference for SV models.

Standard SV models still cannot appropriately capture some important features arising in financial volatility. For example, a large amount of both theoretical and empirical evidence indicates that there exists long-range persistence in the volatility process of many financial returns, see, e.g, Lo (1991); Ding et al. (1993); Crato and de Lima (1994) and Bollerslev and Mikkelsen (1996). The long-memory property of a time series

implies that the decay of the autocorrelations of the series is slower than exponential. The standard SV model of (Taylor, 1982) uses an AR(1) process to model the log of the volatility and hence might fail to capture this type of persistence (Breidt et al., 1998). Another line of the literature shows strong evidence of non-linear auto-dependence in the volatility process of some stock and currency exchange returns (Kiliç, 2011) and that the simple linear AR(1) process cannot effectively capture the underlying non-linear volatility dynamics.

Breidt et al. (1998) proposed the Long Memory Stochastic Volatility (LMSV) model to overcome the short-memory limitation of the standard SV model. LMSV uses an ARFIMA process (Granger and Joyeux, 1980) as an alternative to the AR(1) process to capture the long-memory dependence in the volatility. The empirical evidence in Breidt et al. (1998) suggests that the LMSV model is able to capture the long-memory volatility behaviour in some stock return datasets. However, the literature is unclear about whether the LMSV model can capture non-linear dynamics within the volatility process, because the ARFIMA model is linear. Additionally, it is challenging to estimate the LMSV model as its likelihood is intractable. We are unaware of any available software package that implements the LMSV methodology. In another approach, Yu et al. (2006) introduced a family of non-linear SV (N-SV) models to capture the possible departure from the log transform commonly used in SV models. In the standard SV model, the logarithm of volatility is assumed to follow an AR(1) process; N-SV uses other non-linear transformations, such as the Box-Cox power function, rather than the logarithm. The simulation studies and empirical results on currency exchange and option pricing data in Yu et al. (2006) show that the N-SV model using the Box-Cox transformation is able to detect some interesting effects in the underlying volatility process. The general use of N-SV models requires the user to select an appropriate non-linear transformation for the dataset under consideration, and this might lead to a challenging model selection problem. Neither Breidt et al. (1998) nor Yu et al. (2006) clearly discussed the out-of-sample forecast performance of their LMSV and N-SV models.

Recurrent neural networks (RNN) in the Deep Learning literature have impressive prediction performance and have been successfully deployed in a large number of industrial-level applications (language translation, image captioning, speech synthesis, etc.). The RNN models are well-known for their ability to efficiently capture the long-range memory and non-linear dependence existing within various types of

sequential data, and are considered as the state-of-the-art models for many sequence learning problems (Lipton et al., 2015). Many researchers and practitioners have used RNN for mean modelling in financial time series analysis, but the general consensus is that these machine learning models do not clearly outperform the traditional time series models such as ARMA and ARIMA (see, e.g., Makridakis et al. (2018b) and Zhang (2003b)). Makridakis et al. (2018b) note that without careful modifications, Machine Learning models are usually less accurate than the statistical approaches that have been extensively investigated in the financial time series literature. Recently, the idea of using the RNN models to improve the predictive performance of GARCH-type models has also been proposed for volatility modelling. For example, Kim and Won (2018) use the volatility estimates from several GARCH-type models as inputs to a RNN model, which then non-linearly transforms these inputs to output the final estimate of the volatility. The empirical results on the Korean stock market KOPSI 200 index show a significant improvement of forecast performance of the proposed hybrid model over several GARCH-type benchmark models. However, similar to many engineering-oriented Machine Learning models, Kim and Won's model overlooks the interpretation aspect in volatility modelling, which is often of main interest to econometricians. One of the main motivations of our article is to develop deep learning based volatility models that are not only able to produce accurate prediction, but also interpretable and have meaningful in-sample analysis. These models should not overlook the well-established features of traditional econometric models, that are motivated by the well-known stylized facts in financial time series such as volatility clustering and fat tails.

In the SV literature, there is still lack of research using RNN structures to model the stochastic volatility dynamics of financial time series, perhaps because of two reasons. First, it is non-trivial to sensibly incorporate RNN into the statistical volatility models. Simple adaptations of RNN to volatility models easily overlook the important stylized facts exhibited in financial volatility, which are well captured by the AR(1) process in the SV model. It is important to select appropriate RNN structures that are not only able to produce accurate out-of-sample volatility forecast, but also explain well the volatility dynamics. Second, a stochastic volatility model that incorporates a RNN structure into its latent stochastic process is highly sophisticated and thus challenging to estimate.

This paper combines the SV and RNN models in a non-trivial way, and proposes a new model, called the Statistical Recurrent Stochastic Volatility (SR-SV) model. In particular, we use the Statistical Recurrent Unit (SRU) structure of Oliva et al. (2017), which is a special type of RNN models, to capture complex volatility effects overlooked by an AR(1) process in the standard SV model but still retain the essential components of the SV model. This combination allows the SR-SV model to enjoy much of advances from both worlds of deep learning (e.g., flexibility and excellent predictive performance) and econometric volatility modelling (e.g., excellent interpretability of volatility effects). The SR-SV model belongs to the class of parametric state space models whose Bayesian inference can be performed using recent advances in the Sequential Monte Carlo (SMC) and particle MCMC literature (Andrieu and Roberts, 2009; Andrieu et al., 2010; Duan and Fulop, 2015; Deligiannidis et al., 2018). The simulation studies and empirical results on the five stock index datasets demonstrate that the SR-SV model can efficiently capture the potential non-linear and long-memory effects in the underlying volatility dynamics, and provide better out-of-sample forecasts than the standard SV, N-SV and LMSV models. We note that we have tested SR-SV on a wider range of stock returns but only report in the paper the results for five of them, as we constantly observed a similar improvement of the model compared to the other three counterpart. A Matlab software package implementing Bayesian estimation and inference for SR-SV together with the examples reported in this chapter are available at `https://github.com/vbayeslab`.

## 5.2 The SR-SV model

### 5.2.1 The SV model and its possible weaknesses

Let $y=\{y_t, t=1,...,T\}$ be a series of financial returns. We consider a basic version of SV models (Taylor, 1982)

$$z_t = \mu + \phi(z_{t-1}-\mu) + \epsilon_t^z, \ \ \epsilon_t^z \sim \mathcal{N}(0,\sigma^2), \ \ t=2,...,T, \ \ z_1 \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{1-\phi^2}\right), \quad (5.1)$$

$$y_t = e^{\frac{1}{2}z_t}\epsilon_t^y, \ \ \epsilon_t^y \sim \mathcal{N}(0,1), \ \ t=1,2,...,T. \quad (5.2)$$

The persistence parameter $\phi$ is assumed to be in $(-1,1)$ to enforce stationarity of both the $z$ and $y$ processes. In this SV model, the log volatility process $z$ is assumed to follow an AR(1) model. It is well documented in the financial econometrics literature

### 5.2.1 The SV model and its possible weaknesses

that financial time series data often exhibit a long-term auto-dependence, which forces the persistence parameter $\phi$ to be close to 1 (Jacquier et al., 1994; Kim et al., 1998). Write $p(z|\theta)$ for the density of $z$ given the model parameters $\theta = (\mu,\phi,\sigma^2)$ and $p(y|z)$ for the density of the data $y$ conditional on $z$. We can view $p(z|\theta)$ as the prior with $\theta$ being the hyper-parameters and $p(y|z)$ as the likelihood (Jacquier et al., 1994). Under this perspective, the SV model (5.1)-(5.2) puts non-zero prior mass on AR(1) stochastic processes, and zero or almost-zero mass on stochastic processes that are far from being well approximated by an AR(1). This means that the SV model in (5.1)-(5.2) might not be able to capture more complex dynamics in the posterior behavior of the log volatility process $z$, such as long-term memory or non-linear auto-dependence, and that a more flexible prior distribution should be put on $z$. We will design such a flexible prior by combining the attractive features from both SV and RNN time series modeling techniques.

Yu et al. (2006) propose a class of non-linearity N-SV models as a variant of SV which allows a more flexible link between the variance $\text{Var}(y_t|z_t)$ and the AR(1) process $z$. Their N-SV model, using the Box-Cox transformation for $\text{Var}(y_t|z_t)$, is written as

$$z_t = \mu + \phi(z_{t-1} - \mu) + \epsilon_t^z,\ \epsilon_t^z \sim \mathcal{N}(0,\sigma^2),\ \ t = 2,...,T,\ z_1 \sim \mathcal{N}\Big(\mu, \frac{\sigma^2}{1-\phi^2}\Big), \quad (5.3)$$

$$y_t = (1+\delta z_t)^{1/2\delta}\epsilon_t^y,\ \ \epsilon_t^y \sim \mathcal{N}(0,1),\ \ t = 1,2,...,T, \quad (5.4)$$

where $\delta$ is the auxiliary parameter that measures the degree of non-linearity rather than the log transform. As $\delta \to 0$, $(1+\delta z_t)^{1/2\delta} \to e^{\frac{1}{2}z_t}$ and hence the N-SV model includes the SV model as a special case. The term non-linearity here might cause some confusion, as it does not refer to the non-linear auto-dependence within the log volatility process $z$, but the non-linearity between $\text{Var}(y_t|z_t)$ and $z_t$.

Breidt et al. (1998) suggest to use an ARFIMA$(p,d,q)$ process (Granger and Joyeux, 1980; Hosking, 1981) for the log volatility $z_t$ to capture the long-memory auto-dependence exhibited in financial time series. Their LMSV model is written as

$$(1-B)^d\Phi(B)z_t = \Theta(B)\eta_t,\ \ \eta_t \sim \mathcal{N}(0,\sigma_\eta^2), \quad (5.5)$$

$$y_t = \sigma_t\epsilon_t,\ \ \sigma_t = \kappa e^{\frac{1}{2}z_t},\ \ \epsilon_t \sim \mathcal{N}(0,1),\ \ t = 1,2,...,T, \quad (5.6)$$

where $\Phi(B) = 1-\phi_1 B-\phi_2 B^2-...-\phi_p B^p$, $\Theta(B) = 1+\theta_1 B+\theta_2 B^2+...+\theta_q B^q$, and $B$ is a backshift operator, i.e., $B^s X_t = X_{t-s}$. To ensure the stationarity and invertability of the log volatility process $z_t$, the fractional integration parameter $d$ is assumed to be in $(-0.5,0.5)$ and the roots of $\Phi(B)$ and $\Theta(B)$ have to lie outside the unit circle.

Another notable line of research in the SV literature is the class of semi-parametric stochastic volatility models that incorporate non-parametric techniques into modelling the conditional distribution of financial returns. For example, the stochastic volatility, Dirichlet process mixture (SV-DPM) model of Jensen and Maheu (2010) uses a Dirichlet process prior (Ferguson, 1973) to characterize the conditional distribution of $y_t$. Semi-parametric models are different from the SR-SV model in two important aspects. First, the SV-DPM model is proposed to capture the asymmetries and leptokurtotic behaviors of financial returns, while the SR-SV model focuses on modeling the non-linearity and long-memory auto-dependence in the log-volatility dynamics. Second, the SV-DPM model is a semi-parametric model in the sense that the model cannot be described using a finite number of parameters as it uses a non-parametric prior, e.g. Dirichlet process, to simulate the conditional return and retains the parametric structure, e.g. AR(1), of the log-volatility in the standard SV model. The SR-SV model, on the other hand, is a *fully parametric* model whose mathematical representation will be discussed in Section 5.2.2. Our article therefore uses parametric models including the standard SV, N-SV and LMSV models as the benchmarks to evaluate the SR-SV model.

### 5.2.2 The SR-SV model

This section proposes the SR-SV model that combines SV and SRU for financial volatility modelling. The key idea is that we use the SRU structure to capture the complicated effects such as long-term memory and non-linear auto-dependence, in the volatility dynamics that are overlooked by the basic SV models. This leads to a prior distribution for the log volatility process $z$ that is much more flexible than the AR(1) prior (c.f. Section 5.2.1). Our proposed SR-SV model is as follows

$$
\begin{aligned}
r_t &= \Phi(w_h h_{t-1} + b_r) && (5.7)\\
\varphi_t &= \Phi(w_r r_t + w_\eta \eta_{t-1} + w_z z_{t-1} + b_\varphi) && (5.8)\\
h_t &= \alpha h_{t-1} + (1-\alpha)\varphi_t && (5.9)\\
\eta_t &= \beta_0 + \beta_1 h_t + \epsilon_t^\eta, \;\; \epsilon_t^\eta \overset{iid}{\sim} \mathcal{N}(0,\sigma^2), \;\; t = 1,2,...,T && (5.10)\\
z_t &= \eta_t + \phi z_{t-1}, \;\; t = 1,...,T, && (5.11)\\
y_t &= e^{\frac{1}{2}z_t}\epsilon_t^y, \;\; \epsilon_t^y \overset{iid}{\sim} \mathcal{N}(0,1), \;\; t = 1,2,...,T, && (5.12)
\end{aligned}
$$

that is, we use a SRU to model the dynamics of the hidden states $h_t$. Here, $\Phi(\cdot)$ is the ReLU activation function, $\Phi(x) = \max(0,x)$ and $z_0$ is the initial value of the log volatility process and a convenient choice of $z_0$ is the log of the unconditional variance of the observed series $y$, i.e., $z_0 = \log(\text{var}(y))$. We follow the literature to initialize $h_1 = 0$ as the recurrent units initially have no memory. Figure 5.1 plots the graphical representation of the SR-SV model.

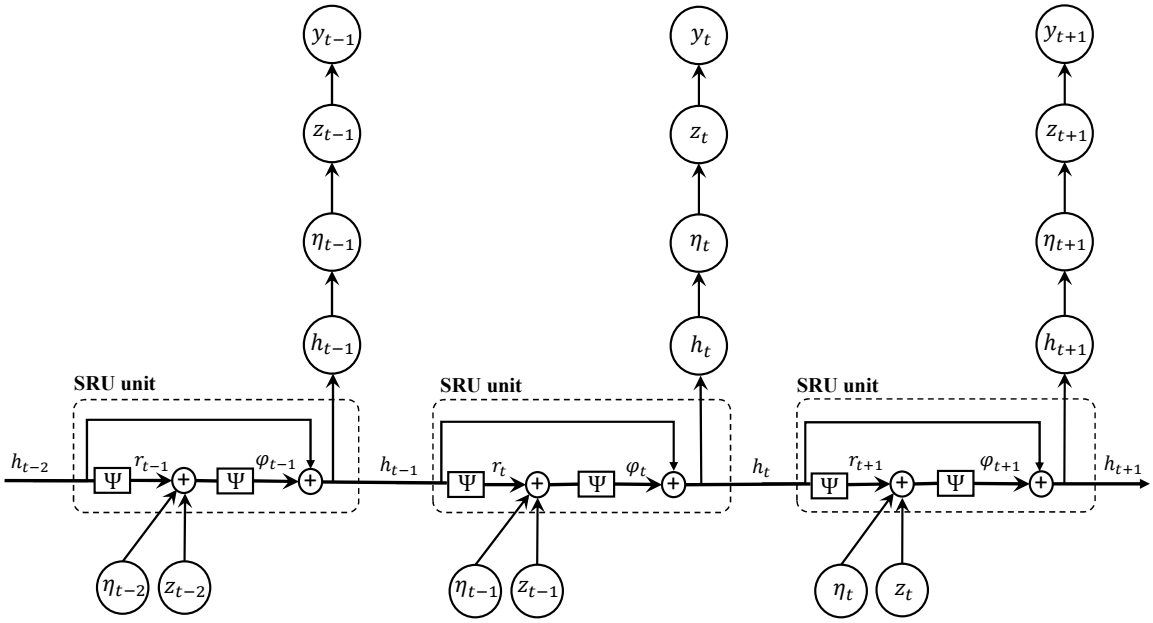


**Figure 5.1.** Graphical representation of the SR-SV model. The $\oplus$ symbol represents the addition operation.

We note the following important properties of the SR-SV model. First, the SR-SV model in (5.9)-(5.12) retains the measurement equation (5.12) and the linear part $\phi z_{t-1}$ of the AR(1) process from the standard SV model, and captures the volatility effects not captured by the AR(1) process, e.g. non-linear and long-memory auto-dependence, via the latent state $h_t$ of the SRU structure. The log volatility at time $t$ in (5.11) can be written as

$$z_t = \beta_0 + \beta_1 \text{SRU}(\eta_{t-1}, z_{t-1}, h_{t-1}) + \phi z_{t-1} + \epsilon_t^{\eta}. \tag{5.13}$$

Therefore, the parameter $\beta_1$ characterizes all the effects in the underlying log volatility process $z$ rather than the short-term linear effect captured by the AR(1) process. We refer to $\beta_1$ as the *non-linearity long-memory* coefficient. If $\beta_1 = 0$ and $\epsilon_1^{\eta} \sim \mathcal{N}(\beta_0/(1-\phi), \sigma^2/(1-\phi^2))$, the SR-SV model becomes the SV model (5.1)-(5.2) and hence the SV model is a special case of the SR-SV model. We therefore follow the SV literature and assume that $|\phi| < 1$. A sufficient condition for stationarity for the $z$ process, and thus

the $y$ process, is that$\beta_1 = 0$ and $\epsilon_1^{\eta} \sim \mathcal{N}(\beta_0/(1-\phi), \sigma^2/(1-\phi^2))$. Non-stationarity for volatility is often argued to be more realistic in practice (e.g. van Bellegem (2012)), although it might be mathematically less appealing. The equation in (5.13) can be further written out as

$$z_t = \beta_0 + \beta_1 \mathfrak{N}(\eta_{t-1}, w_z z_{t-1}, h_{t-1}) + \phi z_{t-1} + \epsilon_t^{\eta}, \tag{5.14}$$

where $\mathfrak{N}(\cdot)$ is a non-linear function and $w_z$ is the weight corresponding to $z_{t-1}$. If $w_z = 0$ in (5.14), then $z_t$ only depends linearly on $z_{t-1}$, therefore this equation indicates that the parameter $w_z$ characterizes the serial dependence rather than linearity that the previous log volatility $z_{t-1}$ has on $z_t$. We will analyse $w_z$ in more details in Section 5.4.

Second, Oliva et al. (2017) set the scales $\alpha$ of the SRU model to several pre-specified values to obtain a vector of summary statistics $h_t$ at different moving average weights. We, however, treat $\alpha$ as a model parameter and learn it from the data. We note that a higher $\alpha$ weighs more on the historical information while a smaller $\alpha$ puts a more weight on the current information. We show later in the empirical study that this parameter $\alpha$ is able to quantify the existence of the long-memory auto-dependence commonly exhibited in the volatility dynamics of the financial time series.

Third, neural networks are highly flexible but often subject to overfitting, i.e., they have over-confident in-sample fit and bad out-of-sample forecasts. Regularization is often needed to avoid overfitting. Injecting noise into the layers of the network has been found an effective regularization approach in the Machine Learning literature, and seen as a form of data augmentation at multiple levels of abstraction (Sietsma and Dow, 1991; Poole et al., 2014; Goodfellow et al., 2016; Dieng et al., 2018). In the SR-SV model, by allowing $z_{t-1}$ and $\eta_{t-1}$ to be the inputs of the SRU structure at time $t$, we inject the noise $\epsilon_{t-1}^{\eta}$ of the volatility process to the input and hidden layers of the SRU. This noise-injecting regularization approach makes the SR-SV model perform well on both in-sample fitting and out-of-sample forecast, even with the simplest specification of the SRU structure where all the $r_t$, $\varphi_t$ and $h_t$ are scalars. Our SR-SV model can be categorized as a *parametric* model with the vector of model parameters $\theta$ consisting of eleven parameters: four main parameters $\beta_0$, $\beta_1$, $\phi$, $\sigma^2$ and the parameters in the SRU including $\alpha$,$w_h$,$b_r$,$w_r$,$w_\eta$,$w_z$ and $b_\varphi$.

Finally, $\beta_0$ plays the role of the scale factor $\tau = e^{\beta_0/2}$ for the variance of $y_t$. One could set $\beta_0 = 0$ and modify (5.12) to $y_t = \tau e^{\frac{1}{2}z_t}\epsilon_t^y$; however, this parameterization might be less statistically efficient in terms of Bayesian estimation, especially for the parameter $\tau$ (see Kim et al., 1998).

It is straightforward to extend the SR-SV model in (5.9)-(5.12) by incorporating other advances in the SV literature. For example, we can use a Student's $t$ distribution instead of a Gaussian for the measurement shock $\epsilon_t^y$ and take into account the leverage effect by correlating $\epsilon_t^y$ with the volatility shock $\epsilon_t^\eta$. We do not consider these extensions here, however, because using the most basic version makes it easier to understand the strengths and weaknesses of the new model.

## 5.3 Bayesian inference

This section discusses Bayesian estimation and inference for the SR-SV model. For a generic sequence $\{x_t\}$ we use $x_{i:j}$ to denote the series $(x_i,...,x_j)$. The SR-SV model is a state-space model with the measurement equation

$$y_t|z_t \sim \mathcal{N}(0, e^{z_t}), \tag{5.15}$$

and the state transition equations

$$z_t|z_{1:t-1}, h_t \sim \mathcal{N}(\phi z_{t-1} + \beta_0 + \beta_1 h_t, \sigma^2), \quad t \geq 2, \quad z_1 \sim \mathcal{N}(\beta_0, \sigma^2). \tag{5.16}$$

with

$$h_t = \text{SRU}(x_t, h_{t-1}), \quad t = 2, ..., T, \text{ and } h_1 = 0, \; x_t = (\eta_{t-1}, z_{t-1})^T \tag{5.17}$$

$$\eta_t|h_t \sim \mathcal{N}(\beta_0 + \beta_1 h_t, \sigma^2) \tag{5.18}$$

We are interested in sampling from the posterior distribution of $\theta$

$$\pi(\theta) = p(\theta|y_{1:T}) = \frac{p(y_{1:T}|\theta)p(\theta)}{p(y_{1:T})}, \tag{5.19}$$

where $p(y_{1:T}|\theta)$ is the likelihood function, $p(\theta)$ is the prior and $p(y_{1:T}) = \int_\Theta p(y_{1:T}|\theta)p(\theta)d\theta$ is the marginal likelihood. Recall that the vector of model parameters $\theta$ consists of $\beta_0$, $\beta_1$, $\phi$, $\sigma^2$ and the 7 parameters within the SRU model (2.10a)-(2.10c).

The likelihood function in (5.19) is

$$p(y_{1:T}|\theta) = \int p(y_{1:T}|z_{1:T}, \theta)p(z_{1:T}|\theta)dz_{1:T}, \tag{5.20}$$

which is computationally intractable for non-linear non-Gaussian state space models like the SV and SR-SV models, but can be estimated unbiasedly by a particle filter (Del Moral, 2004). Bayesian inference for SR-SV can be performed using recent advances in the Sequential Monte Carlo literature that we present next.

### 5.3.1 The Density Tempered Sequential Monte Carlo for the SR-SV model

Duan and Fulop (2015) propose the Density Tempered Sequential Monte Carlo (DT-SMC) approach to Bayesian inference for state space models where the likelihood is intractable. The DT-SMC sampler generalizes the SMC method of Neal (2001) and Del Moral et al. (2006), discussed in Section 2.3.2, when the likelihood can be computed analytically. In order to sample from the posterior $\pi(\theta)$, the DT-SMC method first samples a set of $M$ weighted particles $\{W_0^j,\theta_0^j\}_{j=1}^M$ from an easy-to-sample distribution $\pi_0(\theta)$, such as the prior $p(\theta)$, and then traverses these particles through intermediate distributions $\pi_t(\theta)$, $t=1,...,K$, which target the posterior distribution $\pi(\theta)$ eventually, i.e. $\pi_K(\theta)=\pi(\theta)$. The DT-SMC method uses the following intermediate distributions

$$\pi_t(\theta) := \pi_t(\theta|y_{1:T}) \propto \widehat{p}(y_{1:T}|\theta,u)^{\gamma_t} p(\theta), \tag{5.21}$$

where the $\gamma_t$ is referred to as the level temperature and $0=\gamma_0<\gamma_1<\gamma_2<...<\gamma_K=1$, $\widehat{p}(y_{1:T}|\theta,u)$ is the unbiased estimator of the likelihood $p(y_{1:T}|\theta)$ and $u$ is the set of pseudo random numbers used within a particle filter to estimate the likelihood $p(y_{1:T}|\theta)$. For the purpose of this paper where it is possible to sample from the prior $p(\theta)$, we set $\pi_0(\theta)=p(\theta)$. Algorithm 5.1 summarizes the DT-SMC method for the SR-SV model.

Similar to the SMC algorithm in Section 2.3.2, the DT-SMC method consists of three main steps: reweighting, resampling and Markov move. At the begining of SMC iteration $t$, the set of weighted particles $\{W_{t-1}^j,\theta_{t-1}^j\}_{j=1}^M$ that approximate the intermediate distribution $\pi_{t-1}(\theta)$ is reweighted to approximate the target $\pi_t(\theta)$. The efficiency of these weighted particles is often measured by the effective sample size (ESS) defined in (5.24). If the ESS is below a prespecified threshold, the particles are resampled; the resulting equally-weighted resamples, which are now approximate samples from $\pi_t(\theta)$, are then refreshed by a Markov kernel whose invariant distribution is $\pi_t(\theta)$. For example, Duan and Fulop (2015) uses the pseudo marginal Metropolis-Hastings (PMMH) kernal of Andrieu et al. (2010) with the likelihood estimated unbiasedly by the particle filter in the Markov move step. However, Pitt et al. (2012) suggest that the PMMH approach works efficiently when the variance of the log of the estimated likelihood is around 1. For some state space models like the SR-SV model, a large number of particles might be required to obtain a likelihood estimator with log variance to be around 1, which is computationally

inefficient. To tackle this problem, we incorporate the Correlated Pseudo Marginal (CPM) approach of Deligiannidis et al. (2018) into the Markov move step. The CPM method makes the current set of random numbers $u$ and proposal $u'$ correlated, and helps reduce the variance of the ratio $\widehat{p}(y_{1:T}|\theta',u')/\widehat{p}(y_{1:T}|\theta,u)$ in (5.25), thus leading to a better mixing Markov chain while using less number of particles in the particle filter. Similar to the SMC methods of Del Moral et al. (2006) and Neal (2001), the DT-SMC method is parallelizable as the particles move independently in the Markov move step, and provides an estimate of the marginal likelihood as a by-product.

In Algorithm 5.1, we use a random walk proposal for $q(\theta'|\theta)$. We follow Gunawan et al. (2018) and choose the tempering sequence $\gamma_t$ adaptively to ensure a sufficient level of particle efficiency by selecting the next value of $\gamma_t$ such that ESS stays above a threshold.

## 5.4 Simulation studies and applications

This section evaluates the performance of the SR-SV model relative to the SV, N-SV and LMSV models using a simulation study and real data applications. We do not report the results for GARCH as it performs similarly to SV. We use the DT-SMC sampler for the Bayesian inference in the SV, N-SV and SR-SV models. As the LMSV model does not have an explicit state-space representation and its likelihood function is analytically intractable, we follow Breidt et al. (1998) and estimate the LMSV model on the frequency domain. The implementation of Bayesian inference for the LMSV model is presented in Appendix C.1. Table C.2 in Appendix C.2 lists our implementation details of the DT-SMC sampler. All the computations are run on a High-Performance Computing (HPC) machine[6] with 16-core CPU and 16 Gigabytes of RAM. The DT-SMC sampler was initialized by sampling from the priors in Table 5.1.

We now motivate the choice of the priors in Table 5.1. We follow Yu et al. (2006) and Kim et al. (1998) to set the same prior, which is a Beta distribution, for the persistence parameters $\phi$ of the three models SV, N-SV and SR-SV. We also use an inverse-Gamma prior for the parameters $\sigma^2$ in all models but make it more flat than the priors used in Yu et al. (2006) and Kim et al. (1998). We follow Yu et al. (2006) to use an informative but reasonably flat prior distribution for the intercept $\mu$ in the SV and N-SV models. For the SR-SV model, we found that the posterior distributions of $\beta_1$ and $w_z$ to be

[6]https://sydneyuni.atlassian.net/wiki/spaces/RC/overview

**Algorithm 5.1** The Density Tempered Sequential Monte Carlo for the SR-SV model

1. Sample $\theta_0^j \sim p(\theta)$, $u_0^j \sim p(u)$ and set $W_0^j = 1/M$ for $j = 1...M$
2. **For** $t = 1,...,K$,

**Step 1: Reweighting**: Compute the unnormalized weights

$$w_t^j = W_{t-1}^j \frac{\widehat{p}(y_{1:T}|\theta_{t-1}^j, u_{t-1}^j)^{\gamma_t} p(\theta_{t-1}^j)}{\widehat{p}(y_{1:T}|\theta_{t-1}^j, u_{t-1}^j)^{\gamma_{t-1}} p(\theta_{t-1}^j)} = W_{t-1}^j \widehat{p}(y_{1:T}|\theta_{t-1}^j, u_{t-1}^j)^{\gamma_t - \gamma_{t-1}},\ j = 1,...,M \tag{5.22}$$

and set the new normalized weights

$$W_t^j = \frac{w_t^j}{\sum_{s=1}^M w_t^s},\quad j = 1,...,M. \tag{5.23}$$

**Step 2:** Compute the effective sample size (ESS):

$$\text{ESS} = \frac{1}{\sum_{j=1}^M \left(W_t^j\right)^2}. \tag{5.24}$$

**if** ESS $< cM$ for some $0 < c < 1$, **then**

(i) **Resampling**: Resampling from $\{\theta_{t-1}^j, u_{t-1}^j\}_{j=1}^M$ using the weights $\{W_t^j\}_{j=1}^M$, and then set $W_t^j = 1/M$ for $j = 1...M$, to obtain the new equally-weighted particles $\{\theta_t^j, u_t^j, W_t^j\}_{j=1}^M$.

(ii) **Markov move**: For each $j = 1,...,M$, move the samples $\theta_t^j$, $u_t^j$ according to $N_{\text{CPM}}$ CPM steps:

(a) Sample $\theta_t^{j\prime}$ from the proposal density $q(\theta_t^{j\prime}|\theta_t^j)$.

(b) Sample $\epsilon^j \sim \mathcal{N}(0_D, I_D)$ and set $u_t^{j\prime} = \rho u_t^j + \sqrt{1-\rho^2}\epsilon^j$ with $\rho \in (-1,1)$ is a correlation factor.

(c) Compute the estimated likelihood $\widehat{p}(y_{1:T}|\theta_t^{j\prime}, u_t^{j\prime})$ using a particle filter (see Algorithm C.2 in Appendix C.2)

(d) Set $\theta_t^j = \theta_t^{j\prime}$ and $u_t^j = u_t^{j\prime}$ with the probability

$$\min\left(1, \frac{\widehat{p}(y_{1:T}|\theta_t^{j\prime}, u_t^{j\prime})^{\gamma_t} p(\theta_t^{j\prime})}{\widehat{p}(y_{1:T}|\theta_t^j, u_t^j)^{\gamma_t} p(\theta_t^j)} \frac{q(\theta_t^j|\theta_t^{j\prime})}{q(\theta_t^{j\prime}|\theta_t^j)}\right), \tag{5.25}$$

otherwise keep $\theta_t^j$, $u_t^j$ unchanged.

**end**

| SR-SV | | SV | | N-SV | |
|---|---|---|---|---|---|
| Parameter | Prior | Parameter | Prior | Parameter | Prior |
| $\beta_0$ | $\mathcal{N}(0,0.1)$ | $\mu$ | $\mathcal{N}(0,25)$ | $\mu$ | $\mathcal{N}(0,25)$ |
| $\frac{\phi+1}{2}$ | Beta(20,1.5) | $\frac{\phi+1}{2}$ | Beta(20,1.5) | $\frac{\phi+1}{2}$ | Beta(20,1.5) |
| $\sigma^2$ | $IG(2.5,0.25)$ | $\sigma^2$ | $IG(2.5,0.25)$ | $\sigma^2$ | $IG(2.5,0.25)$ |
| $\beta_1$ | $IG(2.5,1)$ | | | $\delta$ | $\mathcal{N}(0,0.1)$ |
| $\alpha$ | Beta(2,2) | | | | |
| $w_h$,$w_\phi$,$w_\eta$ | $\mathcal{N}(0,0.1)$ | | | | |
| $b_r$,$b_\phi$ | $\mathcal{N}(0,0.1)$ | | | | |
| $w_z$ | $IG(2.5,1)$ | | | | |

**Table 5.1.** Prior distributions for the parameters in the SR-SV, SV and N-SV models. The notation $\mathcal{N}$, $IG$ and Beta denote the Gaussian, inverse-Gamma and Beta distributions, respectively.

unimodal under inverse-Gamma priors. We use a normal prior with a zero mean and a small variance for the SRU parameters, except $w_z$, because empirical results from the SRU literature show that the values of the SRU parameters are often small. Finally, we set a normal prior with a zero mean and a small variance for the intercept $\beta_0$ in the SR-SV model as the empirical results often show small values of $\beta_0$.

Table 5.2 lists the predictive scores used to measure the out-of-sample performance. The smaller the predictive scores, the better.

| **Score** | **Definition** | **Score** | **Definition** |
|---|---|---|---|
| PPS | $-T_{test}^{-1}\sum_{D_{test}}\log p(y_t\|y_{1:t-1},\widehat{\theta})$ | $\text{MSE}_1$ | $T_{test}^{-1}\sum_{D_{test}}(\sigma_t-\widehat{\sigma}_t)^2$ |
| QLIKE | $T_{test}^{-1}\sum_{D_{test}}\left(\log(\widehat{\sigma}_t^2)+\sigma_t^2\widehat{\sigma}_t^{-2}\right)$ | $\text{MSE}_2$ | $T_{test}^{-1}\sum_{D_{test}}(\sigma_t^2-\widehat{\sigma}_t^2)^2$ |
| $\text{R}^2\text{LOG}$ | $T_{test}^{-1}\sum_{D_{test}}\left[\log(\sigma_t^2\widehat{\sigma}_t^{-2})\right]^2$ | $\text{MAE}_1$ | $T_{test}^{-1}\sum_{D_{test}}\lvert\sigma_t-\widehat{\sigma}_t\rvert$ |
| | | $\text{MAE}_2$ | $T_{test}^{-1}\sum_{D_{test}}\lvert\sigma_t^2-\widehat{\sigma}_t^2\rvert$ |

**Table 5.2.** Definition of the predictive scores to measure the out-of-sample performance on simulation and real index data. Here, $\widehat{\sigma}_t$ is an estimate of the volatility $\sigma_t$, $T_{test}$ is the number of observations in test data $D_{test}$ and $\widehat{\theta}$ is a posterior mean estimate of $\theta$.

### 5.4.1 Simulation studies

We consider three volatility models specified in Table 5.3. Model 1 is a GARCH(1,1). Model 2 is an extension of the GARCH(1,1) model by applying a Box-Cox power transformation (Box and Cox, 1964) to both the conditional variance equation and the volatility dynamics. Model 2 is similar to the non-linear ARCH models of Higgins and Bera (1992) but they use lagged innovations to construct the conditional variance. Model 3 is a FIGARCH(1,$d$,1) model of Baillie et al. (1996) which uses a long-memory process AFRIMA(1,$d$,1) to simulate the long-memory auto-dependence.

| Data | Model | Parameters |
|---|---|---|
| SIM I | $\sigma_t^2 = \mu + \alpha y_{t-1}^2 + \beta\sigma_{t-1}^2,\ t=2,...,T$<br>$y_t = \sigma_t\epsilon_t,\ \epsilon_t \sim \mathcal{N}(0,1),\ t=1,...,T$ | $\sigma_1^2 = 0.1,\ \mu = 0.1$<br>$\alpha = 0.07,\ \beta = 0.92$ |
| SIM II | $h_t = \mu + \alpha\dfrac{(y_{t-1}^2)^\delta - 1}{\delta} + \beta h_{t-1},\ t=2,...,T$<br>$y_t = \left(1+\delta h_t\right)^{1/2\delta}\epsilon_t,\ \epsilon_t \sim \mathcal{N}(0,1),\ t=1,...,T$ | $h_1 = 0.1,\ \mu = 0.1$<br>$\alpha = 0.15,\ \beta = 0.82$<br>$\delta = 0.9$ |
| SIM III | $\sigma_t^2 = \mu + \left[1-\beta B-(1-\phi B)(1-B)^d\right]y_t^2 + \beta\sigma_{t-1}^2,\ t=2,...,T$<br>$y_t = \sigma_t\epsilon_t,\ \epsilon_t \sim \mathcal{N}(0,1),\ t=1,...,T$ | $\sigma_1^2 = 0.1,\ \mu = 0.01$<br>$\phi = 0.01,\ \beta = 0.5$<br>$d = 0.62$ |

**Table 5.3.** Simulation: Data generating process.

We generate time series of $T = 3000$ observations from these three models and name the simulation datasets as SIM I, SIM II, SIM III, accordingly. The parameters are set so that $y_t$ somewhat resembles real financial time series data exhibiting volatility clustering with non-linearity (SIM II) and long-memory (SIM III) auto-dependence in the underlying volatility dynamics. For each dataset, the first $T_{in} = 2000$ observations are used for model estimation and the last $T_{out} = 1000$ are for out-of-sample analysis. Table 5.4 shows the posterior mean estimates for the parameters of the SV and SR-SV models, with the posterior standard deviations in brackets; for the SR-SV model we only show the results for the main parameters. The last column in the table shows the marginal likelihood estimates, averaged over 10 different runs of the DT-SMC sampler, together with the Monte Carlo standard errors in the brackets. Figures 5.3, 5.4 and 5.5

plot the filtered values of the $\eta_t$ and $h_t$ components of the SR-SV model in SIM I, SIM II and SIM III datasets, respectively. Figure 5.2 in Appendix C.3 plots the true volatility together with the filtered volatility produced by the SV and SR-SV models, for all the three simulation datasets.

| | $\mu$ | $\phi$ | $\sigma^2$ | $\alpha$ | $\beta_0$ | $\beta_1$ | $w_z$ | Mar.llh |
|---|---|---|---|---|---|---|---|---|
| **SIM I** | | | | | | | | |
| SV | **2.145** | **0.985** | **0.019** | | | | | $-5100.8$ |
| | (0.237) | (0.004) | (0.003) | | | | | (0.131) |
| SR-SV | | **0.974** | **0.020** | **0.534** | **0.027** | **0.388** | $\mathbf{-0.205}$ | $-5099.9$ |
| | | (0.023) | (0.005) | (0.166) | (0.031) | (0.235) | (0.261) | (0.300) |
| **SIM II** | | | | | | | | |
| SV | **1.050** | **0.967** | **0.032** | | | | | $-4060.3$ |
| | (0.125) | (0.008) | (0.006) | | | | | (0.164) |
| SR-SV | | **0.792** | **0.041** | **0.515** | **0.043** | **0.423** | **0.530** | $-4057.7^*$ |
| | | (0.106) | (0.010) | (0.156) | (0.044) | (0.207) | (0.256) | (0.306) |
| **SIM III** | | | | | | | | |
| SV | **0.134** | **0.984** | **0.041** | | | | | $-3146.9$ |
| | (0.329) | (0.005) | (0.007) | | | | | (0.195) |
| SR-SV | | **0.896** | **0.056** | **0.645** | $\mathbf{-0.093}$ | **0.325** | **0.290** | $-3144.2^*$ |
| | | (0.035) | (0.013) | (0.240) | (0.045) | (0.132) | (0.115) | (0.316) |

**Table 5.4.** Simulation: Posterior means of the parameters with the posterior standard deviations in brackets. The last column shows the estimated log marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the DT-SMC sampler. The asterisks indicate the cases when the Bayes factors strongly support the SR-SV model over the SV model. The marginal likelihood are reported in natural log scale.

The estimation results suggest the following conclusions. First, for the SIM I data, the difference of marginal likelihood estimates in Table 5.4 between the SV and SR-SV models is insignificant, the coefficient $\beta_1$ is insignificant, and the filtered volatilities from these two models in Figure 5.2 are identical and close to the true volatility. This implies that the SV and SR-SV models fit equally well to the SIM I data and that the SR-SV model is close to the SV model if the true data generating process, which is GARCH(1,1) in this example, exhibits no other effects rather than short-memory linear auto-dependence within the volatility dynamics.

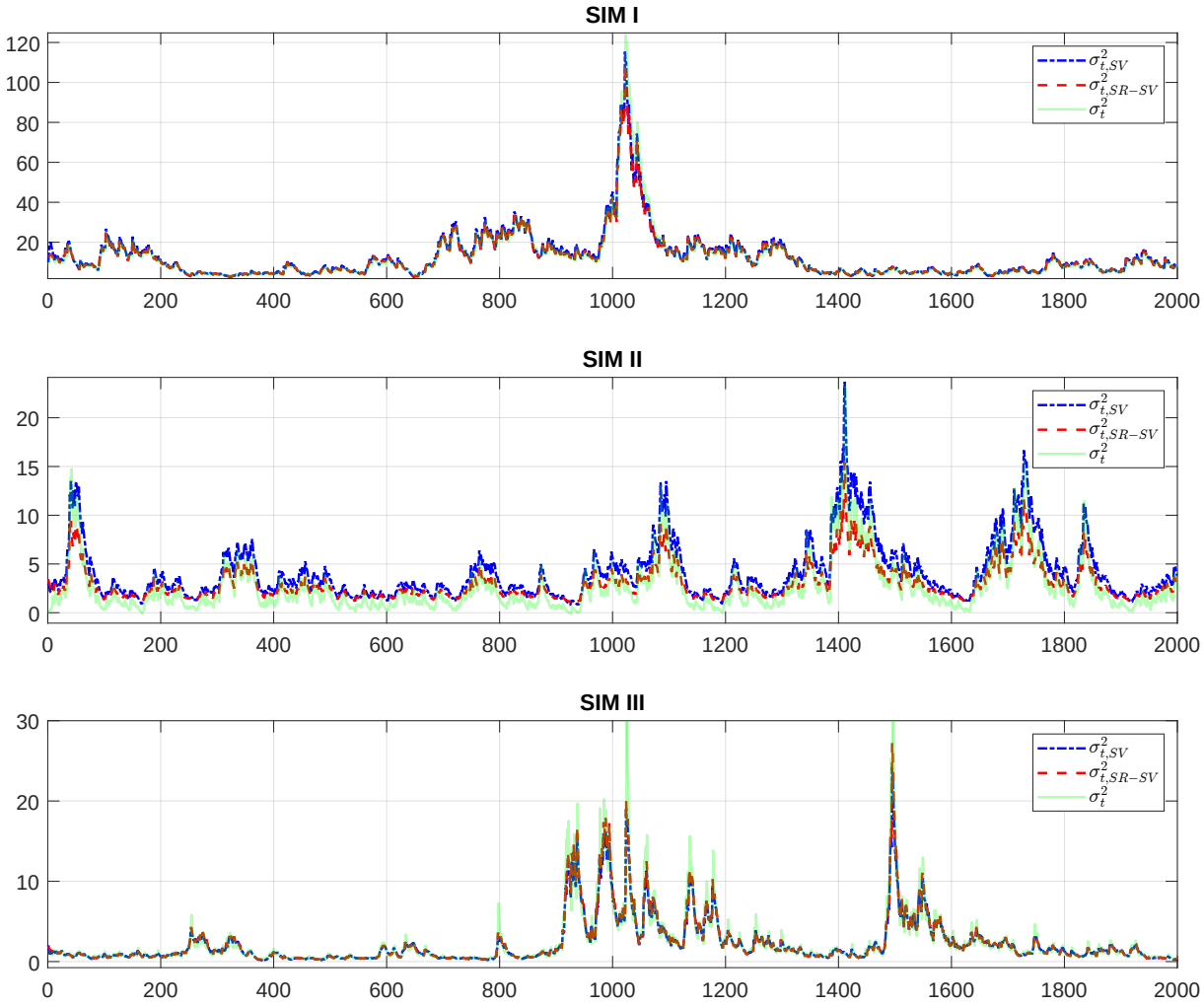


**Figure 5.2.** Simulation: Filtered volatility of the SV and SR-SV models, together with the true volatility, on three simulation datasets. (This is better viewed in colour).

Second, the estimation results on the SIM II and SIM III data show that the additional neural network structure of the SR-SV model is able to efficiently capture the volatility effects overlooked by the basic SV model. This is supported by the Bayes factors of the SR-SV model compared to the SV model of more than $e^{2.3}$, which, according to the interpretation in Table 2.2, strongly support the SR-SV model. The plots of the $h_t$ and $\eta_t$ components of the SR-SV model in Figures 5.3, 5.4 and 5.5 clearly show that $h_t$, and hence $\eta_t$, is well responsive to volatility effects rather than the linear short-memory effects. For example, in the SIM I data when the volatility exhibits no non-linear and long-memory effects, the $h_t$ shown in Figure 5.3 is significantly small at all time steps and hence the $\eta_t$ component simply fluctuates around $\beta_0$ during both low and high volatility periods. Figures 5.4 and 5.5, on the other hand, show that the $h_t$ response adaptively to the changes in the volatility dynamics. As the result, $\eta_t$ is small during

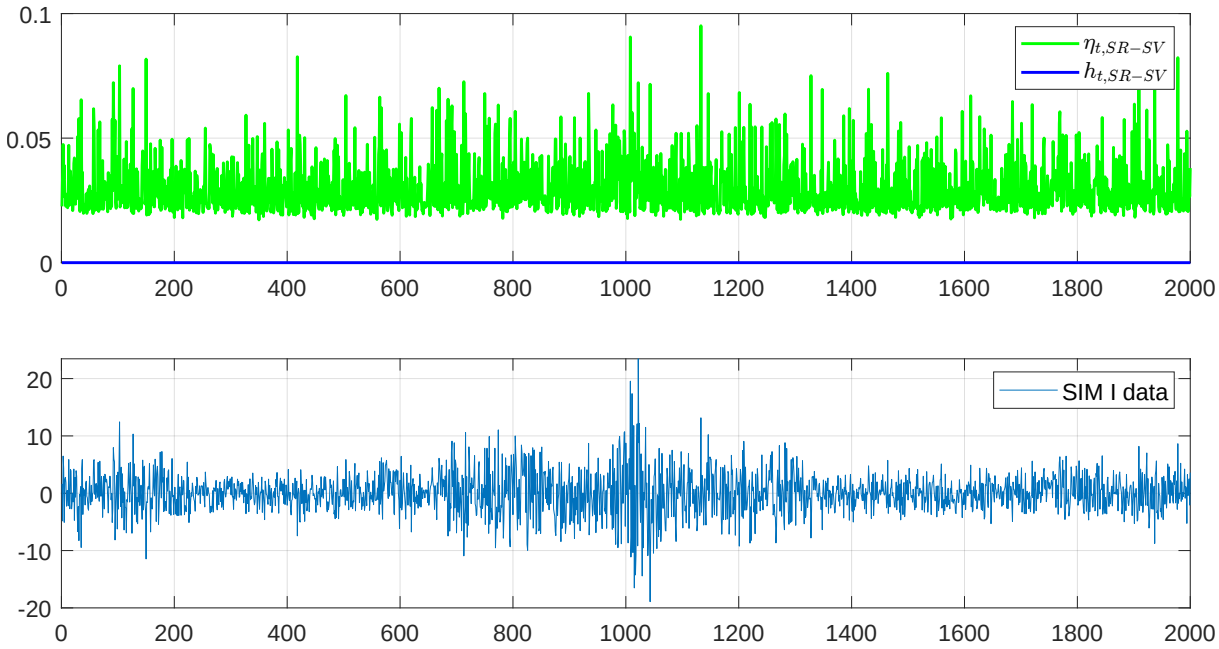


**Figure 5.3.** SIM I: The filtered values of $\eta_t$ and $h_t$ of the SR-SV model, together with the SIM I in-sample data. (This is better viewed in colour).

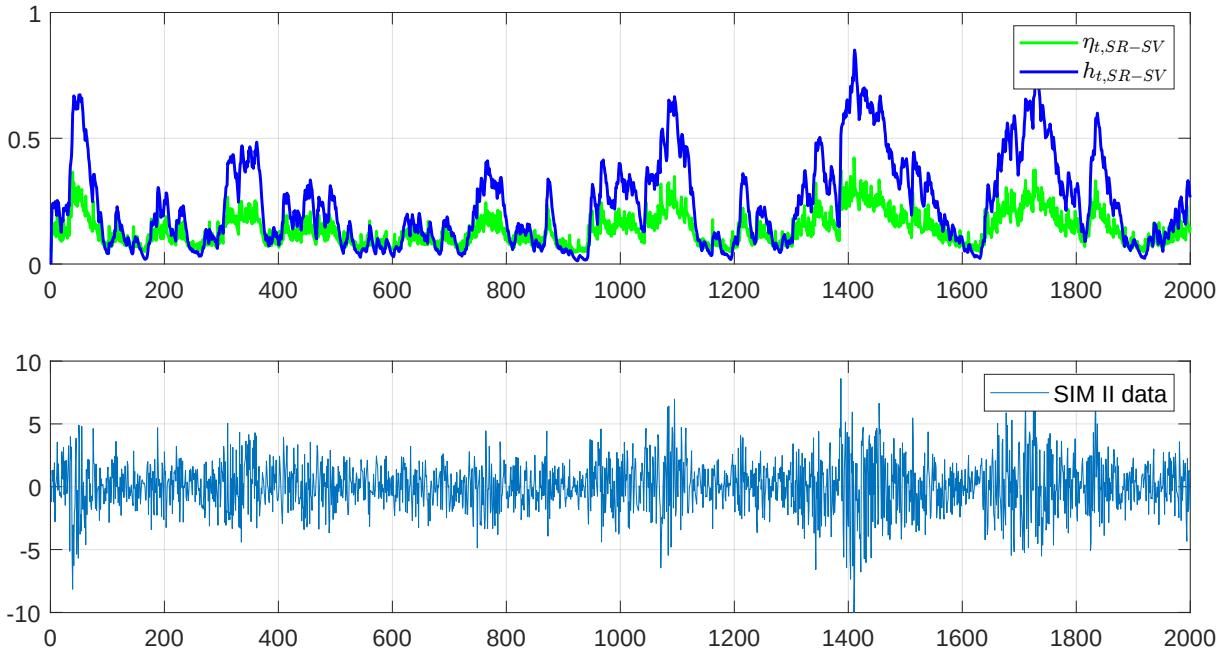


**Figure 5.4.** SIM II: The filtered values of $\eta_t$ and $h_t$ of the SR-SV model, together with the SIM II in-sample data. (This is better viewed in colour).

the low volatility periods and large in the high volatility periods. The non-linear (SIM II) and long-memory (SIM III) auto-dependence of the simulated volatility are well captured by the SRU structure of the SR-SV model. The plots of filtered volatility in

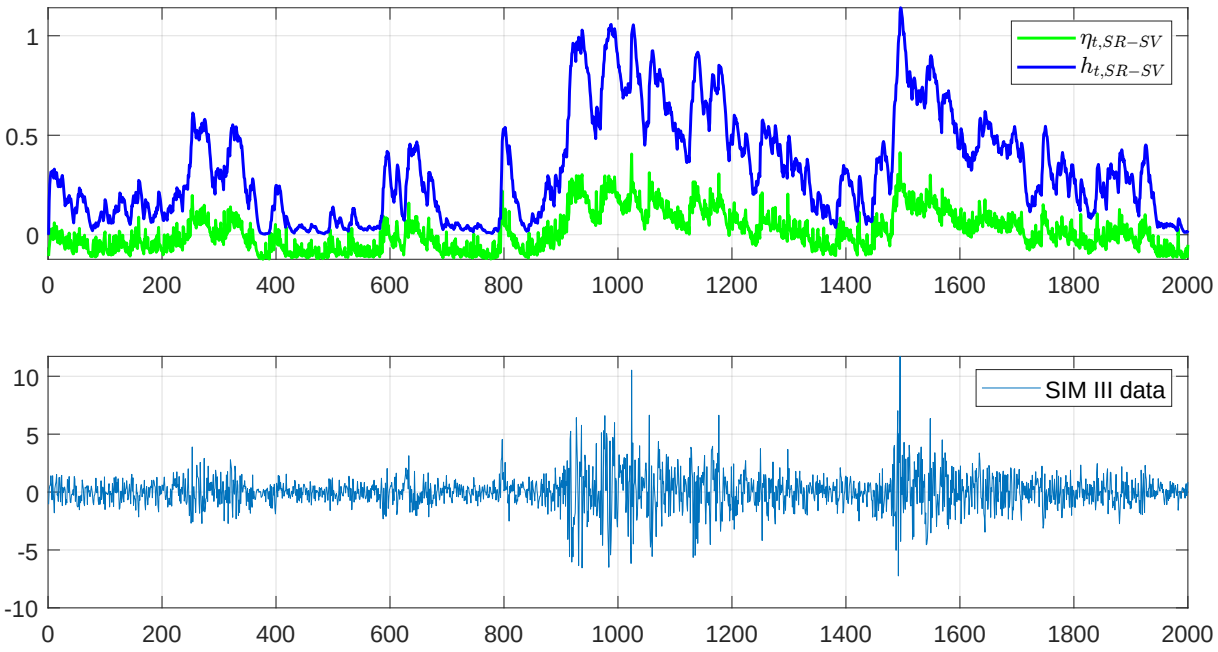


**Figure 5.5.** SIM III: The filtered values of $\eta_t$ and $h_t$ of the SR-SV model, together with the SIM III in-sample data. (This is better viewed in colour).

Figure 5.2 show that the filtered volatility of the SR-SV model are generally closer to the true volatility than those of the SV model.

Third, the parameters of the SR-SV model are able to characterize well the existence of the various volatility effects in these simulation data. The estimated posterior means of parameter $\beta_1$ are more than two standard deviations from zero in the SIM II and SIM III data, suggesting the existence of volatility effects rather than linearity in the volatility dynamics of these two datasets; while $\beta_1$ is less than two standard deviations from zero in the SIM I data, suggesting that only simple linear effects are detectable in the volatility dynamics of this dataset. Similarly, the non-linear coefficient $w_z$ of the SR-SV model (c.f. equation (5.14)) is more than two standard deviations from zero in the SIM II and SIM III data but not in the SIM I data, indicating that the SR-SV model is able to detect the serial dependence rather than the linear dependence that the past log volatility $z_{t-1}$ has on the current log volatility $z_t$. The estimated posterior mean of the moving average weight parameter $\alpha$ in the SIM III data is higher than those in the SIM I and SIM II data, supporting further the evidence of the long-memory auto-dependence exhibited in the volatility dynamic of the SIM III data, which is generated from a FIGARCH(1,$d$,1) model. Figure 5.6 shows that the posterior mode of $\alpha$ in the SIM III data is much closer to 1 than those in the SIM I and SIM II data. Finally, it is worth noting that the persistence parameter $\phi$ of the SR-SV is close to the

persistence parameter $\phi$ of the SV model in the SIM I data but much smaller than that of the SV model in the SIM II and SIM III model. This is probably because the non-linear coefficient $w_z$ with respect to the past log volatility $z_{t-1}$ is significant in the SIM II and SIM III data, and hence the historical information has been also well stored in the $\eta_t$ process.

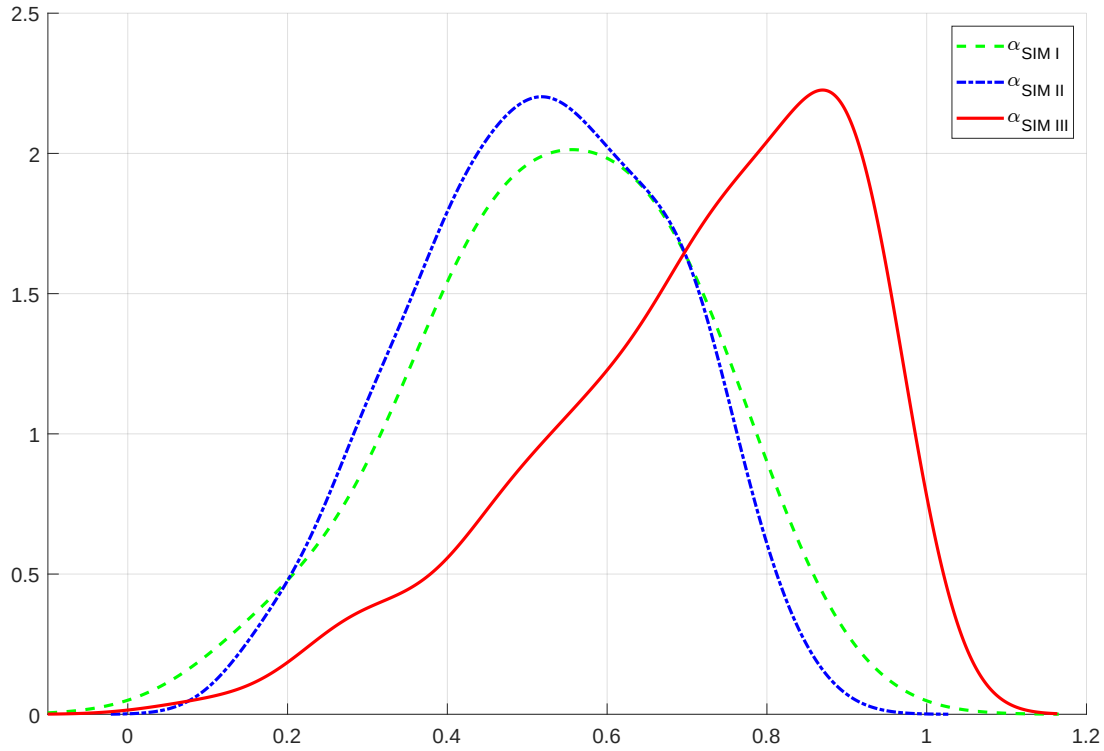


**Figure 5.6.** Simulation: Posterior densities of the moving avarage weight $\alpha$ on the three simulation datasets.

Table 5.5 reports the predictive performance scores of the SV and SR-SV models with the Monte Carlo standard errors in brackets. For the SIM II and SIM III data, the SR-SV model outperforms the SV model for all the predictive scores, which is consistent with the in-sample analysis showing that the SR-SV model fits these simulation datasets better than the SV model. For the SIM I data, the SR-SV model also outperforms the SV model for all scores except the PPS score. These results illustrate the impressive out-of-sample forecast ability of the SR-SV model. The results for the real data applications in the next section further support this claim.

| | PPS | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLIKE | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|
| **SIM I** | | | | | | | | |
| SV | **2.355** | 0.480 | 0.485 | 0.506 | 0.540 | 0.481 | 0.482 | 1 |
| | (0.001) | (0.003) | (0.003) | (0.002) | (0.004) | (0.001) | (0.004) | |
| SR-SV | 2.357 | **0.435** | **0.381** | **0.342** | **0.319** | **0.450** | **0.395** | 6 |
| | (0.000) | (0.002) | (0.003) | (0.001) | (0.002) | (0.001) | (0.003) | |
| **SIM II** | | | | | | | | |
| SV | 1.881 | 0.189 | 0.282 | 0.309 | 0.383 | 0.359 | 0.512 | 0 |
| | (0.000) | (0.000) | (0.001) | (0.001) | (0.001) | (0.001) | (0.001) | |
| SR-SV | **1.878** | **0.076** | **0.122** | **0.139** | **0.189** | **0.172** | **0.284** | 7 |
| | (0.000) | (0.002) | (0.003) | (0.002) | (0.000) | (0.003) | (0.003) | |
| **SIM III** | | | | | | | | |
| SV | 1.722 | 0.919 | 0.967 | 0.707 | 0.796 | 0.754 | 0.529 | 0 |
| | (0.001) | (0.004) | (0.004) | (0.003) | (0.003) | (0.004) | (0.004) | |
| SR-SV | **1.720** | **0.733** | **0.775** | **0.548** | **0.625** | **0.588** | **0.399** | 7 |
| | (0.000) | (0.004) | (0.003) | (0.002) | (0.003) | (0.003) | (0.002) | |

**Table 5.5.** Simulation: Forecast performance of the SR-SV and SV models. In each panel, the bold numbers indicate the best predictive scores and the count indicates the number of times a model has better forecast scores than the other one. Monte Carlo standard errors in brackets, averaged over 10 different runs.

### 5.4.2 Applications

This section evaluates the SR-SV model using five popular daily stock indexes from different international markets: The German stock index DAX30 (DAX), the Hong Kong stock index HS50 (HSI), the France market index CAC40 (FCHI), the US stock market index SP500 (SPX) and the Canada market index TSX250 (TSX).

#### The datasets and exploratory data analysis

The datasets were downloaded from the Realized Library of The Oxford-Man Institute[7]. We used the adjusted closing prices $\{P_t,\ t = 1,...,T_P\}$ and calculated the

[7] https://realized.oxford-man.ox.ac.uk/

demeaned return process as

$$y_t = 100\left(\log\frac{P_{t+1}}{P_t} - \frac{1}{T_P-1}\sum_{i=1}^{T_P-1}\log\frac{P_{i+1}}{P_i}\right), \quad t = 1,2,...,T_P-1, \tag{5.26}$$

and using the first $T_{\text{in}}=2000$ returns for in-sample analysis and the rest $T_{\text{out}}=1000$ for out-of-sample analysis. Table 5.6 describes the relevant aspects of the datasets.

| | **In-sample Period** | **Out-of-sample Period** | $T_{\text{in}}$ | $T_{\text{out}}$ |
|---|---|---|---|---|
| DAX | 23 Apr 2004 – 21 Feb 2012 | 22 Feb 2012 – 05 Feb 2016 | 2000 | 1000 |
| HSI | 27 Oct 2003 – 28 Nov 2011 | 29 Nov 2011 – 21 Dec 2015 | 2000 | 1000 |
| FCHI | 09 Jun 2004 – 22 Mar 2012 | 23 Mar 2012 – 23 Feb 2016 | 2000 | 1000 |
| SPX | 27 Feb 2004 – 06 Feb 2012 | 07 Feb 2012 – 28 Jan 2016 | 2000 | 1000 |
| TSX | 03 Feb 2004 – 01 Feb 2012 | 02 Feb 2012 – 27 Jan 2016 | 2000 | 1000 |

**Table 5.6.** Descriptions of the five index datasets.

| | Min | Max | Std | Skew | Kurtosis | $V_n(10)$ | $V_n(20)$ | $V_n(30)$ |
|---|---|---|---|---|---|---|---|---|
| DAX | −7.437 | 9.993 | 1.267 | 0.115 | 10.960 | 3.226* | 2.501* | 2.146* |
| | | | | | | 2.456* | 1.926* | 1.670 |
| HSI | −11.616 | 12.155 | 1.186 | 0.307 | 17.551 | 3.934* | 3.030* | 2.587* |
| | | | | | | 2.564* | 2.088* | 1.844* |
| FCHI | −7.215 | 6.663 | 1.132 | −0.320 | 7.383 | 3.782* | 2.976* | 2.575* |
| | | | | | | 3.018* | 2.453* | 2.165* |
| SPX | −9.351 | 10.220 | 1.307 | −0.256 | 12.502 | 3.188* | 2.412* | 2.047* |
| | | | | | | 2.664* | 2.040* | 1.748* |
| TSX | −9.879 | 9.194 | 1.262 | −0.727 | 12.202 | 3.558* | 2.692* | 2.277* |
| | | | | | | 2.877* | 2.199* | 1.875* |

**Table 5.7.** Descriptive statistics for the demeaned returns of the DAX, HSI, FCHI, SPX and TSX datasets. $V_n(q)$, $q$=10, 20 and 30, shows the test statistics of Lo's modified R/S test of long memory with lag $q$. Upper and lower values of the 3 last columns are the Lo's test statistics for absolute and squared returns, respectively. The asterisks indicate significance at the 5% level.

Figure C.1 in Appendix C.3 plots the time series data and shows the existence of the volatility clustering effect commonly seen in financial data. Table 5.7 reports some

descriptive statistics together with Lo's modified R/S test (Lo, 1991) for long-range memory in the absolute and squared returns. Lo's modified R/S test is widely used in the financial time series literature; see, e.g., Lo (1991), Giraitis et al. (2003), Breidt et al. (1998). All the index data exhibit some negative skewness, high excess kurtosis and high variation. The result of Lo's modified R/S test for long-memory dependence with several different lags $q$ indicates that there is significant evidence of long-memory dependence in the stock indices.

We use the following four common realized measures including Realized Variance (RV), Bipower Variation (BV), Median Realized Volatility (MedRV), Realized Kernel Variance with the Non-Flat Parzen kernal (RKV) to evaluate the forecast performance of the volatility models using the predictive scores in Table 5.2. We also scale to realized measures by a scale as shown in 4.12 to compensate for the variation of the prices overnight.

**In-sample analysis**

Tables 5.8 and 5.9 summarizes the estimation results of fitting the SV, N-SV and SR-SV models to the five datasets. Table C.1 in Appendix C.3 shows the estimation results for the LMSV model. For the SR-SV model, we only show the results of the key parameters. Figure 5.7 shows the posterior densities of the moving average weight parameter $\alpha$ of the SR-SV model in all simulation and real datasets. We draw some conclusions from Tables 5.8 and 5.9 and the listed figures.

First, the marginal likelihood estimates show that the SR-SV model fits the five index data better than the SV and N-SV models. The Bayes factors of the SR-SV model compared to the SV and N-SV models are more than $e^{2.3}$, which strongly support the SR-SV in all cases. There are no significant differences between the SV and N-SV models in terms of marginal likelihood estimates across the five panels.

Second, the evidence of the volatility effects rather than linearity, e.g. non-linearity and long-memory auto-dependence, in the volatility dynamics of the index datasets is clear as the posterior means of the non-linearity long-memory parameter $\beta_1$ of the SR-SV model are more than two standard deviations from zero in all cases. The estimation results of the LMSV in Table C.1 show that the posterior means of the fractional integration parameter $d$ are also more than two standard deviations from zero and close to 0.5 in all cases, suggesting a strong evidence of the long-memory

| | $\mu$ | $\phi$ | $\sigma^2$ | $\delta$ | $\alpha$ | $\beta_0$ | $\beta_1$ | $w_z$ | Mar.llh |
|---|---|---|---|---|---|---|---|---|---|
| **DAX** | | | | | | | | | |
| SV | **−0.098** | **0.979** | **0.038** | | | | | | −2871.3 |
| | (0.233) | (0.006) | (0.008) | | | | | | (0.171) |
| N-SV | **−0.138** | **0.977** | **0.037** | **−0.198** | | | | | −2872.4 |
| | (0.212) | (0.006) | (0.008) | (0.086) | | | | | (0.224) |
| SR-SV | | **0.863** | **0.064** | | **0.605** | **−0.117** | **0.410** | **0.397** | −2868.8* |
| | | (0.052) | (0.021) | | (0.204) | (0.061) | (0.201) | (0.153) | (0.301) |
| **HSI** | | | | | | | | | |
| SV | **−0.205** | **0.987** | **0.022** | | | | | | −2692.0 |
| | (0.320) | (0.004) | (0.008) | | | | | | (0.184) |
| N-SV | **−0.366** | **0.987** | **0.021** | **−0.242** | | | | | −2691.0 |
| | (0.270) | (0.004) | (0.004) | (0.081) | | | | | (0.214) |
| SR-SV | | **0.824** | **0.054** | | **0.784** | **−0.196** | **0.536** | **0.387** | −2687.8** |
| | | (0.061) | (0.021) | | (0.137) | (0.083) | (0.262) | (0.139) | (0.337) |
| **FCHI** | | | | | | | | | |
| SV | **−0.213** | **0.977** | **0.047** | | | | | | −2787.1 |
| | (0.230) | (0.007) | (0.010) | | | | | | (0.225) |
| N-SV | **−0.217** | **0.979** | **0.041** | **−0.198** | | | | | −2787.3 |
| | (0.257) | (0.006) | (0.009) | (0.089) | | | | | (0.234) |
| SR-SV | | **0.843** | **0.093** | | **0.780** | **−0.179** | **0.449** | **0.363** | −2784.2* |
| | | (0.049) | (0.027) | | (0.197) | (0.070) | (0.199) | (0.134) | (0.326) |

**Table 5.8.** Applications: Posterior means of the parameters with the posterior standard deviations in brackets. The last column shows the estimated log marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the DT-SMC sampler. The single and double asterisks indicate the cases when the Bayes factors strongly and very strongly support the SR-SV model over the SV model, respectively. The marginal likelihood are reported in natural log scale.

dependence in the volatility process of these five index datasets. The posterior means of the non-linear parameter $w_z$ with respect to the past log volatility $z_{t-1}$ are more than two standard deviations from zero, indicating the existence of the serial dependence rather than linearity that the past log volatility $z_{t-1}$ has on the current log volatility $z_t$, and that the SR-SV model is able to detect this serial dependence. The posterior

| | $\mu$ | $\phi$ | $\sigma^2$ | $\delta$ | $\alpha$ | $\beta_0$ | $\beta_1$ | $w_z$ | Mar.llh |
|---|---|---|---|---|---|---|---|---|---|
| **SPX** | | | | | | | | | |
| SV | **−0.228** | **0.985** | **0.034** | | | | | | −2748.3 |
| | (0.344) | (0.005) | (0.006) | | | | | | (0.201) |
| N-SV | **−0.267** | **0.9837** | **0.036** | **−0.121** | | | | | −2749.4 |
| | (0.268) | (0.004) | (0.007) | (0.080) | | | | | (0.211) |
| SR-SV | | **0.844** | **0.056** | | **0.527** | **−0.180** | **0.481** | **0.373** | −2745.6* |
| | | (0.060) | (0.017) | | (0.186) | (0.186) | (0.241) | (0.132) | (0.311) |
| **TSX** | | | | | | | | | |
| SV | **−0.200** | **0.985** | **0.028** | | | | | | −2770.1 |
| | (0.323) | (0.004) | (0.006) | | | | | | (0.231) |
| N-SV | **−0.249** | **0.984** | **0.029** | **-0.141** | | | | | −2769.9 |
| | (0.298) | (0.005) | (0.006) | (0.077) | | | | | (0.245) |
| SR-SV | | **0.868** | **0.051** | | **0.697** | **−0.129** | **0.414** | **0.355** | −2767.2* |
| | | (0.056) | (0.015) | | (0.195) | (0.071) | (0.201) | (0.141) | (0.347) |

**Table 5.9.** Applications: Posterior means of the parameters with the posterior standard deviations in brackets. The last column shows the estimated log marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the DT-SMC sampler. The single and double asterisks indicate the cases when the Bayes factors strongly and very strongly support the SR-SV model over the SV model, respectively. The marginal likelihood are reported in natural log scale.

density plots of parameter $\alpha$ in Figure 5.7 suggest the existence of the long-memory auto-dependence in the volatility processes of the index datasets as the posterior densities of $\alpha$ are highly skewed for all cases and the posterior modes are close to 1, which is similar to the results in the SIM III data.

Third, it is worth noting that, in all cases, the persistence parameter $\phi$ in the SR-SV model is smaller than the persistence parameters in the SV and N-SV models as the parameter $w_z$ is significant and hence the linear effect that $z_{t-1}$ has on $z_t$ is reduced. As illustrated in the SIM I data, if the volatility process exhibits no volatility effects rather than a short-memory linear auto-dependence, the persistence parameters $\phi$ in the SV and SR-SV model are similar and the parameter $w_z$ is insignificant.

Using the posterior mean estimates in Table 5.8, the *filtered* values of $z_t$ of the SR-SV, SV and N-SV models can be computed using the particle filter. We discuss how to obtain

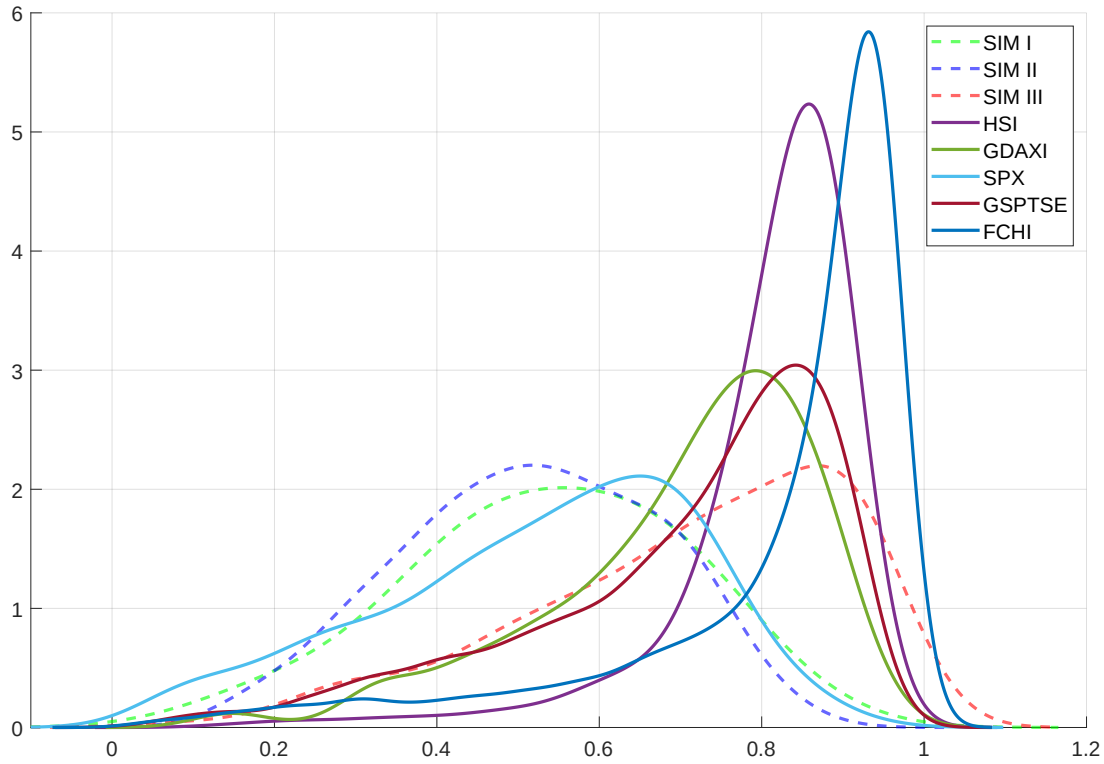


**Figure 5.7.** Posterior densities of the moving average weight $\alpha$ on simulation and real datasets. (This is better viewed in colour).

the filtered values of $z_t$ of the LMSV model in Appendix C.1. Figure 5.8 plots the filtered log volatility of the SV and SR-SV model, together with the filtered values of the components $h_t$ and $\eta_t$ of the SR-SV model in all time steps, for the SPX data. Figure C.3, C.2, C.4 and C.5 in Appendix C.3 show the similar plots for the HSI, DAX, FCHI and TSX data, respectively. Figure 5.8 shows that the component $h_t$, and hence $\eta_t$, of the SR-SV model is well responsive to changes in the volatility dynamics, e.g. being small during the low volatility periods and large in the high volatility periods of the SPX data. The SRU structure of the SR-SV model is able to capture these distinct behaviors of financial time series. We observe the similar behaviors of the $h_t$ and $\eta_t$ components for the other datasets as shown in the Figures C.2-C.5. Table 5.10 provides summary statistics on the in-sample filtered volatilities and residuals $\widehat{\epsilon}_t^y$ of the SR-SV, SV, N-SV and LMSV models. We note that if the assumption about the normality of returns of the three models is justified then the residuals $\widehat{\epsilon}_t^y$ should have a zero skewness and a kurtosis of 3. Table 5.10 shows that all the residual distributions produced from the four models are close to the standard normal distribution, but still slightly skewed and leptokurtic. The $p$-values of the Ljung-Box (LB) autocorrelation test of the residuals are high in all index datasets, suggesting that there is no evidence of autocorrelation in

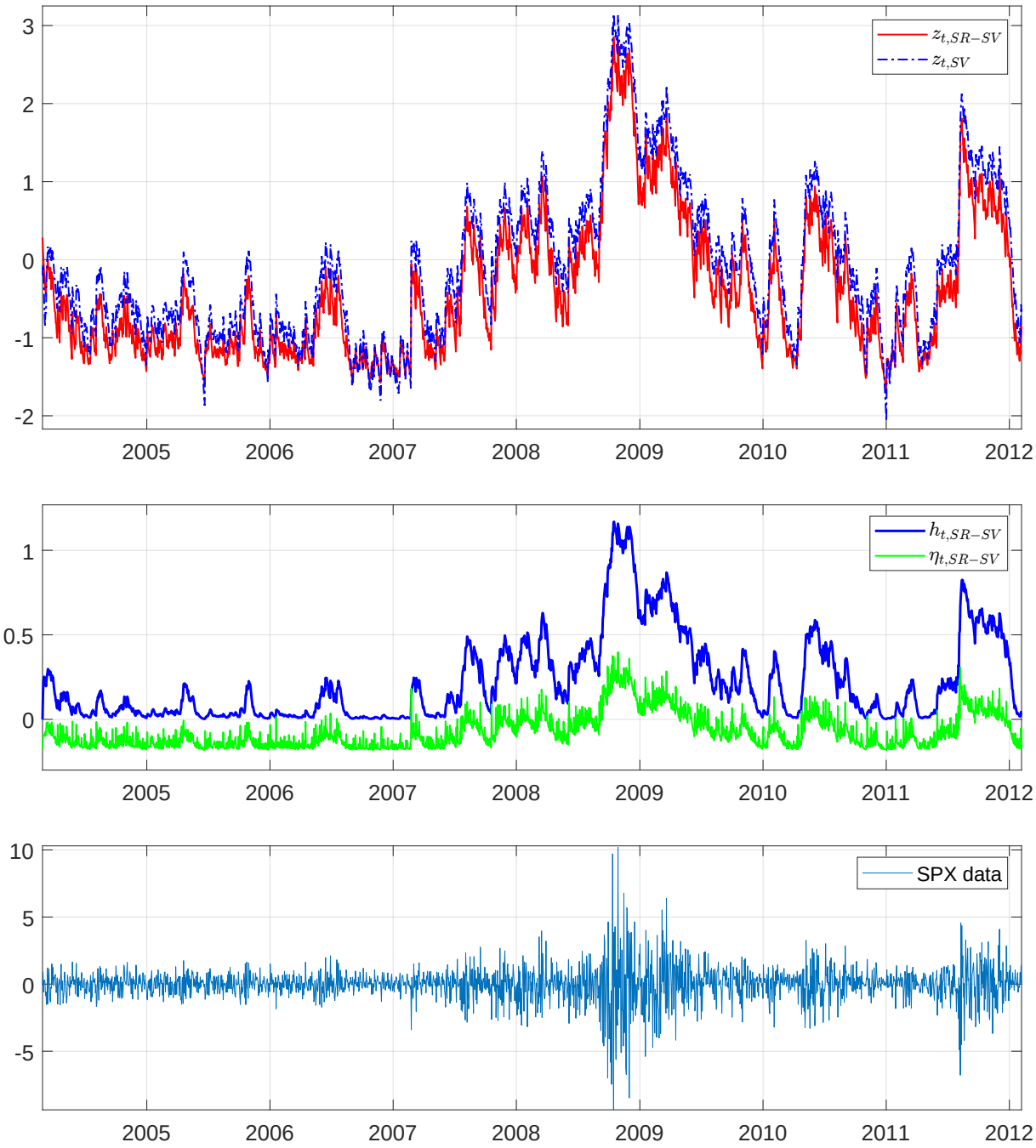


**Figure 5.8.** SPX: (*Top*) The filtered log volatility of the SR-SV and SV models. (*Middle*) The filtered values of $\eta_t$ and $h_t$ of the SR-SV model. (*Bottom*) The SPX in-sample data. (This is better viewed in colour).

the residuals. We conjecture that extending the SR-SV model, for example, by using a Student's $t$ distribution instead of a Gaussian for the measurement shock $\epsilon_t^y$ and taking into account the leverage effect by correlating $\epsilon_t^y$ with the volatility shock $\epsilon_t^\eta$, is likely to lead to better diagnostics for the residuals. However, we do not consider these extensions here.

| | Filtered volatility | | | | Residual $\widehat{\epsilon}_t^y$ | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Kurtosis | Skew | Std | Kurtosis | Skew | LB-$\widehat{\epsilon}_t$ |
| **DAX** | | | | | | | | |
| SV | 1.531 | 1.930 | 24.465 | 4.075 | 0.985 | 2.817 | −0.215 | 0.978 |
| N-SV | 1.591 | 2.290 | 38.239 | 5.112 | 0.982 | 2.742 | −0.213 | 0.978 |
| LMSV | 1.423 | 1.670 | 13.734 | 2.885 | 0.999 | 3.584 | −0.163 | 0.887 |
| SR-SV | 1.325 | 1.559 | 26.276 | 4.182 | 0.991 | 2.819 | −0.207 | 0.983 |
| **HSI** | | | | | | | | |
| SV | 1.343 | 2.084 | 55.696 | 6.187 | 0.966 | 2.801 | −0.040 | 0.162 |
| N-SV | 1.428 | 3.133 | 179.820 | 11.399 | 0.965 | 2.776 | −0.040 | 0.226 |
| LMSV | 1.271 | 1.632 | 22.216 | 3.720 | 0.999 | 3.946 | −0.028 | 0.326 |
| SR-SV | 1.101 | 1.651 | 50.909 | 5.821 | 0.978 | 2.768 | −0.052 | 0.132 |
| **FCHI** | | | | | | | | |
| SV | 1.416 | 1.563 | 13.561 | 2.851 | 0.982 | 2.724 | −0.123 | 0.101 |
| N-SV | 1.483 | 1.867 | 22.140 | 3.705 | 0.981 | 2.690 | −0.117 | 0.103 |
| LMSV | 1.287 | 1.451 | 11.815 | 2.645 | 1.000 | 3.548 | −0.058 | 0.108 |
| SR-SV | 1.167 | 1.159 | 13.285 | 2.756 | 0.985 | 2.722 | −0.136 | 0.105 |
| **SPX** | | | | | | | | |
| SV | 1.648 | 2.712 | 25.696 | 4.395 | 0.994 | 2.769 | −0.254 | 0.136 |
| N-SV | 1.729 | 3.139 | 33.443 | 5.010 | 0.995 | 2.737 | −0.251 | 0.143 |
| LMSV | 1.592 | 2.810 | 25.516 | 4.381 | 0.999 | 3.623 | −0.103 | 0.219 |
| SR-SV | 1.892 | 3.862 | 29.532 | 4.856 | 1.002 | 2.746 | −0.252 | 0.149 |
| **TSX** | | | | | | | | |
| SV | 1.543 | 2.516 | 27.582 | 4.641 | 0.971 | 2.705 | −0.311 | 0.977 |
| N-SV | 1.590 | 2.846 | 35.343 | 5.256 | 0.971 | 2.683 | −0.307 | 0.969 |
| LMSV | 1.395 | 2.124 | 20.325 | 3.893 | 0.999 | 3.353 | −0.301 | 0.915 |
| SR-SV | 1.389 | 2.347 | 28.167 | 4.702 | 0.973 | 2.659 | −0.310 | 0.961 |

**Table 5.10.** Applications: Model diagnostics of the filtered log volatility and residual $\widehat{\epsilon}_t^y$. The LB p-values denote the p-value from the Ljung-Box test with 10 lags.

### Out-of-sample analysis

The marginal likelihood estimates in Table 5.8 suggest that the SR-SV model fits the in-sample data of the five index datasets better than the SV and N-SV models. We now examine if this in-sample performance is consistent with the out-of-sample performance. Table 5.11 provides summary statistics on the one-step-ahead forecasts of volatility and standardized residuals of the SR-SV, SV, N-SV and LMSV models. We note two conclusions from Table 5.11.

First, the SR-SV model does not suffer from overfiting as often observed in neural network based volatility models (Pagan and Schwert, 1990; Donaldson and Kamstra, 1997), as the one-step-ahead forecast volatilities and the forecast residuals appear to be well behaved, compared to those from the more parsimonious SV and N-SV models. We emphasize that, as discussed in Section 5.2.2, the use of noise-injecting regularization in the novel structure of the SR-SV model helps prevent it from the well-known overfitting problem.

Second, the means and standard deviations of one-step-ahead forecast volatility of the SR-SV model are smaller than those of the SV, N-SV and LMSV in all five index datasets. The SR-SV forecasts are generally more conservative in low volatility periods, in the sense that the forecast intervals often have a smaller band compared to the forecasts produced by the other models. The comparison of 99% one-step-ahead forecast intervals during the period Sep 2014 - May 2015 of the SPX data in Figure 5.9 shows that the SR-SV model gives a safe buffer against abrupt changes in low volatility regions, e.g. Nov-Dec 2014, because it maintains a wider forecast band, while it does not produce overly large forecast intervals in high volatility regions, e.g. Oct 2014, Dec 2014 - Jan 2015. Therefore, the SR-SV model is less sensitive to the data values in the shorter time periods, and maintains a good trade-off between the information in recent observations and the information in the long-term memory. The SV, N-SV and LMSV models, compared to the SR-SV model, produce a smaller forecast volatility in low volatility regions and a higher volatility forecast in high volatility regions. The figure also shows that the SV and N-SV forecasts depend mainly on the return at the previous step, as the persistence parameters $\phi$ in the SV and N-SV models are larger than the persistence parameter of the SR-SV model. The SR-SV intervals are closer to the intervals made by the realized variance and hence seem to track the out-of-sample returns better than the SV, N-SV and LMSV models.

| | Forecast Volatility | | | | Forecast Residual $\widehat{\epsilon}_t^y$ | | | |
|---|---|---|---|---|---|---|---|---|
| | Mean | Std | Kurtosis | Skew | Std | Kurtosis | Skew | LB-$\widehat{\epsilon}_t$ |
| **DAX** | | | | | | | | |
| SV | 1.069 | 0.580 | 2.992 | 0.711 | 0.997 | 3.964 | −0.334 | 0.652 |
| N-SV | 1.072 | 0.588 | 3.464 | 0.893 | 0.992 | 3.937 | −0.342 | 0.659 |
| LMSV | 1.083 | 0.616 | 3.467 | 0.879 | 1.001 | 3.942 | −0.310 | 0.577 |
| SR-SV | 0.943 | 0.458 | 3.261 | 0.879 | 1.034 | 3.835 | −0.328 | 0.597 |
| **HSI** | | | | | | | | |
| SV | 0.655 | 0.380 | 14.099 | 2.823 | 0.982 | 4.353 | 0.036 | 0.390 |
| N-SV | 0.649 | 0.408 | 23.440 | 3.827 | 0.981 | 4.298 | 0.021 | 0.379 |
| LMSV | 0.740 | 0.386 | 9.633 | 2.065 | 0.928 | 4.465 | 0.057 | 0.283 |
| SR-SV | 0.491 | 0.215 | 15.963 | 3.102 | 1.091 | 4.135 | −0.008 | 0.368 |
| **FCHI** | | | | | | | | |
| SV | 1.087 | 0.620 | 3.796 | 0.989 | 0.963 | 4.590 | −0.436 | 0.734 |
| N-SV | 1.089 | 0.646 | 4.629 | 1.249 | 0.965 | 4.415 | −0.432 | 0.686 |
| LMSV | 1.144 | 0.633 | 5.126 | 1.400 | 0.971 | 4.656 | −0.371 | 0.657 |
| SR-SV | 0.894 | 0.434 | 3.626 | 0.918 | 0.995 | 4.174 | −0.375 | 0.609 |
| **SPX** | | | | | | | | |
| SV | 0.675 | 0.438 | 9.938 | 2.135 | 0.983 | 3.970 | −0.456 | 0.352 |
| N-SV | 0.679 | 0.453 | 12.583 | 2.503 | 0.978 | 3.970 | −0.458 | 0.365 |
| LMSV | 0.763 | 0.422 | 4.993 | 1.291 | 0.920 | 4.075 | −0.416 | 0.547 |
| SR-SV | 0.523 | 0.287 | 9.978 | 2.242 | 1.074 | 3.797 | −0.406 | 0.450 |
| **TSX** | | | | | | | | |
| SV | 0.551 | 0.314 | 3.231 | 0.976 | 0.960 | 3.859 | −0.537 | 0.123 |
| N-SV | 0.551 | 0.304 | 3.412 | 1.035 | 0.952 | 3.811 | −0.533 | 0.105 |
| LMSV | 0.541 | 0.272 | 3.156 | 0.798 | 0.983 | 4.060 | −0.583 | 0.098 |
| SR-SV | 0.500 | 0.229 | 3.948 | 1.234 | 0.973 | 3.734 | −0.502 | 0.077 |

**Table 5.11.** Applications: Summary statistics on the one-step-ahead out-of-sample forecast conditional variances $\widehat{\sigma}_t^2$ and residual $\widehat{\epsilon}_t$. The LB p-values denote the p-value from the Ljung-Box test with 10 lags.

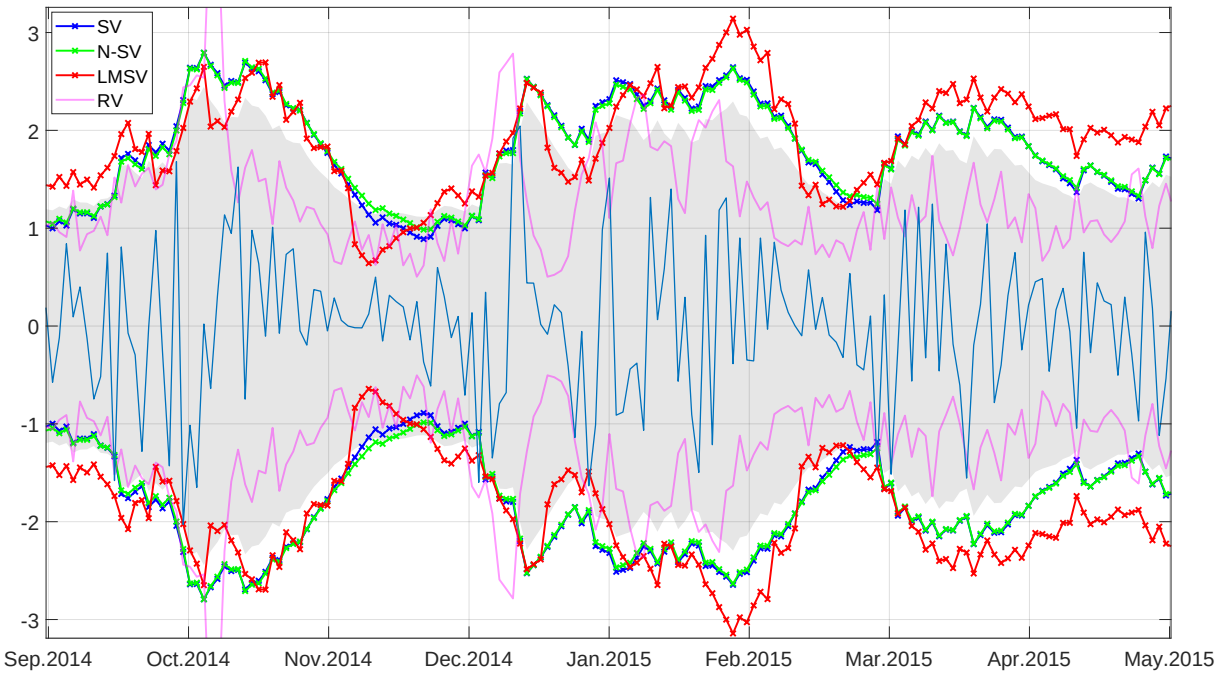


**Figure 5.9.** SPX: 99% One-step-ahead forecast intervals of the SR-SV, SV, N-SV and LMSV models, together with the 99% interval of index data estimated by the realized variance (RV) during Sep 2014 - May 2015. The shaded area is the 99% one-step-ahead forecast interval produced by the SR-SV model.

Table 5.12 shows the out-of-sample performances of the four models on the SPX data, with the Monte Carlo standard errors in brackets. Tables C.3, C.4, C.5 and C.6 in Appendix C.3 show the results for the other four datasets. Each table provides predictive scores separately in four panels, corresponding to the four realized measures BV, MedRV, RKV and RV, as discussed Section 5.4.2. In each panel, we count the number of times a particular model has lowest (best) predictive scores and list these numbers in the last column; the model with the highest count is preferred. We note that the PPS predictive score is independent of the realized measures of volatility, and that the PPS predictive score is not available for the LMSV model.

As shown in these tables, the SR-SV model consistently has the best out-of-sample performance in all the five index datasets. The superior predictive performance of the SR-SV model is consistent with its in-sample performance discussed earlier and provides further evidence to support the conclusion that the SR-SV model does not overfit the five index datasets. We note that the forecast performances of the SV and N-SV models are mixed with no model consistently outperforming the other across the predictive scores and the datasets. The LMSV model consistently makes the least accurate forecasts in all cases .

| Measure | | PPS | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLIKE | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|---|
| | SV | 1.122 | 0.103 | 1.333 | 0.229 | 0.392 | 0.347 | 0.667 | 0 |
| | | (0.001) | (0.001) | (0.002) | (0.001) | (0.002) | (0.001) | (0.002) | |
| BV | N-SV | 1.122 | 0.103 | 1.327 | 0.229 | 0.392 | 0.346 | 0.669 | 0 |
| | | (0.000) | (0.001) | (0.002) | (0.001) | (0.002) | (0.001) | (0.003) | |
| | LMSV | | 0.125 | 1.406 | 0.265 | 0.451 | 0.402 | 0.837 | 0 |
| | SR-SV | **1.113** | **0.091** | **1.321** | **0.209** | **0.354** | **0.330** | **0.572** | 7 |
| | | (0.001) | (0.000) | (0.002) | (0.000) | (0.001) | (0.002) | (0.000) | |
| | SV | | 0.114 | 0.830 | 0.256 | 0.428 | 0.323 | 0.862 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.001) | (0.001) | (0.003) | |
| MedRV | N-SV | | 0.114 | 0.826 | 0.257 | 0.428 | 0.323 | 0.866 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.000) | (0.001) | (0.004) | |
| | LMSV | | 0.140 | 0.901 | 0.294 | 0.488 | 0.385 | 1.064 | 0 |
| | SR-SV | | **0.102** | **0.821** | **0.235** | **0.389** | **0.308** | **0.757** | 6 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.001) | (0.001) | |
| | SV | | 0.114 | 0.834 | 0.256 | 0.419 | 0.363 | 0.915 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.002) | |
| RKV | N-SV | | 0.114 | 0.829 | 0.256 | 0.420 | 0.361 | 0.918 | 0 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | LMSV | | 0.137 | 0.904 | 0.290 | 0.476 | 0.405 | 1.100 | 0 |
| | SR-SV | | **0.101** | **0.822** | **0.237** | **0.384** | **0.345** | **0.808** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | SV | | 0.121 | 1.864 | 0.245 | 0.421 | 0.331 | 0.796 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.002) | (0.000) | (0.002) | |
| RV | N-SV | | 0.120 | 1.863 | 0.246 | 0.422 | 0.329 | 0.799 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.001) | (0.001) | (0.003) | |
| | LMSV | | 0.145 | 1.95 | 0.283 | 0.484 | 0.385 | 0.983 | 0 |
| | SR-SV | | **0.108** | **1.861** | **0.224** | **0.382** | **0.316** | **0.692** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.001) | (0.000) | |

**Table 5.12.** SPX data: Forecast performance of the SR-SV and benchmark models using different realized measures. In each panel, the bold numbers indicate the best predictive scores.

## 5.5 Chapter summary

This chapter proposes a statistical recurrent stochastic volatility (SR-SV) model, by combining the statistical recurrent unit architecture from Machine Learning and the stochastic volatility model from Financial Econometrics. These two techniques are combined in a principled and non trivial way to form a new approach that is, as carefully illustrated through the extensive simulation and empirical studies, highly efficient for volatility modelling and forecasting. It is easy to carry Bayesian inference in the SR-SV model using standard Bayesian computation methods, such as the Density Tempered Sequential Monte Carlo method as used in this paper. The simulation and empirical studies suggest that the SR-SV model is able to capture various volatility effects overlooked by the SV benchmark models, and is able to produce highly accurate forecast volatilities.

# Chapter 6

# Conclusion and future directions

This chapter concludes the dissertation and suggests directions for future work.

The dissertation presents new classes of statistical models with high in-sample and out-of-sample performance. The proposed models combine deep learning and statistical structures in flexible ways to leverage the advantages of both worlds of deep learning and statistical/financial econometrics modeling. We show that our novel hybrid models work efficiently for different data types, e.g. cross-section/panel (Chapter 3) and time series data (Chapters 4 and 5), in a wide range of business and financial econometrics applications.

**Chapter 3** introduces improved versions of generalized linear and generalized linear mixed models by using DFNN structure as the data presentation technique to transform the raw covariates task automatically and efficiently. The applications on real data show that the so-called DeepGLM and DeepGLMM models work efficiently with cross-sectional and panel data, e.g. have better predictive performance than the statistical benchmark models, which are commonly available in many business and financial application. The proposed frameworks also provide a systematic way to perform variable selections, which is a challenging task for neural network based models. The chapter also discusses a novel and computationally efficient variational approximation methods with a parsimonious factor covariance structure to quickly estimate the high-dimensional deep learning based models. The study on the computational performance of the so-called NAGVAC method demonstrate the efficiency of our proposed method over recent advanced alternative Bayesian approximation techniques. The DeepGLM and DeepGLMM models come with an user-friendly software package written on Matlab, R and Python, which is an important contribution to the financial econometrics literature.

The promising results with the DeepGLMM model have shown that the use of neural network basis functions with random effects is a class of models that deserve more consideration in the literature. For example, combining the random effects concept and the Neural Process Family (Garnelo et al., 2018b,a; Kim et al., 2019) to model the panel data more efficiently.

**Chapter 4** proposes a new class of conditional heteroskedastic models, which we call the RECH models, by incorporating a RNN structure into the conditional variance of the GARCH-type models, and study in detail three RECH specifications: SRN-GARCh, SRN-GJR and SRN-EGARCH. The RECH models improve the traditional GARCH-type models by using RNN to learn the volatility effects cannot be captured by the simple linear deterministic process of the GARCH structure. The

proposed additive structure of the volatility process allows the RECH framework to leverage the advantages of both deep learning and GARCH structures. The extensive simulation and empirical studies suggest that the RECH models not only have both impressive in-sample fitting and accurate out-of-sample forecasts, but can also add a satisfactory level of interpretation to explain the volatility movement. A software package to demonstrate the efficiency of our proposed RECH models are also provided.

An attractive feature of the proposed hybrid framework is that it is easy to use advances in both the deep learning and volatility modeling literatures to extend the current RECH models. This opens up many interesting future applications and areas of research. For example, one can use the Fourier Recurrent Unit (FRU) of Zhang et al. (2018) to construct the recurrent component of the RECH framework; FRU is currently considered as the state-of-the-art RNN architecture in deep learning. For the garch component, one can use the Bad Environment - Good Environment (BEGE) model of Bekaert et al. (2015), which can efficiently simulate the heavy tailed behavior of financial returns. Another interesting research direction is extending the univariate RECH models to the multivariate case, e.g. combining RNN and the Dynamic Conditional Correlation (DCC) model of (Engle, 2002). We conjecture that the recurrent neural network architectures will be more powerful for multivariate inputs as they can naturally capture the interaction between the inputs.

**Chapter 5** proposes a statistical recurrent stochastic volatility (SR-SV) model, by combining the statistical recurrent unit (SRU) architecture from Machine Learning and the stochastic volatility model from Financial Econometrics in a principled and non trivial way. Some important modifications of the SRU structure are proposed to make the SR-SV model more interpretable and explain better the volatility dynamics. Consequently, we show that the SR-SV outperform the SV-type benchmark models through an intensive simulation study and real stock indexes, in terms of both in-sample fitting and out-of-sample forecasting.

Extending the SR-SV model by incorporating features such as the leverage effect is an interesting research question. Another interesting research question is also extending the present SR-SV model to multivariate financial time series, e.g. flexibly combine RNN and multivariate factor SV models.

We note that the comparison between the SR-SV and RECH models is similar to the comparison between SV and GARCH models, due to the way to two models are

designed. SR-SV is a parameter-driven model while RECH is an observation-driven model, both offer different perspectives of how to model the financial volatility. Whether or not parameter-driven models are better than observation-driven models is still unclear, and in fact these models are all widely used in the literature. Hence, RECH and SR-SV complement each other. The necessity for introducing both SR-SV and RECH is similar to the necessity for introducing both SV and GARCH models in the literature. More specifically, the volatility dynamics in SR-SV is a *stochastic* latent process rather than a *deterministic* process as in RECH. Also, SR-SV requires the complicated SRU structure in order to work well while RECH only requires the most basic RNN structure. The number of model parameters in SR-SV is therefore much larger than that in RECH. This, together with the intractable likelihood, makes it difficult and computationally expensive to do Bayesian inference in SR-SV. Technically, SR-SV requires methods for dealing with overfitting while this isn't a problem with RECH. Finally, The research questions are also different in the Chapter 4 and 5. Chapter 4 doesn't develop a single best GARCH-type model, but shows that, by incorporating an RNN structure into a given GARCH-type model, one can improve on it. Chapter 5, on the other hand, tries to find a good combination of RNN and SV models.

# Appendix A

# Additional details and results of Chapter 3

## A.1 Rank-1 natural gradient VB estimation method

Recall that $\Sigma = BB^\top + D^2$ with $B = (b_1, \cdots, b_d)^\top$ and $D = \text{diag}(c)$, $c = (c_1, \cdots, c_d)^\top$. Then,

$$\Sigma^{-1} = D^{-2} - \frac{1}{1+\kappa} D^{-2} BB^\top D^{-2},\ \kappa_1 = B^\top D^{-2} B = \sum \frac{b_i^2}{c_i^2},$$

and $B^\top \Sigma^{-1} B = \kappa_1/(1+\kappa_1)$. Hence, $I_{22}^{-1} = \frac{1+\kappa_1}{2\kappa_1}\Sigma$. We still set $I_{23} = 0$ but $I_{33}$ can be computed analytically as follows. Let $\Sigma^{-1} = D^{-2} - hh^\top$ with $h = D^{-2}B = (1/\sqrt{1+\kappa_1})B \circ c^{-2}$ and $h^2 = h \circ h$.

$$\begin{aligned} I_{33} &= 2D(\Sigma^{-1} \circ \Sigma^{-1})D \\ &= 2\left(D^{-2} - D(hh^\top) \circ D^{-1} - D^{-1} \circ D(hh^\top) + D\{(hh^\top) \circ (hh^\top)\}D\right) \\ &= 2\left(\text{diag}(c^2 - 2h^2) + Dh^2(Dh^2)^\top\right) \\ &= 2\left(\text{diag}(v_1) + v_2 v_2^\top\right), \end{aligned}$$

with $v_1 = c^2 - 2h^2$ and $v_2 = Dh^2 = c \circ h^2$. Then,

$$I_{33}^{-1} = \frac{1}{2}\left(\text{diag}(v_1^{-1}) + \frac{1}{1+\sum v_{2i}^2/v_{1i}}(v_1^{-1} \circ v_2)(v_1^{-1} \circ v_2)^\top\right)$$

It's important to note that it's unnecessary to store the matrix $I_F^{-1}$. To obtain the natural gradient, all we need is the matrix-vector product of the form $I_F^{-1} g$. Write $g = (g_1^\top, g_2^\top, g_3^\top)^\top$ with $g_1$ the vector formed by the first $d$ elements of $g$, $g_2$ the vector formed by the next $d$ elements, and $g_3$ the last $d$ elements in $g$. The natural gradient is

$$g^{\text{nat}} = \begin{pmatrix} (g_1^\top B)B + c^2 \circ g_1 \\ \frac{1+\kappa_1}{2\kappa_1}\left((g_2^\top B)B + c^2 \circ g_2\right) \\ \frac{1}{2}v_1^{-1} \circ g_3 + \kappa_2\left[(v_1^{-1} \circ v_2)^\top g_3\right](v_1^{-1} \circ v_2) \end{pmatrix},$$

with $\kappa_2 = \frac{1}{2}(1+\sum_1^d v_{2i}^2/v_{1i})^{-1}$. The complexity of computing the natural gradient is $O(d)$.

## A.2 Further details for the example in Section 3.7.1

The likelihood contribution w.r.t. the panel $(y_i, x_i)$ is

$$\begin{aligned} L_i(\theta) &= \int p(y_i|x_i,w,\beta,\alpha_i)p(\alpha_i|\Gamma)d\alpha_i \\ &= \int \exp\left(\sum_{t=1}^{T_i} y_{it}\mathfrak{N}(x_{it},w,\beta+\alpha_i) - \log\left(1+e^{\mathfrak{N}(x_{it},w,\beta+\alpha_i)}\right)\right) p(\alpha_i|\Gamma)d\alpha_i. \end{aligned}$$

By Fisher's identity (Gunawan et al., 2017)

$$\nabla_\theta \ell_i(\theta) = \int \nabla_\theta \left\{ \log p(\alpha_i|\Gamma) + \sum_{t=1}^{T_i} y_{it}\mathfrak{N}(x_{it},w,\beta+\alpha_i) - \log\left(1+e^{\mathfrak{N}(x_{it},w,\beta+\alpha_i)}\right) \right\} p(\alpha_i|y_i,x_i,\theta)d\alpha_i.$$

We have,

$$\begin{aligned} p(\alpha_i|y_i,x_i,\theta) &\propto p(\alpha_i|\Gamma)p(y_i|x_i,w,\beta,\alpha_i) \\ &\propto \exp\left(\sum_{t=1}^{T_i}\left[y_{it}z_{it}^\top(\beta+\alpha_i) - \log(1+e^{z_{it}^\top(\beta+\alpha_i)})\right] - \frac{1}{2}\alpha_i^\top\Gamma^{-1}\alpha_i\right) = \exp(f(\alpha_i)). \end{aligned}$$

$$\nabla_{\alpha_i} f(\alpha_i) = \boldsymbol{Z}_i^\top(y_i - p_i) - \Gamma^{-1}\alpha_i,\ p_i = p_i(\alpha_i) = (p_{i1},\ldots,p_{iT_i})^\top$$

$$\nabla_{\alpha_i\alpha_i^\top} f(\alpha_i) = -\boldsymbol{Z}_i^\top \mathrm{diag}(p_i \circ (1-p_i))\boldsymbol{Z}_i - \Gamma^{-1}.$$

Let $\widehat{\mu}_{\alpha_i}$ be the maximizer of $f(\alpha_i)$ which can be obtained by the Newton-Raphson method, and let

$$\widehat{\Sigma}_{\alpha_i} = \left(-\nabla_{\alpha_i\alpha_i^\top} f(\alpha_i)|_{\alpha_i=\widehat{\mu}_{\alpha_i}}\right)^{-1} = \left(\boldsymbol{Z}_i^\top \mathrm{diag}(p_i \circ (1-p_i))\boldsymbol{Z}_i + \Gamma^{-1}\right)^{-1},\ \ p_i = p_i(\widehat{\mu}_{\alpha_i}) \tag{A.1}$$

We note that for the Gaussian flexible linear mixed model in Section 3.7.2, $\widehat{\mu}_{\alpha_i}$ and $\widehat{\Sigma}_{\alpha_i}$ can be derived analytically.

The gradient $\nabla_\theta \ell_i(\theta)$ can be estimated as follows.

- Generate $N$ samples $\alpha_i^{(j)} \sim \mathcal{N}(\widehat{\mu}_{\alpha_i}, \widehat{\Sigma}_{\alpha_i})$, $j = 1,\ldots,N$.
- Compute the weights

$$w_i^{(j)} = \exp\left(f(\alpha_i^{(j)}) + \frac{1}{2}(\alpha_i^{(j)} - \widehat{\mu}_{\alpha_i})^\top \widehat{\Sigma}_{\alpha_i}^{-1}(\alpha_i^{(j)} - \widehat{\mu}_{\alpha_i})\right)$$

  and $W_i^{(j)} = w_i^{(j)} / \sum_{k=1}^N w_i^{(k)}$.

- Compute the estimate

$$\widehat{\nabla_\theta \ell_i(\theta)} = \sum_{j=1}^{N} \nabla_\theta \left\{ \log p(\alpha_i^{(j)}|\Gamma) + \sum_{t=1}^{T_i} \left[ y_{it} z_{it}^\top (\beta + \alpha_i^{(j)}) - \log(1 + e^{z_{it}^\top (\beta + \alpha_i^{(j)})}) \right] \right\} W_i^{(j)}.$$

Because the parameters $\Gamma_j$ are positive, a suitable transformation is needed before applying the Gaussian VB approximation. We use the transformation $\theta_{\Gamma_j} = \log(\Gamma_j)$, $j = 0,...,M$. Let $\widetilde{\theta} = (w, \beta, \theta_{\Gamma_0},...,\theta_{\Gamma_m})$, then,

$$\theta = \theta(\widetilde{\theta}) = (w, \beta, \exp(\theta_{\Gamma_0}),...,\exp(\theta_{\Gamma_m})).$$

The posterior distribution of $\widetilde{\theta}$ is

$$p(\widetilde{\theta}|D) \propto \left| \frac{\partial \theta(\widetilde{\theta})}{\partial \widetilde{\theta}} \right| p(\theta(\widetilde{\theta})) p(\theta(\widetilde{\theta})|D) = \exp(\theta_{\Gamma_0} + ... + \theta_{\Gamma_m})) p(\theta(\widetilde{\theta})) p(\theta(\widetilde{\theta})|D).$$

We then approximate $p(\widetilde{\theta}|D)$ by $q_\lambda(\widetilde{\theta})$.

### A.2.1 NAGVAC as a general training method

This section illustrates that the training method NAGVAC can be used as a general estimation method for high-dimensional models rather than neural network based models. The application is concerned with high-dimensional logistic regression using a sparse signal shrinkage prior, the horseshoe prior (Carvalho et al., 2010). Here the variational optimization is challenging because of the strong dependence between local variance parameters and the corresponding coefficients. Using three real datasets we show that the natural gradient estimation method improves the performance of the approach described in Ong et al. (2017a).

Let $y_i \in \{0,1\}$ be a binary response with the corresponding covariates $x_i = (x_{i1},...,x_{ip})^\top$, $i = 1,...,n$. We consider the logistic regression model $\text{logit}(\mu_i) = \beta_0 + x_i^\top \beta$, where $\mu_i = \text{P}(y_i = 1|x_i)$, $\beta_0$ is an intercept and $\beta = (\beta_1,...,\beta_p)^\top$ are coefficients. We consider the setting where $p$ is large, possibly $p \gg n$, and use the horseshoe prior for $\beta$ (Carvalho et al., 2010). Specifically we assume $\beta_0 \sim N(0,10)$ and

$$\beta_j|\lambda_j \sim N(0, \lambda_j^2 g^2), \quad \lambda_j \sim C^+(0,1), j = 1,...,p, \quad g \sim C^+(0,1),$$

where $C^+(0,1)$ denotes the half-Cauchy distribution. The parameters $\lambda_j$, $j = 1,...,p$, are local variance parameters providing shrinkage for individual coefficients, and the

### A.2.1 NAGVAC as a general training method

| | VAFC of Ong et al. (2017a) | | | NAGVAC-4 | | |
|---|---|---|---|---|---|---|
| | Train Error | Test Error | CPU | Train Error | Test Error | CPU |
| Colon | 0/42 | 0/20 | 4.92 | 0/42 | 0/20 | 0.17 |
| Leukemia | 0/38 | 6/34 | 61 | 0/38 | 1/34 | 0.56 |
| Breast | 0/38 | 1/4 | 61.6 | 0/38 | 0/4 | 0.56 |

**Table A.1.** Performance of the ordinary gradient VAFC and natural gradient VAFC methods on three cancer datasets. Training and test errors rates are reported as the ratio of misclassified data points over the number of data points. Computational time CPU (per 100 iterations) is measured in second.

parameter $g$ is a global shrinkage parameter which can adapt to the overall level of sparsity. The above prior settings are the same as those considered in Ong et al. (2017a).

The parameter $\theta$ consists of $\theta = (\beta_0,\beta^\top,\delta^\top,\gamma)^\top$, where $\delta = (\delta_1,...,\delta_p)^\top = (\log\lambda_1,...,\log\lambda_p)^\top$, and $\gamma = \log g$. We consider Gaussian variational approximation for the posterior of $\theta$, using a factor covariance structure. The three gene expression datasets are the Colon, Leukaemia and Breast cancer datasets found at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html. These three datasets Colon, Leukaemia and Breast have training sample sizes of 42, 38 and 38 and test set sizes of 20, 34 and 4 respectively. The number of covariates is $p = 2000$ for the Colon data, and $p = 7120$ for the Leukaemia and Breast datasets. This means that for the Leukaemia and Breast datasets the dimension of $\theta$ is 14,242 so these are examples with a high dimensional parameter. These data were also considered in Ong et al. (2017a) where slow convergence in the variational optimization was observed using their method; we show here that a natural gradient approach offers a significant improvement.

Table A.1 compares the performance of VAFC of Ong et al. (2017a) and our NAGVAC training methods. The table shows the predictive performance and computational time on three cancer datasets. We follow Ong et al. (2017a) and run VAFC with $f = 4$ factors and use only a single sample to estimate the gradient of lower bound ($S = 1$) in two methods. As shown, the NAGVAC training method significantly improves the performance of VAFC.

# Appendix B

# Additional details and results of Chapter 4

## B.1 Additional results for section 4.4

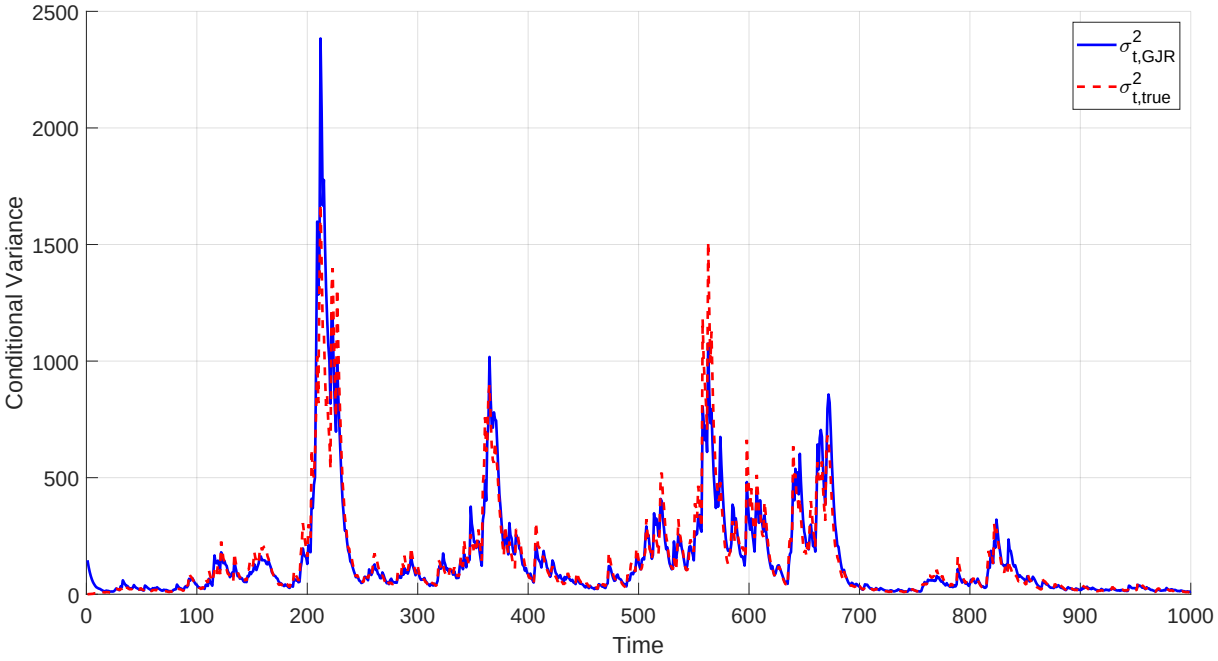


**Figure B.1.** SIM II: The true conditional variance (dashed line) and estimated conditional variance (solid line) using the GJR model. (The figure is better viewed in colour).

## B.2 Application to exchange rate data

This section reports the application of the RECH model to analyse the USD/GBP daily exchange rates observed from 21/03/2001 to 01/03/2009[8]. We use the first 1000 observations for model estimation and the last 1000 observations for evaluating

[8]The dataset was also downloaded from the Realized Library of The Oxford-Man Institute

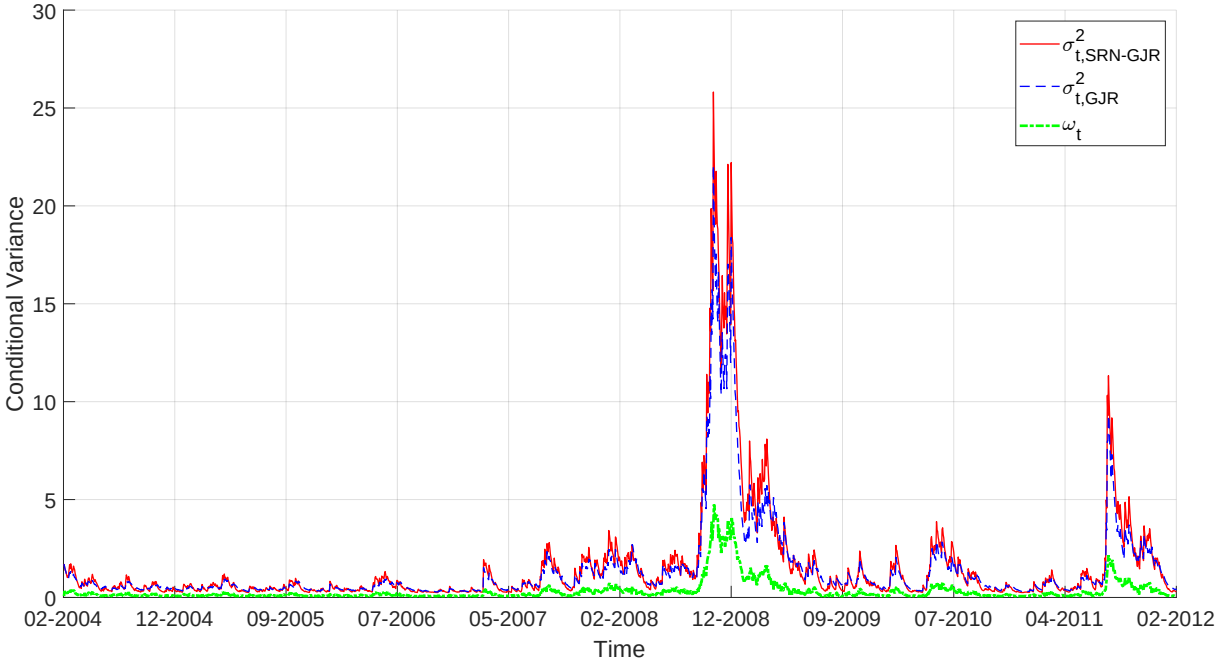


**Figure B.2.** SP500: The in-sample conditional variance of the GJR (dashed line) and SRN-GJR (solid line) at all time steps. The bottom line shows the values of the recurrent component $\omega_t$ of the SRN-GJR specification. (The figure is better viewed in colour).

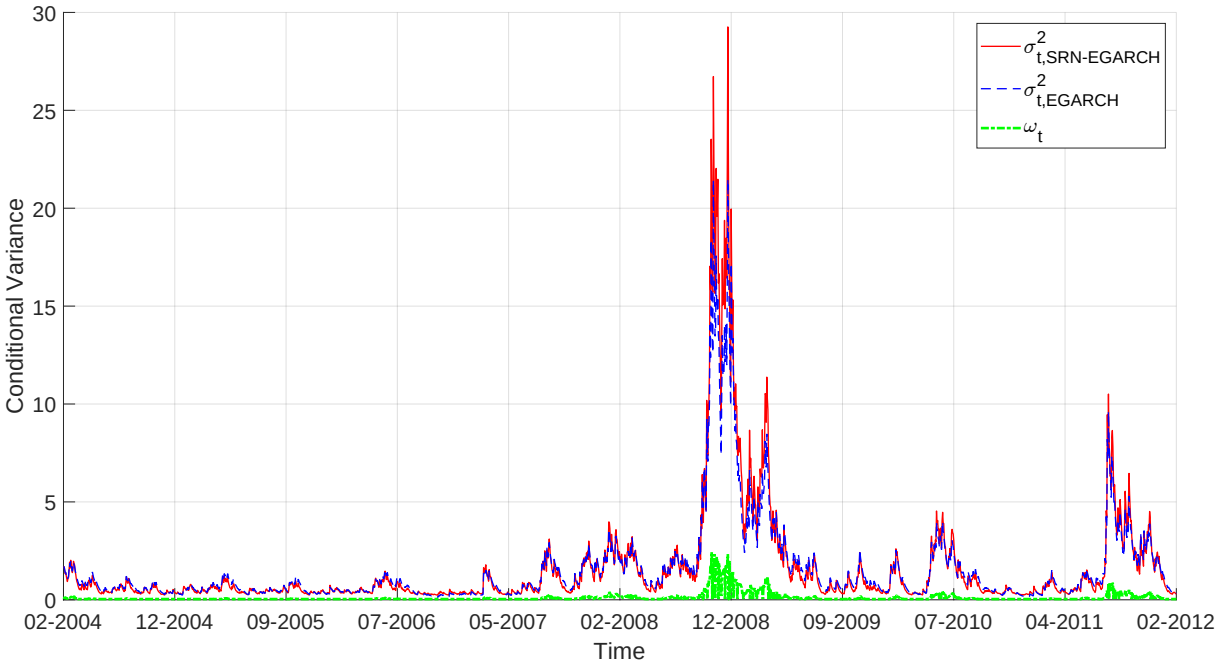


**Figure B.3.** SP500: The in-sample conditional variance of the EGARCH (dashed line) and SRN-EGARCH (solid line) at all time steps. The bottom line shows the values of the recurrent component $\omega_t$ of the SRN-EGARCH specification. (The figure is better viewed in colour).

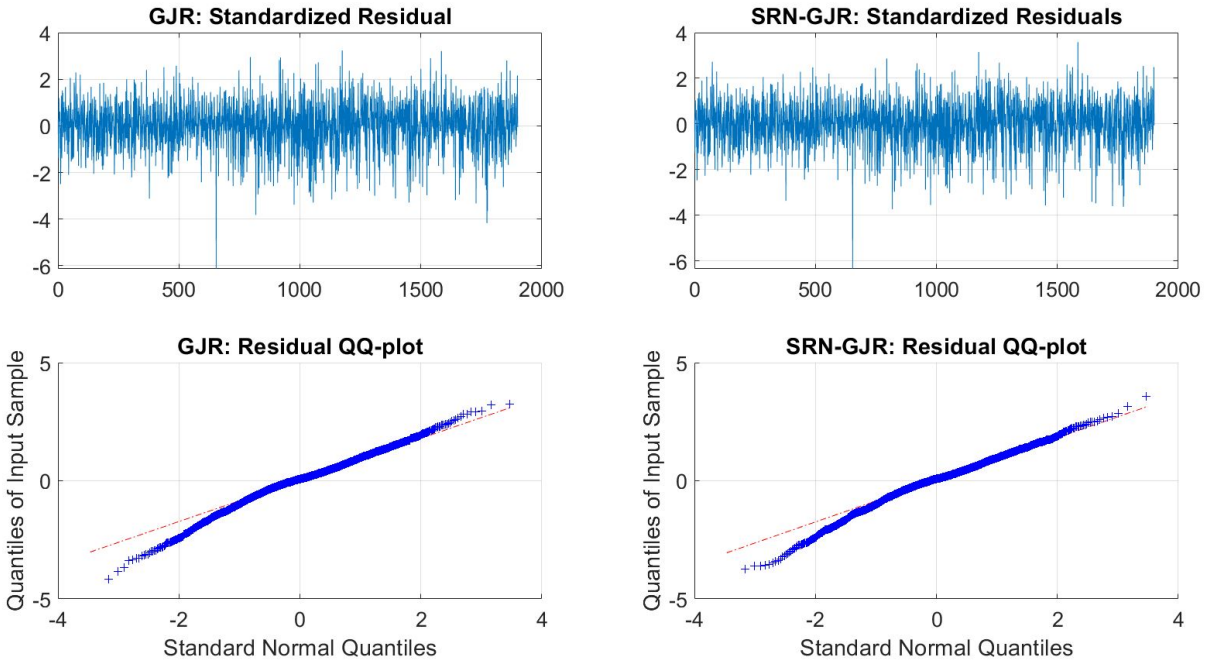


**Figure B.4.** SP500: Estimated residuals $\widehat{\epsilon}_t$ of the SRN-GJR and GJR models and their Q-Q plots.

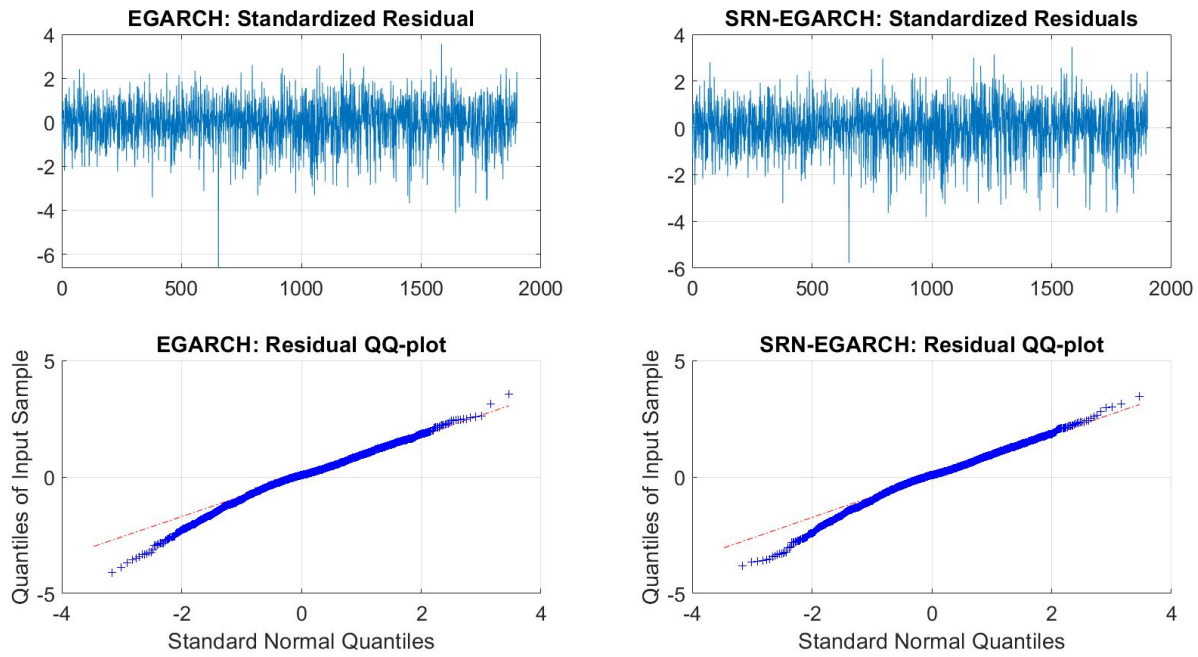


**Figure B.5.** SP500: Estimated residuals $\widehat{\epsilon}_t$ of the SRN-EGARCH and EGARCH models and their Q-Q plots.

predictive performance. The SMC with likelihood and data annealing samplers are used with the same setting shown in Table 4.2. We use realized kernal measures of the conditional variance, available in the dataset, to access the out-of-sample performance

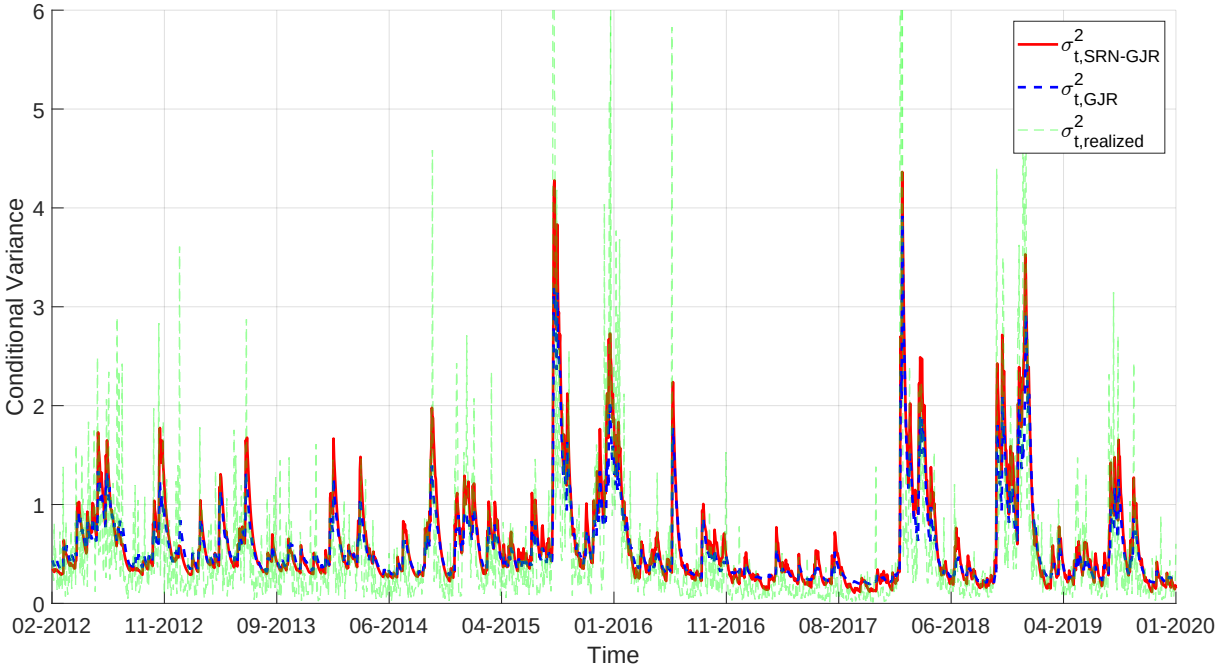


**Figure B.6.** SP500: Forecast conditional variance by the GJR (dashed) and SRN-GJR (solid) models, together with the realized variance (dotted). (The figure is better viewed in colour).

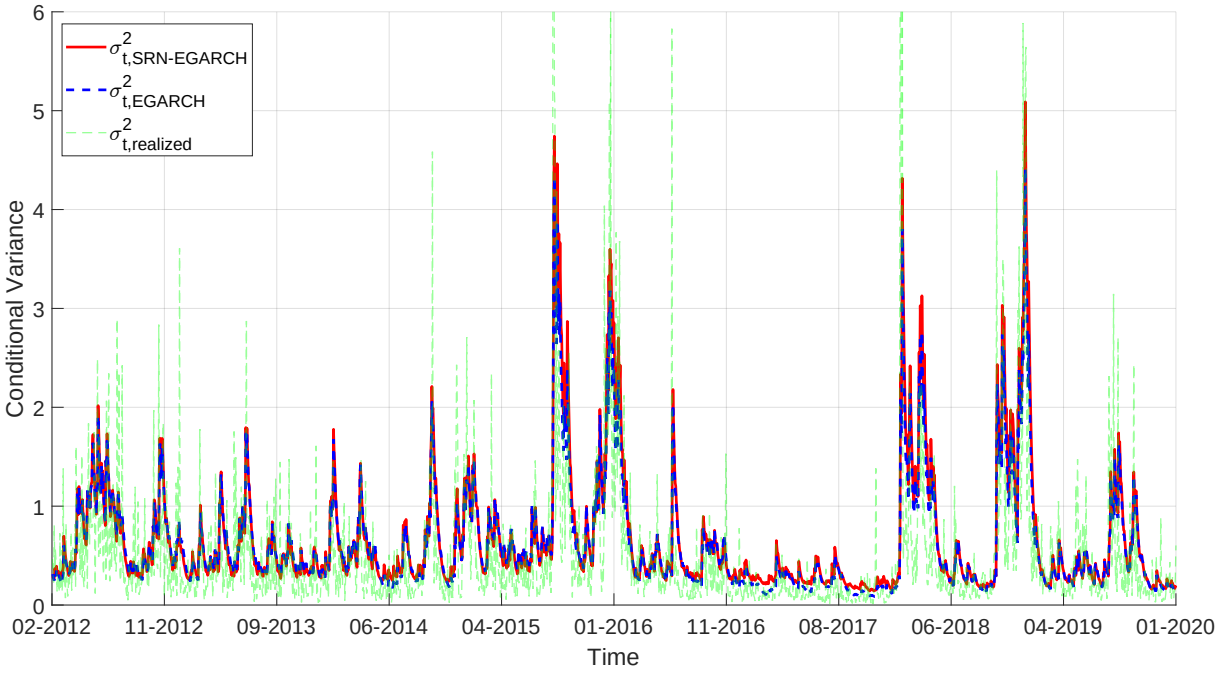


**Figure B.7.** SP500: Forecast conditional variance by the EGARCH (dashed) and SRN-EGARCH (solid) models, together with the realized variance (dotted). (The figure is better viewed in colour).

of the RECH and GARCH-type models. The in-sample result (not shown here) suggests that, unlike for the stock data, for exchange rate data GARCH performs the

best compared to GJR and EGARCH. This is consistent with the study of Hansen and Lunde (2005) who show that there is no evidence that GARCH is outperformed by more sophisticated models on the DM/USD exchange rates. The in-sample result also shows that SRN-GARCH outperforms GARCH, thus being the best model, in terms of marginal likelihood. Table B.1 summarizes the estimation results from which we draw the following conclusions. First, the marginal likelihood estimates show that the RECH models fit the exchange rate dataset better than the GARCH-type models, except for the SRN-EGARCH model. Second, the estimation results suggests that there is possibly difference in the volatility dynamics of stock and exchange rate returns. The GJR and EGARCH models do not fit the exchange data better than the GARCH model, which is different to the results obtained for stock index datasets in Section 4.4.2. This is similar to the study of Hansen and Lunde (2005), showing that there is no evidence that the GARCH model is outperformed by more sophisticated models on the DM/USD exchange rate, whereas the GARCH model is clearly inferior to the models equipped with leverage effects, e.g. GJR, EGARCH, in the analysis of the IBM stock returns. Interestingly, the SRN-GARCH model still fits the exchange data better than the GARCH model and is the best model in term of in-sample fitting compared to other RECH and GARCH-type models.

Table B.2 summarizes the forecast performance measured by the predictive scores discussed in Section 4.4, which suggests that the RECH models are able to improve on their counterpart GARCH-type models in terms of volatility forecast.

## B.3 Runtime of the SMC with likelihood annealing sampler

As the SMC sampler is parallelizable, the running time depends on how we parallelize the algorithm. For example, we can run the algorithm on a single multi-core machine or multiple multi-core machines on a high performance computing cluster. Table B.3 shows the runtime of the SMC sampler, with $M = 1000$ and $M = 10000$ particles, for the GARCH and SRN-GARCH models on a standard laptop with moderate specification: Intel Core i7, 16GB RAM - 6 cores $\sim$ 2.2GHz. In most of cases, using $M = 1000$ particles is sufficient to obtain consistent estimation results.

For comparison, the table also shows the runtime of the MCMC sampler for the GARCH model by the R package `bayesGARCH` of Ardia and Hoogerheide (2010) with

## B.3 Runtime of the SMC with likelihood annealing sampler

| | $\omega$ | $\alpha$ | $\beta$ | $\gamma$ | $\beta_0$ | $\beta_1$ | $v_1$ | $v_2$ | Mar.llh |
|---|---|---|---|---|---|---|---|---|---|
| GARCH | **0.007** | **0.040** | **0.936** | | | | | | $-777.7$ |
| | (0.004) | (0.012) | (0.021) | | | | | | (0.659) |
| SRN-GARCH | | **0.032** | **0.942** | | **0.006** | **0.075** | **0.011** | **-0.009** | $-774.3$* |
| | | (0.012) | (0.023) | | (0.004) | (0.116) | (0.305) | (0.329) | (0.493) |
| GJR | **0.012** | **0.034** | **0.909** | **0.022** | | | | | $-779.9$ |
| | (0.013) | (0.015) | (0.054) | (0.035) | | | | | (0.513) |
| SRN-GJR | | **0.030** | **0.898** | **0.027** | **0.013** | **0.186** | **-0.013** | **-0.031** | $-777.5$* |
| | | (0.017) | (0.062) | (0.039) | (0.013) | (0.144) | (0.266) | (0.326) | (0.544) |
| EGARCH | **-0.047** | **0.091** | **0.962** | **0.001** | | | | | $-780.7$* |
| | (0.066) | (0.027) | (0.052) | (0.025) | | | | | (0.734) |
| SRN-EGARCH | **-4.065** | **-0.529** | **0.832** | **-0.060** | **0.207** | **0.219** | **-0.268** | **0.064** | $-783.6$ |
| | (2.075) | (1.335) | (0.280) | (0.299) | (0.063) | (0.128) | (0.262) | (0.327) | (0.480) |

**Table B.1.** USD/GBP exchange rate: Posterior means (in bold) of the parameters with the posterior standard deviations (in brackets). The last column shows the estimated log marginal likelihood with the Monte Carlo standard errors in brackets, averaged over 10 different runs of the SMC using the likelihood annealing algorithm. The asterisks indicate the cases when the Bayes Factors strongly support the RECH models over their corresponding GARCH-type models.

| | PPS | #Vio | QS | %Hit | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLike | $R^2$Log |
|---|---|---|---|---|---|---|---|---|---|---|
| GARCH | **0.827** | 17 | 0.018 | 0.017 | 0.029 | 0.228 | 0.099 | 0.169 | -0.121 | 0.166 |
| SRN-GARCH* | **0.827** | **14** | **0.017** | 0.017 | **0.024** | 0.180 | **0.094** | **0.154** | **-0.128** | **0.159** |
| GJR | 0.829 | 17 | 0.018 | 0.017 | 0.029 | 0.225 | 0.100 | 0.169 | -0.121 | 0.167 |
| SRN-GJR* | 0.829 | 15 | **0.017** | 0.017 | **0.024** | **0.178** | **0.094** | **0.154** | -0.127 | 0.160 |
| EGARCH | 0.836 | 17 | 0.019 | **0.016** | 0.039 | 0.297 | 0.112 | 0.196 | -0.097 | 0.198 |
| SRN-EGARCH* | 0.843 | 15 | **0.017** | 0.017 | 0.028 | 0.194 | 0.107 | 0.178 | -0.104 | 0.202 |

**Table B.2.** USD/GBP exchange rate: one-step-ahead forecast comparison. The bold numbers denote the best scores. For each pair of the RECH and GARCH-type models, the asterisk indicates the models having better forecast performance.

two different numbers of iterations $N = 20{,}000$ and $N = 200{,}000$, and the runtime of the MCMC sampler for the SV model using the R package `stochvol` of Hosszejni and Kastner (2020) with $N = 50{,}000$ and $N = 500{,}000$. It is important to note that, unlike the MCMC sampler, SMC is parallelizable and hence its runtime can be reduced

significantly when running on a computing cluster. In all of our examples, SMC was run on a high performance computing cluster. [9], and be able to save approximately 80% of computational time shown in Table B.3.

| | SMC | | bayesGARCH | | stochvol | |
|---|---|---|---|---|---|---|
| | $M=1000$ | $M=10000$ | $N=20000$ | $N=200000$ | $N=50000$ | $N=500000$ |
| SRN-GARCH | 73 | 475 | | | | |
| GARCH | 24 | 154 | 38 | 405 | | |
| SV | | | | | 133 | 1398 |

**Table B.3.** The runtime for the GARCH, SRN-GARCH and SV models for analysing the SP500 dataset, using various sampling methods. The runtime is in second.

[9] We use Artemis, the University of Sydney high performing computer cluster.

# Appendix C

# Additional details and results of Chapter 5

## C.1 Bayesian inference and forecast for the LMSV model

Denote by $x=\{x_t=\log y_t^2, t=1,...,T\}$ the series of log squared returns, the LMSV model in (5.5)-(5.6) can be transformed to a stationary process with respect to $x_t$ as

$$(1-B)^d\Phi(B)z_t = \Theta(B)\eta_t,\ \ \eta_t\sim\mathcal{N}(0,\sigma_\eta^2),\ \ t=2,...,T, \tag{C.1}$$

$$x_t = c+z_t+\xi_t,\ \ \xi_t\sim(0,\sigma_\xi^2)\ \ t=1,2,...,T, \tag{C.2}$$

where $\xi_t=\log\epsilon_t^2-\mathbb{E}[\log\epsilon_t^2]$ is i.i.d with mean zero and variance $\sigma_\xi^2$, $c=\log(\kappa^2)+\mathbb{E}[\log\epsilon_t^2]$. The process $x_t$ is the sum of the long-memory ARFIMA$(p,d,q)$ process $z_t$ and a non-Gaussian noise, with $\mathbb{E}[x_t]=c$ and the auto-covariance function (ACVF)

$$\gamma_x(h)=\text{Cov}(x_t,x_{t+h})=\gamma(h)+\sigma_\xi^2\mathbb{1}_{h=0}, \tag{C.3}$$

where $h$ is the lag number, $\gamma(h)$ is the ACVF of the $z_t$ process and $\mathbb{1}_{h=0}$ is an indicator function which equals to 1 if $h=0$ and 0 otherwise. Breidt et al. (1998) estimate the LMSV model by maximizing the Whittle log-likelihood (Whittle, 1953), defined as

$$\ell_W(\beta_x)=2\pi T^{-1}\sum_{k=1}^{[T/2]}\left\{\log f_{\beta_x}(\omega_k)+\frac{J(\omega_k)}{f_{\beta_x}(\omega_k)}\right\}, \tag{C.4}$$

where $[\cdot]$ denotes the integer part, $\beta_x=(d,\phi_1,...,\phi_p,\theta_1,...,\theta_q,\sigma_\eta^2,\sigma_\xi^2,c)$ is the vector of model parameters, $\omega_k=2\pi kT^{-1}$ is the $k$th Fourier frequency, $J(\omega_k)$ is the $k$th normalized periodogram ordinate

$$J(\omega_k)=\frac{1}{2\pi T}\left(\sum_{t=1}^{T}x_t\cos\omega_k t\right)^2+\frac{1}{2\pi T}\left(\sum_{t=1}^{T}x_t\sin\omega_k t\right)^2, \tag{C.5}$$

and $f_{\beta_x}(\omega_k)$ is the spectral density of the LMSV model in (C.1)-(C.2)

$$f_{\beta_x}(\omega_k)=\frac{\sigma_\eta^2\left|\Theta(e^{-i\omega_k})\right|^2}{2\pi\left|1-e^{-i\omega_k}\right|^{2d}\left|\Phi(e^{-i\omega_k})\right|^2}+\frac{\sigma_\xi^2}{2\pi}. \tag{C.6}$$

The Whittle likelihood is an approximation of the time-domain likelihood and is exact if the data are i.i.d. Gaussian.

Let $\pi_W(\beta_x) \propto L_W(\beta_x)\, p(\beta_x)$ be the posterior density based on the Whittle likelihood $L_W(\beta_x) = \exp(\ell_W(\beta_x))$, given the log of squared return series $x = \{x_t,\, t = 1,...,T\}$. To sample from $\pi_W(\beta_x)$, we use an adaptive random walk MCMC method summarized in Algorithm C.1, with the covariance matrix in the random walk proposal adaptively scaled to target an overall acceptance probability of 25% Garthwaite et al. (2010). We note that the vector of parameters $\beta_x$ in Algorithm C.1 does not include the constant $c$ in (C.2), which is simply estimated by the sample mean, i.e., $c = \frac{1}{T}\sum_{t=1}^{T} x_t$ (Harvey, 2007).

**Algorithm C.1** Metropolis-Hasting algorithm for LMSV model

Sample $\beta_x \sim p(\beta_x)$

For each MCMC iteration:

1. Sample $\beta_x'$ from the proposal density $q(\beta_x'|\beta_x)$.
2. Compute the Whittle likelihood $L_W(\beta_x') = \exp(\mathcal{L}_w(\beta_x'))$ with $\mathcal{L}_w(\cdot)$ defined in (C.4).
3. Accept the proposal $\beta_x'$ with the probability

$$\min\left\{1, \frac{L_W(\beta_x')}{L_W(\beta_x)} \frac{p(\beta_x')}{p(\beta_x)} \frac{q(\beta_x|\beta_x')}{q(\beta_x'|\beta_x)}\right\}.$$

Given the samples of model parameters from the posterior density $\pi_W(\beta_x)$, Harvey (2007) suggests a convenient way to obtain the estimated values of conditional variance $\sigma_t^2$ for the LMSV model as follows. Suppose that $z_t$ is a stationary process and denote by $\Sigma_z$ and $\Sigma_\xi$ the covariance matrices of $z_t$ and $\xi_t$, respectively, then the covariance matrix $\Sigma$ of the log squared returns series $x$ is $\Sigma = \Sigma_z + \Sigma_\xi$. The minimum mean square linear estimator of the log volatility $\tilde{z} = \{\tilde{z}_t,\, t = 1,...,T\}$ is calculated as

$$\tilde{z} = (\mathbf{I}_T - \sigma_\xi^2 \Sigma^{-1}) x' + \sigma_\xi^2 \Sigma^{-1} \boldsymbol{\iota}, \tag{C.7}$$

where $\mathbf{I}_T$ is the identity matrix of size $T$, $\boldsymbol{\iota}$ is a $T \times 1$ column vector of ones. As $\xi_t$ are i.i.d and serially uncorrelated, the covariance matrix $\Sigma_\xi$ is $\Sigma_\xi = \sigma_\xi^2 \mathbf{I}_T$. We note that for a general ARFIMA$(p,d,q)$ process, there is no closed form for the ACVF $\gamma(h)$, and hence covariance matrix $\Sigma_z$, so approximations of $\Sigma_z$ are needed, e.g. see Sowell (1992);

Doornik and Ooms (2003). However, Hosking (1981) suggests exact ACVF for some simple cases of $(p,d,q)$ such as ARFIMA$(0,d,0)$, ARFIMA$(1,d,0)$ and ARFIMA$(0,d,1)$.

Given the estimates of $z_t$ in (C.7), the conditional variance $\sigma_t^2$ is computed as

$$\widetilde{\sigma}_t^2 = \tilde{\kappa}^2 \exp(\widetilde{z}_t),$$

where the scale factor $\tilde{\kappa}^2$ is estimated as

$$\tilde{\kappa}^2 = \frac{1}{T}\sum_{t=1}^{T} \widetilde{y}_t, \tag{C.8}$$

with $\widetilde{y}_t = y_t \exp(-\widetilde{z}_t/2)$ the heteroscredasticity corrected observations.

Denote by $R$ the $1 \times T$ the vector of covariance between $x_{T+1}$ and $x$, e.g. $R = [\gamma_x(1),...,\gamma_x(T)]$, the one-step-ahead forecast value of the log squared return is calculated as (Harvey, 2007)

$$\widehat{x}_{T+1} = c + R\Sigma^{-1}(x - c\mathbf{1}_T)$$

The one-step-ahead forecast of the conditional variance is $\widehat{\sigma}_{T+1} = \tilde{\kappa}^2 \exp(\widehat{x}_{T+1} - c)$.

Table C.1 shows the estimation results of the LMSV model using a ARFIMA$(1,d,0)$ process to model the log volatility $z_t$, with the vector of parameters $\beta_x = [d,\phi_1,\sigma_\eta^2,\sigma_\xi^2]$. We also report the estimation of the scale factor $\kappa$ in (5.6) and the constant $c$ in (C.2). For the parameters $\phi_1$ and $\sigma_\eta^2$, we choose the priors to be similar to those of the SV model as shown in Table 4.3. We use the same inverse-Gamma prior, as that of $\sigma_\eta^2$, for the parameter $\sigma_\xi^2$. For the fractional integration parameter $d$, we set the prior $2d \sim$ Beta(20,5). We run $N_{\text{MCMC}} = 100000$ MCMC iterations of Algorithm C.1 and then discard the first 10,000 iterations as burn-ins.

## C.2 Particle filter and implementation details of the DT-SMC sampler

Algorithm C.2 describes the particle filter for the SR-SV model where $\mathbf{Z}_t = (Z_t^1,...,Z_t^N)$ denotes the vector of particles at time $t$. The set of standard normal random numbers $U$ includes two sources of randomness: the set of random numbers $\{U_{t,k}^P, t = 1,...,T; k = 1,...,N\}$ used to propose new particles in each time step, and the set of random numbers $\{U_{t,k}^R, t = 1,...,T-1; k = 1,...,N\}$ used in the resampling step. For the resampling

| | $d$ | $\phi$ | $\sigma_\eta^2$ | $\sigma_\xi^2$ | $\kappa$ | $c$ |
|---|---|---|---|---|---|---|
| DAX | 0.442 | 0.708 | 0.054 | 4.933 | 1.974 | −1.616 |
| | (0.026) | (0.082) | (0.024) | (0.170) | | |
| HSI | 0.431 | 0.747 | 0.052 | 5.469 | 2.047 | −1.526 |
| | (0.030) | (0.069) | (0.021) | (0.183) | | |
| FCHI | 0.434 | 0.716 | 0.062 | 5.241 | 1.984 | −1.562 |
| | (0.029) | (0.083) | (0.029) | (0.182) | | |
| SPX | 0.428 | 0.809 | 0.040 | 5.488 | 2.017 | −1.650 |
| | (0.035) | (0.060) | (0.016) | (0.177) | | |
| TSX | 0.445 | 0.714 | 0.047 | 4.964 | 1.918 | −1.493 |
| | (0.026) | (0.084) | (0.021) | (0.160) | | |

**Table C.1.** Applications: Posterior means of the parameters of the LMSV model with the posterior standard deviations in brackets. We also report the estimation of the scale factor $\kappa$ and the constant $c$.

step, we use multinomial resampling, with sorting, to obtain the vector ancestor indexes $\{A_{t-1}^k, k = 1,...,N\}$ used to propose particles at time $t$. The sorting step helps eliminate the discontinuity issues of the selected particles in the ordinary multinomial resampling scheme (Gerber and Chopin, 2014). This sorted resampling scheme allows the selected particles to still be close after being resampled and hence helps to reduce the variability of the likelihood ratio estimator $\widehat{p}(y_{1:T}|\theta',u')/\widehat{p}(y_{1:T}|\theta,u)$ shown in the Algorithm 5.1 (Deligiannidis et al., 2018).

| Variable | Description | Value |
|---|---|---|
| $K$ | Number of annealing levels | 10000 |
| $M$ | Number of particles | 10000 |
| $N$ | Number of particles in the particle filter | 200 |
| $\rho$ | Correlation factor in the CPM algorithm | 0.999 |
| $c$ | Constant of the ESS threshold | 0.800 |
| $N_{\text{CPM}}$ | Number of CPM moves | 20 |

**Table C.2.** Implementation settings of the DT-SMC sampler.

The multinomial resampling scheme in step 2a and 2b generates the ancestor index $A_{t-1}^k, k = 1,...,N$, from the multinomial distribution denoted as $\mathcal{F}(\cdot|\mathbf{p},\mathbf{u})$ with $\mathbf{p}$ the vector of parameters of the multinomial distribution and $\mathbf{u}$ the uniform random

numbers used within a multinomial random number generator. We use the standard normal cumulative distribution function $\Psi(\cdot)$ in the resampling step to transform the normal random numbers $U^R_{t-1,k}$ to the uniform random numbers, denoted as $\overline{U}^R_{t-1,k}$.

## C.3 Additional results

This section provides the additional tables and figures discussed in Section 5.4.2.

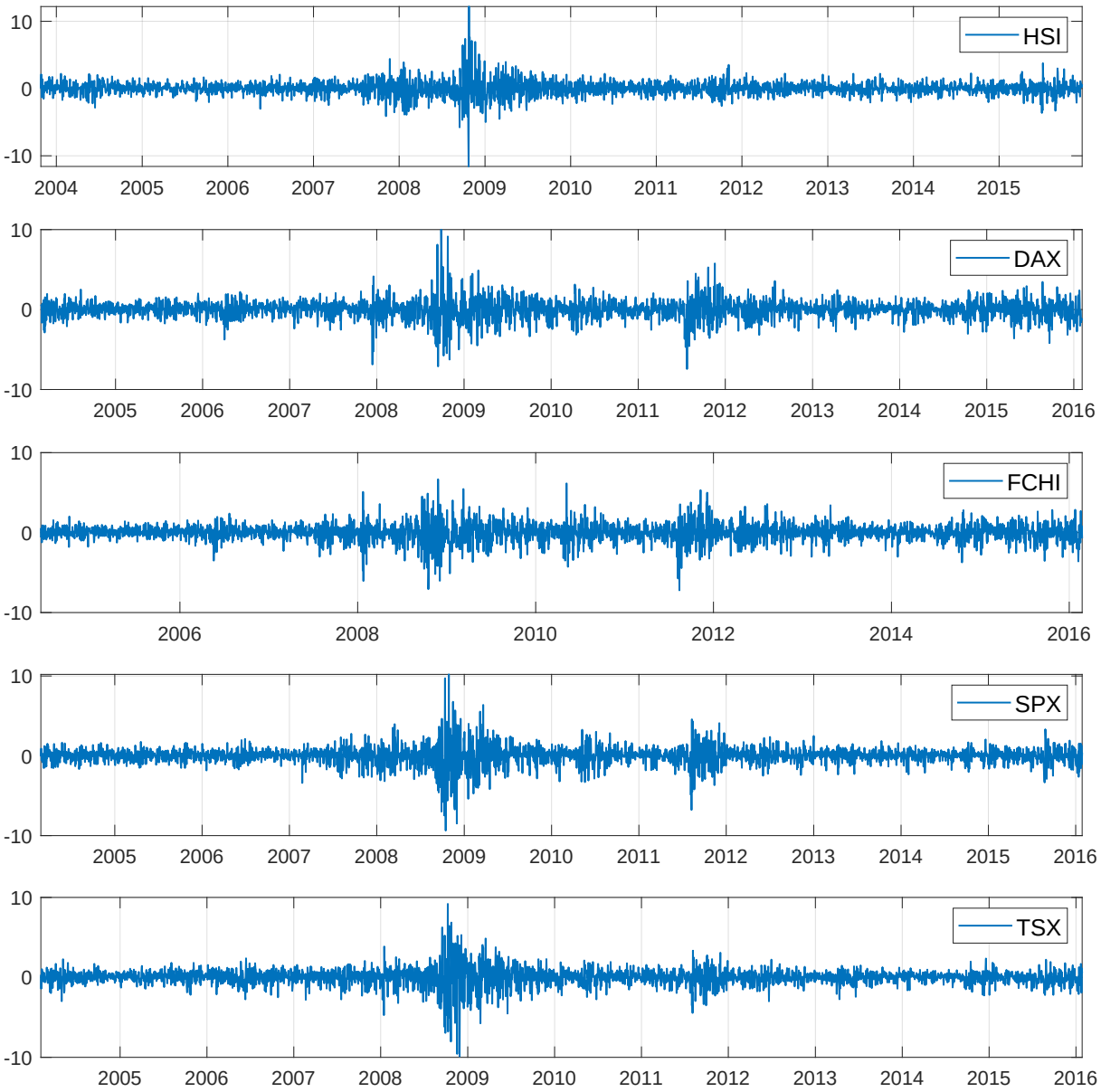


**Figure C.1.** Applications: Time series plots for the HSI, DAX, FCHI, SPX and TSX datasets.

**Algorithm C.2** Particle filter for the SR-SV model

**Input:** $T,N,y_{1:T},\theta,U=(U^P_{1,1},...,U^P_{T,N},U^R_{1,1},...,U^R_{T-1,N})$

1. At time $t=1$,

   (a) for $k=1,...N$, initialize the particles $(H^k_1,\eta^k_1,Z^k_1)$, e.g., $H^k_1=0$, as the SRU unit initially has no memory, and

   $$\begin{aligned}\eta^k_1 &= \beta_0+\sigma U^P_{1,k}\\ Z^k_1 &= \eta^k_1\end{aligned}$$

   (b) compute and normalize the weights

   $$\begin{aligned}w_1(Z^k_1) &= \frac{\mu_\theta(Z^k_1)g_\theta(y_1|Z^k_1)}{q_\theta(Z^k_1|y_1)}=g_\theta(y_1|Z^k_1)\\ W^k_1 &= \frac{w_1(Z^k_1)}{\sum_{m=1}^N w_1(Z^m_1)}\end{aligned}$$

   (c) compute the estimated likelihood $\widehat{p}(y_1|\theta)$ as

   $$\widehat{p}(y_1|\theta,U)=\frac{1}{N}\sum_{k=1}^N w_1(Z^k_1).$$

2. At times $t=2,...,T$,

   (a) sort the particle vector $\mathbf{Z}_{t-1}$ in ascending order to obtain the vector of sorted particles $\overline{\mathbf{Z}}_{t-1}=(\overline{Z}^1_{t-1},...,\overline{Z}^N_{t-1})$. The sorted index vector associated with $\overline{\mathbf{Z}}_{t-1}$ is denoted as $\mathbf{I}_{t-1}=(I^1_{t-1},...,I^N_{t-1})$. In this setting, we have the relation $\overline{Z}^k_{t-1}=Z^{I^k_{t-1}}_{t-1}$ with $k=1,...,N$. Use the sorted index vector $\mathbf{I}_{t-1}$ to define the vector of sorted weights $(\overline{W}^1_{t-1},...,\overline{W}^N_{t-1})$ such that

   $$\overline{W}^k_{t-1} = W^{I^k_{t-1}}_{t-1}$$

   (b) sample $A^k_{t-1}\sim\mathcal{F}(\cdot|\overline{W}^k_{t-1},\overline{U}^R_{t-1,k})$ where $\overline{U}^R_{t-1,k}=\Psi(U^R_{t-1,k})$ for $k=1,...,N$.

(c) for $k=1,...N$, generate particles $Z_t^k$ by

$$\begin{aligned} x_{t-1} &= [\eta_{t-1}^{A_{t-1}^k}, Z_{t-1}^{A_{t-1}^k}] \\ H_t^k &= \text{SRU}(x_{t-1}, H_{t-1}^{A_{t-1}^k}) \\ \eta_t^k &= \beta_0 + \beta_1 H_t^k + \sigma U_{t,k}^P \\ Z_t^k &= \eta_t^k + \phi Z_{t-1}^{A_{t-1}^k} \end{aligned}$$

and set $Z_{1:t}^k = (Z_{1:t-1}^{A_{t-1}^k}, Z_t^k)$.

(d) compute and normalize the weights

$$\begin{aligned} w_t(Z_{1:t}^k) &= \frac{f_\theta(Z_t^k | Z_{t-1}^{A_{t-1}^k}) g_\theta(y_t | Z_t^k)}{q_\theta(Z_t^k | y_t, Z_{t-1}^{A_{t-1}^k})} = g_\theta(y_1 | Z_1^k) \\ W_t^k &= \frac{w_t(Z_{1:t}^k)}{\sum_{m=1}^N w_t(Z_{1:t}^m)} \end{aligned}$$

(e) compute the estimated likelihood $\widehat{p}(y_t | y_{1:t-1}, \theta)$ as

$$\widehat{p}(y_t | y_{1:t-1}, \theta, U) = \frac{1}{N} \sum_{k=1}^N w_t(Z_{1:t}^k).$$

**Output:** Estimate of the likelihood

$$\widehat{p}(y_{1:T} | \theta, U) = \widehat{p}(y_1 | \theta, U) \prod_{t=2}^T \widehat{p}(y_t | y_{1:t-1}, \theta, U).$$

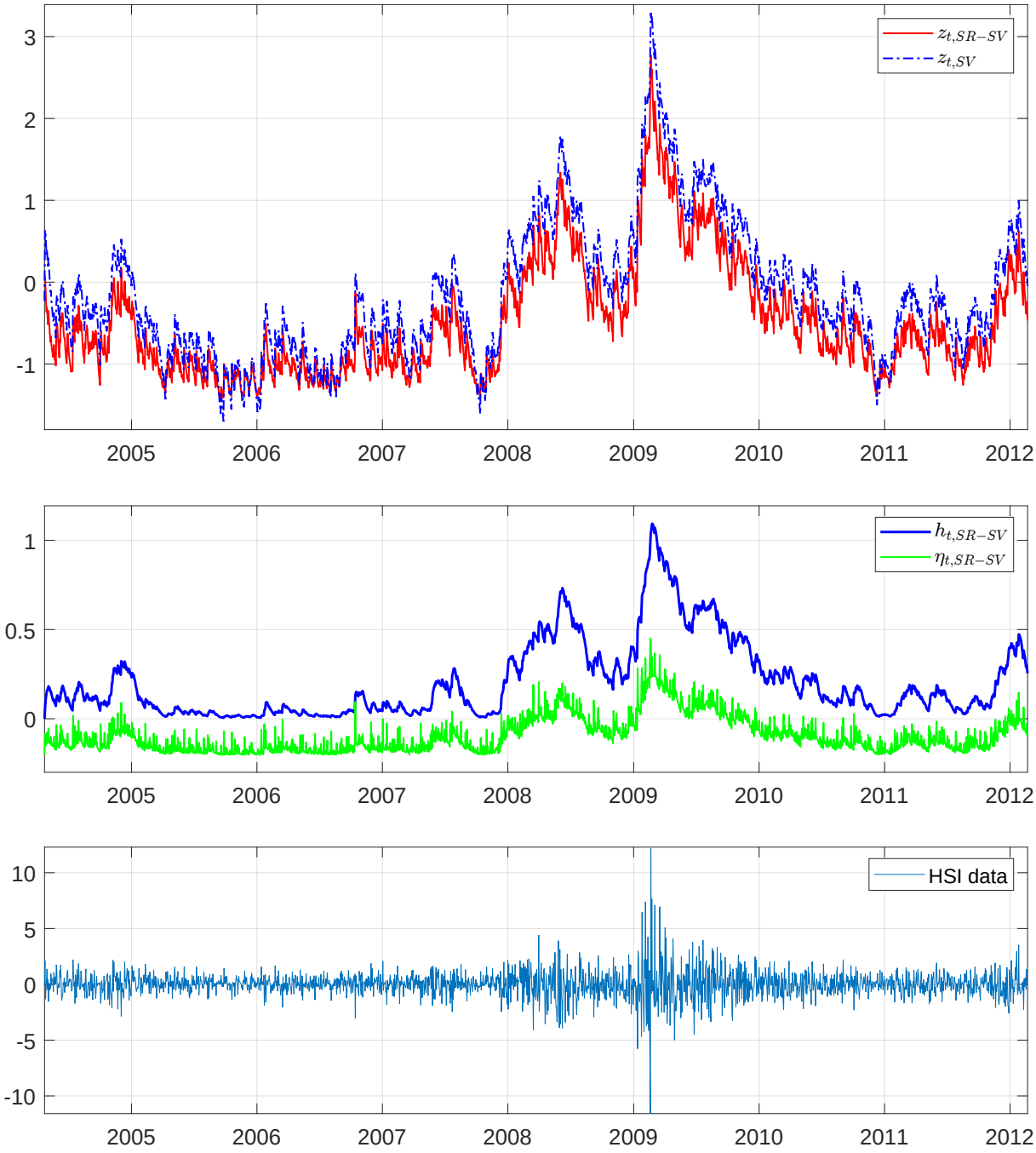


**Figure C.2.** HSI: (*Top*) The filtered log conditional variance of the SR-SV and SV models. (*Middle*) The filtered values of $\eta_t$ and $h_t$ of the SR-SV model. (*Bottom*) The in-sample data. (This is better viewed in colour).

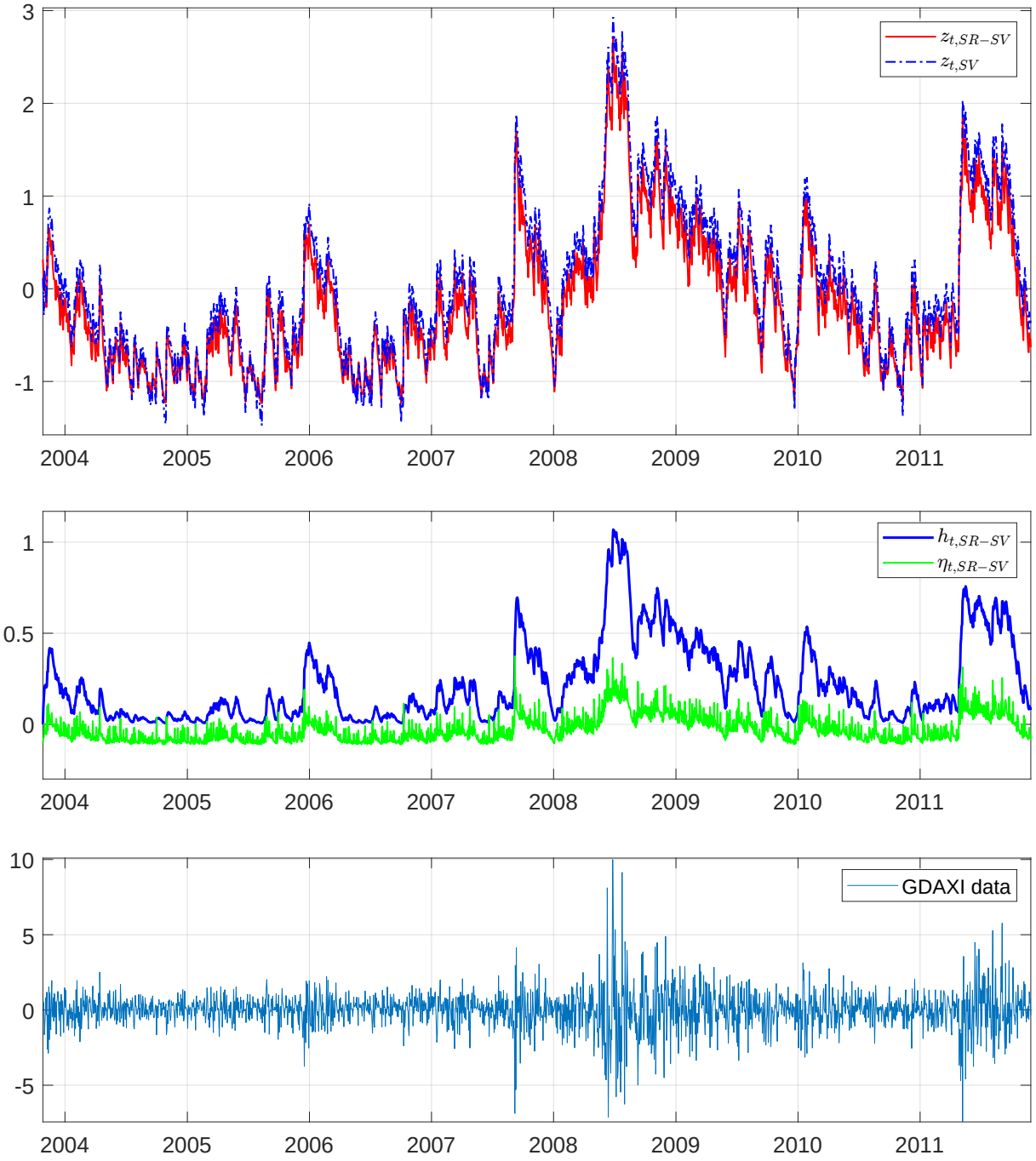


**Figure C.3.** DAX: (*Top*) The filtered log conditional variance of the SR-SV and SV models. (*Middle*) The filtered values of $\eta_t$ and $h_t$ of the SR-SV model. (*Bottom*) The in-sample data. (This is better viewed in colour).

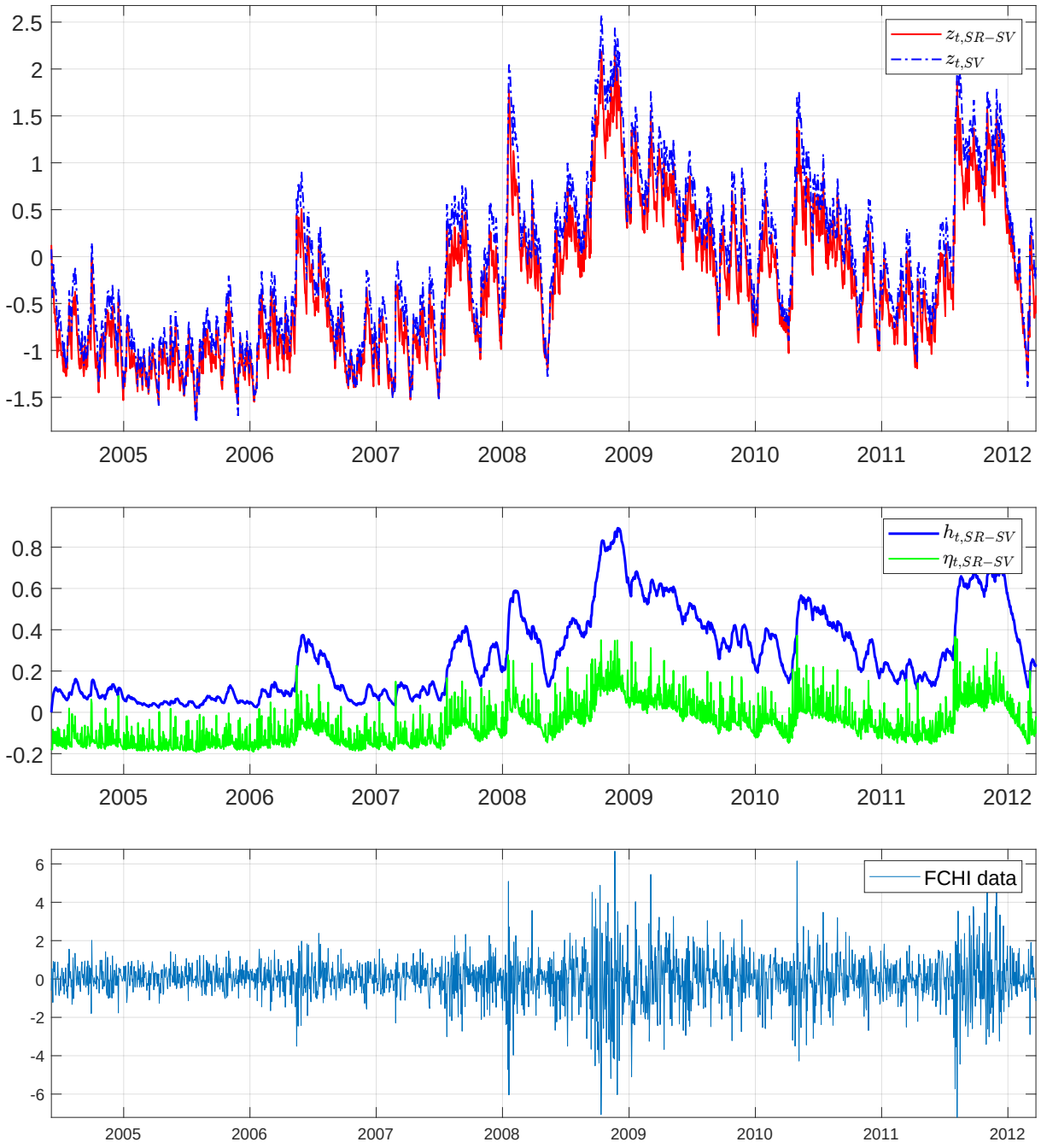


**Figure C.4.** FCHI: (*Top*) The filtered log conditional variance of the SR-SV and SV models. (*Middle*) The filtered values of $\eta_t$ and $h_t$ of the SR-SV model. (*Bottom*) The in-sample data. (This is better viewed in colour).

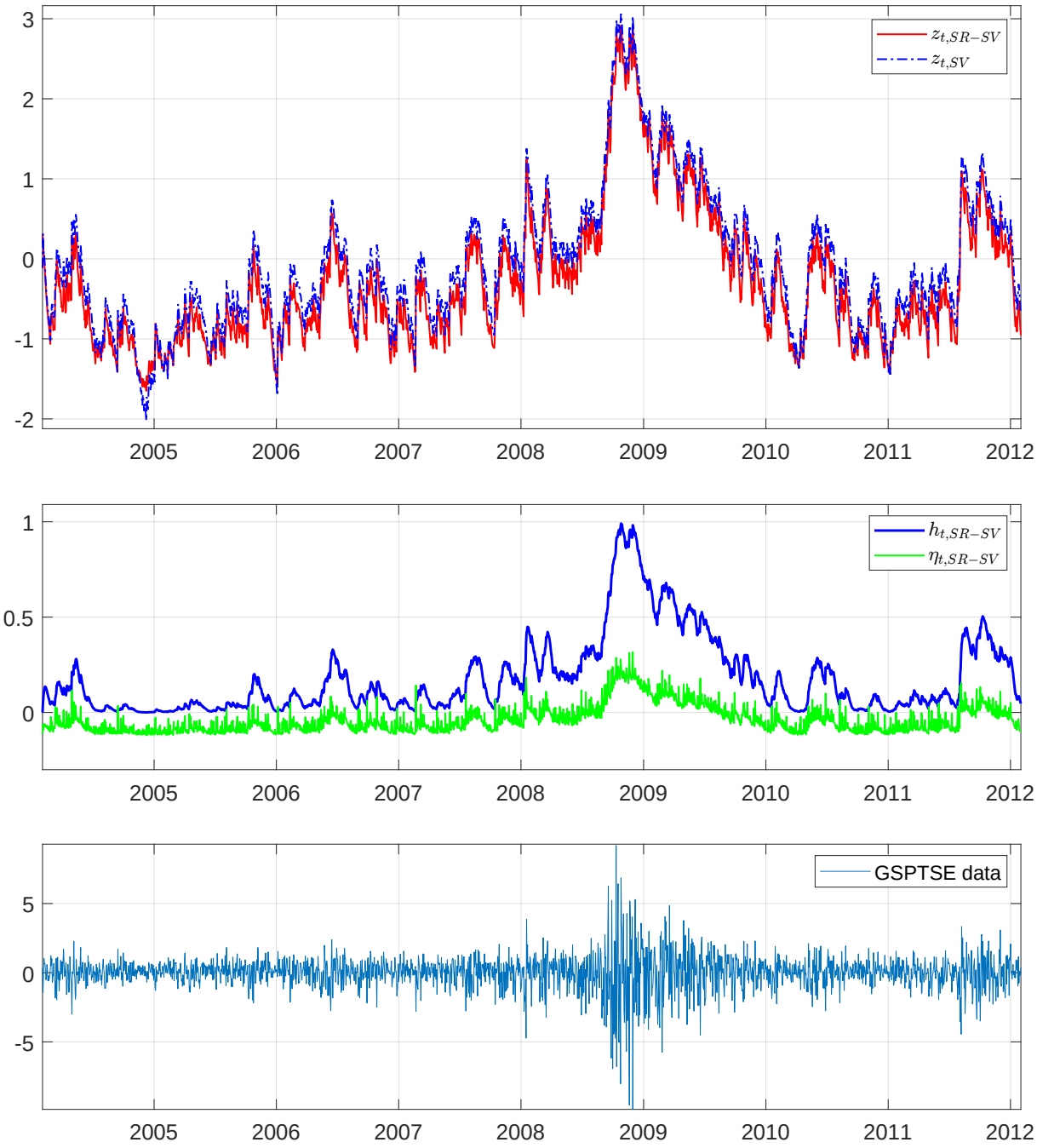


**Figure C.5.** TSX: (*Top*) The filtered log conditional variance of the SR-SV and SV models. (*Middle*) The filtered values of $\eta_t$ and $h_t$ of the SR-SV model. (*Bottom*) The in-sample data. (This is better viewed in colour).

| Measure | | PPS | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLIKE | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|---|
| | SV | 1.368 | 0.099 | 0.763 | 0.234 | 0.485 | 0.853 | 0.423 | 0 |
| | | (0.000) | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.000) | |
| BV | N-SV | 1.368 | 0.099 | 0.769 | 0.234 | 0.487 | 0.850 | 0.422 | 0 |
| | | (0.000) | (0.001) | (0.003) | (0.001) | (0.002) | (0.001) | (0.002) | |
| | LMSV | | 0.108 | 0.805 | 0.245 | 0.501 | 0.862 | 0.435 | 0 |
| | SR-SV | **1.365** | **0.09** | **0.745** | **0.220** | **0.452** | **0.847** | **0.386** | 7 |
| | | (0.000) | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | SV | | 0.094 | 0.582 | 0.237 | 0.486 | 0.810 | 0.443 | 0 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.000) | (0.001) | |
| MedRV | N-SV | | 0.094 | 0.585 | 0.238 | 0.489 | 0.810 | 0.447 | 0 |
| | | | (0.001) | (0.004) | (0.001) | (0.002) | (0.001) | (0.002) | |
| | LMSV | | 0.099 | 0.612 | 0.245 | 0.504 | 0.825 | 0.467 | 0 |
| | SR-SV | | **0.088** | **0.580** | **0.227** | **0.462** | **0.807** | **0.417** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | SV | | 0.126 | 0.994 | 0.268 | 0.558 | 0.858 | 0.581 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| RKV | N-SV | | 0.126 | 1.001 | 0.269 | 0.559 | 0.854 | 0.581 | 0 |
| | | | (0.001) | (0.003) | (0.001) | (0.001) | (0.001) | (0.002) | |
| | LMSV | | 0.135 | 1.023 | 0.279 | 0.578 | 0.878 | 0.603 | 0 |
| | SR-SV | | **0.116** | **0.974** | **0.254** | **0.516** | **0.851** | **0.538** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | SV | | 0.107 | 0.909 | 0.239 | 0.501 | 0.859 | 0.442 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| RV | N-SV | | 0.108 | 0.915 | 0.239 | 0.503 | 0.855 | 0.441 | 0 |
| | | | (0.001) | (0.003) | (0.001) | (0.002) | (0.001) | (0.002) | |
| | LMSV | | 0.115 | 0.936 | 0.250 | 0.527 | 0.875 | 0.464 | 0 |
| | SR-SV | | **0.098** | **0.889** | **0.223** | **0.462** | **0.851** | **0.400** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |

**Table C.3.** DAX data: Forecast performance of the SR-SV and benchmark models using different realized measures. In each panel, the bold numbers indicate the best predictive scores.

| Measure | | PPS | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLIKE | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|---|
| | SV | 1.131 | 0.069 | **0.497** | 0.186 | 0.319 | 0.359 | 0.390 | 1 |
| | | (0.000) | (0.000) | (0.001) | (0.000) | (0.001) | (0.001) | (0.001) | |
| BV | N-SV | 1.130 | 0.067 | 0.499 | 0.182 | 0.313 | 0.357 | 0.376 | 0 |
| | | (0.000) | (0.000) | (0.001) | (0.000) | (0.001) | (0.001) | (0.002) | |
| | LMSV | | 0.076 | 0.516 | 0.205 | 0.343 | 0.370 | 0.423 | 0 |
| | SR-SV | **1.127** | **0.060** | 0.504 | **0.152** | **0.261** | **0.355** | **0.294** | 5 |
| | | (0.000) | (0.00) | (0.002) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | SV | | 0.066 | 0.371 | 0.191 | 0.317 | 0.347 | 0.435 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |
| MedRV | N-SV | | 0.065 | **0.370** | 0.188 | 0.312 | 0.344 | 0.423 | 1 |
| | | | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | (0.002) | |
| | LMSV | | 0.073 | 0.396 | 0.207 | 0.335 | 0.356 | 0.469 | 0 |
| | SR-SV | | **0.059** | 0.389 | **0.164** | **0.272** | **0.341** | **0.338** | 5 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | SV | | 0.100 | **0.740** | 0.230 | 0.385 | 0.366 | 0.665 | 1 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.001) | (0.001) | |
| RKV | N-SV | | 0.098 | 0.741 | 0.226 | 0.380 | 0.364 | 0.648 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.001) | (0.000) | (0.002) | |
| | LMSV | | 0.112 | 0.764 | 0.246 | 0.405 | 0.380 | 0.744 | 0 |
| | SR-SV | | **0.087** | 0.748 | **0.194** | **0.323** | **0.360** | **0.519** | 5 |
| | | | (0.000) | (0.002) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | SV | | 0.069 | **0.522** | 0.186 | 0.318 | 0.367 | 0.390 | 1 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.001) | (0.001) | |
| RV | N-SV | | 0.068 | 0.524 | 0.181 | 0.311 | 0.365 | 0.374 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.001) | (0.002) | |
| | LMSV | | 0.077 | 0.552 | 0.204 | 0.353 | 0.386 | 0.419 | 0 |
| | SR-SV | | **0.060** | 0.530 | **0.150** | **0.258** | **0.361** | **0.291** | 5 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.001) | (0.001) | |

**Table C.4.** HSI data: Forecast performance of the SR-SV and benchmark models using different realized measures. In each panel, the bold numbers indicate the best predictive scores.

| Measure | | PPS | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLIKE | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|---|
| | SV | 1.384 | 0.108 | 1.076 | 0.235 | 0.504 | 0.863 | 0.426 | 0 |
| | | (0.000) | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |
| BV | N-SV | 1.383 | 0.108 | 1.086 | 0.234 | 0.507 | 0.860 | 0.420 | 0 |
| | | (0.000) | (0.000) | (0.002) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | LMSV | | 0.118 | 1.104 | 0.246 | 0.527 | 0.872 | 0.469 | 0 |
| | SR-SV | **1.381** | **0.095** | **1.057** | **0.210** | **0.448** | **0.856** | **0.354** | 7 |
| | | (0.000) | (0.00) | (0.001) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | SV | | 0.100 | 0.670 | 0.238 | 0.500 | 0.833 | 0.543 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.001) | (0.002) | |
| MedRV | N-SV | | 0.100 | 0.672 | 0.237 | 0.501 | 0.832 | 0.538 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.000) | (0.000) | (0.001) | |
| | LMSV | | 0.112 | 0.695 | 0.247 | 0.513 | 0.849 | 0.582 | 0 |
| | SR-SV | | **0.090** | **0.665** | **0.216** | **0.452** | **0.828** | **0.472** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | SV | | 0.158 | 1.271 | 0.301 | 0.624 | 0.901 | 0.750 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.002) | |
| RKV | N-SV | | 0.159 | 1.285 | 0.301 | 0.628 | 0.896 | 0.745 | 0 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | LMSV | | 0.168 | 1.332 | 0.315 | 0.656 | 0.908 | 0.815 | 0 |
| | SR-SV | | **0.139** | **1.229** | **0.275** | **0.562** | **0.890** | **0.645** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | SV | | 0.103 | 0.908 | 0.232 | 0.495 | 0.877 | 0.411 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| RV | N-SV | | 0.103 | 0.920 | 0.232 | 0.498 | 0.873 | 0.404 | 0 |
| | | | (0.000) | (0.002) | (0.000) | (0.001) | (0.000) | (0.001) | |
| | LMSV | | 0.113 | 0.945 | 0.245 | 0.521 | 0.884 | 0.449 | 0 |
| | SR-SV | | **0.090** | **0.880** | **0.209** | **0.440** | **0.869** | **0.340** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |

**Table C.5.** FCHI data: Forecast performance of the SR-SV and benchmark models using different realized measures. In each panel, the bold numbers indicate the best predictive scores and the model with highest count of best predictive scores is preferred.

| Measure | | PPS | $MSE_1$ | $MSE_2$ | $MAE_1$ | $MAE_2$ | QLIKE | $R^2LOG$ | Count |
|---|---|---|---|---|---|---|---|---|---|
| | SV | 1.004 | 0.074 | 0.904 | 0.192 | 0.295 | 0.106 | 0.542 | 0 |
| | | (0.001) | (0.000) | (0.002) | (0.001) | (0.001) | (0.001) | (0.003) | |
| BV | N-SV | 1.003 | 0.074 | 0.902 | 0.192 | 0.294 | **0.105** | 0.543 | 1 |
| | | (0.000) | (0.000) | (0.002) | (0.001) | (0.001) | (0.001) | (0.002) | |
| | LMSV | | 0.091 | 0.965 | 0.217 | 0.331 | 0.197 | 0.676 | 0 |
| | SR-SV | **1.001** | **0.069** | **0.900** | **0.181** | **0.273** | 0.107 | **0.517** | 6 |
| | | (0.000) | (0.00) | (0.001) | (0.000) | (0.001) | (0.001) | (0.001) | |
| | SV | | 0.074 | 0.289 | 0.210 | 0.310 | 0.098 | 1.052 | 0 |
| | | | (0.000) | (0.001) | (0.001) | (0.001) | (0.001) | (0.002) | |
| MedRV | N-SV | | 0.073 | 0.291 | 0.210 | 0.309 | 0.098 | 1.044 | 0 |
| | | | (0.000) | (0.002) | (0.001) | (0.000) | (0.001) | (0.001) | |
| | LMSV | | 0.096 | 0.360 | 0.239 | 0.353 | 0.209 | 0.839 | 0 |
| | SR-SV | | **0.069** | **0.287** | **0.201** | **0.291** | 0.098 | **0.985** | 5 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.001) | (0.001) | |
| | SV | | 0.096 | 0.349 | 0.246 | 0.357 | 0.134 | 0.987 | 0 |
| | | | (0.000) | (0.001) | (0.000) | (0.001) | (0.000) | (0.002) | |
| RKV | N-SV | | 0.096 | 0.347 | 0.246 | 0.355 | 0.134 | 0.991 | 0 |
| | | | (0.000) | (0.001) | (0.001) | (0.000) | (0.001) | (0.001) | |
| | LMSV | | 0.110 | 0.392 | 0.263 | 0.378 | 0.220 | 1.120 | 0 |
| | SR-SV | | **0.089** | **0.341** | **0.236** | **0.336** | **0.131** | **0.951** | 6 |
| | | | (0.000) | (0.001) | (0.000) | (0.000) | (0.000) | (0.001) | |
| | SV | | 0.087 | 1.370 | 0.206 | 0.319 | 0.118 | 0.625 | 0 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.001) | (0.002) | |
| RV | N-SV | | 0.087 | 1.368 | 0.206 | 0.317 | **0.117** | 0.627 | 1 |
| | | | (0.000) | (0.002) | (0.000) | (0.001) | (0.001) | (0.002) | |
| | LMSV | | 0.100 | 1.418 | 0.224 | 0.342 | 0.195 | 0.742 | 0 |
| | SR-SV | | **0.081** | **1.363** | **0.195** | **0.295** | 0.119 | **0.597** | 5 |
| | | | (0.000) | (0.002) | (0.000) | (0.000) | (0.001) | (0.001) | |

**Table C.6.** TSX data: Forecast performance of the SR-SV and benchmark models using different realized measures. In each panel, the bold numbers indicate the best predictive scores.

# Bibliography


Amari, S. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276. (Cited on pages 27, 34, and 35.)

Andersen, T. G. and Bollerslev, T. (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review*, 39(4):885–905. (Cited on page 78.)

Andersen, T. G., Bollerslev, T., and Diebold, F. X. (2010). Chapter 2 - parametric and nonparametric volatility measurement. In AÏT-SAHALIA, Y. and HANSEN, L. P., editors, *Handbook of Financial Econometrics: Tools and Techniques*, volume 1 of *Handbooks in Finance*, pages 67–137. North-Holland, San Diego. (Cited on page 62.)

Andersen, T. G., Dobrev, D., and Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1):75 – 93. Recent Advances in Panel Data, Nonlinear and Nonparametric Models: A Festschrift in Honor of Peter C.B. Phillips. (Cited on page 78.)

Andrews, D. F. and Mallows, C. L. (1974). Scale mixtures of normal distributions. *Journal of the Royal Statistical Society, Series B*, 36:99–102. (Cited on page 39.)

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, 72:1–33. (Cited on pages 98, 101, and 107.)

Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient monte carlo computations. *The Annals of Statistics*, 37(2):697–725. (Cited on pages 18 and 101.)

Ardia, D. and Hoogerheide, L. F. (2010). Bayesian estimation of the GARCH(1,1) model with student-t innovations. *The R Journal*, 2(2):41–47. (Cited on page 143.)

Baillie, R. T., Bollerslev, T., and Mikkelsen, H. O. (1996). Fractionally integrated generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 74(1):3 – 30. (Cited on pages 75 and 111.)

Baltagi (2013). *Econometric Analysis of Panel Data, 5th Edition*. Wiley. (Cited on page 53.)

Barndorff-Nielsen, O., Hansen, P., Lunde, A., and Shephard, N. (2008). Designing realized kernels to measure the ex post variation of equity prices in the presence of noise. *Econometrica*, 76(6):1481–1536. cited By 491. (Cited on page 78.)

Barndorff-Nielsen, O. E. and Shephard, N. (2004). Power and Bipower Variation with Stochastic Volatility and Jumps. *Journal of Financial Econometrics*, 2(1):1–37. (Cited on page 78.)

Bartholomew, D. J., Knott, M., and Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach, 3rd edition*. John Wiley & Sons. (Cited on pages 27 and 36.)

Bekaert, G., Engstrom, E., and Ermolov, A. (2015). Bad environments, good environments: A non-Gaussian asymmetric volatility model. *Journal of Econometrics*, 186(1):258 – 275. (Cited on page 133.)

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In G., M., G.B., O., and KR., M., editors, *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, volume 7700. Springer, Berlin, Heidelberg. (Cited on page 50.)

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer. (Cited on page 33.)

Black, F. (1976). Studies of stock price volatility changes. In *Proceedings of the 1976 Meeting of the Business and Economic Statistics Section, American Statistical Association*, pages 177–181. (Cited on page 56.)

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877. (Cited on page 33.)

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307 – 327. (Cited on pages 55, 56, and 59.)

Bollerslev, T. (2008). Glossary to ARCH (GARCH). Creates research papers, Department of Economics and Business Economics, Aarhus University. (Cited on page 60.)

Bollerslev, T. and Mikkelsen, H. O. (1996). Modeling and pricing long memory in stock market volatility. *Journal of Econometrics*, 73(1):151 – 184. (Cited on page 98.)

Box, G. E. P. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243. (Cited on page 111.)

Box, G. E. P. and Jenkins, G. (1976). *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA. (Cited on pages 3 and 12.)

Breidt, F., Crato, N., and de Lima, P. (1998). The detection and estimation of long memory in stochastic volatility. *Journal of Econometrics*, 83(1):325 – 348. (Cited on pages 77, 99, 102, 108, 119, and 147.)

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97:465–480. (Cited on page 137.)

Casella, G. (2001). Empirical Bayes Gibbs sampling. *Biostatistics*, 2:485–500. (Cited on page 39.)

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). Bart: Bayesian additive regression trees. *Ann. Appl. Stat.*, 4(1):266–298. (Cited on pages 28 and 44.)

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics. (Cited on page 16.)

Chopin, N. (2002). A sequential particle filter method for static models. *Biometrika*, 89(3):539–551. (Cited on pages 19 and 21.)

Cornwell, C. and Rupert, P. (1988). Efficient estimation with panel data: An empirical comparison of instrumental variable estimators. *Journal of Applied Econometrics*, 3(2):149–155. (Cited on page 52.)

Crato, N. and de Lima, P. J. (1994). Long-range dependence in the conditional variance of stock returns. *Economics Letters*, 45(3):281 – 285. (Cited on page 98.)

Del Moral, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, New York. (Cited on page 106.)

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B*, 68:411–436. (Cited on pages 19, 107, and 108.)

Deligiannidis, G., Doucet, A., and Pitt, M. K. (2018). The correlated pseudo marginal method. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):839–870. (Cited on pages 101, 108, and 150.)

Diebold, F. X. (2015). Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of diebold–mariano tests. *Journal of Business & Economic Statistics*, 33(1):1–1. (Cited on page 69.)

Dieng, A. B., Ranganath, R., Altosaar, J., and Blei, D. M. (2018). Noisin: Unbiased regularization for recurrent neural networks. (Cited on page 105.)

Ding, Z., Granger, C. W., and Engle, R. F. (1993). A long memory property of stock market returns and a new model. *Journal of Empirical Finance*, 1(1):83 – 106. (Cited on page 98.)

Donaldson, R. G. and Kamstra, M. (1997). An artificial neural network-garch model for international stock return volatility. *Journal of Empirical Finance*, 4(1):17–46. (Cited on pages 58, 62, 86, and 125.)

Doornik, J. A. and Ooms, M. (2003). Computational aspects of maximum likelihood estimation of autoregressive fractionally integrated moving average models. *Computational Statistics and Data Analysis*, 42(3):333 – 348. Computational Ecomometrics. (Cited on page 149.)

Douc, R., M. E. . S. D. (2013). *Nonlinear Time Series: Theory, Methods and Applications with R Examples, 1st*. Chapman and Hall/CRC, England. (Cited on page 32.)

Duan, J.-C. and Fulop, A. (2015). Density-tempered marginalized Sequential Monte Carlo samplers. *Journal of Business & Economic Statistics*, 33(2):192–202. (Cited on pages 101 and 107.)

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–21. (Cited on pages 13 and 14.)

Engle, R. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics*, 20(3):339–350. (Cited on page 133.)

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987–1007. (Cited on page 56.)

Engle, R. F. and Rangel, J. G. (2008). The Spline-GARCH Model for Low-Frequency Volatility and Its Global Macroeconomic Causes. *The Review of Financial Studies*, 21(3):1187–1222. (Cited on page 62.)

Fan, K., Wang, Z., Beck, J., Kwok, J., and Heller, K. (2015). Fast second-order stochastic backpropagation for variational inference. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1387–1395. Curran Associates, Inc. (Cited on page 29.)

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230. (Cited on page 103.)

Fleming, J., Kirby, C., and Ostdiek, B. (2003). The economic value of volatility timing using "realized" volatility. *Journal of Financial Economics*, 67(3):473 – 509. (Cited on page 79.)

Gardner, E. S. (2006). Exponential smoothing: The state of the art—part ii. *International Journal of Forecasting*, 22(4):637–666. (Cited on page 3.)

Gardner Jr., E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–28. (Cited on page 3.)

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. A. (2018a). Conditional neural processes. (Cited on page 132.)

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., and Teh, Y. W. (2018b). Neural processes. (Cited on page 132.)

Garthwaite, P., Fan, Y., and Sisson, S. (2010). Adaptive optimal scaling of Metropolis-Hastings algorithms using the Robbins-Monro process. *Communications in Statistics - Theory and Methods*, 45. (Cited on page 148.)

Gerber, M. and Chopin, N. (2014). Sequential Quasi-Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77. (Cited on page 150.)

Geweke, J. and Zhou, G. (1996). Measuring the pricing error of the arbitrage pricing theory. *Review of Financial Studies*, 9(2):557–587. (Cited on page 36.)

Giraitis, L., Kokoszka, P., Leipus, R., and Teyssière, G. (2003). Rescaled variance and related tests for long memory in volatility and levels. *Journal of Econometrics*, 112(2):265 – 294. (Cited on pages 77 and 119.)

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR*, volume 9, pages 249–256. (Cited on page 42.)

Glosten, L. R., Jagannathan, R., and Runkle, D. E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, 48(5):1779–1801. (Cited on pages 55, 58, and 60.)

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378. (Cited on page 68.)

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. (Cited on pages 12, 14, 16, 32, 42, 57, 62, and 105.)

Granger, C. W. J. and Joyeux, R. (1980). An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, 1(1):15–29. (Cited on pages 99 and 102.)

Gunawan, D., Dang, K., Quiroz, M., Kohn, R., and Tran, M. (2018). Subsampling sequential Monte Carlo for static Bayesian models. *To appear in Statistics and Computing*. (Cited on pages 19 and 108.)

Gunawan, D., Tran, M.-N., and Kohn, R. (2017). Fast inference for intractable likelihood problems using variational Bayes. Technical report. arXiv:1705.06679. (Cited on page 136.)

Han, S., Liao, X., Dunson, D. B., and Carin, L. C. (2016). Variational Gaussian copula inference. In Gretton, A. and Robert, C. C., editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 829–838, Cadiz, Spain. JMLR Workshop and Conference Proceedings. (Cited on page 34.)

Hansen, P. R. and Lunde, A. (2005). A forecast comparison of volatility models: does anything beat a GARCH(1,1)? *Journal of Applied Econometrics*, 20(7):873–889. (Cited on pages 57, 68, 79, and 143.)

Härdle, W. K. and Turlach, B. A. (1992). Nonparametric approaches to generalized linear models. In Fahrmeir, L., Francis, B., Gilchrist, R., and Tutz, G., editors, *Advances in GLIM and Statistical Modelling*, pages 213–225, New York, NY. Springer New York. (Cited on page 32.)

Harvey, A. C. (2007). 16 - long memory in stochastic volatility. In Knight, J. and Satchell, S., editors, *Forecasting Volatility in the Financial Markets (Third Edition)*, Quantitative Finance, pages 351 – 363. Butterworth-Heinemann, Oxford. (Cited on pages 148 and 149.)

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109. (Cited on page 18.)

Higgins, M. L. and Bera, A. K. (1992). A class of nonlinear ARCH models. *International Economic Review*, 33(1):137–158. (Cited on page 111.)

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80. (Cited on pages 16 and 64.)

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347. (Cited on page 35.)

Honkela, A., Raiko, T., Kuusela, M., Tornio, M., and Karhunen, J. (2010). Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes. *Journal of Machine Learning Research*, 11:3235–3268. (Cited on page 35.)

Hosking, J. R. M. (1981). Fractional differencing. *Biometrika*, 68(1):165–176. (Cited on pages 102 and 149.)

Hosszejni, D. and Kastner, G. (2020). Modeling univariate and multivariate stochastic volatility in R with stochvol and factorstochvol. *Journal of Statistical Software*. (Cited on page 144.)

Jacquier, E., Polson, N. G., and Rossi, P. E. (1994). Bayesian analysis of stochastic volatility models (with discussion). *Journal of Business and Economic Statistics*, 12:371–417. (Cited on page 102.)

Jank, W. (2011). *Business Analytics for Managers*. Springer-Verlag New York. (Cited on pages 48 and 49.)

Jeffreys, H. (1935). Some tests of significance, treated by the theory of probability. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31(2):203–222. (Cited on page 23.)

Jeffreys, H. (1961). *Theory of Probability, 3rd*. Clarendon Press, Oxford, England. (Cited on page 23.)

Jensen, M. J. and Maheu, J. M. (2010). Bayesian semiparametric stochastic volatility modeling. *Journal of Econometrics*, 157(2):306 – 316. (Cited on page 103.)

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998). Markov Chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100. (Cited on page 19.)

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795. (Cited on page 23.)

Kiliç, R. (2011). Long memory and nonlinearity in conditional variances: A smooth transition FIGARCH model. *Journal of Empirical Finance*, 18(2):368 – 378. (Cited on page 99.)

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. (2019). Attentive neural processes. (Cited on page 132.)

Kim, H. Y. and Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple garch-type models. *Expert Systems with Applications*, 103:25 – 37. (Cited on pages 58 and 100.)

Kim, S., Shephard, N., and Chib, S. (1998). Stochastic volatility: likelihood inference and comparison with ARCH models. *Review of Economic Studies*, 65:361–393. (Cited on pages 98, 102, 105, and 108.)

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. Technical report. arXiv:1312.6114v10. (Cited on pages 28 and 34.)

Koo, B. and Linton, O. (2015). Let's get lade: Robust estimation of semiparametric multiplicative volatility models. *Econometric Theory*, 31(4):671–702. (Cited on page 62.)

Koopman, S. J., Lucas, A., and Scharth, M. (2016). Predicting time-varying parameters with parameter-driven and observation-driven models. *The Review of Economics and Statistics*, 98(1):97–110. (Cited on pages 57 and 62.)

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017). Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45. (Cited on page 29.)

Kyung, M., Gill, J., Ghosh, M., and Casella, G. (2010). Penalized regression, standard errors and Bayesian Lassos. *Bayesian Statistics*, 5:369–412. (Cited on pages 27, 38, and 39.)

Lai, T. L., Shih, M.-C., and Wong, S. P. (2006). A new approach to modeling covariate effects and individualization in population pharmacokinetics-pharmacodynamics. *Journal of Pharmacokinetics and Pharmacodynamics*, 33(1):49–74. (Cited on page 26.)

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941. (Cited on pages 12 and 63.)

Leng, C., Tran, M.-N., and Nott, D. J. (2014). Bayesian adaptive lasso. *The Annals of the Institute of Statistical Mathematics*, 66:221–44. (Cited on pages 27 and 38.)

Li, D., Clements, A., and Drovandi, C. (2021). Efficient bayesian estimation for garch-type models via sequential monte carlo. *Econometrics and Statistics*, 19:22–46. (Cited on page 67.)

Linton, O. B. (2009). *Semiparametric and Nonparametric ARCH Modeling*, pages 157–167. Springer Berlin Heidelberg, Berlin, Heidelberg. (Cited on page 62.)

Lipton, Z., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. arXiv:1804.04359. (Cited on pages 57 and 100.)

Liu, J. S. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044. (Cited on page 19.)

Liu, Y. (2019). Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. *Expert Systems with Applications*, 132:99–109. (Cited on page 58.)

Lo, A. W. (1991). Long-term memory in stock market prices. *Econometrica*, 59(5):1279–1313. (Cited on pages 77, 98, and 119.)

Luo, R., Zhang, W., Xu, X., and Wan, J. (2018). A neural stochastic volatility model. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. (Cited on page 58.)

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018a). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13. (Cited on page 3.)

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018b). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3):1–26. (Cited on page 100.)

Mandelbrot, B. (1967). The variation of some other speculative prices. *The Journal of Business*, 40(4):393–413. (Cited on page 56.)

Martens, J. (2010). Deep learning via Hessian-free optimization. In *27th International Conference on Machine Learning*, Haifa, Israel. (Cited on page 29.)

Martens, J. (2014). New insights and perspectives on the natural gradient method. Technical report. arXiv:1412.1193. (Cited on page 29.)

Martens, M. (2002). Measuring and forecasting S&P 500 index-futures volatility using high-frequency data. *Journal of Futures Markets*, 22(6):497–518. (Cited on page 79.)

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. Chapman and Hall/CRC. (Cited on page 30.)

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092. (Cited on page 18.)

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress. (Cited on pages 12 and 63.)

Neal, R. (2001). Annealed importance sampling. *Statistics and Computing*, 11:125–139. (Cited on pages 19, 107, and 108.)

Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, 59(2):347–370. (Cited on pages 55, 60, and 98.)

Nguyen, N., Tran, M.-N., Gunawan, D., and Kohn, R. (2019). A long short-term memory stochastic volatility model. *arXiv e-prints*, page arXiv:1906.02884. (Cited on pages 57 and 62.)

Oliva, J. B., Póczos, B., and Schneider, J. G. (2017). The statistical recurrent unit. In *ICML2017*. (Cited on pages 16, 101, and 105.)

Ong, V. M.-H., Nott, D. J., and Smith, M. S. (2017a). Gaussian variational approximation with factor covariance structure. *Journal of Computational and Graphical Statistics*, To Appear. (Cited on pages 28, 35, 36, 137, and 138.)

Ong, V. M.-H., Nott, D. J., Tran, M.-N., Sisson, S., and Drovandi, C. (2018). Variational Bayes with synthetic likelihood. *Statistics and Computing*, To appear. (Cited on pages 41 and 43.)

Ong, V. M.-H., Nott, D. J., Tran, M.-N., Sisson, S. A., and Drovandi, C. C. (2017b). Likelihood-free inference in high dimensions with synthetic likelihood. Technical report, Queensland University of Tehcnology, https://eprints.qut.edu.au/112213/. (Cited on pages 28 and 37.)

Pagan, A. R. and Schwert, G. (1990). Alternative models for conditional stock volatility. *Journal of Econometrics*, 45(1):267 – 290. (Cited on pages 86 and 125.)

Pascanu, R. and Bengio, Y. (2014). Revisiting natural gradient for deep networks. Technical report. arXiv:1301.3584v7. (Cited on pages 29 and 35.)

Pitt, M. K., dos Santos Silva, R., Giordani, P., and Kohn, R. (2012). On some properties of Markov chain Monte Carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134 – 151. Bayesian Models, Methods and Applications. (Cited on page 107.)

Polson, N. G. and Sokolov, V. O. (2017). Deep learning: A Bayesian perspective. Technical report. arXiv:1706.00473v1. (Cited on page 28.)

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5). (Cited on page 41.)

Poole, B., Sohl-Dickstein, J., and Ganguli, S. (2014). Analyzing noise in autoencoders and deep networks. (Cited on page 105.)

Poon, S.-H. and Granger, C. W. (2003). Forecasting volatility in financial markets: A review. *Journal of Economic Literature*, 41(2):478–539. (Cited on page 60.)

Ranganath, R., Wang, C., Blei, D. M., and Xing, E. P. (2013). An adaptive learning rate for stochastic variational inference. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 298–306. (Cited on pages 41 and 43.)

Rao, C. R. (1945). Information and accuracy attainable in the estimation of statistical parameters. *Bull. Calcutta. Math. Soc*, 37:81–91. (Cited on page 35.)

Regier, J., Jordan, M. I., and McAuliffe, J. (2017). Fast black-box variational inference through stochastic trust-region optimization. *arXiv preprint arXiv:1706.02375*. (Cited on page 29.)

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286. (Cited on pages 28 and 34.)

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407. (Cited on page 33.)

Roberts, G. O. and Rosenthal, J. S. (2009). Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349–367. (Cited on page 19.)

Roeder, G., Wu, Y., and Duvenaud, D. (2017). Sticking the landing: Simple, lower-variance gradient estimators for variational inference. *arXiv preprint arXiv:1703.09194*. (Cited on pages 28 and 34.)

Roh, T. H. (2007). Forecasting the volatility of stock price index. *Expert Systems with Applications*, 33(4):916 – 922. (Cited on page 58.)

Salimans, T. and Knowles, D. A. (2013). Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):741–908. (Cited on page 35.)

Sato, M. (2001). Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681. (Cited on page 35.)

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117. (Cited on page 12.)

Shephard, N. and Sheppard, K. (2010). Realising the future: forecasting with high-frequency-based volatility (heavy) models. *Journal of Applied Econometrics*, 25(2):197–231. (Cited on pages 78 and 79.)

Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67 – 79. (Cited on page 105.)

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85. M4 Competition. (Cited on page 3.)

Sowell, F. (1992). Maximum likelihood estimation of stationary univariate fractionally integrated time series models. *Journal of Econometrics*, 53(1):165 – 188. (Cited on page 148.)

Stoer, J. (1983). Solution of large linear systems of equations by conjugate gradient type methods. In Bachem, A., Korte, B., and Grötschel, M., editors, *Mathematical Programming The State of the Art: Bonn 1982*, pages 540–565. Springer Berlin Heidelberg, Berlin, Heidelberg. (Cited on page 37.)

Stroup, W. W. (2012). *Generalized Linear Mixed Models: Modern Concepts, Methods and Applications*. Chapman and Hall/CRC. (Cited on page 31.)

Tan, L. S. L. and Nott, D. J. (2017). Gaussian variational approximation with sparse precision matrices. *Statisics and Computing*, To Appear. (Cited on page 34.)

Taylor, J. W. (2019). Forecasting value at risk and expected shortfall using a semiparametric approach based on the asymmetric laplace distribution. *Journal of Business & Economic Statistics*, 37(1):121–133. (Cited on page 68.)

Taylor, S. (1986). *Modelling Financial Time Series*. John Wiley, Chichester. (Cited on pages 56 and 98.)

Taylor, S. J. (1982). Financial returns modelled by the product of two stochastic processes — a study of daily sugar prices 1961-79. In Anderson, O. D., editor, *Time Series Analysis: Theory and Practice*, page 203–226. Amsterdam: North-Holland. (Cited on pages 98, 99, and 101.)

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979. (Cited on page 29.)

Tran, M., Nott, D., and Kohn, R. (2017). Variational Bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 26(4):873–882. (Cited on pages 28, 32, 35, and 41.)

Tran, M.-N., Kohn, R., Quiroz, M., and Villani, M. (2016a). Block-wise pseudo-marginal Metropolis-Hastings. Technical report. arXiv:1603.02485. (Cited on page 32.)

Tran, M.-N., Nguyen, N., Nott, D., and Kohn, R. (2020). Bayesian deep net GLM and GLMM. *Journal of Computational and Graphical Statistics*. (Cited on page 6.)

Tran, M.-N., Nguyen, T.-N., and Dao, V.-H. (2021). A practical tutorial on variational bayes. (Cited on page 23.)

Tran, M.-N., Nott, D. J., Kuk, A. Y., and Kohn, R. (2016b). Parallel variational Bayes for large datasets with an application to generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 26:626–646. (Cited on page 40.)

Trippe, B. L. and Turner, R. E. (2018). Overpruning in variational Bayesian neural networks. arXiv:1801.06230v1. (Cited on page 38.)

van Bellegem, S. (2012). Locally stationary volatility modeling. In Bauwens, L., Hafner, C., and Laurent, S., editors, *Volatility Models and Their Applications*. Wiley & Sons. (Cited on pages 65 and 105.)

Whittle, P. (1953). Estimation and information in stationary time series. *Ark. Mat.*, 2(5):423–434. (Cited on page 147.)

Yu, J. (2002). Forecasting volatility in the New Zealand stock market. *Applied Financial Economics*, 12(3):193–202. (Cited on page 98.)

Yu, J., Yang, Z., and Zhang, X. (2006). A class of nonlinear stochastic volatility models and its implications for pricing currency options. *Computational Statistics and Data Analysis*, 51(4):2218 – 2231. (Cited on pages 99, 102, and 108.)

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67. (Cited on page 39.)

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. Technical report. arXiv:1212.5701. (Cited on page 43.)

Zhang, G. (2003a). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159 – 175. (Cited on page 57.)

Zhang, G. (2003b). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159 – 175. (Cited on page 100.)

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35 – 62. (Cited on page 57.)

Zhang, J., Lin, Y., Song, Z., and Dhillon, I. S. (2018). Learning long term dependencies via Fourier recurrent units. In *ICML2017*. (Cited on page 133.)

# Biography

**Trong Nghia Nguyen** received his Bachelor degree in mechatronical engineering from The Ho Chi Minh University of Technology, Vietnam, in 2012. He then obtained a Master degree in Computer Science at Kookmin University, South Korea, in 2015, under the supervision of Professor Gu-Min Jeong.

In 2018, he was awarded an post-graduate scholarship for international students, under the International Postgraduate Research Scholarship (IPRS) scheme, to pursue his doctoral degree at the University of Sydney Business School under the supervision of Associate Professor Minh-Ngoc Tran and Scientia Professor Robert Kohn (UNSW Business School). His research interests include computational statistics, machine learning and financial econometrics modeling. Trong Nghia is also a graduate student member of Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS).

Trong Nghia Nguyen
nghia.nguyen@sydney.edu.au