Fast Greeks: Case of Credit Valuation Adjustments

Vytautas Savickas (3444945)

Department of Mathematics

Utrecht University

A thesis submitted for the degree of

Master of Science

7 July 2011

1. Reviewer: Prof. Rob Bisseling

2. Reviewer: Prof. Karma Dajani

3. Reviewer: Dr. Norbert Hari

4. Reviewer: Dr. Drona Kandhai

Day of the defense: 7 July 2011

Abstract

(Counterparty) Credit Valuation Adjustments (CVA) has become a prevailing form of pricing default risk on over-the-counter (OTC) contracts. Due to the large size of portfolios included in the CVA calculation and its computational complexity, large computing grids are needed for the evaluation.

The main purpose of this thesis is to investigate an even more computationally demanding problem, namely computing the sensitivities of CVA to the market and model parameters, a topic which was hardly addressed in the literature so far. We show that the pathwise sensitivities method can be applied for CVA and that it gives significant speed improvement over the conventional finite-differencing techniques. Additionally, we take advantage of the GPU technology to obtain the Greeks fast enough for daily hedging and risk management activities.

Contents

List of Figures

1

Introduction

Counterparty risk is traditionally interpreted as credit risk between derivatives counterparties. After the 2007 credit crisis, when important institutions as Bear Sterns, Lehman Brothers, Fannie Mae and Freddie Mac failed, counterparty credit risk has been recognized by the majority of market participants as one of the key financial risks.

In the current trading environment, whenever a derivative is traded over-the-counter (OTC[1]), a default risk charge should be factored into the total price. What does it mean to include counterparty risk? This means that we additionally charge the counterparty, which owes us payments, for the possibility that it defaults during the period of the contract and we lose the rest of the cashflows, which may have been profitable for us.

As we move from single trades to full portfolio level CVA, pricing gets a lot more complicated as the potential losses and gains, in case of default, can cancel out and are subject to, often complicated, legal agreements such as netting rules[2] and collaterization[3]. Moreover, the contracts in question can be heavily correlated and influence our overall risk profile due to lack of diversification. Portfolio-level CVA calculation and management will be mandatory for all banks as of 2013, due to the new Basel 3 regulations [1].

[1] OTC contracts are traded (and privately negotiated) directly between the counterparties, without going through an exchange or other intermediary. The OTC market is much less regulated with respect to disclosure of information between the counterparties.

[2] A netting agreement between two entities, say A and B, means that if A defaults and does not pay B, then B can offset their losses by the same amount out of all payments it owes A. Without netting, B would have to pay back A all of the payments if A defaults.

[3] Collaterization is the act where a borrower A pledges some asset to the lending counterparty B, so that B is insured against the default of A (in case A defaults, B keeps the asset).

To control the counterparty risk at a portfolio level one has to hedge it, in other words, to buy or sell liquid, basic (vanilla) contracts that offset either market or counterparty credit exposure. In order to obtain the proper amounts of the corresponding contracts, the trader needs to know the sensitivities of portfolio CVA to observable market parameters (yield curves, stocks prices, volatility surface points, CDS rates).

As we have already mentioned, portfolio CVA evaluation is itself a very computationally intensive task and using a naive way to find its sensitivities (the so-called Greeks) would introduce unnecessary computational burden. In this thesis we will show that pathwise Greeks methodology can be applied to CVA and that it is possible to take advantage of the GPU technology for pathwise Greeks calculation.

The fundamentals of CVA pricing are introduced in the next chapter and the mathematical framework for pricing basic financial instruments is described in chapter 3 together with the multi-currency one-factor Hull-White interest rate model that will be used for CVA modelling. Then, a step-by-step framework for CVA and pathwise Greeks calculation is detailed in chapter 4. In chapter 5 the fundamentals behind GPU computing and specifically the NVIDIA CUDA API are introduced and it is shown how to compute pathwise Greeks in this setting. Afterwards the results of all simulations for both CPU and GPU-based CVA Greeks are given in chapter 6 and the final discussion of their significance is given in chapter 7.

2

Counterparty Credit Valuation Adjustments

Counterparty risk is a combination of credit risk (default of the counterparty) and market risk (uncertain potential value of the derivative contract at the time point when a credit event happens). Counterparty risk typically arises from a broad class of financial products:

- OTC derivatives, such as:
  - interest rate swaps and swaptions
  - foreign exchange (FX) forwards and options
  - credit default swaps
- Securities financing transactions:
  - repo and reverse repo agreements
  - securities borrowing and lending

The most commonly traded are interest rate derivatives (see Figure 2.1), hence during this thesis we will be concentrating on them.

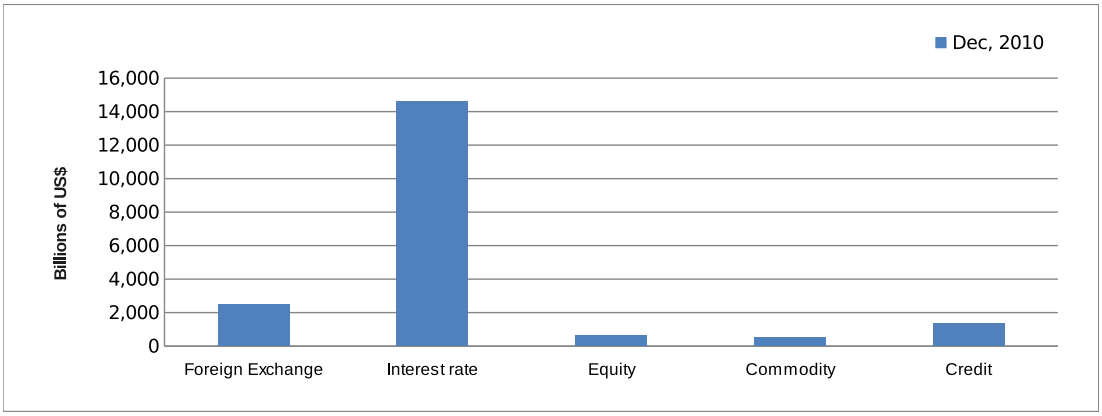


Figure 2.1: Market values of OTC contracts by type, data taken from BIS database [4]

Credit Exposure

The term exposure defines the loss we incur in the event of a counterparty default. Exposure is characterized by the fact that a positive value of a financial instrument corresponds to a claim on a defaulted counterparty, whereas in case of negative value, we cannot walk away. This means that if at the time of default, they owed us (the bank) money - we incur a loss, but if we owed them - we would still need to pay them and would not incur a gain from the default.

Potential Future Exposure

The concept of potential future exposure (PFE) arises from the need to characterize what the value of our OTC contract might be over time. The PFE illustrated in Figure 2.2 characterises the value of an interest rate swap over time. At the current time t we know the current market value of our swap and its past value, but we do not know its future value. Hence we have to assume some model for its price and generate future scenarios, obtaining in this way a distribution of future prices. PFE gives certain exposure bounds at a given confidence level (99%), often considered as a worst case scenario. The expected positive exposure (EPE) is the average positive future exposure. We illustrate both PFE and EPE in Figure 2.2.

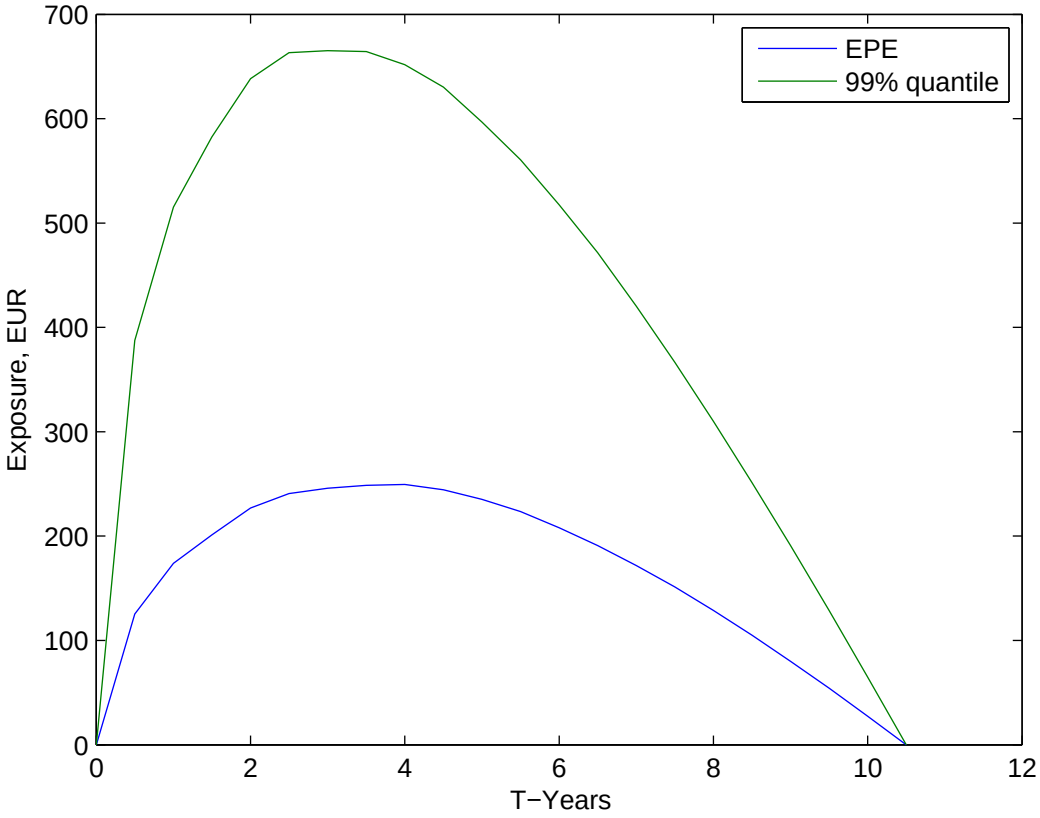


Figure 2.2: Potential Future Exposure Profile of a single 10 year swap starting in 6 months. The exposure becomes non-zero before the swap starts because even if the forward swap had a fair rate at time-zero in 6 months time this swap rate might not be fair and the contract has a potential positive or negative value. Hence if our counterparty goes into default before the start date of the swap - we loose potentially profitable cashflows.

Pricing Counterparty Risk

Intuitively the risk-free price of a contract must be different from a risky (counterparty might default) contract. The price of a risky derivative could be thought as the risk-free price less the component correcting for the counterparty risk.

$$P_{risky} = P_{riskfree} - CVA$$

The latter component is called (Counterparty) Credit Valuation Adjustment (CVA). This counterparty risk charge should be calculated in a sophisticated way to account for all aspects that

define CVA, including:

- the default probability of the counterparty
- the default of the bank (in case of bilateral CVA, see Brigo and Mercurio [6])
- the underlying contract(s)
- netting of existing transactions with the same counterparty
- collaterisation (will not be addressed in this thesis, see Gregory [18])
- hedging aspects (Gregory [18])

2.1 CVA Pricing

By general definition the Credit Valuation Adjustment can be computed as:

$$CVA(t) = L_{GD} \times \mathbb{E}_t^{\mathbb{Q}} \left[D(t,\tau) \cdot (NPV(\tau))^+ \cdot \mathbf{1}_{\{\tau \leq T\}} \right] \tag{2.1}$$

where

- L_{GD} - is the loss fraction (of total value), given default
- $D(t,\tau)$ - is the discount factor, discounting the value from time of default back to the valuation time.
- $NPV(\tau)^+ = max\{0, NPV(\tau)\}$ - is the Net Present Value at time τ of the underlying contract(s). Here we only take the positive side of the value as we are only interested in how much we stand to loose, if the contract value is negative - this amount will be paid back to the counterparty in case of their default; although this was already accounted for in the original price of the contract.
- $\mathbf{1}_{\{\tau \leq T\}}$ - gives the condition that our counterparty defaults before the maturity of our product T.

The CVA formula can be rewritten discretely (see Brigo and Mercurio [6]) as:

$$CVA(t) = L_{GD} \times \left[\int_t^T \mathbb{E}_t^{\mathbb{Q}} \left[D(t,s) \cdot (NPV(s))^+ \cdot \lambda(s) \right] ds \right] \tag{2.2}$$

where $\lambda(s)$ is the instantaneous default probability at time s which is discussed in detail in section 3.4

In this formulation we assume that the seller of the contract (the bank) cannot default, which is a simplification. There are other formulations which consider conditional defaults of both counterparties: BVA, DVA[1].

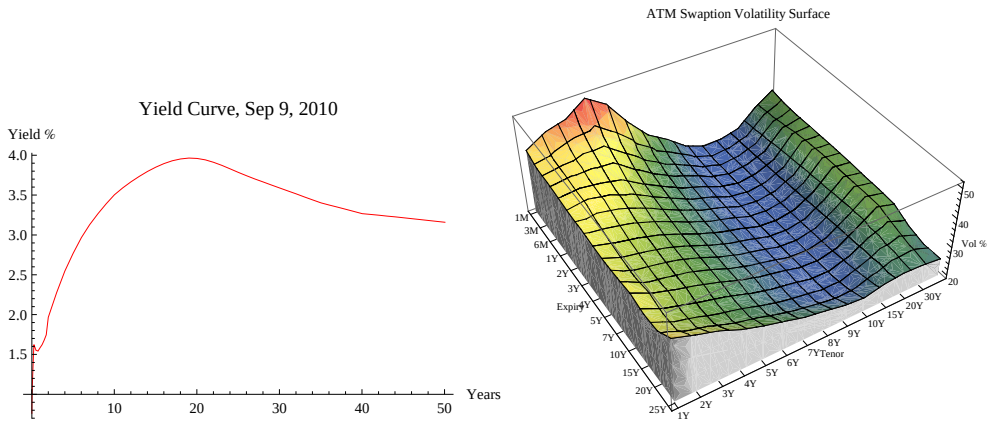
[1] Bilateral Valuation Adjustment and Debt Valuation Adjustment

2.2 CVA Greeks

The sensitivities of CVA to observable market data (the so called ”Greeks”) are critical to trading activities. Greeks define the rate at which the value of CVA changes when some market parameter changes. Most importantly the Greeks guide traders how to hedge the risks associated to the CVA. Mathematically speaking we are interested in the quantities:

$$\frac{\partial CVA(t,\theta)}{\partial \theta}$$

where θ can be a single parameter or a vector of market parameters. As we are mainly interested in CVA for portfolios composed of interest rate products - we shall look into cases when θ is a vector of zero-bond prices (or discrete yield curve points illustrated in Figure 2.3a) that are directly quoted in the market, or a matrix of swaption volatilities which are shown in Figure 2.3b (see product definition in equation (3.8) and book by Brigo and Mercurio [6]).



(a) EUR Yield Curve

(b) EUR ATM Swaption Volatility Surface

Figure 2.3: Observable Market Data

3

Theory

In this chapter we will start by introducing the necessary mathematical background for derivatives pricing, in particular interest rate swaps and swaptions. Then we introduce the one-factor Hull-White interest rate model, show how it extends to two and more currencies and quickly review credit default modelling. In the last section we review the methods used in computational finance to compute sensitivities of derivative contracts to input parameters and finalize with an application to CVA.

3.1 Basic Definitions

The Financial Instruments that are considered throughout this thesis are:

- Zero-Coupon Bonds
- Forward Rate Agreements
- Interest Rate Swaps
- Interest Rate Swaptions

Definition We denote by $P(t,T)$ the price of a zero-coupon bond at time t, with maturity date T, for $T \geq t$.

Definition The simply compounded spot interest rate prevailing at time t for the maturity T is denoted by $L(t,T)$; it is the constant rate at which an investment has to be made to produce an amount of one unit of currency at maturity starting from $P(t,T)$ units of currency at time t when accruing occurs proportionally to the investment time $\tau(t,T) =$

$T - t$, more precisely:

$$L(t,T) = \frac{1 - P(t,T)}{\tau(t,T)P(t,T)},\ t \leq T,\ \tau(t,T) = T - t \tag{3.1}$$

Definition The Forward rate is a simply compounded interest rate nearly equivalent to the definition above, but it has starting date not at current time t, but at future T and ends at S. In other words, it is the rate $L(T,S)$ measured at time t:

$$F(t,T,S) = \frac{1 - P(T,S)}{\tau(T,S)P(T,S)} = \frac{1}{\tau(T,S)}\left(\frac{P(t,T)}{P(t,S)} - 1\right),\ t \leq T \leq S \tag{3.2}$$

Note: $P(T,S)$ measured at time t is equal to $P(t,S)/P(t,T)$

Definition The instantaneous forward rate with maturity T, contracted at t, is defined by:

$$f(t,T) = -\frac{\partial \log P(t,T)}{\partial T} \tag{3.3}$$

Definition The instantaneous short rate at time t is given by:

$$r(t) = f(t,t) \tag{3.4}$$

Definition The (stochastic) discount factor $D(t,T)$ between two time instants t and T is the amount at time t that is "equivalent" to one unit of currency payable at time T and is given by:

$$D(t,T) = exp(-\int_t^T r(s)ds) \tag{3.5}$$

Note: The quantity above is a random number when measured at time t, if $r(t)$ is non-deterministic. The discount factor is directly related to the zero-coupon bonds:

$$P(t,T) = \mathbb{E}\left[D(t,T)\right] = \mathbb{E}^{\mathbb{Q}}\left[e^{-\int_t^T r(s)ds}\right] \tag{3.6}$$

where $\mathbb{Q}$ is the risk-neutral measure (see chapter 1 from Brigo and Mercurio [6] book for detailed explanation).

Definition The (receiver) forward rate agreement (FRA) involves 3 time instants: the current time t, the expiry time $T > t$ and the maturity time $S > T$. This contract gives the holder an interest-rate payment for the period between T and S. At the maturity S, a fixed payment based on a fixed rate K is exchanged against a floating payment based on

the LIBOR[1] spot rate $L(T,S)$. Basically this contract allows one to lock-in the interest rate between times T and S at a desired rate K (e.g 3%).

Formally, at time S one receives $\tau(T,S)K \cdot N$ units of currency and pays the amount $\tau(T,S)L(T,S) \cdot N$, where N is the contract nominal value and $\tau(T,S)$ is the year fraction between T and S. The value of the contract at time S is therefore:

$$N \cdot [\tau(T,S)(K - L(T,S))]$$

The total value of the contract at time t is :

$$FRA(t,T,S,N,K) = N[P(t,S)\tau(T,S)K - P(t,T) + P(t,S)]$$

Setting the fixed rate to simply compounded forward interest rate $F(t,T,S)$ renders the FRA contract fair: the value at time-t is zero.

Definition A prototypical payer (forward-start) interest rate swap (PFS) is a contract that exchanges payments between two differently indexed legs[2], starting from a future time instant. At every instant T_i in a specified set of dates $T_1, ..., T_M$ the fixed leg pays out the amount $N\tau_i K$ corresponding to a fixed interest rate K, a nominal value N and a year fraction τ_i between T_{i-1} and T_i, whereas the floating leg pays the amount $N\tau_i L(T_{i-1}, T_i)$ corresponding to the interest rate $L(T_{i-1}, T_i)$ resetting at the previous instant T_{i-1}.

The interest rate swap can be easily viewed as a series of forward-rate-agreements, with $\mathcal{T} := \{T_1, ..., T_M\}$ as the set of payment exchange dates and $\tau = \{\tau_1, ..., \tau_M\}$ the interval sizes in-between them. Here we assume that the fixed and floating payments occur at the same time, although this may not always be the case.

$$\begin{aligned} PFS(t,\mathcal{T},\tau,N,K) &= (-1)\sum_{i=1}^{M} FRA(t,T_{i-1},T_i,\tau_i,N,K) \\ &= NP(t,T_0) - NP(t,T_M) + -N\sum_{i=1}^{M}\tau_i K P(t,T_i) \end{aligned} \tag{3.7}$$

For the derivation see Appendix A.1.

[1] LIBOR stands for London-Inter-Bank-Offer-Rate, this is the rate quoted in the market for which you can immediately deposit money for a fixed period of time. The common periods are: Overnight, 3 day, 1 month, 3 month, 6 month, 1 year

[2] A leg we call a series of cash-exchanges from one counterparty to another (paying of fixed or floating interest rate on agreed nominal value of the contract), usually the payments of both "legs" happen at the same time, but this is not a restriction.

Definition The swap rate is defined as:

$$S_{T_0,T_M}(0) = \frac{P(0,T_0) - P(0,T_M)}{\sum_{i=1}^{M} \tau_i P(0,T_i)},$$

where T_0 is the first reset date and T_M is the last payment date. The swap (either payer or receiver) has a value zero if its fixed rate is the same as the swap rate.

Why is the swap rate so important? Because on the market, instead of quoting the price of a specific swap, banks quote the (fair) swap rate, hence the quotes are independent of the nominal and this simplifies trading and pricing (swap value at the time of purchase is always zero, just the trader adds a commission on it, often in a form of adjusted fixed rate).

Definition Swaption is an option on a swap, hence its current value can be written as:

$$Swaption(t) = \mathbb{E}^{\mathbb{Q}} \left[D(t,T_0) \cdot (Swap(T_0, \mathcal{T}, \Theta_{swp}))^{+} \right] \tag{3.8}$$

where $D(t,T_0)$ is the discount factor, discounting from the exercise data T_0, back to the current time t, and $\{\mathcal{T}\}$ are the set of Swap cashflow exchange dates. The parameter vector Θ_{swp} contains all the swap related details, like fixed rate, currency, yield curve data, etc.

Swaption can be priced in a Monte-Carlo framework and analytically in several ways. For analytic swaption pricing in the benchmark tests in chapter 6 we are using the Hull and White analytical formula. For detailed formulation and derivation please refer to chapter 3.3.2 from the book by Brigo and Mercurio [6].

3.2 Pricing by Simulation

Simulation has already become an industry-approved method for estimating financial security prices for which a simple closed-form solution does not exist.

Many problems in mathematical finance entail the computation of a particular integral (for instance the problem of finding the arbitrage-free value of a particular derivative). In many cases these integrals can be valued analytically, and in still more cases they can be valued using numerical integration, or computed using a partial differential equation (PDE). However when the number of dimensions (or degrees of freedom) in the problem is large, PDEs and numerical integrals become intractable, and in these cases Monte Carlo methods often give better results.

For more than three or four state variables, formulae such as Black-Scholes (i.e. analytic solutions) do not exist, while other numerical methods such as the Binomial options pricing model and finite difference methods face several difficulties and are not practical. In these cases, Monte Carlo methods converge to the true solution, require less memory and are easier to program than other numerical methods. For simpler situations, however, simulation is not the better solution because it is, in general, very time-consuming and computationally intensive.

Monte Carlo methods can deal with derivatives which have path dependent payoffs in a fairly straight-forward manner. On the other hand Finite Difference and other PDE-solvers struggle with path dependence.

3.3 Multi-Currency Hull-White Interest Rate Model

with piece-wise constant volatility

The first step towards pricing with Monte-Carlo simulation is model definition. We need to define a set of SDEs[1] that we are going to simulate and how they relate to the price of our derivative products. This section is written in the notation of Brigo and Mercurio [6]. We introduce the model and give the model-based bond-pricing formulas essential for interest rate swap pricing in the future, which is needed to obtain the potential future exposure profile (see chapter 2).

We assume that the dynamics of the instantaneous short-rate process for base currency d, under associated risk-adjusted measure $\mathbb{Q}$, is given by:

$$r_d(t) = x_d(t) + \phi_d(t),\ r_d(0) = r_d^0 \tag{3.9}$$

where the process $\{x_d(t) : t \geq 0\}$ satisfies:

$$dx_d(t) = -a_d x_d(t)dt + \sigma_d(t)dW_d(t),\ x_d(0) = 0 \tag{3.10}$$

where $W_d(t)$ is the standard Brownian motion under $\mathbb{Q}$ measure, r_d^0, a_d suitable constants. The functions ϕ_d, σ_d are deterministic and well defined on the time interval $[0, T]$ with T the given time horizon (i.e. 50 years). In particular $\phi_d(0) = r_d^0$.

[1] Stochastic differential equations: differential equations in which one or more terms is a stochastic process, i.e. Brownian motion (A.1)

If we denote by $\mathcal{F}_t^d$-the sigma-field generated by x_d up to time t, then simple integration of equation (3.9) gives for $s < t$:

$$r_d(t) = x_d(s)e^{-a_d(t-s)} + \int_s^t e^{-a_d(t-u)}\sigma_d(u)dW_d(u) + \phi_d(t) \tag{3.11}$$

The second "foreign" currency short-rate process is modelled analogously under its own measure $\mathbb{Z}$:

$$r_f(t) = x_f(t) + \phi_f(t),\ r_f(0) = r_f^0 \tag{3.12}$$

$$dx_f(t) = -a_f x_f(t)dt + \sigma_f(t)dW_f(t),\ x_f(0) = 0 \tag{3.13}$$

Now, consider the market instantaneous forward rates for the two curves "d" and "f" at time 0:

$$f_d^M(0,T) = -\frac{\partial \log P_d^M(0,T)}{\partial T} \tag{3.14}$$

$$f_f^M(0,T) = -\frac{\partial \log P_f^M(0,T)}{\partial T} \tag{3.15}$$

We can perfectly reproduce the initial term structure of discount factors $T \to P_d^M(0,T)$ and $T \to P_f^M(0,T)$ by the above models for r_d and r_f, if we set (see Brigo and Mercurio [6] for details):

$$\phi_d(t) = f_d^M(0,T) + \frac{\sigma_d(t)^2}{2a_d^2}(1 - e^{-a_d T})^2 \tag{3.16}$$

$$\phi_f(t) = f_f^M(0,T) + \frac{\sigma_d(t)^2}{2a_f^2}(1 - e^{-a_f T})^2 \tag{3.17}$$

Equivalent condition for both curves ($q = \{d, f\}$) is:

$$\exp\left[-\int_t^T \phi_q(u)du\right] = \frac{P_q^M(0,T)}{P_q^M(0,t)} \exp\left(-\frac{1}{2}\left[V_q(0,T) - V_q(0,t)\right]\right) \tag{3.18}$$

where $V_q(t,T) = \tilde{V}(t,t_j) + \sum_{s=j}^{n-1} \tilde{V}(t_s,t_{s+1})$, $t_n = T$ as in our model we assume piecewise constant volatility: $\sigma(t)$ is constant for $t \in (t_{j-1},t_j] : \sigma(t) = \sigma(j)$ and for $(t_s,t_e] \subset (t_{j-1},t_j]$ we have:

$$\tilde{V}_q(t_s,t_e) = \int_{t_s}^{t_e} \left(B_q(u,T)^2\sigma_q^2(j)\right) du \tag{3.19}$$

where $B_q(t,T) = \dfrac{1-e^{-a_q(t-s)}}{a_q}$.

After solving the integral ([11]), this expression becomes:

$$\tilde{V}_q(t_s,t_e) = \frac{\sigma_q^2(j)}{2a_q^3}\left(e^{-2a_qT}\left(e^{a_qt_e}-e^{a_qt_s}\right)\left(e^{a_qt_e}+e^{a_qt_s}-4e^{a_qT}\right)+2a_q(t_e-t_s)\right) \quad (3.20)$$

Now the bond pricing function under their respective measures for both currencies becomes:

$$\begin{aligned} P_q(t,x(t),T) = &\quad \mathbb{E}^*\left[\exp\left(-\int_t^T r_q(s)ds\right)\right] && (3.21)\\ = &\quad \mathbb{E}^*\left[\exp\left(-\int_t^T \phi_q(s)+x_q(s)ds\right)\right] && (3.22)\\ = &\quad \exp\left[A_q(t,T)-B_q(t,T)x_q(t)\right] && (3.23)\end{aligned}$$

where

$$A_q(t,T) = \log\frac{P_q^M(0,T)}{P_q^M(0,t)} + \frac{1}{2}\left[V_q(t,T)-V_q(0,T)+V_q(0,t)\right] \quad (3.24)$$

for proofs see [20, 6]

3.3.1 2-Currency Model with Correlations

To be able to price derivatives dependent on two currencies, we introduce a Spot-FX[1] process $S(t)$ denoting the amount of foreign currency needed to buy one unit of domestic currency.We will consider the FX process to be a Geometric Brownian Motion process with mean given by the difference between the two underlying short rates r_d ad r_f. We assume the following no-arbitrage dynamics for S under foreign risk-neutral measure $\mathbb{Z}$:

$$dS(t) = \left(r_f(t)-r_d(t)\right)S(t)dt + v(t)S(t)dW_S(t) \quad (3.25)$$

Under domestic risk-neutral measure $\mathbb{Q}$ everything can be rewritten as:

$$\begin{aligned} r_d(t) = &\quad \phi_d(t)+x_d(t) && (3.26)\\ r_f(t) = &\quad \phi_f(t)+x_f(t) && (3.27)\\ dx_d(t) = &\quad -a_dx_d(t)dt+\sigma_d(t)dW_d^{\mathbb{Q}}(t),\ x_d(0)=0 && (3.28)\\ dx_f(t) = &\quad -a_fx_f(t)dt+\sigma_f(t)\rho_{S,f}\sigma_S(t)dt+\sigma_f(t)dW_f^{\mathbb{Q}}(t),\ x_f(0)=0 && (3.29)\\ \frac{dS(t)}{S(t)} = &\quad (r_d(t)-r_f(t))dt+\sigma_S(t)dW_S^{\mathbb{Q}},\ S(0)>0 && (3.30)\end{aligned}$$

[1] Instantaneous foreign-exchange rate, giving the current (time-t) currency exchange rate

Under the domestic measure the drift term of the foreign short rate process $r_f(t)$ has an additional term $-\sigma_f(t)\rho_{f,S}\sigma_S(t)$. This term ensures that the shifted foreign short rate process (defined in 3.12) is a martingale with respect to domestic measure $\mathbb{Q}$. For details see chapter [5.2.1] from [20]

In the model we assume a full matrix of correlation between the Brownian motions $\mathbf{W}(t) = \left[dW_d^{\mathbb{Q}}, dW_f^{\mathbb{Q}}, dW_S^{\mathbb{Q}}\right]^T$

$$d\mathbf{W}(t) \cdot d\mathbf{W}(t)^T = \begin{bmatrix} 1 & \rho_{d,f} & \rho_{d,S} \\ \rho_{d,f} & 1 & \rho_{f,S} \\ \rho_{d,S} & \rho_{f,S} & 1 \end{bmatrix}$$

3.3.2 m-Currency Model Extension

For pricing of a highly multi-currency portfolio ($m = 50$ currencies involved), we need to simulate 50 short rate processes and 49 exchange rate processes relating the base currency with each of the foreign ones. If we need the foreign-foreign exchange rates - we can simply make a switch: $\frac{S_{f_1}}{S_{f_2}}$ where each S_{f_i} is a domestic-to-foreign exchange rate, in this way obtaining the r_{f_1} to r_{f_2} exchange rate. Each of the short rate processes under domestic $\mathbb{Q}$-measure gets an additional drift term $-\sigma_{f_i}(t)\rho_{f_i,S_i}\sigma_{S_i}(t)$ as we need to simulate all the processes in a consistent framework. Also we use a full $m \times m$ correlation matrix for the Brownian motions $W_{f_i}^{\mathbb{Q}}(t)$, example in equation (4.1).

3.4 Credit Modelling

The Hull and White process only models the potential value of our underlying interest rate portfolio. Another essential part of the computation is modelling the default process. There are two main types of models used in the industry: reduced form and structural models. In the following section we briefly describe the fundamentals behind the prior type.

3.4.1 Reduced Form Models

Also called intensity models, reduced form models describe default by means of exogenous jump process: the default time τ is the first jump of an important kind of stochastic process, namely the Poisson process which can have deterministic or stochastic (Cox process) intensity. With these models, the default is not triggered by market observables, but by exogenous components independent of all market information. This family of models is particularly suited

to model credit spread and in its basic formulation is easy to calibrate to Credit Default Swap (CDS) or corporate bond data [6]. The independence of other market data assumption can be broken, i.e. we can introduce dependence of the default probability on some asset value (i.e. interest rate, if our counterparty has very significant exposure to interest rate products).

In the simplest Poisson process (time-inhomogeneous Poisson process), the risk neutral probability of a default (a jump) in the next dt time, conditioned that we have not defaulted so far is:

$$\mathbb{Q}[\tau \in [t, t+dt) | \tau > t] = \lambda(t)dt$$

The $\lambda(t)$ is called the intensity or the hazard rate at time t. It is possible to show [6] that the probability of survival after time t is:

$$PS(t) = \mathbb{Q}\{\tau > t\} = e^{-\int_0^t \lambda(u)du}$$

if the hazard function is deterministic and if not, then:

$$PS(t) = \mathbb{Q}\{\tau > t\} = \mathbb{E}[e^{-\int_0^t \lambda(u)du}]$$

$\lambda(t)$ can have a similar form as short rate models, for example:

$$\lambda(t) = \mu(t)dt + \sigma(t)dW(t)$$

As stochastic credit modelling is a research topic by itself, during this thesis we will only use a bootstrapped piecewise constant default probability:

$$PD(t_j, t_{j+1}) = PS(t_{j+1}) - PS(t_j), \text{ for } t \in [t_j, t_{j+1}),\ j = \{0, ..., N\}$$

where $t_0 = 0$, the current time (or simulation starting time, when the base data is taken) and $T_{N+1} = T =$ is the time horizon. See chapter 22 from Brigo and Mercurio [6] book for detailed discussions on credit models and related issues.

3.5 Greek Computations for Derivative Instruments

There are several ways of computing Greeks, each of them with its own advantages and disadvantages. In this section we will refer to function $F(t, \theta)$ as the value of the derivative instrument at time t and market input parameter vector θ and to the function $f(T, \theta)$ as the payoff of this derivative at time $T \geq t$. Hence, $F(t, \theta) = \mathbb{E}\left[f(T, \theta)\right]$

3.5.1 Finite Difference Method

$$\frac{\partial F(t,\theta)}{\partial \theta} \approx \frac{F(t,\theta + h \cdot e_k) - F(t,\theta)}{h} \tag{3.31}$$

In the above formulation, we can calculate the forward difference derivative of $f(t,\theta)$ with respect to the k'th market parameter using a small $h \in \mathbb{R}^+$ (bump size), where e_k is a unit vector. This method is also known as "bump & revalue". This approach is numerically stable, but often gives a bit biased derivatives if the function f is highly non-linear (as this derivative is taken from first order Taylor series approximation) and we do not capture higher order effects. For larger precision one might use higher order methods, but it is not that common in practice.

3.5.2 Likelihood-Ratio Method

Talking in the general setting, the value of a derivative, in most cases, can be expressed as:

$$\mathbb{E}[f(S)] = \int_{\mathbb{R}} f(S) \cdot p_\theta(S) dS \tag{3.32}$$

where $f(S)$ is the payoff function of underlying S and $p_\theta(S)$ is the probability density of the underlying factor with parameters θ. For the derivatives, under certain conditions (to be discussed later) one can interchange the integral and the derivative, obtaining:

$$\frac{\partial \mathbb{E}[f(S)]}{\partial \theta} = \int_{\mathbb{R}} f(S) \cdot \frac{\partial p_\theta(S)}{\partial \theta} dS = \int_{\mathbb{R}} f(S) \cdot \frac{\partial \log p_\theta(S)}{\partial \theta} p_\theta(S) dS = \mathbb{E}\left[f(S) \frac{\partial \log p_\theta(S)}{\partial \theta} \right] \tag{3.33}$$

where the last expression should be evaluated either analytically or with numerical integration. Pros:

- No differentiation of $f(S)$, can handle discontinuous payoffs
- Simplifies implementation if payoffs are complicated, non-differentiable (binary option)

Cons:

- The derivative of the density might be complicated to obtain
- Large variance $O(h^{-1})$ of the estimator, limit on time-step size (see chapter 7.2.1 from the book by Glasserman [16]).

Of course for this approach one needs an expression for the density function of the underlying $p(s)$ and this is very complicated to do in the case of CVA computation. We will come back to this method shortly when discussing the Vibrato Monte Carlo method.

3.5.3 Pathwise Method

The ability to interchange the expectation and the derivative allows for one more method to be used to calculate the Greeks for derivatives:

$$\frac{\partial \mathbb{E}[f(S)]}{\partial \theta} = \int_{\mathbb{R}} \frac{\partial f(S)}{\partial \theta} \cdot p(S) dS = \mathbb{E}\left[\frac{\partial f(S)}{\partial \theta}\right] \tag{3.34}$$

To evaluate the expression on the right hand side we either compute it analytically (if possible) or we assume a model for the dynamics of the underlying S; then we perform a Monte-Carlo simulation of the underlying process S, compute the derivative of the payoff

$$\frac{\partial f\,(S, w)}{\partial \theta} \tag{3.35}$$

on every simulated path w, and then average the values, obtaining an unbiased estimate of the sensitivity as we have not introduced any truncation errors as in a finite-difference style Greeks computation method.

Advantages of the pathwise method:

- handles path-dependent payoffs
- handles multi-asset options
- handles square-root diffusion processes

Disadvantages:

- does not handle well discontinuous payoffs

For more details see the book on Monte-Carlo methods [9], and the paper by M.Giles [26].

3.5.4 Forward/Adjoint Differentiation

Very often the payoff function f is complicated and can be written as a composition of functions $f \circ X \circ q \circ S$. Now our main problem is finding a good method to compute the derivative of the payoff (call it $f(S)$) with respect to some market parameter (or a vector of them) θ. Let's say that the payoff function depends primarily on some stochastic process vector $X(S) = (X_1(S), X_2(S), ..., X_n(S))$, which depends on the value of another function $q(S) = (q_1(S), ..., q_m(S))$, which depends on our market observable S and on parameter(vector) θ. Then, we can write our derivative of interest using the chain rule as:

$$\frac{df(S)}{d\theta} = \frac{df(S)}{dX(S)} \cdot \frac{dX(S)}{dq(S)} \cdot \frac{dq(S)}{d\theta} \tag{3.36}$$

This formulation does not change the value of the derivative, but the way we compute it:

- We now need to compute 3 derivatives (or Jacobians in case of vector valued functions) instead of one.
- These part-by-part Jacobians are usually much easier than computing the complete derivative immediately
- We have matrix-vector multiplications present and hence the order of multiplication matters and there are two possibilities:
  - Forward mode means that we multiply the Jacobians starting from the back, going forward
  - Backward mode means that we start at the front and go to the back of the chain.

Example: Pathwise Delta Calculation

Let's take $X(t)$ to be a multi-dimensional stock price process with:

$$dX(t) = a(X(t))dt + \sigma(X(t))dW(t) \tag{3.37}$$

where $X(t) = [X_1(t), X_2(t), \ldots, X_m(t)]$, $W(t)$ is d-dimensional Brownian Motion (see Appendix A.1), $a : \mathbf{R}^m \to \mathbf{R}^m$ and $\sigma : \mathbf{R}^m \to \mathbf{R}^{m\times d}$ are deterministic functions and t stands for time. In our case X could be a vector of stock prices.

Then, we define function $g(X(T)) : \mathbf{R}^m \to \mathbf{R}$ to be the discounted payoff at time T of a derivative, that has price $\mathbf{E}\left[g(X(T))\right]$ at time zero.

As $X(t)$ is a continuous-time stochastic process, before starting the simulation, we discretize the process using the forward Euler discretization scheme, obtaining:

$$\hat{X}(n+1) = \hat{X}(n) + a(\hat{X}(n)) \cdot h + \sigma(\hat{X}(n)) \cdot Z(n+1) \cdot \sqrt{h},\ \hat{X}(0) = X(0) \tag{3.38}$$

where n is the discretized time index, h is the timestep, $\hat{X}(n)$ is the value of $\hat{X}$ at time $n \cdot h$ and $Z(n+1)$ is an independently drawn d-dimensional standard normal random number.

The price of the derivative with discounted payoff $g(\hat{X}(N))$ with $N = \dfrac{T}{h}$ is calculated using the average of independent simulations of $g(X(N))$. Now we consider estimating the derivative vector:

$$\frac{\partial \mathbf{E}\left[g(X(N))\right]}{\partial X(0)} = \left[\frac{\partial \mathbf{E}\left[g(X(N))\right]}{\partial X_1(0)}, \frac{\partial \mathbf{E}\left[g(X(N))\right]}{\partial X_2(0)}, \ldots, \frac{\partial \mathbf{E}\left[g(X(N))\right]}{\partial X_m(0)}\right] \tag{3.39}$$

By using the pathwise method, we can obtain an unbiased estimate if:

$$\frac{\partial \mathbf{E}\left[g(X(N))\right]}{\partial X(0)} = \mathbf{E}\left[\frac{\partial g(X(N))}{\partial X(0)}\right] \tag{3.40}$$

The minimum condition in order for (3.40) to hold is that process X is smooth and function g is Lipschitz continuous. These are further explained in Appendix A.1.

Hence now we can estimate the derivative by taking the average derivative of every simulated path. The derivative vector (for each path $i \in \{1, \cdots, L\}$)

$$\frac{\partial g(\hat{X}^i(N))}{\partial X(0)} = \left[\frac{\partial g(\hat{X}^i(N))}{\partial X_1(0)}, \frac{\partial g(\hat{X}^i(N))}{\partial X_2(0)}, \ldots, \frac{\partial g(\hat{X}^i(N))}{\partial X_m(0)}\right] \tag{3.41}$$

might be complicated to compute directly hence we split it into parts by the chain rule (from now on we will have in mind one simulated path and ignore the path index i):

$$\frac{\partial g(\hat{X}(N)}{\partial X(0)} = \overbrace{\frac{\partial g(\hat{X}(N))}{\partial \hat{X}(N)}}^{A} \cdot \overbrace{\frac{\partial \hat{X}(N)}{\partial X(0)}}^{B} \tag{3.42}$$

Above A is an m-dimensional row vector and B is an $m \times m$ Jacobian matrix. If B is still complicated to compute directly, we split the expression into smaller subparts:

$$\frac{\partial g(\hat{X}(N)}{\partial X(0)} = \frac{\partial g(\hat{X}(N))}{\partial \hat{X}(N)} \cdot \frac{\partial \hat{X}(N)}{\hat{X}(N-1)} \cdot \frac{\partial \hat{X}(N-1)}{\hat{X}(N-2)} \cdot \ldots \cdot \frac{\partial \hat{X}(1)}{X(0)} \tag{3.43}$$

Each of the above matrices $D(n) = \dfrac{\partial \hat{X}(n+1)}{\partial \hat{X}(n)}$ has entries:

$$D_{i,k} = \partial_{i,k} + \frac{\partial a_i}{\partial x_k} \cdot h + \sum_{l=1}^{d} \frac{\partial \sigma_{i,l}}{\partial x_k} \cdot Z_l(n+1),\ \partial_{i,k} = \begin{cases} 1 & i = k \\ 0 & \text{else} \end{cases} \tag{3.44}$$

where a and σ are evaluated at time $n \cdot h$ on the current path value $\hat{X}(n)$

The derivative can now be rewritten as:

$$\frac{\partial g(\hat{X}(N)}{\partial X(0)} = \frac{\partial g(\hat{X}(N))}{\partial \hat{X}(N)} \cdot D(N-1) \cdot D(N-2) \cdot \ldots \cdot D(0) \tag{3.45}$$

Forward & Backward Evaluation

For forward style evaluation we define:

$$\begin{aligned} F(0) &= \mathbb{I},\ \text{identity matrix } m \times m \\ F(n) &= D(n) \cdot F(n-1) \end{aligned} \tag{3.46}$$

Thus $\frac{\partial g(\hat{X}(N)}{\partial X(0)} = \frac{\partial g(\hat{X}(N))}{\partial \hat{X}(N)} \cdot F(N)$ can be evaluated iteratively by computing (in this order) $F(0), F(1), \ldots, F(N)$. Here, as each of $D(n)$ matrices is $m \times m$, the number of FLOPS [1] at every iteration is of order $O(m^3)$, so we have in total $O(m^3 \cdot N)$ operations to obtain $F(N)$ and then one more vector-matrix multiplication.

In a backward (adjoint) style we define:

$$\begin{aligned} B(N) = & \left(\frac{\partial g(\hat{X}(N))}{\partial \hat{X}(N)} \right)^T \\ B(n) = & \; D(n)^T \cdot B(n+1) \end{aligned} \tag{3.47}$$

This way $\frac{\partial g(\hat{X}(N)}{\partial X(0)} = B(0)^T$. For each of the N iterations to compute $B(0)$ starting with $B(N)$, we perform a vector-matrix multiplication, costing us $O(m^2)$ FLOPS per iteration. In total we make $O(m^2 \cdot (N+1))$ FLOPS - which is an order of magnitude faster!

Extension: Vibrato Monte Carlo

This method is a mixture of likelihood ratio method 3.5.2 and pathwise sensitivities. The main idea is to simulate a set of Wiener A.1 increments $W = \{\partial W_1, \partial W_2, \ldots, \partial W_{N-1}\}$ for each path: $\hat{X}(0), \ldots, \hat{X}(N-1)$, excluding the last step. Then we compute a conditional (i.e. Gaussian, dependent on underlying model) probability distribution $p_X(\hat{X}(N)|W)$. To increase the performance this distribution should have an analytical formula which is usually the case with stochastic processes used in finance. Assuming that $\hat{X}$ has Gaussian distribution:

$$\hat{X}(N)(W, Z) = \mu_W + \sigma_W \cdot Z \tag{3.48}$$

where μ_W is the given mean, σ_W is the standard deviation and Z is a standard normal random variable. Then, the price of our derivative can be written as:

$$V = \mathbb{E}_W \left[\mathbb{E}_Z \left[f(X(N)) | W \right] \right] \tag{3.49}$$

To compute the Greeks the derivative is interchanged with the first expectation and then the log-likelihood method is applied for the last timestep:

$$\frac{\partial V}{\partial \theta} = \mathbb{E}_W \left[\frac{\partial}{\partial \theta} \mathbb{E}_Z \left[f(X(N)) | W \right] \right] = \mathbb{E}_W \left[\mathbb{E} \left[f(\hat{X}(N)) \frac{\partial \log(p_X)}{\partial \theta} | W \right] \right] \tag{3.50}$$

[1] Floating point operations

where p_X is the distribution of $\hat{X}(N)$ conditioned on W.

The benefit of this method is that one can apply the pathwise Greeks to discontinuous payoffs, as the payoff function is not differentiated in the process. Hence, a big part of the speed-up offered by the the pathwise Greeks method is kept and at the same time the suitable product range is expanded. The method was initially described by M. Giles [26]. As we currently are not covering fast Greeks for CVA on binary-type products we will not use this method in the thesis, however this is a very promising direction for further research.

3.6 Pathwise Greeks: CVA

Using the assumption that credit process is deterministic, we can rewrite the CVA pricing formula (2.1) as:

$$CVA(t) = L_{GD} \times \left[\int_t^T \mathbb{E}_t \left[D(t,s) \cdot (NPV(s))^+ \right] \cdot \lambda(s) ds \right] \tag{3.51}$$

To apply the pathwise Greeks method for CVA, one has to be able to interchange the derivative with both the outer integral and the inner expectation. The conditions for a single integral-derivative interchange

$$\frac{\partial}{\partial \theta} \int f(X(s,\theta)) ds = \int \frac{\partial f(X(s,\theta))}{\partial \theta} ds \tag{3.52}$$

are:

- differentiability of $X(s,\theta)$ with respect to θ.
- a.e. differentiability of function f w.r.t. θ
- Lipschitz continuity of f (hence f has a.e. bounded derivative).

In our case,

$$NPV(s) = \sum_{k=1}^{N} Swap(s, \mathcal{T}_k, \Theta_{swp}) \tag{3.53}$$

and each swap value is a linear function of the zero-coupon bond prices at time s (3.7).

The zero-coupon bond price at (future) time s, as given by (3.21) is a linear function of time-zero bond prices $P^M(0,T)$ with an additional non-linear part depending on the simulated state variable $x(s)$, which is continuous and differentiable w.r.t. σ_j points and independent of the zero-coupon bond prices. This leads to the conclusion that $NPV(s)$ is a continuous function, differentiable w.r.t. market zero-coupon bond prices $P^M(0,T)$ and model volatility

points σ_j. The detailed derivations of swap derivatives are presented in the Appendix A.3, if the reader would like to see the details.

The stochastic, path-dependent, discount factor $D(t,s) = e^{-\int_t^s (x(u)+\phi(u))du}$ is again a continuous function of zero-bond prices via $\phi(u)$ (3.18) and volatilities σ_j through the $x(u)$ part. Furthermore the function $(\cdot)^+ = \max(\cdot, 0)$ is Lipschitz continuous. Hence the product $D(t,s)\cdot(NPV(s))^+$ is a Lipschitz continuous function, differentiable a.e. with respect to zero bonds and volatility parameters.

As the expectation of a Lipschitz continuous and differentiable function stays Lipschitz continuous and differentiable (see Appendix A.1 for proofs), hence $\mathbb{E}_t\left[D(t,s)\cdot(NPV(s))^+\right]$ also satisfies the needed conditions for the interchange of derivative and expectation.

The last term, $\lambda(s)$, is deterministic and independent of zero bond or volatility data, therefore again $\mathbb{E}_t\left[D(t,s)\cdot(NPV(s))^+\right]\cdot\lambda(s)$ is a.e. Lipschitz continuous and differentiable (w.r.t $P^M(0,T)$ and σ_j).

Applying the theorem from Appendix A.1, we are able to interchange the derivative with the outer integral and inner expectation in order to use the pathwise Greeks method for CVA, obtaining:

$$\frac{\partial CVA(t)}{\partial\theta} = L_{GD}\times\left[\int_t^T \mathbb{E}_t\left[\mathbf{1}_{D(t,s)\cdot NPV(s)>0}\frac{\partial(D(t,s)\cdot NPV(s))}{\partial\theta}\right]\cdot\lambda(s)ds\right] \tag{3.54}$$

4

Methods

In this chapter we will introduce step-by-step the methodology used to compute CVA and to obtain the sensitivities both in the single-core and in the parallel GPU-based implementations. The advantages of using GPU co-processors for fast Greeks computation are discussed in the end of the chapter.

4.1 CVA Computation Scheme

Computing CVA on a portfolio of multi-currency swaps can be separated into several steps:

- Calibration of underlying models to simple market instruments
- Generation of short rate and credit scenarios
- Computation of the exposure profile for each of the netting sets[1]
- Discounting exposures to current time and weighting them with the corresponding default probabilities

4.1.1 Model Calibration

The model is calibrated when its parameters are determined in such a way that it captures current prices and dynamics of liquidly traded market instruments. Only then we use the model for pricing of exotic market instruments and, in our case, credit valuation adjustments.

[1] A netting set is a portfolio of contracts with one single entity that can be treated as a default-linked group, meaning that in the event of default, this group will lose value all together, while other netting sets will not be directly affected

For calibration one has to choose a set of traded instruments $\{S_1, ..., S_n\}$ that depend on the market variables of interest $\{x_1, ..., x_k\}$. These values can be stock or commodity prices, interest rates or volatilities. For the CVA problem in particular, one tries to take the set $\{S_i\}$ of all instruments in the portfolio exposed to the counterparty risk.

Each of the $S_i = f_i(t, x_1, ..., x_k, y_1, ..., y_g\}$, where f_i is a function returning the current price, which often (but not always) has an analytical form and $\{y_1, ..., y_g\}$ are a set of model parameters unobservable in the market that have to be estimated. The calibration process is finding the set $\mathcal{Y} = \{\tilde{y}_1, ..., \tilde{y}_g\}$ such that:

$$\sum_{i=1}^{n} ||f_i(\mathcal{Y}) - S_i||_d$$

is minimized w.r.t a chosen distance measure d. The details of the calibration process will not be covered in this thesis. A lot of work on calibration, especially global calibration has been done by C.Albanese [3]. For instrument specific calibration see the book by Brigo and Mercurio [6].

4.1.2 Scenario Generation

We first consider a discretized version of our SDE system (3.26). Using forward Euler discretization we obtain a list of $M + 1$ (one domestic and M foreign) discretized short-rate processes:

$$\begin{aligned}
\hat{r}_d(n) &= \hat{X}_d(n) + \phi_d(n) \\
\hat{r}_{f1}(n) &= \hat{X}_{f1}(n) + \phi_{f1}(n) \\
&\vdots \\
\hat{r}_{fM}(n) &= \hat{X}_{fM}(n) + \phi_{fM}(n)
\end{aligned}$$

and $M+1$ zero-mean martingale processes under $\mathbb{Q}$ measure:

$$\begin{aligned}
\hat{X}_d(n+1) &= \hat{X}_d(n) - a_d\hat{X}_d(n)dt + \sigma_d(n)\sqrt{dt}Z_0,\ \hat{X}_d(0) = X_d(0) = 0 \\
\hat{X}_{f1}(n+1) &= \hat{X}_{f1}(n) - a_{f1}\hat{X}_{f1}(n)dt + \sigma_{f1}(n)\rho_{S1,f1}\sigma_{S1}(n) + \sigma_{f1}(n)\sqrt{dt}Z_1, \\
&\quad \hat{X}_{f1}(0) = X_{f1}(0) = 0 \\
&\vdots \\
\hat{X}_{fM}(n+1) &= \hat{X}_{fM}(n) - a_{fM}\hat{X}_{fM}(n)dt + \sigma_{fM}(n)\rho_{SM,fM}\sigma_{SM}(n) \\
&\quad +\sigma_{fM}(n)\sqrt{dt}Z_M, \\
&\quad \hat{X}_{fM}(0) = X_{fM}(0) = 0
\end{aligned}$$

Furthermore we also discretize the M geometric Brownian motion processes describing the stochastic exchange rate between domestic and each of the foreign currencies:

$$\begin{aligned}
\hat{S}_1(n+1) &= \hat{S}_1(n)\left(1 + (\hat{r}_d(n) - \hat{r}_{f1}(n))\,dt + \sigma_{S1}(n)\sqrt{dt}Z_{M+1}\right), \\
&\quad S_1(0) = S_{1,0} > 0 \\
&\vdots \\
\hat{S}_M(n+1) &= \hat{S}_M(n)\left(1 + (\hat{r}_d(n) - \hat{r}_{fM}(n))\,dt + \sigma_{SM}(n)\sqrt{dt}Z_{2M}\right), \\
&\quad S_M(0) = S_{M,0} > 0
\end{aligned}$$

in all the above the dt is the time-step and $0 \leq n \leq N = \dfrac{T}{dt}$ is the step index with T being the chosen time horizon. The vector $Z = (Z_0, Z_1, ..., Z_{2M})$ is a multivariate normal random vector, independently drawn at each time-step with symmetric covariance matrix:

$$\Sigma = \begin{bmatrix}
1 & \rho_{d,f_1} & \rho_{d,f_2} & \cdots & \rho_{d,S_1} & \cdots & \rho_{d,S_M} \\
* & 1 & \rho_{f_1,f_2} & \cdots & \rho_{f1,S1} & \cdots & \rho_{f_1,S_M} \\
* & * & 1 & \ddots & \ddots & \ddots & \vdots \\
* & * & * & 1 & \ddots & \ddots & \vdots \\
* & * & * & * & \ddots & \ddots & \vdots \\
* & * & * & * & * & 1 & \rho_{S_{M-1},S_M} \\
* & * & * & * & * & * & 1
\end{bmatrix} \tag{4.1}$$

4.1.3 Computation of CVA

For CVA evaluation we pick a discrete set of dates $\{\mathcal{T}_i\} = \mathcal{T}$ to compute the exposure profile (Figure 2.2). During our test we did have a monthly exposure grid, which gave us a reasonably accurate CVA value.

Then, we simulate $\#sim$ number of interest rate and foreign exchange rate scenarios $X_j(\mathcal{T}) = \{X_{j,d}(\mathcal{T}), ..., X_{j,f_M}(\mathcal{T})\}$ (as defined in section 4.1.2) and for each $\mathcal{T}_i$, we compute the value of the portfolio in the base currency for each of the generated paths:

$$P(\mathcal{T}_i, X_j(\mathcal{T}_i)) = \sum_{k=1}^{K} \text{Swap}_k(X_j(\mathcal{T}_i), \phi_k(\mathcal{T}_i), \Theta_k) \cdot \text{FX}_k(\mathcal{T}_i) \tag{4.2}$$

where $X_j(\mathcal{T}_i)$ is the (zero-mean) state variable vector, $\phi_k(\mathcal{T}_i)$ is the (time-zero) yield curve of the same currency as the swap, Θ_k describes all instrument-specific details (fixed rates, payment frequencies, start date, tenor) and $\text{FX}_k(\mathcal{T}_i)$ is the currency conversion rate to base currency if the swap is valued in foreign currency.

Now we have state variables $X_j(\mathcal{T}_i)$ on every simulated path and can compute the zero-coupon bond prices $P(\mathcal{T}_i, T, X_j(\mathcal{T}_i))$ defined in (3.21). Therefore, the value of each swap at time $\mathcal{T}_i$ can be computed analytically (3.7) for every simulated scenario.

In practice (for CVA computation) portfolios are composed of netting sets (description in chapter 2), which do not overlap and sum up to the full portfolio exposed to the counterparty risk. After we obtain the simulated values of the swaps (and other instruments) we aggregate their values into the respective netting sets $q \in \{1, \ldots, Q\}$:

$$\text{Nset Value}_q\left(\mathcal{T}_i, X_j(\mathcal{T}_i)\right) = \sum_{k \in \text{Nset}_q} \text{Swap}_k\left(X_j(\mathcal{T}_i), \phi_k(\mathcal{T}_i), \Theta_k\right) \cdot \text{FX}_k(\mathcal{T}_i)$$

Then we obtain the expected loss of each netting set by multiplying its value with the corresponding default probability (as a difference between consecutive survival probabilities):

$$\text{ExpectedLoss}(q, \mathcal{T}_i, X_j(\mathcal{T}_i)) = (\text{Nset Value}_q(\mathcal{T}_i, X_j(\mathcal{T}_i)))^+ \cdot (SP_q(\mathcal{T}_i) - SP_q(\mathcal{T}_{i-1}))$$

Note: we assume independence between the dynamics of interest rate and credit processes

To obtain the CVA of the whole portfolio on a single simulated path, we sum over all discounted expected losses:

$$CVA(t_0, X_j(\mathcal{T})) = \sum_i \sum_q D(t_0, \mathcal{T}_i, X_j(\mathcal{T}_i)) \cdot \text{ExpectedLoss}(q, \mathcal{T}_i, X_j(\mathcal{T}_i)) \tag{4.3}$$

where $D(t_0, \mathcal{T}_i, X_j(\mathcal{T}_i))$ is the stochastic discount factor (3.5), discounting from time $\mathcal{T}_i$ back to time zero t_0. Afterwards we finalize by averaging the obtained values over all simulated scenarios X_j:

$$CVA(t_0) = \frac{1}{\#sim} \sum_{j=1}^{\#nsim} CVA(t_0, X_j(\mathcal{T})) \tag{4.4}$$

The general CVA valuation schema is shown below:

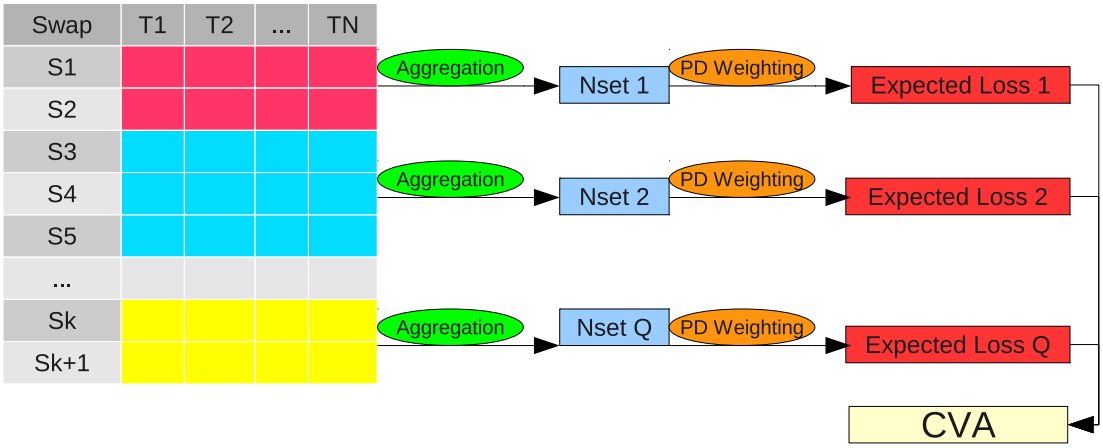


Figure 4.1: CVA valuation scheme: The swap values at each of the exposure dates are aggregated to corresponding netting sets, then all of them are weighted with default probabilities and in the end the expected losses are summed up to total CVA.

4.2 Computing Pathwise Greeks

Our results will be concentrated on two types of Greeks:

- Deltas - sensitivities of the price to movements of each of the yield curve points.
- Vegas - sensitivities of the price to movements of each of the σ_j points (see section 3.3 for definition). *Note that this is not the direct sensitivity to the volatility surface points; to convert the sensitivity from "model sigma" to volatility surface points, one needs to do a bump & recalibrate type of routine, which is out of the scope of this thesis.*

In section 3.6 we have shown that the pathwise Greeks method is applicable to CVA computation with deterministic default rates, Hull and White model and portfolio of analytically valuable derivatives(we give some insight into CVA on exotic derivatives in section 7.1).

Hence, if we want to compute the sensitivity of CVA of this portfolio to some market parameter (or parameter vector) θ, we evaluate the partial derivative per simulated path and

average the results to obtain an unbiased Greek estimate:

$$\frac{\partial CVA(t_0)}{\partial \theta} = \sum_{j=1}^{\#sim} \frac{\partial CVA(t_0, X_j(\mathcal{T}))}{\partial \theta} \tag{4.5}$$

In the described CVA computation framework, we can compute CVA per path as a sum of discounted expected losses at different points in time (4.3), which yields a very convenient way to compute Greeks by using simple derivative rules:

$$\begin{aligned} \frac{\partial CVA(t_0, X_j(\mathcal{T}))}{\partial \theta} &= \sum_i \sum_q \left[\frac{\partial D(t_0, \mathcal{T}, X_j(\mathcal{T}))}{\partial \theta} \cdot \text{ExpectedLoss}(q, \mathcal{T}_i, X_j(\mathcal{T}_i)) \right. \\ &+ \left. D(t_0, \mathcal{T}, X_j(\mathcal{T})) \cdot \frac{\partial \text{ ExpectedLoss}(q, \mathcal{T}_i, X_j(\mathcal{T}_i))}{\partial \theta} \right] \end{aligned} \tag{4.6}$$

Having a realised state variable vector $X_j(\mathcal{T})$ on every path gives us an analytic formula to compute the exposure. Thus, we can take an analytic derivative of it:

$$\begin{aligned} \frac{\partial \text{ ExpectedLoss}(q, \mathcal{T}_i, X_j(\mathcal{T}_i))}{\partial \theta} &= \mathbf{1}_{\text{Nset Value}_q(\mathcal{T}_i, X_j(\mathcal{T}_i))>0} \cdot (SP_q(\mathcal{T}_i) - SP_q(\mathcal{T}_{i-1})) \\ &\cdot \frac{\partial \text{Nset Value}_q(\mathcal{T}_i, X_j(\mathcal{T}_i))}{\partial \theta} \end{aligned} \tag{4.7}$$

for the same reason we can take the derivative of the values of each netting set:

$$\begin{aligned} \frac{\partial \text{Nset Value}_q(\mathcal{T}_i, X_j(\mathcal{T}_i))}{\partial \theta} &= \sum_{k \in \text{Nset}_q} \left[\frac{\partial \text{Swap}_k\left(X_j(\mathcal{T}_i), \phi_k(\mathcal{T}_i), \Theta_k\right)}{\partial \theta} \cdot \text{FX}_k(\mathcal{T}_i) \right. \\ &+ \left. \text{Swap}_k\left(X_j(\mathcal{T}_i), \phi_k(\mathcal{T}_i), \Theta_k\right) \cdot \frac{\partial \text{FX}_k(\mathcal{T}_i)}{\partial \theta} \right] \end{aligned} \tag{4.8}$$

we left the derivations for partial derivatives of interest rate swap and foreign exchange rate in Appendix A.3. A very important observation was made when analytically looking into derivatives and Jacobian matrices of interest rate swaps priced in the future under the one-factor Hull and White model: the chain of Jacobians has diagonal structures, hence using matrix formulation of the problem and applying the backward differentiation, investigated in 3.5.4, does not bring significant speed improvements and introduces a large memory overhead. Therefore in all implementations the derivatives were evaluated using vector operations. We give a detailed picture of this argument in the Appendix A.3.

5

GPU computing

GPGPU (General-Purpose-Graphics-Processing-Unit) technology has shown great potential in conventional derivatives pricing (European, American Options, etc. Abbas-Turki and Lapeyre [2]) as well as in some risk-management applications (VaR, Chong et al. [13])).

The primary motivation for computing CVA and Fast Greeks on a GPU co-processor is the potential speed-up the technology offers. The computation of CVA itself on a specialized proprietary CPU grid [1] for a large (> 50000 instruments) portfolio using highly optimized code takes often more than 2 hours. In comparison, the same computation on one NVIDIA Fermi architecture based card (full computer details in Appendix A.3) takes approximately 4 minutes.

Using conventional methods (bump & revalue), the computation time of CVA Greeks increases linearly with the number of sensitivities:

- As the portfolio is based on up to 50 currencies, each of them having yield curves with 35 points , we would need to bump & revalue 1750 times!
- This is nearly 5 days of parallel GPU computation, or around 145 days on the mentioned CPU grid.
- If we include sensitivities to the volatility surface for each of the currencies and foreign-exchange rates, the computational burden grows even more.

Hence we have investigated the applicability of the GPU technology to the pathwise Greek computations for the Counterparty Valuation Adjustments.

[1] Details could not be disclosed due to ING Bank policy.

5.1 CUDA Environment

To investigate the adequacy of GPU co-processors for the CVA Greeks computation we have used a NVIDIA GPU card (see details in Appendix A.6), which was programmed using the NVIDIA CUDA programming language. CUDA (Compute Unified Device Architecture) is the name of the general purpose parallel computing architecture of modern NVIDIA GPUs. The name CUDA is commonly used in a wider context to refer not only to the hardware architecture of the GPU, but also to the software components used to program that hardware. In this sense, the CUDA environment also includes the NVIDIA CUDA compiler and the system drivers and libraries for the graphics adapter.

From the hardware point of view, CUDA is implemented by organizing the GPU as a collection of streaming multiprocessors, which operate according to the Single-Instruction-Multiple-Thread (SIMT) paradigm[1] . A modern GPU can contain tens of multiprocessors (MPs), each consisting of 32 stream processors (SPs) capable of executing an independent thread (see figure 5.1). The Tesla 2070 device that was used for this project has 14 MPs with four types of on-chip memory available:

- a set of 32-bit registers (local, one set per stream processor)
- shared memory (48 kB per block for CUDA 2.0 Compute Capability devices, shared between SPs in a MP)
- a constant cache (64 kB, shared between SPs in single MP, read-only)
- a texture cache (shared between SPs in single MP, read-only)

[1] Other paradigms are: SISD (single instruction, single data), SIMD (single instruction multiple data), MIMD (multiple instruction multiple data), see book by Patterson and Hennessy [28] for more information.

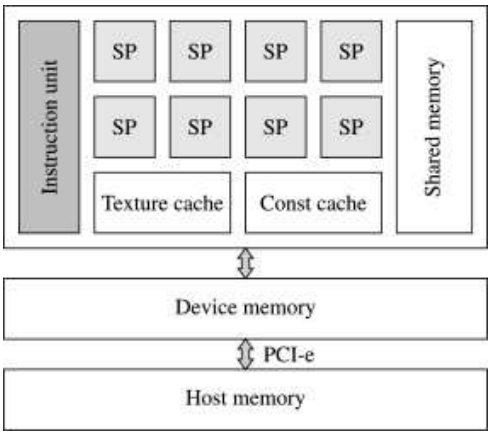


Figure 5.1: Organization scheme of a typical NVIDIA CUDA Stream multi-processor

The amount of on-chip memory is very limited in comparison to the total global memory available on a graphics device (hundreds of kilobytes vs several gigabytes). However, the advantage is given by the access time, which is two orders of magnitude lower than the global memory access time.

The CUDA programming model is based upon the concept of a kernel, which is a function executed multiple times in parallel, each instance running in a separate thread. Then, the threads are organized into one-, two- or three-dimensional blocks, which in turn are organized into one- or two-dimensional grids. The blocks are completely independent of each other and can be executed in any order. Threads within a block are guaranteed to be run on a single multiprocessor. This makes synchronization and efficient information sharing using the on-chip memory of the MP possible.

In a device having Compute Capability 2.0 or higher, each multiprocessor is capable of concurrently executing 1024 active threads ([30, 25]). In practice, the number of concurrent threads per SM is also limited by the amount of shared memory and thus, it does not always reach the maximum allowed value.

The CUDA environment also includes a software stack. For example, CUDA v4.0 consists of a hardware layer, system libraries implementing the CUDA API, a CUDA C compiler and two higher level mathematical libraries (CUBLAS and CUFFT). CUDA C is a simple extension of the C programming language, which includes several new keywords and expressions that make it possible to distinguish the host (i.e. CPU) and the GPU code [30]. The CUDA API contains as well the **Thrust** library which supports multiple CUDA-optimized algorithms

(vector operations, sequence generators, lists), which ought to be a parallel equivalent of the C++ Standard Template Library.

5.2 Pathwise Greeks on GPU

The algorithm for computing CVA sensitivities is suitable for parallel computation on the GPU, because we are evaluating potential losses and the sensitivities to changes in market data per generated Monte-Carlo scenario. In particular, the portfolio is evaluated per netting-set using the "path-per-thread" parallelism. Thus, we have a 3 stage CVA computation scheme:

1. Market and portfolio data preparation and upload to the GPU card
2. Evaluation of portfolio sensitivities per generated path in parallel
3. Parallel aggregation of sensitivities from all threads into a single structure and sending of the data back to the host.

Due to careful design of the CVA computation scheme, the shift from CPU to GPU computation of Greeks is mainly an implementation challenge. There are a couple of optimizations that we used to achieve higher computation speeds:

- Coalesced memory reads & writes [25]
- Pre-computation of common values for all threads (i.e. $A(t,T)$ (3.24))
- Special treatment of sparse sensitivity vectors, whose structure is known beforehand

Additionally, we were using Fermi architecture cards which have an automatic caching feature, hence shared memory was handled automatically.

6

Results

In this chapter we present various results showing both the convergence and the speed up properties of the pathwise Greeks method. Our first set of experiments was based on the benchmark single-threaded interest rate library while the last set was based on the parallel GPU-based implementation.

In the beginning of this chapter we show the results for a swaption (3.8) which was chosen in the early stages of development to demonstrate the potential of the pathwise Greeks method for interest rate derivatives. A swaption ("an option on a swap") was chosen over a simple swap, because it is sensitive to changes in the yield curve and the volatility parameter $\sigma(t)$. Therefore, we could easily make tests for both pathwise Deltas and Vegas. Afterwards we display the results of computing CVA Greeks with a single-threaded CPU implementation and assess the overall potential of pathwise Greeks method. In the last section we show convergence, speed up and scaling results of GPU-based pathwise CVA Greeks implementation.

6.1 Proof-of-Concept: Swaption Greeks

During our first experiment we considered a swaption with:

- 1 year expiration time T_0
- 10 year tenor of the underlying swap T_N
- $K = 4.6\%$ fixed rate, received every 6 months
- 6M EURIBOR paid every 6 months
- 10,000 EUR Notional

The input data we had was 35 points on the yield curve and 5 calibrated piece-wise constant volatility points, so our goal was to compute 35 Deltas and 5 Vegas. We priced the mentioned swaption and computed its Greeks in 3 ways:

- Computed Quasi-Monte Carlo (QMC) based swaption price and evaluated Greeks using the pathwise method.
- Computed QMC based swaption price and evaluated Greeks using finite-differencing with a bump size $h = 10^{-8}$.
- Computed the analytic Hull and White model based swaption price and used finite-differencing with $h = 10^{-8}$ to obtain the Greeks.

6.1.1 Swaption Deltas

In Figure 6.1 we display the sensitivity of the swaption price to the discrete yield curve points. The first and last Deltas come from the underlying floating rate payments and the intermediate, small Deltas come from the fixed leg sensitivities to changes in the yield curve.

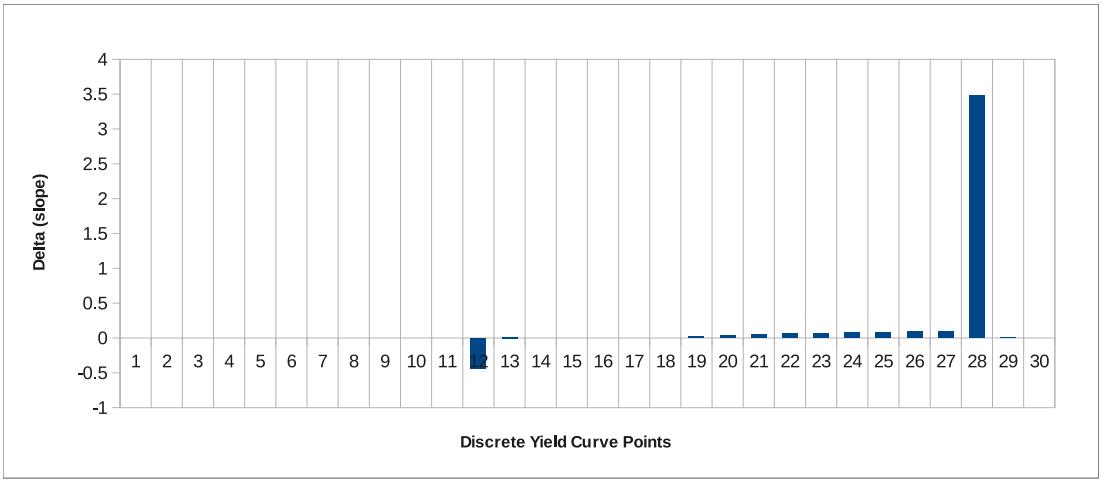


Figure 6.1: Swaption price sensitivities to changes in yield curve. On the x-axis we have the quoted yield curve point indices (only 30 significant ones were illustrated); on y-axis we have the derivative(slope) of the swaption value with respect to changes in each of the buckets. A change of 1 basis point in yield curve gives a Slope$\times$EUR change in the swaption value.

In Table 6.1 we compare the convergence of QMC Deltas under the two discussed methods, pathwise and bump & revalue towards the reference bump & revalue Deltas based on the analytic formula in section 3.8. Additionally, to give a better illustration we show the relative

convergence of all Deltas to the analytic ones in Figure 6.2. In theory the bumped Greeks should have a small bias, however it is not visible in the graph due to a very small "bump" $h = 10^{-8}$.

	Pathwise QMC			Bump & Revalue QMC		
Paths	Point 1	Point 2	Point 3	Point 1	Point 2	Point 3
500	0.0884	3.1732	0.0087	0.0884	3.1732	0.0087
1000	0.0924	3.3203	0.0091	0.0924	3.3203	0.0091
2000	0.0949	3.4085	0.0093	0.0949	3.4085	0.0093
5000	0.0954	3.4288	0.0094	0.0954	3.4288	0.0094
10000	0.0964	3.4643	0.0095	0.0964	3.4643	0.0095
50000	0.0969	3.4802	0.0095	0.0969	3.4828	0.0095
Analytic	0.0969	3.4788	0.0095	0.0969	3.4788	0.0095

Table 6.1: Convergence of Pathwise and Bumped Deltas based on QMC to Bumped Deltas based on analytic formula for 3 randomly chosen points on the yield curve. The numbers in both cases converge at equivalent speeds, as we increase the number of simulated paths.

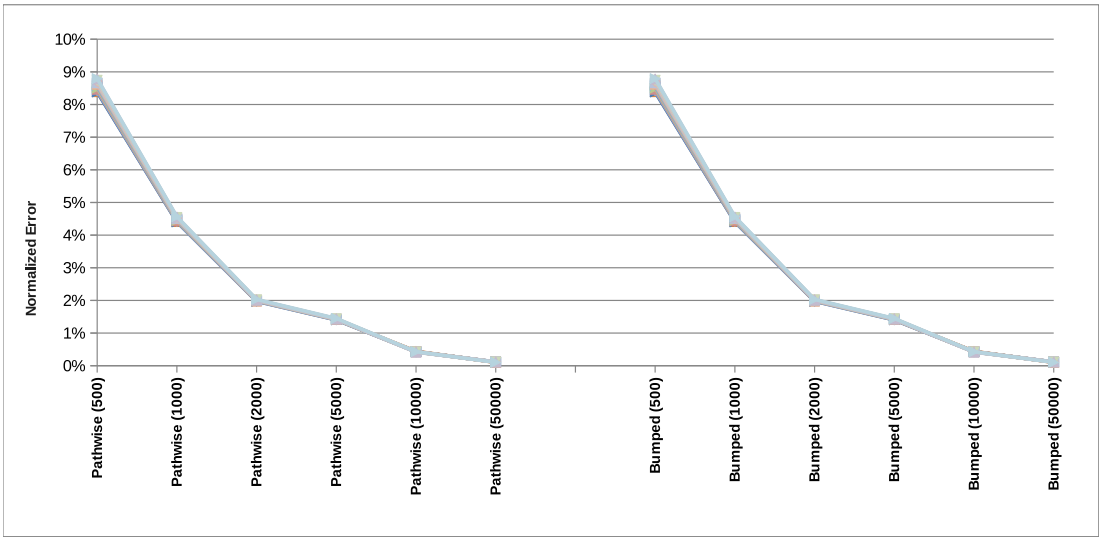


Figure 6.2: Relative error convergence of swaption Deltas. The left half of the graph shows the relative error of the pathwise Deltas, while the right half illustrates the relative error of the bumped Deltas. In each case the number of simulated paths increases from left to right. The convergence rates of both methods are practically identical.

Additionally we compare the time necessary to compute the pathwise and bumped Deltas. Our results show that the pathwise method has the same accuracy as the bumped method but speeds up the computation approximately 5 times. Moreover, we found that the speed up is independent of the number of simulated paths, indicating a fine scalability of the method. Detailed results are charted in Figure 6.3.

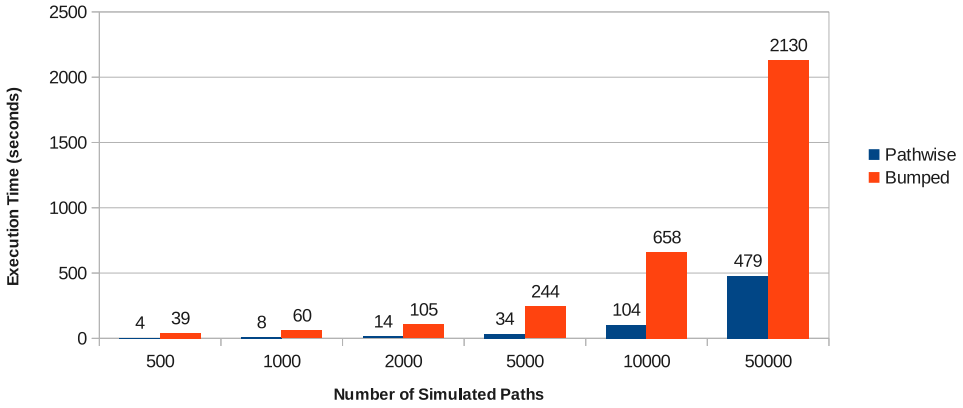


Figure 6.3: Timings of CPU-based Deltas. The red and blue bars show the timings for the pathwise and, respectively, bumped Deltas for different number of paths. The Greeks were computed to every single point in the quoted EUR yield curve.

6.1.2 Swaption Vegas

We made an equivalent experiment to see the convergence of pathwise and bumped swaption Vegas. In Table 6.2 we show only one Vega out of five as our swaption was mainly sensitive to the first volatility piece while the other values were close to zero and prone to machine precision errors.

Using the information from Figure 6.4 we can conclude that yet again pathwise Vegas exhibit equivalent accuracy as the bumped ones. However the speed up of Vega computation (Figure 6.5) for a single swaption is small (5%) because a lot of time is spent evaluating the Vega of state variable $x(t)$ (see derivations in Appendix A.3). Performance improvement should be visible when the number of Vegas becomes much larger.

Paths	Pathwise QMC	Bump & Revalue QMC
500	3.2926	3.2926
1000	3.3391	3.3391
2000	3.3886	3.3886
5000	3.4036	3.4036
10000	3.4211	3.4211
50000	3.4443	3.4443
Analytic	3.4458	3.4458

Table 6.2: Convergence of pathwise and bumped Vegas based on Quasi-Monte-Carlo simulation towards the bumped Vegas based on the analytic Hull and White formula for swaptions.

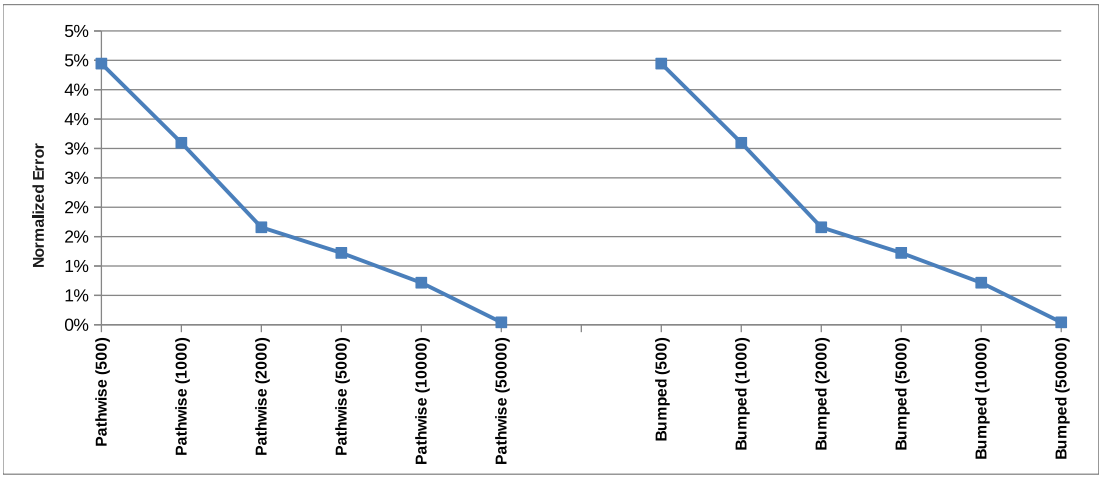


Figure 6.4: Relative error convergence of swaption Vegas with increasing number of simulated paths. Again, the pathwise Vegas are on the left side of the graph and the bumped ones are on the right. We observe similar rates of convergence in either case.

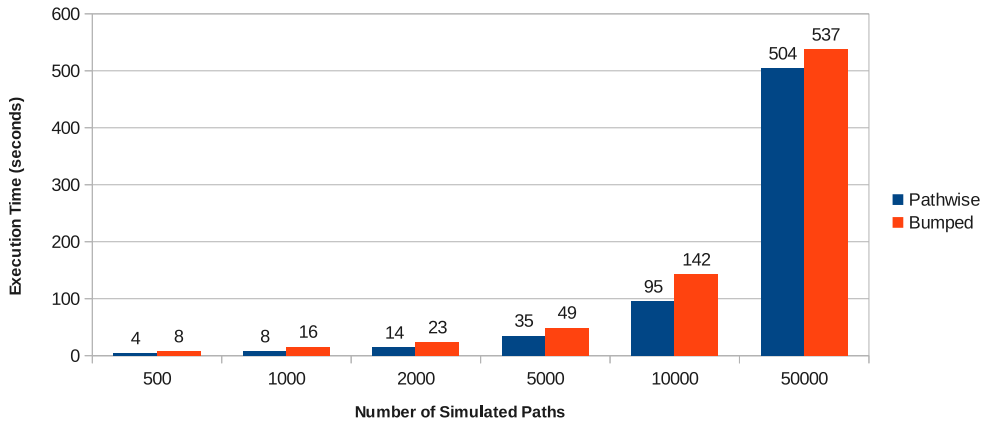


Figure 6.5: Timings of swaption Vegas calculation. The pathwise method gives only a 5% advantage because the evaluation of Vegas is much more computationally intensive and just a small number (5) of sensitivities is considered.

6.2 CVA Greeks

Our next task was to investigate the Greeks of CVA on portfolios of interest rate swaps. First we tested the method using the benchmark single-threaded CPU implementation. In the following subsections we show the results for Deltas in case of CVA on a single swap, and Vegas in case of a swap triplet (EUR, USD, GBP). As it was very time consuming to compute bumped CVA Greeks on a single-threaded CPU implementation we have used the bumped Greeks from a GPU implementation as a reference.

6.2.1 CVA Deltas

For CVA Deltas computation we considered CVA on:

- $T_N = 10$-year Swap (EUR)
- Receiving $K = 4.6\%$ fixed rate (EUR)
- Paying 6-month floating EURIBOR rate
- Notional of the Swap being 10,000 EUR

We computed pathwise and bumped Deltas to 35 yield curve points (EUR), but we display only the significant ones in Figure 6.6. To show convergence of the Greeks with increasing number of simulated paths, we took the bumped Deltas done with 65536 simulations on the

GPU as the reference point. Figure 6.7 shows the relative convergence of pathwise, CPU-based Deltas to the GPU-based bumped Deltas. Already with 3000 paths we are, on average, within 3% relative error, which is a very good result - we can obtain quite accurate Greeks for hedging purposes while keeping the computational costs low.

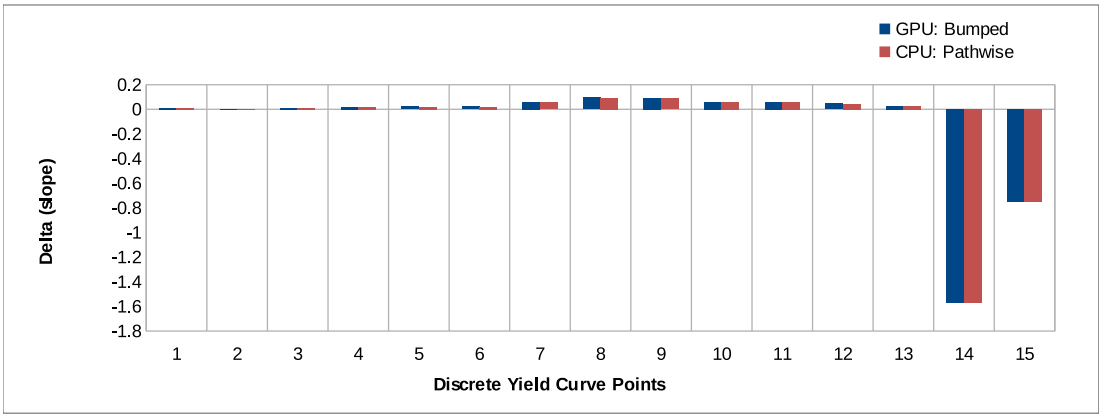


Figure 6.6: CVA sensitivities to changes in the yield curve. On the x-axis we have the quoted yield curve point indices; on the y-axis we have the derivative(slope) of the CVA with respect to changes in each of the buckets. A change of 1 basis point in the yield curve gives a Slope$\times$EUR change in CVA.

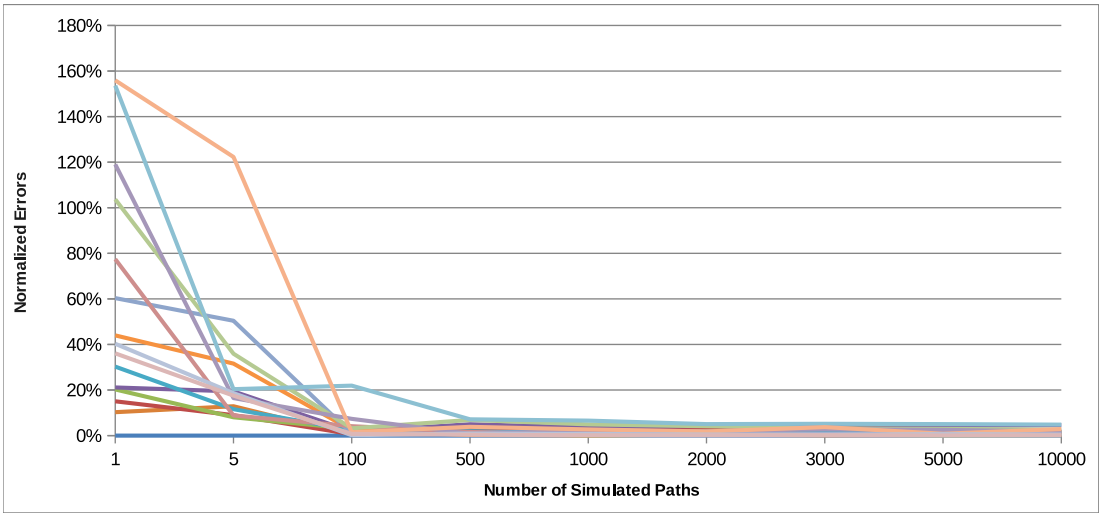


Figure 6.7: Relative convergence of pathwise CVA Deltas to the bumped GPU CVA-calculator based Deltas. Each of the lines represent the relative error of a Delta with respect to one of the yield curve points.

One should note that there is hardly any more convergence shown after 3000 paths: we remain within 3% relative error. The error comes not from the pathwise Greeks but from the bumped GPU-calculator Greeks. The problem is that the fast, parallel floating-point operations on the GPU are only single-precision and they introduce some truncation error as well as slight uncertainty of the result. Therefore each CVA valuation is a bit imprecise (more on this in Appendix A.5). Pathwise Greeks do not have this problem as we do not take numerical derivatives absolutely anywhere. Therefore the accuracy of pathwise Greeks is as high as the estimated CVA value itself.

6.2.2 CVA Vegas

The CVA calculator uses a single constant σ parameter calibrated to the whole swaption surface (Figure 2.3a), instead of the piecewise constant construct mentioned in the swaption results (section 6.1.1). Hence, for better visualization purposes, we chose to use a triplet of interest rate swaps instead of a single swap used with CVA Deltas. Thus, for Vega computation we considered CVA on:

- Three 10-year Swaps (EUR, USD, GBP)
- Receiving 4.6% fixed rate(s)
- Paying 6-month floating rate(s)
- Notional of each of the Swaps was 10,000 (in their respective currencies)

We made convergence tests for Vegas by changing the number of paths used in Monte-Carlo simulation from 1 to 3000. We did not test with larger numbers of paths as the computation using the single-threaded CPU implementation was already very time-consuming. As a benchmark, we again used the GPU-based bumped Vegas computed with 65536 paths. The convergence results, pictured in Figure 6.9, show that with 3000 simulated paths we are at $< 6\%$ relative error for each currency Vega. Their values are highly dependent on the simulated state variables, hence for good convergence we need to use more paths than for Delta estimation.

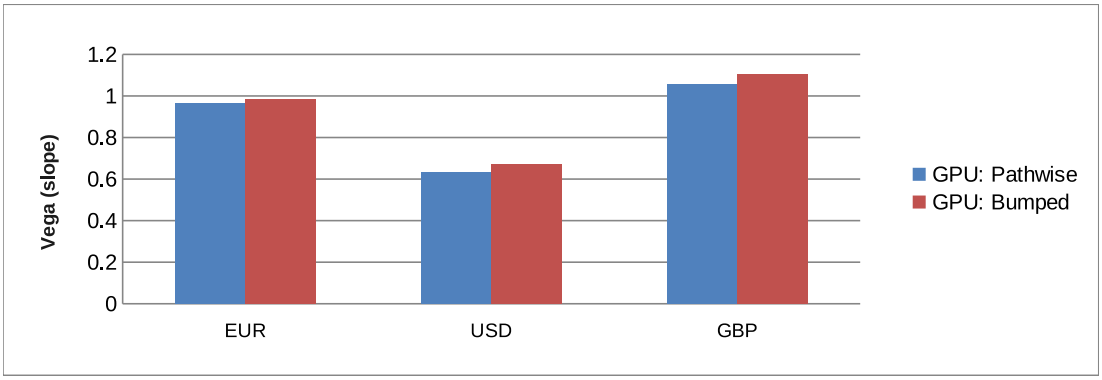


Figure 6.8: CVA Vegas, displayed for every currency.

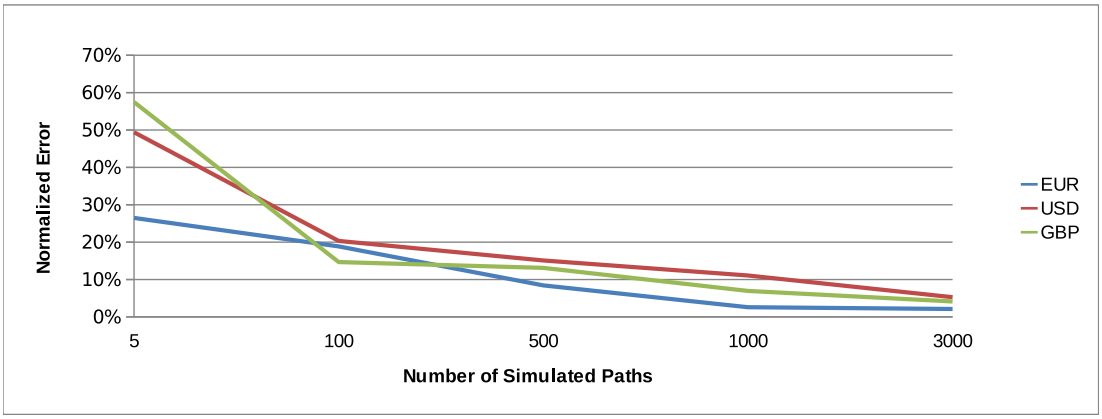


Figure 6.9: CVA Vegas convergence with increasing number of paths used in QMC simulation. We used GPU-based bumped Vegas as a benchmark.

6.2.3 Scaling

The last test on our single-threaded implementation was to perform an initial investigation of the scaling properties of the pathwise and the bump & revalue methods. First, we made multiple timed runs for both cases while keeping the number of simulated paths fixed (only 3 due to time constraints) and changing the number of sensitivities and the portfolio size:

- We started with a single (EUR) swap.
- Then added one more (USD) swap (increase in number of sensitivities)
- and one more (GBP) swap (increase in number of sensitivities).
- Afterwards, we used 2, 3, 4 and 5 swaps for each currency (no increase in number of sensitivities, but additional valuations)

The results of the test show that increasing the number of sensitivities (adding different currency swaps) yields a sharp increase in bumped Greeks computation time, which even quadruples after adding a USD valued swap in the portfolio (see Figure 6.10). However, the pathwise Greeks evaluation time increases proportionally to the number of swaps and seems to be independent of the number of computed sensitivities. In fact, the speed-up introduced by the pathwise method increases from 27 to 78 times as we increment the number of sensitivities from 35 to 105 (3 yield curves) and stays constant when raising the portfolio size without including new currencies. These observations are in line with the results from M.Giles [15], who mentions that the pathwise method brings speed improvements only with increasing number of sensitivities.

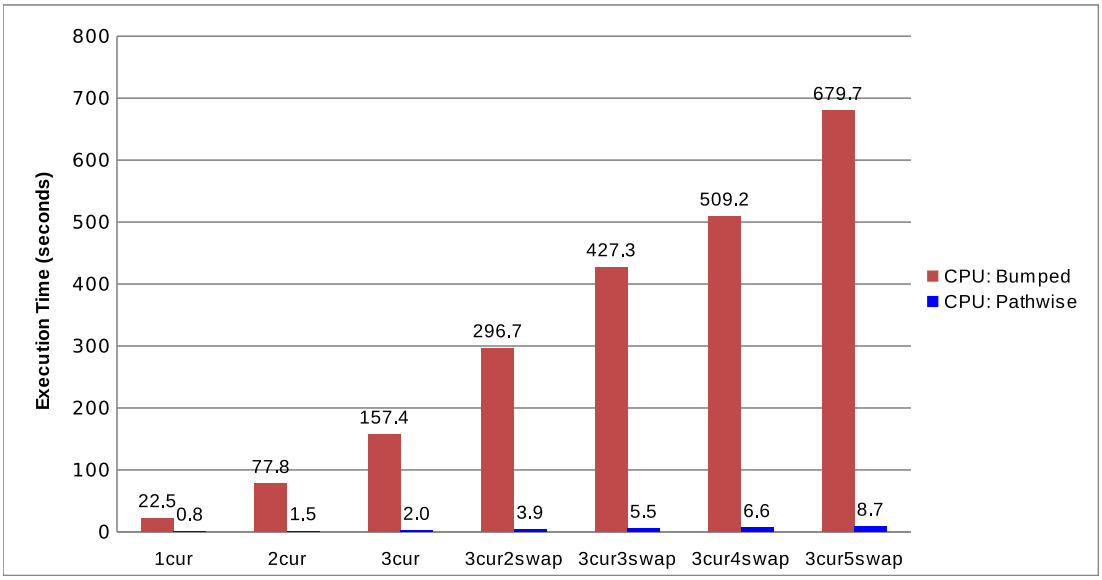


Figure 6.10: Scaling of CVA execution time depending on number of products and number of sensitivities for both pathwise and bump & revalue methods.

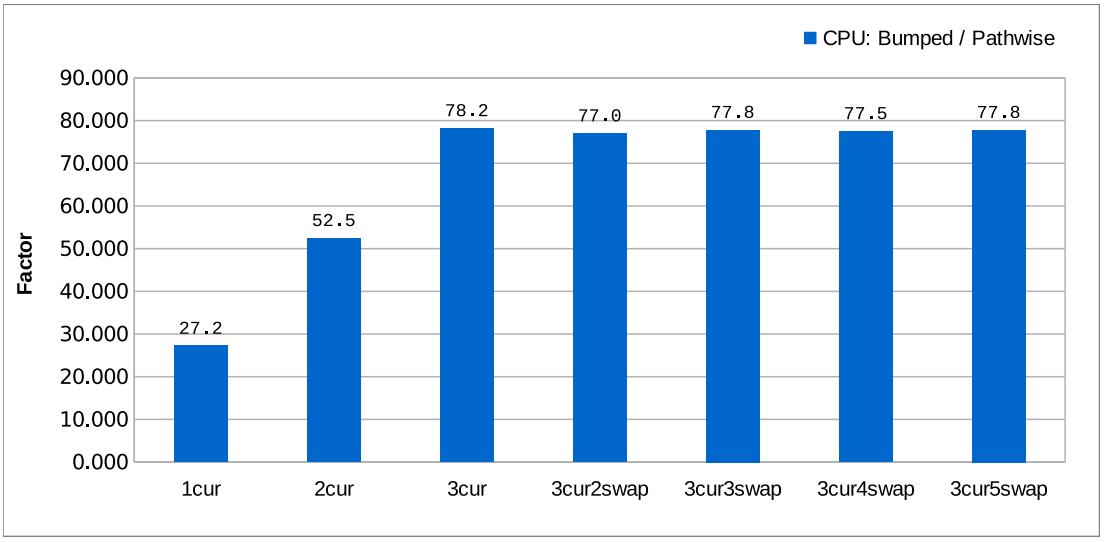


Figure 6.11: Speed up factors of the pathwise method for CVA Deltas. Speed-up stands for the ratio of the times spent on 1 CVA valuation + 35 bumps and 1 valuation & pathwise Deltas computed on the fly.

6.3 GPU Accelerated CVA Greeks

In the following section we show our results for the convergence and the speed-up of GPU accelerated CVA Greeks.

6.3.1 Convergence

We start by showing convergence of CVA Deltas and Vegas for a Swap triplet (EUR, USD, GBP) with identical details for each swap as in section 6.2.2. We show the convergence of pathwise Greeks towards bumped Greeks, both valued on the GPU. We display the error for Deltas as average normalized error $\frac{1}{\#points}\sum_{i=1}^{\#points}\frac{|\tilde{x}_i - x_i|}{|x_i|}$ for better visualization purposes (Figure 6.12). Pathwise Vegas convergence is displayed with simple normalized errors as we have a single σ per currency (Figure 6.13). Our obtained results are reasonably consistent with convergence results obtained with the single-threaded CPU implementation (section 6.2): pathwise Deltas have average normalized error $< 3\%$ with 3072 simulated paths and Vegas at the same time have $< 6\%$ normalized error. With an increasing number of paths the pathwise sensitivities converge to the bumped ones with remaining $< 1\%$ error due to floating point precision errors on the GPU (explained in Appendix A.5).

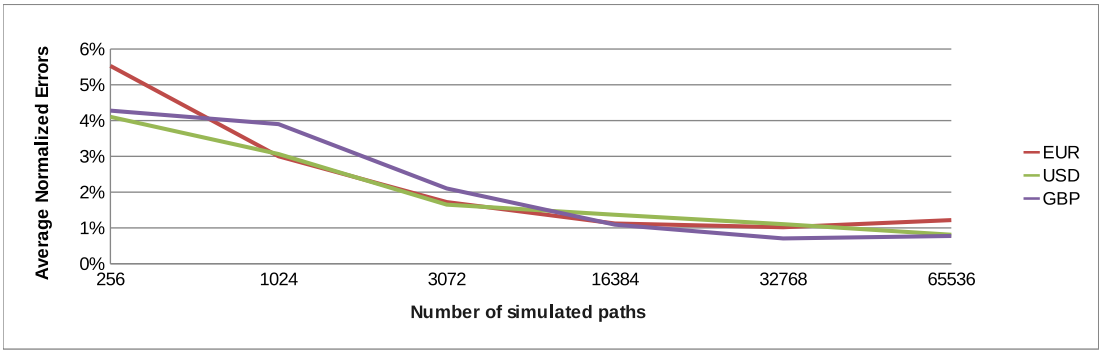


Figure 6.12: Convergence of GPU-based CVA Deltas. The average normalized error is shown per currency.

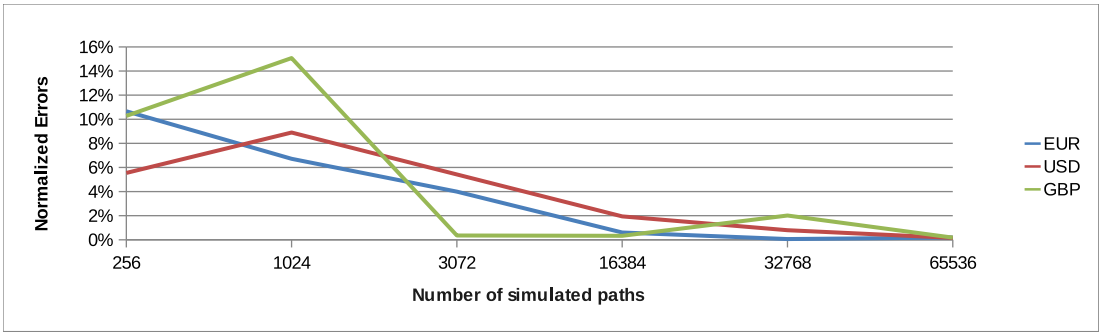


Figure 6.13: Convergence of GPU-based CVA Vegas. The normalized error is shown per currency.

6.3.2 Scaling of the Parallel Implementation

In this subsection we show how the computing time of the Greeks scales for the two evaluation methods previously discussed. The first test we did was to change the number of computed sensitivities. This was achieved by increasing the number of different currency swaps from 1 to 18 where each of them had:

- $T_N = 10$-year tenor
- $K = 4.6\%$ fixed rate
- Fixed and floating payments every 6 months
- Notional of 10,000 in the denominated currency

The test has shown that the relative speed up of GPU-based pathwise Greeks versus bumped Greeks increases up to 70 times (Figure 6.15) and has a potential to be larger if we have a

more diverse portfolio and compute even larger number of sensitivities. During our test the time spent on computation of Greeks (Deltas and Vegas) was brought from 717 seconds to approximately 10 seconds as it is shown in Figure 6.14.

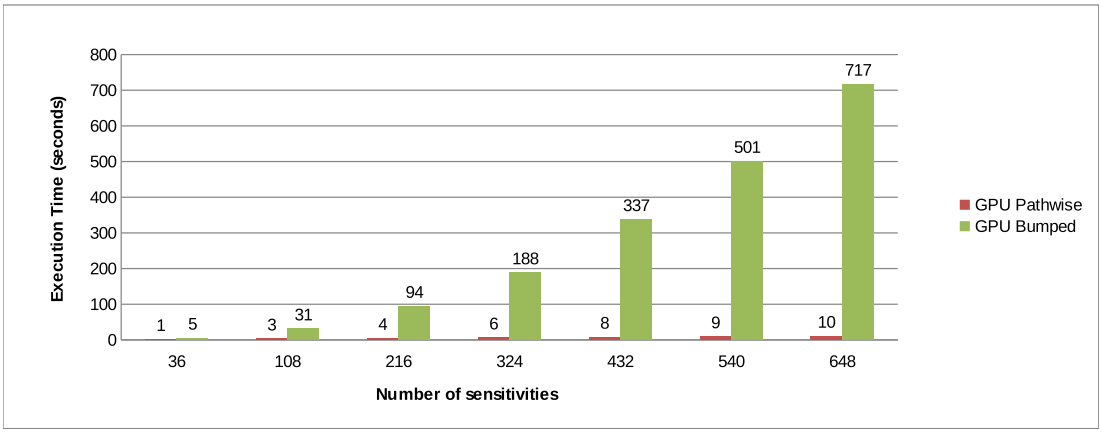


Figure 6.14: Scaling of CVA Greeks computation time with increasing sensitivities.

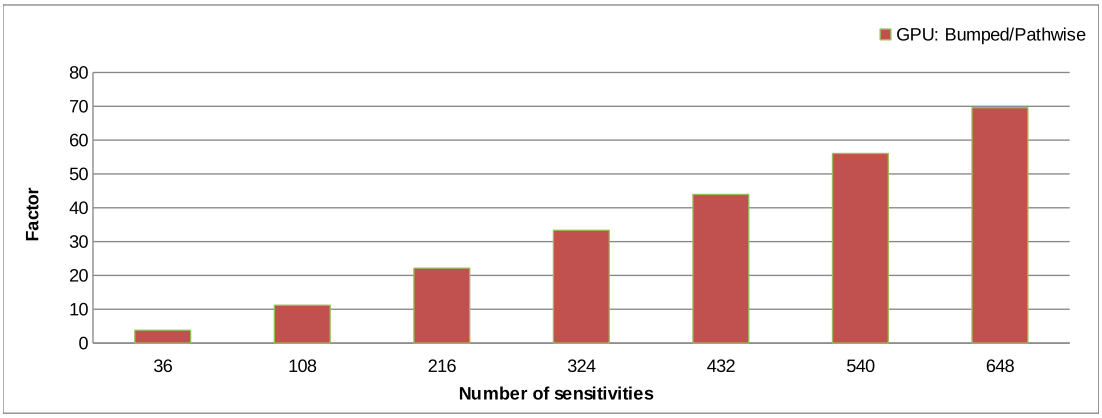


Figure 6.15: Speed-up of pathwise Greeks method on the GPU as a function of number of sensitivities.

Additionally, we investigated the stability of the speed up given by the pathwise Greeks method. We fixed the number of currencies in the portfolio to 3 (EUR, USD, GBP) and increased the number of identical swaps per currency from 1 to 1000. Figure 6.16 illustrates the observed execution times for both methods and Figure 6.17 shows their ratio. We see that the speed up factor stays stable while increasing the portfolio 1000 times.

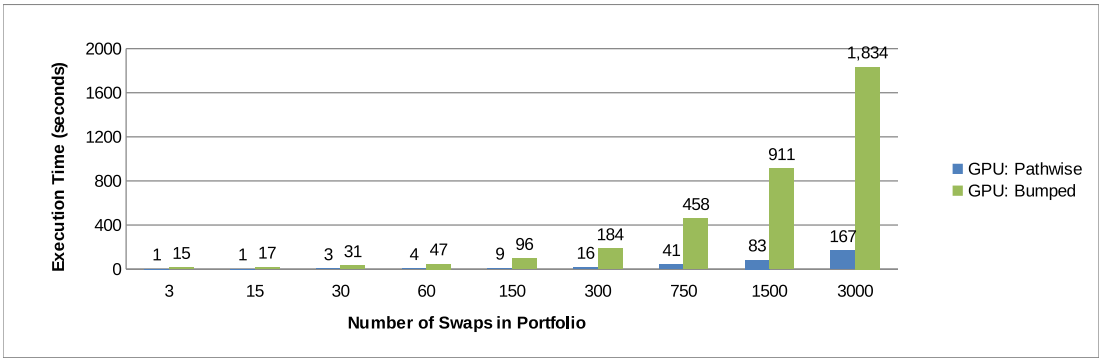


Figure 6.16: Scaling of computation time for GPU-based CVA Greeks with the number of swaps in portfolio.

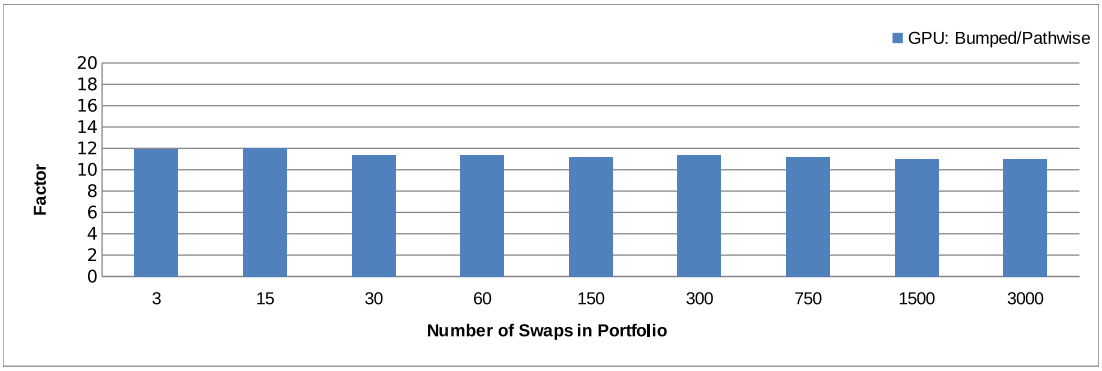


Figure 6.17: Speed up of pathwise Greeks method on the GPU as a function of the number of swaps in portfolio.

For the last test of scalability we gradually increased the number of simulated paths from 256 to 65536 while keeping the number of swaps constant and measured the ratio of the execution times between bumped and pathwise Greeks. The outcome is consistent with our previous results for CVA Greeks on CPU: the speed up factor stays approximately constant, despite the changing number of simulated paths (see Figure 6.18).

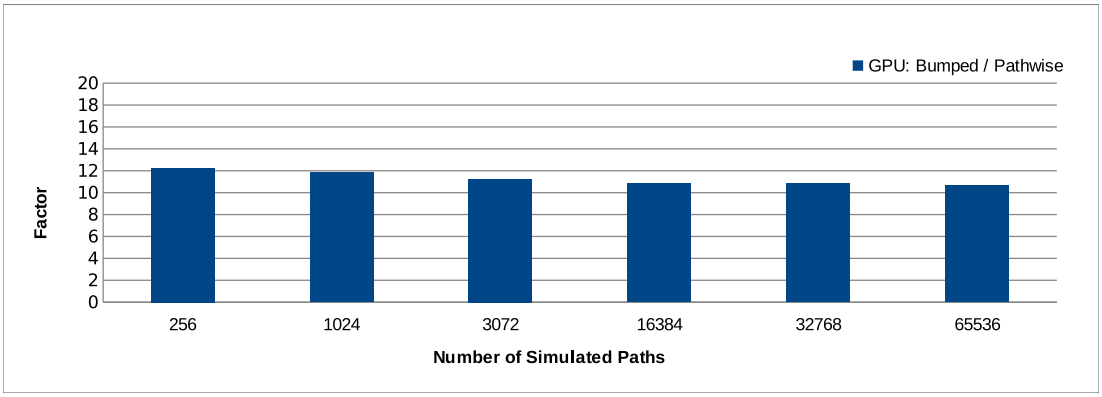


Figure 6.18: Speed-up of pathwise Greeks method on the GPU as a function of the number of simulated paths.

In general, we can conclude that the parallel pathwise method running on a GPU gives increasing speed up of the execution with increasing number of sensitivities and shows stable speed-up factors when increasing portfolio size (Figure 6.17) or number of simulated paths (Figure 6.18). Hence, we expect improvement in computational time when evaluating CVA Greeks on a full portfolio with > 20 currencies and $50,000$ trades of at least 70 times.

7

Discussion

In this thesis we explored the possible ways to accelerate the sensitivities computation for Counterparty Credit Valuation Adjustments. In chapter 3 we described three generic approaches to compute Greeks for various derivative instruments and decided to further investigate the pathwise Greeks method due to its suitability for Monte-Carlo simulation based pricing schemes for derivatives. Then, we found that the pathwise method can be applied to CVA sensitivities when certain smoothness conditions on the payoff and the underlying stochastic process are satisfied. As the largest part of the ING portfolio in question consists of rather simple, vanilla interest rate derivatives, priced under one or two factor Hull and White models - we can directly apply this method for CVA on these products.

On the other hand, this method might not be suitable for binary-type payoffs including indicator functions $\mathbf{1}_{\{\cdot\}}$, since they are not Lipschitz continuous and a problem arises when we make the second interchange of the derivative and the expectation (section 3.6). However, this is not the case if the expectation of our discounted payoff function has an analytic and differentiable expression. Then, we simply skip the second interchange and find the derivative after taking the expectation. Generally, if this is not possible, we can use the extension proposed by M.Giles, the Vibrato Monte-Carlo method, to get around this limitation.

There are two main benefits of the pathwise method: computational time speed up and added accuracy of the estimated Greeks. In the simulation we found that the pathwise Greeks can be computed up to 80 times faster than bumped ones with our CPU implementation. We additionally have shown that the evaluation of pathwise Greeks can be accelerated by using GPU co-processors, where the method brought a speed up of up to 70 times over the GPU-based bumped Greeks. Due to the excellent scalability of the method, the latter speed up

factor could increase even more if we constructed a larger and more diverse test portfolio with additional sensitivities to compute.

Furthermore, throughout our tests comparing pathwise and bumped Greeks, we showed that the accuracy difference between the methods is negligible when using small enough bumps. This yielded an easy way to test for correctness of the pathwise method in different test cases.

We conclude that, within its framework, the pathwise Greeks method offers a significant advantage for computing CVA sensitivities. This improvement allows for on-the-fly sensitivity computations instead of the usual overnight runs for large portfolios.

7.1 Future Work

There are several directions to expand this work. Currently, we have implemented and tested this method for CVA on interest rate swaps only. For full coverage the product range should be extended to include caps, floors, swaptions and more exotic interest rate and foreign-exchange derivatives.

Valuation of CVA and its Greeks on exotic derivatives is a rather complicated topic. If we decide to price the exotics in a Monte-Carlo fashion, then it would be logical to use a "pathwise within pathwise" approach to estimate CVA Greeks. This might be very expensive in both compute time and memory usage, but still gives an advantage compared to the bump & revalue approach for the same situation. On the other hand, some approximations like Gaussian quadrature or Longstaff & Schwartz methods might be used for pricing exotics. This would yield a convenient framework for estimating pathwise Greeks as in both cases the payoff functions are approximated by smooth and differentiable polynomials or basis functions. In any case, smart memory management and pre-computation of reused values will be essential for having reasonably short execution times.

Furthermore, very often the payoffs of exotic derivatives are path-dependent (which is well managed with the pathwise method) and have some binary-type features like range-accruals and barrier options. In this case it might be wise to have a good look at the applicability of Vibrato-Monte-Carlo method to handle the discontinuities while keeping the speed up of the pathwise method.

A second direction to expand this work is to assume more complicated dynamics of the underlying interest rate and credit models. One choice is to use the two-factor Hull and White model for interest rates which would enable proper pricing and Greeks evaluation of some of

the callable contracts. Another possibility is to use a stochastic credit model which introduces the needed tools to describe correlations between default probabilities and interest rates, a problem also known as the “wrong-way risk” (counterparty credit quality deteriorating while the amount of money owed to the bank is increasing). In my opinion, it should be possible to evaluate pathwise Deltas, Vegas and even sensitivities to changes in the credit spreads. Of course, one should be careful with these models: the pathwise method does not work well if the underlying processes have jumps, because analytic derivatives do not exist at jump times. There are two possible solutions for this issue: approximation of the derivative by smoothing the jump or using Malliavin-calculus methods (see book by Malliavin and Thalmaier [22]).

Another, completely different idea is to automate the differentiation process, as there are multiple implementations, often open-source, of the so-called automated differentiation libraries. These software packages scan through C, C++ or Fortran code, find marked functions and produce code for “copy-cat” functions computing analytic derivatives of the main functions with respect to their input parameters. At the time of writing this thesis, there was no implementation capable of handling CUDA code or global objects and variables in C++ codes. Nonetheless, it might be worthwhile to develop extensions for these libraries, as it would significantly decrease the amount of time spent coding and testing the sensitivity-evaluation functions.

Acknowledgements

I am grateful to all people who brought their contribution in one way or another to this work. I want to thank my supervisor Dr. Drona Kandhai who introduced me to the topic of Counterparty Credit Valuation Adjustments and Dr. Norbert Hari who was my direct supervisor during my internship at ING and was always ready to discuss issues on this topic and ideas that arose. I am also thankful to Prof. Rob Bisseling for discussions on parallel algorithms and to Tim Wood for introducing me to GPU programming and his advice on optimizing the code. Last but not least, I want to thank Andrada for her support and invaluable comments.

Appendix A

Appendices

A.1 Interchange of Derivative & Expectation

In the first section of the appendix we give the needed theory to support the interchange of derivative and the integrals in chapter 3. We start with some definitions:

Martingale

A discrete-time martingale is a stochastic process $\{X_i\}$ that satisfies for all n:

$$\mathbb{E}(|X_n|) < \infty \tag{A.1}$$

$$\mathbb{E}(X_{n+1}|X_1, ..., X_n) = X_n \tag{A.2}$$

Similarly, a continuous-time martingale is a stochastic process Y_t such that for all t:

$$\mathbb{E}(|X_t|) < \infty \tag{A.3}$$

$$\mathbb{E}(X_t|\{X_q, q \leq s\}) = X_s, \forall s \leq t \tag{A.4}$$

Brownian Motion

The Brownian motion, also known as the Wiener process W_t is characterized by three properties;

1. $W_0 = 0$
2. The function $t \to W_t$ is almost surely continuous
3. W_t has independent increments with $W_t - W_s$ $\mathcal{N}(0, t - s)$, for $0 \leq s < t$

where $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2.

An alternative definition is so-called Łévy characterization which says that the Wiener process is an almost surely continuous martingale with $W_0 = 0$ and quadratic variation $[W_t, W_t] = t$ (which means that $W_t^2 - t$ is also a martingale)

Expectation of a Lipschitz continuous function

In section 3.6 we use the argument that the expectation preserves Lipschitz continuity of a function. First we give a proof for a uniformly continuous function.

Take a uniformly continuous function $f(x, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Then we define function

$$g : \mathbb{R} \to \mathbb{R},\ g = \int_\Omega f(z(\omega), y) d\mathbb{P}$$

where $z : \Omega \to \mathbb{R}$ is a real-valued random variable mapping from the probability space Ω and $y \in \mathbb{R}$.

Take $y_0 \in \mathbb{R}$. By uniform continuity, given $\epsilon > 0$, $\exists \delta > 0$ s.t. $|f(x_0, y_0) - f(x, y)| < \epsilon$ for all $(x_0, y_0), (x, y) \in \mathbb{R} \times \mathbb{R}$ for which $||(x_0, y_0) - (x, y)|| < \delta$. Take an arbitrary $\omega \in \Omega$ and $y \in \mathbb{R}$ such that $|y_0 - y| < \delta$, then $||(z(\omega), y_0) - (z(\omega), y)|| < \delta$ which implies that $|f(z(\omega), y_0) - f(z(\omega), y)| < \epsilon$. Since this happens for arbitrary ω, this inequality holds for all $\omega \in \Omega$ and we have that:

$$\begin{aligned} \int_\Omega |f(z(\omega), y_0) - f(z(\omega), y)|\, d\mathbb{P} &< \epsilon \\ \left| \int_\Omega f(z(\omega), y_0) - f(z(\omega), y) d\mathbb{P} \right| &< \epsilon \\ |g(y_0) - g(y)| &< \epsilon \end{aligned}$$

Hence if the function f is uniformly continuous g is as well.

A function f is Lipschitz continuous (along its second argument) if:

$$\forall y \in \mathbb{R}, \forall \delta > 0, \exists c \in \mathbb{R}_0^+ :\ |f(\cdot, y + \delta) - f(\cdot, y)| \leq c \cdot \delta \tag{A.5}$$

Lipschitz continuity is implied by uniform continuity, hence we can directly conclude that expectation preserves it.

Conditions for the interchange of derivative and expectation

In the following we state the required conditions for the interchange of expectation and derivative:

$$\frac{\partial \mathbb{E}[f(X)]}{\partial \theta} = \mathbb{E}\left[\frac{\partial f(X)}{\partial \theta}\right] \tag{A.6}$$

Let $\{X_n, n \geq 0\}$ be a vector-valued state process representing, for example, the interest rate (short rate) process. The process (X_n) may be the discretization of a continuous-time process.

Denote the discounted payoff of our derivative $f(X),\ X = (X_1, \ldots, X_T)$, where T is the maturity and f is a real valued function. Thus the price of the security is $p = \mathbb{E}[f(X)]$

Now suppose (X_n) is a random function of parameter θ ranging in the open interval Θ. For the existence of pathwise derivatives, we require the following conditions to hold:

C1 $\dfrac{\partial X_n(\theta)}{\partial \theta} = \lim\limits_{h \to 0} \dfrac{X_n(\theta + h) - X_n(\theta)}{h}$ exists with probability 1.

C2 Take the set D_f of points where f is differentiable, then $\mathbb{P}(X(\theta) \in D_f) = 1$ for all $\theta \in \Theta$

C3 The function f is Lipschitz continuous and hence has an a.e. bounded derivative

The following theorem is an adapted and expanded (to multiple dimensions) version of original theorem by McShane [24]. For clarity of the proof we introduce direct dependence of f on $\omega \in \Omega$ skipping the random function X

Interchange of Derivative and Integral. Let B be a set in Ω and $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_K\}$ be an open set in $\mathbb{R}^K$. Let f be a real-valued function $(\omega, \alpha) \to f(\omega, \alpha)$ defined for $\omega \in \hat{B} = \{\forall \omega \in B :\ \mathbb{P}(\omega) > 0\}$ and all $\alpha \in \mathcal{A}$ such that for these ω and α, the partial derivatives

$$f_i(\omega, \alpha) = \frac{\partial f(\omega, \alpha)}{\partial \alpha_i},\ i \in \{1, ..., K\}, K < \infty \tag{A.7}$$

exist and also that for each $\alpha \in \mathcal{A}$ the integral

$$F(\alpha) = \int_B f(\omega, \alpha) d\mathbb{P}(\omega) \tag{A.8}$$

exists and is finite. If there exist functions b_i integrable over B such that for all $\omega \in \hat{B}$ and $\alpha \in \mathcal{A}$

$$|f_i(\omega, \alpha)| \leq b_i(\omega) \tag{A.9}$$

then, for each $\alpha \in \mathcal{A}$, F has a derivative with respect to α_i and

$$DF_i(\alpha) = \int_B f_i(\omega, \alpha) d\mathbb{P}(\omega) \tag{A.10}$$

Let α_i be a point of $\mathcal{A}_i$ and let $\alpha_{i,1}, \alpha_{i,2}, \alpha_{i,3}, ..., \alpha_{i,n}, ...$ be a sequence of points in $\mathcal{A}_i$ all different from α_i, but converging to α_i as n increases. Define $\beta_{i,n} = (\alpha_1, \alpha_2 ..., \alpha_{i,n}, ..., \alpha_K)$ to be α with perturbation in i direction.

$$q_{i,n}(\omega) = \frac{f(\omega, \beta_{i,n}) - f(\omega, \alpha)}{\alpha_{i,n} - \alpha_i} \tag{A.11}$$

by definition of F, we have

$$\frac{F(\beta_{i,n}) - F(\alpha)}{\alpha_{i,n} - \alpha_i} = \int_B q_{i,n}(\omega) d\mathbb{P}(\omega) \tag{A.12}$$

By the theorem of mean value, for all $\omega \in \hat{B}$ there is a number $\alpha(\omega)$ between α and $\beta_{i,n}$ such that $q_{i,n}(\omega) = f_i(\omega, \alpha(\omega))$ this gives us that:

$$|q_{i,n}(\omega)| \leq b_i(\omega), \ \forall \omega \in \hat{B} \tag{A.13}$$

Moreover the definition of derivative implies that the a.e. limit of $q_{i,n}(\omega)$ is $f_i(\omega, \alpha)$ for all. $\omega \in \hat{B}$. So, by the dominated convergence theorem, (see theorem 10-1 [24]) we have:

$$\lim_{n \to \infty} \int_B q_{i,n}(\omega) d\mathbb{P}(\omega) = \int_B f_i(\omega, \alpha) d\mathbb{P}(\omega) \tag{A.14}$$

Combining with (A.12) we obtain:

$$\lim_{n \to \infty} \frac{F(\beta_{i,n}) - F(\alpha)}{\alpha_{i,n} - \alpha_i} = \int_B f_i(\omega, \alpha) d\mathbb{P}(\omega) \tag{A.15}$$

Hence $\mathbb{E}[f_i(\alpha)]$ exists, and for $\beta_{i,n} \to \alpha$, $\beta_{i,n} \in \mathcal{A}$, $\forall i \in K$, the limit $\frac{F(\beta_{i,n}) - F(\alpha)}{\alpha_{i,n} - \alpha_i}$ exists and is equal to $\int_B f_i(\omega, \alpha) d\mathbb{P}(\omega)$. The latter being true for all $i \in \{1, ..., K\}$ □

A.2 Proof of SWAP final formula

In the following we show a quick derivation of the Swap pricing formula starting from the definition of a Swap as a series of Forward-Rate-Agreements as it was referred in the section 3.1. Recall that FRA price at time t is calculated as:

$$FRA(T, S, N, K) = N \left[P(t, T) \cdot \tau(T, S) \cdot K - P(t, T) + P(t, S) \right] \tag{A.16}$$

Hence:

$$\begin{aligned} Swap(t,\mathcal{T},N,K) &= \sum_{i=1}^{M} FRA(T_{i-1},T_i,N,K) \\ &= \sum_{i=1}^{M} N\cdot[P(t,T_i)\tau(T_{i-1},T_i)K - P(t,T_{i-1}) + P(t,T_i)] \\ &= N\cdot K\cdot\sum_{i=1}^{M}\tau(T_{i-1},T_i)\cdot P(t,T_i) - N\sum_{i=1}^{M}P(t,T_{i-1}) + N\sum_{i=1}^{M}P(t,T_i) \\ &= N\cdot\left[K\cdot\sum_{i=1}^{M}\tau(T_{i-1},T_i)P(t,T_i) - P(t,T_0) + P(t,T_M)\right] \end{aligned}$$

As wanted. □

A.3 Pathwise Greek Derivations

To efficiently use the pathwise Greek method and to decide if to use forward or backward differentiation, one has to investigate the structure of the whole chain of derivatives (Jacobian matrices).

In this section we derive and display the structure of Deltas and Vegas of a swaption. We will represent the swaption price using definition (3.8) on a specific, generated Monte-Carlo scenario as:

$$Swaption(t,T_0,\mathcal{T},x(T_0),\Theta_{mkt}) = D(t,x(T_0),T_0)\cdot(Swap(T_0,\mathcal{T},x(T_0),\Theta_{mkt}))^{+}$$

where $x(T_0)$ is the realisation of the state variable on this path and Θ_{mkt} is all market data used for swaption pricing. Then, we decompose the swap into a sum of floating (FlC) and fixed (FxC) coupons:

$$Swap(T_0,\mathcal{T},x(T_0),\Theta_{mkt}) = \sum_{i=0}^{N-1}(FxC(T_0,T_i,T_{i+1}) - FlC(T_0,T_i,T_{i+1}))$$

where the fixed coupon is dependent on the zero-coupon bond price P and the floating one is dependent on Forward rate ($FwdRt$):

$$\begin{aligned} FxC(T_0,T_i,T_{i+1}) &= N\cdot K\cdot P(T_0,T_{i+1})\cdot\tau(T_i,T_{i+1}) \\ FlC(T_0,T_i,T_{i+1}) &= N\cdot FwdRt(T_0,T_i,T_{i+1})\cdot\tau(T_i,T_{i+1}) \\ FwdRt(T_0,T_i,T_{i+1}) &= \frac{1}{\tau(T_i,T_{i+1})}\left(\frac{P(T_0,T_i)}{P(T_0,T_{i+1})} - 1\right) \end{aligned}$$

In the above we skipped the parameters $x(T_0)$ and Θ_{mkt} to shorten notation, but the dependence is still implied. Going forward, we use the zero-bond definition as in (3.21), just now our reference time of evaluation is t:

$$\begin{aligned} P(T_0, x(T_0), T_i) &= \exp\left[A(T_0,T_i) - B(T_0,T_i)\cdot x(T_0)\right] \\ &= \frac{P^M(t,T_i)}{P^M(t,T_0)} \cdot \exp\left[\frac{1}{2}\left[V(T_0,T_i) - V(t,T_i) + V(t,T_0)\right] - B(T_0,T_i)\cdot x(T_0)\right] \end{aligned}$$

with $B(T_0,T_i) = \dfrac{1-e^{-a\cdot\tau(T_i-T_0)}}{a}$ and $V(t,T)$ as defined in (3.20).

Swaption Deltas

Delta for a swaption can be called a derivative with respect to each of the quoted zero bond prices $P^M(0,Q_i)$ due to the direct relation to the yields (A.40). In the following derivations we distinguish between the time indices $\mathcal{T} = \{T_0, T_1, ..., T_N\}$ and $\mathcal{Q} = \{Q_0, Q_1, ..., Q_M\}$ - the first ones represent the reset and maturity dates of the swap coupons, while the second ones give the maturity dates of the quoted market zero-coupon bonds. As Deltas are quite straightforward to derive, we will immediately leap into the matrix formulation. Each $< \mathbf{BOLD} >$ entry stands for a vector of function values or parameters in the formulations below and $\begin{bmatrix}\cdot\\-\\\cdot\end{bmatrix}$ constructs represents the Jacobian matrices/vectors of corresponding partial derivatives. The overall expression for swaption Delta is of the form:

$$\begin{aligned} \left[\frac{\partial Swaption(t,T_0,\mathcal{T},x(T_0),\Theta_{mkt})}{\partial < \mathbf{P(t,Q_j)} >}\right] &= \mathbf{1}_{Swap(T_0,\mathcal{T},x(T_0),\Theta_{mkt})>0}\cdot D(t,T_0,x(T_0)) \\ &\cdot \left\{\left[\frac{\partial \mathbf{SWAP(T_0,\mathcal{T},x(T_0),\Theta_{mkt})}}{\partial ` < \mathbf{FxC(T_0,T_i)} >}\right]\right. \\ &\cdot \left[\frac{\partial < \mathbf{FxC(T_0,T_i)} >}{\partial < \mathbf{P(T_0,T_i)} >}\right]\cdot\left[\frac{\partial < \mathbf{P(T_0,T_i)} >}{\partial < \mathbf{P(t,Q_j)} >}\right] \\ &+ \left[\frac{\partial \mathbf{SWAP(T_0,\mathcal{T},x(T_0),\Theta_{mkt})}}{\partial < \mathbf{FlC(T_0,T_i)} >}\right]\cdot\left[\frac{\partial < \mathbf{FlC(T_0,T_i)} >}{\partial < \mathbf{FwdRt(T_0,T_i,T_{i+1})} >}\right] \\ &\cdot \left.\left[\frac{\partial < \mathbf{FwdRt(T_0,T_i,T_{i+1})} >}{\partial < \mathbf{P(T_0,T_i)} >}\right]\cdot\left[\frac{\partial < \mathbf{P(T_0,T_i)} >}{\partial < \mathbf{P(t,Q_j)} >}\right]\right\} \\ &+ (Swap(T_0,\mathcal{T},x(T_0),\Theta_{mkt}))^+\cdot\frac{\partial D(t,T_0,x(T_0))}{\partial < \mathbf{P(t,Q_j)} >} \end{aligned}$$

We can decompose the stochastic discount factor

$$D(t,T_0,x(T_0)) = \exp\left(-\int_t^{T_0} x(u)du - \int_t^{T_0}\phi(u)du\right) \tag{A.17}$$

derivative using equation (3.18) as:

$$\left[\frac{\partial D(t,T_0,x(T_0))}{\partial < \mathbf{P(t,Q_j)} >}\right] = \frac{D(t,T_0,x(T_0))}{P(t,T_0)} \cdot \left[\frac{\partial P(t,T_0)}{\partial < \mathbf{P(t,Q_j)} >}\right] \tag{A.18}$$

and the derivative of the Forward rate as:

$$\left[\frac{\partial < \mathbf{FwdRt(T_0,T_i,T_{i+1})} >}{\partial < \mathbf{P(t,Q_j)} >}\right] = \left[\frac{\partial < \mathbf{FwdRt(T_0,T_i,T_{i+1})} >}{\partial < \mathbf{P(T_0,T_i)} >}\right] \cdot \left[\frac{\partial < \mathbf{P(T_0,T_i)} >}{\partial < \mathbf{P(t,Q_j)} >}\right]$$

Then, the sensitivity of future zero-coupon bond $P(T_0,T_i)$ to time-zero bonds $P(t,Q_j)$ is decomposed to:

$$\left[\frac{\partial < \mathbf{P(T_0,T_i)} >}{\partial < \mathbf{P(t,Q_j)} >}\right] = \left[\frac{\partial < \mathbf{P(T_0,T_i)} >}{\partial < \mathbf{P(t,T_i)} >}\right] \cdot \left[\frac{\partial < \mathbf{P(t,T_i)} >}{\partial < \mathbf{P(t,Q_j)} >}\right] \tag{A.19}$$

where the Q_j times do not necessarily match the cash exchange dates T_i.

How the Jacobian matrices look like

Say we have an underlying swap with N fixed and N floating coupons with cash exchanges on $\mathcal{T} = \{T_1, ..., T_N\}$, $t =$ current time, $T_0 =$first fixing date. In the following we go though each of the mentioned Jacobians and discuss their structure. We start with the sensitivities of the swap price to the values of the coupons:

$$\left[\frac{\partial \mathbf{SWAP(t,\mathcal{T})}}{\partial` < \mathbf{FxC(t,T_i,T_{i+1})} >}\right] = \begin{bmatrix} 1, & ..., & 1 \end{bmatrix} \tag{A.20}$$

is a $[1 \times N]$ vector with ones for fixed coupons and equivalent one with -1s for the sensitivity to floating coupons (we assume that we pay floating, receive fixed).

The relation of fixed rate coupons and bond prices yields a diagonal $N \times N$ matrix as we have unique $P(T_0,T_i)$ per coupon:

$$\left[\frac{\partial < \mathbf{FxC(t,T_i,T_{i+1})} >}{\partial < \mathbf{P(T_0,T_i)} >}\right] = \begin{bmatrix} N \cdot K \cdot \tau(T_0,T_1) & 0 & \cdots & 0 \\ & N \cdot K \cdot & & \\ 0 & \tau(T_1,T_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & N \cdot K \cdot \tau(T_{N-1},T_N) \end{bmatrix} \tag{A.21}$$

The derivatives of floating rate coupons w.r.t. forward rates yield again a $N \times N$ diagonal matrix due to one-to-one correspondence of coupons and used forward rates.

$$\left[\frac{\partial < \mathbf{FlC}(\mathbf{t}, \mathbf{T_i}) >}{\partial < \mathbf{FwdRt}(\mathbf{T_0}, \mathbf{T_i}, \mathbf{T_{i+1}}) >}\right] = \begin{bmatrix} N \cdot \tau(T_0, T_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & N \cdot \tau(T_{N-1}, T_N) \end{bmatrix} \quad \text{(A.22)}$$

Then, the forward rates are constructed from pairs of zero bond prices, hence the Jacobian

$$\left[\frac{\partial < \mathbf{FwdRt}(\mathbf{T_0}, \mathbf{T_i}, \mathbf{T_{i+1}}) >}{\partial < \mathbf{P}(\mathbf{T_0}, \mathbf{T_i}) >}\right]$$

takes a form of $N \times N$ bi-diagonal matrix:

$$\begin{bmatrix} \frac{-1}{P(T_0,T_1)^2 \cdot \tau(T_0,T_1)} & 0 & \cdots & 0 \\ \frac{1}{\tau(T_1,T_2)P(T_0,T_2)} & \frac{-P(T_0,T_1)}{P(T_0,T_2)^2 \cdot \tau(T_1,T_2)} & \cdots & 0 \\ 0 & \frac{1}{\tau(T_2,T_3)P(T_0,T_3)} & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\tau(T_{N-1},T_N)P(T_0,T_N)} & \frac{-P(T_0,T_{N-1})}{P(T_0,T_N)^2 \cdot \tau(T_{N-1},T_N)} \end{bmatrix}$$

The future bond prices $P(T_0, T_i)$ have column + diagonal relationship to the time-t zero-coupon bond prices:

$$\left[\frac{\partial < \mathbf{P}(\mathbf{T_0}, \mathbf{T_i}) >}{\partial < \mathbf{P}(\mathbf{t}, \mathbf{T_i}) >}\right] = \begin{bmatrix} \frac{-P(T_0,T_1)}{P(t,T_0)} & \frac{P(T_0,T_1)}{P(t,T_1)} & 0 & \cdots & 0 \\ \frac{-P(T_0,T_2)}{P(t,T_0)} & 0 & \frac{P(T_0,T_2)}{P(t,T_2)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{-P(T_0,T_N)}{P(t,T_0)} & 0 & 0 & \cdots & \frac{P(T_0,T_N)}{P(t,T_N)} \end{bmatrix}$$

The final matrix is the Jacobian of the zero-bonds with time indices matching the coupon times with respect to zero-bonds with time indices matching the actual, underlying zero-bond maturity times. We are not displaying the internals of it as it depends on the interpolation scheme used (see Appendix A.4), but it should be bi-diagonal, as each of non-quoted bond prices is approximated by the two neighbouring ones.

$$\left[\frac{\partial < \mathbf{P}(\mathbf{t}, \mathbf{T_i}) >}{\partial < \mathbf{P}(\mathbf{t}, \mathbf{Q_j}) >}\right] = \begin{bmatrix} \frac{\partial P(t,T_{L_1})}{\partial P(t,Q_0)} & \cdots & \frac{\partial P(t,T_{L_1})}{\partial P(t,Q_M)} \\ \vdots & \cdots & \vdots \\ \frac{\partial P(t,T_{L_N})}{\partial P(t,Q_0)} & \cdots & \frac{\partial P(t,T_{L_N})}{\partial P(t,Q_M)} \end{bmatrix} \quad \text{(A.23)}$$

One should note that these matrices are extremely sparse as we are using the 1-factor Hull-White model to describe the dynamics of the interest rates in time. M.Giles has assumed the Libor Market model in his paper [15], where he had reasonably filled matrices and obtained significant efficiency gains by using the backward multiplication scheme. We do not have the large fill in our case and diagonal structures, hence it is not very beneficial to compute the matrices in the middle of the algorithm. In our implementation we have decided to restrict ourselves to the standard pathwise derivative method, using only vector operations as it is more memory efficient. When performing computations on the GPU - memory access overhead can become a very serious bottleneck.

Swaption Vegas

In the following we derive the analytic derivatives between the swaption price and calibrated piecewise constant $\sigma(t)$ points $\sigma_j,\ j \in \{1, ..., n\}$. For simplicity we derive this again for a single Monte-Carlo path in the swaption pricing scheme:

$$\frac{\partial Swaption(t, T_0, \mathcal{T}, x(T_0), \Theta_{mkt})}{\partial \sigma_j} = (Swap(T_0, \mathcal{T}, x(T_0), \Theta_{mkt}))^+ \frac{\partial D(t, T_0)}{\partial \sigma_j} + D(t, T_0) \cdot \frac{\partial (Swap(T_0, \mathcal{T}, x(T_0), \Theta_{mkt}))^+}{\partial \sigma_j}$$

Continuing the derivations for sensitivities:

$$\frac{\partial (Swap(T_0, \mathcal{T}, x(T_0), \Theta_{mkt}))^+}{\partial \sigma_j} = \mathbf{1}_{\{(Swap(T_0, \mathcal{T}, x(T_0), \Theta_{mkt}))>0\}} \cdot \sum_{i=0}^{N-1} \frac{\partial (FxC(T_0; T_i, T_{i+1}) - FlC(T_0; T_i, T_{i+1}))}{\partial \sigma_j}$$

For any i the following holds by the chain rule:

$$\begin{aligned} \frac{\partial FxC(T_0; T_i, T_{i+1})}{\partial \sigma_j} &= N \cdot \frac{\partial P(T_0, T_{i+1})}{\partial \sigma_j} \cdot \tau(T_i, T_{i+1}) \cdot K \\ \frac{\partial FlC(T_0; T_i, T_{i+1})}{\partial \sigma_j} &= N \cdot \tau(T_i, T_{i+1}) \frac{\partial FwdRt(T_0; T_i, T_{i+1})}{\partial \sigma_j} \end{aligned}$$

To continue the derivations, we revise some of the details from the model. We expressed (3.21) the zero-bond price as:

$$P(T_0, T_i) = \exp\left[A(T_0, T_i) - B(T_0, T_i) \cdot x(T_0)\right]$$

where:

$$\begin{aligned} A(T_0, T_i) &= \log \frac{P^M(t, T_i)}{P^M(t, T_0)} + \frac{1}{2}\left[V(T_0, T_1) - V(t, T_i) + V(t, T_0)\right] \\ B(T_0, T_i) &= \frac{1 - \exp\left(\tau(T_i - T_0) \cdot a\right)}{a} \end{aligned}$$

The $x(T_0)$ random component was defined by the SDE:

$$\begin{cases} dx(s) &= -ax(s)ds + \sigma(s)dW(s),\ s > t \\ x(t) &= 0 \end{cases}$$

using the Ito rule with $f(s,x) = xe^{as}$, we get a simpler solution:

$$x(s) = x(t)e^{-a(s-t)} + \int_t^s \sigma(u)e^{-a(s-u)}dW(u)$$

considering the piecewise constant volatility (section 3.3) and assuming that $s > t = t_1 = 0$ (for convenience), we obtain:

$$x(s) = \sum_{i=1}^{t_s}\left[\sigma_i \int_{t_i}^{t_{i+1}} e^{-a(s-u)}dW(u)\right] + \sigma_{t_{s+1}}\int_{t_s}^{s} e^{-a(s-u)}dW(u),\ t_s \le s < t_{s+1}$$

Then the derivative of the zero-coupon bond w.r.t. volatility σ_j can be derived:

$$\frac{\partial P(T_0,T_i)}{\partial \sigma_j} = P(T_0,T_i)\cdot\left(\frac{\partial A(T_0,T_i)}{\partial \sigma_j} - B(T_0,T_i)\cdot\frac{\partial x(T_0)}{\partial \sigma_j}\right) \tag{A.24}$$

$$\frac{\partial A(T_0,T_i)}{\partial \sigma_j} = \frac{1}{2}\left[\frac{\partial V(T_0,T_i)}{\partial \sigma_j} - \frac{\partial V(t,T_i)}{\partial \sigma_j} + \frac{\partial V(t,T_0)}{\partial \sigma_j}\right] \tag{A.25}$$

$$\frac{\partial x(T_0)}{\partial \sigma_j} = \sum_{i=1}^{n}\frac{\partial x(T_0,\sigma_i)}{\partial \sigma_j} \tag{A.26}$$

$$\frac{\partial x(T_0,\{\sigma_i\})}{\partial \sigma_j} = \mathbf{1}_{\{\sigma_i=\sigma_j\}}\int_{t_i}^{t_{i+1}\wedge T_0} e^{-a(T_0-u)}dW^0(u) \tag{A.27}$$

$$\frac{\partial V(T_0,T_i)}{\partial \sigma_j} = \sum_{k=1}^{n-1}\frac{\partial \tilde{V}(t_k,t_{k+1},\sigma_k)}{\partial \sigma_j} + \frac{\partial \tilde{V}(t_n,T,\sigma_n)}{\partial \sigma_j} \tag{A.28}$$

$$\frac{\partial \tilde{V}(t_k,t_{k+1})}{\partial \sigma_j} = \mathbf{1}_{\{\sigma_i=\sigma_j\}}\cdot\frac{2\tilde{V}(t_k,t_{k+1})}{\sigma_j} \tag{A.29}$$

where V and $\tilde{V}$ were defined initially in (3.3) and $t_i \wedge T_0 = \min(t_i, T_0)$.

The forward rate by definition is:

$$F(T_0;T_{i-1},T_i) = \frac{1}{\tau(T_{i-1},T_i)}\cdot\left(\frac{P(T_0,T_{i-1})}{P(T_0,T_i)} - 1\right)$$

hence its derivative with respect to the volatility σ_j is expressed as:

$$\frac{\partial F(T_0;T_{i-1},T_i)}{\partial \sigma_j} = \frac{1}{\tau(T_{i-1},T_i)}\left(\frac{1}{P(T_0,T_i)}\frac{\partial P(T_0,T_{i-1})}{\partial \sigma_j} - \frac{P(T_0,T_{i-1})}{P(T_0,T_i)^2}\cdot\frac{\partial P(T_0,T_i)}{\partial \sigma_j}\right)$$

where the zero-coupon bond Vegas have been already defined in (A.24).

Next we go to the discount factor term $D(t,T_0)$ in (A.17). The Vega of the discount factor is evaluated as:

$$\frac{\partial D(t,T_0)}{\partial \sigma_j} = D(t,T_0)\cdot\left(\frac{\partial\left(-\int_t^{T_0}\varphi(s)ds\right)}{\partial\sigma_j}+\frac{\partial\left(-\int_t^{T_0}x(s)ds\right)}{\partial\sigma_j}\right) \quad \text{(A.30)}$$

$$\frac{\partial\left(-\int_t^{T_0}\varphi(s)ds\right)}{\partial\sigma_j} = \frac{\partial\left[\frac{1}{2}V(t,t)-V(t,T_0)\right]}{\partial\sigma_j} \quad \text{(A.31)}$$

$$\frac{\partial\left(-\int_t^{T_0}x(s)ds\right)}{\partial\sigma_j} = \frac{\partial\left(-\int_t^{T_0}x(s)ds\right)}{\partial\sigma_j} = -\int_t^{T_0}\frac{\partial x(s)}{\partial\sigma_j}ds \quad \text{(A.32)}$$

and the latter two were computed in (A.28) and (A.26).

FX-Greeks

If the swaption was valued not in domestic currency as the case analysed so far, we have the additional term $FX(T)$ to convert the swaption value from foreign to domestic currency. This term is sensitive to both zero-coupon bond prices and volatilities of both currencies.

By definition, foreign-exchange rate is defined as an exponentiated difference between the two short rates with additional Brownian Motion component. The integration of the FX SDE in (3.26) yields:

$$FX(T) = FX(t)\cdot\exp\left[\int_t^T\left(r_d(t)-r_f(t)\right)dt+\int_t^T\sigma_S(t)dW_S(t)\right] \quad \text{(A.33)}$$

$$= FX(t)\cdot\underbrace{\exp\left[\int_t^T r_d(t)dt\right]}_{1/D_d(t,T,x_d(t))}\cdot\underbrace{\exp\left[-\int_t^T r_f(t)dt\right]}_{D_f(t,T,x_f(t))}\cdot\exp\left[\int_t^T\sigma_S(t)dW_S(t)\right]$$

These marked domestic and foreign discount factor parts are each sensitive to different sets of market data and we discuss them separately. When computing Deltas w.r.t. domestic data we obtain:

$$\frac{\partial FX(T)}{\partial P_d^M(t,Q_j)} = FX(T)\cdot\frac{-1}{D_d(t,T,x_d(t))}\cdot\frac{\partial D_d(t,T,x_d(t))}{\partial P_d^M(t,Q_j)} \quad \text{(A.34)}$$

where the last term is as in (A.18). In case of pathwise Vega w.r.t. $\sigma_{d,j}$ we have:

$$\frac{\partial FX(T)}{\partial \sigma_{d,j}} = FX(T)\cdot\frac{-1}{D_d(t,T,x_d(t))}\cdot\frac{\partial D_d(t,T,x_d(t))}{\partial \sigma_{d,j}} \quad \text{(A.35)}$$

with the rightmost derivative already computed in (A.30).

The sensitivities w.r.t. foreign Deltas are equivalent:

$$\frac{\partial FX(T)}{\partial P_f^M(t,Q_j)} = FX(T) \cdot \frac{-1}{D_f(t,T,x_f(t))} \cdot \frac{\partial D_f(t,T,x_f(t))}{\partial P_f^M(t,Q_j)} \tag{A.36}$$

where the last term is solved equivalently to (A.18). The main part of the foreign currency Vegas looks the same as for the domestic currency:

$$\frac{\partial FX(T)}{\partial \sigma_{f,j}} = FX(T) \cdot \frac{-1}{D_f(t,T,x_f(t))} \cdot \frac{\partial D_f(t,T,x_f(t))}{\partial \sigma_{f,j}} \tag{A.37}$$

but here the last derivative term is slightly different, as we have a modified drift of the state variable process (as in equation (3.26)):

$$dx_f(s) = -a_f x_f(s)ds + \sigma_f(s)\rho_{S,f}\sigma_S(s)ds + \sigma_f(s)dW_f^{\mathbb{Q}}(s),\ x_f(t) = 0 \tag{A.38}$$

integrating it with the help of function $f(x,s) = x \cdot e^{a_f s}$ we obtain:

$$x_f(s) = x_f(t)e^{-a_f(s-t)} + \int_t^s e^{-a_f(s-u)}\sigma_f(u)\rho_{S,f}\sigma_S(u)du + \int_t^s e^{-a_f(s-u)}\sigma_f(u)dW_f(u) \tag{A.39}$$

The latter expression can be rewritten, keeping in mind the piecewise constant volatility function $\sigma_f(s) = \sigma_{f,i},\ t_i \le s < t_{i+1},\ i \in \{1,...,N\}$ and assuming that $t = 0$ to:

$$x_f(s) = \sum_{i=1}^{n} \left[\sigma_{f,i} \left(\int_{t_i}^{t_{i+1}} e^{-a(s-u)}dW_f(u) + \int_{t_i}^{t_{i+1}} e^{-a(s-u)}\sigma_S(u)\rho_{S,f}du \right) \right.$$
$$\left. + \sigma_{f,n+1} \left(\int_{t_n}^{t} e^{-a(s-u)}dW_f(u) + \int_{t_n}^{t} e^{-a(s-u)}\sigma_S(u)\rho_{S,f}du \right) \right]$$

where $t_n < s < t_{n+1}$. Hence now, just like in (A.26) $x_f(s)$ is sensitive to one piece $\sigma_{f,i}$ of volatility function at a time. The rest of derivation is similar to single-currency case in the same array of equations.

A.4 Dangers of Interpolation for Greeks

In the derivation of (A.23) matrix we have skipped the internals of it. Here we would like to shortly discuss the different zero-coupon-bond price interpolation schemes and their differences.

The continuously compounded interest rate $Y(t,T)$ and discount factors $P(t,T)$ are related via:

$$Y(t,T) = \frac{\log P(t,T)}{\tau(T-t)} \tag{A.40}$$

$$P(t,T) = \exp\left[Y(t,T)\cdot(T-t)\right] \tag{A.41}$$

Say that two bonds maturing at times T_0 and T_1 are liquidly traded on the market. We want to find the price of a "custom" bond maturing in T years, $T_0 \leq T \leq T_1$. There are 3 main ways to interpolate the bond price:

1. Interpolation of the zero-yield curve points

$$Y(t,T) = \frac{\tau(T-T_0)Y(t,T_1) + (T_1-T)Y(t,T_0)}{\tau(T_1-T_0)} \tag{A.42}$$

2. Interpolation of the discount factors

$$P(t,T) = \frac{\tau(T-T_0)P(t,T_1) + (T_1-T)P(t,T_0)}{\tau(T_1-T_0)} \tag{A.43}$$

3. Interpolation of the log discount factors

$$\log P(t,T) = \frac{\tau(T-T_0)\log P(t,T_1) + (T_1-T)\log P(t,T_0)}{\tau(T_1-T_0)} \tag{A.44}$$

To show the differences of these methods we have picked a fictional case with 2 quoted zero coupon bonds (3-year and 5-year). The interpolated zero coupon bond prices do not differ much, even if we interpolate using different methods as it is shown in Figure A.1:

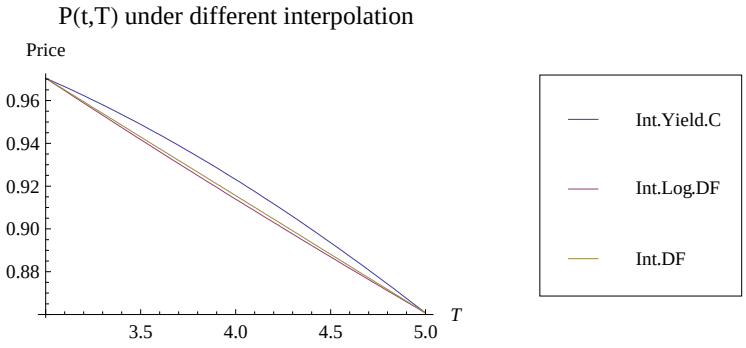
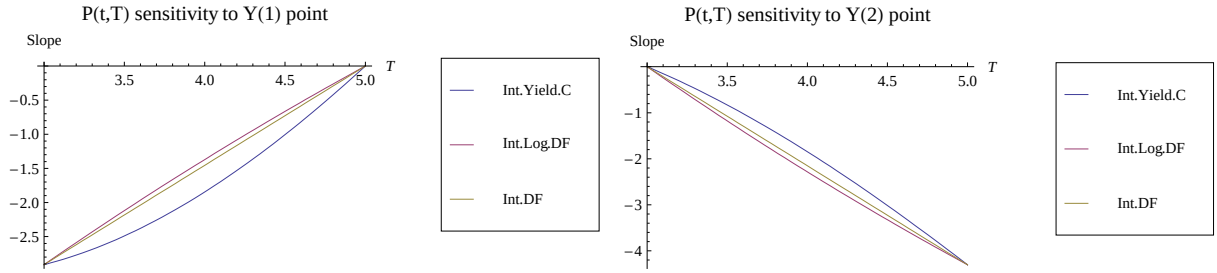


Figure A.1: Interpolated Bond Prices

but if we try to compare the sensitivities of the interpolated bond price with respect to the quoted bond prices, we obtain significant differences in the weights the neighbouring (quoted in the market) points carry, this is pictured in Figure A.2. In our case we have more a 40% difference (see Figure A.3) in the computed Greeks.



(a) Sensitivity to 3y point

(b) Sensitivity to 5y point

Figure A.2: Greeks of Interpolated Bond Price. Left figure shows the sensitivity of the interpolated price to the 3-year point and right figure shows sensitivity to the 5-year point under different interpolation methods.

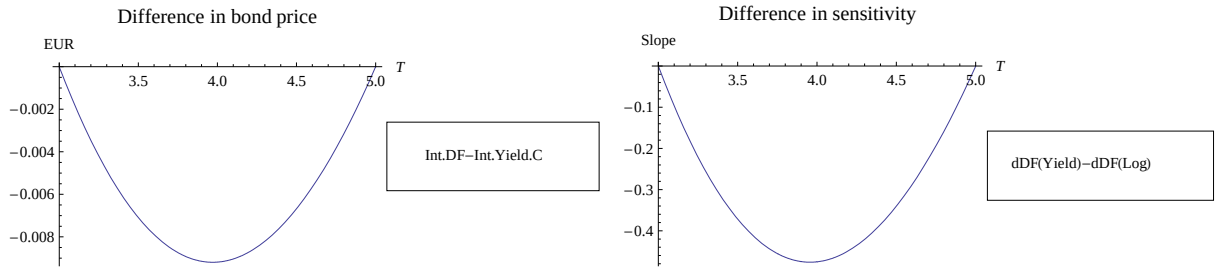


Figure A.3: Bond price and sensitivity differences arising from chosen interpolation methods. Left figure shows the difference between direct discount factor and yield curve interpolated zero bond prices depending on value of T. The right figure shows the difference between yield curve and discount factor interpolation based sensitivities.

Hence even if the interpolation method does not yield much difference in the interpolated bond price, it does matter in the case of Greeks computation.

The practitioner should (at least) be consistent throughout the system how things are being interpolated. Additionally, if one has the freedom to choose the method of interpolation, the smartest choice would be to interpolate pure zero-coupon-bond prices as hedging is done

directly with them.

A.5 GPU: single precision limitations

In chapter 5 we have mentioned the floating point precision problem when performing bumped Greeks evaluation using GPUs:

$$\frac{\partial CVA}{\partial \theta} = \frac{CVA(\theta) - CVA(\theta + h)}{h}$$

The problem rises from the fact that the currently supported parallel add operations of NVIDIA cards are only single-precision. As one is performing the single-precision operations, only 23-bits are available for the "fraction" part of the number. In this case the numerical "epsilon" (smallest x, s.t. $1 + x > 1$) is around $1.9E - 7$ (see article by Goldberg [17]). Hence, we can, roughly, add around 7 digits of 2 numbers correctly:

$$11,111,111 + 1.31516 = 11,111,112.0$$

and the rest of the number is rounded. Additionally, if we don't use sorting procedures, due to non-associativity of floating point operations, the parallel add operations return numbers that have the tail part (after the 7 digits) - somewhat random. Hence as we were performing CVA calculation, thousands of single-precision numbers were added and these errors left us with precise 6-digits of the final answer.

This difference does not matter for CVA pricing, as we are talking about missing cents from multi-million Euro values, but this rounding and parallel add error affects the Greeks. Say that the bump $h = 1E-10$ and the parallel-add error is of size $1E-6\times$CVA size. Then, if the true Greek in question is 20,000 (i.e. sensitivity of CVA to movement of some yield curve point) and $CVA = 2000$, then the finite-difference approximation of it will be:

$$\tilde{\Delta} = \frac{CVA(\theta) - CVA(\theta + h) \pm \overbrace{2 \cdot (2000 \cdot 1E - 6)}^{\text{p-add error}}}{1E - 10} = 20,000 \pm 40,000,000 \qquad \text{(A.45)}$$

which makes our Greek estimate unusable. The solution is to use a larger bump size, like $h = 1E - 4$, then the parallel add error, in this case, would become ± 40.

Note that Greeks are usually normalized to show changes in CVA value if a yield curve point moves by 1 basis point $= 1E - 4$, hence the final sensitivity number used by the trader would be 2 ± 0.004 and this error is small enough to ignore, as CVA is an approximation itself.

On the other hand, making a larger bump might give biased Greek due to higher-order effects in the CVA pricing formula, which might be undesirable in practice.

A.6 Development Platform Details

All the code was written and tested on a computer provided by ING with:

- Intel Xeon 3.2GHz (8-core) CPU
- NVIDIA Tesla C2070 (Fermi) GPU card
- 16GB memory.

References

[1] Basel III: A global regulatory framework for more resilient banks and banking systems. Technical report, Basel Committee on Banking Supervision, BIS. 1

[2] L. Abbas-Turki and B. Lapeyre. American options pricing on multi-core graphic cards. In *Business Intelligence and Financial Engineering, 2009. BIFE'09. International Conference on*, pages 307–311. IEEE, 2009. 30

[3] C. Albanese, T. Bellaj, G. Gimonet, and G. Pietronero. Coherent global market simulations for counterparty credit risk, Oct. 2010. 25

[4] BIS. Semiannual OTC derivatives statistics at end-December 2010. URL http://www.bis.org/statistics/derstats.htm. 4

[5] D. Brigo. CCFEA. Credit and Default Modeling. Unit 7: Counterparty Risk with Stochastic Dynamical Models: Impact of Volatilities and Correlations. *courses.essex.ac.uk*, 2009. URL http://courses.essex.ac.uk/CF/CF907/unit7essex.pdf.

[6] D. Brigo and F. Mercurio. *Interest rate models: theory and practice: with smile, inflation, and credit*. Springer Verlag, 2006. ISBN 3540221492. 6, 7, 9, 11, 12, 13, 14, 16, 25

[7] D. Brigo and A. Pallavicini. Counterparty risk and Contingent CDS valuation under correlation between interest-rates and default. *papers.ssrn.com*, (August 2006), 2008.

[8] D. Brigo, A. Pallavicini, and R. Torresetti. Credit models and the crisis, or: how I learned to stop worrying and love the CDOs. *Quantitative Finance Papers*, pages 1–66, 2009.

[9] R. Caflisch. Monte Carlo and Quasi-Monte Carlo methods. *Acta numerica*, 7(-1):1–49, 1998. 18

[10] L. Capriotti and M. Giles. Fast correlation greeks by adjoint algorithmic differentiation. *Quantitative Finance Papers*, 2010.

[11] N. H. Carlos A. Gonzalez Sterling. Technical Note on Hull and White Model with piecewise constant volatility. Technical report, ING, Amsterdam, 2009. 14

[12] N. Chen and P. Glasserman. Malliavin Greeks without Malliavin calculus, July 2007. ISSN 0304-4149.

[13] J. Chong, K. Keutzer, and M. Dixon. Acceleration of Market Value-at-Risk Estimation. *papers.ssrn.com*. 30

[14] D. Duffy. Monte Carlo Frameworks: Building Customisable High-Performance C++ Applications. 2009.

[15] M. Giles and P. Glasserman. Smoking adjoints: Fast monte carlo greeks. *Risk*, 19:88–92, Jan. 2006. 43, 61

[16] P. Glasserman. *Monte Carlo methods in financial engineering*. Springer Verlag, 2004. 17

[17] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991. 67

[18] J. Gregory. *Counterparty credit risk: the new challenge for global financial markets*. Wiley, 2010. 6

[19] L. Grzelak, C. Oosterlee, and S. Van Weeren. Extension of stochastic volatility equity models with the Hull-White interest rate process. *Quantitative Finance*, pages 1–17, Dec. 2009. ISSN 1469-7688. doi: 10.1080/14697680903170809.

[20] L. A. Grzelak. On Cross-Currency Models with Stochastic Volatility and Correlated Interest Rates Multi-Currency Model with Short-Rate Interest Rates. *Centrum*, pages 1–27, 2011. 14, 15

[21] I. Kechagioglou. Credit/Interest rate hybrid models of the short rate for pricing counterparty risk adjustment. *papers.ssrn.com*, (September):1–41, 2009.

[22] P. Malliavin and A. Thalmaier. *Stochastic calculus of variations in mathematical finance*, volume 3. Springer Verlag, 2006. 51

[23] D. McLeish. *Monte Carlo simulation and finance*, volume 276. Wiley, 2005.

[24] E. J. McShane. *Unified integration*. 1989. 55, 56

[25] NVIDIA. CUDA programming Guide 4.0. Technical report, NVIDIA, 2011. 32, 33

[26] *Monte Carlo evaluation of sensitivities in computational finance.*, 2007. Oxford University Computing Laboratory, Oxford-Man Institute of Quantitative Finance. 18, 22

[27] N. Packham. Correlation parameterization and calibration for the LIBOR market model. (4122659), 2005. URL `http://packham.net/data/Thesis.pdf`.

[28] D. Patterson and J. Hennessy. *Computer organization and design: the hardware/software interface*. Morgan Kaufmann, 2008. 31

[29] M. Pykhtin. Counterparty Credit Risk. *Wiley Online Library*.

[30] S. Ryoo, C. Rodrigues, S. Baghsorkhi, S. Stone, D. Kirk, and W. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 73–82. ACM, 2008. 32

[31] D. Williams. *Probability with martingales*. Cambridge Univ Pr, 1991. ISBN 0521406056.