Credit Exposure Modelling Using Differential Machine Learning

Måns Karp, Samuel Wagner

Master's thesis
2023:E47

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Abstract

Exposure modelling is a critical aspect of managing counterparty credit risk, and banks worldwide invest significant time and computational resources in this task. One approach to modelling exposure involves pricing trades with a counterparty in numerous potential future market scenarios. Suitable for this type of pricing is a framework presented in 2020 by Huge and Savine, which they call differential machine learning. It approximates the pricing function with a neural network that trains on Monte Carlo paths and the gradients along these paths. This thesis aims to demonstrate the application of differential machine learning in the context of exposure modelling. To better comply with this context, training is done on market variables, rather than some hidden model state. Simulated data is used from Heston type models to estimate the future exposure distribution of a portfolio consisting of European options. The conducted experiments reveal that training the machine learning model on market observables yields similar results to those obtained when training on hidden model states. Furthermore, the exposure modelling approach is subject to stress testing by evaluation of its performance under different levels of compatibility between the pricing model and future market scenarios in which the portfolio is priced. Results show that low compatibility leads to decreased accuracy of the predicted exposure distributions.

Keywords: Counterparty credit risk, Differential machine learning, Exposure modelling, Heston model, Option pricing.

Acknowledgements

Firstly, we would like to thank our industry supervisors at Danske Bank, Copenhagen, Brian Fuglsbjerg and Jesper Christiansen, for their time and patience throughout the project. Furthermore, we would like to thank our academic supervisors Erik Lindström and Carl Lindberg for insightful feedback and guidance. Lastly, we would like to express our gratitude towards our two examiners, Annika Lang at Chalmers University of Technology and Magnus Wiktorsson at the Faculty of Engineering at Lund University for allowing us to write this thesis together, despite coming from different universities.

Samuel Wagner & Måns Karp, June 2023

Contents

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order.

AAD	Automatic Adjoint Differentiation
ANN	Artificial Neural Network
ATM	At the Money, i.e. option with moneyness $m = 1$
CCR	Counterparty Credit Risk
DML	Differential Machine Learning
EE	Expected Exposure
FX	Foreign Exchange
GBP	Pound Sterling
MSE	Mean Squared Error
MtM	Mark-to-Market
PCA	Principal Component Analysis
PFE	Potential Future Exposure
ReLU	Rectifier Linear Unit
RMSE	Root Mean Squared Error
USD	US Dollar

Nomenclature

Below is the nomenclature of parameters and variables that have been used throughout this thesis.

Heston model variables and parameters

S_t	Spot asset price at time t
ν_t	Stochastic variance at time t
μ	Drift of spot asset price process
κ	Mean reversion rate of variance process
θ	Long-term mean of variance process
ξ	Volatility of variance process
ρ	Correlation between spot asset price and variance processes
$\tilde{\rho}_{ij}$	Correlation between spot asset price processes of asset i and j

Variables and Parameters

K	Option strike price
m	Option moneyness
T	Option time of expiry
τ	Option time to expiry, $\tau = T - t$
P	Number of observed option prices
N	Number of modelled assets
N_T	Number of simulated time steps in each scenario
M	Number of simulated market scenarios
ϵ_1	Threshold of eigenvalues in traditional PCA
ϵ_2	Threshold of eigenvalues in differential PCA
$c^i_{m,\tau}(t)$	Value of European call option on asset i with moneyness m and time to expiry τ at time t

List of Figures

List of Tables

1

Introduction

This chapter begins by providing a brief background to the topic of this thesis in section 1.1. We then proceed to specify the objective in general terms in section 1.2, followed by a study of previous work on related topics in section 1.3. The chapter is then concluded by a procedural overview of the thesis as a whole in section 1.4.

1.1 Background

A central part of banking activities is managing risk. Risks come in many shapes and forms, and a great deal of time and effort is put into computing, quantifying and assessing these everyday by banks all over the world. In 1988 eleven countries known as the Group of Ten (G-10), there amongst the United States, the United Kingdom, Germany, France, and Switzerland, agreed to the "Basel Capital Accord" [3], also known as "Basel I", aiming to regulate banks' risk appetite by restricting the amount of acquired *credit risk*, namely the risk of e.g. a private person or a company, also known as a *counterparty*, not being able to pay what is owed. The Basel framework was subsequently adopted by virtually all countries with active international banks. The framework has since been updated twice, the most recent of which, "Basel III" [2], followed after the 2007-09 financial crisis. Basel III introduced stricter requirements on regulatory capital, i.e. how much capital a bank needs to hold at any given time.

The quantity of regulatory capital legally required to be held by any bank is dependent on the bank's credit risk, which in turn is a risk whose magnitude is determined by two primary factors. The first factor is the probability that a given counterparty defaults before completing the last transaction. However, in this thesis, we will focus on the second factor, which is the *credit exposure*, namely the size of what is owed to the bank [11]. For example, in a conventional loan with a fixed interest rate and predetermined instalments, credit exposure can be estimated at any future time in a straightforward manner. However, a bank and a counterparty may enter into contracts, the values of which are dependent on the market price of some underlying asset, or assets, say a given stock price or a foreign exchange (FX) rate. In such a setting, credit exposure estimation may be more complicated. In figure 1.1, we find a schematic illustration of how credit exposure estimation may be performed, where the flow from generation of market scenarios to exposure distribution estimation found in the top half presents a traditional approach. In this case, potential market scenarios are generated in the order of tens of thousands, and then, for each market scenario, a pricing model is calibrated and used to price all contracts between

the bank and its counterparties. Although this approach may result in reliable exposure distribution estimations, calibration of pricing models is a computationally expensive, and thus time-consuming, endeavour. However, in this thesis, a framework called *differential machine learning* (DML) [16] will be employed, which is illustrated in the bottom half of figure 1.1. In the given setting, the method may be utilised to produce swift estimations. As seen in the illustration, the otherwise computationally expensive model calibration in the traditional approach is replaced by additional market scenario generation, which in turn may be time-consuming as well. However, the trained DML model can then be used to rapidly price contracts for each market scenario.

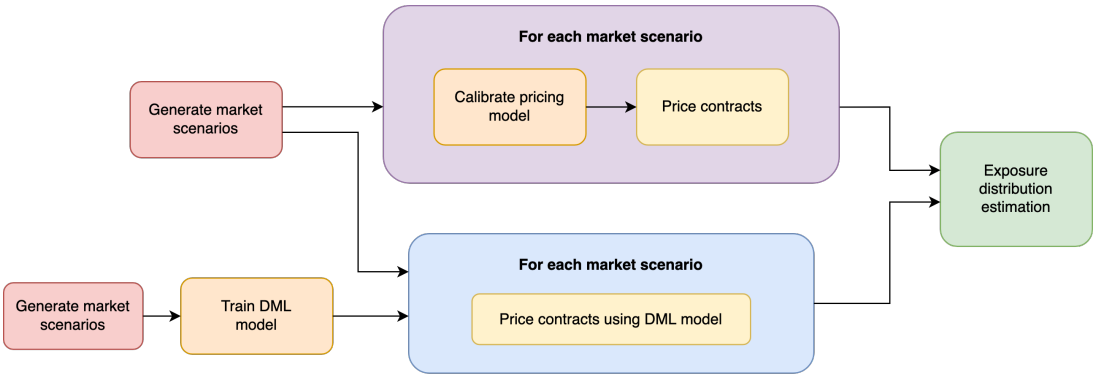


Figure 1.1: Schematic illustration of conventional credit exposure estimation, and credit exposure estimation using a DML approach.

Furthermore, in the application of the DML method, we use *market observables*, i.e. observed market prices of instruments dependent on the underlying asset being modelled, in order to promptly provide information on the driving forces in the spot price process. To maintain authenticity of the data, the instruments used as market observables are those typically traded in the market.

1.2 Objective

The objective of this project is to provide insights into how the DML framework can be used for counterparty credit risk applications, and more specifically how it can be used for the purpose of exposure modelling. In doing so, the effectiveness of using market observables, previously defined in section 1.1, will be investigated. We aim to show that the methodology can be efficiently applied in exposure modelling of a set of *European put options* and *European call options*. The reason for limitation to sets of European options is that these are the most common and relevant instruments to model using the suggested framework. Furthermore, we wish to investigate circumstances under which the framework is less suitable or yields less accurate results.

1.3 Previous work

Exposure modelling for risk management poses a great problem to practitioners because of the high computational cost. Various methods to reduce the computational cost has been developed, for example in Glau et al. [9] and Stein [25]. On a more general level, exposure modelling is concerned with two major research areas: model calibration and derivative pricing. The application of machine learning in these areas is a topic which has recently been under research. The main advantage of machine learning approaches is that they may easily be generalised and may reduce the computational cost of various procedures by utilising a pre-trained prediction model. An example of this can be found in Horvath et al. [14].

Calibration of a model is the task of finding parameter values such that the model produces values close to the observed market data. This is in general solved by minimisation of a cost function that measures the discrepancy between the values produced by the model and the observed market data, a minimisation procedure which can be computationally expensive. Hernandez [12] introduces a machine learning approach where mapping from market data to model parameters is approximated by a pre-trained neural network. Horvath et al. [14] suggests to instead first approximate the pricing function, i.e. map parameters and variables to derivatives prices. This fast pricing function approximation is then used to speed up the calibration procedure. Several applications of *feedforward artificial neural networks* (ANN) as pricing function approximators have been presented, for example in Liu et al. [20] and McGhee [23].

This thesis will, to a large extent, build on the work presented in Huge and Savine [16], where a fast method to approximate a pricing function together with its gradient by an extension of the classical feedforward ANN. The speed of the training of the method enables it to be trained online, just prior to application, in contrast to previously discussed works which use a pre-trained model. This is achieved by leveraging gradients of labels with respect to inputs in the training of the network. These gradients are cheaply produced by using *automatic adjoint differentiation* (AAD), which is a computerised algorithm that retrieves the gradient of a function by sequential application of the chain rule [24].

1.4 Procedural overview

The thesis will begin in a relatively simple setting to progressively introduce more complicated models and netting sets. Throughout the thesis, artificially generated data is used, the details of which may be found in section 3.3 and 3.4. We will only use data and instruments which would normally be observable in the market, whilst maintaining the benefit of knowing the underlying model and model parameters governing the dynamics in the artificial data, allowing for a more intricate analysis of the results.

Firstly, we perform exposure modelling of a netting set consisting of instruments with a single underlying asset to prove functionality of the market observables methodology in this relatively simple setting. This is done by comparing results using different levels of information, first giving the prediction model information about *hidden states*, i.e. data

which cannot be seen in the market, to then proceed to only give data which can be observed in the market. Having access to the two results allows for analysis of the ability of market observables to capture hidden dynamics. Thereafter, we perform exposure modelling of portfolio consisting of multiple assets. Once again, results using different levels of information are compared to investigate the accuracy of using market observable data. Lastly, we stress test the general framework by altering model dynamics in an underlying asset to investigate circumstances under which the methodology is potentially less accurate.

Python is the programming language used for all code implementations in this thesis. Data generation and manipulation are performed using the `NumPy`, `SciPy`, and `AlgoPy` packages. Furthermore, neural network models are implemented using the `TensorFlow` package. Lastly, the `Matplotlib` and `seaborn` packages are used for data visualisation purposes. For further implementation details, the complete code is available on GitHub.[†]

[†] `https://github.com/swagnerutb/mastersthesis`

2

Theory

This chapter presents underlying theory and related research relevant to the topic of this thesis. We begin by introducing exposure modelling in section 2.1, along with relevant metrics and bank regulation. We then proceed to define the Heston model, along with a multi-asset generalisation of the model in sections 2.2 and 2.3. Thereafter, machine learning methods are discussed, starting with a short review of feedforward artificial neural networks (ANN) in section 2.4.1. Central to this thesis is an extension of the feedforward ANN architecture presented in Huge and Savine [16], which they call differential machine learning, discussed in section 2.4.2. The chapter ends with section 2.4.3 containing a presentation of the data preprocessing technique *differential principal component analysis* (PCA), also introduced in Huge and Savine.

2.1 Exposure Modelling

All concepts, definitions, and theory presented in this section build on Gregory [11]. We begin by defining *credit risk*, which is a general concept defined as the risk that a counterparty will not fulfil contractual obligations, or in other words, default. The subcategory *counterparty credit risk* (CCR) is the credit risk between derivatives counterparties. Most often, at least one of the counterparties is a bank, and the other might be another bank or a large corporation. Assessing CCR is a difficult task and demands answers to two general questions. Firstly, what are the potential losses in the event of the counterparty defaulting? Secondly, what is the probability of such an event? This thesis will only concern the former question. Furthermore, CCR is relevant in different settings, such as in pricing CCR in different valuation adjustments (xVA), or for risk management purposes. The setting one considers affects how CCR should be examined. An example of risk management applications is to decide the amount of capital a bank must hold because of its CCR. This thesis will treat CCR for risk management purposes.

Having described the general area of interest, we will define some central concepts relating to CCR. The *Mark-to-Market* (MtM) of a trade is the cost of entering an identical trade with another counterparty, ignoring any potential transaction costs. MtM can identically be viewed as the net present value of all future transactions with the counterparty, i.e. the difference between future payments by and future payments to the counterparty. These transactions may very well depend heavily on *market risk factors*, for example interest rates, FX rates, implied volatility surfaces, etc. Furthermore, a *netting agreement* is a legal contract between two parties allowing aggregation of trades in the event of

default of a party. Given that there is a netting agreement with the counterparty, the MtM with respect to this counterparty is the sum of the MtM of all the trades with the counterparty. Throughout this thesis it is assumed that there is a netting agreement with the counterparty and that the trades can be treated together in what is called a *netting set*. The value of the netting set is referred to as the MtM with respect to the counterparty and is defined as

$$\text{MtM} = \sum_{i=1}^{B} \text{MtM}_i,$$

where MtM_i is the MtM of trade i, and B is the total number of trades with the counterparty. If the MtM with respect to the counterparty is positive and the counterparty defaults with zero recovery rate, referring to the percentage of outstanding transactions retrieved from a defaulting counterparty, the MtM equals the loss. Throughout this thesis a recovery rate of zero is assumed. However, if the MtM is negative, meaning that the counterparty is owed money, and the counterparty defaults, the loss is zero. This is because in the event of a defaulting counterparty, one still has to fulfil contractual obligations. This characterisation of a minimum loss of zero in the event of default leads us to the concept of exposure. *Exposure* with respect to a counterparty is defined as

$$\text{E} = \max\{\text{MtM}, 0\} = \text{MtM}^+,$$

where MtM is with respect to the counterparty. In reality, exposure is only relevant in the event of default, as the amount of exposure with respect to a counterparty becomes an actual loss first when that counterparty defaults. However, exposure in this thesis will be considered independently of the event of default.

Exposure clearly varies with time and stochastic future values. Since default of a counterparty may occur at any time point, it is of interest to model the distribution of the exposure at different time points in the future. Hence, E_t refers to the exposure at time point t. If t is in the future, E_t is a random variable. These time points are referred to as *exposure dates*. The modelling of exposure can be done in many different ways with varying degree of approximation. In this thesis, the general framework of Pykhtin and Zhu [28] is followed:

1. For exposure dates of interest $t = T_1, \ldots, T_{N_E}$, generate market scenarios $i = 1, \ldots, M$ of the relevant market risk factors $\mathbf{x}_{it}$.
2. For all scenarios $i = 1, \ldots, M$ and exposure dates $t = T_1, \ldots, T_{N_E}$, price the netting set to obtain MtM_{it}.
3. For each exposure date $t = T_1, \ldots, T_{N_E}$, the empirical distribution formed by the set $\{\text{MtM}_{it}^+ : i = 1, \ldots, M\}$ serves as an approximate distribution of the exposure E_t.

As this thesis considers CCR and exposure modelling for risk management purposes, the scenarios generated in step 1 above should come from a scenario generating model with real-world probabilities, not some risk neutral pricing model. This is important, as risk metrics, such as expected exposure computed under a risk neutral measure of today can be manipulated by changing numéraire, leading to very different results [25]. In Pykhtin and Zhu, the pricing in step 2 above should be done according to a risk neutral pricing model calibrated to the specific values of the market risk factors in that scenario $\mathbf{x}_{it}$. However, a different approach to this step will be presented in section 3.1.

2.1.1 Basel Regulation

Exposure modelling for risk management purposes is an exceedingly important practice by banks. For example, it is needed to calculate economic and regulatory capital the bank needs to hold. If the exposure modelling approach proposed in this thesis would be used in a bank for these purposes it would fall under chapter CRE53 in the Basel Framework: "Internal models method for counterparty credit risk" [2]. The framework stipulates that it is up to the bank to choose the distribution of E_t, under certain conditions. This means that the bank could use the risk neutral measure in calculating exposure in this context, according to regulation.

2.1.2 Exposure Metrics

The exposure modelling approach previously described in section 2.1 approximates the entire exposure distribution by an empirical distribution. Often, it is only specific metrics from this distribution that is used for various purposes. Note that since we consider exposure modelling for risk management purposes, the following measures are computed under the real-world measure $\mathbb{P}$ [25]. An important metric is the *expected exposure* at a exposure date t and is defined as

$$\mathrm{EE}_t = \mathbb{E}_{\mathbb{P}}[\mathrm{E}_t].$$

Other metrics used for risk management purposes include various percentiles of the distribution. The metric *potential future exposure* (PFE) at a confidence level of α at time t is defined as

$$\mathrm{PFE}_t^{\alpha} = \inf\{x \in \mathbb{R} \mid \mathbb{P}(\mathrm{E}_t \leq x) \geq \alpha\}.$$

In other words, the exposure at time t does not exceed the PFE_t^{α} with a probability of α. By convention, exposure date t is given in years, but to facilitate notation, the exposure date will be given in months, as, for example, $t = 6M$ denoting $t = 0.5$ years.

2.2 Heston Model

We begin by modelling a netting set with a single underlying asset. In order to generate data and to be able to price options, we assume the price process of an underlying asset to be described by a given model. In a single asset setting, we will assume the underlying spot asset price process to be described by the Heston model, introduced in Heston [13], where it is applied to bond options and FX options. The Heston model is a *stochastic volatility model*, i.e. both the evolution of the spot asset price and its volatility are assumed to be stochastic processes [13]. More specifically, the Heston model is defined by the stochastic differential equations

$$\begin{aligned} \mathrm{d}S(t) &= \mu S(t)\mathrm{d}t + \sqrt{\nu(t)}S(t)\mathrm{d}W^S(t) \\ \mathrm{d}\nu(t) &= \kappa\left(\theta - \nu(t)\right)\mathrm{d}t + \xi\sqrt{\nu(t)}\mathrm{d}W^{\nu}(t) \end{aligned} \tag{2.1}$$

for $t > 0$ and $S(0), \nu(0) > 0$, where $S(t)$ is the spot asset price, $\nu(t)$ is the stochastic variance, both at time t, and $W^S(t)$, $W^{\nu}(t)$ are Wiener processes with correlation ρ.

Furthermore, μ describes the drift of $S(t)$, θ is the long-term mean of $\nu(t)$, κ is the mean reversion rate of $\nu(t)$, and ξ is the volatility of $\nu(t)$. Observing the diffusion of $\nu(t)$, assuming $\nu(0) > 0$, we see that it is guaranteed to be non-negative. However, if the *Feller condition* given by

$$2\kappa\theta > \xi^2 \tag{2.2}$$

is fulfilled, the process is guaranteed to be strictly positive [6].

2.2.1 Heston Call Price Formula

Given the Heston model, there exists an expression for the price at time t of a European call option with strike price K and time of expiry T. The formula takes advantage of the known characteristic function of the logarithm of the spot asset price from (2.1), which, as seen in [1], is given by

$$\begin{aligned}
\phi(u,t,T) &= \mathbb{E}\left[\exp\left(\mathrm{i}u\log(S(T))\right) \mid S(t),\nu(t)\right] \\
&= \exp\left(\mathrm{i}u\left(\log S(t) + \mu\tau\right)\right) \\
&\times \exp\left(\theta\kappa\xi^{-2}\left((\kappa - \mathrm{i}\rho\theta u - d)\tau - 2\log\left(\frac{1-g\mathrm{e}^{-d\tau}}{1-g}\right)\right)\right) \\
&\times \exp\left(\nu(t)\xi^{-2}\left(\kappa - \mathrm{i}\rho\xi u - d\right)\frac{1-\mathrm{e}^{-d\tau}}{1-g\mathrm{e}^{-d\tau}}\right), \\
d &= \sqrt{\left(\mathrm{i}\rho\xi u - \kappa\right)^2 + \xi^2\left(\mathrm{i}u + u^2\right)}, \\
g &= \frac{\kappa - \mathrm{i}\rho\xi u - d}{\kappa - \mathrm{i}\rho\xi u + d}, \\
\tau &= T - t,
\end{aligned} \tag{2.3}$$

where parameters are defined as seen in (2.1). The formula of the European call price in the Heston model is then given by

$$C(S(t),\nu(t),K,t,T) = S(t) - \frac{\sqrt{S(t)K}\ e^{-\mu\tau/2}}{\pi} I(S(t),\nu(t),K,t,T) \tag{2.4}$$

where

$$I(S(t),\nu(t),K,t,T) = \int_0^\infty \Re\left[e^{\mathrm{i}u\left(\log\left(\frac{S(t)}{K}\right)+\mu\tau\right)}\phi(u-\mathrm{i}\frac{1}{2},t,T)\right]\frac{1}{u^2+\frac{1}{4}}\mathrm{d}u$$

as seen in [19]. The evaluation of the formula requires a numerical computation of the integral.

2.3 Multi-Asset Heston Model

When modelling multiple assets with correlated spot asset prices, such as foreign exchange (FX) pairs in a Heston model setting, the Heston model seen in (2.1) can be extended to

the *multi-asset Heston model* [7]. The multi-asset Heston model for N assets is defined by

$$\begin{aligned} \mathrm{d}S_i(t) &= \mu_i S_i(t)\mathrm{d}t + \sqrt{\nu_i(t)}S_i(t)\mathrm{d}W_i^S(t) \\ \mathrm{d}\nu_i(t) &= \kappa_i\left(\theta_i - \nu_i(t)\right)\mathrm{d}t + \xi_i\sqrt{\nu_i(t)}\mathrm{d}W_i^\nu(t) \end{aligned} \tag{2.5}$$

for $t > 0$ and $S_i(0), \nu_i(0) > 0$, where $S_i(t)$ is the spot asset price, and $\nu_i(t)$ is the instantaneous variance for asset $i = 1, \ldots, N$ at time t. Furthermore, $W_i^S(t)$, $W_i^\nu(t)$ are Wiener processes with correlation ρ_i. In cases of correlated spot prices, $W_i^S(t)$ and $W_j^S(t)$, for $i \neq j$, have correlation $\tilde{\rho}_{ij} \in [-1, 1]$. Similarly to (2.1), μ_i is the drift of $S_i(t)$, θ_i is the long-term mean of $\nu_i(t)$, κ_i is the mean reversion rate of $\nu_i(t)$, and ξ_i is the volatility of $\nu_i(t)$.

We notice that for $N = 1$, the multi-asset Heston model is equivalent to the Heston model described in section 2.2. Moreover, the Heston call price formula, described in section 2.2.1, may be used to price European call options on assets modelled by the multi-asset Heston model individually. Similarly, the Feller condition, as described for the Heston model in (2.2), generalises to the multi-asset Heston model for each asset $i = 1, \ldots, N$ as well.

2.4 Neural Networks

2.4.1 Feedforward Artificial Neural Network

We now proceed to a brief review of feedforward artificial neural networks (ANN), where all concepts and definitions are based on Goodfellow [10]. A feedforward ANN, also called multilayer perceptron, is a machine learning model that attempts to approximate a function $f : \mathbb{R}^p \to \mathbb{R}$, where inputs $\mathbf{x} \in \mathbb{R}^{1\times p}$ and labels $y \in \mathbb{R}$ could be virtually any kind of numerical data. The network architecture is built up by an input layer of p nodes, $L-1$ *hidden layers* of potentially varying number of nodes, and an output layer of 1 node. A feedforward ANN is defined by the feedforward equations

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{x} \\ \mathbf{z}_l &= \mathbf{g}_{l-1}(\mathbf{z}_{l-1})\mathbf{W}_l + \mathbf{b}_l, \quad l = 1, \ldots, L \\ y &= z_L \end{aligned} \tag{2.6}$$

which describes a series of activation functions, $\mathbf{g}_l(\cdot)$, acting elementwise, such as the softplus function defined in (2.7), as well as weights, $\mathbf{W}_l$, and biases, $\mathbf{b}_l$.

$$(\mathbf{g}_l(\mathbf{x}))_i = \ln(1 + \mathrm{e}^{\mathbf{x}_i}) \tag{2.7}$$

In order to train the network, i.e. find optimal values for the weights and biases, we need a *training set* of data, consisting of M sample inputs $\mathbf{x}$ and label y^* combinations, or simply M *examples*. The sample inputs are gathered as rows in matrix $\mathbf{X} \in \mathbb{R}^{M\times p}$ and sample labels are collected in column vector $\mathbf{y}^* \in \mathbb{R}^M$. A cost function $C(\mathbf{y}, \mathbf{y}^*)$, where $\mathbf{y}$ are network predictions given inputs $\mathbf{X}$, is then used to quantify the error in the model on the training set. Optimal parameter values in the model are then found by

$$\{\mathbf{W}_l^*, b_l^* \mid l = 1, \ldots, L\} = \underset{\{\mathbf{W}_l, \mathbf{b}_l \mid l=1,\ldots,L\}}{\operatorname{argmin}} C(\mathbf{y},\mathbf{y}^*).$$

For example, the cost function can be chosen as the mean squared error (MSE), i.e.

$$C(\mathbf{y}, \mathbf{y}^*) = \text{MSE}(\mathbf{y}, \mathbf{y}^*) = \frac{1}{M}\sum_{i=1}^{M}(y_i - y_i^*)^2.$$

Minimising C is generally done by some form of iterative method that uses the gradient of C with respect to the weights and biases. Many methods approximate the gradient by iteratively considering different subsets of the training set when forming the MSE, to then use the gradient of this instead. Such a subset is called a *batch*. An important hyperparameter in connection to this is the number of *epochs*, which is how many times a single sample is used in a gradient approximation.

2.4.2 Differential Machine Learning

The *differential machine learning* (DML) framework, introduced in Huge and Savine [16], extends the previously discussed traditional feedforward ANN structure of section 2.4.1 and incorporates knowledge of the gradient of label y^* with respect to input $\mathbf{x}$, in the context of pricing an arbitrary portfolio. The DML framework has its main advantage when alternative pricing methods are very time consuming and when pricing of the portfolio is required in many different scenarios. In such cases it can prove to be much faster than performing e.g. Monte Carlo pricing in each scenario, as shown in Huge and Savine [16]. The framework is also highly flexible, as the portfolio as well as the underlying stochastic models can be of virtually any type.

The goal in the DML framework is to approximate a pricing function f given by

$$f(\mathbf{x}) = \mathbb{E}[Y \mid \mathbf{x}],$$

where the expectation is taken under a risk neutral pricing measure, Y is the discounted payoff of the portfolio and $\mathbf{x}$ is the initial state of the market, such as initial values of market variables. The framework requires a training set consisting of examples ($\mathbf{x}$, y^*, $\bar{\mathbf{x}}^*$), where y^* is a sampled payoff of the portfolio given $\mathbf{x}$, and $\bar{\mathbf{x}}^*$ is the gradient of the sampled payoff with respect to the initial state of the market and is henceforth referred to as the *pathwise gradient*.

One attains the sample payoff y^* by a simulation scheme starting in $\mathbf{x}$. By considering the simulation of y^* as a sequence of mathematical operations, one can view the sampled payoff y^* as a function of $\mathbf{x}$. Furthermore, if the simulation scheme is a sequence of operations such as addition, multiplication, or application of the exponential function, which most computerised simulations are, the gradient of y^* with respect to $\mathbf{x}$, can be computed by sequential application of the chain rule. An algorithm that solves this automatically and efficiently is called *Automatic Adjoint Differentiation* (AAD) [24].

The DML architecture is built in two parts. Firstly, we wish to approximate the pricing function $f : \mathbb{R}^p \to \mathbb{R}$. Its approximate function $\hat{f} : \mathbb{R}^p \to \mathbb{R}$ is a standard feedforward ANN defined by the feedforward equations, as seen in (2.6). Secondly, as we wish to efficiently compute the gradient of $\hat{f}(\mathbf{x}) = y$ with respect to input $\mathbf{x}$, i.e. $\nabla_{\mathbf{x}} y$. To do so, we define h

as

$$\begin{aligned} \bar{z}_L &= \bar{y} = 1 \\ \bar{\mathbf{z}}_{l-1} &= (\bar{\mathbf{z}}_l \mathbf{W}_l^\top) \circ \mathbf{g}'_{l-1}(\mathbf{z}_{l-1}), \quad l = L, \dots, 1 \\ \bar{\mathbf{x}} &= \bar{\mathbf{z}}_0, \end{aligned} \tag{2.8}$$

where weights are shared with $\hat{f}$, $\circ$ is the Hadamard (elementwise) product, $\mathbf{g}'_{l-1}$ denotes the elementwise derivative of activation function $\mathbf{g}_{l-1}$ from $\hat{f}$, and

$$\bar{\mathbf{x}} = \nabla_{\mathbf{x}} y, \quad \bar{\mathbf{z}}_l = \nabla_{\mathbf{z}_l} y, \quad \bar{y} = \nabla_y y = 1. \tag{2.9}$$

The second network h is then used in conjunction with the standard feedforward network $\hat{f}$ to form a *twin network*, where $\hat{f}$ forms the first half and h forms the second half of the twin network, as seen in figure 2.1. In this twin network, there are certain layers of particular interest, firstly the layer consisting of a single neuron, namely the output of the standard feedforward ANN, y, and secondly the output layer furthest to the right in figure 2.1, containing the predicted gradient $\bar{\mathbf{x}}$.

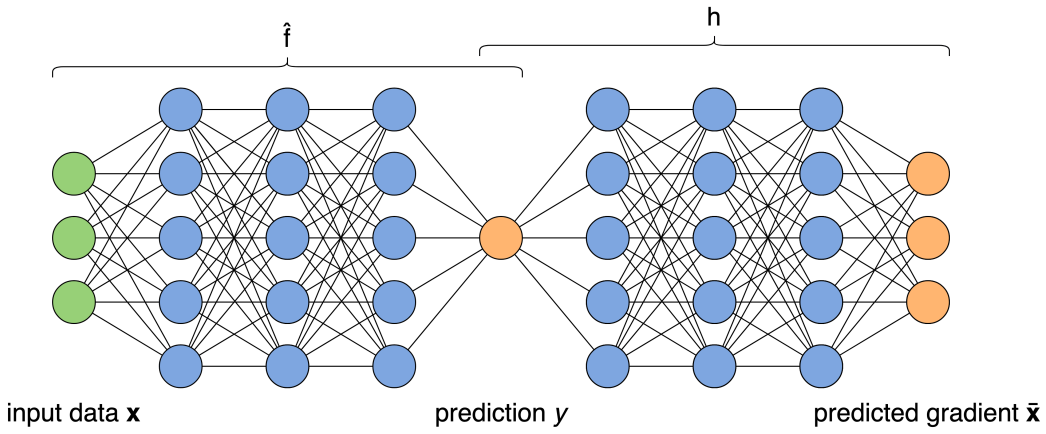


Figure 2.1: Example structure of a twin network with inputs in $\mathbb{R}^3$ (green neurons to the left), 3 hidden layers on each side of 5 hidden units each (blue neurons). The output of the feedforward ANN defined in (2.6) is the orange neuron in the middle of the network. The output of the section of the network defined in (2.8), i.e. the gradient of prediction y with respect to input data $\mathbf{x}$, is the orange neurons furthest to the right.

In training the twin network, we are reminded that the first and second half of the network share the same weights, as seen in (2.6) and (2.8). With this in mind, we realise that the training efforts are proportionate to that of a standard feedforward ANN, while we increase the amount of information about the structure of the function f we wish to approximate. However, in order to incorporate the pathwise gradients in the training of the network, we need to amend the cost function. Continuing the example of using MSE, the amended cost function may be set as

$$C(\mathbf{y}, \mathbf{y}^*, \bar{\mathbf{X}}, \bar{\mathbf{X}}^*) = \alpha \ \mathrm{MSE}(\mathbf{y}, \mathbf{y}^*) + \beta \ \sum_{i=1}^{p} \mathrm{MSE}(a_i \bar{\mathbf{X}}_i, a_i \bar{\mathbf{X}}_i^*),$$

where $\bar{\mathbf{X}}_i$ denotes the ith column of $\bar{\mathbf{X}}$, and $\alpha, \beta, a_1, \dots, a_p \geq 0$ are fixed constants. Optimal weights and biases, $\{\mathbf{W}_l^*, \mathbf{b}_l^* \ : \ l = 1, \dots, L\}$, in the network are then found by minimising

the cost function, i.e.

$$\{\mathbf{W}_l^*, \mathbf{b}_l^* \ : \ l = 1, \ldots, L\} = \underset{\{\mathbf{W}_l, \mathbf{b}_l \ : \ l=1,\ldots,L\}}{\operatorname{argmin}} C(\mathbf{y}, \mathbf{y}^*, \bar{\mathbf{X}}, \bar{\mathbf{X}}^*).$$

Huge and Savine [16] argue that the specific values of hyperparameters α and β used in the cost function are not crucial to the optimisation procedure. Therefore, as suggested in the code implementation of Huge and Savine's article [15],

$$\alpha = \frac{1}{1+p}, \quad \beta = \frac{p}{1+p},$$

will be used. We note that $\alpha + \beta = 1$, making the cost function a linear combination of the cost of incorrectly approximating both the function and its gradient, where the error of the predicted gradient is given greater significance as the dimensionality of the input data increases. Furthermore, as also seen in the code implementation of Huge and Savine's article [15], the constants $a_1, \ldots, a_p$ can be set to

$$a_i = \frac{1}{\sqrt{\text{MSE}(\bar{\mathbf{X}}_i^*, 0)}}, \quad i = 1, \ldots, p.$$

For the purpose of using the DML methods in exposure predictions, it is beneficial, if possible, to use smaller network structures. This is because a smaller network can be trained and used for predictions in a more rapid manner than larger, more complex architectures. Therefore, a smaller network structure will be employed in this thesis, the details of which are discussed in section 3.6.

2.4.3 Differential PCA

Differential principal component analysis (PCA) is a dimension reduction scheme, introduced in Huge and Savine [16], leveraging the information of pathwise gradients. Meaningful dimension reduction schemes are relevant in many different scenarios, but more specifically for the purpose of performing DML, we recognise that input dimensions may in many applications become rather large and in need of dimension reduction. Furthermore, we would like to comply with standard practice within machine learning to perform some kind of data normalisation, as without such a step, it may be difficult to train a network [18].

To illustrate differential PCA, we begin by assuming that we have a data collection consisting of input data $\mathbf{X}_0 \in \mathbb{R}^{M \times p}$ with corresponding labels $\mathbf{y}_0 \in \mathbb{R}^{M \times 1}$, and $\bar{\mathbf{X}}_0 \in \mathbb{R}^{M \times p}$, following the notation seen in (2.9), where p is the number of features in the input data and M is the number of samples. We start by centring $\mathbf{X}_0$ data, normalising $\mathbf{y}_0$ labels, and since we want the pathwise gradients to still be gradients after these operations we scale $\bar{\mathbf{X}}_0$ accordingly, such that

$$\mathbf{X}_1 = \mathbf{X}_0 - \mu_{\mathbf{x}}, \quad \mathbf{y}_1 = \frac{\mathbf{y}_0 - \mu_y}{\sigma_y}, \quad \bar{\mathbf{X}}_1 = \frac{\bar{\mathbf{X}}_0}{\sigma_y}.$$

where $\mu_{\mathbf{x}}$ is a vector of mean values of all features of $\mathbf{X}_0$, μ_Y is the mean value of the labels $\mathbf{y}_0$, and σ_y is the standard deviation of $\mathbf{y}_0$. We then proceed to apply traditional PCA by performing eigenvalue decomposition of the sample covariance matrix

$$\frac{\mathbf{X}_1^\top \mathbf{X}_1}{M} = \mathbf{P}_1 \mathbf{D}_1 \mathbf{P}_1^{-1}.$$

obtaining the $\mathbf{P}_1$ and $\mathbf{D}_1$ matrices. We note that $\mathbf{D}_1$ is a diagonal matrix containing eigenvalues and $\mathbf{P}_1$ is a matrix with corresponding eigenvectors in each column. We proceed by truncating dimensions of $\mathbf{P}_1$ and $\mathbf{D}_1$ where the corresponding eigenvalues in $\mathbf{D}_1$ are below a prespecified threshold ε_1, the reason being that eigenvalues correspond to variance in the later transformed dimensions. The threshold should not be too large, as that could risk removing important information, and the primary filtering is performed in the next step. From this we obtain the two matrices $\tilde{\mathbf{P}}_1$ and $\tilde{\mathbf{D}}_1$, possibly in a lower dimension $p_1 \leq p$. We transform $\mathbf{X}_1$ by

$$\mathbf{X}_2 = \mathbf{X}_1 \tilde{\mathbf{P}}_1 \tilde{\mathbf{D}}_1^{-\frac{1}{2}}.$$

Furthermore, we update the differentials $\bar{\mathbf{X}}_1$ to

$$\bar{\mathbf{X}}_2 = \bar{\mathbf{X}}_1 \tilde{\mathbf{P}}_1 \tilde{\mathbf{D}}_1^{\frac{1}{2}}$$

completing the traditional PCA, where we have effectively removed redundant information. Having obtained $\bar{\mathbf{X}}_2$, we can initialise the differential PCA step. The purpose of this step is to filter dimensions that in general have little impact on the label, i.e. small absolute pathwise partial derivatives. We start by performing eigenvalue decomposition of

$$\frac{\bar{\mathbf{X}}_2^\top \bar{\mathbf{X}}_2}{M} = \mathbf{P}_2 \mathbf{D}_2 \mathbf{P}_2^{-1}. \tag{2.10}$$

Now, to understand what the eigenvalues in the $\mathbf{D}_2$ matrix tell us we define $\bar{\mathbf{X}}_3 = \bar{\mathbf{X}}_2 \mathbf{P}_2$, and since $\mathbf{P}_2$ is an orthonormal matrix, we rewrite (2.10) as

$$\frac{\mathbf{P}_2^{-1} \bar{\mathbf{X}}_2^\top \bar{\mathbf{X}}_2 \mathbf{P}_2}{M} = \frac{(\bar{\mathbf{X}}_2 \mathbf{P}_2)^\top \bar{\mathbf{X}}_2 \mathbf{P}_2}{M} = \frac{\bar{\mathbf{X}}_3^\top \bar{\mathbf{X}}_3}{M} = \mathbf{D}_2.$$

Hence, the diagonal entries of $\mathbf{D}_2$ are given by

$$(\mathbf{D}_2)_{j,j} = \frac{1}{M} \sum_{i=1}^{M} \left((\bar{\mathbf{X}}_3)_{i,j} \right)^2,$$

where $(\bar{\mathbf{X}}_3)_{i,j}$ denotes the entry at row i, column j of matrix $\bar{\mathbf{X}}_3$. This means that the eigenvalues found in $\mathbf{D}_2$ show the average squared size of a certain transformed pathwise partial derivative found in the columns of $\bar{\mathbf{X}}_3$. Since labels $\mathbf{y}_1$ have been normalised and the transformed input data $\mathbf{X}_3 = \mathbf{X}_2 \mathbf{P}_2$ has undergone traditional PCA, a diagonal entry in $\mathbf{D}_2$ of e.g. 10^{-4} for a certain dimension implies that, on average, approximately a change of 100 standard deviations in this input dimension is needed to produce a change of 1 standard deviation in the label. Therefore, eigenvalues less than a given threshold ε_2 are truncated from $\mathbf{D}_2$. Based on the mentioned example, one can set ε_2 relatively large, compared to ε_1. Once again, $\mathbf{P}_2$ is truncated accordingly, and we obtain $\tilde{\mathbf{P}}_2$. $\mathbf{X}_2$ and $\bar{\mathbf{X}}_2$ are then linearly transformed, such that

$$\mathbf{X}_4 = \mathbf{X}_2 \tilde{\mathbf{P}}_2, \quad \bar{\mathbf{X}}_4 = \bar{\mathbf{X}}_2 \tilde{\mathbf{P}}_2.$$

which completes the differential PCA process. We obtain the modified input data $\mathbf{X}_4$, with sample covariance matrix equal to the identity matrix. The transformed data $\mathbf{X}_4$, $\mathbf{y}_1$, and $\bar{\mathbf{X}}_4$ is then used as training data in the network structure described in section 2.4.2.

3

Methods

Having presented necessary background information and underlying theory in chapter 2, this chapter concerns the concrete design and construction of the examples brought up in the thesis, the results of which is presented in chapter 4. In section 3.1, we begin by explaining the exposure modelling approach, which is the core of this thesis. To illustrate the use of DML in exposure modelling, the framework is applied on a certain kind of exposure modelling examples, as discussed in section 3.2. In sections 3.3 and 3.4, the discretisation schemes used to simulate from the Heston type models considered in this thesis are presented. Section 3.5 discusses computation of pathwise gradients, also relevant to the simulation of samples. Thereafter, we declare settings of various hyperparameters in section 3.6. Design of the last series of experiments is discussed in section 3.7, the results of which are found in section 4.3, which aims to stress test the exposure modelling approach. Lastly, section 3.8 defines various performance metrics used in assessing results in chapter 4. The complete code implementation of the methods presented in this chapter is available on GitHub.[‡]

3.1 Exposure Modelling Using Differential Machine Learning

In employing machine learning for exposure modelling for risk management, the general method of obtaining an approximate distribution of the future exposure E_t, as described in section 2.1, is followed. The approach to model exposure presented in this section is henceforth referred to as *exposure modelling using DML*. The exercise at hand is to, for each exposure date $t = T_1, \ldots, T_{N_E}$, price a netting set in different market scenarios specified by market risk factors $\{\mathbf{x}_{it}^{\mathbb{S}} \mid i = 1, \ldots, M\}$, obtaining $\{\mathrm{MtM}_{it} \mid i = 1, \ldots, M\}$. Examples of market risk factors includes FX rates, interest rates, various option prices, etc. The model creating these market scenarios is a risk management model and could potentially be very different from a pricing model. Such a model is hereafter referred to as an $\mathbb{S}$-*model*. The set $\{\mathbf{x}_{it}^{\mathbb{S}} \mid i = 1, \ldots, M, t = T_1, \ldots, T_{N_E}\}$ is referred to as $\mathbb{S}$-*scenarios*. The standard approach to solve this exercise is to calibrate a pricing model to each and every scenario, hereafter referred to as the $\mathbb{Q}_{it}$-*model*, to then price the netting set with this model. In general, this would require M calibrations for each exposure date. In addition, the pricing method would depend on what contracts the netting set consists of, as well as

[‡] `https://github.com/swagnerutb/mastersthesis`

the pricing model, ranging from analytical functions to Monte Carlo methods. Instead we propose the following procedure, which also is illustrated in figure 3.1.

1. In a pricing model calibrated to current market conditions, hereafter referred to as the $\mathbb{Q}$-*model*, simulate $(\mathbf{x}_{it}, y^*_{it}, \bar{\mathbf{x}}^*_{it})$ for $t = T_1, \ldots, T_{N_E}$ and $i = 1, \ldots, M$, where $\mathbf{x}_{it}$ is the market risk factors, y^*_{it} is the sum of all discounted future cash flows at time t in scenario i, and $\bar{\mathbf{x}}^*_{it}$ is the pathwise gradient, i.e. the gradient of y^*_{it} with respect to $\mathbf{x}_{it}$.

2. For each time $t = T_1, \ldots, T_{N_E}$, learn a function $\hat{f}_t$ s.t. $\hat{f}_t(\mathbf{x}_{it}) \approx \mathrm{MtM}_{it}$. $\hat{f}_t$ is a DML model, as described in section 2.4.2. The training set $(\mathbf{x}_{it}, y^*_{it}, \bar{\mathbf{x}}^*_{it})$ for $i = 1, \ldots, M$ is preprocessed by differential PCA, as described in section 2.4.3, before it is used to train $\hat{f}_t$.

3. For each time $t = T_1, \ldots, T_{N_E}$, apply $\hat{f}_t$ to the $\mathbb{S}$-scenarios, $\mathbf{x}^{\mathbb{S}}_{it}$, for $i = 1, \ldots, M$, i.e. $\hat{f}_t(\mathbf{x}^{\mathbb{S}}_{it}) = \mathrm{MtM}_{it}$, and use these values to form an empirical distribution of the exposure at time point t.

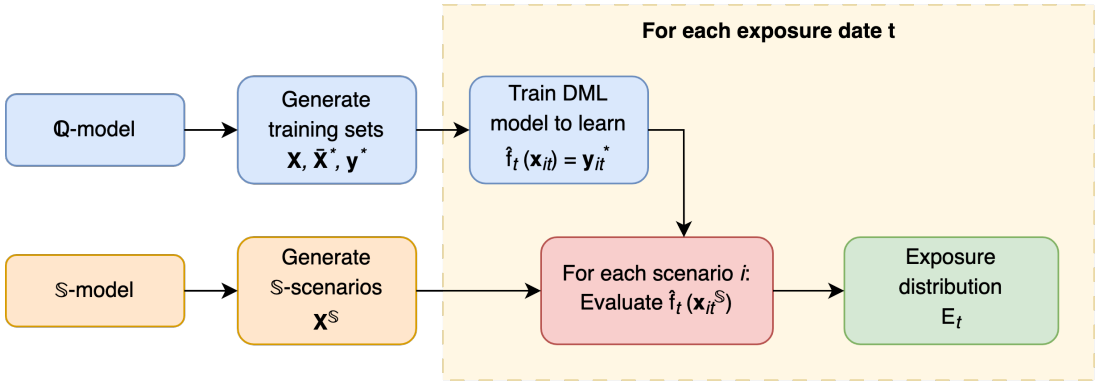


Figure 3.1: Schematic illustration of exposure modelling using DML.

The main potential benefits of exposure modelling using DML concern flexibility and speed. Since the $\mathbb{S}$- and $\mathbb{Q}$-models are allowed to be of different type, risk managers that produce the scenarios in which the netting set must be priced could potentially put an increased focus on creating scenarios that are relevant for the risk assessment, not having to adapt as much to the pricing aspects. Meanwhile, it potentially allows pricing models to be chosen with greater freedom. This means that the presented methodology could possibly lead to more accurate risk management. In addition, the only restriction on what type of netting sets can be treated by the framework is that payoffs and pathwise gradients of this netting set can be simulated. Secondly, DML is shown to be a computationally efficient method, as discussed in Huge and Savine [16]. Furthermore, generation of training sets by the $\mathbb{Q}$-model is efficient in the sense that the same simulation paths are used for all training sets, only at different exposure dates.

The main potential drawbacks of exposure modelling using DML concern guarantees of performance. Firstly, the performance of the method is stochastic. Both the generation of the training sets and the training of the DML model are stochastic procedures, inevitably leading to varying performance of the model from time to time. Secondly, $\hat{f}_t$ learned in

step 2 above approximates the pricing function of the netting set under the $\mathbb{Q}$-model, since it is only trained on data from that model. This $\mathbb{Q}$-model may very well have different parameter values than the $\mathbb{Q}_{it}$-models, where we remind ourselves that the latter is achieved by calibrating a pricing model of same type as the $\mathbb{Q}$-model to $\mathbb{S}$-scenario $\mathbf{x}_{it}^{\mathbb{S}}$. This means that although $\hat{f}_t$ may be a close to perfect approximator, in terms of what it has trained on, it might provide non-satisfactory results in pricing the netting set in the $\mathbb{S}$-scenarios. In other words, it may not be a good approximator of the pricing function under the $\mathbb{Q}_{it}$-model. This is caused by a discrepancy between the $\mathbb{S}$- and $\mathbb{Q}$-model, and how this discrepancy affects performance is a difficult question to answer. Additionally, the answer most certainly differs from case to case. In practice, this demands extensive testing and backtesting to ensure that the method works as intended.

3.2 Example Settings

To illustrate the use of exposure modelling using DML, the methodology is employed to model exposure of a netting set consisting of European call and put options where both the $\mathbb{S}$- and $\mathbb{Q}$-model are Heston models or multi-asset Heston models, as discussed in section 2.2 and 2.3, respectively. Underlying assets are thought to be *FX pairs*, such as GBP/USD[§], influencing the choice of parameter values throughout the thesis. The market risk factors used to describe the market are limited to the spot asset prices and prices of European call options with specified *moneyness* m, meaning that the strike price is a prespecified proportion of the spot asset price at initialisation, and time to expiry τ months. These observed call option prices are denoted $c^i_{m,\tau}(t)$ for underlying asset i. As an example, $c^i_{1.10,3}(t)$ denotes the value of a call option on underlying asset i at time t, initialised at time t with time to expiry 3 months and moneyness $m = 1.10$, i.e. strike price $K = 1.10 \cdot S_i(t)$. The features of the input data will hence consist of spot asset prices and prices on European option that has the spot asset prices as underlyings.

Throughout this thesis all experiments shown will share some features. Firstly, only one exposure date is considered for all exposure modelling at $T_1 = 6$ months. Secondly, all instruments in considered netting sets have a date of expiry at $T = 12$ months. Furthermore, parameter values, as suggested in Heston [13], can be found in table 3.1, henceforth referred to as *standard values*, as these parameter values frequently occur throughout the thesis.

Heston type models are chosen as these have been extensively studied, have (almost) closed form solutions, as seen in section 2.2.1, and have a hidden state, namely the stochastic variance process. The fact that stochastic variance is a model construction not observable in the market makes additional observed European call option prices meaningful input to the network, as these may provide information regarding the hidden state imperative to the valuation of the netting set. If, for example, a simpler model such as the Black-Scholes model [5] would be considered, market observables might not be as meaningful to include, as option prices in this case are functions of spot asset price and fixed model parameters only. Netting sets consisting exclusively of European options are used as these are the most relevant, and arguably the simplest, instruments appropriate to price using DML,

[§]For an FX pair such as GBP/USD, the spot asset price $S(t)$ is the price of 1 Pound Sterling (GBP) in United States dollars (USD).

and having (almost) closed form solutions in the Heston model allows for fast verification. Pricing with DML is only useful if instruments cannot be directly prices from market risk factors, or, in other words, where the computation of a conditional expectation is needed, which is the case for a European call option in the Heston model.

Parameter	**Standard value**
Mean reversion κ	2
Long-run average variance θ	0.01
Current variance $\nu(0)$	0.01
Correlation of Wiener processes in price and variance processes ρ	0
Volatility of variance process ξ	0.1
Drift of spot price process μ	0
Option time of expiry T [months]	12

Table 3.1: Standard parameter values in the Heston model, as found in Heston [13].

3.3 Sample Generation in the Heston Model

To use the exposure modelling approach explained in section 3.1, using the setting of section 3.2, sampling from the Heston model described in section 2.2 is required. There are many discretisation schemes for the Heston model, and in this thesis the full truncation scheme of Lord et al. [21] is employed, suggesting that

$$\begin{aligned} S(t+\Delta t) &= S(t)\exp\left((\mu - \frac{\nu^+(t)}{2})\Delta t + \sqrt{\nu^+(t)\Delta t}\; w_t^S\right) \\ \nu(t+\Delta t) &= \nu(t) + \kappa(\theta - \nu^+(t))\Delta t + \xi\sqrt{\nu^+(t)\Delta t}\; w_t^\nu \end{aligned} \tag{3.1}$$

for $t = 0, \ldots, N_T - 1$, where

$$\begin{bmatrix} w_t^S \\ w_t^\nu \end{bmatrix}$$

are simulated realisations of a two-dimensional normal distribution with expectation $[0,0]^\top$ and covariance matrix

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}.$$

Furthermore, N_T is the number of time steps, Δt is the size of one time step, and $\nu^+(t) = \max\{\nu(t), 0\}$. Although the Feller condition is fulfilled, the discretisation scheme leads to a non-zero probability of a negative variance $\nu(t)$. The full truncation scheme handles this event by using $\nu^+(t)$ instead. The presented discretisation scheme was implemented using the Python package `NumPy`. Throughout all simulations used in this thesis, the number of time steps is set to $N_T = 100$.

3.4 Sample Generation in the Multi-Asset Heston Model

In order to generate samples of the multi-asset Heston model of N assets, as described in section 2.3, we use an Euler scheme [7], stating that

$$\begin{aligned} S_i(t+\Delta t) &= S_i(t)\exp\left(\left(\mu_i(t)-\frac{1}{2}\nu_i^+(t)\right)\Delta t+\sqrt{\nu_i^+(t)\Delta t}\ w_i^S(t)\right) \\ \nu_i(t+\Delta t) &= \nu_i(t)+\kappa_i\left(\theta_i-\nu_i^+(t)\right)\Delta t+\xi_i\sqrt{\nu_i^+(t)\Delta t}\ w_i^\nu(t) \end{aligned} \tag{3.2}$$

for $t = 0, \ldots, N_T - 1$ and $i = 1, \ldots, N$, where N_T is the number of simulated time steps, and Δt is the size of one time step. Once again, the use of $\nu_i^+(t) = \{\nu_i(t),0\}$ is required to handle cases of a negative $\nu_i(t)$ resulting from discretisation.

As previously discussed in section 2.3, the Wiener processes $W_i^S(t)$, $W_i^\nu(t)$ have correlation ρ_i, and $W_i^S(t)$, $W_j^S(t)$ for $i \neq j$ have correlation $\tilde{\rho}_{ij}$. In generating the Wiener process increments $\mathbf{w}^S$, an N-dimensional multivariate normal distribution with expectation 0 in all dimensions and covariance matrix $\Sigma_1 = (\tilde{\rho}_{ij})_{i\leq N, j\leq N}$ is sampled N_T times, where N is the number of assets and $\tilde{\rho}_{ij} = 1$ for $i = j$. To generate $\mathbf{w}^\nu$, we start by sampling an N-dimensional standard multivariate normal distribution N_T times and obtain $\tilde{\mathbf{w}}$. We then let

$$w_i^\nu = \rho_i w_i^S(t) + \sqrt{(1-\rho_i^2)}\ \tilde{w}_i$$

for $i = 1, \ldots, N$, $t = 1, \ldots, N_T$, ensuring that $\text{corr}(w_i^S(t), w_i^\nu(t)) = \rho_i$. The presented discretisation scheme was implemented using the Python package `NumPy`. Throughout all simulations used in this work, the number of time steps is set to $N_T = 100$.

3.5 Pathwise Gradients

As seen in section 3.1, the application of DML demands pathwise gradients of labels with respect to inputs. In this section discuss the computation of these in the setting considered in the thesis. First, it is explained how the pathwise gradient with respect to the spot asset price and the stochastic variance, $(S(t), \nu(t))$, is attained. However, as previously mentioned, we also have option prices, instead of $\nu(t)$, as input data, meaning that we will need pathwise gradients with respect to these. The procedure of finding these are discussed towards the end of this section.

The pathwise gradients with respect to the spot asset price and the stochastic variance, $(S(t), \nu(t))$, are obtained by AAD using the Python library `Algopy` [26]. Firstly, we emphasise that pathwise gradients are calculated sample by sample, and all notation in this section refers to a single sample, a single simulation path. Secondly, we note that a netting set consisting of European put and call options allows us to compute the pathwise gradients asset by asset, and thus we only consider one asset in this section. Computation of gradients using AAD were introduced in section 2.4.2, and here follows a more thorough explanation related to the setting of the thesis.

Let y be the payoff, then the pathwise gradient of the payoff with respect to the spot asset price and the stochastic variance

$$\begin{bmatrix} \frac{\partial y}{\partial S(t)} \\ \frac{\partial y}{\partial \nu(t)} \end{bmatrix}$$

is computed by considering the payoff y as a function of the spot asset price $S(t)$ and stochastic variance $\nu(t)$ at time t. The function for this specific sample mapping $(S(t), \nu(t))$ to y is specified by the progression of the spot asset price and the stochastic variance according to the discretisation schemes (3.1) or (3.2), and then application of the payoff function on $S(T)$, where T is the date of expiry. The progression of the spot asset price and the stochastic variance involves several realisations of random variables, and these are considered as constants in the gradient computation. The payoff function is decided by the instruments in the netting set. For example, if the netting set would contain a European call option with strike price K_c and a European put option with strike price K_p, both expiring at time T, the payoff at time T is

$$y(S(T)) = \max\{S(T) - K_c, 0\} + \max\{K_p - S(T), 0\}.$$

$S(T)$ is in turn related to $(S(t), \nu(t))$ by the discretisation scheme of the processes, as previously discussed. AAD then automatically computes the gradient of y with respect to $(S(t), \nu(t))$.

Instead of stochastic variance $\nu(t)$, P European call option prices are used together with the spot asset price $S(t)$ as input data to the DML framework. The question is how to get the pathwise gradient with respect to the call option prices. We use the fact that that the pathwise gradient with respect to $(S(t), \nu(t))$, and gradients of the European call option prices with respect to $(S(t), \nu(t))$ are known in a given simulation. The call option prices have a given moneyness m and time to expiry τ, the value of which at time t with underlying asset i is denoted $c^i_{m,\tau}(t)$. As we can consider pathwise gradients asset by asset in this section, $i = 1$. Furthermore, since the underlying price process is modelled with the multi-asset Heston model, option prices, $c^1_{m_p,\tau_p}(t)$, for $p = 1 \dots P$, are created by using the Heston call price formula.

Here follows the procedure used to retrieve the pathwise gradient with respect to $(S(t), c^1_{m_1,\tau_1}(t))$, i.e. for the case $P = 1$. We let

$$\begin{aligned} y(S(t), \nu(t)) &= f \circ g(S(t), \nu(t)), \\ f(S(t), c^1_{m_1,\tau_1}(t)) &= y(S(t), \nu(t)), \\ g(S(t), \nu(t)) &= (S(t), c^1_{m_1,\tau_1}(t)), \end{aligned}$$

where y is the payoff function, and g is a mapping from the model state $(S(t), \nu(t))$ to $(S(t), c^1_{m_1,\tau_1}(t))$, the latter being called the *market state*. Namely, f is the function of which we want the gradient with respect to the market state. The chain rule gives that

$$\mathbb{J}_y(S(t), \nu(t)) = \mathbb{J}_f(g(S(t), \nu(t)))\mathbb{J}_g(S(t), \nu(t)) = \mathbb{J}_f(S(t), c^1_{m_1,\tau_1}(t))\mathbb{J}_g(S(t), \nu(t)),$$

where $\mathbb{J}_a(b, c)$ denotes the Jacobian of function a evaluated at the point (b, c). We obtain

the desired gradient by solving the linear system

$$\nabla f(S(t), c^1_{m_1,\tau_1}(t)) = \mathbb{J}_f(S(t), c^1_{m_1,\tau_1}(t))^\top = \left(\mathbb{J}_g(S(t), \nu(t))^\top\right)^{-1} \mathbb{J}_y(S(t), \nu(t))^\top =$$
$$= \begin{bmatrix} \frac{\partial S(t)}{\partial S(t)} & \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial S(t)} \\ \frac{\partial S(t)}{\partial \nu(t)} & \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial \nu(t)} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial y}{\partial S(t)} \\ \frac{\partial y}{\partial \nu(t)} \end{bmatrix} = \begin{bmatrix} 1 & \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial S(t)} \\ 0 & \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial \nu(t)} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial y}{\partial S(t)} \\ \frac{\partial y}{\partial \nu(t)} \end{bmatrix} =$$
$$= \begin{bmatrix} \frac{\partial y}{\partial S(t)} - \frac{\frac{\partial y}{\partial \nu(t)} \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial S(t)}}{\frac{\partial c^1_{m_1,\tau_1}(t)}{\partial \nu(t)}} \\ \frac{\frac{\partial y}{\partial \nu(t)}}{\frac{\partial c^1_{m_1,\tau_1}(t)}{\partial \nu(t)}} \end{bmatrix}.$$

$\partial y/\partial S(t)$ and $\partial y/\partial \nu(t)$ are computed by AAD when simulating the data set, as previously discussed. $\partial c^1_{m_1,\tau_1}(t)/\partial S(t)$ and $\partial c^1_{m_1,\tau_1}(t)/\partial \nu(t)$ are obtained by differentiating the Heston option formula using the finite difference method. A central difference scheme is employed with $h = 10^{-8}$, i.e.

$$\frac{\partial c^1_{m_1,\tau_1}(t)}{\partial S(t)} \approx \frac{1}{2h}(C(S(t)+h, \nu(t), m(S(t)+h), t, T) - \\ - C(S(t)-h, \nu(t), m(S(t)-h), t, T))$$
$$\frac{\partial c^1_{m_1,\tau_1}(t)}{\partial \nu(t)} \approx \frac{1}{2h}(C(S(t), \nu(t)+h, m(S(t)), t, T) - \\ - C(S(t), \nu(t)-h, m(S(t)), t, T))$$

where the function C is defined as seen in (2.4). For $P > 1$ we undertake a similar procedure as for the case of $P = 1$. We let

$$y(S(t), \nu(t)) = f \circ g(S(t), \nu(t)),$$
$$f(S(t), c^1_{m_1,\tau_1}(t), \ldots, c^1_{m_P,\tau_P}(t)) = y(S(t), \nu(t)),$$
$$g(S(t), \nu(t)) = (S(t), c^1_{m_1,\tau_1}(t), \ldots, c^1_{m_P,\tau_P}(t)).$$

Here, we also get a system of linear equations,

$$\mathbb{J}_g(S(t), \nu^1(t), \ldots, \nu^P(t))^\top \nabla f(S(t), c^1_{m_1,\tau_1}(t), \ldots, c^1_{m_P,\tau_P}(t)) =$$
$$= \nabla y(S(t), \nu(t)) = \begin{bmatrix} \frac{\partial S(t)}{\partial S(t)} & \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial S(t)} & \cdots & \frac{\partial c^1_{m_P,\tau_P}(t)}{\partial S(t)} \\ \frac{\partial S(t)}{\partial \nu(t)} & \frac{\partial c^1_{m_1,\tau_1}(t)}{\partial \nu(t)} & \cdots & \frac{\partial c^1_{m_P,\tau_P}(t)}{\partial \nu(t)} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial S(t)} \\ \frac{\partial f}{\partial c^1_{m_1,\tau_1}(t)} \\ \vdots \\ \frac{\partial f}{\partial c^1_{m_P,\tau_P}(t)} \end{bmatrix} = \begin{bmatrix} \frac{\partial y}{\partial S(t)} \\ \frac{\partial y}{\partial \nu(t)} \end{bmatrix}, \tag{3.3}$$

where we have 2 equations but $P + 1$ unknowns, and since $P > 1$, the system is underdetermined with infinitely many solutions. The solution with smallest Euclidean norm is chosen and is obtained using the `NumPy` package in Python. Thus, the problem is viewed as a high dimensional regression problem using ridge regression. A motivation for this is that we expect the function to be rather "nice", and choosing this solution helps to spread out the gradient on the different partial derivatives. In contrast, taking a Lasso approach would encourage setting some partial derivatives low and other high, which does not correspond to the current definition of a "nice" function.

3.6 Differential Machine Learning Hyperparameter Settings

For the implementation of the twin network discussed in section 2.4.2, the Python library `TensorFlow` [22] is used. The same network architecture and training procedure is used throughout all experiments in the thesis. The inspiration for choosing the given network specifications stems from Huge and Savine [16] and the code implementation of the article [15]. The network specifications can be seen in table 3.2.

Parameters	**Options**
Hidden layers	4
Neurons per hidden layer	20
Activation function	Softplus
Optimiser	Adam
Batch size	256
Epochs	10

Table 3.2: Parameter specifications for the ANN architecture used throughout the thesis. Definition of the softplus function can be found in (2.7).

Adaptive moment estimation (Adam) [17] is one of the most common optimisers used in `TensorFlow`. It is a gradient based optimiser that keeps track of previous gradients in order to dynamically improve the optimisation procedure.

As high dimensional data is present, thresholds $\epsilon_1 = 10^{-12}$ and $\epsilon_2 = 2 \cdot 10^{-4}$ for the standard PCA and differential PCA data preprocessing steps, respectively, will be used in the differential PCA methodology, as seen in section 2.4.3. The given values were decided after extensive testing.

3.7 Changing the $\mathbb{Q}$-model Parameters

In one set of experiments, with results given in section 4.3, it is examined how well exposure modelling using DML performs for $\mathbb{Q}$-models differing from a fixed $\mathbb{S}$-model in a single-asset setting. More specifically, values of the following parameters in the $\mathbb{Q}$-model are altered: long-term mean of the variance process θ, mean reversion rate of the variance process κ, volatility of the variance process ξ, and correlation of Wiener processes in price and variance processes ρ. To decide which sets of values to include for the $\mathbb{Q}$-model parameters, reasonable spans for the different parameters with regards to the FX pair application are required. Secondly, the Feller condition of (2.2) needs to hold. Given these constraints, we randomly generate the set of $\mathbb{Q}$-model parameters, the details of which are found in section 3.7.1. Parameter values of the $\mathbb{S}$-model are chosen as an average of the corresponding parameter values in the $\mathbb{Q}$-model. This is done to effectively demonstrate when exposure modelling using DML fails.

3.7.1 Spans of Heston Parameters

In Winter [27], 10 years of daily calibrations of the Heston model to the FX pair GBP/USD are presented. These calibration results are used as inspiration to set possible spans for the Heston parameters when applied to a setting of heavily traded FX pairs, as found in table 3.3.

Parameter	**Span**
Mean reversion κ	[0.25, 8]
Long-run average variance θ	[0.005, 0.1]
Correlation of Wiener processes in price and variance processes ρ	[-0.9, 0]
Volatility of the variance process ξ	[0.01, 1.5]

Table 3.3: Considered spans for four different Heston parameters.

As paths are generated from the Heston model according to the discretisation scheme described in section 3.3, it is important that the Feller condition holds. When the Feller condition is not fulfilled, cases of negative realisations of the discretised variance processes become frequent, leading to inadequate samples. This is why all combinations not fulfilling the Feller condition are filtered out, leaving us with the parameter space $\mathcal{B}$ as defined in (3.4).

$$\mathcal{B} = \left\{ (\kappa, \theta, \rho, \xi) \mid \begin{array}{l} \kappa \in [0.25, 8], \\ \theta \in [0.005, 0.1], \\ \rho \in [-0.9, 0], \\ \xi \in [0.01, 1.5], \\ \xi^2 < 2\kappa\theta \end{array} \right\} \tag{3.4}$$

To obtain a set of 1,000 $\mathbb{Q}$-model parameter combinations, rejection sampling [8] is used to draw uniformly from the set $\mathcal{B}$. To be specific, each of the 1,000 combinations are obtained in the following way:

1. Draw $\theta \sim U([0.005, 0.1])$, $\kappa \sim U([0.25, 8])$, $\rho \sim U([-0.9, 0])$, and $\xi \sim U([0.01, 1.5])$ independently.

2. If $\xi^2 \geq 2\kappa\theta$, reject the proposed combination and return to step 1. Otherwise, accept the parameter combination $(\kappa, \theta, \rho, \xi)$.

Parameter values for the $\mathbb{S}$-model are then set to the average of the corresponding $\mathbb{Q}$-model parameter values in the generated set.

3.7.2 Training and Test Set Overlap

It is expected that simulating data from Heston models with different parameter values may lead to data sets contained in different subregions of the space in question. Since the $\mathbb{Q}$-model is used to create the training set in the DML approach, and the $\mathbb{S}$-model to create the test set, i.e. the $\mathbb{S}$-scenarios in which the model will be used, this may lead to disjoint training and test sets. With this in mind, we may be interested in describing and measuring the union of the two sets of data points in $\mathbb{R}^n$. A metric that aims to capture this, referred to as the *training and test set overlap*, is calculated according to:

1. Get the convex hulls of the training set and the test set. Calculate the Lebesgue measure of the convex hull of the test set, L_1.**
2. Get the intersection of the two convex hulls. Calculate the Lebesgue measure of this intersection, L_2.
3. Set the *training and test set overlap* to L_2/L_1.

To implement this procedure, the Python packages `SciPy` and `Shapely` are used. See figure 3.2 for an example illustrating the procedure in two dimensions.

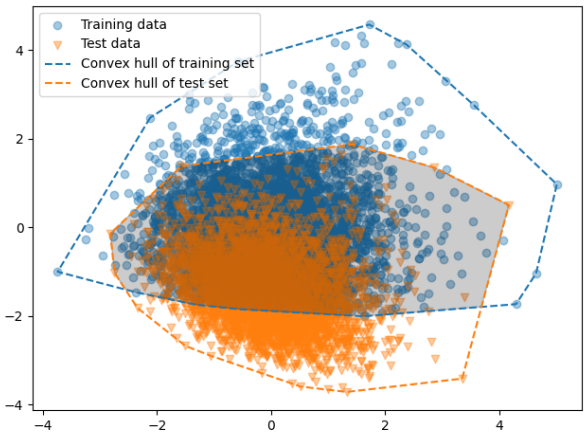


Figure 3.2: Illustration of a training set and a test set together with their convex hulls. The shaded area is the intersection of the convex hulls.

3.8 Performance Measuring

Performance of the methodology can be measured using different approaches. As previously discussed in section 3.1, we wish to estimate the future exposure at time $T_1 \in (0,T]$, where T is the expiry time furthest away of any instrument in the netting set. From pricing the generated $\mathbb{S}$-scenarios, we obtain predictions $\mathbf{y}$ of the future value of the netting set. For a

**In two dimensions, the Lebesgue measure coincides with the concept of area, and in three dimensions it coincides with the concept of volume.

set of M such scenarios, we can determine the accuracy of a prediction by comparison to the "true" value $\mathbf{y}^*$ using e.g. the root mean square error

$$\text{RMSE}(\mathbf{y}, \mathbf{y}^*) = \sqrt{\frac{\sum_{i=1}^{M}(y_i - y_i^*)^2}{M}}.$$

By using RMSE, we find a way to quantify the combined accuracy of predictions for all M paths in a Euclidean fashion. From a machine learning point of view, this might be interesting as it gives an indication of the accuracy of the prediction model, although it might be less relevant for risk management purposes.

Instead, a risk manager is in practice most often after specific metrics of the future exposure distribution, such as its expectation, 95th or 97.5th percentile. In order to test the accuracy of the overall framework for the purpose of exposure modelling, these distribution metrics are most relevant. Proceeding, we will use counterparty credit risk terms, as defined in section 2.1.2, and refer to these metrics as EE_t, $\text{PFE}_t^{0.95}$, and$\text{PFE}_t^{0.95}$, respectively, for an exposure date t months into the future. Furthermore, for illustrative purposes only, kernel density estimations using a Gaussian kernel will be employed to produce smooth exposure distributions using the `seaborn` package in Python.

4

Results

In this chapter we present the results of various experiments that aim to show the functionality and performance of exposure modelling using DML, as explained in section 3.1. The chapter can be seen as consisting of two primary parts, the first being covered in sections 4.1 and 4.2, and the second covered in section 4.3. The first part examines the use of different sets of market observables and compares performance to information provided to the DML framework in different settings. It is also shown how the dimensionality of the problem affects performance. The Heston model and the multi-asset Heston model are employed in section 4.1 and 4.2, respectively. The second part consists of a series of experiments aiming to stress test the exposure modelling approach by keeping the $\mathbb{S}$-model parameters fixed whilst altering parameters of the $\mathbb{Q}$-model for a relatively simple exposure modelling example.

4.1 Heston Model

We begin in a Heston model setting, as described in section 2.2, where exposure modelling is performed on a netting set consisting of a single European call option. We wish to display the applicability of market observables for exposure modelling using DML, as described in section 3.1. In doing so, we alter the information available to the DML framework at the exposure date T_1. Firstly, we examine how the method performs using information of $S(T_1)$ and $\nu(T_1)$, where we note that the latter is a hidden state which cannot be observed in the market. This set of information will henceforth be referred to as *model state information*. In this particular case of the same $\mathbb{S}$- and $\mathbb{Q}$-model, we know, by the Heston call price formula described in section 2.2.1, that the netting set may be priced directly using only these two factors. Therefore, model state information serves as a benchmark for the accuracy of DML predictions in terms of the quality and quantity of information available. Secondly, we take on a somewhat naïve approach and only supply the DML framework with information about the spot asset price at time T_1, i.e. $\{S(T_1)\}$. We do so despite knowing that $S(T_1)$ is insufficient to accurately price the netting set in order to obtain a second benchmark for the accuracy of the framework. Lastly, we also include a number of different market observables combinations, for example the spot asset price of the underlying at time T_1 and the observed price of an ATM call option with time to expiry $\tau = 3$ months. Such a set of market observables is denoted $\{S(T_1), c_{1,3}(T_1)\}$. This is in turn compared to supplying the framework with both model state information and the naïve approach $\{S(T_1)\}$. It is primarily the exposure modelling performances,

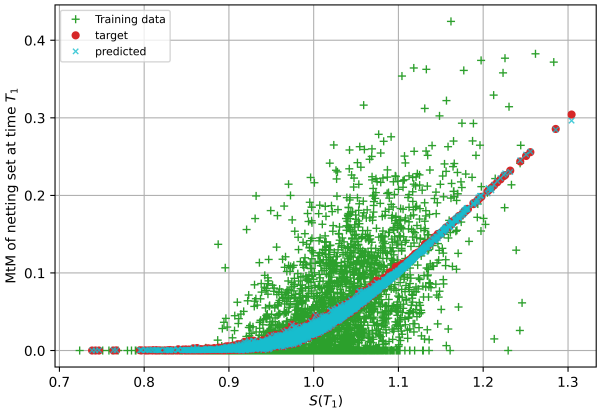
namely how close the estimated distribution comes to the target distribution, that is compared.

The netting set is specified by 1 call option with strike price $K = 1$ and time to expiry $\tau = 12$ months at time $t = 0$. The exposure date T_1 is 6 months into the future, meaning that the the call option will have a time to expiry $\tau = 6$ months at time T_1. The progression of the underlying asset is described by the Heston model with the standard parameter values found in table 3.1. These parameter values are used for both the $\mathbb{Q}$-model and the $\mathbb{S}$-model. Samples are created according to the discretisation scheme described in section 3.3. Furthermore, a total of $M = 4{,}096$ scenarios are evaluated in all cases, where initial values of the spot asset price process and the variance process are set to $S_0 = 1$ and $\nu_0 = 0.01$, respectively. M was chosen such that the twin network will be given enough data to produce meaningful predictions, whilst still being rather restrictive to display the full capabilities of the DML method. In connection with this, it should be noted that the longest training time of any network in this section was 3.2 seconds, running on a machine with 2,3 GHz Dual-Core Intel Core i5 CPU and 16 GB 2133 MHz LPDDR3 RAM. For visualisation purposes, the same single seed is used to produce all figures shown in this section. Towards the end of the section, general results for a larger number of seeds are shown.

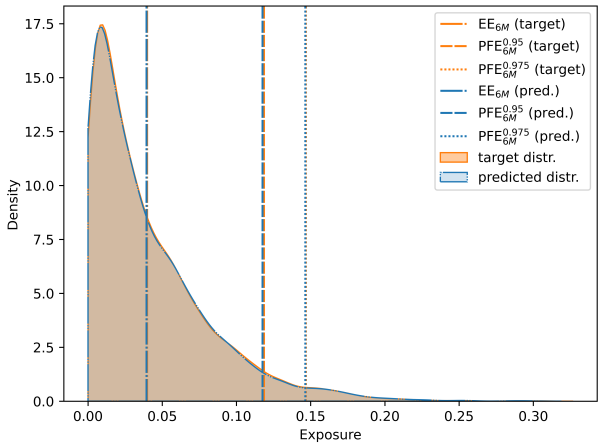
In figure 4.1, we see predictions when supplying the DML framework with model state information $\{S(T_1), \nu(T_1)\}$. In figure 4.1a, we see the training data as well as test data, i.e. $\mathbb{S}$-scenarios where the true MtM of the netting set in these scenarios are obtained using the Heston call price formula described in section 2.2.1. In order to assess the DML prediction performance, the netting set is priced in the $\mathbb{S}$-scenarios using the twin network, and we find the corresponding predictions in figure 4.1a as well. We see that a relatively accurate prediction is produced. We also see that the shape of the test data is captured, indicating that the DML methodology operates as intended. Furthermore, while barely being noticeable, predictions tend to be somewhat less accurate in the upper interval of X_1, caused by a scarcity of training data in this region.

Although results of figure 4.1a are not presented in this manner for the purpose of exposure modelling, it is important from a machine learning perspective and helps to illustrate how the DML operates. The target values and predictions of figure 4.1a are then used to form the future exposure distributions seen in figure 4.1b. Here, we see that the target distribution is captured rather closely.

Proceeding to a more realistic, albeit naïve, setting, the twin network is now given $\{S(T_1)\}$ only, which may be observed directly in the market. In figure 4.2a, we see the resulting MtM predictions in the $\mathbb{S}$-scenarios. As only one variable is supplied to the network, the pricing function we aim to approximate is, in the perspective of the twin network, a function $\mathbb{R} \to \mathbb{R}$, which is why the resulting predictions only depend on $S(T_1)$, as seen in figure 4.2a. In figure 4.2b, we see the predicted future exposure distribution, which is not quite as accurate in the tail, i.e. predicted values of $\text{PFE}_{6M}^{0.95}$ and $\text{PFE}_{6M}^{0.975}$, as that seen in figure 4.1b, where the network was supplied with model state information. Nevertheless, the prediction of EE_{6M} is, in fact, more accurate than in the model state information case.

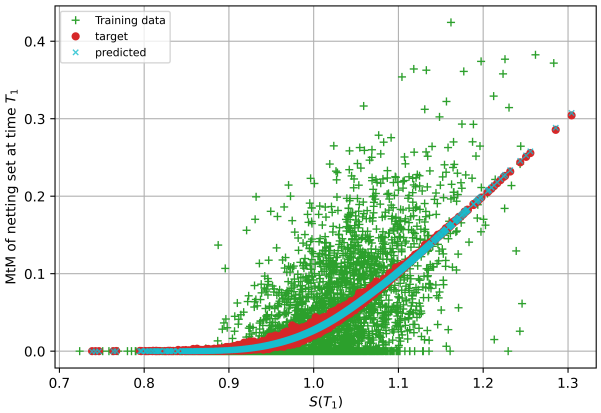


(a) Training data (payoffs) of all M training paths, predictions on test set, and target values (MtM using the Heston call price formula) of test set. RMSE = 0.00065.

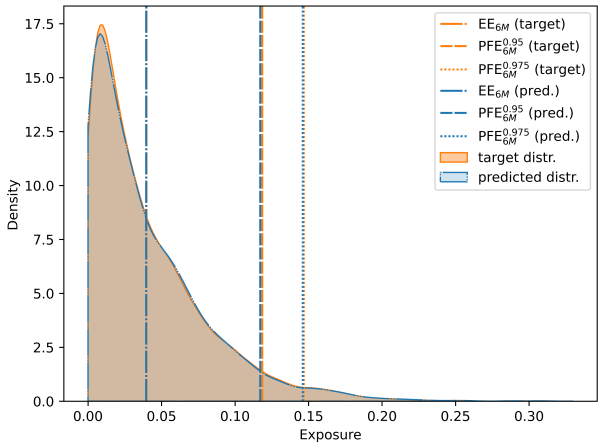


(b) Target and predicted future exposure distribution. Relative deviation between predicted and target EE_{6M} is -1.016%, $PFE^{0.95}_{6M}$ is -0.822%, $PFE^{0.975}_{6M}$ is -0.270%.

Figure 4.1: Netting set consisting of 1 call option with strike $K = 1$ at time T_1, where the network was given $\{S(T_1), \nu(T_1)\}$ in $M = 4{,}096$ simulated scenarios. Data from (a) is used to form the future exposure distribution for time T_1 seen in (b).



(a) Training data (payoffs) of all M training paths, predictions on test set, and target values (MtM using the Heston call price formula) of test set. RMSE = 0.00303.

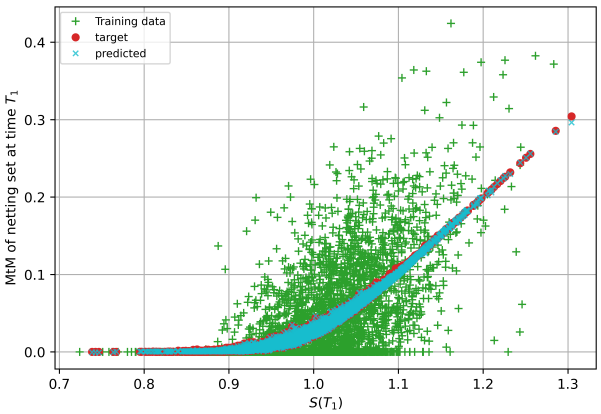


(b) Target and predicted future exposure distribution. Relative deviation of EE_{6M} is -0.763%, $PFE^{0.95}_{6M}$ is -1.216%, $PFE^{0.975}_{6M}$ is -0.522%.

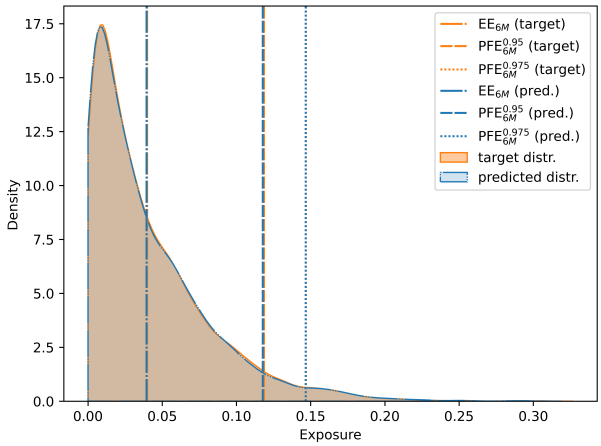
Figure 4.2: Netting set consisting of 1 call option with strike $K = 1$ at time T_1, where the network was given $\{S(T_1)\}$ in $M = 4{,}096$ simulated scenarios. Data from (a) is used to form the future exposure distribution for time T_1 seen in (b).

We now proceed to introduce additional market observables including observed call option prices, as previously discussed in section 3.2. At first, we investigate a set of spot asset price and the price of an ATM call option at time T_1 on the same underlying asset as that which the value of our netting set is dependent on. In figure 4.3, we see the predicted

future exposure distribution when supplying the DML framework with $\{S(T_1), c_{1,3}(T_1)\}$. Once again, we show the predicted and target MtM in the $\mathbb{S}$-scenarios, along with the training data, in figure 4.3a. The corresponding predicted and target future exposure distributions are seen in figure 4.3b. It should be noted that the choice of τ can be made in different ways. In table 4.1, we see the absolute relative deviation of the EE_{6M}, $\text{PFE}_{6M}^{0.95}$, and $\text{PFE}_{6M}^{0.975}$ between the predicted distribution and the target distribution for sets $\{S(T_1), c_{1,\tau}(T_1)\}$ with $\tau = 1$, 3, or 12 months. For this given seed, we find that $\tau = 1$ month provides data resulting in the most accurate prediction of EE_{6M} and $\text{PFE}_{6M}^{0.95}$, and $\tau = 3$ months results in the most accurate predictions of $\text{PFE}_{6M}^{0.975}$. We note that for all values of τ, we obtain a more precise prediction of the tail metrics than in the case previously seen for only supplying the network with $S(T_1)$. However, predictions of the EE_{6M} are less accurate than in figure 4.2, but approximately the same as seen in figure 4.1 where the twin network was given model state information. Despite the decreased accuracy in EE_{6M} predictions when using $\{S(T_1), c_{1,3}(T_1)\}$ data instead of $\{S(T_1)\}$, it seems the shape of the predicted exposure distribution replicates the shape of the target distribution more closely in the case of the observed call option price, as seen in comparing figure 4.3b to 4.2b, particularly the modes.



(a) Training data (payoffs) of all M training paths, predictions on test set, and target values (MtM using the Heston call price formula) of test set. RMSE = 0.00066.



(b) Target and predicted future exposure distribution. Relative deviation EE_{6M} is -1.048%, $\text{PFE}_{6M}^{0.95}$ is -0.678%, $\text{PFE}_{6M}^{0.975}$ is -0.133%.

Figure 4.3: Netting set consisting of 1 call option with strike $K = 1$ at time T_1, where the network was given $\{S(T_1), c_{1,3}(T_1)\}$ in $M = 4{,}096$ simulated scenarios. Data from (a) is used to form the future exposure distribution for time T_1 seen in (b).

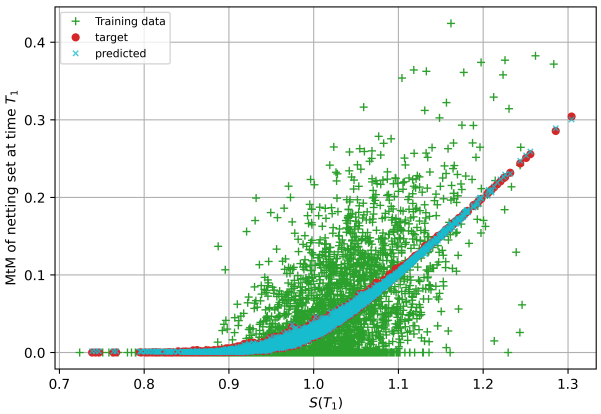
As noted in the previous paragraph, the predicted distribution reflected the target distribution more accurately when introducing an additional market observable. With this in mind, we continue to introduce market observables and now investigate results for supplying the DML framework with information of $S(T_1)$ and 3 additional market observables, where we examine two different combinations. Firstly, we have market observables $\{S(T_1), c_{0.95,3}(T_1), c_{1,3}(T_1), c_{1.05,3}(T_1)\}$, i.e. time to expiry $\tau = 3$ months is kept constant in observed call option prices and moneyness is varied for $m = 0.95, 1.00, 1.05$. Secondly, we

Market observables	**Abs. rel. dev. from $\mathbf{EE}_{6M}$**	**Abs. rel. dev. from $\mathbf{PFE}_{6M}^{0.95}$**	**Abs. rel. dev. from $\mathbf{PFE}_{6M}^{0.975}$**
$\{S(T_1), c_{1,1}(T_1)\}$	$\mathbf{-1.029}\%$	$\mathbf{-0.649}\%$	-0.472%
$\{S(T_1), c_{1,3}(T_1)\}$	-1.048%	-0.678%	$\mathbf{-0.133}\%$
$\{S(T_1), c_{1,12}(T_1)\}$	-1.030%	-0.719%	-0.176%

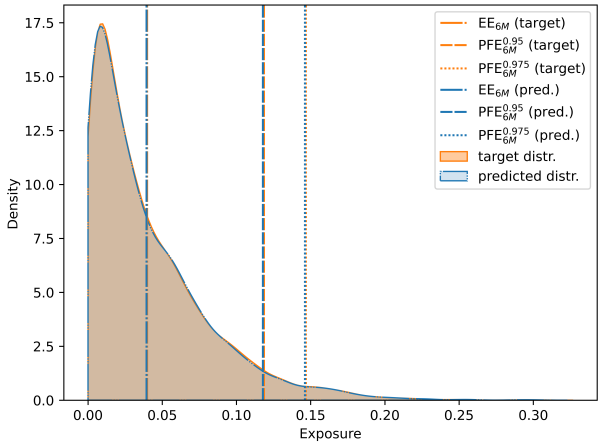
Table 4.1: Absolute relative deviation of predicted to target future exposure distribution when DML method is given three different sets of market observables.

have market observables $\{S(T_1),\, c_{1,1}(T_1),\, c_{1,3}(T_1),\, c_{1,12}(T_1)\}$, i.e. the moneyness is kept constant such that all observed call options are ATM at time T_1 and times to expiry are $\tau = 1, 3, 12$. The predictions when using these different combinations of market observables can be seen in figure 4.4. For this specific seed, we find that having additional market observables of varying moneyness produces the best prediction of EE_{6M} and $\mathrm{PFE}_{6M}^{0.95}$. Meanwhile, varying the τ in the additional market observables, whilst moneyness is kept constant, produces the most accurate prediction of $\mathrm{PFE}_{6M}^{0.975}$. Furthermore, we note that the shape of the predicted distributions continue to reflect the target distribution rather well, especially in comparison to the naïve case in figure 4.2, although predictions of EE_{6M} are less accurate in figure 4.4 than in figure 4.2. Moreover, we find that in both of cases of having 4 market observables, the differential PCA step described in section 2.4.3 successfully truncates the 4-dimensional data to 2 dimensions, as we aim to approximate the Heston call price formula of section 2.2.1, which is dependent on $S(T_1)$ and $\nu(T_1)$, i.e. is a 2-dimensional problem. We also note that this is true due to the $\mathbb{S}$- and $\mathbb{Q}$-model having the same constant model parameters.

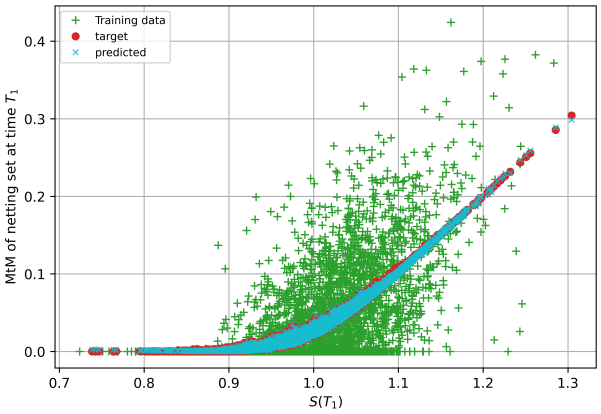
As previously stated, results have thus far been restricted to a single seed to display functionality in a specific case. We proceed to examine results as an average of 100 different seeds for all discussed combinations of input data to the DML framework, found in table 4.2. We note that when having 2 or more market observables, the differential PCA described in section 2.4.3 gives 2-dimensional data, which shows us that the differential PCA method works as intended, since we, as previously mentioned, have a 2-dimensional problem. In predicting EE_{6M}, we find that having market observables $\{S(T_1),\, c_{1,1}(T_1)\}$ tends to be the most accurate. We also see that data of market observables $\{S(T_1),\, c_{0.95,1}(T_1),\, c_{1,1}(T_1),\, c_{1.05,1}(T_1)\}$ produces relatively precise estimations. For other combinations of market observables, we find that the average relative absolute deviation is greater. Furthermore, predictions of potential future exposure is most accurate when having market observables $\{S(T_1),\, c_{1,3}(T_1)\}$. In fact, using data of these market observables to predict $\mathrm{PFE}_{6M}^{0.975}$ is the only case of greater accuracy than having model state information, where we see an improvement of 0.1 percentage points. At first glance, it may seem counter intuitive that greater accuracy can be achieved for a set of market observables than when using model state information, as the model state is sufficient to price the netting set as a whole using the Heston call price formula. However, we note that information of $\nu(T_1)$ is embedded in the market observables, as the observed option prices have the same underlying asset as the netting set and are priced using the Heston call price formula as well. Therefore, from a machine learning perspective, the same information may be presented in both cases, but in different structures. Lastly, we note that $\{S(T_1),\, c_{1,1}(T_1),\, c_{1,3}(T_1),\, c_{1,12}(T_1)\}$ is the least accurate in predicting all metrics of the distribution.



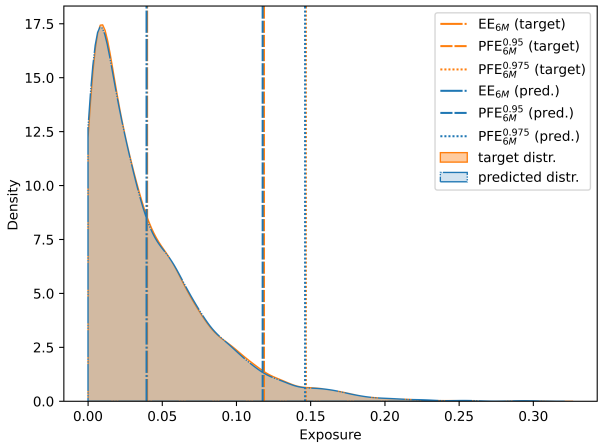
(a) Market observables $\{S(T_1),\ c_{0.95,3}(T_1),\ c_{1,3}(T_1),\ c_{1.05,3}(T_1)\}$. RMSE = 0.00066.



(b) Market observables $\{S(T_1),\ c_{0.95,3}(T_1),\ c_{1,3}(T_1),\ c_{1.05,3}(T_1)\}$. Relative deviation of EE_{6M} is -0.981%, $\mathrm{PFE}^{0.95}_{6M}$ is -0.595%, $\mathrm{PFE}^{0.975}_{6M}$ is -0.445%.



(c) Market observables $\{S(T_1),\ c_{1,1}(T_1),\ c_{1,3}(T_1),\ c_{1,12}(T_1)\}$. RMSE = 0.00071.



(d) Market observables $\{S(T_1),\ c_{1,1}(T_1),\ c_{1,3}(T_1),\ c_{1,12}(T_1)\}$. Relative deviation of EE_{6M} is -1.059%, $\mathrm{PFE}^{0.95}_{6M}$ is -0.767%, $\mathrm{PFE}^{0.975}_{6M}$ is -0.390%.

Figure 4.4: Netting set consisting of 1 call option with strike $K = 1$ at time T_1, where the network was given one set of market observables in (a) and (b), and another in (c) and (d) for $M = 4{,}096$ simulated scenarios. Figures (a) and (c) show training data (payoffs) of all M training paths, predictions on test set, and target values (MtM using the Heston call price formula) of test set. Figures (b) and (d) show target and predicted future exposure distribution, where data from (a) and (c), respectively, is used to form the future exposure distribution for time T_1.

In conclusion, supplying the DML method with market observables data seems to produce rather accurate potential future exposure predictions which are comparable to the case of model state information. Moreover, we also have several sets of market observables where predictions are considerably better than the naïve case where spot asset price of the underlying asset was the only observed market variable.

Input data	Avg. rel. abs. dev. of $\mathbf{EE}_{6M}$	Avg. rel. abs. dev. of $\mathbf{PFE}_{6M}^{0.95}$	Avg. rel. abs. dev. of $\mathbf{PFE}_{6M}^{0.975}$	Post differential PCA dim.
$\{S(T_1), \nu(T_1)\}$	**1.254**%	**1.045**%	1.163%	2
$\{S(T_1)\}$	1.610%	1.336%	1.233%	1
$\{S(T_1), c_{1,1}(T_1)\}$	1.377%	1.163%	1.198%	2
$\{S(T_1), c_{1,3}(T_1)\}$	1.612%	1.109%	**1.063**%	2
$\{S(T_1), c_{1,12}(T_1)\}$	1.583%	1.313%	1.299%	2
$\{S(T_1), c_{1,1}(T_1), c_{1,3}(T_1), c_{1,12}(T_1)\}$	1.666%	1.398%	1.398%	2
$\{S(T_1), c_{0.95,3}(T_1), c_{1,3}(T_1), c_{1.05,3}(T_1)\}$	1.476%	1.182%	1.271%	2

Table 4.2: Average relative absolute deviation of predicted to target future exposure distribution of netting set of 1 European call option when DML method is given different sets of market observables in $M = 4{,}096$ simulated scenarios. Results shown are based on an average of 100 randomly sampled seeds.

4.2 Multi-Asset Heston Model

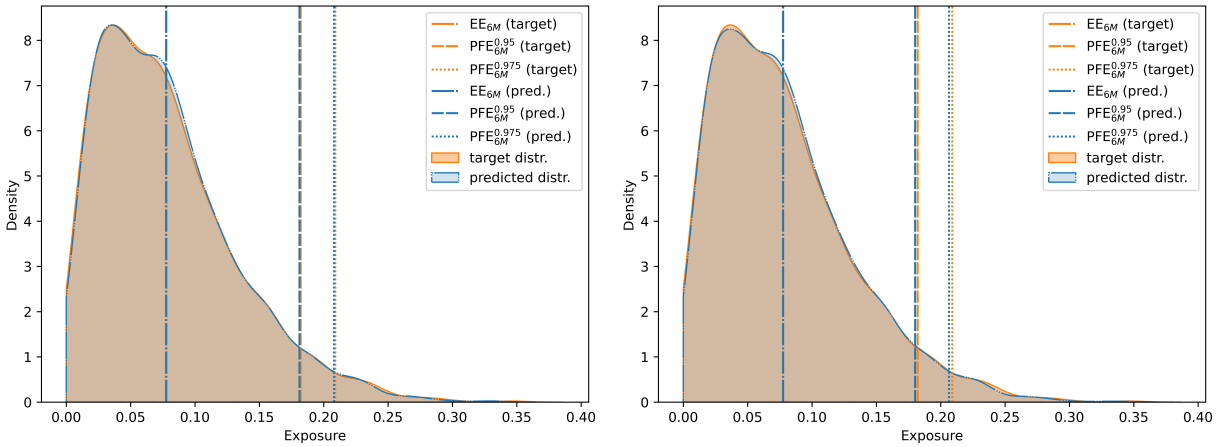
Thus far, we have only examined netting sets where a single underlying asset is modelled using the Heston model. We now generalise the example setting such that the netting set, consisting of European put and call options, depends on a larger number of underlying assets to show continued functionality of a market observables approach. In doing so, we assume that the underlying assets follow a multi-asset Heston model, as described in section 2.3. Once again, different sets of market observables are benchmarked against a model state information case and a naïve case where only spot asset price of the underlying assets is observed. We note that in a multi-asset setting, observed call option prices are evaluated for each underlying asset. Performances of the DML method for netting sets with 2, 16, 32, and 64 underlying assets are investigated to stress test the DML framework and analyse how prediction accuracy is affected. We note that by the Heston call price formula, found in section 2.2.1, the problem we aim to solve is of dimension twice the size of the number of underlying assets. Yet, while increasing the number of underlying assets, the network architecture, described in section 3.6, and number of training samples are held constant. This is done to show the effect the dimensionality of the problem has on prediction performance, where we note that the maximum training time of any network in this section was 4.4 seconds, running on a machine with 2,3 GHz Dual-Core Intel Core i5 CPU and 16 GB 2133 MHz LPDDR3 RAM.

The investigated netting sets consist of an equal number of European put and call options with different underlying assets, where each modelled assets is the underlying of a given option in the netting set. For 2 assets, the strike price is set to $K = 1$ for both options, but in all other cases $K_i \sim \mathcal{N}(1, 0.05^2)$, where K_i denotes the strike price of the put or call option corresponding to underlying asset $i = 1, \ldots, N$. As in section 4.1, options in the netting set have a time to expiry $\tau = 12$ months at time $t = 0$ and exposure date $T_1 = 6$ months into the future, meaning that all options in our netting set have a time to expiry $\tau = 6$ months at time T_1. As previously mentioned, the underlying assets are

modelled using the multi-asset Heston model, where standard parameter values shown in table 3.1 are used. The same model parameter values are used in both the $\mathbb{S}$-model and the $\mathbb{Q}$-model. Furthermore, correlation between different assets, as discussed in section 2.3, is set such that $\tilde{\rho}_{ij} = 0$ for $i,j = 1,\dots,N$, $i \neq j$. Such a setting might not be the most realistic, as different FX pairs most likely are correlated in some way. However, from a DML perspective, having no correlation is the most difficult case, as it becomes more complicated for the differential PCA to truncate dimensions. A total of $M = 4{,}096$ scenarios are evaluated in all cases, where initial values of the spot asset price process and the variance process are set to $S_i(0) = 1$ and $\nu_i(0) = 0.01$, respectively, for all assets $i = 1,\dots,N$.

4.2.1 Netting set dependent on 2 underlying assets

We begin by examining a netting set consisting of 1 European call option and 1 European put option with 2 separate underlying assets, both with strike $K_1 = K_2 = 1$. In figure 4.5, we see the predicted future exposure distribution for two combinations of market observables $\{S_i(T_1), c^i_{1,1}(T_1) \mid i = 1{,}2\}$, and $\{S_i(T_1), c^i_{1,12}(T_1) \mid i = 1{,}2\}$ for the same fixed randomly selected seed. In comparison to estimated distributions found in section 4.1, such as figure 4.4b, we find that the target exposure distribution has been shifted rightwards. The reason is that the netting set now consists of 2 European options, rather than 1, i.e. in addition to the MtM of the call option, there is a positive MtM of the put option in this netting set. Thus, the netting set will have a higher MtM than that consisting of a single call option, as found in section 4.1. Furthermore, figure 4.5a and 4.5b show one of



(a) Market observables $\{S_i(T_1), c^i_{1,1}(T_1) \mid i = 1{,}2\}$, relative deviation of EE_{6M} is 0.305%, $\text{PFE}^{0.95}_{6M}$ is −0.527%, $\text{PFE}^{0.975}_{6M}$ is −0.463%.

(b) Market observables $\{S_i(T_1), c^i_{1,12}(T_1) \mid i = 1{,}2\}$, relative deviation of EE_{6M} is 0.275%, $\text{PFE}^{0.95}_{6M}$ is −1.070%, $\text{PFE}^{0.975}_{6M}$ is −1.248%.

Figure 4.5: Target and predicted future exposure distribution for netting set consisting of 1 European call option with strike $K = 1$, and 1 European put option with strike $K = 1$ with different underlying assets. Exposure estimation performed at time T_1 derived from $M = 4{,}096$ simulated scenarios.

the most and one of the least accurate predictions of the future exposure distribution, respectively, for a fixed seed. When comparing the two figures, there are substantial differences from a visual perspective, although the predicted distribution in figure 4.5b does not seem to follow the target distribution quite as well around its mode in comparison to that seen in figure 4.5a.

From a quantitative perspective, the differences are primarily seen in the tail metrics, namely the $\text{PFE}_{6M}^{0.95}$ and $\text{PFE}_{6M}^{0.975}$. In fact, the EE_{6M} prediction is more precise in figure 4.5b than in 4.5a, although the difference is rather small. The reason for the difference between the two figures lies in the information embedded in $c_{1,1}^i$ and $c_{1,12}^i$. Call option prices with a shorter time to maturity is more dependent on the current state of the variance process $\nu_i(T_1)$, and, as seen in the Heston call price formula of section 2.2.1, this in turn affects the value of the netting set at time T_1. Meanwhile, having observed call option prices further into the future tells us more about the long-term mean of the variance process θ_i, as defined in the multi-asset Heston model (2.5). We also note that accurate results are produced in this case as a consequence of having the same $\mathbb{S}$- and $\mathbb{Q}$-model parameters, as this implies identical characteristics of the spot asset price and variance processes under both models.

We now proceed past the case brought up in figure 4.5, where results for a specific seed were investigated. In table 4.3, prediction accuracies of EE_{6M}, $\text{PFE}_{6M}^{0.95}$, and $\text{PFE}_{6M}^{0.975}$ with the same netting set, but for a larger variety of market observables sets, are displayed.

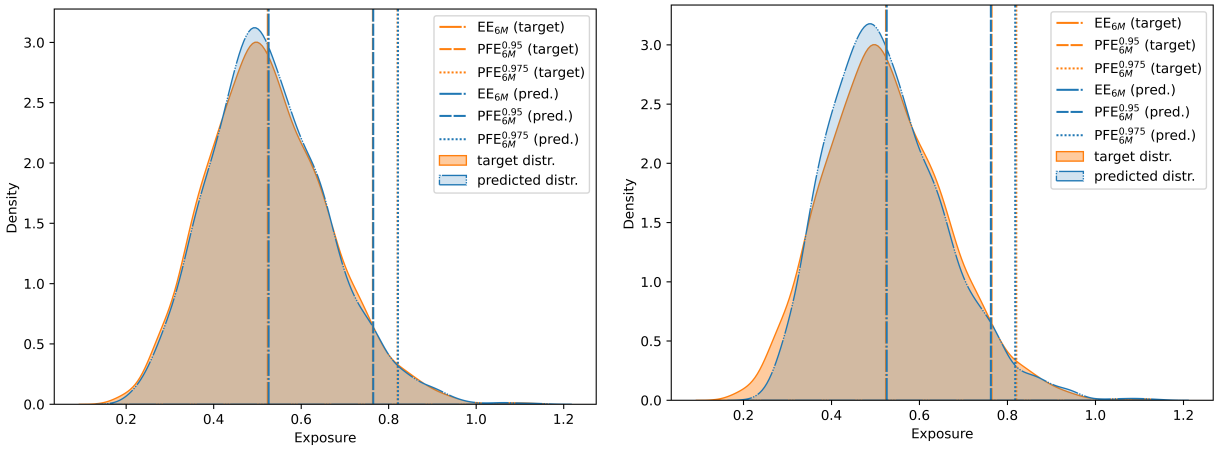
Input data	**Avg. rel. abs. dev. from EE_{6M}**	**Avg. rel. abs. dev. from $\text{PFE}_{6M}^{0.95}$**	**Avg. rel. abs. dev. from $\text{PFE}_{6M}^{0.975}$**	**Post differential PCA dim.**
$\{S_i(T_1)$, $\nu_i(T_1)$, $i = 1,2\}$	0.967%	0.827%	0.913%	4
$\{S_i(T_1)$, $i = 1,2\}$	1.045%	0.903%	**0.849**%	2
$\{S_i(T_1)$, $c_{1,1}^i(T_1)$, $i = 1,2\}$	**0.954**%	0.842%	0.858%	4
$\{S_i(T_1)$, $c_{1,3}^i(T_1)$, $i = 1,2\}$	1.066%	0.875%	0.872%	4
$\{S_i(T_1)$, $c_{1,12}^i(T_1)$, $i = 1,2\}$	1.021%	0.855%	0.894%	4
$\{S_i(T_1)$ $c_{1,1}^i(T_1)$, $c_{1,3}^i(T_1)$, $c_{1,12}^i(T_1)$, $i = 1,2\}$	1.021%	0.852%	0.940%	4
$\{S_i(T_1)$ $c_{0.95,3}^i(T_1)$, $c_{1,3}^i(T_1)$, $c_{1.05,3}^i(T_1)$, $i = 1,2\}$	0.995%	**0.806**%	0.869%	4

Table 4.3: Deviation of predicted future exposure distribution from target future exposure distribution of netting set of 2 options, 1 European put option and 1 European call option, when giving various levels of information of market observables in $M = 4{,}096$ simulated scenarios. Results are shown from a randomly sampled selection of 100 different seeds.

The values are shown as an average of 100 randomly selected seeds in order to determine general characteristics. We begin by noting that the naïve market observables set $\{S_i(T_1) \mid i = 1,2\}$ produces the least accurate predictions of EE_{6M} and $\text{PFE}_{6M}^{0.95}$, whilst being the most accurate for $\text{PFE}_{6M}^{0.975}$, although not by much. Furthermore, notably, $\{S_i(T_1),\, c^i_{1,1}(T_1) \mid i = 1,2\}$ predicts EE_{6M} the most accurately, and $\{S_i(T_1),\, c^i_{0.95,3}(T_1), c^i_{1,3}(T_1), c^i_{1.05,3}(T_1) \mid i = 1,2\}$ produces the most accurate prediction of $\text{PFE}_{6M}^{0.95}$, both of which are more precise than corresponding predictions when supplying the DML framework with model state information. With regard to $\text{PFE}_{6M}^{0.975}$, all market observable combinations, except $\{S_1(T_1), S_2(T_1), c^i_{1,1}(T_1), c^i_{1,3}(T_1), c^i_{1,12}(T_1) \mid i = 1,2\}$, produce more accurate estimations of the $\text{PFE}_{6M}^{0.975}$ than model state information. We also note that for sets of market observables with a cardinality greater than 4, differential PCA has truncated the data to 4 dimensions, which shows that differential PCA works as intended, as a 4-dimensional problem is being solved. In general, when observing table 4.3, we find that the accuracies in metric predictions are somewhat consistent for all types of input data. With that, we may state that market observables are informative enough to produce estimations on a level similar to that of model state information for a netting set of 2 options, whilst performing better than the naïve approach, despite a discrepancy in $\text{PFE}_{6M}^{0.975}$ predictions.

4.2.2 Netting set dependent on 16 underlying assets

The complexity of the netting set is now increased further to be dependent on 16 underlying assets, namely 8 European put options and 8 European call options with strike price $K_i \sim \mathcal{N}(1, 0.05^2)$, $i = 1, \ldots, 16$. In figure 4.6, we see the future exposure distribution



(a) Market observables $\{S_i(T_1),\, c^i_{1,1}(T_1) \mid i = 1, \ldots, 16\}$, relative deviation of EE_{6M} is 0.369%, $\text{PFE}_{6M}^{0.95}$ is 0.115%, $\text{PFE}_{6M}^{0.975}$ is 0.138%.

(b) Market observables $\{S_i(T_1),\, c^i_{1,1}(T_1),\, c^i_{1,3}(T_1),\, c^i_{1,12}(T_1) \mid i = 1, \ldots, 16\}$, relative deviation of EE_{6M} is 0.320%, $\text{PFE}_{6M}^{0.95}$ is −0.205%, $\text{PFE}_{6M}^{0.975}$ is −0.295%.

Figure 4.6: Target and predicted distributions for netting set consisting of 16 options, 8 European put options and 8 European call options, with strike prices $K_i \sim \mathcal{N}(1, 0.05^2)$, $i = 1, \ldots, 16$, and different underlying assets. Exposure estimation performed at time T_1 derived from 4,096 simulated scenarios.

prediction when supplying the DML method with 2 different combinations of market observables for a fixed seed. Once again, we find that the target exposure distribution has been shifted further rightwards, compared to that seen in figure 4.5, due to an increased number of options in the netting set. In observing figures 4.6a and 4.6b, we see that the predicted distribution has a higher density around the mode than the target distribution in both cases. However, figure 4.6b shows a predicted distribution with a greater positive skew. We see that this results in underestimated values of $\text{PFE}_{6M}^{0.95}$ and $\text{PFE}_{6M}^{0.975}$, whereas the corresponding predicted values are more precise, but overestimated, in figure 4.6a.

In table 4.4, we find the average relative absolute deviation of EE_{6M}, $\text{PFE}_{6M}^{0.95}$, and $\text{PFE}_{6M}^{0.975}$ between the predicted and target distributions in 15 randomly selected seeds for different sets of market observables. We begin by noting that the most accurate prediction of EE_{6M} is produced by the set $\{S_i(T_1), c^i_{1,3}(T_1) | i = 1, \ldots, 16\}$. On the contrary to previously presented results, the most accurate predictions of potential future exposure is here produced by the set $\{S_i(T_1) | i = 1, \ldots, 16\}$. At this point, we note that that the dimension of input data to the network is 32 in all cases, apart from the naïve approach, the reason being that we are dealing with a 32-dimensional problem. The higher the dimension of the problem, the more training samples are required, which is an established problem known as the *curse of dimensionality*, stating that the number of samples required to estimate an arbitrary function while maintaining the same level of accuracy grows exponentially

Input data	**Avg. rel. abs. dev. from EE_{6M}**	**Avg. rel. abs. dev. from $\text{PFE}_{6M}^{0.95}$**	**Avg. rel. abs. dev. from $\text{PFE}_{6M}^{0.975}$**	**Post differential PCA dim.**
$\{S_i(T_1),\ \nu_i(T_1),\ i = 1, \ldots, 16\}$	0.358%	1.073%	1.512%	32
$\{S_i(T_1),\ i = 1, \ldots, 16\}$	0.396%	**0.539**%	**0.648**%	16
$\{S_i(T_1),\ c^i_{1,1}(T_1),\ i = 1, \ldots, 16\}$	0.356%	0.872%	0.883%	32
$\{S_i(T_1),\ c^i_{1,3}(T_1),\ i = 1, \ldots, 16\}$	**0.258**%	0.855%	1.073%	32
$\{S_i(T_1),\ c^i_{1,12}(T_1),\ i = 1, \ldots, 16\}$	0.400%	0.903%	1.111%	32
$\{S_i(T_1),\ c^i_{1,1}(T_1),\ c^i_{1,3}(T_1),\ c^i_{1,12}(T_1),\ i = 1, \ldots, 16\}$	0.501%	0.844%	1.044%	32
$\{S_i(T_1),\ c^i_{0.95,3}(T_1),\ c^i_{1,3}(T_1),\ c^i_{1.05,3}(T_1),\ i = 1, \ldots, 16\}$	0.390%	0.937%	1.166%	32

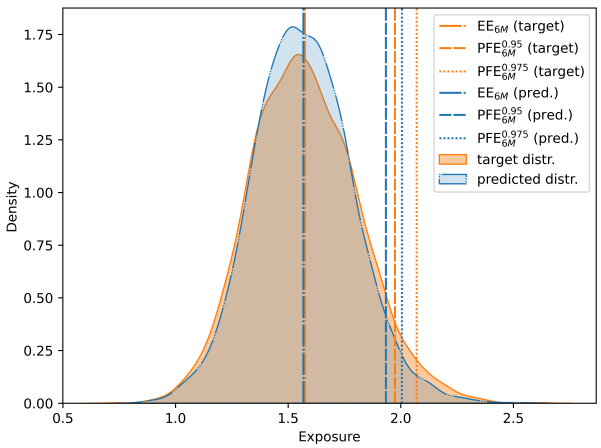
Table 4.4: Deviation of predicted future exposure distribution from target future exposure distribution of netting set of 16 options, 8 European put option and 8 European call option, when giving various levels of information of market observables in $M = 4{,}096$ simulated scenarios. Results are shown from a randomly sampled selection of 15 different seeds.

with respect to the number of features in input data [4]. However, the number of training samples is kept constant at 4,096 samples, which is why, for higher dimensionality of the data, we see the decreased accuracy in the tail predictions where training data is scarcer. Meanwhile, the predictions of EE_{6M} are somewhat consistent with only a couple of outliers. This shows that the dimensionality of the problem versus the number of training samples is not necessarily an issue affecting accuracy of the mean of the predicted distribution. Notably, we find that the set $\{S_i(T_1), \nu_i(T_1) \mid i = 1, \ldots, 16\}$ produces the least accurate predictions of potential future exposure, whilst being somewhat accurate in EE_{6M} predictions, suggesting that the DML method trains more easily on the market observables data than market state information in this case.

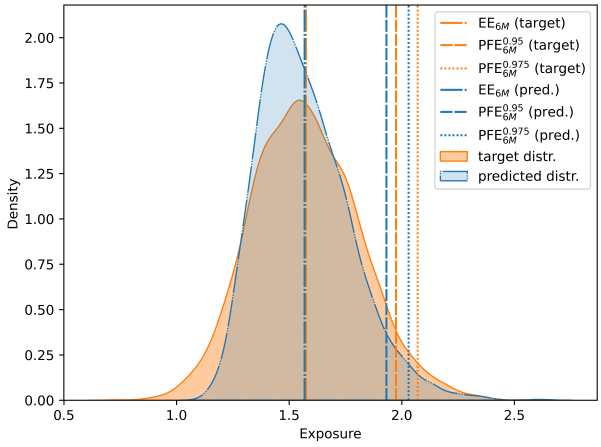
4.2.3 Netting set dependent on 32 underlying assets

We now proceed to investigate a netting set dependent on 32 underlying assets, consisting of 16 European put options and 16 European call options with strike price $K_i \sim \mathcal{N}(1, 0.05^2)$ for $i = 1, \ldots, 16$. Firstly, the target and predicted future exposure distributions of 2 sets of market observables for a fixed, randomly selected seed is shown in figure 4.7. Again, we note that the target distribution has been shifted further rightwards compared to the previously shown results due to an increased number of options in the netting set. At first glance, we see that figure 4.7a seems to show a prediction that replicates the target distribution more accurately overall than that of figure 4.7b, where we see an apparent positive skew and greater density around the mode of the predicted distribution with clear underestimation in the tails. However, the predicted distribution of figure 4.7b produces the more precise estimations of both EE_{6M} and $\mathrm{PFE}_{6M}^{0.975}$. This is partly explained by the predicted distribution in figure 4.7a in turn underestimating the density in the right tail, leading to an underestimation of the potential future exposure metrics.

Once again, we proceed to a more general case for different combinations of market observables and a larger number of randomly selected seeds for the same netting set, the results of which can be found in table 4.5. As the netting set is dependent on 32 underlying assets, we are dealing with a 64-dimensional problem. However, the differential PCA scheme now truncates the raw data of the market state set to 59 dimensions in all cases. For combinations including observed call option prices, the raw data is projected to slightly lower dimensions, where we note that the number of dimensions post differential PCA is not constant for any of the observed sets. As discussed in section 2.4.3, differential PCA truncates dimensions based on information contribution. As seen in table 4.5, it seems a greater number of dimensions are informative in the model state set, whereas it is slightly lower in the sets containing observed call option prices. Meanwhile, no dimensions are truncated when applying differential PCA to the naïve set. Moreover, upon investigating the EE_{6M} predictions for all sets of input data, we see a somewhat consistent accuracy, where the most accurate predictions are made when input data is $\{S_i(T_1),\ c^i_{1,1},\ c^i_{1,3},\ c^i_{1,12} | i = 1, \ldots, 32\}$, and the least accurate are made when using the naïve set $\{S_i(T_1) | i = 1, \ldots, 32\}$. However, we find that the naïve set is the most accurate in predicting the potential future exposures, where all other sets of input data seemingly struggle. As previously discussed, this is due to a lack of training samples combined with a scarcity of training data in the tails of the distribution.



(a) Market observables $\{S_i(T_1),\, c^i_{1,1} \mid i = 1,\ldots,32\}$, relative deviation of EE_{6M} is -0.452%, $\mathrm{PFE}^{0.95}_{6M}$ is -2.035%, $\mathrm{PFE}^{0.975}_{6M}$ is -3.180%.



(b) Market observables $\{S_i(T_1),\, c^i_{0.95,3}(T_1),\, c^i_{1,3}(T_1),\, c^i_{1.05,3}(T_1) \mid i = 1,\ldots,32\}$, relative deviation of EE_{6M} is -0.382%, $\mathrm{PFE}^{0.95}_{6M}$ is -2.158%, $\mathrm{PFE}^{0.975}_{6M}$ is -1.942%.

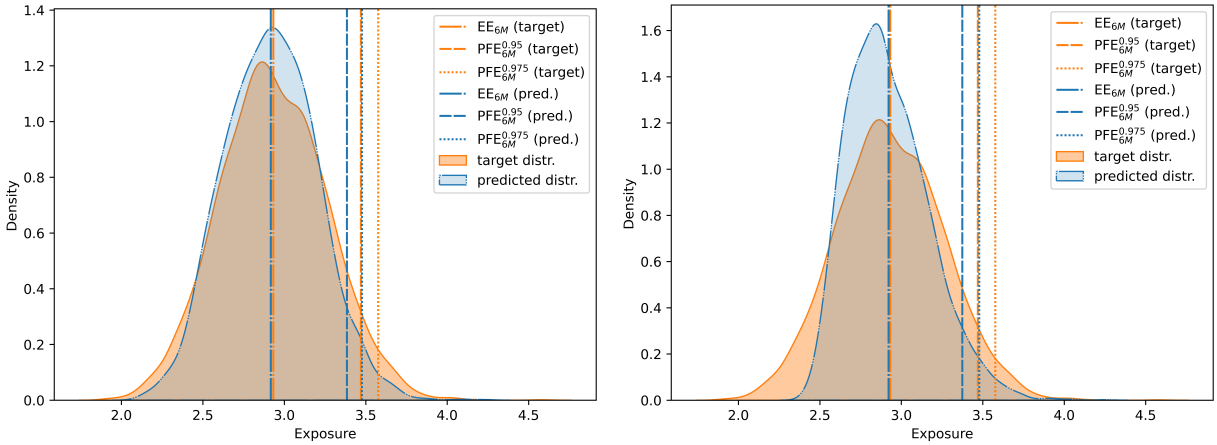
Figure 4.7: Target and predicted distributions for netting set consisting of 32 options, 16 European put options and 16 European call options, with strike prices $K_i \sim \mathcal{N}(1{,}0.05^2)$, $i = 1,\ldots,32$, and different underlying assets. Exposure estimation performed at time T_1 derived from 4,096 simulated scenarios.

Input data	**Avg. rel. abs. dev. from EE_{6M}**	**Avg. rel. abs. dev. from $\mathrm{PFE}^{0.95}_{6M}$**	**Avg. rel. abs. dev. from $\mathrm{PFE}^{0.975}_{6M}$**	**Avg. post differential PCA dim.**
$\{S_i(T_1),\, \nu_i(T_1),$ $i = 1,\ldots,32\}$	0.265%	2.287%	2.534%	59.00
$\{S_i(T_1),$ $i = 1,\ldots,32\}$	0.283%	$\mathbf{0.802}\%$	$\mathbf{0.899}\%$	32.00
$\{S_i(T_1),\, c^i_{1,1}(T_1),$ $i = 1,\ldots,32\}$	0.197%	1.895%	2.038%	57.73
$\{S_i(T_1),\, c^i_{1,3}(T_1),$ $i = 1,\ldots,32\}$	0.212%	2.366%	2.770%	57.13
$\{S_i(T_1),\, c^i_{1,12}(T_1),$ $i = 1,\ldots,32\}$	0.210%	2.242%	2.600%	57.67
$\{S_i(T_1),\, c^i_{1,1}(T_1),$ $c^i_{1,3}(T_1),\, c^i_{1,12}(T_1),$ $i = 1,\ldots,32\}$	$\mathbf{0.168}\%$	2.049%	2.473%	57.47
$\{S_i(T_1),\, c^i_{0.95,3}(T_1),$ $c^i_{1,3}(T_1),\, c^i_{1.05,3}(T_1),$ $i = 1,\ldots,32\}$	0.199%	1.969%	1.732%	57.53

Table 4.5: Deviation of predicted future exposure distribution from target future exposure distribution of netting set of 32 options, 16 European put options and 16 European call options, when giving various levels of information of market observables in $M = 4{,}096$ simulated scenarios. Results are shown from a randomly sampled selection of 15 different seeds.

4.2.4 Netting set dependent on 64 underlying assets

We have seen the DML method having difficulties in making predictions with the increasing number of underlying assets and dimensions. One last time, we increase the netting complexity further to be dependent on a total of 64 underlying assets, namely 32 European put options and 32 European call options. By using the multi-asset Heston model, this implies that we have a 128-dimensional problem at hand. In figure 4.8, we find predictions of the future exposure distribution of 2 different market observables sets. In both cases, we find that the predicted distribution densities are underestimated in the tails and are more centred around the mean in comparison to the target distribution. Furthermore, figure 4.8b shows a predicted distribution with a clear overestimated density in the mode and a positive skew, whilst figure 4.8a shows an overestimation of the density in the mode and a slight negative skew. Although the predicted distributions look different, the $\text{PFE}_{6M}^{0.975}$ is approximately the same in both cases. Furthermore, the $\text{PFE}_{6M}^{0.95}$ is slightly more accurate in figure 4.8a, although not by much, and the EE_{6M} prediction is more accurate in figure 4.8b.



(a) Market observables $\{S_i(T_1), c^i_{0.95,3}(T_1), c^i_{1,1}(T_1), c^i_{1.10,3}(T_1) \mid i = 1,\ldots,64\}$, relative deviation of EE_{6M} is -0.527%, $\text{PFE}_{6M}^{0.95}$ is -2.455%, $\text{PFE}_{6M}^{0.975}$ is -2.784%.

(b) Market observables $\{S_i(T_1), c^i_{1,1}(T_1), c^i_{1,3}(T_1), c^i_{1,12}(T_1) \mid i = 1,\ldots,64\}$, relative deviation of EE_{6M} is -0.364%, $\text{PFE}_{6M}^{0.95}$ is -2.750%, $\text{PFE}_{6M}^{0.975}$ is -2.740%.

Figure 4.8: Target and predicted distributions for netting set consisting of 64 options, 32 European call options and 32 European put options, all with different underlying assets, where strike prices $K \sim \mathcal{N}(1{,}0.05^2)$. Exposure estimation performed at time T_1 derived from 4,096 simulated scenarios.

Again, we proceed to a more general case for a wider selection of market observables sets for a greater number of randomly selected seeds to investigate general tendencies in prediction accuracies, the results of which can be found in table 4.6. Firstly, we find that no dimensions are truncated in the naïve set. However, we find that differential PCA on average truncates data sets containing observed option prices to dimensions in the interval [85,89]. The number of informative dimensions tend to be the largest for the market state set, where input data to the twin network has an average dimension of 97.53. As previously discussed, the relatively high-dimensional data in turn results in low accuracies

of potential future exposure predictions, which is evident in table 4.6. We also find that the accuracy of $\text{PFE}_{6M}^{0.95}$ and $\text{PFE}_{6M}^{0.975}$ predictions is rather low when using sets containing observed call option prices, although it is comparable to the results previously seen in table 4.5. The naïve set is once again the most accurate in predicting the potential future exposure. Furthermore, in predicting the EE_{6M}, the naïve set is the most precise out of the sets containing market observable data only, although $\{S_i(T_1),\, c^i_{1,12}(T_1) \mid i = 1, \ldots, 64\}$ only differs by 0.015 percentage points.

Input data	**Avg. rel. abs. dev. from EE_{6M}**	**Avg. rel. abs. dev. from $\text{PFE}_{6M}^{0.95}$**	**Avg. rel. abs. dev. from $\text{PFE}_{6M}^{0.975}$**	**Avg. post differential PCA dim.**
$\{S_i(T_1),\, \nu_i(T_1),$ $i = 1, \ldots, 64\}$	**0.197**%	4.020%	4.327%	97.53
$\{S_i(T_1),$ $i = 1, \ldots, 64\}$	0.229%	**1.140**%	**1.050**%	64.00
$\{S_i(T_1),\, c^i_{1,1}(T_1),$ $i = 1, \ldots, 64\}$	0.260%	2.605%	2.779%	85.53
$\{S_i(T_1),\, c^i_{1,3}(T_1),$ $i = 1, \ldots, 64\}$	0.284%	2.849%	2.945%	86.27
$\{S_i(T_1),\, c^i_{1,12}(T_1),$ $i = 1, \ldots, 64\}$	0.244%	2.712%	2.947%	89.00
$\{S_i(T_1),\, c^i_{1,1}(T_1),$ $c^i_{1,3}(T_1),\, c^i_{1,12}(T_1),$ $i = 1, \ldots, 64\}$	0.263%	2.785%	2.904%	86.53
$\{S_i(T_1),\, c^i_{0.95,3}(T_1),$ $c^i_{1,3}(T_1),\, c^i_{1.05,3}(T_1),$ $i = 1, \ldots, 64\}$	0.284%	2.597%	2.696%	88.13

Table 4.6: Deviation of predicted future exposure distribution from target future exposure distribution of netting set of 64 options, 32 European put options and 32 European call options, when giving various levels of information of market observables in $M = 4{,}096$ simulated scenarios. Results are shown from a randomly sampled selection of 15 different seeds.

In summary, we find that sets of market observables produced predictions comparable to, and at times more accurate than, that of market state information for netting sets consisting of up to 16 options. For netting sets of 1 or 2 options, the predictions of the distribution metrics were in general as precise or more accurate than predictions when supplying the DML framework with observed spot asset prices, i.e. the naïve set. For a netting set of 16 options, the prediction of EE_{6M} was comparable between sets including observed call option prices and the naïve set. However, potential future exposure predictions stemming from the naïve set are clearly superior as the DML method struggles to produce valuable predictions due to increased dimensionality. The netting set of 16 options presented a breaking point of the DML method in the case of $M = 4096$, and the issue of insufficient training data was emphasised for netting sets of 32 and 64 options, as the curse of dimensionality becomes increasingly tangible, where predictions of potential future exposure deviated further from the target distribution and the predictions produced by the naïve set. At first glance, it may be hypothesised that the issue of

large dimensions could be solved by a differential PCA scheme truncating dimensions to achieve a predetermined number of dimension, rather than truncating data based on thresholds ε_1 and ε_2. However, this would results in the DML framework overall losing a large proportion of its flexibility and poses additional questions. Nevertheless, predictions of EE_{6M} remained somewhat precise even for the larger netting sets, where all other sets of market observables outperformed the naïve set for the netting set of 32 options, and predictions were comparable for the case of the 64 options netting set.

4.3 Different $\mathbb{S}$- and $\mathbb{Q}$-models

In this section, we apply the proposed exposure modelling approach to settings where $\mathbb{S}$- and $\mathbb{Q}$-models are different. This is highly relevant for risk management purposes and is one of the main proposed advantages of the DML approach. We examine the capability of the approach to cope with different $\mathbb{Q}$- and $\mathbb{S}$-models, although both of Heston type, by considering 1,000 different $\mathbb{Q}$-models and a fixed $\mathbb{S}$-model. The parameter values for the 1,000 different $\mathbb{Q}$-models are generated as described in section 3.7. The parameter values for the v are then set as an average of the $\mathbb{Q}$-model parameter values. For the given draw used in this section, this led to parameter values for the $\mathbb{S}$-model as seen in table 4.7.

Parameter	**Value**
Mean reversion $\kappa^{\mathbb{S}}$	4.9
Long-run average variance $\theta^{\mathbb{S}}$	0.061
Correlation of Wiener processes in price and variance processes $\rho^{\mathbb{S}}$	-0.45
Volatility of variance process $\xi^{\mathbb{S}}$	0.37

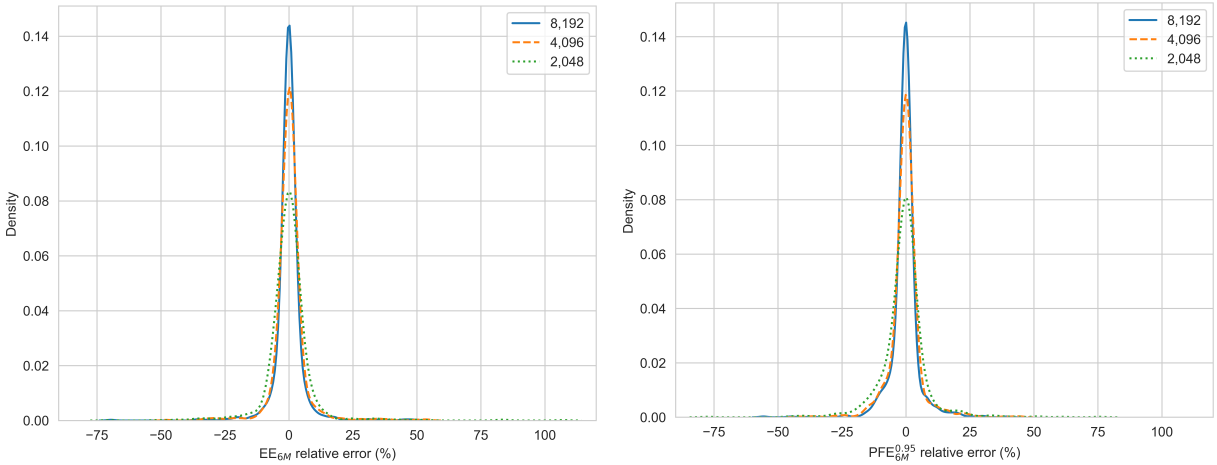
Table 4.7: Parameter values in the $\mathbb{S}$-model used in section 4.3.

The exposure of a netting set consisting of a European call option with a single underlying asset is modelled using all different parameter combinations of the $\mathbb{Q}$-model. The reason to have a relatively simple netting set is to speed up simulation times and to focus the analysis on the difference between the $\mathbb{S}$- and $\mathbb{Q}$-models. The option in our netting set has a strike price $K = 1$ and time of expiry is $T = 1$ year. The exposure date T_1 is 6 months into the future, meaning that the option in our netting set will have a time to expiry of 6 months at time T_1. Initial values of the spot asset price process and the variance process are set to $S(0) = 1$ and $\nu(0) = \theta$, respectively, where θ is model specific. Apart from the spot asset price $S(t)$, the input data consists of three ATM call options with times to expiry $\tau = 1, 6, 12$ months. As previously discussed, these are generated by the Heston call option formula as described in section 2.2.1. The reason we present results of only this choice of market observables is that our goal is not to find the best market observables to use in this specific example setting, but to examine how the differences in the $\mathbb{S}$- and $\mathbb{Q}$-model affect the prediction accuracies.

4.3.1 Effect of Training Set Size on Accuracy

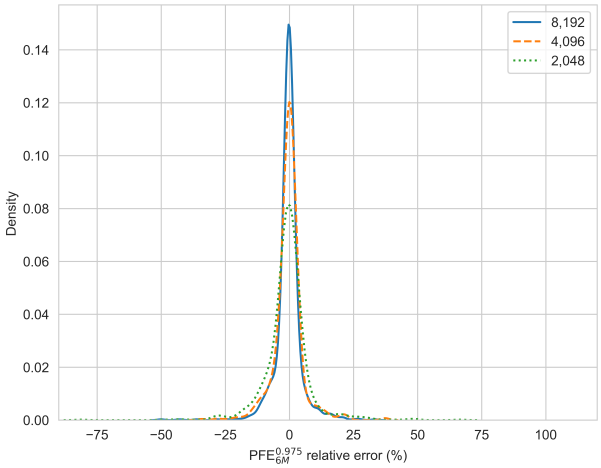
The size of the training set affects the accuracy of the DML method, which is extensively studied in the work of Huge and Savine [16]. To show the effect of altering the training set size on the accuracy of exposure modelling using DML, the exposure of the netting set explained previously in this section is estimated for the 1,000 different $\mathbb{Q}$-models, with varying training set size $M = 2{,}048$, $M = 4{,}096$, and $M = 8{,}192$, respectively. The target exposure distributions that the predictions are compared to, come from pricing in the same $\mathbb{S}$-scenarios by using the known $\mathbb{S}$-model and Heston call price formula.

In figure 4.9, we see that the method performance improves, in terms of accuracy of metrics of the predicted distribution, when the number of training samples increase, as one can expect. As seen in the previous results presented in sections 4.1 and 4.2, the errors of the



(a) EE_{6M} relative error

(b) $\text{PFE}_{6M}^{0.95}$ relative error



(c) $\text{PFE}_{6M}^{0.975}$ relative error

Figure 4.9: Each curve in the three plots above show a Gaussian kernel density estimation of the three different sets of results for varying M, where each set corresponds to prediction accuracies using 1,000 different $\mathbb{Q}$-models. The `seaborn` package in Python is used for the kernel density estimation.

expected exposure and the percentiles are not necessarily correlated. However, the errors of $\text{PFE}_{6M}^{0.95}$ and $\text{PFE}_{6M}^{0.975}$ do seem to be highly correlated. With this in mind, we henceforth only consider results of EE_{6M} and $\text{PFE}_{6M}^{0.95}$.

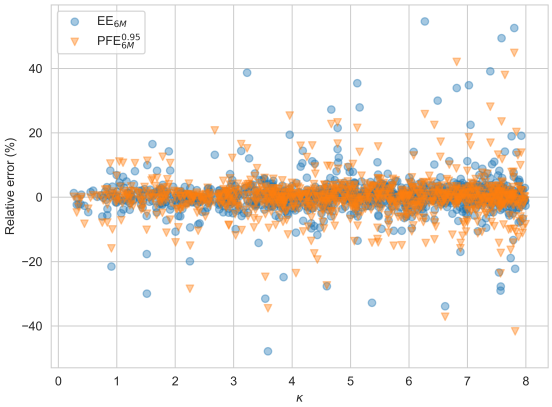
Due to constraints on time and computational power, we do not increase the amount of training samples more than $M = 8{,}192$ for all 1,000 $\mathbb{Q}$-models. However, it may be speculated that not all 1,000 cases converge to zero due to the large discrepancies between certain $\mathbb{Q}$-models and the $\mathbb{S}$-model. An example indicating this is the $\mathbb{Q}$-model with $\kappa = 3.23$, $\theta = 0.0184$, $\xi = 0.0267$, and $\rho = -0.573$. Using a very large training set of size $M = 1{,}048{,}576$, we get a relative error of the EE_{6M} at 14%, $\text{PFE}_{6M}^{0.95}$ at -1.1%, and $\text{PFE}_{6M}^{0.975}$ at -2.2%. These are large errors considering the very large training set, suggesting a non-zero convergence of the error for increasing training set size.

4.3.2 Effect of Parameter Values on Accuracy

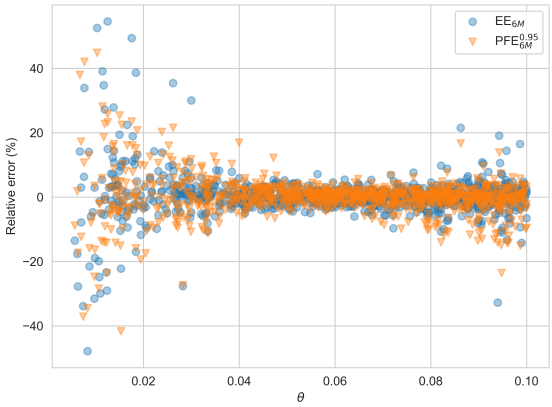
Continuing with a training set size of $M = 4{,}096$, we will try to identify what characterises the cases where the method performance is satisfactory as well as unsatisfactory. In figure 4.10, we find accuracy of prediction metrics using the 1,000 different $\mathbb{Q}$-model. In observing figure 4.10c, we start by noting that the correlation between the Wiener processes in the price and variance processes, ρ, appears to be uncorrelated with the size of the relative error of the two metrics shown. For the three remaining parameters that are varied, the mean reversion κ, the long-run average variance θ, and the volatility of the variance ξ, we can more easily see where the predictions are less accurate, by studying the results in figure 4.10a, 4.10b, and 4.10d, respectively. Further details of accuracy of EE_{6M} predictions for all combinations of κ, θ, and ξ may be seen in figures A.1 and A.2 in appendix A. Firstly, it is important to note that since the Feller condition, seen in (2.2), is imposed in the sampling of parameter combinations, we see relatively few examples at small θ, small κ and large ξ. One can see that the largest relative absolute errors have a low value of θ, and thus a low value of ξ. For θ there is also a tendency for larger errors in both ends of the sampled interval, when the value differs from $\theta^{\mathbb{S}} = 0.06$. The clearest trend is that of ξ, where the relative absolute error generally decreases with increasing ξ. Note that large ξ implies a large value of the product of θ and κ, due to the Feller condition. What is interesting to note, however, is that only deviations from the $\mathbb{S}$-model parameter values in certain directions have a negative average effect on the performance. We can also see a tendency of greater relative absolute error when increasing κ, although this may be due to an increased abundance of parameter value combinations with larger κ, compared to smaller values, due to the Feller condition. Lastly, we note that relatively high accuracy can be found for $\mathbb{Q}$-models with parameter values in the general vicinity of the $\mathbb{S}$-model, as seen in figures A.1 and A.2 in appendix A.

It is not surprising that the values of ξ and θ have such a large impact as these are important parameters of the Heston call price formula. Furthermore, ξ denotes the volatility of the variance process and the lowest values of ξ are 0.01, while the $\mathbb{S}$-model has $\xi^{\mathbb{S}} = 0.37$. This means that for the lowest values of ξ, we have considerably smaller fluctuations in the variance process compared to what is seen in the $\mathbb{S}$-model, which could explain the observed trends.

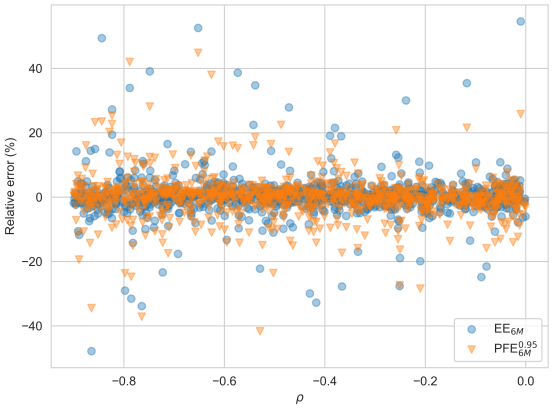
Lastly, we note that some parameter value combinations for the $\mathbb{Q}$-model may be unrealistic



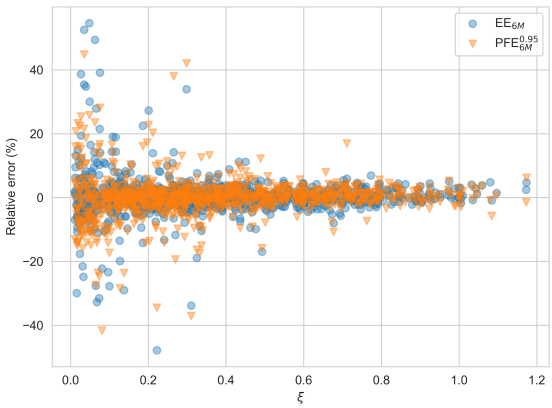
(a) Varying κ where $\kappa^{\mathbb{S}} = 4.9$



(b) Varying θ where $\theta^{\mathbb{S}} = 0.061$



(c) Varying ρ where $\rho^{\mathbb{S}} = -0.45$



(d) Varying ξ where $\xi^{\mathbb{S}} = 0.37$

Figure 4.10: Display of the relative error of EE_{6M} and $\text{PFE}_{6M}^{0.95}$ against the $\mathbb{Q}$-model's (a) κ, (b) θ, (c) ρ, and (d) ξ. The $\mathbb{S}$-model parameter values are denoted by $\kappa^{\mathbb{S}}$, $\theta^{\mathbb{S}}$, $\rho^{\mathbb{S}}$, and $\xi^{\mathbb{S}}$, respectively.

and affect accuracy. The reason for the unrealistic combinations stems from how these are generated. Firstly, only 4 parameters of the Heston model are varied. Secondly, uniform sampling of the parameter values, as explained in section 3.7.1, may also lead to unrealistic parameter value combinations. This is because certain parameter values seem to be highly correlated in calibration, as seen in Winter [27], which has not been taken into account in section 3.7.1.

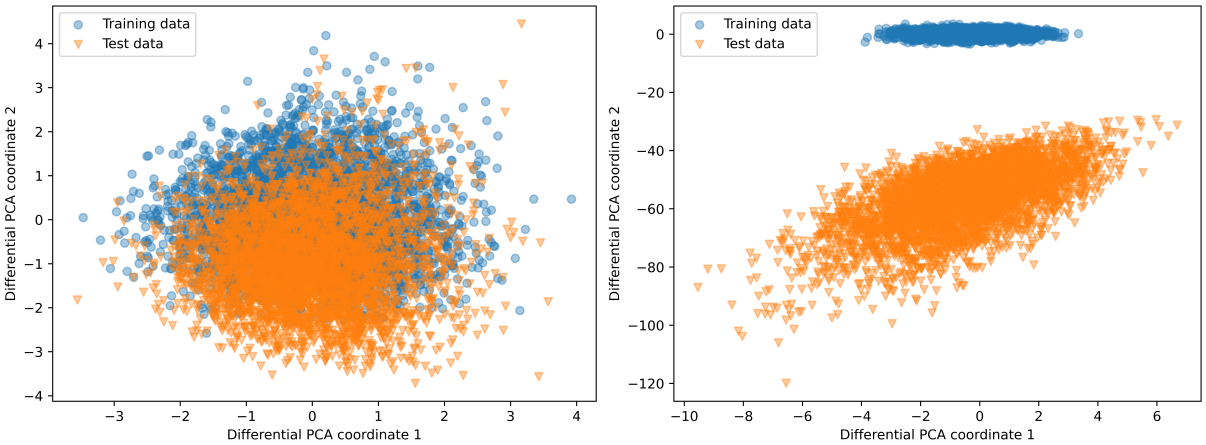
4.3.3 Effect of Domain of Generated Data on Accuracy

Greater differences between the $\mathbb{S}$- and $\mathbb{Q}$-model may also be manifested in that the two models simulate data that are in different regions of the input data space, $(S(T_1), c_{1,1M}(T_1), c_{1,6M}(T_1), c_{1,12M}(T_1))$. Therefore, to further investigate accuracy, we examine how the training data differs from the test data in the 1,000 different parameter value combinations. Since the $\mathbb{Q}$-model produces the training data and the $\mathbb{S}$-model produces the test data, the exposure modelling approach may encounter problems when the two data sets become increasingly different. Furthermore, the preprocessing step used, i.e. differential PCA as explained in section 2.4.3, is only based on the training data and may lead to an even

more prominent difference between the training and test sets when they are transformed and truncated according to the differential PCA procedure.

For the set of 1,000 different parameter value combinations examined in this section, the differential PCA procedure in general leads to a dimension reduction from the original 4 dimensions, $(S(T_1), c_{1,1M}(T_1), c_{1,6M}(T_1), c_{1,12M}(T_1))$, to 2 dimensions. In a few cases, the dimensionality is reduced to 1 dimension, and these cases are characterised by a low value of the parameter of the volatility of the variance process, ξ. More specifically, all cases where the differential PCA procedure reduces the dimension to 1 dimension, ξ has a value less than 0.08. For reference, the span of examined ξ values is $[0.01, 1.5]$. The reason for the strong dimension reduction in these cases may, as previously noted, stem from that such small values of ξ lead to a stochastic variance process that is close to constant. In such a case, the Heston model becomes more similar to a Black-Scholes model [5], where the spot asset price is the only stochastic process.

In figure 4.11, we see the training and test sets, after transformation and truncation according to the differential PCA procedure, in two out of the 1,000 considered cases. In figure 4.11a, we see an example where the resulting DML prediction is relatively good, whilst figure 4.11b shows the example with largest relative error of EE_{6M}. We note that the training and test set in the latter are widely separated, compared to the previous, implying that the $\mathbb{S}$- and $\mathbb{Q}$-model provide simulated data in different regions, i.e. they simulate very different spot asset prices or observed call option prices at exposure date T_1. This is not very surprising as parameter values affect the Heston call option formula and hence the observed call option prices, and the $\mathbb{S}$- and $\mathbb{Q}$-model have significantly different parameter values in the example shown in figure 4.11b.



(a) $\kappa = 6.21$, $\theta = 0.0718$, $\rho = -0.712$, and $\xi = 0.505$

(b) $\kappa = 6.27$, $\theta = 0.0125$, $\rho = -0.0101$, and $\xi = 0.0481$

Figure 4.11: Display of the training and test sets of two different experiments. For (a) there was a relative error of EE_{6M} at -2.71%, of $\text{PFE}_{6M}^{0.95}$ at -2.51%, and of $\text{PFE}_{6M}^{0.975}$ at -2.34%. For (b) there was a relative error of EE_{6M} at 54.6%, of $\text{PFE}_{6M}^{0.95}$ at 25.8%, and of $\text{PFE}_{6M}^{0.975}$ at 23.3%. For comparison: $\kappa^{\mathbb{S}} = 4.9$, $\theta^{\mathbb{S}} = 0.061$, $\rho^{\mathbb{S}} = -0.45$, and $\xi^{\mathbb{S}} = 0.37$.

To show the effect of differing training and test sets for all parameter combinations, we quantify the difference of the sets using the metric that is referred to as the *training and*

test set overlap, defined in section 3.7.2. The results are presented in figure 4.12, where it can quite clearly be seen that the largest relative errors tend to have little to no training and test set overlap.

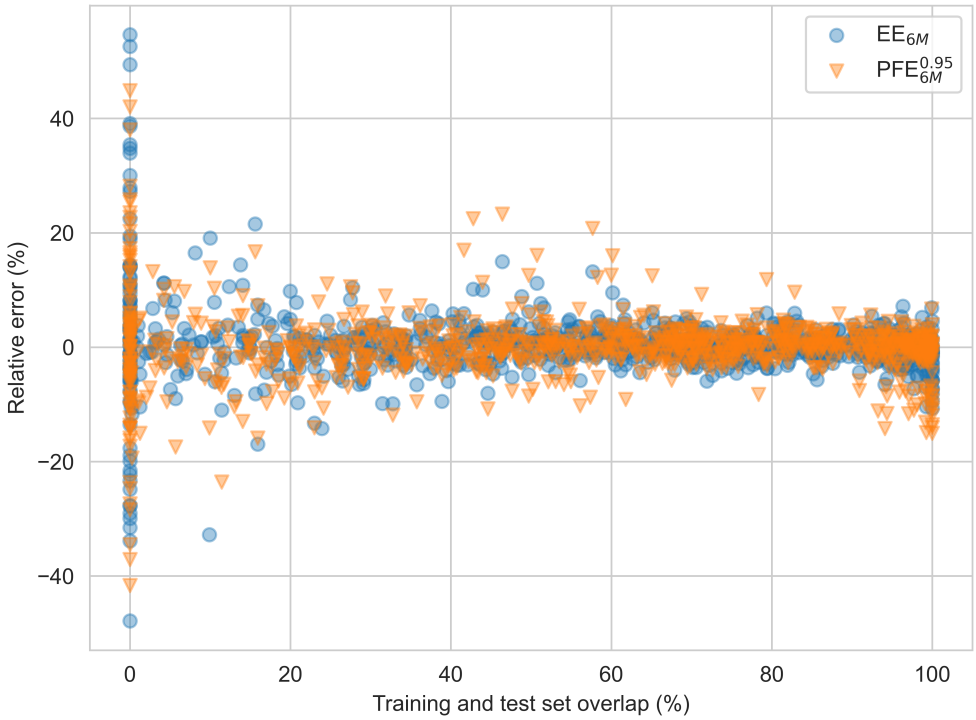


Figure 4.12: Relative error of EE_{6M} and $\text{PFE}_{6M}^{0.95}$ plotted against training and test set overlap for the 1,000 different parameter combinations.

To summarise, we firstly note the positive effect that increased training set size had on the accuracy of exposure modelling using DML. Secondly, we saw that the methodology tends to less accurate as the $\mathbb{S}$- and $\mathbb{Q}$-models became increasingly different. In particular, we observed large errors for relatively low values of ξ and θ in the $\mathbb{Q}$-model, while the framework was able to cope with relatively large values of ξ, although there were fewer examples of these cases. Lastly, in discussing the discrepancies between the $\mathbb{S}$- and $\mathbb{Q}$-model, we found that the largest errors occurred when the two models produced samples in different regions.

5

Conclusion

In chapter 4, it was seen that DML can be applied to exposure modelling, incorporating the use of market observables in a Heston setting for a netting set of European call and put options. As seen in section 4.1, using an observed market state instead of a model state in a single-asset setting does not produce equally accurate, but comparable, results. In section 4.2, where dimensionality is increased, a similar trend is observed, and there are even cases where market state data leads to improved prediction accuracy, primarily in tail predictions for the data with the largest number of dimensions. That being said, predictions using either model state or market state information perform rather poorly in a high-dimensional setting, as prediction errors generally increase for a larger number of underlying assets in the netting set. As previously discussed, this is due to the curse of dimensionality, which is a greater issue when using the informative model state data in conjunction with differential PCA. Nevertheless, within the employed DML framework, we draw the conclusion that using market state data, instead of model state data, results in similar prediction accuracy for identical $\mathbb{S}$- and $\mathbb{Q}$-models, given the used network specifications.

When exposure modelling using DML is applied in settings of different $\mathbb{S}$- and $\mathbb{Q}$-models, as seen in section 4.3, the accuracy tends to decrease with increased difference between the two models, as expected. Specifically, it is when the $\mathbb{Q}$-model has small values for the parameters governing the long-run average variance, θ, and the volatility of the variance process, ξ, in comparison to the $\mathbb{S}$-model, that we see the largest errors. On the other hand, for large values of ξ as well as when θ is not too different in the two models, we get relatively good accuracy. We capture the difference between the two models by how the data they simulate differ. Here we can draw the conclusion that accuracy generally decrease when the two models do not simulate data in the same region.

Bibliography

[1] Hansjörg Albrecher et al. "The little Heston trap". In: *Wilmott* 1 (2007), pp. 83–92.

[2] Basel Committee on Banking Supervision. *Basel Framework*. URL: https://www.bis.org/basel_framework/index.htm?m=2697 (visited on 03/29/2023).

[3] Basel Committee on Banking Supervision. *History of the Basel Committee*. URL: https://www.bis.org/bcbs/history.htm (visited on 04/21/2023).

[4] Richard Bellman and Robert Kalaba. "On adaptive control processes". In: *IRE Transactions on Automatic Control* 4.2 (1959), pp. 1–9.

[5] Fischer Black and Myron Scholes. "The pricing of options and corporate liabilities". In: *Journal of political economy* 81.3 (1973), pp. 637–654.

[6] John C. Cox, Jonathan E. Ingersoll, and Stephen A. Ross. "A Theory of the Term Structure of Interest Rates". In: *Econometrica* 53.2 (1985), pp. 385–407. ISSN: 00129682, 14680262. DOI: 10.2307/1911242.

[7] Georgi Dimitroff, Stefan Lorenz, and Alexander Szimayer. "A parsimonious multi-asset Heston model: Calibration and derivative pricing". In: *International Journal of Theoretical and Applied Finance* 14.08 (2011), pp. 1299–1333.

[8] Geof H. Givens and Jennifer A. Hoeting. *Computational statistics*. 2nd ed. Hoboken, NJ, USA: John Wiley & Sons, 2012.

[9] Kathrin Glau, Ricardo Pachon, and Christian Pötz. "Speed-up credit exposure calculations for pricing and risk management". In: *Quantitative Finance* 21 (July 2020), pp. 1–19. DOI: 10.1080/14697688.2020.1781236.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[11] Jon Gregory. *Counterparty Credit Risk: The New Challenge for Global Financial Markets*. Wiley Finance series. John Wiley & Sons, Ltd, 2010. ISBN: 1118316673.

[12] Andres Hernandez. "Model calibration with neural networks". In: *Risk* (July 2016). DOI: 10.2139/ssrn.2812140.

[13] Steven L. Heston. "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options". In: *The Review of Financial Studies* 6.2 (1993), pp. 327–343. DOI: 10.1093/rfs/6.2.327.

[14] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. "Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models". In: *Quantitative Finance* 21 (Oct. 2020), pp. 1–17. DOI: 10.1080/14697688.2020.1817974.

[15] Brian Huge and Antoine Savine. *Differential Machine Learning*. 2020. URL: https://github.com/differential-machine-learning/notebooks (visited on 05/21/2023).

[16] Brian Huge and Antoine Savine. "Differential machine learning: The shape of things to come". In: *Risk* (2020), pp. 76–81.

[17] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[18] Yann LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[19] Alan Lewis. "A Simple Option Formula for General Jump-Diffusion and Other Exponential Levy Processes". In: *SSRN Electronic Journal* (May 2002). DOI: 10.2139/ssrn.282110.

[20] Shuaiqiang Liu, Cornelis W Oosterlee, and Sander M Bohte. "Pricing options and computing implied volatilities using neural networks". In: *Risks* 7.1 (2019), p. 16.

[21] Roger Lord, Dick van Dijk, and Remmert Koekkoek. "A Comparison of Biased Simulation Schemes for Stochastic Volatility Models". In: *Quantitative Finance* 10 (Feb. 2010), pp. 177–194. DOI: 10.2139/ssrn.903116.

[22] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[23] William McGhee. "An Artificial Neural Network Representation of the SABR Stochastic Volatility Model". In: *Journal of Computational Finance* 25 (2020).

[24] Richard D Neidinger. "Introduction to automatic differentiation and MATLAB object-oriented programming". In: *SIAM review* 52.3 (2010), pp. 545–563.

[25] Harvey J. Stein. "Fixing Risk Neutral Risk Measures". In: *International Journal of Theoretical and Applied Finance* 19.03 (2016), p. 1650021. DOI: 10.1142/S0219024916500217.

[26] Sebastian F. Walter and Lutz Lehmann. "Algorithmic differentiation in Python with AlgoPy". In: *Journal of Computational Science* 4.5 (2013), pp. 334–344. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2011.10.007.

[27] Alexander Cameron Winter. "FX Volatility Calibration Using Artificial Neural Networks". In: *Econometric Modeling: International Financial Markets - Volatility & Financial Crises eJournal* (2020).

[28] Steven H Zhu and Michael Pykhtin. "A guide to modeling counterparty credit risk". In: *GARP Risk Review, July/August* (2007).

A

Appendix 1

In figures A.1 and A.2 one can see how values of parameters κ, θ, and ξ in the $\mathbb{Q}$-model affect the accuracy of the exposure modelling in section 4.3.

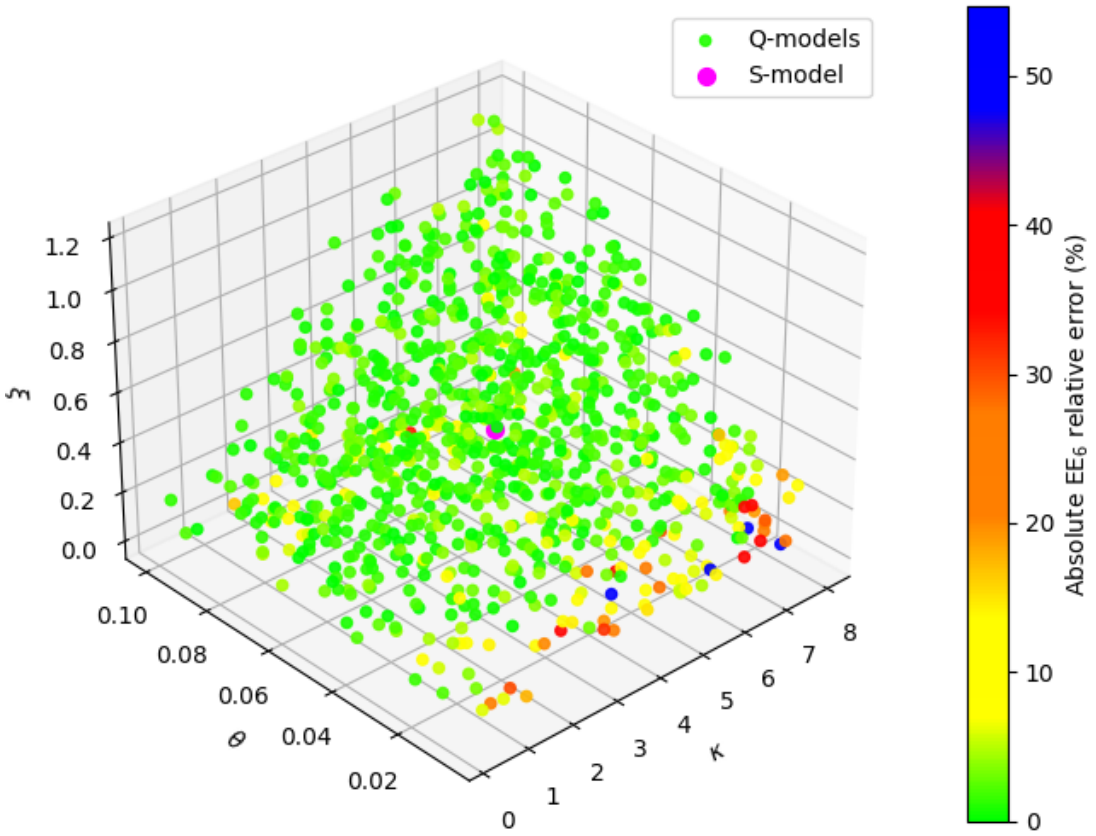


Figure A.1: Absolute relative error of EE_{6M} for the $\mathbb{Q}$-model's κ, θ, and ξ. Green colour corresponds to lower errors, while yellow, red, or blue colour corresponds to higher errors. The $\mathbb{S}$-model parameter values correspond to the magenta coloured dot.

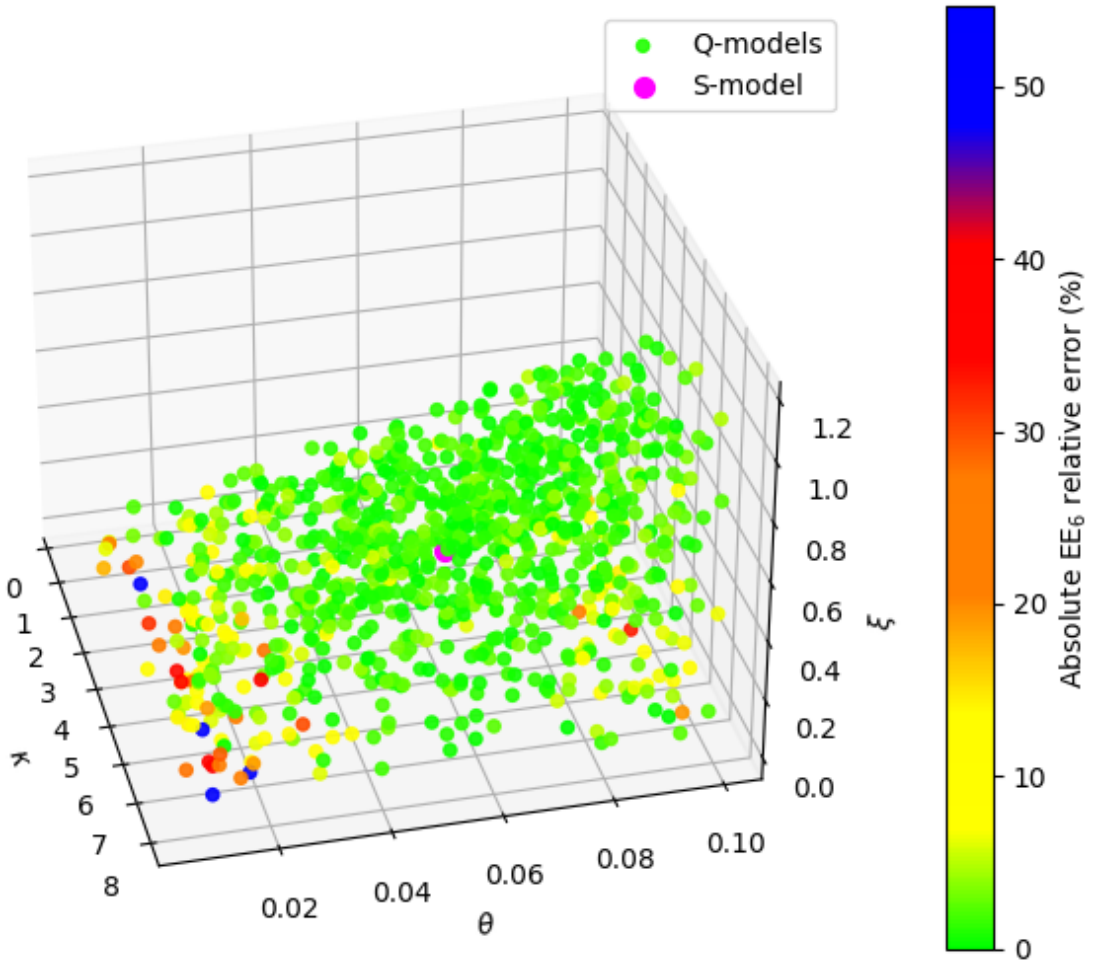


Figure A.2: Absolute relative error of EE_{6M} for the $\mathbb{Q}$-model's κ, θ, and ξ. Green colour corresponds to lower errors, while yellow, red, or blue colour corresponds to higher errors. The $\mathbb{S}$-model parameter values correspond to the magenta coloured dot.

Master's Theses in Mathematical Sciences 2023:E47
ISSN 1404-6342

LUTFMS-3480-2023

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/