Calibration Of Multi-Period Single-Factor Gaussian Copula Models For CDO Pricing

by

Max S. Kaznady

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Calibration Of Multi-Period Single-Factor Gaussian Copula Models For CDO Pricing

Max S. Kaznady

Master of Science

Graduate Department of Computer Science

University of Toronto

April 2011

A Collaterized Debt Obligation (CDO) is a multi-name credit derivative, which redistributes the risk of defaults in a collection (also known as the basket or pool) of underlying assets, into fixed income securities, known as the tranches. Each tranche is associated with a certain fraction of first-to-default underlyings. Synthetic CDOs have a pool that consists of Credit Default Swaps (CDSs). If all CDSs have equal notionals, then the pool is termed homogeneous.

Single-period single-factor copula models approximate the probability of underlying defaults using a percentile to percentile transformation, and incorporate the underlying pool correlation structure for multi-name credit derivatives, such as CDOs. Currently, such models are static in time and do not calibrate consistently against market quotes. Recently Jackson, Kreinin and Zhang (JKZ) proposed a discrete-time Multi-period Single-factor Copula Model (MSCM), for which the default correlations are time-independent, allowing the model to systematically fit the market quotes. For homogeneous pools, the JKZ MSCM provides a chaining technique, which avoids expensive Monte Carlo simulation, previously used by other multi-period copula models. However, even for homogeneous pools, the tree-based example of MSCM presented by JKZ has three drawbacks: derivatives are difficult to obtain for calibration, probabilities of the copula correlation parameter paths do not accurately represent its movements, and the model is not extremely parsimonious.

In this thesis, we develop an improved implementation of MSCM: we use an alternative multi-path parameterization of the copula correlation parameter paths and the corresponding probabilities. This allows us to calculate first-order derivatives for the MSCM in closed form for a reasonable range of parameter values, and to vary the number of parameters used by the model. We also develop and implement a practical error control heuristic for the error in the pool loss probabilities and their derivatives. We develop theoretical error bounds for the pool loss probabilities as well. We also explore a variety of optimization algorithms and demonstrate that the improved MSCM is inexpensive to calibrate. In addition, we show how MSCM calibrates to CDO data for periods before, during and after the September 2008 stock market crash.

Dedication

To my family.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Kenneth R. Jackson, for his deep insight into the problem area and for his constant support and thoughtful supervision of my research! I would also like to thank all the graduate students in the Numerical Analysis and Scientific Computing group for creating a great working atmosphere, and to all the graduate students who kept me company outside of my research.

Special thanks to Professors Christina C. Christara, Alex Kreinin and Sebastian Jaimungal for discussing possible research directions with me at the start of my program. Also, many thanks to Doctor Tom Fairgrieve for reading this thesis.

I would also like to thank:

- the Natural Sciences and Engineering Research Council (NSERC) of Canada for graduate funding support (Canada Graham-Bell Master's Award, CGS M);
- the University of Toronto Rotman School of Business for providing me with free access to the Bloomberg and Reuters Thomson data streams;
- the University of Toronto Computer Science Department for providing me with computing facilities and for extra financial support in the form of two departmental scholarships.

Last, but not least, I would like to thank my family for their love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The valuation of a credit derivative[1] is associated with the credit risk of the underlying asset, or a collection of assets, also known as the pool. Hence there are two classes of credit derivatives: single-name and multi-name, respectively. The mathematical modeling of credit derivatives is very complex in nature; the stock market crash of 2008-2009 was blamed, in part, on the quantitative models for credit derivatives. Increasingly sophisticated models are being developed, which attempt to improve the fit to market quotes by better capturing market dynamics.

In this thesis, we improve on the implementation of the example of the Multi-period Single-factor Copula Model (MSCM), originally proposed by Jackson, Kreinin and Zhang (JKZ) [11]. We provide an alternative multi-path parameterization to MSCM, which allows us to improve the model's existing ability to capture market dynamics over time, and which further allows us to calibrate the model in reasonable time by using optimization routines which exploit our ability to write the first-order derivatives of the objective function in closed form for a reasonable range of parameter values. We also develop an error control heuristic for the error in the pool loss probabilities and their derivatives, as well as a useful theoretical result about the errors in pool loss probabilities. In addi-

[1] A derivative is a financial instrument whose value is derived from some underlying asset, for example, an option on a stock.

tion, we examine the behavior of the MSCM on market data for periods before, during and after the September 2008 stock market crash, and demonstrate that a parsimonious parameterization of MSCM fits the market quotes better than the industry-standard single-period single-factor copula model.

1.1 Mechanism Of Collaterized Debt Obligations

To understand Collaterized Debt Obligations (CDOs), depicted in Figure 1.2, we must first understand simpler single-name Credit Default Swaps (CDSs), shown in Figure 1.1. A CDS is a financial contract, in which the underlying asset (also referred to as the underlying credit, or just the underlying) has a certain market value (also called the notional, face or par value), and might default before the maturity (or expiry) of the contract at time T. The simplest example of such an asset is a bond issued by a company or a firm. The owner of the asset wants insurance against a credit event, such as the bankruptcy of the company and its associated default on the bond interest payments and/or principal repayment. Consequently, the owner of the asset enters a CDS contract, in which they are the buyer of protection, and pay fixed premiums[2], quoted as a fraction of the notional (insured value of the underlying asset) usually expressed in basis points (bps), to the seller of protection. In case of a default, the seller of the CDS pays back the notional to the buyer, and retains any market value that the asset still has. In practice, the underlying might not be worthless after a default[3]. Premium payments stop after the credit event and the CDS contract terminates [1].

A CDO is a multi-name credit derivative, which redistributes the risk of defaults in a collection (also known as the basket or pool) of underlying assets, into fixed income securities, known as the tranches [2]. Tranches are ranked in order of seniority; in increasing

[2]Premiums are usually paid quarterly.

[3]A realistic market assumption is that about 40% of the underlying asset's value can be recovered after a default. This fraction is known as the recovery rate.

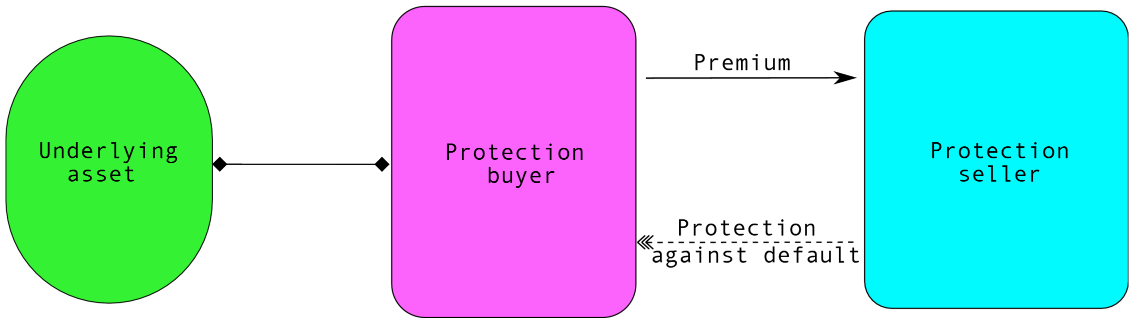

Figure 1.1: Mechanism of a Credit Default Swap.

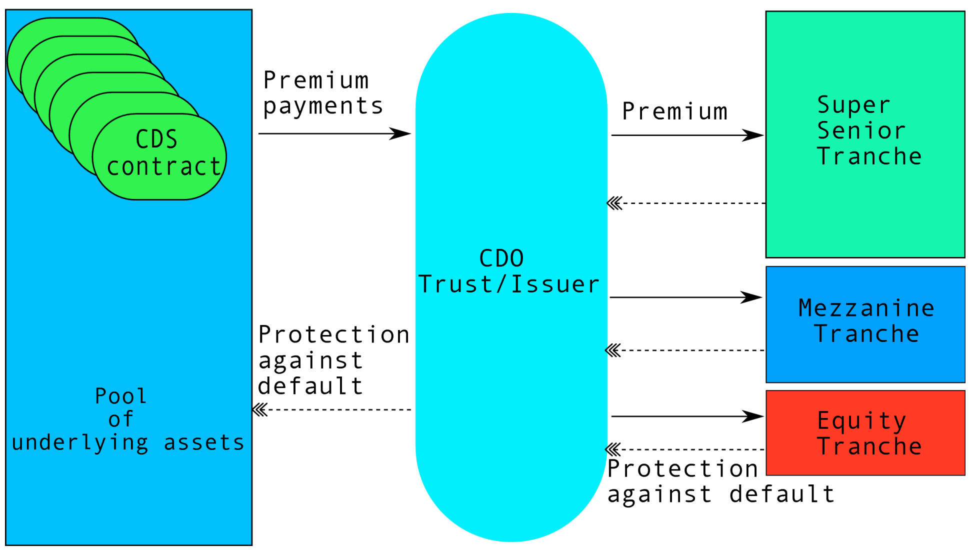

Figure 1.2: Mechanism of a synthetic Collaterized Debt Obligation.

order, we have the Equity, Mezzanine and Super Senior tranches. Each tranche is associated with a certain fraction of defaults, specified by attachment $a^{(\mathrm{tr})}$ and detachment $b^{(\mathrm{tr})}$ points in percent, where tr indexes the tranche; the difference $S^{(\mathrm{tr})} = b^{(\mathrm{tr})} - a^{(\mathrm{tr})}$ is known as the tranche size. For example, if the Equity tranche has an attachment point of 0% and a detachment point of 3%, then this tranche covers the first 3% of defaults in the pool. If more than 3% of underlyings default, then the next tranche starts covering the losses, and so on. The issuer of the CDO is known as the trust. The trust sells tranches to investors, who are ultimately responsible for covering portfolio losses, as the underlyings associated with their tranche begin to default, or experience other credit events, such as credit downgrades.

A CDO is called synthetic if the underlying pool consists of CDSs. If all CDSs have the same notional N, then the pool is called homogeneous. We illustrate the functionality of a synthetic CDO with homogeneous pool using the following example: consider an investor in a Mezzanine tranche[4] with an attachment point of 3% and a detachment point of 7%. If there are K underlyings, each with a notional N, then the investor receives payments of

$$\text{tranche spread} \cdot \text{Notional}, \tag{1.1}$$

usually quarterly. If a CDS defaults, the investor in the Equity tranche must cover the loss. Once the first 3% of underlying CDSs have defaulted, the contract of the investor in the Equity tranche is terminated[5]. The investor in the Mezzanine tranche now begins to cover the losses, and so on. The tranches are ranked by risk, with the Equity tranche being the riskiest tranche to enter, and the Super Senior tranche being the least risky.

1.2 Brief Literature Overview

The Gaussian factor copula model is a type of structural model used to model credit risk; structural models were originally introduced by Merton [40] and associate risk with economic driving forces. On the other hand, reduced form models characterize defaults via a stochastic process, that generally has no associated economic interpretation [47, 46]. Gaussian single-factor copula models have become an industry standard due to their computational efficiency. The earliest cited use of Gaussian copula models was to characterize the pool loss distribution of loans in 1987 by Vasicek [41]. The first cited application to multi-name credit derivatives was by Li [42] in 2000. Many generalizations

[4]Mezzanine tranches usually refer to the range of tranches between the Equity and Super Senior tranche.

[5]Premium payments stop; the investor covers the losses and collects the recovery values.

of Gaussian copula models followed [43, 44, 45]; for example, the copula approach does not have to use a Gaussian probability density [29].

For single-name credit derivatives, it is not difficult to associate the probability of default with the value of the credit derivative via some probability model. However, multi-name credit derivatives require the added knowledge about the correlations between the defaults of the underlyings. This can be added to the structural model via another driving factor, which the copula relates to the probability of default. For CDO pricing, structural models have been known to provide poor fits to market quotes, because the driving factors assumed either constant default correlation over time, or constant default correlation across CDO tranches [2]. In reality, these correlations change spatially over the tranches, and also over time for each tranche [2]; the former change in correlation is commonly known as the tranche correlation smile, and the latter is simply referred to as the correlation smile. The tranche correlation problem can be avoided by simply performing calibration over tranches with roughly the same tranche implied correlation[6]. However, single-period single-factor Gaussian copula models still assume that the tranche implied correlation is fixed over time.

Chaining techniques have been proposed, which link a number of single-period single-factor copulas, responsible for each time period, into a multi-period single-factor copula model, thus combating the problem of the correlation smile by associating a different value for the copula correlation parameter with each time period[7]. However, these models suffer a computational drawback, in that a unique driving factor is associated with each period, and in order to compute the expected pool loss, multi-dimensional integration has to be carried out over all driving factors. Monte Carlo (MC) simulation is typically used to approximate this integration. Hence, in practice, these chaining techniques do

[6]It should also be noted that the copula correlation parameter represents the true tranche implied correlation.

[7]In order to compute the expected spread, we also need to associate a discrete probability measure with possible values of the copula correlation parameter over time, to model the market dynamics of the copula correlation parameter.

not generalize well to more than two periods. The original extension of the single-period single-factor copula model was proposed by Fingers [33] and soon after Andersen [34] and Sidenius [35] popularized construction of multi-period single-factor copula models.

Jackson, Kreinin and Zhang [11] have recently proposed a recursion relationship which avoids MC simulation in multi-period Gaussian copulas for homogeneous pools, where all underlying assets have the same correlation[8]; for non homogeneous pools, a combinatorial problem arises, which, to the best of our knowledge, cannot be solved in polynomial time, so the proposed model is applicable only to homogeneous pools. The example of the computationally tractable MSCM in [11] uses a binary tree structure to parameterize the time evolution of the copula correlation parameter. This example suffers three drawbacks: first-order derivatives are difficult to obtain for calibration, probabilities of the copula correlation parameter paths do not accurately represent its movements, and the number of model parameters cannot be easily varied to keep the model parsimonious for different calibration data sets.

1.3 Main Contributions

In this thesis, we develop an improved implementation of the MSCM originally proposed by Jackson, Kreinin and Zhang [11]. The original implementation used an optimization method without derivatives for calibration; this is one of the reasons why calibration is very time consuming. We use an alternative multi-path parameterization of the copula correlation paths and the corresponding probabilities. This multi-path parameterization allows us to formulate the first-order derivatives associated with the MSCM in closed form, for all reasonable parameter values; in the original binary tree implementation, settings of the copula correlation parameters in consecutive periods depended on the settings in previous periods (see Figure A.2 for example), and this created a complicated

[8]This recursion relationship replaces expensive multidimensional integration by a series of one dimensional integrals, for which we develop a quadrature routine in this thesis.

dependence relationship in the derivatives. With the multi-path parameterization, the copula correlation parameters can switch to any reasonable value with a unique transition probability. Hence we can write the first-order derivatives of each period independently from the other periods.

Multi-path parameterization allowed us to generalize the model to any number of periods with any number of copula correlation parameter values per period, something which was not practical with the binary tree implementation[9]. The original parameterization also suffered a computational drawback; the optimization routine would set the copula correlation parameter values to be outside of the unit interval, whence the copula correlation parameter values had to be adjusted.

The derivative values associated with any implementation are expensive to compute. We explore a variety of optimization algorithms to determine which methods are both robust and computationally efficient. Finally, we explore the model's ability to match market data over the periods before, during and after the 2008-2009 stock market crash.

1.4 Thesis Outline

In this thesis we explore efficient implementations of MSCM and demonstrate numerically that our improved implementation is relatively inexpensive to calibrate. We also assess model performance on data collected before, during and after the 2008-2009 stock market crash and discuss future research directions.

In Chapter 2 we provide the necessary background for this thesis. Since our research draws from different areas, the reader may refer to this chapter if they feel that some parts of the model discussion are new to them.

In Chapter 3 we develop the alternative multi-path parameterization for the MSCM and compute closed forms of the first-order derivatives associated with these parameters.

[9]Even if we can determine a general structure for the original tree model, it is still difficult to vary the number of copula correlation parameter paths per period.

These derivatives can be computed for all reasonable ranges of parameter values. We discuss what it means for the range of parameter values to be reasonable in the same chapter. The chapter also lays out the framework used later to parallelize the software implementation to improve computational efficiency.

In Chapter 4 we develop a quadrature heuristic used for one dimensional integration over each common factor in the structural model and determine theoretical error bounds for the numerically computed default probabilities. We argue that, in practice, it is very likely that our error control heuristic produces an error in pool loss probabilities and their derivatives a few orders of magnitude smaller than required.

In Chapter 5 we describe the C++ source code implementation of the model. This chapter outlines various parallel sections, and stringent error control heuristics used by the source code, as well as efficient implementation provided by the Boost C++ libraries.

Chapter 6 provides the numerical results. Specifically it contains calibration runtimes, comparison of different calibration algorithms and a discussion of model performance on different CDO data sets over the periods before, during and after the 2008-2009 stock market crash.

Chapter 7 describes future research directions which can be undertaken to justify some numerical results obtained in the previous chapter.

Finally, Chapter 8 provides concluding remarks.

Chapter 2

Background

This chapter provides background material needed to understand the Multi-period Single-factor Copula Model (MSCM) proposed by Jackson, Kreinin and Zhang (JKZ) [11]. Section 2.1 starts by explaining the general pricing mechanism of CDOs. Section 2.2 reviews MSCM and Section 2.3 explains how MSCM applies to CDO pricing. MSCM requires a fixed set of input parameters; Section 2.4 explains how to obtain these parameters from CDS spreads. Section 2.5 reviews the original parameterization proposed by JKZ. MSCM relies heavily on numerical integration rules, briefly surveyed in Section 2.6. Calibration of MSCM requires an objective function, which can be based on a variety of error functions, surveyed in Section 2.7. The goal of the calibration procedure is to pick a set of model parameters, which is accomplished by minimizing the objective function. To this end, we review several optimization algorithms in Section 2.8.

2.1 Pricing

Consider pricing a synthetic CDO with a homogeneous pool of K underlying CDSs[1]. The loss given default on each CDS is $L^{\mathrm{GD}} = N \cdot (1 - R)$, where N is the notional value of

[1]The proposed MSCM is only applicable to homogeneous pools.

each CDS and R is the recovery rate (market value as a percent of par value immediately after the default).

We are ultimately interested in pricing exotic CDOs, based on the same underlying pool of CDSs. Some examples include the CDO of CDOs (called CDO^2), options on CDO tranches, etc. All these products require the knowledge of the dynamics (time evolution) of the correlation structure of the pool on which the CDO is based [28]. Once these dynamics are known, the simplest example of CDO pricing is to know what the price of a given tranche should be.

Pricing a tranche refers to computing the spread, which is the ratio of the premiums being paid relative to the tranche size. Once an investor enters a tranche, they are paid a certain amount (quoted as the spread in bps) which depends on the tranche and the tranche size, until the underlyings start defaulting. If the level of defaults is below the attachment point of the investor's tranche, then they receive premiums only. Once the level of defaults rises above the attachment point, the investor starts covering losses, while still receiving premiums on the fraction of the CDSs which their tranche covers that have not yet defaulted. Once all CDSs that an investor's tranche covers default, the investor stops receiving premiums and covering losses; the investor collects the recovered values of underlyings and the contract terminates.

We assume that the premiums are paid quarterly; we denote the premium payment dates by $0 < t_1 < \cdots < t_{n_T-1} < t_{n_T} = T$, where T is the maturity date of the contract. For convenience, we set $t_0 = 0$. Usually, $t_i - t_{i-1} = 1/4$ for all $i \in [1, 2, \cdots, n_T]$ (n_T is the number of quarterly steps until time T), since we measure the time in years. For simplicity, anything which occurs at time t_i is denoted with subscript i. The premium cashflow is termed the premium leg (denoted $P_{n_T}^{(\mathrm{tr})}$, where tr is the tranche index, for now assume $\mathrm{tr} = 1, 2, \cdots, n_{\mathrm{tr}}$) and the default cashflow is termed the default leg (denoted $D_{n_T}^{(\mathrm{tr})}$). In the risk neutral world, assuming no arbitrage, we must have that $E_{(\mathrm{pool})}\left[P_{n_T}^{(\mathrm{tr})}\right] = E_{(\mathrm{pool})}\left[D_{n_T}^{(\mathrm{tr})}\right]$, where the expectations are calculated under the risk neutral pool loss probability measure, denoted by the subscript "(pool)". Our modeling assumption is that a default can only occur at a time t_i, otherwise computation of the premium leg becomes very cumbersome.

Let us denote the attachment and detachment points of the CDO tranche by $a^{(\text{tr})}$ and $b^{(\text{tr})}$ respectively (where $a^{(\text{tr})} < b^{(\text{tr})}$ for all tr) and the size of the tranche by $S^{(\text{tr})} = b^{(\text{tr})} - a^{(\text{tr})}$. We can think of attachment and detachment points in different ways: we can either let the attachment and detachment point be a percentage of the pool size, for example $a^{(\text{tr})} = 3\%$ and $b^{(\text{tr})} = 7\%$ is typical of a Mezzanine tranche, where all tranche sizes $S^{(\text{tr})}$ add up to 100%; or, since the pool of CDSs contains K names, all with the same notional value N, we can also think of them as the fraction of underlyings, for example $a^{(\text{tr})} = 0.03 \cdot K$ and $b^{(\text{tr})} = 0.07 \cdot K$, or perhaps the easiest way is to convert everything into dollar values (because we are working with a homogeneous pool anyway) and set, for example, $a^{(\text{tr})} = 0.03 \cdot K \cdot N$ and $b^{(\text{tr})} = 0.07 \cdot K \cdot N$. We use the first interpretation above for $a^{(\text{tr})}$ and $b^{(\text{tr})}$ (i.e., percentage of the pool size) throughout this thesis.

If we are working in dollar values (which is arguably the most intuitive approach), then the loss taken by a specific tranche tr is

$$L_i^{(\text{tr})} = \min\left(K \cdot N \cdot S^{(\text{tr})}, \max\left(0, L_i^{(\text{pool})} - K \cdot N \cdot a^{(\text{tr})}\right)\right), \tag{2.1}$$

where $L_i^{(\text{pool})} = N \cdot (1 - R) \cdot l_i^{(\text{pool})}$ is the loss of the entire pool of underlyings, $0 \le l_i^{(\text{pool})} \le K$, $l_i^{(\text{pool})} \in \mathcal{Z}^+$ and the size of the tranche $S^{(\text{tr})}$ and the attachment point $a^{(\text{tr})}$ are in terms of the percentage of the pool size convention described above. We can compute the present value of the default and premium legs as

$$D_{n_T}^{(\text{tr})} = \sum_{i=1}^{n_T} \left(L_i^{(\text{tr})} - L_{i-1}^{(\text{tr})}\right) \cdot F_i, \tag{2.2}$$

$$P_{n_T}^{(\text{tr})} = \sum_{i=1}^{n_T} s_{n_T}^{(\text{tr})} \cdot (t_i - t_{i-1}) \cdot \left(K \cdot N \cdot S^{(\text{tr})} - L_i^{(\text{tr})}\right) \cdot F_i, \tag{2.3}$$

where n_T is the number of quarterly time steps until time T and F_i is the discount factor at time t_i:

$$F_i = \exp\left(-\int_{t_0}^{t_i} r(t)dt\right), \tag{2.4}$$

where $r(t)$ is the risk-free interest rate at time t. The equation for $D_{n_T}^{(\mathrm{tr})}$ can be interpreted as the loss in each time period $(t_{i-1}, t_i]$, summed over the time periods and discounted to the present value; the equation for $P_{n_T}^{(\mathrm{tr})}$ can be interpreted as the part of the tranche that has not yet defaulted and so still pays premiums, $K \cdot N \cdot S^{(\mathrm{tr})} - L_i^{(\mathrm{tr})}$, multiplied by the spread $s_{n_T}^{(\mathrm{tr})}$ adjusted by the fraction $(t_i - t_{i-1})$ of a year, and discounted to the present value. If we assume for simplicity that F_i and $L_i^{(\mathrm{tr})}$ are independent random variables and take the expectation under the risk neutral pool loss probability measure, then we obtain

$$E_{(\mathrm{pool})}\left[D_{n_T}^{(\mathrm{tr})}\right] = \sum_{i=1}^{n_T}\left(E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] - E_{(\mathrm{pool})}\left[L_{i-1}^{(\mathrm{tr})}\right]\right) \cdot f_i, \tag{2.5}$$

$$E_{(\mathrm{pool})}\left[P_{n_T}^{(\mathrm{tr})}\right] = s_{n_T}^{(\mathrm{tr})} \cdot \sum_{i=1}^{n_T}\left(K \cdot N \cdot S^{(\mathrm{tr})} - E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i, \tag{2.6}$$

where $f_i = E_{(\mathrm{pool})}[F_i]$ and the spread value $s_{n_T}^{(\mathrm{tr})}$ above is given as a fraction, for example for a 5% spread, $s_{n_T}^{(\mathrm{tr})} = 0.05$. We often assume that the interest rate is a fixed deterministic value. In this case,

$$E_{(\mathrm{pool})}[F_i] = \exp(-r \cdot t_i) = f_i. \tag{2.7}$$

Notice that the computation of the default leg can be rewritten as

$$E_{(\mathrm{pool})}\left[D_{n_T}^{(\mathrm{tr})}\right] = \sum_{i=1}^{n_T-1} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] \cdot (f_i - f_{i+1}) + E_{(\mathrm{pool})}\left[L_{n_T}^{(\mathrm{tr})}\right] \cdot f_{n_T}, \tag{2.8}$$

because $L_0^{(\mathrm{tr})} = 0$ with probability 1. Since, as noted earlier, $E_{(\mathrm{pool})}\left[P_{n_T}^{(\mathrm{tr})}\right] = E_{(\mathrm{pool})}\left[D_{n_T}^{(\mathrm{tr})}\right]$, the spread $s_{n_T}^{(\mathrm{tr})}$ can be estimated by

$$s_{n_T}^{(\mathrm{tr})} = \frac{\sum_{i=1}^{n_T-1} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] \cdot (f_i - f_{i+1}) + E_{(\mathrm{pool})}\left[L_{n_T}^{(\mathrm{tr})}\right] \cdot f_{n_T}}{\sum_{i=1}^{n_T}\left(K \cdot N \cdot S^{(\mathrm{tr})} - E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i}. \tag{2.9}$$

The pricing equation is different for the Equity tranche, which is often referred to as as the 500 bps on-the-run tranche. First of all, the quote for the tranche itself is given usually in percent, and there is a fixed premium of 500 bps. The quote is the amount paid up front (when investor enters the tranche), as a fraction of the quote spread $s_{n_T}^{(1)}$. Hence the pricing equation for the premium leg is

$$E_{(\text{pool})}\left[P_{n_T}^{(1)}\right] = s_{n_T}^{(1)} \cdot K \cdot N \cdot S^{(1)} + 0.05 \sum_{i=1}^{n_T} \left(K \cdot N \cdot S^{(1)} - E_{(\text{pool})}\left[L_i^{(1)}\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i. \tag{2.10}$$

Hence, using $E_{(\text{pool})}\left[P_{n_T}^{(\text{tr})}\right] = E_{(\text{pool})}\left[D_{n_T}^{(\text{tr})}\right]$ again, we obtain

$$\begin{aligned} s_{n_T}^{(1)} = &\left(\sum_{i=1}^{n_T-1} E_{(\text{pool})}\left[L_i^{(1)}\right] \cdot (f_i - f_{i+1}) + E_{(\text{pool})}\left[L_{n_T}^{(1)}\right] \cdot f_{n_T}\right. \\ &\left. -0.05 \cdot \sum_{i=1}^{n_T} \left(K \cdot N \cdot S^{(1)} - E_{(\text{pool})}\left[L_i^{(1)}\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i \right) / K \cdot N \cdot S^{(1)}. \end{aligned} \tag{2.11}$$

Hence, the problem of estimating the spread is reduced to the problem of estimating $E_{(\text{pool})}\left[L_i^{(\text{tr})}\right]$.

Also, notice that we can rewrite the pricing equations using a different convention. Let

$$l_i^{(\text{tr})} = \min\left(\frac{K \cdot S^{(\text{tr})}}{1-R}, \max\left(0, l_i^{(\text{pool})} - \frac{K \cdot a^{(\text{tr})}}{1-R}\right)\right), \tag{2.12}$$

where we have previously defined $L_i^{(\text{pool})} = N \cdot (1-R) \cdot l_i^{(\text{pool})}$, and the superscript "(pool)" denotes the risk neutral pool loss probability dependence. Then

$$s_{n_T}^{(\text{tr})} = \frac{\sum_{i=1}^{n_T-1} E_{(\text{pool})}\left[l_i^{(\text{tr})}\right] \cdot (f_i - f_{i+1}) + E_{(\text{pool})}\left[l_{n_T}^{(\text{tr})}\right] \cdot f_{n_T}}{\sum_{i=1}^{n_T} \left(\frac{K \cdot S^{(\text{tr})}}{1-R} - E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i}, \tag{2.13}$$

$$\begin{aligned} s_{n_T}^{(1)} = &\left(\sum_{i=1}^{n_T-1} E_{(\text{pool})}\left[l_i^{(1)}\right] \cdot (f_i - f_{i+1}) + E_{(\text{pool})}\left[l_{n_T}^{(1)}\right] \cdot f_{n_T} - \right. \\ &\left. -0.05 \cdot \sum_{i=1}^{n_T} \left(\frac{K \cdot S^{(1)}}{1-R} - E_{(\text{pool})}\left[l_i^{(1)}\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i \right) \Big/ \frac{K \cdot S^{(1)}}{1-R}. \end{aligned} \tag{2.14}$$

Note that, for the Super Senior tranche, we have to adjust the detachment point to K, and not $K/(1-R)$, because we cannot have more than K underlyings default.

To compute expectation, we can start with (2.1) and using $L_i^{(\text{pool})} = N \cdot (1-R) \cdot l_i^{(\text{pool})}$, where $l_i^{(\text{pool})} = 0, 1, \cdots K$, factor out the term $N \cdot (1-R)$ to obtain

$$L_i^{(\text{tr})} = N \cdot (1-R) \cdot \min\left(\frac{K \cdot S^{(\text{tr})}}{1-R}, \max\left(0, l_i^{(\text{pool})} - \frac{K \cdot a^{(\text{tr})}}{1-R}\right)\right). \tag{2.15}$$

The above equation relates the number of defaults to the loss of the specific tranche. Hence we can weight the tranche loss by the probability of r defaults in the pool to compute the expected value

$$E_{(\text{pool})}\left[L_i^{(\text{tr})}\right] = N \cdot (1-R) \cdot \sum_{r=1}^{K} \min\left(\frac{K \cdot S^{(\text{tr})}}{1-R}, \max\left(0, r - \frac{K \cdot a^{(\text{tr})}}{1-R}\right)\right) \cdot P\left(l_i^{(\text{pool})} = r\right), \tag{2.16}$$

where $a^{(\text{tr})}$ and $S^{(\text{tr})}$ are given in terms of the percentage of the pool size convention described above[2].

Therefore, the problem is reduced to estimating $P\left(l_i^{(\text{pool})} = r\right)$.

2.2 Multi-Period Single-Factor Copula Model

For a homogeneous pool, let

$$\alpha_i = P(\tau_k \in (t_{i-1}, t_i] | \tau_k > t_{i-1}) = \frac{P(\tau_k \le t_i) - P(\tau_k \le t_{i-1})}{1 - P(\tau_k \le t_{i-1})} \tag{2.17}$$

be the probability that the k-th underlying entity defaults in the time interval $(t_{i-1}, t_i]$, conditional on no earlier default (because a certain entity can default once only). The random variable τ_k us the default time of the k-th entity. Further, let us introduce random variables

$$U_{k,i} = \beta_{k,i} X_i + \eta_{k,i} \epsilon_{k,i}, \tag{2.18}$$

[2]When $r = 0$, the pool loss is zero and so the corresponding term is omitted from the sum for the computation of the expected value.

where $\beta_{k,i}$ is the copula tranche implied correlation parameter[3], $X_i \sim N(0,1)$ are independent and identically distributed (iid), $\epsilon_{k,i} \sim N(0,1)$ are iid, X_i is independent of $\epsilon_{k,i}$, for $k = 1, 2, \cdots, K$ and $i = 0, 1, \cdots, n_T$, and $N(0,1)$ denotes the standard normal probability density; parameter $\eta_{k,i}$ is determined to be $\eta_{k,i} = \sqrt{1 - \beta_{k,i}^2}$ in the next paragraph. We partition the time in years into quarterly payments, so for 5 years, $i = 0, 1, \cdots, 20$, where $i = 0$ is included for completeness in the base cases later on. X_i is the common factor driving the change in $U_{k,i}$; X_i affects all underlyings at time t_i (for example, some economic shock). The factor $\epsilon_{k,i}$ is associated with the variability of individual names. For a homogeneous pool, we have

$$U_{k,i} = \beta_i X_i + \eta_i \epsilon_{k,i}. \tag{2.19}$$

For ease in implementing the Gaussian copula model, we want $U_{k,i} \sim N(0,1)$, and since the Gaussian density is characterized by its first two moments, we want $E\left[U_{k,i}\right] = 0$ (satisfied automatically) and $\text{Var}\left[U_{k,i}\right] = E\left[U_{k,i}^2\right] - E\left[U_{k,i}\right]^2 = 1$. Solving the last equation for $\eta_{k,i}$, we obtain $\eta_{k,i} = \sqrt{1 - \beta_{k,i}^2}$. Since $\text{Var}\left[U_{k,i}\right] = 1$ for all $k = 1, 2, \cdots, K$, this also implies that for two names $k_1 \neq k_2$, we have $\text{Corr}(U_{k_1,i}, U_{k_2,i}) = E\left[\left(U_{k_1,i}\right)\left(U_{k_2,i}\right)\right] = \beta_{k_1,i}\beta_{k_2,i}$. For a homogeneous pool, the MSCM tranche implied correlation between all underlyings at time t_i is β_i^2. Each $\beta_{k,i}$ can be thought of as the copula correlation factor[4] for name k with respect to all other names in the pool.

Since $U_{k,i} \sim N(0,1)$, we can use the standard percentile to percentile transformation (similar to [1]):

$$P(U_{k,i} < u_{k,i}) = \Phi(u_{k,i}) = P(\tau_k \in (t_{i-1}, t_i] | \tau_k > t_{i-1}) = \frac{P(\tau_k \leq t_i) - P(\tau_k \leq t_{i-1})}{1 - P(\tau_k \leq t_{i-1})}, \tag{2.20}$$

for $u_{k,i} \in \mathcal{R}$, where $\Phi(\cdot)$ is the standard normal Cumulative Distribution Function (CDF).

---

[3] We must also note that there are other measures of tranche implied correlations available, which do no necessarily require expensive computation schemes. This thesis focuses on extending single-period single-factor copula correlation, but there are also compound [37] and base [38] correlations available. In addition, one can also emphasize pricing bespoke CDOs [36].

[4] The copula correlation factor represents the true correlation.

Therefore,

$$u_{k,i} = \Phi^{-1}\left(P(\tau_k \in (t_{i-1}, t_i] | \tau_k > t_{i-1})\right), \tag{2.21}$$

where Φ^{-1} denotes the inverse of the standard normal CDF. Under this Gaussian copula model, a default happens when

$$\Phi(U_{k,i}) < P\left(\tau_k \in (t_{i-1}, t_i] | \tau_k > t_{i-1}\right), \tag{2.22}$$

or equivalently, if we condition on the value of the common factor $X_i = x_i$, when

$$U_{k,i} = \beta_{k,i} x_i + \epsilon_{k,i}\sqrt{1-\beta_{k,i}^2} < \Phi^{-1}\left[\frac{P(\tau_k \le t_i) - P(\tau_k \le t_{i-1})}{1 - P(\tau_k \le t_{i-1})}\right]. \tag{2.23}$$

We can rearrange this inequality to obtain

$$\epsilon_{k,i} < \frac{\Phi^{-1}\left[\frac{P(\tau_k \le t_i) - P(\tau_k \le t_{i-1})}{1 - P(\tau_k \le t_{i-1})}\right] - \beta_{k,i} x_i}{\sqrt{1-\beta_{k,i}^2}}. \tag{2.24}$$

Since we know the probability density for $\epsilon_{k,i}$, we can determine that the conditional default probability is

$$p_{k,i}(x_i) = P(\tau_k \in (t_{i-1}, t_i] | \tau_k > t_{i-1}, X_i = x_i) = \Phi\left[\frac{\Phi^{-1}\left[\frac{P(\tau_k \le t_i) - P(\tau_k \le t_{i-1})}{1 - P(\tau_k \le t_{i-1})}\right] - \beta_{k,i} x_i}{\sqrt{1-\beta_{k,i}^2}}\right]. \tag{2.25}$$

Notice that on the time interval $(t_0, t_1]$ (first time interval), this multi-period single-factor copula model reduces to a single-period single-factor copula model provided in [1]. Namely, we obtain

$$P(\tau_k < t_1 | X_1 = x_1) = \Phi\left[\frac{\Phi^{-1}\left[P(\tau_k \le t_1)\right] - \beta_{k,i} x_i}{\sqrt{1-\beta_{k,i}^2}}\right], \tag{2.26}$$

since $P(\tau_k \le t_0) = 0$. Also, we can replace Φ by some other density, with other parameters of interest, such as, for example, the Normal Inverse Gaussian distribution with two fixed and two variable parameters [29]. This yields a different copula model, but with an added set of parameters which makes the model less parsimonious. In this thesis, we restrict our work to the standard normal density.

2.3 Pricing With The Multi-Period Single-Factor Copula Model

Fingers [33] was the first to extend the single-period copula model to a multi-period copula model. Soon after Andersen [34] and Sidenius [35] proposed alternative multi-period factor copula models. However, these schemes suffer a computational drawback: to calibrate parameters of the stochastic process (possible paths and associated risk neutral probability values) we need to perform Monte Carlo (MC) simulation, which makes the calibration extremely expensive. Hence their multi-period factor copula models are not practical for more than a few common factors. Jackson, Kreinin and Zhang [11] proposed another model which avoids MC simulation for homogeneous pools. This section provides an overview of their approach.

For a homogeneous pool,

$$L_i^{(\text{pool})} = N \cdot (1-R) \sum_{k=1}^{K} \mathcal{I}(\tau_k \leq t_i) = N \cdot (1-R) \cdot l_i^{(\text{pool})}, \tag{2.27}$$

where $\mathcal{I}$ is the indicator function, whence

$$P\left(L_i^{(\text{pool})} = N \cdot (1-R) \cdot r\right) = P\left(l_i^{(\text{pool})} = r\right). \tag{2.28}$$

Then [8] derives the following recursive relationship (please refer to Appendix A.1 for proof):

$$\begin{aligned} P\left(l_i^{(\text{pool})} = r\right) &= \sum_{m=0}^{r} P\left(l_{i-1}^{(\text{pool})} = m\right) \cdot P\left(l_{(i-1,i]}^{(\text{pool}),K-m} = r-m\right) \\ &= \sum_{m=0}^{r} \left[P\left(l_{i-1}^{(\text{pool})} = m\right) \cdot \int_{-\infty}^{\infty} P\left(l_{(i-1,i]}^{(\text{pool}),K-m} = r-m | X_i = x_i\right) d\Phi(x_i) \right], \end{aligned} \tag{2.29}$$

where $l_{(i-1,i]}^{(\text{pool}),K-r}$ denotes the number of defaults for a pool of size $K-m$ during the time interval $(t_{i-1}, t_i]$. For a homogeneous pool, $p_{k,i}(x) = p_i(x)$ for all $k = 1, 2, \cdots K$, $x \in \mathcal{R}$, whence

$$
\begin{aligned}
P\left(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m | X_i = x_i\right) &= \begin{pmatrix} K-m \\ r-m \end{pmatrix} p_i(x_i)^{r-m}(1-p_i(x_i))^{K-r} \\
&= \text{Bin}(r-m; K-m, p_i(x_i)), \qquad (2.30)
\end{aligned}
$$

where $\text{Bin}(k; n, p)$ denotes the Binomial probability of k out of n events occurring with individual success probability p. Notice that (2.29) is just matrix multiplication with a lower triangular matrix which has $P\left(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m\right)$ as the value in its r-th row and m-th column. For the base cases, $P\left(l_0^{(\text{pool})} = 0\right) = 1$ and $P\left(l_0^{(\text{pool})} = m\right) = 0$ for all $m = 1, 2, \cdots, K$.

2.4 Bootstrapping Default Probabilities From CDS Spreads

Default probabilities α_i defined in (2.17) are fixed input parameters into the multi-period multi-factor copula model. This section explains how to calculate α_i from the CDS spreads $s_i^{(\text{CDS})}$[5].

There are several different approaches to bootstrapping default probabilities of the underlying entities from CDS spreads. Computing times-to-default can be accomplished with dynamic hazard rates as in Section 6 of [4], with constant hazard rates between CDS maturities in different, yet, similar approaches presented as Solutions 1 & 2 in [5] and as described in [3]; arguably, the most intuitive and simplest approach is described as Solution 3 in [5] and on pages 18-19 of [2].

Let $p_i^{(\text{CDS})} = P(\tau \leq t_i)$ be the default probability that we wish to bootstrap from the CDS quotes. Then the pricing equations for a CDS with maturity at time T are given by

$$E\left[D_{n_T}^{(\text{CDS})}\right] = (1-R) \cdot N \cdot \sum_{i=1}^{n_T} (p_i^{(\text{CDS})} - p_{i-1}^{(\text{CDS})}) \cdot f_i; \qquad (2.31)$$

[5]This is typically referred to as "bootstrapping"in the finance literature

$$E\left[P_{n_T}^{(\mathrm{CDS})}\right] = N \cdot s_{n_T}^{(\mathrm{CDS})} \cdot \sum_{i=1}^{n_T} (t_i - t_{i-1}) \cdot (1 - p_i^{(\mathrm{CDS})}) \cdot f_i, \tag{2.32}$$

where the expectation is taken with respect to the risk neutral probability measure. We can understand the default leg as the probability of default in time interval $(t_{i-1}, t_i]$ multiplied by the loss given default $(1-R) \cdot N$ and discounted back to the present value with f_i. Hence at each time t_i we are computing the expected loss given default. The premium payment is the spread $s_{n_T}^{(\mathrm{CDS})}$ times the notional N multiplied by the fraction of the year $(t_i - t_{i-1})$ associated with this payment, times the probability of the entity not defaulting by time t_i, again discounted back to the present value with f_i.

To bootstrap the default probability, we obtain the spread $s_1^{(\mathrm{CDS})}$ for a CDS that matures at time t_1 and solve for $p_1^{(\mathrm{CDS})}$. We then obtain the spread $s_2^{(\mathrm{CDS})}$ for a CDS that matures at time t_2 and solve for $p_2^{(\mathrm{CDS})}$ using $p_1^{(\mathrm{CDS})}$ and repeat this procedure recursively. As a technical note, we perform linear interpolation of CDS quotes[6]. The spread is

$$s_{n_T}^{(\mathrm{CDS})} = \frac{(1-R) \cdot \sum_{i=1}^{n_T} \left(p_i^{(\mathrm{CDS})} - p_{i-1}^{(\mathrm{CDS})}\right) \cdot f_i}{\frac{1}{4} \cdot \sum_{i=1}^{n_T} \left(1 - p_i^{(\mathrm{CDS})}\right) \cdot f_i}, \tag{2.33}$$

where we have used $(t_i - t_{i-1}) = 1/4$. Using $p_0^{(\mathrm{CDS})} = 0$, we can solve to obtain

$$p_1^{(\mathrm{CDS})} = P(\tau \leq t_1) = \frac{\frac{1}{4} \cdot s_1^{(\mathrm{CDS})}}{(1-R) + \frac{1}{4} \cdot s_1^{(\mathrm{CDS})}}. \tag{2.34}$$

This is our base case. We can solve for the other $p_i^{(\mathrm{CDS})}$ for $i = 2, 3, \cdots, n_T$ using the recursive bootstrapping formula:

$$\begin{aligned} p_i^{(\mathrm{CDS})} = &\left(\frac{1}{4} s_i^{(\mathrm{CDS})} \cdot \left(f_i + \sum_{j=1}^{i-1} f_j \cdot (1 - p_j^{(\mathrm{CDS})})\right) + \right. \\ &\left.(1-R) \cdot \left(f_i \cdot p_{i-1}^{(\mathrm{CDS})} - \sum_{j=1}^{i-1} (p_j^{(\mathrm{CDS})} - p_{j-1}^{(\mathrm{CDS})}) \cdot f_j\right)\right) \Big/ \left((1-R) \cdot f_i + \frac{1}{4} \cdot s_i^{(\mathrm{CDS})} \cdot f_i\right). \end{aligned} \tag{2.35}$$

[6] Standard industry practice is to use a linear interpolant [2].

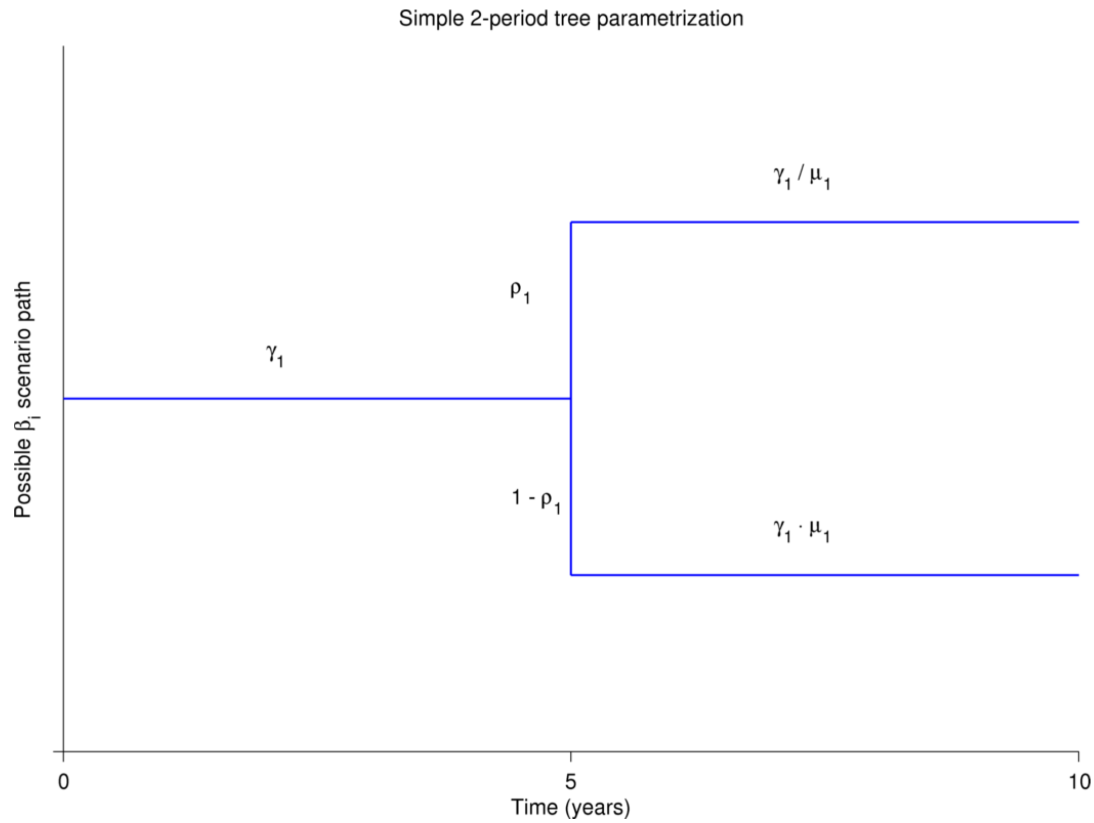

Figure 2.1: Simple 2-period tree parameterization with 3 parameters: $\vec{\psi} = (\gamma_1, \mu_1, \rho_1)$, used originally in [11].

2.5 Original Model Parameterization

The Single-period Single-factor Copula Model (SSCM) produces an approximation $s_{n_T}^{(\mathrm{tr})}$ to the CDO market spread $m_{n_T}^{(\mathrm{tr})}$ using the ratio

$$s_{n_T}^{(\mathrm{tr})} = \frac{E_{(\mathrm{pool})}\left[D_{n_T}^{(\mathrm{tr})}\right]}{E_{(\mathrm{pool})}\left[P_{n_T}^{(\mathrm{tr})}\right]}, \tag{2.36}$$

where the default and premium leg expectations were previously given by (2.5) and (2.6), respectively, and expectations with respect to the risk neutral pool loss probability are computed using SSCM, for example [1]. In this section, we describe how MSCM (proposed by JKZ [11]) computes the approximation to the CDO market spread.

In their example in [11], JKZ model the dynamics of the market using a tree structure, depicted in Figure A.2. To illustrate the approach, we consider a simpler tree parameterization in Figure 2.1, where the copula correlation parameter β_i, introduced in Section 2.2, follows a specific path (referred to as the *scenario*) in time with a specific probability: scenario values are parameterized using γ_1 and μ_1 and probabilities are parameterized using ρ_1. There are two possible scenarios for the β_i's in Figure 2.1:

- $\beta_i = \gamma_1$ for all $i = 1, 2, \cdots, 20$ and $\beta_i = \gamma_1/\mu_1$ for all $i = 21, 22, \cdots, 40$ with probability ρ_1;

- $\beta_i = \gamma_1$ for all $i = 1, 2, \cdots, 20$ and $\beta_i = \gamma_1 \cdot \mu_1$ for all $i = 21, 22, \cdots, 40$ with probability $1 - \rho_1$.

The full set of variable model parameters in this example is $\vec{\psi} = (\gamma_1, \mu_1, \rho_1)$, over which the model calibration is performed. The fixed set of model parameters are the default probabilities α_i (2.17), which are bootstrapped from CDS market spreads, as explained in Section 2.4. Figure A.2 depicts a more general tree parameterization with more periods, but the idea is the same: each new period scenario branches from the previous scenario using a different factor μ_j, responsible for each period j, with a new probability ρ_j, also responsible for each period.

Recall that $\beta_i \in [0, 1]$ in Section 2.2, and hence $\psi_j \in [0, 1]$ for $j = 1, 2, 3$. However, one drawback to this tree parameterization is that if μ_1 is close to zero, then $\beta_i > 1$, for $i = 21, 22, \cdots, 40$ and a separate set of constraints have to be added into the calibration routine to overcome this. Other difficulties with this parameterization are summarized later in Section 3.1. In the same section, we propose an alternative multi-path parameterization, which overcomes these difficulties. This multi-path parameterization consists of β_i scenario-setting values $\vec{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_{n_\gamma})$ and path probabilities $\vec{\rho} = (\rho_1, \rho_2, \cdots, \rho_{n_\rho})$. We describe certain restrictions which must be placed on $\vec{\rho}$ in Section 3.1. The complete multi-path parameter vector $\vec{\psi}$ is partitioned into $\vec{\psi} = (\vec{\gamma}, \vec{\psi}_{n_\gamma+1:n_\psi})$, where $\vec{\psi}$ has $n_\psi = 2n_\gamma = 2n_\rho$ elements and $\vec{\psi}_{n_\gamma+1:n_\psi} = (\psi_{n_\gamma+1}, \psi_{n_\gamma+2}, \cdots, \psi_{n_\psi})$. Probabilities $\vec{\rho}$ are set using $\vec{\psi}_{n_\gamma+1:n_\psi}$ in a trivial manner, described in Section 3.1. For simpler multi-path parameterizations, $\vec{\rho} = \vec{\psi}_{n_\gamma+1:n_\psi}$ and the complete set of model parameters becomes $\vec{\psi} = (\vec{\gamma}, \vec{\rho})$.

Let $\vec{\beta} = (\beta_1, \beta_2, \cdots, \beta_{n_T})$ and $\vec{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_{n_T})$. In MSCM, $\vec{\beta}$ is a discrete random vector, with scenario probabilities specified by $\vec{\rho}$; in SSCM, $\vec{\beta} = (\beta,\beta, \cdots, \beta)$ for the copula correlation parameter β, with probability 1. In both SSCM and MSCM, the CDO spread is a function of $\vec{\beta}$ and $\vec{\alpha}$, i.e. $s_{n_T}^{(\mathrm{tr})} = s_{n_T}^{(\mathrm{tr})}(\vec{\beta}, \vec{\alpha})$ [7]. Hence in MSCM, the spread $s_{n_T}^{(\mathrm{tr})}$ is a random variable through $\vec{\beta}$, and MSCM approximates the CDO market spread by computing the expectation

$$e_{n_T}^{(\mathrm{tr})}(\vec{\psi}) = E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta}, \vec{\alpha})|\vec{\gamma}\right], \tag{2.37}$$

where the fixed set of parameters $\vec{\alpha}$ is included for completeness. From a functional point of view, the expected spread is a function $e_{n_T}^{(\mathrm{tr})}(\vec{\psi})$ of the model parameters $\vec{\psi}$. In our model, (2.37) reduces to

$$e_{n_T}^{(\mathrm{tr})}(\vec{\psi}) = \sum_{\substack{\text{all scenarios } \vec{\zeta} \\ \text{of } \vec{\beta}}} s_{n_T}^{(\mathrm{tr})}(\vec{\zeta}, \vec{\alpha}) \cdot P_{\vec{\rho}}\left(\vec{\beta} = \vec{\zeta}|\vec{\Gamma} = \vec{\gamma}\right), \tag{2.38}$$

where $\vec{\zeta}$ specifies a specific scenario value of $\vec{\beta}$, and the conditional probability is specified by $\vec{\rho}$. For the scenarios depicted in Figure 2.1, $P_{\vec{\rho}}\left(\vec{\beta} = \vec{\zeta}|\vec{\Gamma} = \vec{\gamma}\right)$ takes on values ρ_1 and $1 - \rho_1$; we can think of other realizations of β_i as occurring with probability zero. Thus, for the scenarios in Figure 2.1, (2.38) reduces to

$$\begin{aligned} e_{n_T}^{(\mathrm{tr})}(\vec{\psi}) = s_{n_T}^{(\mathrm{tr})}&\left((\gamma_1, \gamma_1, \cdots, \gamma_1, \gamma_1/\mu_1, \gamma_1/\mu_1, \cdots, \gamma_1/\mu_1), \vec{\alpha}\right) \cdot \rho_1 + \\ s_{n_T}^{(\mathrm{tr})}&\left((\gamma_1, \gamma_1, \cdots, \gamma_1, \gamma_1 \cdot \mu_1, \gamma_1 \cdot \mu_1, \cdots, \gamma_1 \cdot \mu_1), \vec{\alpha}\right) \cdot (1 - \rho_1). \end{aligned} \tag{2.39}$$

Now, notice that to compute $s_i^{(\mathrm{tr})}$ efficiently, we have to store previous values of $P\left(l_i^{(\mathrm{pool})} = r\right)$ for time t_i. Moreover, these probabilities depend on the values of β_i,

[7]The only difference is that the risk neutral pool loss probabilities in (2.36) are modeled slightly differently in SSCM and MSCM. For a description of SSCM, see [1]. MSCM's risk neutral pool loss probabilities were described in Section 2.3.

which follow a particular scenario. So, we can keep track of all possible values that $P\left(l_i^{(\text{pool})} = r\right)$ can take for different β_i's. The expectations $E_{(\text{pool})}\left[L_i^{(\text{tr})}\right]$ can also be reused, but more copies have to be stored in memory due to the dependence on the tranche. Chapter 5 explains such implementation details more completely.

2.6 Quadrature Methods

Due to the nature of the problem and to make the integration as efficient as possible, we consider Gaussian quadrature formulas on a finite interval $[a, b]$ as possible approaches. An overview of these methods is given in Chapters 2 and 4 of [6]; for a more detailed discussion, please see [7].

The goal is to compute the lower triangular probability matrix A_i (see (2.43) below) with entries $P\left(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m\right)$. Let us denote the standard normal probability density by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right) \tag{2.40}$$

and the rest of the integrand in (2.29) by

$$h(x) = \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)^{r-m} \left(1 - \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)\right)^{K-r} \tag{2.41}$$

with the constant of integration

$$c = \begin{pmatrix} K-m \\ r-m \end{pmatrix}. \tag{2.42}$$

For each time t_i, scenario for β_i and entry given by r and m, we need to compute

$$\begin{aligned} [A_i]_{r,m} &= P\left(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m\right) \\ &= c \int_{-\infty}^{\infty} h(x)\phi(x)dx. \end{aligned} \tag{2.43}$$

Gauss-Chebyshev and Gauss-Legendre n-point quadrature rules are of the form

$$\int_{-1}^{1} W(x)\chi(x)dx \approx \sum_{j=1}^{n} w_j \chi(x_j), \tag{2.44}$$

where $W(x)$ is the weight function associated with the rule, $\chi(x)$ is the function which we would like to integrate[8], w_j are the quadrature weights and x_j are the quadrature nodes. The weight function $W(x)$ is $W(x) = \sqrt{1-x^2}$ or $W(x) = 1/\sqrt{1-x^2}$ for the Gauss-Chebyshev quadrature rule, and simply $W(x) = 1$ for the Gauss-Legendre quadrature rule. Gauss-Hermite quadrature rules are of the form

$$\int_{-\infty}^{\infty} \exp(-x^2)\chi(x)dx \approx \sum_{j=1}^{n} w_j \chi(x_j), \tag{2.45}$$

where the weight function is $W(x) = \exp(-x^2)$.

A change of variables can be made to change the interval of integration in (2.44) from $[-1, 1]$ to $[a, b]$ using

$$\begin{aligned} \int_a^b w(x)\chi(x)dx &= \frac{b-a}{2}\int_{-1}^{1} W(x)\chi\left(\frac{b-a}{2}x + \frac{a+b}{2}\right)dx \\ &\approx \frac{b-a}{2}\sum_{j=1}^{n} w_j \chi\left(\frac{b-a}{2}x_j + \frac{a+b}{2}\right), \end{aligned} \tag{2.46}$$

where we have assumed $w\left(\frac{b-a}{2}x + \frac{a+b}{2}\right) = W(x)$ from (2.44). These three quadrature rules are discussed in the following subsections. The error formulas are not discussed, because we provide an alternative strategy for determining the interval of integration $[a, b]$ and the number of quadrature nodes n in Chapter 4. For a full discussion of why we are only considering these methods and how they apply, please also see Chapter 4.

2.6.1 Gauss-Chebyshev

We can use both variants of the Gauss-Chebyshev formula:

$$\int_{-1}^{1} \frac{\chi(x)}{\sqrt{1-x^2}}dx \qquad \text{and} \qquad \int_{-1}^{1} \chi(x)\sqrt{1-x^2}dx, \tag{2.47}$$

where the nodes are respectively given by

$$x_j = \cos\left(\frac{2j-1}{2n}\pi\right) \qquad \text{and} \qquad x_j = \cos\left(\frac{j}{n+1}\pi\right) \tag{2.48}$$

[8] In our case, $\chi(x) = c \cdot h(x) \cdot \phi(x)$.

and the weights are given by

$$w_j = \frac{\pi}{n} \qquad \text{and} \qquad w_j = \frac{\pi}{n+1} \sin^2\left(\frac{j}{n+1}\pi\right). \tag{2.49}$$

2.6.2 Gauss-Legendre

This is often the simplest rule to use, as the weight function is $W(x) = 1$. The j-th node x_j is the j-th root of the Legendre polynomial $P_n(x)$, where $P_n(x)$ is normalized to give $P_n(1) = 1$. The weights are

$$w_j = \frac{2}{(1 - x_j^2)\left[P_n'(x_j)\right]^2}. \tag{2.50}$$

2.6.3 Gauss-Hermite

This is possibly the most intuitive method to use for problem (2.43), since $W(x) = \exp(-x^2)$, and the interval of integration is $(-\infty, \infty)$. The j-th node x_j is the j-th root of the Hermite polynomial $H_n(x)$ and the weights are

$$w_j = \frac{2^{n-1} n! \sqrt{\pi}}{n^2\left[H_{n-1}(x_j)\right]^2}. \tag{2.51}$$

2.7 Error Functions

The purpose of calibration is to fit the model parameters to the CDO market quotes $m_{n_T}^{(\mathrm{tr})}$, according to some error criterion. More specifically, for the MSCM described in Section 2.5, our goal is to fit a set of model parameter values $\vec{\psi} = (\vec{\gamma}, \vec{\rho})$ to the market quotes $m_{n_T}^{(\mathrm{tr})}$ by minimizing the objective function

$$f(\vec{\psi}) = \sum_{\mathrm{tr} \in \mathrm{Tr}} \sum_{T \in M} \operatorname{error}\left(E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta}, \vec{\alpha}) | \vec{\gamma}\right], m_{n_T}^{(\mathrm{tr})}\right), \tag{2.52}$$

for some error function defined in this section, where Tr is the set of tranches, and M is the set of maturities. For notational convenience, we can re-write (2.52) using a double-index $k = (\mathrm{tr}, T)$ as

$$f(\vec{\psi}) = \sum_{k \in \{(\mathrm{tr},T)|\mathrm{tr}\in\mathrm{Tr},T\in M\}} \mathrm{error}\left(E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right], m_{n_T}^{(\mathrm{tr})}\right), \tag{2.53}$$

where there are $|\mathrm{Tr}| \cdot |M|$ terms in the sum (2.53), and $|\mathrm{Tr}|$ and $|M|$ are the number of terms in the sets Tr and M, respectively. Using this double-index notation, we also abbreviate, for notational convenience in this section only, $\mathcal{E}_k = E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right]$, $m_k = m_{n_T}^{(\mathrm{tr})}$, and let

$$f_k(\vec{\psi}) = \mathrm{error}\left(\mathcal{E}_k, m_k\right). \tag{2.54}$$

Hence we can also write (2.53) as

$$f(\vec{\psi}) = \sum_k f_k(\vec{\psi}). \tag{2.55}$$

For an efficient implementation of the calibration procedure, the error function has to be cheap to compute, has to be convex to ensure the uniqueness of the solution (at least in simple cases) and has to be resilient to outliers [19]. Moreover, it is preferable for the error function to have a continuous first derivative. The least squares error function

$$\mathrm{error}_{\mathrm{LS}}\left(\mathcal{E}_k, m_k\right) = \left(\mathcal{E}_k - m_k\right)^2 \tag{2.56}$$

satisfies three of these four conditions, but it is not resilient to outliers.

The linear ϵ-insensitive error function [18]

$$\mathrm{error}_{\epsilon}\left(\mathcal{E}_k, m_k\right) = \max(|\mathcal{E}_k - m_k| - \epsilon, 0),\ \ \epsilon \geq 0, \tag{2.57}$$

is resilient to outliers, but it has a discontinuous first derivative. The Soft Error Function (SEF), described in [17] as a Soft Loss Function (SLF), is smooth and is as resilient to outliers as (2.57):

$$\mathrm{error}_{\epsilon,\delta}\left(\mathcal{E}_k, m_k\right) = \begin{cases} -(\mathcal{E}_k - m_k) - \epsilon, & \text{if } \mathcal{E}_k - m_k < -(1+\delta)\epsilon; \\ \frac{(\mathcal{E}_k - m_k + (1-\delta)\epsilon)^2}{4\delta\epsilon}, & \text{if } -(1+\delta)\epsilon \le \mathcal{E}_k - m_k \le -(1-\delta)\epsilon; \\ 0, & \text{if } -(1-\delta)\epsilon < \mathcal{E}_k - m_k < (1-\delta)\epsilon; \\ \frac{(\mathcal{E}_k - m_k - (1-\delta)\epsilon)^2}{4\delta\epsilon}, & \text{if } (1-\delta)\epsilon \le \mathcal{E}_k - m_k \le (1+\delta)\epsilon; \\ \mathcal{E}_k - m_k - \epsilon, & \text{if } (1+\delta)\epsilon < \mathcal{E}_k - m_k, \end{cases} \tag{2.58}$$

where $0 < \delta \le 1$ and $\epsilon > 0$. See Figure A.1 for a comparison plot of these three error functions. The first derivative of SLF is given as (9) in [17].

Recall that in Sec 2.5, the CDO tranches are ranked in order of seniority, with the more senior, less risky tranches receiving smaller premium payments than the less senior, more risky tranches. The premium payments are quoted as CDO market spreads m_k. Hence, we need to match m_k in a relative error sense with MSCM approximation $\mathcal{E}_k$, i.e. we need to match the most significant digits in each CDO market quote m_k. For example, if m_k is small in magnitude, then, if the MSCM approximation $\mathcal{E}_k$ does not match m_k precisely, the absolute error using any of the three error functions (2.56), (2.57) and (2.58) will be small, but the relative error may be large. This behavior will result in poor fits to the more senior, less risky tranches.

Relative error is computed by rescaling the absolute error by m_k. For (2.56), we obtain

$$\mathrm{error}^{\mathrm{rel}}_{\mathrm{LS}}\left(\mathcal{E}_k, m_k\right) = \left(\frac{\mathcal{E}_k - m_k}{m_k}\right)^2 = \mathrm{error}_{\mathrm{LS}}\left(\frac{\mathcal{E}_k}{m_k}, 1\right). \tag{2.59}$$

Using the same change of variables for (2.57) and (2.58), produces

$$\mathrm{error}^{\mathrm{rel}}_{\epsilon}\left(\mathcal{E}_k, m_k\right) = \mathrm{error}_{\epsilon}\left(\frac{\mathcal{E}_k}{m_k}, 1\right), \tag{2.60}$$

$$\mathrm{error}^{\mathrm{rel}}_{\epsilon,\delta}\left(\mathcal{E}_k, m_k\right) = \mathrm{error}_{\epsilon,\delta}\left(\frac{\mathcal{E}_k}{m_k}, 1\right). \tag{2.61}$$

The derivatives of relative error functions with respect to $\mathcal{E}_k$ are computed with a single application of the chain rule, to yield a multiplicative factor of $1/m_k$[9].

Notice that parameter ϵ in (2.61) controls the precision with which $\mathcal{E}_k$ matches m_k. For example, if we want the quotes to match to 3 significant digits, then an appropriate value for ϵ is $\epsilon = 9 \cdot 10^{-4}$. For model results used in later sections, we simply set $\epsilon = 10^{-4}$.

2.8 Optimization Algorithms

In our context, the goal of an optimization algorithm is to minimize the objective function (2.52) by changing the set of model parameters $\vec{\psi}$, introduced in Section 2.5. For the models that we introduce in Chapter 3, the parameters $\vec{\psi} = (\psi_1, \psi_2, \cdots, \psi_{n_\psi})$ must satisfy $\psi_j \in [0, 1]$ for $i = 1, 2, \cdots, n_\psi$. We can turn the associated constrained optimization problem for $f(\vec{\psi})$ into an unconstrained optimization problem by introducing the change of variables

$$\psi_j = \mathcal{L}(u_j) = \frac{1}{1 + \exp(-u_j)}. \tag{2.62}$$

Note that for all $u_j \in \mathcal{R}$, $\psi_j = \mathcal{L}(u_j) \in [0, 1]$ for all $j = 1, 2, \cdots, n_\psi$. Hence, to calibrate our model, we can solve an unconstrained optimization problem for

$$F(\vec{u}) = f\left(\left(\mathcal{L}(u_1), \mathcal{L}(u_2), \cdots, \mathcal{L}(u_{n_\psi})\right)\right). \tag{2.63}$$

In the following subsections, we provide a brief description of each optimization algorithm we considered for calibration. Our goal is to determine an efficient algorithm to calibrate MSCM. Since it is expensive to compute derivatives for this problem, we consider optimization algorithms with and without derivatives. The Jacobian J is specific to the Levenberg-Marquardt algorithm, and due to the nature of the algorithm we can only use the least squares (2.56) and relative least squares (2.59) error functions in the

[9] We assume that realistically, the CDO market spread is never zero. Otherwise, this creates an unfair situation for the investor in the tranche, since they are only covering losses in the event of a certain number of defaults, but are not receiving any payments in return.

Algorithm	Gradient	Jacobian	Hessian	Rate of Convergence
NMS	No	No	No	Linear (parameter-dependent) [20]
NMRS	No	No	No	Linear (parameter-dependent) [20]
NMSHD	No	No	No	Linear (parameter-dependent) [20]
NEWUOA	No	No	No	Superlinear [21, 22]
SD	Yes	No	No	Linear [23]
CGFR	Yes	No	No	Linear to Superquadratic [14]
CGPR	Yes	No	No	Linear to Superquadratic [14]
LM	No	Yes	No	Quadratic [25]
BFGS	Yes	No	No	Superlinear [24]
BFGS2	Yes	No	No	Superlinear [24]

Table 2.1: Algorithms used for model calibration, along with information about which derivatives they use and approximate rates of convergence. Most rate-of-convergence theory assumes exact line searches.

objective function (2.63). The Levenberg-Marquardt Jacobian computation is defined in Subsection 2.8.2.2.

The gradient $\vec{g}$ and the Hessian H are computed with respect to the unconstrained objective function $F(\vec{u})$ (2.63).

We considered the following optimization algorithms to calibrate the dynamic copula model: Nelder-Mead Simplex (NMS), Nelder-Mead Random Simplex (NMRS), Nelder-Mead Simplex for Higher Dimensions (NMSHD), Powell's Method (NEWUOA), Steepest Descent (SD), Conjugate Gradient Fletcher-Reeves (CGFR), Conjugate Gradient Polak-Ribière (CGPR), Levenberg-Marquardt Nonlinear Least Squares (LM) and Broyden-Fletcher-Goldfarb-Shanno (both BFGS and BFGS2, a more efficient implementation for higher dimensions). These optimization methods are summarized in Table 2.1.

2.8.1 Methods Without Derivatives

2.8.1.1 Nelder-Mead Simplex

The algorithm takes an input vector $\vec{u} = (u_1, u_2, \cdots, u_{n_\psi})$ and forms an n_ψ-dimensional simplex with $n_\psi + 1$ vertices $j = 0, 1, \cdots, n_\psi$ given by

$$\begin{aligned} \vec{v}_0 &= \vec{u}, \qquad (2.64) \\ \vec{v}_j &= (u_1, u_2, \cdots, u_j + s, \cdots, u_{n_\psi}) \text{ for } j = 1, 2, , \cdots, n_\psi, \end{aligned}$$

where s is the initial step size. The step size s changes for each dimension as the algorithm progresses. A single iteration consists of sorting the objective function values $F(\vec{v}_j)$ at each vertex v_j, and updating the simplex vertices using an algorithm which consists of geometrical operations on the simplex, such as reflection, reflection followed by expansion, contraction and multiple contraction. The simplex eventually contracts within some neighborhood of the minimum. A full description of the algorithm can be found in [12]. The GNU Scientific Library's (GSL) routine *nmsimplex*, which we denote by NMS, is one implementation of this algorithm. The GSL contains another variant of this algorithm, called *nmsimplex2rand*, which we denote by NMRS, for which the basis vectors are randomly oriented, and do not necessarily follow the coordinate axes. The GSL contains a third implementation of the Nelder-Mead algorithm, called *nmsimplex2*, which we denote by NMSHD, which is more efficient for higher dimensional problems. See [16] for implementation details.

2.8.1.2 Powell's Method (NEWUOA)

Powell's method, NEWUOA, is similar to the Nelder-Mead algorithm, but uses a set of coordinate axes as basis vectors, along which a bi-directional search is performed [13]. The function minimum can be expressed as a linear combination of these basis vectors. The algorithm keeps a set of basis vectors along which a significant improvement is achieved and ignores the rest, until convergence. For a detailed generic description of Powell's method (with pseudocode), see [15]. An implementation of the algorithm, deemed efficient for higher dimensions, is NEWUOA; see [13] for a detailed description of the software.

2.8.2 Methods With Derivatives

2.8.2.1 Gradient Methods

Steepest Descent (SD) This inefficient method is included for completeness. More efficient gradient search methods exist, such as Conjugate Gradient methods [14]. The GSL implementation of the steepest descent algorithm performs a line search in the direction of the gradient, doubling the step size after each successful step and decreasing the step size using a tolerance parameter if the step is unsuccessful; see [16] for a more detailed description.

Conjugate Gradient (Fletcher-Reeves (CGFR) & Polak-Ribière (CGPR)) The conjugate gradient method improves upon the steepest descent method by conjugating the gradient, thus implicitly accumulating information about the Hessian matrix [24]. If the objective function at step k of the algorithm is $F(\vec{u}_j)$, κ_j is the step size and $\vec{g}(\vec{u}_j)$ is the gradient at step k, then the line search is performed along the direction $\vec{s}_j$ using $F(\vec{u}_j + \kappa_j \vec{s}_j)$, where Fletcher and Reeves specify

$$\vec{s}_j = -\vec{g}(\vec{u}_j) + \frac{\left(\vec{g}(\vec{u}_j)\right)^T \vec{g}(\vec{u}_j)}{\left(\vec{g}(\vec{u}_{j-1})\right)^T \vec{g}(\vec{u}_{j-1})} \vec{s}_{j-1} \tag{2.65}$$

and Polak and Ribière specify

$$\vec{s}_j = -\vec{g}(\vec{u}_j) + \frac{\left(\vec{g}(\vec{u}_j) - \vec{g}(\vec{u}_{j-1})\right)^T \vec{g}(\vec{u}_j)}{\left(\vec{g}(\vec{u}_{j-1})\right)^T \vec{g}(\vec{u}_{j-1})} \vec{s}_{j-1} \tag{2.66}$$

as the two possible conjugations. Using exact arithmetic, both algorithms are exact for linear problems after n_ψ iterations [24].

2.8.2.2 Jacobian Methods

Levenberg-Marquardt Nonlinear Least Squares (LM) Let $\vec{m} \in \mathcal{R}^{n_m,+}$ denote a vector of CDO market quotes $m_{n_T}^{(\mathrm{tr})}$. Using the double-index notation from Section 2.7, let

$$E_k(\vec{\psi}) = E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right] \tag{2.67}$$

denote each expected spread term, and let $\mathcal{E}_k(\vec{u}) = E_k\left(\mathcal{L}(\vec{u})\right)$ denote the k-th element of vector $\vec{\mathcal{E}}$ of expected spreads across all tranches and maturities (containing n_m elements, as does $\vec{m}$). Then the least squares error function for vectors can we written as

$$\mathrm{error}_{\mathrm{LS,vec}}\left(\vec{\mathcal{E}}(\vec{u}),\vec{m}\right) = \left\|\vec{\mathcal{E}} - \vec{m}\right\|_2^2, \tag{2.68}$$

where $\vec{u} \in \mathcal{R}^{n_\psi}$. The relative least squares error function for vectors is given by

$$\mathrm{error}_{\mathrm{LS,vec}}^{\mathrm{rel}}\left(\vec{\mathcal{E}}./\vec{m},\vec{1}\right) = \left\|\vec{\mathcal{E}}./\vec{m} - \vec{1}\right\|_2^2 = \mathrm{error}_{\mathrm{LS,vec}}\left(\vec{\mathcal{E}}./\vec{m},\vec{1}\right), \tag{2.69}$$

where "./" denotes vector element-wise division and $\vec{1}$ denotes the vector of length n_m with all elements equal to 1. For (2.68), using a vector of small increments $\vec{\delta} \in \mathcal{R}^{n_m}$, we can approximate a change in parameters $\vec{u}$ using the Jacobian matrix $J_{k,j} = \left.\frac{\partial \mathcal{E}_k}{\partial u_j}\right|_{\vec{u}}$ as

$$\vec{\mathcal{E}}(\vec{u}+\vec{\delta}) \approx \vec{\mathcal{E}}(\vec{u}) + J\vec{\delta}, \tag{2.70}$$

where we are starting with some initial approximation J. Then we can compute the $\vec{\delta}$ that minimizes

$$\left\|\vec{\mathcal{E}}(\vec{u}) - \vec{m} - J\vec{\delta}\right\|_2^2. \tag{2.71}$$

Using the regularization parameter $\lambda \geq 0$ (superscript $'$ denotes transpose in this subsection) a regularized approximate solution to (2.71) is

$$\left(J'J + \lambda I\right)\vec{\delta} = J'\left(\vec{\mathcal{E}}(\vec{u}) - \vec{m}\right). \tag{2.72}$$

This completes the description of the general version of the Levenberg Marquardt optimization algorithm for (2.68), see [16] for implementation details. For the relative version

Algorithm 1 Generic BFGS algorithm for unconstrained optimization (transpose is denoted by superscript $'$).

$\vec{u}_0$= `initial guess`

H_0= `initial Hessian approximation`

`for j=0,1,2,...`

  `Solve` $H_j\vec{s}_j = -\nabla F(\vec{u}_j)$ `for` $\vec{s}_j$ `// compute quasi-Newton step`

  $\vec{u}_{j+1} = \vec{u}_j + \vec{s}_j$ `// update solution`

  $\vec{y}_j = \nabla F(\vec{u}_{j+1}) - \nabla F(\vec{u}_j)$

  $H_{j+1} = H_j + (\vec{y}_j\vec{y}_j{}')/(\vec{y}_j{}'\vec{s}_j) - (H_j\vec{s}_j\vec{s}_j{}'H_j)/(\vec{s}_j{}'H_j\vec{s}_j)$

`end`

of the the least squares function for vectors, the same derivations apply, using (2.69) as the error function, and interchanging $\vec{\mathcal{E}}$ and $\vec{m}$ in (2.68) in an obvious way, as specified in the definition of the relative error function for vectors (2.69).

2.8.2.3 Hessian Methods

BFGS The BFGS method uses an approximation to the Hessian matrix and preserves its symmetry and positive definiteness. For linear problems, it terminates at the exact solution after at most n_ψ iterations, if exact line searches and exact arithmetic are used. For the Hessian approximation formula, see Algorithm 6.5 in [24], restated as Algorithm 1.

GSL implements a more efficient version of the BFGS algorithm, which we denote by BFGS2, which is specified by Algorithms 2.6.2 and 2.6.4 in [10]; see [16] for implementation details.

Chapter 3

Calibration

As introduced in Section 2.5, the recently-proposed Multi-period Single-factor Copula Model (MSCM) [11] has two sets of parameters:

- default probabilities α_i satisfying (2.17) introduced in Section 2.2. These are a set of fixed parameters, denoted by $\vec{\alpha}$, which are calculated using a bootstrapping process from CDS market quotes;

- a variable set of constrained model parameters $\vec{\psi}$, which model market dynamics of the homogeneous pool of underlyings.

To calibrate the model, we need to determine a set of the constrained model parameters $\vec{\psi}$, so that the expected spreads $E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right]$ match the CDO market quotes $m_{n_T}^{(\mathrm{tr})}$ across a range of tranches tr and maturities T. This is accomplished by first choosing some error function: either one of the absolute error functions (2.56), (2.57) or (2.58), or one of the relative error functions (2.59), (2.60) or (2.61). Once the error function is fixed, we minimize the objective function (2.52), restated here for convenience:

$$f(\vec{\psi}) = \sum_{\mathrm{tr}\in\mathrm{Tr}} \sum_{T\in M} \mathrm{error}\left(E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right], m_T^{(\mathrm{tr})}\right), \tag{3.1}$$

where, for the models we develop in Section 3.1, $\psi_j \in [0,1]$ for all $j = 1, 2, \cdots, n_\psi$. An optimization algorithm has to be used in order to minimize this constrained objective function.

As noted in Section 2.8, we can convert the constrained optimization problem described above into an unconstrained one using the logistic function

$$\psi_j = \mathcal{L}(u_j) = \frac{1}{1+\exp(-u_j)}. \tag{3.2}$$

Note that $u_j \in \mathcal{R}$, $\psi_j = \mathcal{L}(u_j) \in [0,1]$ for all $j = 1, 2, \cdots, n_\psi$. The initial starting guess can be set using the inverse of the logistic function

$$u_j = -\ln\left(\frac{1-\psi_j}{\psi_j}\right), \tag{3.3}$$

assuming that we have a starting guess for $\vec{\psi}$. The unconstrained optimization problem can be stated as

$$\min_{\vec{u}\in\mathcal{R}^{n_\psi}} F(\vec{u}), \tag{3.4}$$

where

$$F(\vec{u}) = f\left(\left(\mathcal{L}(u_1), \mathcal{L}(u_2), \cdots, \mathcal{L}(u_{n_\psi})\right)\right). \tag{3.5}$$

To use the optimization methods surveyed in Section 2.8, the unconstrained objective function $F(\vec{u})$ and its derivatives with respect to elements of $\vec{u}$ have to be defined for parameter values α_i and β_i. In Section 3.2 we describe how to compute the unconstrained objective function $F(\vec{u})$. Firstly, $F(\vec{u})$ contains massively parallel regions, and we can also re-use certain data structures when computing $F(\vec{u})$ and its derivatives. This is outlined in Subsection 3.2.1. Next, in Subsection 3.2.2, we prove that the computation of $F(\vec{u})$ is defined for all $\alpha_i \in [0,1]$ for $i = 2, 3, \cdots, n_T$ and $\alpha_1 \in [0,1)$ and for all $\beta_i \in [0,1]$. We also show that the expected spread (2.37) quoted by MSCM is undefined when $\alpha_1 = 1$[1]. In Subsection 3.2.3, we describe how to compute the derivatives of $F(\vec{u})$ for all $\alpha_i \in (0,1)$ and for all $\beta_i \in [0,1)$. Unfortunately, we are unable to prove the existence of derivatives for all $\alpha_i \in [0,1)$ and all $\beta_i \in [0,1]$.

[1]Realistically, if $\alpha_1 = 1$, then it is unreasonable to create a CDO contract in the first place. Such scenarios should never occur in practice, yet have to be handled numerically as part of the pre-processing step when using MSCM.

3.1 Multi-Path Model Parameterization

Reference [8] proposes a tree model similar to that shown in Figure 2.1 in Section 2.5. In this parameterization, the β_i values are associated with μ_j: each β_i branches from a previous β_{i-1} value using $\beta_i = \beta_{i-1} \cdot \mu_j$ and $\beta_i = \beta_{i-1}/\mu_j$ at the start of each new model period, where $\mu_j \in (0, 1]$. An obvious difficulty in this approach is that we could have $\beta_i = \beta_{i-1}/\mu_j > 1$ for some value of i. The author simply truncates β_i at 1, and leaves $\mu_j \in (0, 1]$. We could avoid having $\beta_i > 1$ by adding constraints to the optimization problem, but we prefer to use models for which the change of variables described above allows us to use unconstrained optimization methods.

Other shortcomings of this parameterization, illustrated in Figure A.2, are:

1. during the first period, the β_i's follow a certain scenario with probability 1, i.e. the model does not account for market dynamics during that period;

2. during the last period, the probability of moving up when $\beta_i \approx 0$, $\beta_i > 0$ to a higher value is equivalent to the probability of moving from a value of $\beta_i \approx 1$, $\beta_i < 1$ to almost perfect correlation;

3. Figure A.2 shows Extreme Cases 1 and 2, where certain scenarios of β_i are not taken into account, i.e. they do not occur with probability 1;

4. as noted above, the parameterization could produce a $\beta_i > 1$;

5. each next period depends on the previous value of μ_j, which makes it difficult to obtain derivatives with respect to β_i for subsequent periods.

The alternative multi-path parameterization described below addresses these deficiencies by letting the model adjust the possible values of β_i in each period, independently of other periods, with unique probabilities. Furthermore, the probability of transitioning to another period does not depend on the previous period.

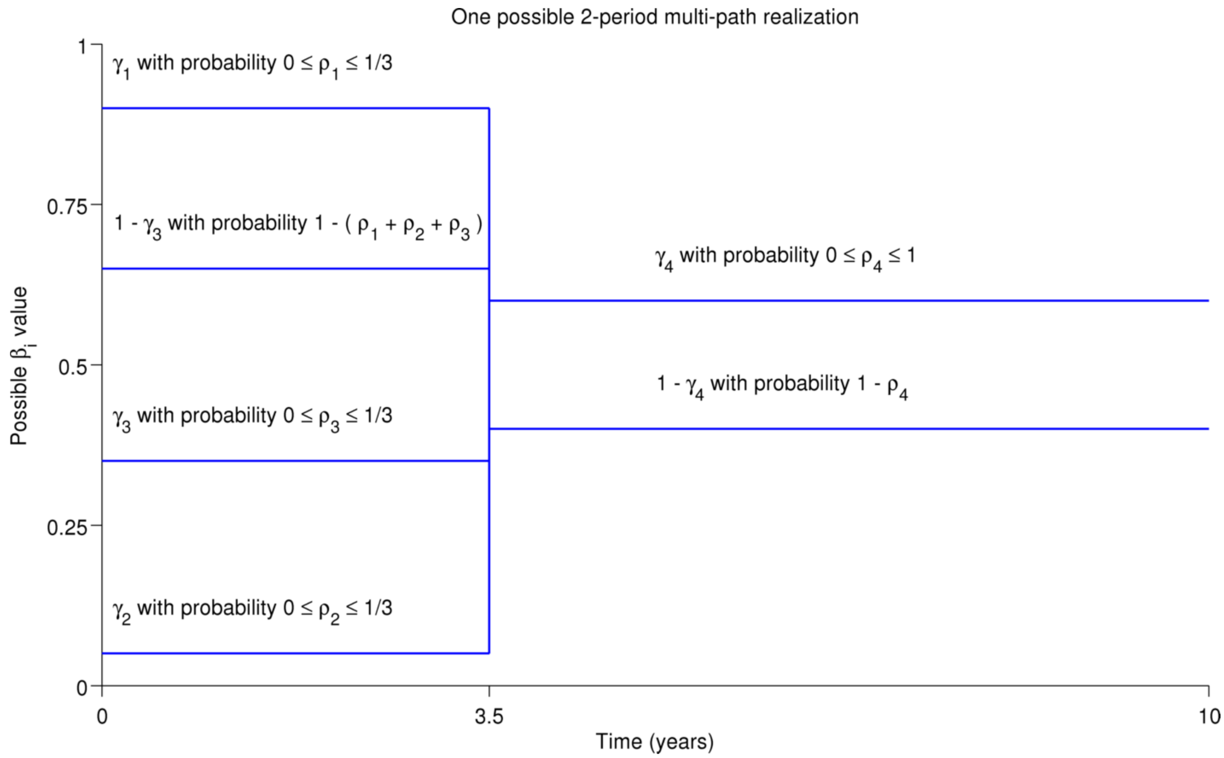

Figure 3.1: A simple example of one possible configuration of the alternative multi-path parameterization used in this thesis. The switch from one period to the next can be adjusted arbitrarily, and more periods can be added in more complicated parameterizations.

Multi-path parameterization associates a set of branch parameters γ and probabilities ρ with each period. Figure 3.1 depicts a simple example of this parameterization using 2 periods, where the first period ends and the second period begins at 3.5 years. The point at which one period ends and another begins is chosen as an arbitrary fixed value in our models, although it could be a model parameter in more sophisticated models. We associate one γ_4 parameter and one ρ_4 probability parameter with the second period. Thus, the second period has two possible scenarios. This is the minimum number of scenarios per period that we use in the multi-path parameterization. We can add more paths to each period, as shown in the first period, in this case, but we have to restrict the probabilities ρ, because the sum of the path probabilities must be 1 in each period. We

Algorithm 2 Pseudocode to restrict the probabilities ρ_j to smaller intervals, if needed. Probabilities ρ_j are parameterized by $\psi_{n_\gamma+j}$.

```
// All indexes start at 0
per = 0; // indexes the period
nbefore = 0;
nparam = n_ψ/2; // number of ρ parameters, n_ψ is always even
for j=0:(nparam - 1)
  // skip over γ parameters in the gradient
  ρ_j = (1.0 / number of ρ parameters in period per)*ψ_{j+nparam}
  if ((j+1) - nbefore >= number of ρ parameters in period per)
    // record number of ρ parameters that we've passed
    nbefore += number of ρ parameters in period per
    per += 1  // move to the next period
```

can enforce such constraints by modifying $\vec{\psi}_{n_\gamma+1:n_\psi}$ in $\vec{\psi} = (\vec{\gamma}, \vec{\psi}_{n_\gamma+1:n_\psi})$ in the obvious way, where each $\psi_j \in [0,1]$ for all $j = 0, 1, \cdots, n_\psi$, originally given by (3.2). We simply divide each $\psi_{n_\gamma+j}$, responsible for ρ_j, by the number of ρ parameters in each period. For example, in Figure 3.1, there are only 2 paths in the second period, associated with ρ_4, so in that particular case, $\rho_4 = \psi_8$. In the first period, $\rho_j = \psi_{n_\gamma+j}/3$ for all $j = 1, 2, 3$, because there are three ρ probabilities associated with the first period. This adjustment of the $\vec{\psi}_{n_\gamma+1:n_\psi}$ values is detailed in the pseudocode in Algorithm 2.

The parameterization shown in Figure 3.1 has 8 distinct paths. For example, one such path is $\beta_i = \gamma_1$ for all $i = 1, 2, \cdots, 14$ (the period switch at 3.5 years occurs after 14 quarterly payments), and $\beta_i = 1 - \gamma_4$ for all $i = 15, 16, \cdots, 40$ with probability $\rho_1 \cdot (1 - \rho_4)$.

Let r_j denote the period, where $j = 1, 2, \cdots, n_r$ and n_r is the number of periods. Each r_j is a time interval that has associated with it a set of parameters $\gamma_{r_j,k}$ and $\rho_{r_j,k}$,

$k = 1, 2, \cdots, n_j$. Then all the β_i's associated with period r_j satisfy either

$$\begin{aligned} \beta_i &= \gamma_{r_j,k}, \text{ with probability } \rho_{r_j,k},\ 0 \le \rho_{r_j,k} \le 1/n_j \\ &\text{or} \\ \beta_i &= 1 - \gamma_{r_j,n_j}, \text{ with probability } 1 - \sum_{k=1}^{n_j} \rho_{r_j,k}. \end{aligned} \tag{3.6}$$

For example, if we have a single period r_1 that covers the fill lifetime of the CDO and if $n_1 = 1$, then all the β_i's are either γ_1 or $1 - \gamma_1$ with probabilities ρ_1 and $1 - \rho_1$, respectively, where γ_1 and $\rho_1 \in [0, 1]$. We have double-indexed elements of $\vec{\gamma}$ and $\vec{\rho}$ to succinctly represent the parameterization, but the optimization algorithm can only be given a single vector $\vec{\psi}$. We now provide pseudocode for associating β_i with a particular scenario, indexed by r_Θ and c_Θ.

The set of parameters can be partitioned into $\vec{\psi} = (\vec{\gamma}, \vec{\psi}_{n_\gamma+1:n_\psi})$, where $\vec{\gamma} = (\gamma_1, \gamma_2, \cdots, \gamma_{n_\gamma})$ and $\vec{\psi}_{n_\gamma+1:n_\psi} = (\psi_{n_\gamma+1}, \psi_{n_\gamma+2}, \cdots, \psi_{n_\psi})$ and $n_\psi = 2n_\gamma$[2]. We have to efficiently extract the parameters associated with each period from the parameter vector $\vec{\psi}$. Consider an indexing convention with rows r_Θ indexing the possible parameter scenarios (also called period branches) for each period r_j, and with columns c_Θ indexing the period[3]. Then given the time index i, r_Θ and c_Θ we can determine the corresponding value of β_i and the corresponding probability. The pseudocode for extracting parameter values is given by Algorithms 3 and 4 below. For example, in Figure 3.1 the constrained parameter vector is $\vec{\psi} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4, \psi_5, \psi_6, \psi_7, \psi_8)$. If we start indexing at 0, then for $(r_\Theta, c_\Theta) = (3, 0)$, $\beta_i = 1 - \gamma_3$ with probability $1 - (\rho_1 + \rho_2 + \rho_3)$, i.e. there are three $\gamma \in \vec{\gamma}$ parameters associated with the first period, and the last branch in the period is parameterized by the value $1 - \gamma_3$. Note that $1 - \gamma_3 \in [0, 1]$m since $\gamma_3 \in [0, 1]$.

For example, for the multi-path parameterization depicted in Figure A.2, there would be $2^4 = 16$ possible scenarios (paths) that β_i could take, each with its own unique

[2] The number of elements in $\vec{\psi}$ is always even by construction.

[3] Θ denotes a hypothetical matrix. Imagine the paths in Figure 3.1 as elements in the matrix Θ, then the depicted parameterization forms a 4×2 matrix Θ, with empty entries in $\Theta_{3,2}$ and $\Theta_{3,3}$, if Θ indexing starts at zero.

Algorithm 3 Pseudocode to extract the probability related to the period c_Θ and to the branch r_Θ from the parameter vector $\vec{\rho}$.

```
// Indexes for rθ and cθ start at 0
function prob = get_prob(rΘ,cΘ,ρ⃗)
// Determine how many probs ρ were in previous periods
nprobs_before = 0;
// Count the number of parameters in previous periods
// and skip over them
for j=0:(cΘ-1)
  nprobs_before += number of parameters used in period j
// Determine which ρ value to currently use
if (number of parameters used in period cΘ == rΘ)
  // subtract the last probs in current branch column
  prob = 1
  for j=0:(cΘ-1)
    prob -= ρ⃗_{nprobs_before+j} // ρ⃗ index starts at zero
else if (rΘ < number of parameters used in period cΘ)
  prob = ρ⃗_{nprobs_before+rΘ}    // ρ⃗ index starts at zero
else
  error("This should never happen.")
return prob;
```

probability. Figure A.3 shows more possible parameterizations. It is possible to have many market quotes for the calibration (see Section 6.1 for the description of the data sets available). Hence the depicted parameterizations still result in a parsimonious model.

Algorithm 4 Pseudocode to extract the value of β_i related to the period c_Θ and to the branch r_Θ from the parameter vector $\vec{\gamma}$.

```
// Indexes for r_θ and c_θ start at 0
function β_i = get_beta(r_Θ,c_Θ,γ⃗)
// Determine how many γ were in previous periods
ngammas_before = 0;
// Count the number of parameters in previous periods
// and skip over them
for j=0:(c_Θ-1)
  ngammas_before += number of parameters used in period j
// Determine which γ value to currently use
if (number of parameters used in period c_Θ == r_Θ)
  β_i = 1 - γ⃗_{ngammas_before+r_Θ-1} // γ⃗ index starts at zero
else if (r_Θ < number of parameters used in period c_Θ)
  β_i = γ⃗_{ngammas_before+r_Θ}        // γ⃗ index starts at zero
else
  error("This should never happen.")
return β_i;
```

3.2 Objective Function And Derivative Computation For Optimization

In this section, we first describe how one can exploit the massive parallelism of MSCM when evaluating the constrained objective function $f(\vec{\psi})$ (2.52), or equivalently, the unconstrained objective function $F(\vec{u})$ (3.5). We also outline in Subsection 3.2.1 how to improve the efficiency of objective function evaluation by re-using various probabilities and expectations during pricing. We then prove that either objective function can be evaluated for all $\alpha_i \in [0,1]$ for $i = 2, 3, \cdots, n_T$ and $\alpha_1 \in [0,1)$ and for all $\beta_i \in [0,1]$ in Subsection 3.2.2; we also show that the expected spread (2.37) quoted by MSCM is undefined when $\alpha_1 = 1$. In Subsection 3.2.3 we proceed to compute the derivatives of the objective function $F(\vec{u})$ with respect to elements of $\vec{u}$ for all $\alpha_i \in (0,1)$ and all $\beta_i \in [0,1)$. Unfortunately, we are unable to prove anything about the existence of derivatives for $\alpha_i \in [0,1)$ for $i = 2, 3, \cdots, n_T$ and $\alpha_1 \in [0,1)$ and for all $\beta_i \in [0,1]$.

3.2.1 Parallelism

The goal is to minimize either the constrained objective function $f(\vec{\psi})$ (2.52), or equivalently, the unconstrained objective function $F(\vec{u})$ (3.5), which reduces to computing spreads for different scenarios of β_i. The computation is partitioned into three stages, each of which is massively parallel:

1. integrate the probability matrices A_i (2.43) for all scenarios of β_i and store them in memory. This can be done in parallel, since each A_i depends on α_i and β_i only.

2. compute data structures which store $P\left(l_i^{(\text{pool})} = r\right)$ and $E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$, which are reused during the next pricing stage. This can also be done in parallel by creating a parallel process for each scenario in the multi-path parameterization.

3. price the CDO spreads $s_{n_T}^{(\text{tr})}(\vec{\beta}, \vec{\alpha})$ for different scenarios of $\vec{\beta}$, using the data structures in stage 2, which are dynamically populated during pricing (see Subsection 5.2.1 for a specific description of the dynamic programming implementation). This too can be done in parallel.

3.2.2 Objective Function Evaluation

The unconstrained objective function $F(\vec{u})$ (3.5) is defined for all $\alpha_i, \beta_i \in (0,1)$. The goal of this subsection is to extend the definition of $F(\vec{u})$ for all $\alpha_i \in [0,1]$ for $i = 2, 3, \cdots, n_T$ and $\alpha_1 \in [0,1)$ and all $\beta_i \in [0,1]$. This is accomplished by considering the four limits:

$\alpha_i \to 0^+$, $\alpha_i \to 1^-$, $\beta_i \to 0^+$ and $\beta_i \to 1^-$, denoted for brevity as $\lim_{\alpha_i,\beta_i \to 0^+,1^-}$, of the objective function $F(\vec{u})$

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} F(\vec{u}). \tag{3.7}$$

We also demonstrate that the pricing equation for the spread quote (2.9), restated below for convenience

$$s^{(\mathrm{tr})}_{n_T} = \frac{\sum_{i=1}^{n_T-1} E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right] \cdot (f_i - f_{i+1}) + E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_{n_T}\right] \cdot f_{n_T}}{\sum_{i=1}^{n_T} \left(K \cdot N \cdot S^{(\mathrm{tr})} - E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right]\right) \cdot (t_i - t_{i-1}) \cdot f_i} \tag{3.8}$$

is undefined when $\alpha_1 = 1$, which makes the expected spread $E_{\vec{\rho}}\left[s^{(\mathrm{tr})}_{n_T}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right]$ (2.37), quoted by MSCM also undefined[4].

Recall from Section 2.5, that the computation of $E_{\vec{\rho}}\left[s^{(\mathrm{tr})}_{n_T}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right]$ (2.37) is reduced in Section 2.1 to the computation of the MSCM spread approximation $s^{(\mathrm{tr})}_{n_T}(\vec{\beta},\vec{\alpha})$ (2.9), restated as (3.8). In the same section, this is further reduced to the computation of $E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right]$ (2.16), restated below in (3.9). Computation of $E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right]$ is further reduced to the problem of estimating $P\left(l^{(\mathrm{pool})}_i = r\right)$, given by the recursion relationship (2.29), restated below for convenience in (3.17). It is trivial matter to verify that, provided the denominator of the spread (3.8) is non-zero, each of the four aforementioned limits $\alpha_i, \beta_i \to 0^+, 1^-$ is defined up to the stage of computing $P\left(l^{(\mathrm{pool})}_i = r\right)$. Recall from Section 2.1 that $E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right]$ (2.16) is given by

$$E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right] = N \cdot (1-R) \cdot \sum_{r=1}^{K} \min\left(\frac{K \cdot S^{(\mathrm{tr})}}{1-R}, \max\left(0, r - \frac{K \cdot a^{(\mathrm{tr})}}{1-R}\right)\right) \cdot P\left(l^{(\mathrm{pool})}_i = r\right), \tag{3.9}$$

where $a^{(\mathrm{tr})}$ and $S^{(\mathrm{tr})}$ are given in percent, and $L^{(\mathrm{pool})}_i = N \cdot (1-R) \cdot l^{(\mathrm{pool})}_i$, $l^{(\mathrm{pool})}_i \in 0, 1, \cdots, K$. Notice that from the no arbitrage argument in [11], the expected pool loss $E_{(\mathrm{pool})}\left[L^{(\mathrm{tr})}_i\right]$ is a monotonically increasing function. Hence, the denominator in (3.8)

[4]Recall from Section 2.1 that the Equity tranche is priced differently, and its pricing equation is defined when $\alpha_i = 1$. However, we are later concerned with the Mezzanine tranches in this thesis, and the setting of $\alpha_i = 1$ has to be handled.

can only be zero if for time $i = 1$ (first time step) for any

$$r \geq \frac{K}{1-R}\left(S^{(\mathrm{tr})} + a^{(\mathrm{tr})}\right) \tag{3.10}$$

we have $P\left(l_1^{(\mathrm{pool})} = r\right) = 1$. Following the no arbitrage argument in [8], the expected tranche losses $E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right]$ are monotonically increasing. We later show that when $\alpha_1 = 1$,

$$\lim_{\alpha_1 \to 1^-} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] = K \cdot N \cdot S^{(\mathrm{tr})} \tag{3.11}$$

for all $i = 1, 2, \cdots, n_T$ and all tranches tr. This makes (3.8) undefined for all n_T and all tr.

To compute $\lim_{\alpha_i,\beta_i \to 0^+,1^-} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right]$, we have to compute $\lim_{\alpha_i,\beta_i \to 0^+,1^-} P\left(l_i^{(\mathrm{pool})} = r\right)$. Let $P_i \in \mathcal{R}^{K+1}$ denote the exact value of the probability vector at time t_i (pool loss probability) with $K+1$ elements $P\left(l_i^{(\mathrm{pool})} = r\right)$, $r = 0, 1, \cdots, K$, and let A_i denote the exact value of the lower triangular integration matrix with entries $[A_i]_{r,m} = P\left(l_{(i-1,i]}^{(\mathrm{pool}),K-m} = r - m\right)$ (2.43), restated here for convenience

$$[A_i]_{r,m} = c\int_{-\infty}^{\infty} h(x)\phi(x)dx, \tag{3.12}$$

where

$$h(x) = \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)^{r-m}\left(1 - \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)\right)^{K-r}, \tag{3.13}$$

is the Riemann integrand, where the functions $p_{k,i}(x) = p_i(x)$ (2.25) are

$$p_i(x) = \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right), \tag{3.14}$$

where $\Phi(\cdot) \in [0, 1]$ is the standard normal Cumulative Density Function (CDF) and Φ^{-1} is its inverse. The Riemann constant of integration is given by

$$c = \begin{pmatrix} K - m \\ r - m \end{pmatrix} \tag{3.15}$$

and the standard normal probability density function is given by

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right). \tag{3.16}$$

Then the recursion (2.29) can be written succinctly as

$$P_i = A_i P_{i-1} \tag{3.17}$$

using matrix-vector multiplication.

The computation of the four limits of the objective function reduces to

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} c \int_{-\infty}^{\infty} h(x)\phi(x)dx. \tag{3.18}$$

In each of the four cases, $\beta_i \to 1^-$, $\beta_i \to 0^+$, $\alpha_i \to 1^-$ and $\alpha_i \to 0^+$, for any $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$, one way of finding

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} c \int_{-\infty}^{\infty} h(x)\phi(x)dx \tag{3.19}$$

is to prove

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} c \int_{-\infty}^{\infty} h(x)\phi(x)dx = c \int_{-\infty}^{\infty} \lim_{\alpha_i,\beta_i \to 0^+,1^-} h(x)\phi(x)dx \tag{3.20}$$

and then to compute the right hand side in the above equation. Otherwise, we cannot say anything about the limits, because the integral specifying $[A_i]_{r,m}$ is intractable analytically. We can invoke the Dominated Convergence Theorem (DCT) and it corollaries, stated in [48][5]. We can rewrite (3.12) as a Riemann-Stieltjes integral using

$$c \int_{-\infty}^{\infty} h(x)\phi(x)dx = c \int_{-\infty}^{\infty} h(x)d\Phi(x), \tag{3.21}$$

where Φ denotes the standard normal Cumulative Distribution Function (CDF) that has a probability density function $\phi(x)$ (3.16) with respect to Lebesgue measure.

We are now in the position to state the assumptions, required to use DCT. First, we must obtain an integrable dominator $G(x)$, such that

$$|h(x)| \le G(x) \tag{3.22}$$

[5] DCT and its corollaries are well-known, and are available from other functional analysis texts.

for all $x \in \mathcal{R}$. Setting $G(x) = 1$ provides such a bound, because $h(x)$ only contains probabilities, raised to positive powers. Clearly $G(x)$ is integrable with respect to $\phi(x)$, because

$$\int_{-\infty}^{\infty} d\Phi(x) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = 1 < \infty \tag{3.23}$$

is just the area under the standard normal probability density.

Next, before applying DCT, we must show that any of the four limits exist, i.e.

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} h(x) = h^{(\text{conv})}(x) \tag{3.24}$$

for all $x \in \mathcal{R}$[6] for some function $h^{(\text{conv})}(x)$. In the results below, we are using the fact that the standard normal Cumulative Density Function (CDF) $\Phi(x)$ and its inverse $\Phi^{-1}(x)$ are continuous functions, whence

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} \Phi\left(\chi(x)\right) = \Phi\left(\lim_{\alpha_i,\beta_i \to 0^+,1^-} \chi(x)\right), \tag{3.25}$$

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} \Phi^{-1}\left(\chi(x)\right) = \Phi^{-1}\left(\lim_{\alpha_i,\beta_i \to 0^+,1^-} \chi(x)\right), \tag{3.26}$$

if $\lim_{\alpha_i,\beta_i \to 0^+,1^-} \chi(x)$ exists for some function $\chi(x)$. We now determine $h^{(\text{conv})}(x)$ for all $r = 0, 1, \cdots, K$ and all $m = 0, 1, \cdots, r$ for each of the four limit cases:

1. Case $\alpha_i \to 1^-$: for all $r = K$ and $m = 0, 1, \cdots, K$, we obtain $\lim_{\alpha_i \to 1^-} h(x) = 1$. Otherwise, $\lim_{\alpha_i \to 1^-} h(x) = 0$.

2. Case $\alpha_i \to 0^+$: for all $r = 0, 1, \cdots, K$ and $m = r$, we obtain $\lim_{\alpha_i \to 0^+} h(x) = 1$. Otherwise, $\lim_{\alpha_i \to 0^+} h(x) = 0$.

3. Case $\beta_i \to 0^+$: for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$, we obtain $\lim_{\beta_i \to 0^+} h(x) = \alpha_i^{r-m} \left(1 - \alpha_i\right)^{K-r}$, because $\Phi\left(\Phi^{-1}(\alpha_i)\right) = \alpha_i$.

[6] We can relax convergence to hold almost everywhere [48], but in our case we can use this stronger result.

4. Case $\beta_i \to 1^-$: we obtain

$$h^{(\text{conv})}(x) = \begin{cases} 1^{r-m}0^{K-r}, & \text{if } x < \Phi^{-1}(\alpha_i); \\ (1/2)^{r-m}\,(1/2)^{K-r}, & \text{if } x = \Phi^{-1}(\alpha_i); \\ 0^{r-m}1^{K-r}, & \text{if } x > \Phi^{-1}(\alpha_i). \end{cases} \tag{3.27}$$

We have established all assumptions necessary to use DCT. We can now apply DCT, which states that under the above conditions

$$\lim_{\alpha_i,\beta_i \to 0^+,1^-} \int_{-\infty}^{\infty} h(x) d\Phi(x) = \int_{-\infty}^{\infty} \lim_{\alpha_i,\beta_i \to 0^+,1^-} h(x) d\Phi(x). \tag{3.28}$$

In the first three cases below, we are integrating a constant with respect to the standard normal Lebesgue probability measure. The last case is more tricky and requires another proof, stated as part of the case 4 below. We now obtain the following four cases:

1. Case $\alpha_i \to 1^-$: for $r = K$ and $m = 0, 1, \cdots, K$, we obtain $[A_i]_{K,m} = 1$ (i.e. last row of A_i is filled with 1's); otherwise $[A_i]_{r,m} = 0$.

2. Case $\alpha_i \to 0^+$: for $r = m$ and $m = 0, 1, \cdots, K$, we obtain $[A_i]_{m,m} = 1$ (i.e. the diagonal of A_i is filled with 1's); otherwise $[A_i]_{r,m} = 0$.

3. Case $\beta_i \to 0^+$: for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$, we obtain $[A_i]_{r,m} = \begin{pmatrix} K-m \\ r-m \end{pmatrix} \alpha_i^{r-m} (1-\alpha_i)^{K-r}$.

4. Case $\beta_i \to 1^-$: we start with the following equality, and then consider different cases for different values of r and m:

$$\begin{aligned} c\int_{-\infty}^{\infty} h^{(\text{conv})}(x) d\Phi(x) = \\ \lim_{\xi \to 0} \Bigg[\int_{-\infty}^{\Phi^{-1}(\alpha_i)-\xi} h^{(\text{conv})}(x) d\Phi(x) + \int_{\Phi^{-1}(\alpha_i)-\xi}^{\Phi^{-1}(\alpha_i)+\xi} h^{(\text{conv})}(x) d\Phi(x) \\ + \int_{\Phi^{-1}(\alpha_i)+\xi}^{\infty} h^{(\text{conv})}(x) d\Phi(x) \Bigg]. \end{aligned} \tag{3.29}$$

First, consider the middle term $\int_{\Phi^{-1}(\alpha_i)-\xi}^{\Phi^{-1}(\alpha_i)+\xi} h^{(\mathrm{conv})}(x)d\Phi(x)$ and notice that the integrand $0 \leq h^{(\mathrm{conv})}(x) \leq 1$ for all $x \in [\Phi^{-1}(\alpha_i)-\xi, \Phi^{-1}(\alpha_i)+\xi]$ and for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$. Hence, we can use the squeeze theorem [49] to prove that $\lim_{\xi\to 0} \int_{\Phi^{-1}(\alpha_i)-\xi}^{\Phi^{-1}(\alpha_i)+\xi} h^{(\mathrm{conv})}(x)d\Phi(x) = 0$ using the following result:

$$\begin{aligned} 0 &\leq \lim_{\xi\to 0} \int_{\Phi^{-1}(\alpha_i)-\xi}^{\Phi^{-1}(\alpha_i)+\xi} h^{(\mathrm{conv})}(x)d\Phi(x) \\ &\leq \lim_{\xi\to 0} \int_{\Phi^{-1}(\alpha_i)-\xi}^{\Phi^{-1}(\alpha_i)+\xi} d\Phi(x) = \lim_{\xi\to 0} \left[\Phi\left(\Phi^{-1}(\alpha_i)+\xi\right) - \Phi\left(\Phi^{-1}(\alpha_i)-\xi\right)\right] = 0. \end{aligned} \tag{3.30}$$

For all $r = K$ and all $m = 0, 1, \cdots, K-1$, and using the fact that $\Phi\left(\Phi^{-1}(\alpha_i)\right) = \alpha_i$ we obtain $\int_{-\infty}^{\Phi^{-1}(\alpha_i)-\xi} h^{(\mathrm{conv})}(x)d\Phi(x) = \Phi\left(\Phi^{-1}(\alpha_i)-\xi\right)$, and taking the limit $\xi \to 0$ we obtain α_i. The term $h^{(\mathrm{conv})}(x)$ is zero for all $x \in [\Phi^{-1}(\alpha_i)+\xi, \infty]$ for these settings of r and m. Hence, $c\int_{-\infty}^{\infty} h^{(\mathrm{conv})}(x)d\Phi(x) = c \cdot \alpha_i = \alpha_i$, since $c = \begin{pmatrix} K-m \\ r-m \end{pmatrix} = \begin{pmatrix} K-m \\ K-m \end{pmatrix} = 1$.

Similarly, for all $m = r$ and $r = 0, 1, \cdots, K-1$ we obtain $h^{(\mathrm{conv})}(x) = 0$ for all $x \in [-\infty, \Phi^{-1}(\alpha_i)-\xi]$ and $\int_{\Phi^{-1}(\alpha_i)+\xi}^{\infty} h^{(\mathrm{conv})}(x)d\Phi(x) = 1 - \Phi\left(\Phi^{-1}(\alpha_i)+\xi\right)$ and taking the limit as $\xi \to 0$ we obtain $1-\alpha_i$. Hence, $c\int_{-\infty}^{\infty} h^{(\mathrm{conv})}(x)d\Phi(x) = c(1-\alpha_i) = 1-\alpha_i$, since $c = \begin{pmatrix} K-m \\ r-m \end{pmatrix} = \begin{pmatrix} K-r \\ 0 \end{pmatrix} = 1$.

For all other r and m, except $r = m = K$, $h^{(\mathrm{conv})}(x) = 0$ on $[-\infty, \Phi^{-1}(\alpha_i)-\xi]$ and $[\Phi^{-1}(\alpha_i)+\xi, \infty]$. Notice that at $r = K$ and $m = r$ the first and last terms in (3.29) are α_i and $1-\alpha_i$, respectively, and sum to 1 after taking individual limits as $\xi \to 0$. Hence, $c\int_{-\infty}^{\infty} h^{(\mathrm{conv})}(x)d\Phi(x) = c = 1$, since $c = \begin{pmatrix} K-r \\ r-m \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = 1$.

In summary, for all $r = K$ and all $m = 0, 1, \cdots, K-1$, we obtain $[A_i]_{K,m} = \alpha_i$ (i.e. last row of A_i is filled with α_i) and for all $m = r$ and $r = 0, 1, \cdots, K-1$, we obtain $[A_i]_{r,r} = 1-\alpha_i$ (i.e. diagonal of A_i is filled with $1-\alpha_i$). For $r = K$ and $m = K$, we obtain $[A_i]_{K,K} = 1$.

Notice that in all of the above cases for $\alpha_i, \beta_i \to 0^+, 1^-$, the columns of A_i sum to 1. It could also happen that a combination of the above cases could occur, for example $\alpha_i \to 0^+$ and $\beta_i \to 1^-$. In this case, we would handle the α_i cases first, because the spreads $s_{n_T}^{(\text{tr})}$ (2.9) have to be defined for any realization of β_i, before computing the expectation $E_{\vec{\rho}}\left[s_{n_T}^{(\text{tr})}(\vec{\beta}, \vec{\alpha})|\vec{\gamma}\right]$. Notice, however, that for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$ we can verify the following result by direct computation:

$$\begin{aligned} \lim_{\beta_i \to 0^+, 1^-} \lim_{\alpha_i \to 0^+, 1^-} c \int_{-\infty}^{\infty} h(x)\phi(x)dx &= \lim_{\alpha_i \to 0^+, 1^-} \lim_{\beta_i \to 0^+, 1^-} c \int_{-\infty}^{\infty} h(x)\phi(x)dx \\ &= \lim_{\alpha_i \to 0^+, 1^-} c \int_{-\infty}^{\infty} h(x)\phi(x)dx, \end{aligned} \tag{3.31}$$

because α_i limit cases produce values which do not depend on β_i.

Using the notation from the recursion relationship (3.17), the multi-path parameterization branches described in Section 3.1 are started with the column vector $P_0 = (1, 0, \cdots, 0)$ and $E_{(\text{pool})}\left[l_0^{(\text{tr})}\right] = 0$ for all tranches tr. Now, consider the first time step in the recursion relationship (3.17). For each of the four aforementioned limit cases of $[A_i]$ we obtain the following values for the column vector P_1:

1. Case $\alpha_1 \to 1^-$: $P_1 = (0, 0, \cdots, 0, 1)$.

2. Case $\alpha_1 \to 0^+$: $P_1 = (1, 0, 0 \cdots, 0)$.

3. Case $\beta_1 \to 0^+$: $P_1 = (1 - \alpha_1, 0, 0 \cdots, 0, \alpha_1)$.

4. Case $\beta_1 \to 1^-$: the r-th element of P_1 is given by $[P_1]_r = \begin{pmatrix} K \\ r \end{pmatrix} \alpha_1^r (1 - \alpha_1)^{K-r}$, for $r = 0, 1, \cdots, K$.

From case 1 above, we get that $P\left(l_1^{(\text{pool})} = K\right) \to 1$ whenever $\alpha_1 \to 1^-$. The condition (3.10) is then satisfied[7], and, as mentioned at the beginning of this subsection, the spread

[7]Recall that, following the no-arbitrage argument in [11] the default probabilities α_i are monotonically increasing.

pricing equation (3.8) becomes undefined in the limit $\alpha_1 \to 1^-$ because relationship (3.11) produces a zero in the denominator of (3.8) for any maturity T and tranche tr.

Hence, we have shown that the objective function $F(\vec{u})$ (3.5) is defined for $\alpha_i \in [0,1]$ for $i = 2, 3, \cdots, n_T$ and $\alpha_1 \in [0,1)$ and for all $\beta_i \in [0,1]$. The next section attempts to derive a similar result for the derivatives of the objective function with respect to the unconstrained set of MSCM parameters $\vec{u}$.

3.2.3 Derivatives Of The Objective Function

In this subsection we compute the derivatives of the unconstrained objective function $F(\vec{u})$ (3.5) with respect to the elements u_ν of the parameter vector $\vec{u}$. We also prove that we can compute these derivatives for all $\alpha_i \in (0,1)$ and for all $\beta_i \in [0,1)$.

The derivative of the logistic function $\mathcal{L}$ is

$$\frac{\partial}{\partial u_\nu}\mathcal{L}(u_\nu) = \mathcal{L}(u_\nu) \cdot (1 - \mathcal{L}(u_\nu)) \,. \tag{3.32}$$

Recall that the probabilities $\vec{\rho}$ are scaled from $\vec{\psi}_{n_\gamma+1:n_\psi}$, as described in Section 3.1. Using the chain rule, this simply adds the same scaling factor of the reciprocal of the number of ρ parameters responsible for a certain period, in an obvious way, as was described in Section 3.1. Detailed pseudocode for gradient computation is provided by Algorithm 5.

The derivatives of the error functions are provided in Section 2.7 and they are easy to compute.

Let us denote the expected spread, conditional on the values of parameters $\vec{\gamma}$ and $\vec{\rho}$ by $E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta})|\vec{\gamma}\right] = E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta},\vec{\alpha})|\vec{\gamma}\right]$, removing the notation which specifies dependence on the vector of fixed parameters $\vec{\alpha}$. The derivatives of this expected spread with respect to probabilities $\rho_\nu \in \vec{\rho}$ is easy to compute. Expression $E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta})|\vec{\gamma}\right]$ either has the term ρ_ν multiplying some realization $s_{n_T}^{(\mathrm{tr})}$ of the spread, or the term $-\rho_\nu$, or $E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta})|\vec{\gamma}\right]$ may not depend on ρ_ν at all. We illustrate the computation of the gradient with the example in Figure 3.1. If we would like to compute the expected spread with maturity

Algorithm 5 Pseudocode to scale the gradient when computing the derivative with respect to probabilities, parameterized by $\psi_{j+n_\psi/2}$.

```
// All indexes start at 0
c_Θ = 0; //period index
nbefore = 0;
nparam = n_ψ/2; // number of ρ parameters, n_ψ is always even
for j=0:(nparam - 1)
  // skip over γ parameters in the gradient
  gradient(j+nparam) = (1.0 / number of ρ parameters in period c_Θ)
    *L_logistic_function_derivative(u_{j+nparam})*(∂_{ρ_j} error_function at u_{j+nparam})
  if ((j+1) - nbefore >= number of ρ parameters in period c_Θ)
    // record number of ρ parameters that we've passed
    nbefore += number of ρ parameters in period c_Θ
    c_Θ += 1  // move to the next period
```

of $T = 3$ years, then there are 4 possible spread scenarios available, let us label them by s_j, where $j = 1, 2, 3, 4$. Then the expected spread is given by

$$E_{\vec{\rho}}\left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta})|\vec{\gamma}\right] = \rho_1 s_1 + \rho_2 s_2 + \rho_3 s_3 + (1 - \rho_1 - \rho_2 - \rho_3) s_4, \tag{3.33}$$

and it is a trivial matter to compute the derivatives with respect to ρ_ν, $\nu = 1, 2, 3, 4$. If we consider a maturity of $T = 5$ years, then there are 8 possible scenarios available for the spread, and ρ_4 is now involved in the computation, unlike in (3.33). For example, one possible path out of 8 could have the probability $(1 - \rho_1 - \rho_2 - \rho_3)(1 - \rho_4)$ multiplying some spread realization. Again, it is trivial to compute the derivatives with respect to ρ_ν.

We now describe how to compute the derivatives with respect to γ_ν. Notice that due to parameterization, given by (3.6), this reduces to just computing the derivatives with respect to β_i, and in the case of $1 - \gamma_\nu$ a negative sign appears in the derivative. The derivative of the realization of the spread $s_{n_T}^{(\mathrm{tr})}$ (2.9), restated as (3.8) in this chapter, can be computed with the quotient rule using

$$\frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[D_{n_T}^{(\mathrm{tr})}\right] = \sum_{i=1}^{n_T} \left(\frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] - \frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[L_{i-1}^{(\mathrm{tr})}\right] \right) \cdot f_i; \tag{3.34}$$

$$\frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[P_{n_T}^{(\mathrm{tr})}\right] = s_{n_T}^{(\mathrm{tr})} \cdot \sum_{i=1}^{n_T} \left(- \frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] \right) \cdot (t_i - t_{i-1}) \cdot f_i. \tag{3.35}$$

Notice that since the spread computation is undefined for $\alpha_1 = 1$, this result propagates into the quotient rule, and the spread derivative computation is also undefined when $\alpha_1 = 1$.

The computation of the gradient further reduces to

$$\frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right] = N \cdot (1-R) \cdot \frac{\partial}{\partial \gamma_\nu} E_{(\mathrm{pool})}\left[l_i^{(\mathrm{tr})}\right] =$$

$$= N \cdot (1-R) \cdot \sum_{r=1}^{K} \min\left(\frac{K \cdot S^{(\mathrm{tr})}}{1-R}, \max\left(0, r - \frac{K \cdot a^{(\mathrm{tr})}}{1-R}\right)\right) \cdot \frac{\partial}{\partial \gamma_\nu} P\left(l_i^{(\mathrm{pool})} = r\right). \tag{3.36}$$

Hence the computation is further reduced to

$$\frac{\partial}{\partial \gamma_\nu} P\left(l_i^{(\mathrm{pool})} = r\right) = \begin{cases} \frac{\partial}{\partial \beta_\iota} P\left(l_i^{(\mathrm{pool})} = r\right), & \text{if } \beta_\iota = \gamma_\nu; \\ -\frac{\partial}{\partial \beta_\iota} P\left(l_i^{(\mathrm{pool})} = r\right), & \text{if } \beta_\iota = 1 - \gamma_\nu; \\ 0, & \text{otherwise,} \end{cases} \tag{3.37}$$

where γ_ν denotes some $\gamma_\nu \in \vec{\gamma}$ and β_ι denotes some $\beta_\iota \in \vec{\beta}$. Notice that if β_ι was not used for time t_i, $1 \le i \le \iota$, or $P\left(l_i^{(\mathrm{pool})} = r\right)$ was not created recursively from a particular scenario where β_ι was used, then $\partial_{\beta_\iota} P\left(l_i^{(\mathrm{pool})} = r\right) = 0$.

Let $P_i' = \partial_{\beta_\iota} P\left(l_i^{(\mathrm{pool})} = r\right)$ and let $[A_i']_{r,m}$ denote entry $\partial_{\beta_\iota} P\left(l_{(i-1,i]}^{(\mathrm{pool}),K-m} = r - m\right)$. Then the derivative of the recursion relationship (3.17) can be written as

$$P_{i+1}' = A_{i+1}' P_i + A_{i+1} P_i'. \tag{3.38}$$

Notice that (3.38) abstracts different scenarios of dependence of P_i, P_i', A_i and A_i' on β_ι[8]. If A_i does not depend on β_ι, then $\partial_{\beta_\iota} [A_i]_{r,m} = 0$ for all $r = 0, 1, \cdots, K$ and all $m = 0, 1, \cdots, r$.

[8]In most cases, either P_i' is a zero vector, or A_{i+1}' is a zero matrix.

Up to this point, all equations have been defined for all $\alpha_i, \beta_i \in [0, 1]$, except the spread pricing equation (3.8), which was discussed in Subsection 3.2.2. In order to compute $[A_i']_{r,m}$, when A_i depends on $\beta_\imath$, we can prove that

$$c\frac{\partial}{\partial \beta_i}\int_{-\infty}^{\infty} h(x)\phi(x)dx = c\int_{-\infty}^{\infty}\frac{\partial}{\partial \beta_i}h(x)\phi(x)dx, \tag{3.39}$$

by invoking another corollary of DCT for derivatives [48] of the integrand $h(x)$ (3.13). We can then approximate the right hand side of (3.39) using some quadrature rule from Section 2.6. Recall the equivalence of Riemann and Riemann-Stieltjes integrals in (3.21), hence proving (3.39) is equivalent to proving the result using Riemann-Stieltjes integrals. Let us denote $h_{\beta_i} = \partial_{\beta_i} h(x)$. To use the DCT corollary for derivatives, we must show that

$$|h_{\beta_i}(x)| \leq G_{\beta_i}(x), \tag{3.40}$$

for all $x \in \mathcal{R}$ for some dominator $G_{\beta_i}(x)$, which does not depend on α_i or β_i. We can switch to a Riemann-Stieltjes integral, because the standard normal probability density term $\phi(x)$ (3.16) does not depend on β_i; recall the equivalence (3.21). Then

$$h_{\beta_i}(x) = \frac{1}{\sqrt{2\pi}} \cdot h_{\beta_i}^{(1)}(x) \cdot h_{\beta_i}^{(2)}(x) \cdot h_{\beta_i}^{(3)}(x), \tag{3.41}$$

where

$$h_{\beta_i}^{(1)}(x) = \exp\left(-\frac{1}{2}\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)^2\right), \tag{3.42}$$

$$h_{\beta_i}^{(2)}(x) = \frac{\beta_i \Phi^{-1}(\alpha_i) - x}{\sqrt{1-\beta_i^2}^3}, \tag{3.43}$$

$$h_{\beta_i}^{(3)}(x) = (r-m)\, p_i(x)^{r-m-1}\left(1-p_i(x)\right)^{K-r} - (K-r)\left(1-p_i(x)\right)^{K-r-1} p_i(x)^{r-m}, \tag{3.44}$$

and $p_i(x) = \Phi\left(\left(\Phi^{-1}\left(\alpha_i\right) - \beta_i \cdot x\right) / \sqrt{1-\beta_i^2}\right)$. Now we have to find an integrable dominator $G_{\beta_i}(x)$, which does not depend on α_i and β_i, and which satisfies (3.40). Consider some small real constant ξ which can be infinitely close to zero, but which can never equal zero. Let us restrict $\alpha_i \in [\xi, 1-\xi]$ and $\beta_i \in [0, 1-\xi]$. Then term $\left|h_{\beta_i}^{(2)}(x)\right|$ can be bounded in absolute sense by

$$\left|h_{\beta_i}^{(2)}(x)\right| \leq \frac{\beta_i \left|\Phi^{-1}(\alpha_i)\right| + |x|}{\sqrt{1-\beta_i^2}^3}, \tag{3.45}$$

and the bound is maximized when $\beta_i = 1 - \xi$ (denote this value by $\bar{\beta}$) and when α_i is either ξ or $1-\xi$. Without loss of generality, let us pick $\alpha_i = \xi$ and denote it by $\bar{\alpha}$. We can bound $h_{\beta_i}^{(2)}(x)$ in absolute sense, if we fix ξ. The most sensible bound that we can find for $h_{\beta_i}^{(1)}(x)$ is $\left|h_{\beta_i}^{(1)}(x)\right| \leq 1$, because even if we fix β_i and α_i, the absolute maximum of $h_{\beta_i}(x)$ depends on the interaction of $h_{\beta_i}^{(1)}(x)$ and $h_{\beta_i}^{(2)}(x)$ *for all* x, and eventually we could encounter a value $x = \Phi^{-1}(\bar{\alpha})/\bar{\beta}$, at which point the exponential term simply becomes 1. The bound on the term $h_{\beta_i}^{(3)}(x)$ is easy to compute.

We obtain the following dominator

$$G_{\beta_i}(x) = (2r + K + m)\frac{1}{\sqrt{2\pi}}\frac{\bar{\beta}\left|\Phi^{-1}(\bar{\alpha})\right| + |x|}{\sqrt{1-\bar{\beta}^2}^3}. \tag{3.46}$$

Using the fact that $\int_{-\infty}^{\infty}|x|d\Phi(x) = 2/\sqrt{2\pi}$ and the fact that Φ is the standard normal CDF, we obtain

$$\int_{-\infty}^{\infty} G_{\beta_i}(x)d\Phi(x) = (2r + K + m)\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{1-\bar{\beta}^2}^3}\left[\bar{\beta}\left|\Phi^{-1}(\bar{\alpha})\right| + \frac{2}{\sqrt{2\pi}}\right] < \infty. \tag{3.47}$$

This completes the proof that we can interchange the integral and the derivative in (3.39) for all $\alpha_i \in [\xi, 1-\xi]$ and all $\beta_i \in [0, 1-\xi]$ for arbitrarily small but non-zero ξ.

The proof techniques used for $\beta_i < 1$ do not work for $\beta_i = 1$, so we were unable to compute $[A_i']_{r,m}$ for $\beta_i \in [0,1]$. The same situation occurs for $\alpha_i \in [0,1]$. Realistically, values $\alpha_i = 0$ and $\alpha_i = 1$ are highly unlikely in practice. This is discussed at the end of this subsection.

We can determine that for all $r = 0, 1, \cdots, K$ and all $m = 0, 1, \cdots, r$

$$\lim_{\beta_i \to 0^+} h_{\beta_i}(x) = c_1 \cdot x, \tag{3.48}$$

where c_1 is just some constant. Recall that $\int_{-\infty}^{\infty} x\exp(-x^2/2)dx = 0$, and using DCT corollary again, we determine that

$$\lim_{\beta_i \to 0^+}\int_{-\infty}^{\infty} h_{\beta_i} d\Phi(x) = \int_{-\infty}^{\infty}\lim_{\beta_i \to 0^+} h_{\beta_i} d\Phi(x) = 0, \tag{3.49}$$

for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$.

The constant ξ can be arbitrarily close to 0, and at least up to floating point precision, we proved that we can interchange the derivative ∂_{β_i} with the integral $\int_{-\infty}^{\infty} h(x) d\Phi(x)$ for all $\alpha_i \in (0,1)$ and all $\beta_i \in [0,1)$. We were unable to prove anything about the closed intervals of $\alpha_i \in [0,1]$ and $\beta_i \in [0,1]$. Realistically, α_i are usually not equal to 0 or 1, because if they were, then we would know that either underlyings cannot default with probability 1, or they default with probability 1, respectively. Surely, if either case were to happen, then it would not be reasonable to create a CDO contract in the first place. The data sets used for calibration in this thesis (see Section 6.1 for the description of data sets) never produce these default probability values of α_i. It could happen that during calibration, $\beta_i = 1$, but this is very unlikely, and in any event we can switch to any gradient-free optimizer from Section 2.8 if this were to happen.

In summary, we have established that for all $\alpha_i \in (0,1)$ and all $\beta_i \in [0,1)$ the matrix entries of $A'_i = \partial_{\beta_i} [A_i]_{r,m}$ are given by

$$
\begin{aligned}
[A'_i]_{r,m} = \frac{1}{2\pi} \begin{pmatrix} K-m \\ r-m \end{pmatrix} \int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{2}\right) \exp\left(-\frac{1}{2}\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)^2\right) \\
\times \left[\frac{\beta_i \Phi^{-1}(\alpha_i) - x}{\sqrt{1-\beta_i^2}^3}\right] \\
\times \left[(r-m)\, p_i(x)^{r-m-1} (1-p_i(x))^{K-r} - (K-r)(1-p_i(x))^{K-r-1} p_i(x)^{r-m}\right] dx, \quad (3.50)
\end{aligned}
$$

where $p_i(x) = \Phi\left(\left(\Phi^{-1}(\alpha_i) - \beta_i \cdot x\right) / \sqrt{1-\beta_i^2}\right)$. Notice that this derivative is taken with respect to β_i.

The Jacobian J for the Levenberg-Marquardt optimization algorithm in Subsection 2.8.2.2 can be generated from the individual terms $\mathcal{E}_k$ (2.67) used in the objective function F, as outlined in the same subsection. None of the algorithms surveyed and implemented by the GSL library require Hessian evaluation.

As can be seen from the discussion above, calculating gradients is quite complex for this model. Calculating Hessians is even more complex. Therefore, we restricted our optimization methods to algorithms that do not require Hessians explicitly.

Parameter Base Cases The multi-path parameterization branches are started with the column vectors $P_0 = (1, 0, \cdots, 0)$, $P_0' = (0, 0, \cdots, 0)$, and $E_{(\text{pool})}\left[l_0^{(\text{tr})}\right] = 0 = \frac{\partial}{\partial \gamma_\nu} E_{(\text{pool})}\left[l_0^{(\text{tr})}\right]$ for all tr and ν, where P_0 and P_0' are column vectors.

Chapter 4

Error Analysis

Recall from Chapter 3, that recursion relationship (3.17) requires the computation of the following lower triangular matrix A_i, with entries

$$[A_i]_{r,m} = P(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m) = c \int_{-\infty}^{\infty} h(x)\phi(x)dx, \tag{4.1}$$

for $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$, where

$$c = \begin{pmatrix} K - m \\ r - m \end{pmatrix}, \tag{4.2}$$

$$h(x) = p_i(x)^{r-m} \left(1 - p_i(x)\right)^{K-r}, \tag{4.3}$$

$$p_i(x) = \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1 - \beta_i^2}}\right). \tag{4.4}$$

We proved in Chapter 3 that the computation of A_i is defined for all $\alpha_i, \beta_i \in [0, 1]$ and all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$. We derived the analytic expressions corresponding to (4.1) for $\alpha_i, \beta_i = 0, 1$. Also recall that the r-th entry of the column vector $P_i \in \mathcal{R}^{K+1}$ from Chapter 3 is given by $P\left(l_i^{(\text{pool})} = r\right)$.

Recall that the recursion relationship (2.29), restated as (3.17) in Chapter 3, can be written succinctly as

$$P_i = A_i P_{i-1} \tag{4.5}$$

using matrix-vector multiplication. However, because integrals (4.1) are intractable analytically, we must approximate them numerically, using some quadrature rule (possible quadrature rules were presented in Section 2.6). Let $\hat{A}_i$ denote the numerical approximation to A_i using some quadrature rule, and let $\hat{P}_i$ denote the resultant numerical approximation to the pool loss probability vector, given by

$$\hat{P}_i = \hat{A}_i \hat{P}_{i-1}, \tag{4.6}$$

which demonstrates that the numerical approximation error in $\hat{A}_i$ propagates into the pool loss probability vector $\hat{P}_i$.

In this chapter, we determine an integration strategy for $\hat{A}_i$, which guarantees that the error in the pool loss probability vector P_i satisfies

$$\left\| \hat{P}_i - P_i \right\|_1 \leq \text{tol} \tag{4.7}$$

for all $i = 1, 2, \cdots, n_{T_{\max}}$, where $T_{\max}$ is the maximum maturity of the CDO contract[1], for some tolerance parameter tol. We determine the error $\epsilon = \epsilon(\text{tol})$ due to the quadrature rule approximation in $\left[\hat{A}_i\right]_{r,m}$ as a function of the tolerance parameter tol, which guarantees (4.7). Let $Q\left(\chi(x); [a,b]\right)$ denote the quadrature rule approximation of the integral of a function $\chi(x)$ over some interval $[a,b]$, and let $I\left(\chi(x); [a,b]\right)$ denote the exact integral of $\chi(x)$ over $[a,b]$. Precisely, we prove that if either relationship

$$\left| I\left(c \cdot h(x)\phi(x); (-\infty, \infty)\right) - Q\left(c \cdot h(x)\phi(x); (-\infty, \infty)\right) \right| \leq \epsilon \tag{4.8}$$

holds for an open quadrature rule or relationships

$$\begin{aligned} \left| I\left(c \cdot h(x)\phi(x); (-\infty, a)\right) \right| &\leq d_1 \cdot \epsilon \\ \left| I\left(c \cdot h(x)\phi(x); [a,b]\right) - Q\left(c \cdot h(x)\phi(x); [a,b]\right) \right| &\leq d_2 \cdot \epsilon \\ \left| I\left(c \cdot h(x)\phi(x); [b, \infty)\right) \right| &\leq d_3 \cdot \epsilon \end{aligned} \tag{4.9}$$

[1] Usually the longest life span of a CDO contract is $T_{\max} = 10$ years, which is equivalent to $n_{T_{\max}} = 40$ quarterly payments.

hold for some constants $d_j \in [0,1]$, $j = 1,2,3$ and $\sum_{j=1}^{3} d_j = 1$[2], for a closed quadrature rule on $[a,b]$, then (4.7) is automatically satisfied for all $i = 1, 2, \cdots, n_{T_{\max}}$.

We derive this relationship between ϵ and tol in Subsection 4.1. We also determine the interval of integration $[a,b]$ for a closed quadrature rule. Recall that the derivatives of the probability vector P_i with respect to β_ι were denoted by P_i' in Subsection 3.2.3. We have attempted to determine a similar relationship between the quadrature error ϵ_{β_i} in the quadrature approximation to $\partial_{\beta_i} [A_i]_{r,m}$, which guarantees that the error in numerical approximation $\hat{P}_i'$ to P_i' satisfies $\left\| P_i' - \hat{P}_i' \right\|_1 \leq \text{tol}$. Unfortunately, the theoretical error bounds computed in all our attempts were too pessimistic, and did not result in a practical value of ϵ_{β_i}.

In Section 4.2 we describe which quadrature routines can be used in practice to compute $\hat{A}_i$ and we justify our choice of the Gauss-Legendre quadrature rule on $[a,b]$. Routines which guarantee the error bounds in (4.8) or (4.9) are very slow in practice, and we cannot guarantee these bounds if we want to use a faster quadrature routine. However, we develop an error control heuristic, which makes it very unlikely for (4.7) to not hold in practice. We also suggest an integration strategy to approximate pool loss probability derivative vectors P_i', and demonstrate with numerical results that our error control heuristic is very likely to produce errors which are a few orders of magnitude smaller than required.

As a side note, we have attempted a number of changes of variables in (4.1) to undo the step function behavior of $p_i(x)$ in the limit as $\beta_i \to 1^-$, but this did not result in any usable bounds on the errors of quadrature approximations, and we were unable to reduce the number of quadrature points needed to satisfy the requirements of our error control heuristic.

[2] A natural choice is $d_1 = d_3 = 1/4$ and $d_2 = 1/2$.

4.1 Pool Loss Probability Error

In this section, we derive the aforementioned error control strategy for the recursion relationship (4.5), which guarantees that (4.7) is satisfied, as long as either (4.8) or (4.9) is satisfied for an appropriately chosen ϵ.

Define ϵ_i to be the maximum absolute error in integral approximations (4.8) or (4.9) at time t_i for all $r = 0, 1, \cdots, K$ and all $m = 0, 1, \cdots, r$ and let δ_i be the maximum absolute error of the column sums of the matrix $\hat{A}_i$ at time t_i ($\epsilon_i, \delta_i \in \mathcal{R}^+$). Notice that while β_i is held constant during each period, α_i changes in value at every time step t_i. We can further bound the quadrature error by letting

$$\epsilon = \max_{i \in \{1,2,\cdots,n_{T_{\max}}\}} \epsilon_i, \tag{4.10}$$

where $T_{\max}$ is the longest maturity of a CDO that we are using in the calibration; we could have a single branch which spans the entire time frame, i.e. 10 years with quarterly payments create 40 quadrature locations.

We can also bound

$$\delta_i \leq (K+1)\epsilon_i \leq (K+1)\epsilon, \tag{4.11}$$

which accounts for making maximum error in the same direction every time. So the error bound becomes

$$\begin{aligned} \left\| P_{i+1} - \hat{P}_{i+1} \right\|_1 &= \left\| A_{i+1} P_i - \hat{A}_{i+1} \hat{P}_i \right\|_1 = \left\| A_{i+1} P_i - \hat{A}_{i+1} P_i + \hat{A}_{i+1} P_i - \hat{A}_{i+1} \hat{P}_i \right\|_1 \\ &\leq \left\| A_{i+1} - \hat{A}_{i+1} \right\|_1 \|P_i\|_1 + \left\| \hat{A}_{i+1} \right\|_1 \left\| P_i - \hat{P}_i \right\|_1 \leq \delta_{i+1} + (1+\delta_{i+1}) \left\| P_i - \hat{P}_i \right\|_1 . \end{aligned} \tag{4.12}$$

Let us further denote $Y = (K+1)\epsilon$ and then the error bound above becomes

$$\left\| P_{i+1} - \hat{P}_{i+1} \right\|_1 \leq Y + (1+Y) \left\| P_i - \hat{P}_i \right\|_1 . \tag{4.13}$$

The initial error is $\left\| P_0 - \hat{P}_0 \right\|_1 = 0$, because we know the pool loss probability vector exactly at time t_0. This together with (4.13) implies that

$$\left\| P_{n_{T_{\max}}} - \hat{P}_{n_{T_{\max}}} \right\|_1 \leq Y \sum_{j=0}^{n_{T_{\max}}-1} \left[(1+Y)^j \right] = (1+Y)^{n_{T_{\max}}} - 1. \tag{4.14}$$

We can use (4.14) to find ϵ, the bound on the errors in the integral approximations $Q\left(c \cdot h(x)\phi(x); (-\infty, \infty)\right)$ or $Q\left(c \cdot h(x)\phi(x); [a, b]\right)$ that will ensure that $\left\| P_{n_{T_{\max}}} - \hat{P}_{n_{T_{\max}}} \right\|_1 \leq$ tol, for some appropriate tolerance tol. For example, for tol $= 10^{-8}$, $K = 125$ and $n_{T_{\max}} = 40$, a simple calculation shows that $\epsilon \leq 2 \cdot 10^{-12}$ suffices.

For quadrature rules on a finite interval $[a, b]$, we can bound the errors due to interval truncation for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$ by

$$\left| c \int_{-\infty}^{a} h(x)\phi(x)dx \right| \leq c_{\max} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} \exp\left(-x^2/2\right) dx \leq \epsilon/4 \tag{4.15}$$

$$\left| c \int_{b}^{\infty} h(x)\phi(x)dx \right| \leq c_{\max} \frac{1}{\sqrt{2\pi}} \int_{b}^{\infty} \exp\left(-x^2/2\right) dx \leq \epsilon/4, \tag{4.16}$$

where

$$c_{\max} = \left(\begin{array}{c} K \\ \lfloor K/2 \rfloor \end{array} \right), \tag{4.17}$$

for a natural choice of $d_1 = d_3 = 1/4$ and $d_2 = 1/2$ in (4.9). For this choice of d_j, $j = 1, 2, 3$, notice that because the function $\exp\left(-x^2/2\right)$ is symmetric about the origin, we can set $a = -b$. The value of a which satisfies (4.15) is given by

$$a \leq \Phi^{-1}\left(\frac{4}{\epsilon \cdot c_{\max}}\right), \tag{4.18}$$

where Φ^{-1} denotes the inverse of the standard normal Cumulative Density Function (CDF). For the aforementioned computation of $\epsilon \leq 2 \cdot 10^{-12}$, we determine that $a \leq -13.099507$ by solving (4.18).

Hence, we have determined that if we can guarantee (4.8) for an open quadrature rule, then for the entire duration of the CDO contract, (4.7) holds. We found the intervals of integration for (4.9) for a natural choice of error constants d_j, $j = 1, 2, 3$, and if we can guarantee that on this pre-determined interval of integration $[a, b]$, relationship

$$\left| I\left(c \cdot h(x)\phi(x); [a, b]\right) - Q\left(c \cdot h(x)\phi(x); [a, b]\right) \right| \leq \epsilon/2 \tag{4.19}$$

holds for some closed quadrature rule Q, then (4.7) also holds. The next subsection addresses the practicality of these theoretical results.

4.2 Error Control Strategy

Quadrature routines which guarantee (4.8) or (4.19) are too slow for our applications. To be as efficient as possible, we would like to use a pre-generated set of quadrature nodes x_j and weights w_j (for closed interval quadrature rules, the interval of integration $[a, b]$ is pre-determined from Section 4.1). We were unable to compute analytic error bounds, developed for such quadrature rules, because we were unable to determine closed form error equations for the integrand $h(x)$ (4.3) for more than a few quadrature nodes. Instead of guaranteeing (4.8) or (4.19), we develop an error control heuristic in this section which in practice results in very small errors in the pool loss probability vector P_i, because the error analysis derivation placed very pessimistic error bounds in (4.12).

Notice that the bound (4.18) does not depend on time t_i. In an attempt to satisfy (4.14), we can check that

$$\left| \sum_{j=0}^{K} \left[\hat{A}_i\right]_{j,m} - 1 \right| \leq (K+1)\epsilon, \tag{4.20}$$

after computing each column $m = 0, 1, \cdots, K$ in the matrix $\hat{A}_i$. If the bound (4.20) is satisfied, we move to the next column, otherwise we double the number of quadrature nodes and weights in a particular quadrature rule, and repeat the computation.

This error control heuristic does not guarantee (4.14), because

$$\begin{aligned} \left| \sum_{j=0}^{K} \left[\hat{A}_i\right]_{j,m} - 1 \right| &\leq \left\| \hat{A}_i - A_i \right\|_1, \\ \left| \sum_{j=0}^{K} \left[\hat{A}_i\right]_{j,m} \right| &\leq \left\| \hat{A}_i \right\|_1, \end{aligned} \tag{4.21}$$

and the error bounds (4.12) do not necessarily hold. However, the above inequalities (4.21) are unlikely to be very different in practice, since the entries of $\hat{A}_i$ are all positive (for our later choice of the quadrature rule, all weights are positive), and we are integrating a positive function $h(x)$. Also, the bounds developed from (4.12) are highly pessimistic, and we suggest, using the numerical results discussed at the end of this subsection, that the error in practice in (4.7) is a few orders of magnitude less than tol.

Recall from Chapter 3 that the derivative of the integral (4.1) with respect to β_i, $\partial_{\beta_i} P(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m)$ is given by (3.50), using the integrand

$$h_{\beta_i}(x) = \exp\left(-\frac{1}{2}\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}}\right)^2\right) \times \left[\frac{\beta_i \Phi^{-1}(\alpha_i) - x}{\sqrt{1-\beta_i^2}^3}\right] \\ \times \left[(r-m)\, p_i(x)^{r-m-1}\,(1-p_i(x))^{K-r} - (K-r)\,(1-p_i(x))^{K-r-1}\, p_i(x)^{r-m}\right]. \quad (4.22)$$

As mentioned previously, we used a similar error bound strategy to (4.12) in our attempts to determine an ϵ_{β_i}, which guarantees that

$$\left\| \hat{P}_i' - P_i' \right\|_1 \leq \text{tol} \quad (4.23)$$

is satisfied for any $i = 1, 2, \cdots, n_{T_{\max}}$. Our attempts did not result in a practical value of ϵ_{β_i}, because of extremely pessimistic error bounds in this case. We were unable to form tighter error bounds. However, notice that (4.22) contains two decaying exponential terms and one polynomial term. The same standard normal probability density term $\exp(-x^2/2)$ which decays the integrand $h(x)$ (4.3) when $|x|$ is large, also decays $h_{\beta_i}(x)$. These similarities between $h(x)$ and $h_{\beta_i}(x)$ suggest that it is likely that both integrands require about the same number of quadrature points (and for closed interval quadrature rules, on the same interval $[a, b]$). Now recall from Section 3.1, that the β_i's follow various scenario paths. For each time t_i and each scenario value of β_i, after we finish integrating all entries of A_i for some i, we can re-use the same number of quadrature points for the computation of A_i' (3.50), as dictated by (4.20).

We must now consider quadrature rules and routines for the computation of $\hat{A}_i$ and $\hat{A}_i'$. For a single scenario of β_i's, the CDO contract usually has up to 40 quarterly payments. Matrix A_i has $(K+1)(K+2)/2$ elements. So, for $K = 125$, we must compute the integral (4.1) 320040 times for just a single scenario of β_i's[3]. In addition we must compute the same number of integral derivatives (3.50), not to mention other

[3]There are at least 2 scenarios in our multi-path parameterization.

derivative data structures from Subsection 3.2.3. Hence our quadrature routine must use as few quadrature points as possible to guarantee (4.20) and it must be as fast as possible. Our preliminary numerical tests showed that adaptive integration on $(-\infty, \infty)$ is too expensive, and this would make the computation of A_i too inefficient. Adaptive integration of this kind, and Gauss-Laguerre on $[0, \infty)$, map the interval of integration to $(0, 1]$ and those maps usually result in singularities at 0, which are then dampened [16].

Gauss-Hermite quadrature suits the integrand $h(x)$ (4.3) best and requires fewer nodes and weights, because of the Gaussian probability density term. However, we run into the problem of generating a sufficient number of nodes and weights: the algorithm given in [15] is poorly conditioned for more than 100 nodes and an alternative algorithm provided by [30] suffers the curse of dimensionality, as an internal data structure does not allow us to generate a significant number of quadrature points [31]. It could happen that the error control heuristic (4.20) requires a high number of quadrature nodes. This is not likely in practice, however, and one could use Gauss-Hermite quadrature for this problem.

We compared Gauss-Chebyshev and Gauss-Legendre rules and determined that the latter rule produces the same quadrature error in (4.20) with a fewer number of nodes. Hence, we decided to use Gauss-Legendre as our quadrature rule, however as mentioned previously, other quadrature rules and routines can also be used.

We now demonstrate that when the error heuristic (4.20) is being used together with our error control strategy for derivatives,

$$\max_{i=1,2,\cdots,n_{T_{\max}}} \left| \sum_{j=0}^{K} \left[\hat{P}_i \right]_j - 1 \right| \leq \text{tol} \qquad \text{and} \qquad \max_{i=1,2,\cdots,n_{T_{\max}}} \left| \sum_{j=0}^{K} \left[\hat{P}_i' \right]_j \right| \leq \text{tol}, \tag{4.24}$$

for a realistic choice of α_i's and for multiple values of β, where each $\beta_i = \beta$ for all $i = 1, 2, \cdots, n_{T_{\max}}$. This suggests that it is very likely that both (4.14) and (4.23) are satisfied in practice. Table 4.1 on page 66 quotes values of (4.24) for various values of β, using α_i's from the first day of the CDX NA IG S8 data set, which is described later in Section 6.1 and is ultimately used for MSCM calibration in Chapter 6. Other values of α_i's produce similar results, so we only quoted the results for one particular setting of

α_i's. We can see that elements of $\hat{P}_i$ add up to a value very close to 1 and elements of derivative vector $\hat{P}_i'$ sum to a value even closer to 0. Values of $\hat{P}_i'$ accumulate a negligible error for $\beta_i = 1 - 10^{-5}$. We verified that elements of $\hat{P}_i'$ sum to 0 for all values of β_i in the neighborhood of $\beta_i = 1 - 10^{-5}$ with the same magnitude of error on the order of 10^{-12}.

β	$\max_{i=1,2,\cdots,n_{T_{\max}}} \left\lvert \sum_{j=0}^{K} \left[\hat{P}_i\right]_j - 1 \right\rvert$	$\max_{i=1,2,\cdots,n_{T_{\max}}} \left\lvert \sum_{j=0}^{K} \left[\hat{P}_i'\right]_j \right\rvert$
10^{-16}	$1.8540724511e-14$	$8.2205037147e-17$
10^{-15}	$1.8207657604e-14$	$1.9036418513e-16$
10^{-10}	$3.1752378504e-14$	$6.5829436586e-16$
10^{-7}	$3.3084646134e-14$	$2.9336517232e-16$
10^{-5}	$1.7319479184e-14$	$2.2885867635e-16$
10^{-2}	$2.0095036746e-14$	$2.3135653416e-16$
0.1	$1.2212453271e-14$	$5.8443449842e-16$
0.2	$2.5535129566e-14$	$1.0496920693e-15$
0.3	$2.9976021665e-14$	$6.8283821508e-16$
0.4	$2.3425705820e-14$	$1.6223643494e-15$
0.5	$2.1316282073e-14$	$2.3904273990e-15$
0.6	$1.2878587086e-14$	$1.6555792431e-15$
0.7	$2.4868995752e-14$	$2.9262500168e-15$
0.8	$9.4368957093e-15$	$2.8840184364e-15$
0.9	$3.0642155480e-14$	$1.2623582735e-14$
$1-10^{-2}$	$1.0880185641e-14$	$4.8572257327e-14$
$1-10^{-5}$	$1.3544720900e-14$	$4.7617681316e-12$
$1-10^{-7}$	$1.3322676296e-14$	$1.5407439555e-31$
$1-10^{-10}$	$1.3322676296e-14$	0
$1-10^{-15}$	$1.3322676296e-14$	0
$1-10^{-16}$	$1.3322676296e-14$	0

Table 4.1: Errors defined by (4.24) for the default probabilities α_i from the first day of the CDX NA IG S8 data set, discussed in Chapter 6, for various settings of the copula correlation parameters $\beta_i = \beta$ for all $i = 1, 2, \cdots, n_{T_{\max}}$.

Chapter 5

Code Implementation

Calibrating the Multi-period Single-factor Copula Model (MSCM) is very computationally demanding. The goal is to be able to solve the optimization problem (3.4) in an efficient manner using some optimization algorithm from Section 2.8. As outlined in Subsection 3.2.1, the MSCM calibration process possesses many stages which can be computed in parallel, thus improving the efficiency of evaluation of the objective function (2.52) and its first-order derivatives, described in Section 3.2. In addition, the MSCM calibration process has many complicated data structures, which have to be handled efficiently.

We implement the MSCM calibration process in C++, using Boost [26] libraries for data structures, GNU Scientific Library (GSL) [16] for optimization routines and OpenMP [27] for parallelization. Matlab was used to generate plots and to parse CDO data sets (originally available in Microsoft Excel) into text files, from which the model loads the data. Thread safety was guaranteed using Valgrind's thread checker Helgrind.

5.1 Lazy Computation With Boost C++ Libraries For Vectorized Quadrature

The most expensive procedure in the MSCM calibration process is the initialization of the lower triangular matrices A_i (2.43) and A'_i (3.50). Therefore, the computation of A_i and A'_i has to be efficient. For example, the quadrature rule Q in Chapter 4 requires the computation of a sum of a product of weights and function values at quadrature nodes. This must be done $(K+1)(K+2)/2 = 8001$ times for $K = 125$ for each A_i or A'_i.

Instead of looping over nodes and weights, we can use a function object, also known as a *functor* [32], which performs like a function when called on an object. For example, a matrix can be stored as a contingent array in memory, or we can create an object which behaves like an array of dimension 2, but has the added advantage of memory management during compilation [26]. The compiler can then selectively manage memory as it becomes needed, hence the term "lazy computation". Then we can define, for example, a multiplication functor: another object which multiplies two matrix objects. There is also an added benefit of code readability.

Consider a Gauss-Legendre quadrature weight vector $\vec{w} \in \mathcal{R}^{n_{\mathrm{GL}}}$ and a vector of nodes $\vec{x} \in \mathcal{R}^{n_{\mathrm{GL}}}$ for some $n_{\mathrm{GL}} \in \mathcal{Z}^+$. Recall from Section 2.6 that we can perform Gauss-Legendre quadrature on an arbitrary interval $[a, b]$ for some function $\chi(x)$[1] using

$$\int_a^b \chi(x)dx \approx \frac{b-a}{2} \sum_{j=1}^{n_{\mathrm{GL}}} w_j \cdot \chi\left(\frac{b-a}{2}x_j + \frac{a+b}{2}\right), \tag{5.1}$$

where w_j is the Gauss-Legendre quadrature weight for the interval $[a, b]$. Algorithms 6 and 7 demonstrate two ways of performing quadrature (5.1) in C++. We believe that the implementation with Boost is more readable. The performance depends on the compiler,

[1] For example, in (4.1) we set $\chi(x) = c \cdot h(x)\phi(x)$, where $c = \begin{pmatrix} K-m \\ r-m \end{pmatrix}$, $h(x) = p_i(x)^{r-m}\left(1-p_i(x)\right)^{K-r}$, $p_i(x) = \Phi\left(\left(\Phi^{-1}\left(\alpha_i\right) - \beta_i x\right)/\sqrt{1-\beta_i^2}\right)$ and $\phi(x)$ is the standard normal probability density.

Algorithm 6 C++ implementation of a quadrature sum using GSL.

```
const unsigned n_GL = 64;  // number of Gauss-Legendre points
gsl_vector *x = gsl_vector_alloc(n_GL);  // allocate vector memory
gsl_vector *w = gsl_vector_alloc(n_GL);  // allocate vector memory
double s = 0; // sum accumulator
// Initialize vectors with Gauss-Legendre nodes and weights on [a,b]
init_GL(x,w); // nodes and weights are scaled for [a,b] in init_GL
// Perform quadrature
for (unsigned j = 0; j < n_GL; ++j)
  s += gsl_vector_get(w, j)*chi(gsl_vector_get(x,j));
gsl_vector_free(x);  // free vector memory
gsl_vector_free(w);  // free vector memory
```

compiler optimization flags, operating system and the actual hardware used.

5.2 OpenMP Parallel Implementation

The C++ implementation of the MSCM calibration process has several parallel regions, as well as nested parallel regions. In practice, some of these regions need to be disabled, because the overhead in thread creation nullifies the performance gain. There is also an added aspect of thread safety when using an *omp_set_nested*() library call. There are also three different thread schedulers available in OpenMP 3.0 [27].

The following is a description of each region, which can be computed in parallel. In practice, too many parallel regions increase the execution time, due to the overhead in thread creation, coordination and termination. In practice, to improve the effect of parallelization, we need to disable some of the following parallel regions:

1. Parallel integration of matrices $\hat{A}_i$ (2.43) for each multi-path branch, i.e. paral-

Algorithm 7 C++ implementation of a quadrature sum using Boost.

```
using namespace boost::numeric::ublas;
const unsigned n_GL = 64;  // number of Gauss-Legendre points
vector<double> x(n_GL), w(n_GL);  // invoke vector object constructors
double s = 0; // sum accumulator
// Initialize vectors with Gauss-Legendre nodes and weights on [a,b]
init_GL(x,w); // nodes and weights are scaled for [a,b] in init_GL
// Perform quadrature
s = prec_inner_prod(w, apply_to_all<functor::chi<double> > (x));
// garbage collection is handled automatically by each vector object,
// so no need to remember to deallocate memory with Boost
```

lelization of multi-path branches.

2. Computation of entries of $\hat{A}_i$ and optionally $\hat{A}'_i$. Rows of $\hat{A}_i$ for each column and then elements of $\hat{A}'_i$ can be computed in parallel.

3. If the first period has ν paths in the multi-path parameterization, then we can create ν parallel processes for the computation of $E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right]$ (2.16).

4. We can recursively nest the paths from the previous step for subsequent periods. For example, if there are μ multi-path periods with ν branches per period, then the last period will have $\nu^{\mu-1}$ paths computing $E_{(\mathrm{pool})}\left[L_i^{(\mathrm{tr})}\right]$ (2.16) in parallel.

5. We can compute the nested error function loops in the unconstrained objective function $F(\vec{u})$ (3.5) in parallel.

6. Similarly to the previous item, we can compute entries of the Levenberg Marquardt vector $\vec{\mathcal{E}}$ from Paragraph 2.8.2.2 in parallel.

7. Similarly to item 5 we can compute derivatives of the unconstrained objective

function $F(\vec{u})$ (3.5) by parallelizing the nested loops in (3.1).

8. We can compute the gradient and the Jacobian of the unconstrained objective function $F(\vec{u})$ (3.5) in parallel.

In practice, only items 2, 5, 6, 7 and 8 need to be enabled. The adjustment of the above model performance parameters and all numerical results have been performed on a system with two Intel Xeon E5355 quad core CPUs (maximum of eight parallel threads) with 4MB of CPU cache per CPU. Figure 5.1 on the following page depicts the speedup factor when performing the computation of the objective function and its derivatives for all 6 model parameterizations, described later in detail in Section 6.2. These are average speedup factors when computing the objective function and its derivatives for the first day of each of the four data sets used in Chapter 6 with the MSCM parameterizations, which are later used in the numerical results in Chapter 6.

The overhead in thread creation is evident in Figure 5.1 on the next page. For example, the parameterization with a single period spanning 10 years with 4 paths per period parallelizes best when executed with 4 parallel threads. However, the speedup factor decreases when the same parameterization is executed with 5 parallel threads. This is because the OpenMP scheduler attempts to schedule 4 parallel processes over 5 threads, and time is lost in copying the data between processes. In general, if the program has an even number of parallel regions, then executing them over an odd number of threads decreases performance. Figure 5.1 on the following page shows that different parallelizations require a different minimum number of parallel threads. For example, a parameterization with a single period spanning 10 years with only 2 paths per period requires only 2 parallel threads. However, all parameterizations parallelize well on average when presented with the maximum number of parallel threads, and this is the implementation that we've used in the numerical results, presented in the next chapter.

When checking the parallel implementation with Valgrind's thread safety detector Helgrind, thread safety using $omp\_set\_nested()$ was not guaranteed, and any nesting

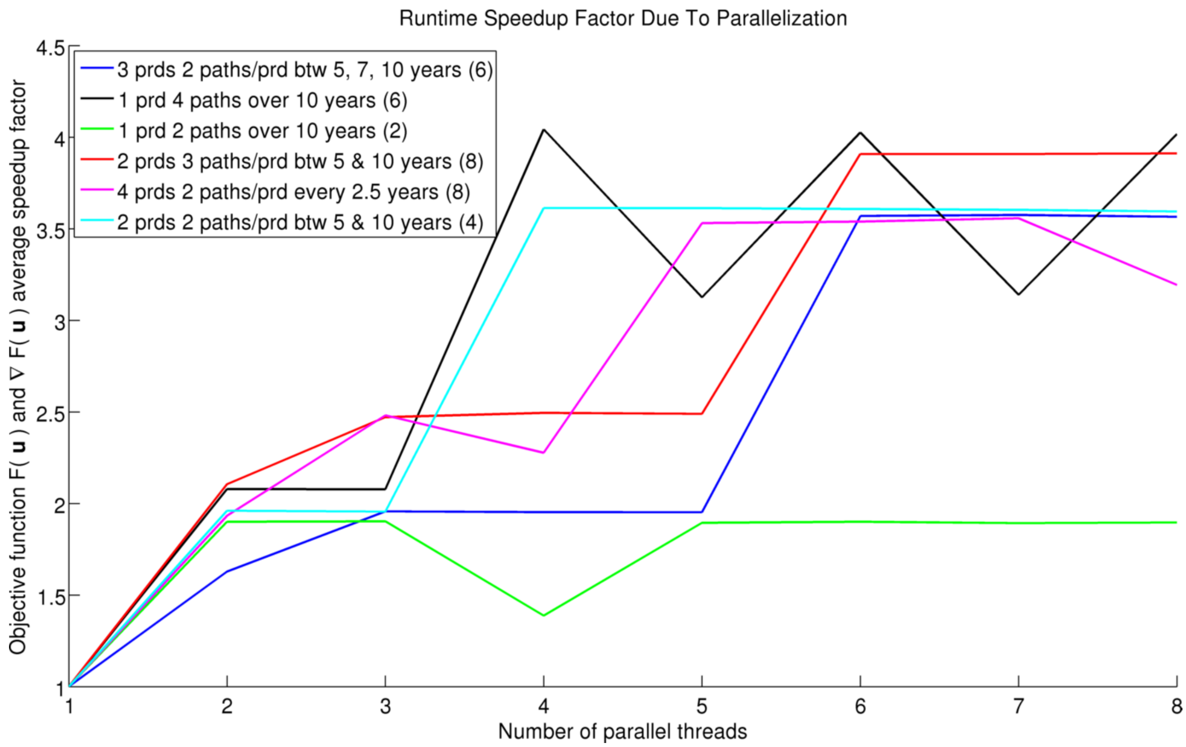

Figure 5.1: Speeedup factors when computing the objective function $F(\vec{u})$ (3.5) and its gradient with respect to the unconstrained parameters in $\vec{u}$. The six different model parameterizations are described in Section 6.2. The number of parameters which each parameterization set is given in brackets. For example, $\vec{u}$ contains 4 elements for the 2 period 2 paths per period multi-path parameterization.

had to be disabled. Also, by trial and error, we found that the fastest scheduler is *static* and runtimes improved slightly after disabling the *omp_set_dynamic*() library call.

5.2.1 Pricing In Parallel

As mentioned in Subsection 3.2.1, we can integrate matrices A_i and A'_i for $i = 1, 2, \cdots, n_{T_{\max}}$ in parallel. We need to price the model without recomputing values of $P(l_i^{(\text{pool})} = r)$ and $E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$; when computing the derivatives of the objective function (2.52) we need to do the same for $\partial_{\gamma_\nu} P(l_i^{(\text{pool})} = r)$ and $\partial_{\gamma_\nu} E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$. This task is further complicated by parallelization, i.e. race conditions have to be avoided in parallel data structures. These problems can be solved by recursively generating the pricing scenarios and reusing computed values of $P\left(l_i^{(\text{pool})} = r\right)$ and $E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$. These values would be re-used during the computation of the expected spread $E_{\vec{\rho}}\left[s_{n_T}^{(\text{tr})}(\vec{\beta}, \vec{\alpha})|\vec{\gamma}\right]$ (2.37) when computing model quotes (the same $E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$ are needed when computing spread quotes $s_{n_T}^{(\text{tr})}$ (2.9), for example, the spread quotes for any tranche with maturities of 5 and 7 years use the same $E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$ for $i = 1, 2, \cdots, 20$ with quarterly payments). When computing derivatives, we can similarly store and re-use $\partial_{\gamma_\nu} P\left(l_i^{(\text{pool})} = r\right)$ and $\partial_{\gamma_\nu} E_{(\text{pool})}\left[l_i^{(\text{tr})}\right]$ when computing $\partial_{\gamma_\nu} E_{\vec{\rho}}\left[s_{n_T}^{(\text{tr})}(\vec{\beta}, \vec{\alpha})|\vec{\gamma}\right]$.

5.3 Correctness During Development

We also have to make sure that the MSCM calibration process produces valid results. Below is a list of things that we check when testing the validity of the results:

- $\left|\left\|\hat{P}_i\right\|_1 - 1\right| \leq \text{tol}$ for each setting of β_i after the computation;
- $E\left[l_{i-1}^{(\text{tr})}\right] \leq E\left[l_i^{(\text{tr})}\right]$, but to accommodate numerical errors we actually check $\hat{E}\left[l_{i-1}^{(\text{tr})}\right] - \hat{E}\left[l_i^{(\text{tr})}\right] \leq \text{tol}$ for each scenario of β_i;
- gradient and Jacobian entries are checked for certain test values of $\vec{\psi}$ with a separate routine using forward finite differencing[2] to produce a relative error of $\approx 10^{-7}$.

[2]For some settings of γ_j, the forward finite difference approximation would be very different to the analytic answer. For example, for a parameterization using a single period with two paths, $\gamma_1 = \frac{1}{2} = \rho_1$ would produce $\beta_i = 1/2$ with probability $\rho_i = 1/2$ in both paths. We know that the derivative is zero (local minimum), but numerically $\sum_{\text{tr}} \sum_T \partial_{\gamma_j} \text{error}(E_{\vec{\rho}}\left[s_{n_T}^{(\text{tr})}(\vec{\beta}, \vec{\alpha})|\vec{\gamma}\right], m_T^{(\text{tr})}) \approx 40 \neq 0$, for a forward step size of 10^{-3} due to the accumulation of numerical errors introduced by the finite difference approximation.

5.4 Error Control

We employ the quadrature error control heuristic (4.20) and quadrature strategy from Subsection 4.2. We also check that $\left|1 - \sum_{j=0}^{K} \left[\hat{P}_i\right]_j\right| \leq \mathrm{tol}$ and $\left|\sum_{j=0}^{K} \left[\hat{P}_i'\right]_j\right| \leq \mathrm{tol}$ after each integration time step.

Chapter 6

Numerical Results

We present the Multi-period Single-factor Copula Model (MSCM) calibration results in this chapter. Specifically, we calibrate the model on the first day of the four data sets, described in Section 6.1 with different parameterizations, presented in Section 6.2, using a variety of unconstrained optimization algorithms, previously described in Section 2.8. We determine that on average, the most robust and efficient derivative-free unconstrained optimization algorithm is NEWUOA (presented in Subsubsection 2.8.1.2), and the most robust and efficient derivative-based unconstrained optimization algorithm is BFGS2 (presented in Paragraph 2.8.2.3). We then present and discuss calibration results and runtimes for daily data for each of the four aforementioned data sets, for the periods before, during and after the September 2008 stock market crash. We also justify our choice of the relative Soft Error Function (SEF) (2.61) in the unconstrained objective function $F(\vec{u})$ (3.5).

6.1 Data Sets Used For Calibration

CDO and CDS quotes were obtained from the Reuters Thomson data stream[1], which provides active trading data as long as part of the CDO data is still needed by either investors or sellers. The data stream does not provide historical prices; only most recent CDO data is available, although as of July 2010 some data sets extend back in time as far as December 2008. CDX NA IG series 8 (S8) data set was previously acquired by Dr. Wanhe Zhang from the same data stream, but was no longer available from the Reuters Thomson data stream in July 2010.

The credit crunch in July 2007 and the crash of September 2008 disrupted data available for calibration in a severe way. Figure A.4 shows the CDX NA IG S8 data set, available from March 2007 until December 2008 and partitioned into three time frames across the rows of the figure. Plots on the left show raw CDS quotes with maturities at 3, 5, 7 and 10 years, and plots on the right show the bootstrapped default probabilities. We can see that the CDS quotes change drastically and this affects the shape of default probabilities: as time progresses, default probabilities become linear with respect to time, loosing their convex shape, and are no longer monotonically increasing with respect to time.

Figure A.5 shows the CDX NA IG S10 & S11 and CMA ITRAXX EU S10 data sets, which were available after the crash of 2008; only CDS data which resulted in monotonically increasing default probabilities was plotted. For the first two data sets, the default probabilities are concave in shape, while the third (European) data set has almost linear default probabilities (just at the point of changing convexity).

Convex default probabilities indicate that we expect more defaults to happen at a

[1] University of Toronto Rotman School of Business provides free access to students to otherwise proprietary CDO market quotes. Reuters Thomson data stream supplies the same quotes as Bloomberg, but also has the added functionality of collectively pooling multiple tickers across large time ranges into a Microsoft Excel spreadsheet; the Bloomberg system only provides individual quotes for a single ticker on a specific date.

later date, because currently the economy/market is stable. Concave shapes indicate the opposite, that we are expecting the defaults to happen sooner rather than later, so it is not unreasonable for the concavity to change after the crash of 2008, due to market instability.

We should also mention that the data available from the Thomson Reuters data stream had many missing values, and that the raw data had to be severely reduced until we could find contingent segments without missing data across all tranches and all maturities. Table A.1 provides the summary of each data set obtained; there were no other usable CDO series in the Reuters Thomson data stream at the time of data collection.

6.2 Calibration Results

First, we consider which optimization algorithms from Section 2.8 suit the optimization problem (3.4), restated for convenience below:

$$\min_{\vec{u} \in \mathcal{R}^{n_\psi}} F(\vec{u}). \tag{6.1}$$

We explore the following MSCM parameterizations[2]:

1. three periods between 5, 7 and 10 years, with two possible paths per period (6 parameters);
2. single period over 10 years, with four possible paths (6 parameters);
3. single period over 10 years, with two possible paths (2 parameters);
4. two periods between 5 and 10 years, with three possible paths per period (8 parameters);

[2]We are restricted in the range of multi-path parameterizations that we can explore, because the lowest number of CDO quotes per data set is 12

5. four periods between 2.5, 5, 7.5 and 10 years, with two possible paths per period (8 parameters);

6. two periods between 5 and 10 years, with two possible paths per period (4 parameters).

For the first day of each data set, we pick one of the above MSCM parameterizations and plot the unconstrained objective function $F(\vec{u})$ values versus runtime for each optimization algorithm from Section 2.8. The unconstrained objective function uses a relative Soft Error Function (SEF) (2.61) with $\delta = 0.5$ and $\epsilon = 10^{-4}$. All algorithms were executed with the same starting guess of $\vec{\psi} = (-1, -1, \cdots, -1)$[3]. Derivative-free methods were executed for 500 iterations[4] and derivative-based algorithms were executed for 40 iterations, unless the algorithms converged before the number of iterations was exceeded. Figures A.6 to A.29 on pages 102–125 show these results. We can conclude that on average the most robust and efficient derivative-free algorithm is NEWUOA. The most robust and efficient derivative-based algorithm is BFGS2.

It was difficult to specify convergence criteria for each algorithm, as these differ with each day of the data set and for each MSCM parameterization: most of the time the algorithms either don't realize that they've converged to the local minimum, or they terminate prematurely. Also, it is difficult for the algorithms to avoid local minima: sometimes different algorithms find different local minima. The objective function is not necessarily convex, as seen in the argument given in Footnote 2 on page 73: a certain setting of parameters can produce a zero gradient, but this would not necessarily produce quotes which match the market data. Numerical results shows that it is also highly unlikely for the derivative-based method to encounter a value of $\beta_i = 1$, as discussed in

[3]To provide a starting guess, we need to calibrate at least a single-period single-factor copula model, which is significantly more expensive than a single step of any optimization algorithm, the latter resulting in a good starting guess.

[4]A single iteration can perform more than one function evaluation. The number of function evaluations per iteration is algorithm-specific, so the runtimes vary slightly between different algorithms.

Subsection 3.2.3. Multiple paths per period result in longer calibration times, and make MSCM less parsimonious, but do not considerably reduce the objective function (2.52). On the other hand, addition of periods significantly decreases the objective function, but also makes the parameterization less parsimonious.

6.2.1 Daily Calibration Results

We pick the parameterization with two periods between 5 and 10 years, with two paths per period (4 parameters, referred to for brevity as the two-period two-path parameterization), and calibrate MSCM on daily data using both NEWUOA (limited to 500 iterations) and BFGS2 (limited to 40 iterations) optimization algorithms, with the aforementioned relative SEF in $F(\vec{u})$. These results are presented in Figure 6.1 and Figure 6.2. To demonstrate the dynamics of MSCM, we also calibrate the four period parameterization of MSCM with two paths per period every 2.5 years (8 parameters, referred to for brevity as the four-period two-path parameterization) on daily data with a BFGS2 algorithm, this time limited to 120 iterations. These daily results are presented in Figure 6.3. Average runtimes are presented in Tables 6.1, 6.2 and 6.3. For reference, we also include the the same daily calibration results for the industry-standard single-period single-factor copula model [1], referred to as the Hull Copula in Figure 6.4.

We want to place the objective function values of all four CDO data sets on the same plot, and because data sets all have a different number of CDO quotes, the value of $F(\vec{u})$ would be higher in data sets with a higher number of CDO quotes. Therefore, we plot $F(\vec{u})$ per number of data points in all daily calibration result figures.

We should also note that we chose the two-period two-path and four-period two-path parameterizations because they are the most intuitive to understand. Normally, we would choose the parameterization based on some market insight. For example, if we are expecting a volatile market between 2.5 to 5 years, then we would place more paths between 2.5 and 5 years. In the industry, we would pick a different parameterization for each day based on market insight.

We can see that for the two-period two-path parameterization, NEWUOA has more variability in the objective function values compared to the BFGS2, and the algorithm never converged and was terminated after 500 iterations[5]. On many days, NEWUOA produces similar objective function values to BFGS2 algorithm. The BFGS2 algorithm has less variability in the objective function values and detects convergence for some days. As hypothesized in Subsection 3.2.3, it is also very unlikely to encounter the value $\beta_i = 1$, and we have an efficient (see Table 6.2 for runtimes) and robust NEWUOA algorithm should the value of $\beta_i = 1$ occur in practice.

From all four daily calibration result figures, we can see that the credit crunch of July 2007 affected the CDX NA IG S8 data set. Just before the crash in September 2008, the data is unusable due to monotonically decreasing default probabilities (a gap in the data). Right before the crash of 2008, the CDO quotes tend to stay the same over time. Calibrating on CDO data after the crash shows that the model is no longer applicable, however later in 2009 the quotes start to return to pre-crash status and this results in lower error function values.

Calibrated parameter values and CDO quotes produced by MSCM for the two-period two-path parameterization are given in Tables A.2 to A.7 on pages 128–133 for the first day of each data set[6]. We see that only calibration with CDX NA IG S8 data set produced meaningful copula correlation parameter values. While the MSCM matched the CDO quotes of all CDO data sets reasonably well, for some reason data sets apart from CDX NA IG S8, sometimes produce low copula correlations with low probabilities, and high copula correlations with high probabilities. This observation is explained in the next chapter, by comparing the single-period single-factor copula from [1] to the seemingly equivalent multi-period single-factor copula parameterization (single path with

[5]Due to time constraints, we had to limit NEWUOA daily runs at 500 iterations per day. In practice, we need more than 500 iterations to reduce the variability in objective function values in Figure 6.2.

[6]General description of all calibration result tables is provided at the beginning of Section A.3.

probability 1).

High objective function values for CDX NA IG S10 and S11 data sets could be due to:

- default probabilities α_i (2.17) not accurately representing a volatile market;
- CDO quotes are provided for a different set of pricing equations;
- CDO quotes were adjusted due to some business contract (usual market assumptions are not applicable).

Zhang [8] mentions that his MSCM parameterization is not extremely parsimonious, whereas in our case the model uses 4 parameters only and still matches the market quotes reasonably well; Zhang's parameterization used 7 parameters. CDO data sets used in Table A.1 have 12-63 CDO quotes to match, so the two-period two-path parameterization is very parsimonious in practice.

The CMA ITRAXX EU S10 data set was also fitted reasonably well, with small objective function variability. We think that this is because the stock market crash of September 2008 had not yet had a big effect on the European market at the time.

6.2.2 Increasing Model Flexibility

Notice that we can decrease the error per number of data points by increasing the number of MSCM parameters, as demonstrated with the four-period two-path parameterization in Figure 6.3. Tables A.8 to A.13 on pages 134–139 show the CDO quotes produced by the four-period two-path parameterization with 8 parameters for the first day of each of the four data set. Because we are calibrating a larger number of model parameters, we also need to increase the number of BFGS2 iterations. However, due to time constraints, we limited BFGS2 to 120 iterations only, and there is a lot more variability in the objective function values in Figure 6.3 than in Figure 6.1. This could also be due to the BFGS2 algorithm converging to local minima for certain days in Figure 6.3, and this might also produce unrealistic copula tranche implied correlations in Tables A.8 to A.13 on pages 134–139 for the four-period two-path parameterization.

The industry-standard Hull Copula has higher errors in Figure 6.4 than the rest of our daily run figures, as expected. We also note that the CDX NA IG S11 data set has a few days where the Hull Copula produces very low objective function values. We believe that this is because the agency providing the CDO quotes could have used some variant of the Hull Copula for these quotes. Naturally, when calibrating the Hull Copula on those days, the error is fairly low. We demonstrate later in Chapter 7 that the MSCM and the Hull Copula are not entirely equivalent when estimating pool loss probabilities.

6.2.3 Choice Of The Error Function

We hypothesized in Section 2.7 that it is best to use the relative SEF (2.61) in the unconstrained objective function $F(\vec{u})$ (3.5). We show that this is true, by calibrating MSCM for the first day of each data set with the two-period two-path parameterization and an absolute SEF (2.58), with $\epsilon = 10^{-4}$ and $\delta = 0.5$. Calibration results are presented in Tables A.14 to A.19 on pages 140–145. When comparing the results to the relative SEF, presented in Tables A.2 to A.7 on pages 128–133, we observe that the more senior tranches, which have lower spreads, don't calibrate well, as argued in Section 2.7. Also, even the CDX NA IG S8 data set does not calibrate to reasonable copula correlation parameter values with the absolute SEF. The same data set calibrated well with the relative SEF for all days before the credit crunch of July 2007. Hence, we conclude that it is best to use a relative SEF in the unconstrained objective function $F(\vec{u})$ (3.5).

6.2.4 Runtimes

Negligible differences in runtimes between different data sets in Tables 6.1, 6.2 and 6.3 suggest that pricing is not affecting the runtimes and parallelization of pricing was han-

Data set	Mean Calibration Time ± Std. Dev. (minutes)
CDX NA IG S8	11.6 ± 2.52
CDX NA IG S10	9.91 ± 3.53
CDX NA IG S11	9.14 ± 3.47
CMA ITRAXX EU S10	11.5 ± 2.54

Table 6.1: Mean calibration times for data sets across all days in each CDO series using the BFGS2 algorithm with at most 40 iterations and a 2-period 2-path multi-path parameterization (4 parameters) with periods each of 5 years.

Data set	Mean Calibration Time ± Std. Dev. (minutes)
CDX NA IG S8	7.45 ± 2.62
CDX NA IG S10	7.36 ± 2.45
CDX NA IG S11	7.31 ± 2.42
CMA ITRAXX EU S10	7.43 ± 2.62

Table 6.2: Mean calibration times for data sets across all days in each CDO series using the NEWUOA algorithm with at most 500 iterations and a 2-period 2-path multi-path parameterization (4 parameters) with periods each of 5 years.

dled successfully (the data sets used require anywhere between 12 to 63 data points, so we can price a varying number of CDO quotes in roughly the same amount of time). Also, the runtimes are clearly reasonable and demonstrate that MSCM can be used in practice.

Data set	Mean Calibration Time $\pm$ Std. Dev. (minutes)
CDX NA IG S8	20.0 ± 4.09
CDX NA IG S10	20.9 ± 4.83
CDX NA IG S11	17.5 ± 7.21
CMA ITRAXX EU S10	19.9 ± 4.29

Table 6.3: Mean calibration times for data sets across all days in each CDO series using the BFGS2 algorithm with at most 120 iterations and a 4-period 2-path multi-path parameterization (8 parameters) with periods each of 2.5 years.

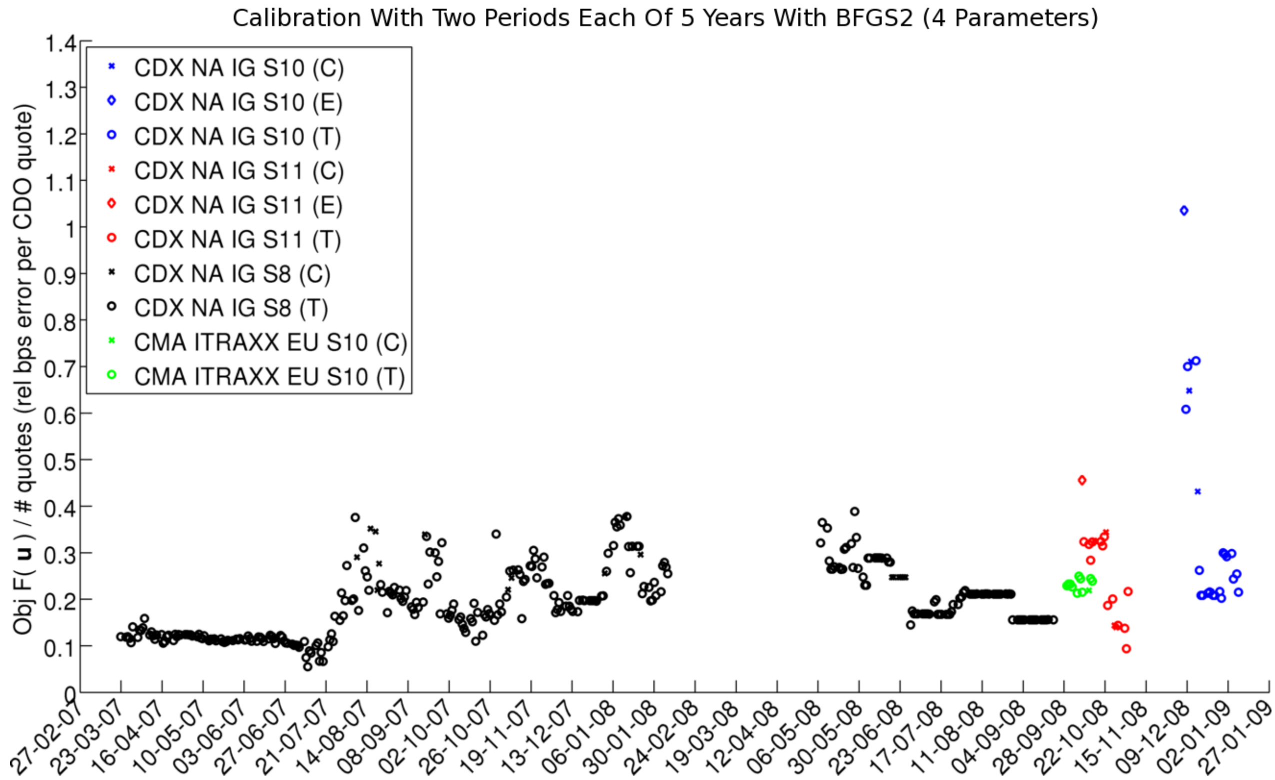

Figure 6.1: Calibration results with the BFGS2 algorithm [16] (described in Paragraph 2.8.2.3) limited to 40 iterations, for all four data sets with the relative Soft Error Function (SEF) (2.61) used in the objective function $F(\vec{u})$ (3.5); error is shown after dividing $F(\vec{u})$ by the number of data points available in each data set. MSCM parameterization uses two periods each of 5 years, with two paths per period. The letter (C) indicates that BFGS2 converged, the letter (T) indicates that the algorithm was terminated after 40 iterations, and the letter (E) indicated that a value of $\beta_i = 1$ was encountered during the optimization and BFGS2 signaled for termination.

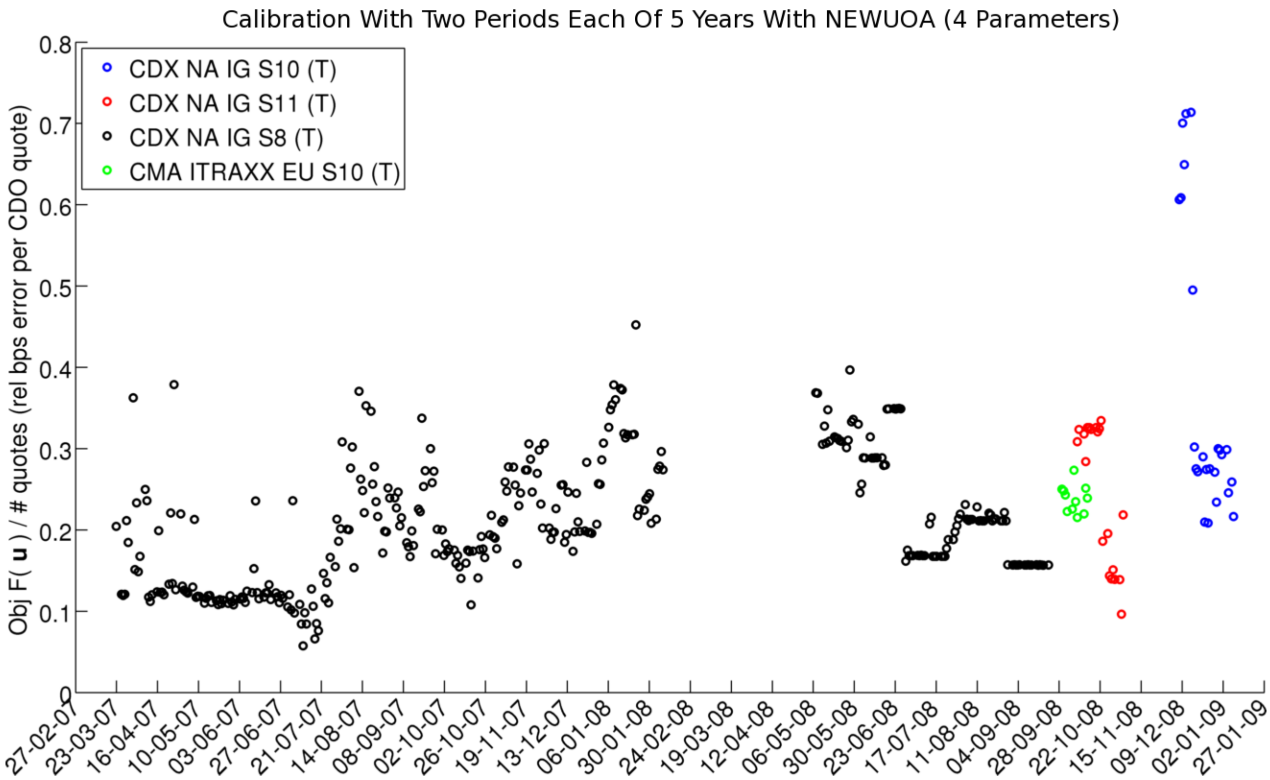

Figure 6.2: Calibration results with the NEWUOA algorithm [13] (described in Subsubsection 2.8.1.2) limited to 500 iterations, for all four data sets with the relative Soft Error Function (SEF) (2.61) used in the objective function $F(\vec{u})$ (3.5); error is shown after dividing $F(\vec{u})$ by the number of data points available in each data set. MSCM parameterization uses two periods each of 5 years, with two paths per period. The letter (T) indicates that NEWUOA was terminated after 500 iterations.

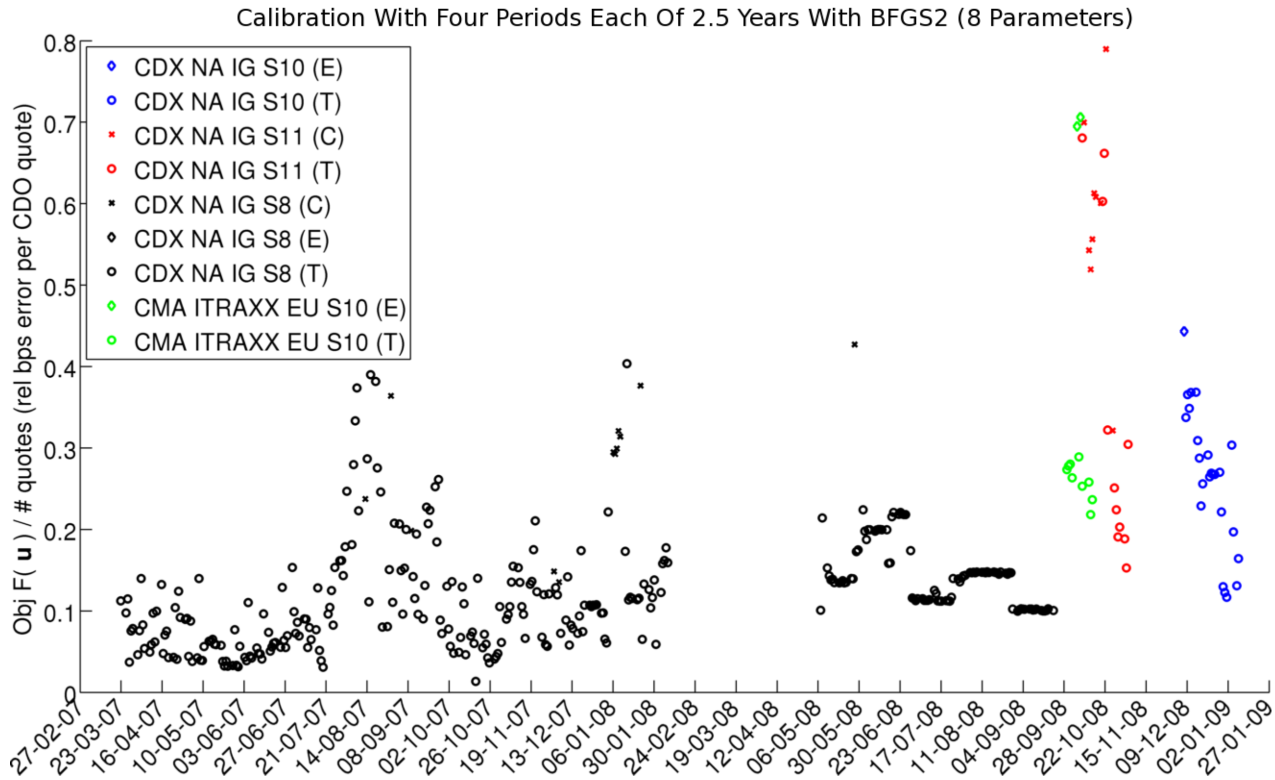

Figure 6.3: Calibration results with the BFGS2 algorithm [16] (described in Paragraph 2.8.2.3), limited to 120 iterations, for all four data sets with the relative Soft Error Function (SEF) (2.61) used in the objective function $F(\vec{u})$ (3.5); error is shown after dividing $F(\vec{u})$ by the number of data points available in each data set. MSCM parameterization uses four periods each of 2.5 years, with two paths per period. The letter (C) indicated that BFGS2 has converged, the letter (T) indicates that the algorithm was terminated after 120 iterations, and the letter (E) indicated that a value of $\beta_i = 1$ was encountered during optimization and BFGS2 signaled for termination.

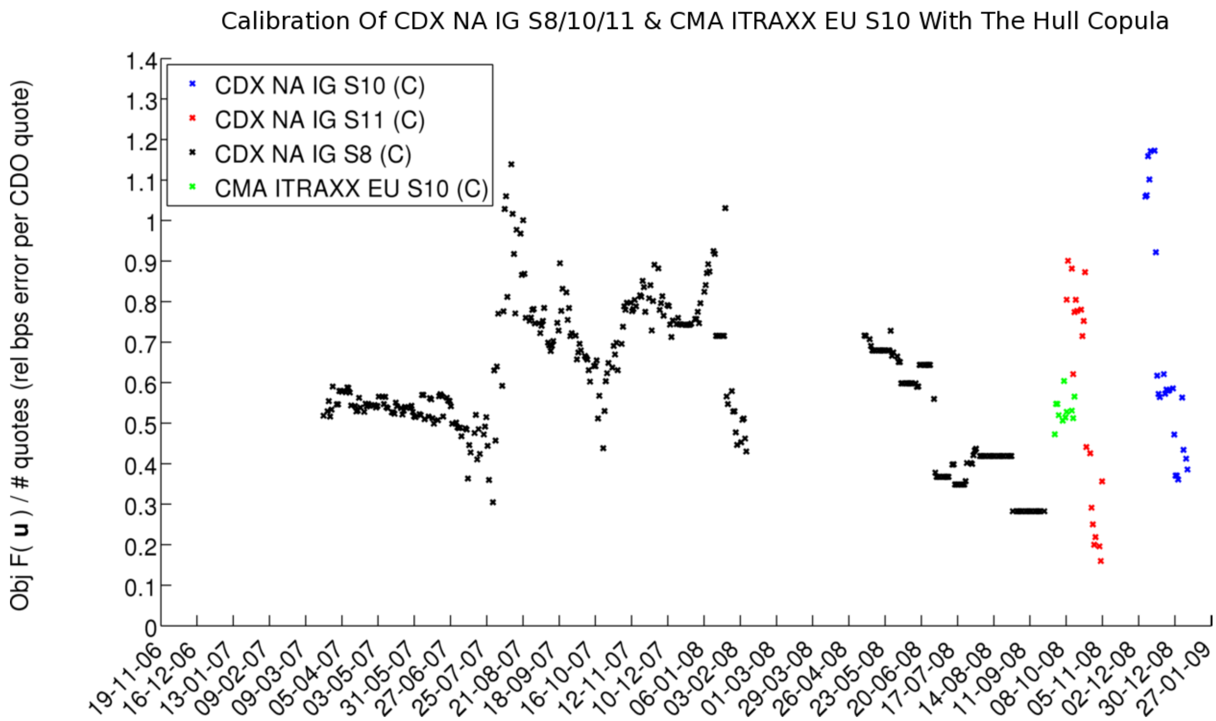

Figure 6.4: Calibration results for the regular industry-standard Hull Copula [1] single-factor single-period model for all four data sets with the relative Soft Error Function (SEF) (2.61) used in the objective function $F(\vec{u})$ (3.5); the error is shown after dividing $F(\vec{u})$ by the number of data points available in each data set.

Chapter 7

Some Observations And Questions

We compare the single-period single-factor copula from [1] (denoted for brevity as the Hull Copula in Section 6.2) to the Multi-period Single-factor Copula Model (MSCM) by setting $\beta_i = \beta$ for $i = 1, 2, \cdots, 40$ with probability 1 in the multi-path parameterization of MSCM discussed in Section 3.1, where β is the copula correlation parameter. Figure 7.1 shows the spreads produced by both models as a function of β, where we are trying to match the $3 - 7\%$ tranche at maturities of 5, 7 and 10 years on March 23, 2007 for the CDX NA IG S8 data set[1].

We make the following observations:

- for $T = 5$ and $T = 7$, two values of β match the market quote;

- the models are not necessarily equivalent for $\beta \in (0, 1]$, but are guaranteed to produce identical quotes for $\beta = 0$; we believe that this is because the default probabilities α_i (2.17) are modeled as a step function on each $(t_{i-1}, t_i]$ in the MSCM model, whereas the Hull Copula model assumes a continuous underlying function for the α_i's. As a result, the pool loss probabilities $P(l_i^{(\text{pool})} = r)$ differ for $\beta \in (0, 1]$.

  For shorter maturities (for example, $T = 5$ and $T = 7$ in Figure 7.1), the two

[1] The same data set calibrated well in Figure 6.1

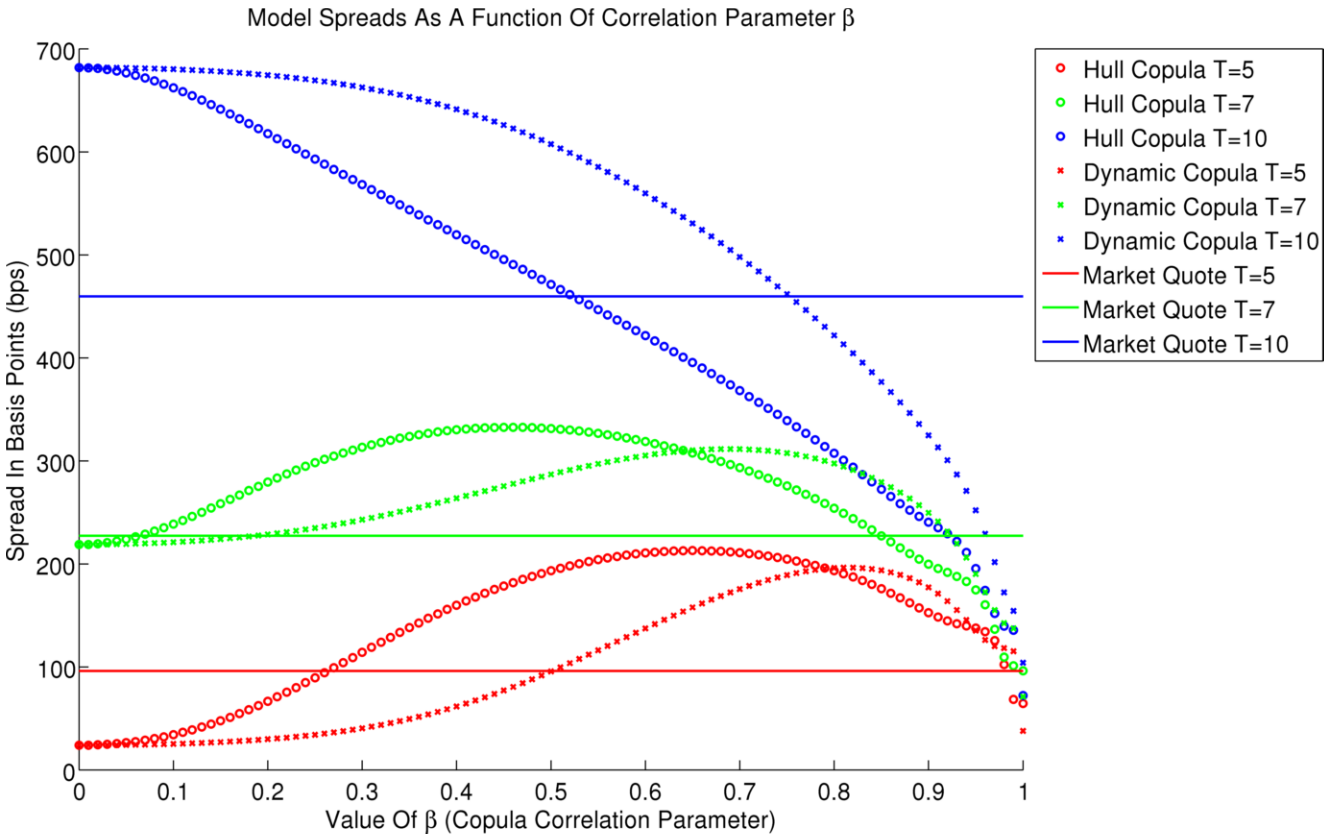

Figure 7.1: Spreads produced by single and multi-period single-factor copula models as a function of the copula correlation parameter β, plotted against the $3-7\%$ tranche at maturities of 5, 7 and 10 years on March 23, 2007 for the CDX NA IG S8 data set.

models can produce the same quote value for some other $\beta_i \in (0,1]$, but this is not always true.

- CDO spreads, plotted as a function of β, change shape for both models as the spread quote maturity T increases.

This suggests that when calibrating either model, we need to decide which range of β is plausible for shorter maturities, and constrain β. However, how do we know this range? One possible reason why some data sets result in $\beta \approx 1$ with probability ≈ 1 is because of this behavior: CDO spread quotes for longer maturities suggest higher β values and because those values also calibrate the shorter maturity quotes well, we see an overly inflated β value with probability ≈ 1. Figure 7.1 also suggests that tranche implied copula correlations are less ambiguous for long range correlation representation.

Research Questions In order to use the dynamics of the multi-period single-factor copula model, we need to be able to calibrate the model for any value of T on post-crash market data. Further research is needed to answer the following questions:

- Why are the CDO spread quotes so different between the two models?
- Why do different values of β match the CDO market quotes at shorter maturities for both models?
- What causes the large increase in the objective function $F(\vec{u})$ (3.5) per number of data points in the multi-period single-factor copula model when applied to data sets that include crash periods?
- Why can we match market quotes with fairly low error per data point in, for example, Figure 6.1, yet calibrate to unrealistic tranche implied copula correlation values?

Chapter 8

Conclusion

In this thesis, we developed an alternative multi-path parameterization to the Multi-period Single-factor Copula Model (MSCM), recently proposed by Jackson, Kreinin and Zhang [11]. This parameterization allowed us to compute the first-order derivatives of the objective function

$$f(\vec{\psi}) = \sum_{\mathrm{tr} \in \mathrm{Tr}} \sum_{T \in M} \mathrm{error}\left(E_{\vec{\rho}} \left[s_{n_T}^{(\mathrm{tr})}(\vec{\beta}, \vec{\alpha}) | \vec{\gamma} \right], m_{n_T}^{(\mathrm{tr})} \right),$$

discussed in Section 2.5, in closed form, for all reasonable values of $\vec{\alpha}$ and $\vec{\beta}$. This enables us to use derivative-based optimization algorithms to calibrate the MSCM, thereby improving the efficiency of the calibration process. In addition, multi-path parameterization provides a more intuitive structure for the market dynamics, by associating a unique copula correlation parameter path with a unique probability for each period of the MSCM.

We also developed a robust and efficient software implementation of the MSCM by determining an error control heuristic for the pool loss probabilities and their derivatives. We also provide a useful theoretical result that if a quadrature routine can guarantee a certain error in approximating the integral

$$\binom{K-m}{r-m} \int_{-\infty}^{\infty} \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}} \right)^{r-m} \left(1 - \Phi\left(\frac{\Phi^{-1}(\alpha_i) - \beta_i x}{\sqrt{1-\beta_i^2}} \right) \right)^{K-r} d\Phi(x)$$

for all $r = 0, 1, \cdots, K$ and $m = 0, 1, \cdots, r$, then we can guarantee that the error in the pool loss probability is below a certain threshold.

We further tested the MSCM on four distinct data sets from periods before, during and after the 2008-2009 stock market crash, and compared a simple parameterization of MSCM to the seemingly equivalent single-period single-factor copula model discussed in [1]. This comparison suggests that copula models are accurate for modeling long-term tranche implied correlations for CDO pricing, as discussed in Chapter 7, but may produce inaccurate tranche implied copula correlations for shorter maturities. This suggests several research questions, outlined in Chapter 7.

Regarding the multiple period structure of the MSCM, market quote fits are greatly improved my adding more periods to the MSCM, but we did not see a significant improvement with the addition of more paths in each period. We showed that the multi-period nature of the MSCM improves market quote fits over the single-period single-factor copula model. Regardless of the number of CDO market quotes and the number of periods, we demonstrated that multi-path parameterization of the MSCM is relatively inexpensive to calibrate, and that the MSCM can be used effectively in practice.

Appendix A

Appendix

A.1 Proofs

A.1.1 Recursion Relationship Of Pool Loss Probabilities

Reference [8] shows that

$$P(l_i^{(\text{pool})} = r) = \sum_{m=0}^{r} P(l_{i-1}^{(\text{pool})} = m) \cdot P(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m), \tag{A.1}$$

using a lemma from Section A.1.2, restated here for convenience:

$$q_{k,i}^{(\text{def})} = P(\tau_k \le t_i | X_1 = x_1, X_2 = x_2, ..., X_i = x_i) = q_{k,i-1}^{(\text{def})} + (1 - q_{k,i-1}^{(\text{def})}) \cdot p_{k,i}^{(\text{def})}, \tag{A.2}$$

where

$$p_{k,i}^{(\text{def})} = P(\tau_k \le t_i | \tau_k > t_{i-1}, X_i = x_i). \tag{A.3}$$

For homogeneous pools, $p_{k,i}^{(\text{def})} = p_i^{(\text{def})}$ and $q_{k,i}^{(\text{def})} = q_i^{(\text{def})}$. Notice that

$$P(l_i^{(\text{pool})} = r) = \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} P(l_i = r | X_1 = x_1, X_2 = x_2, ..., X_i = x_i) d\Phi(x_1)...d\Phi(x_i), \tag{A.4}$$

so we need to determine the conditional probability $P(l_i^{(\text{pool})} = r | X_1 = x_1, X_2 = x_2, ..., X_i = x_i)$. We determine the conditional probability of r defaults given the conditional probability of each default using the Binomial probability distribution:

$$P(l_i^{(\text{pool})} = r | X_1 = x_1, X_2 = x_2, ..., X_i = x_i) = \begin{pmatrix} K \\ r \end{pmatrix} \left(q_i^{(\text{def})}\right)^r \left(1 - q_i^{(\text{def})}\right)^{K-r}. \quad (A.5)$$

Next, we substitute (A.2), use the binomial expansion on the first probability term, and then group terms to obtain the conditional recurrence relation:

$$\begin{aligned}
&\begin{pmatrix} K \\ r \end{pmatrix} \left(q_{i-1}^{(\text{def})} + (1 - q_{i-1}^{(\text{def})}) p_i^{(\text{def})}\right)^r \left((1 - q_{i-1}^{(\text{def})}) \cdot (1 - p_i^{(\text{def})})\right)^{K-r} = \\
&\begin{pmatrix} K \\ r \end{pmatrix} \left(\sum_{m=0}^{r} \begin{pmatrix} r \\ m \end{pmatrix} \left(q_{i-1}^{(\text{def})}\right)^m \left(1 - q_{i-1}^{(\text{def})}\right)^{r-m} \left(p_i^{(\text{def})}\right)^{r-m}\right) \left(\left(1 - q_{i-1}^{(\text{def})}\right) \left(1 - p_i^{(\text{def})}\right)\right)^{K-r} = \\
&\sum_{m=0}^{r} \begin{pmatrix} K \\ m \end{pmatrix} \left(q_{i-1}^{(\text{def})}\right)^m \left(1 - q_{i-1}^{(\text{def})}\right)^{K-m} \begin{pmatrix} K-m \\ r-m \end{pmatrix} \left(p_i^{(\text{def})}\right)^{r-m} \left(1 - p_i^{(\text{def})}\right)^{K-m-(r-m)} = \\
&\sum_{m=0}^{r} P(l_{i-1}^{(\text{pool})} = m | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) P(l_{(i-1,i]}^{(\text{pool}),K-m} = r - m | X_i = x_i). \quad (A.6)
\end{aligned}$$

Now notice that when integrating out the common factors we obtain the required equation, because

$$\int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} P(l_{i-1}^{(\text{pool})} = r | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) d\Phi(x_1)...d\Phi(x_{i-1}) = P(l_{i-1}^{(\text{pool})} = m). \quad (A.7)$$

A.1.2 Lemma

We need to show that

$$q_{k,i}^{(\text{def})} = P(\tau_k \leq t_i | X_1 = x_1, X_2 = x_2, ..., X_i = x_i) = q_{k,i-1}^{(\text{def})} + (1 - q_{k,i-1}^{(\text{def})}) \cdot p_{k,i}^{(\text{def})}. \quad (A.8)$$

This is given by regular manipulation of probabilities

$$
\begin{aligned}
& q_{k,i}^{(\mathrm{def})} = P(\tau_k \le t_i | X_1 = x_1, X_2 = x_2, ..., X_i = x_i) \\
& = P(\tau_k \le t_{i-1} | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) + \\
& \qquad P(\tau_k \in (t_{i-1}, t_i] | X_1 = x_1, X_2 = x_2, ..., X_i = x_i) \\
& = P(\tau_k \le t_{i-1} | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) + \\
& \qquad P(\tau_k > t_{i-1} | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) \cdot P(\tau_k \in (t_{i-1}, t_i] | \tau_k > t_{i-1}, X_i = x_i) \\
& = q_{k,i-1}^{(\mathrm{def})} + (1 - q_{k,i-1}^{(\mathrm{def})}) \cdot p_{k,i}^{(\mathrm{def})}. \qquad (A.9)
\end{aligned}
$$

A.2 Figures

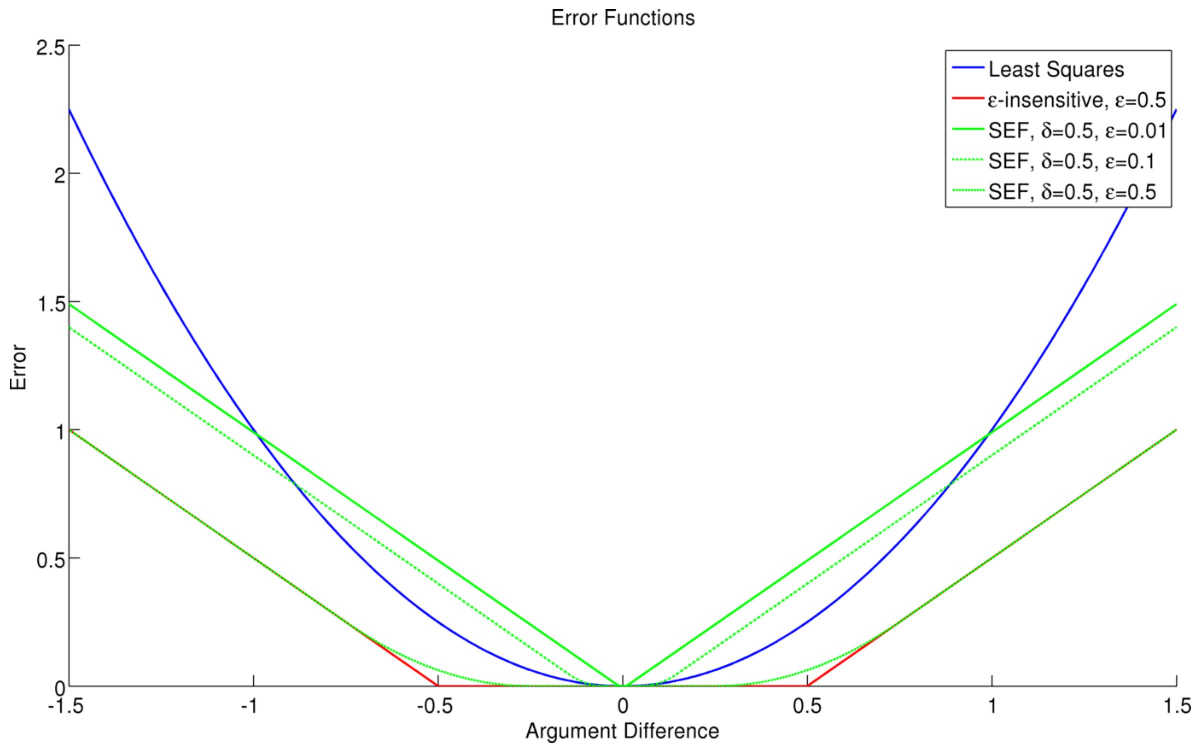

Figure A.1: Error functions used to calibrate the model, given by (2.56), (2.57) and (2.58).

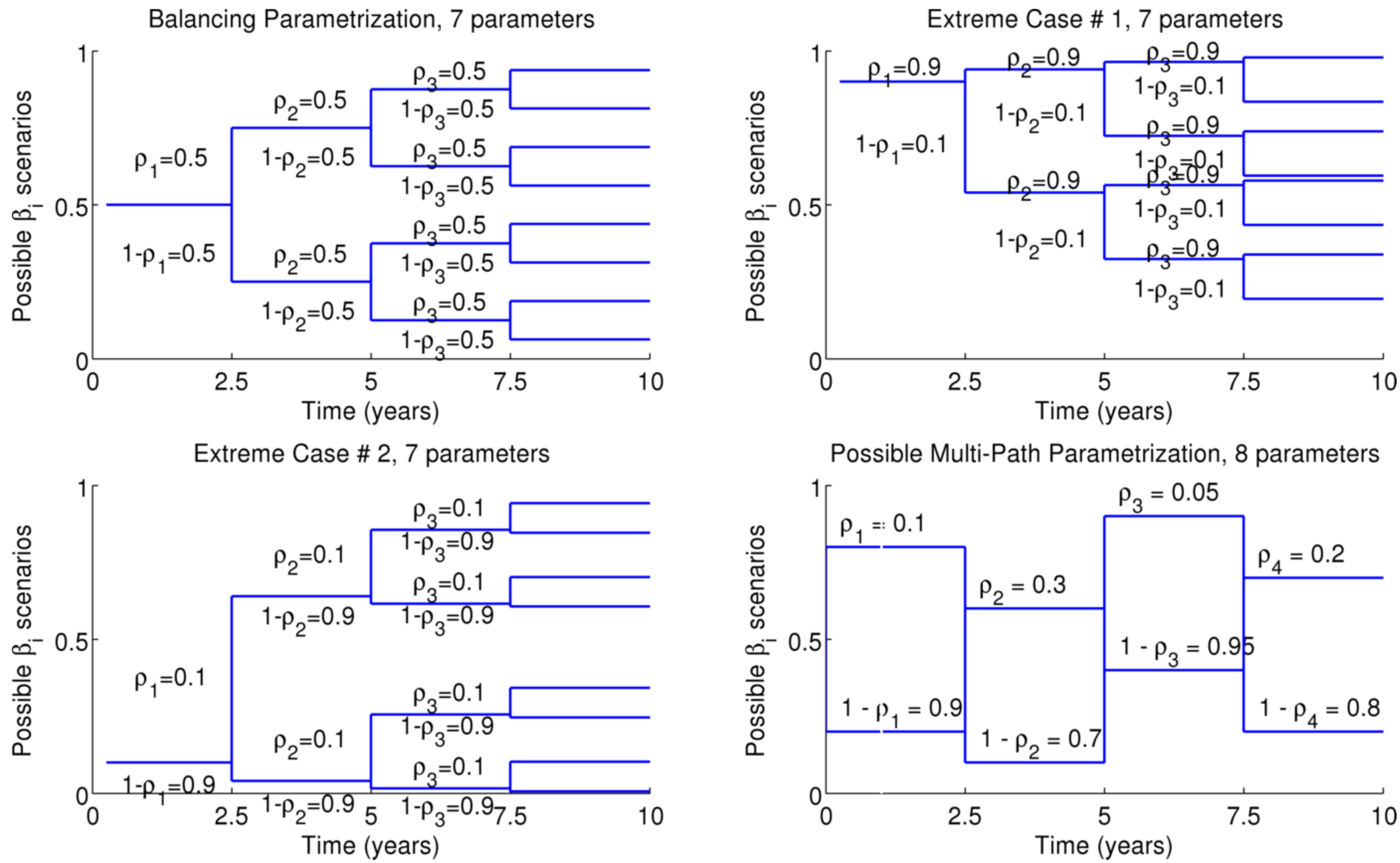

Figure A.2: Three loading factor binomial tree parameterizations and an alternative multi-path parameterization, which eliminates monotonic β_i behavior and provides better coverage of the stochastic process scenarios with more realistic probabilities.

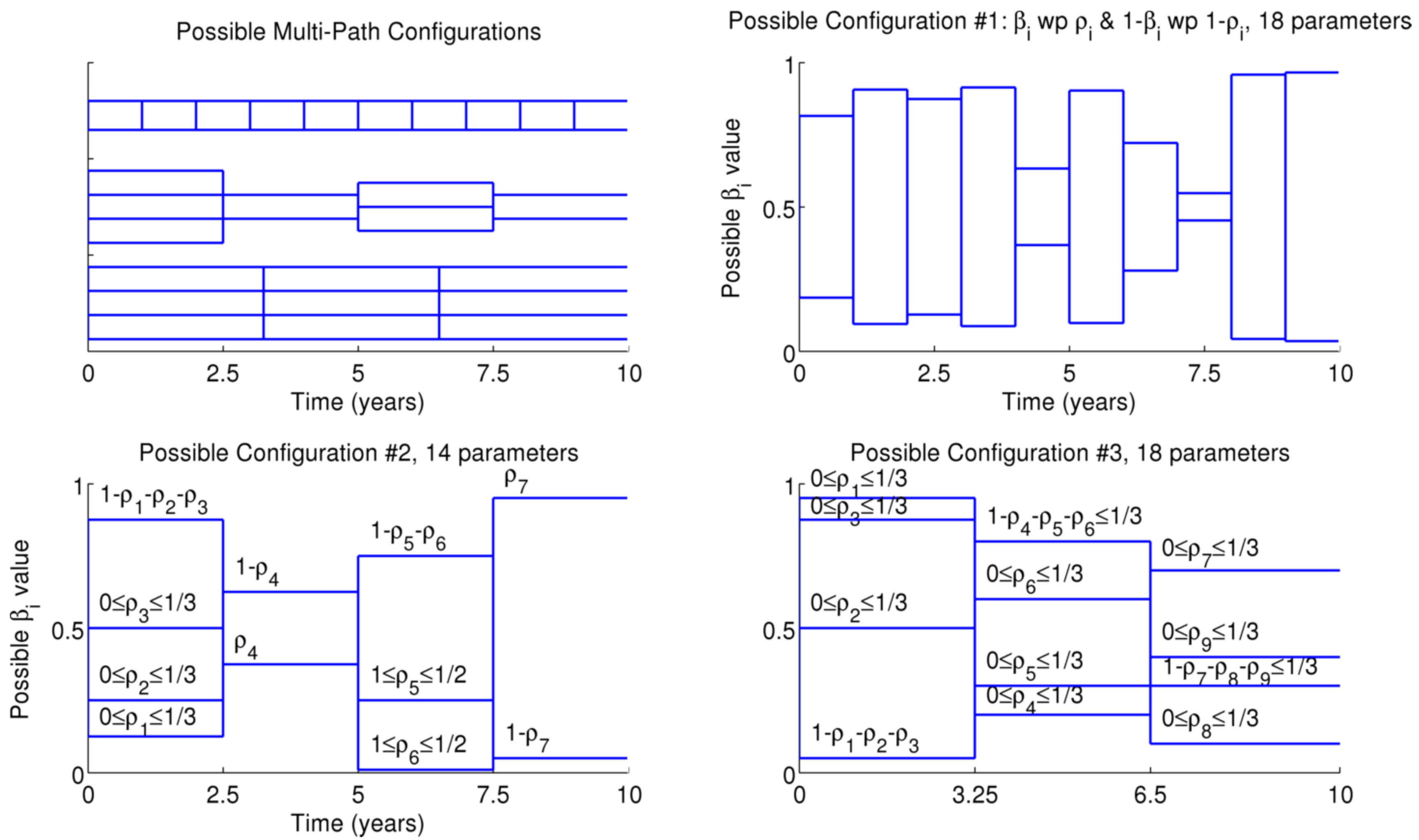

Figure A.3: Three possible multi-path branch configurations, along with a possible realization for each configuration.

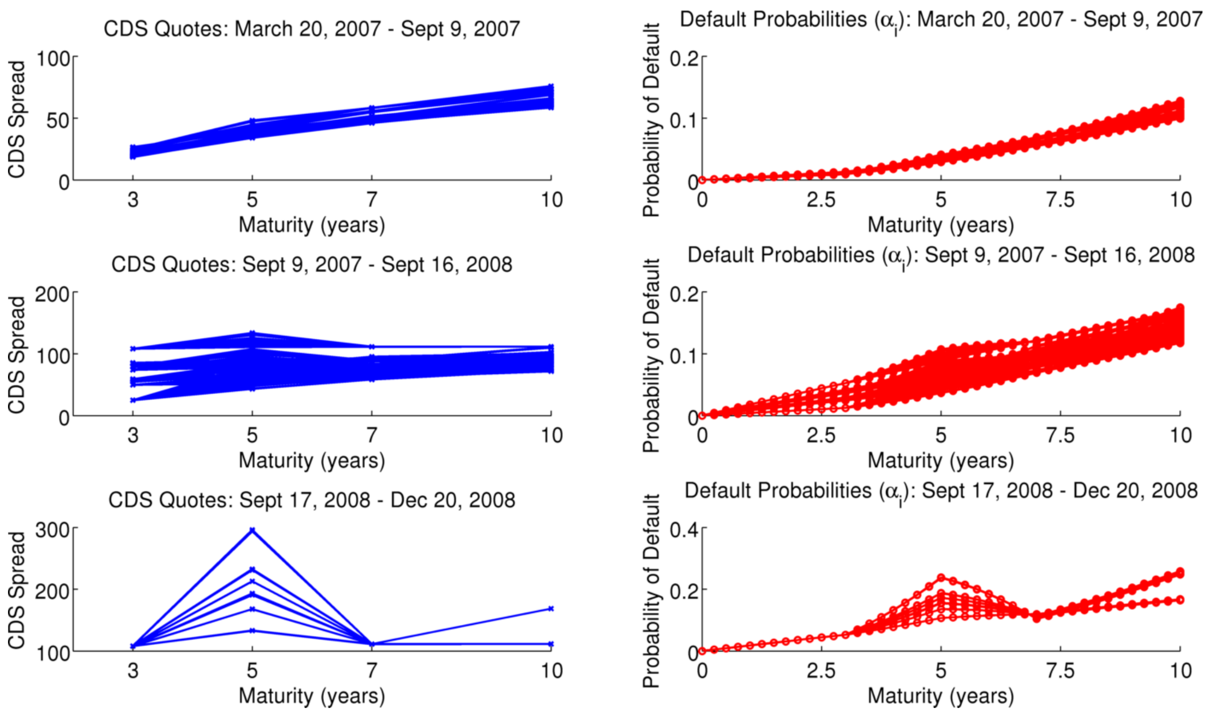

Figure A.4: CDS spreads and bootstrapped default probabilities used to calibrate CDX NA IG series 8 data set, split into three time regions across the rows of the figure.

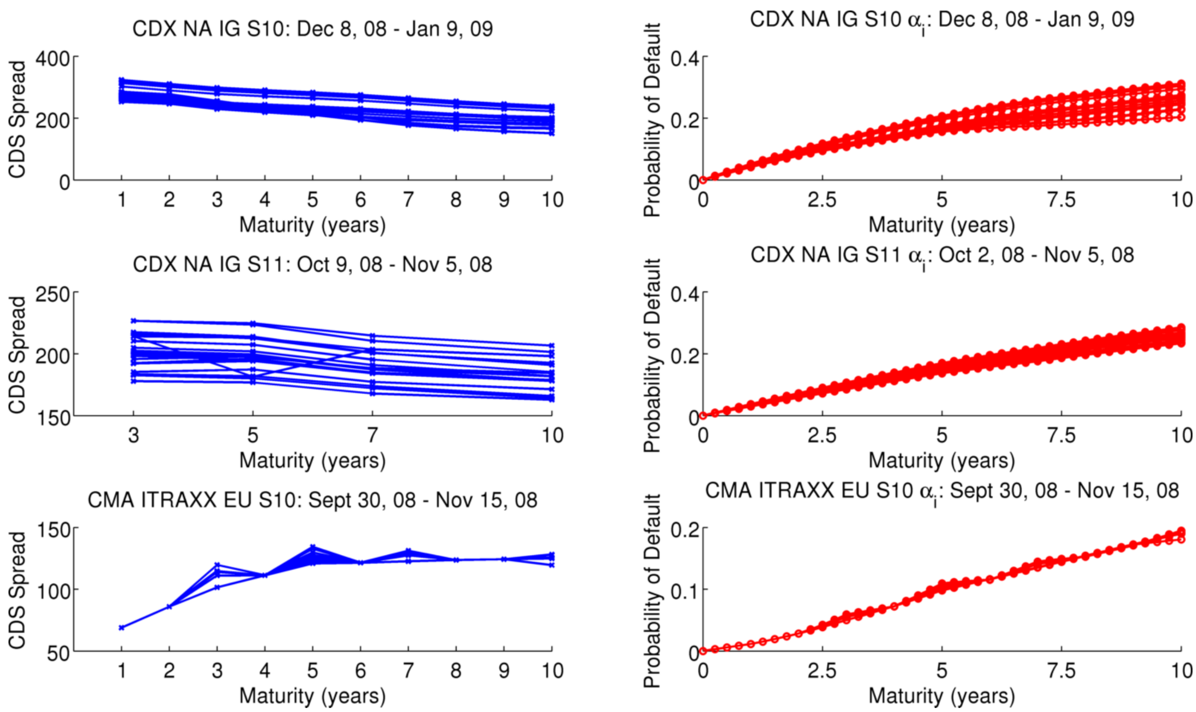

Figure A.5: CDS spreads and bootstrapped default probabilities from CDX NA IG series 10, 11 and CMA ITRAXX EU series 10 data sets.

A.2.1 Calibration Results

A.2.1.1 Single Period Multi-Path Parameterization With Two Paths Per Period

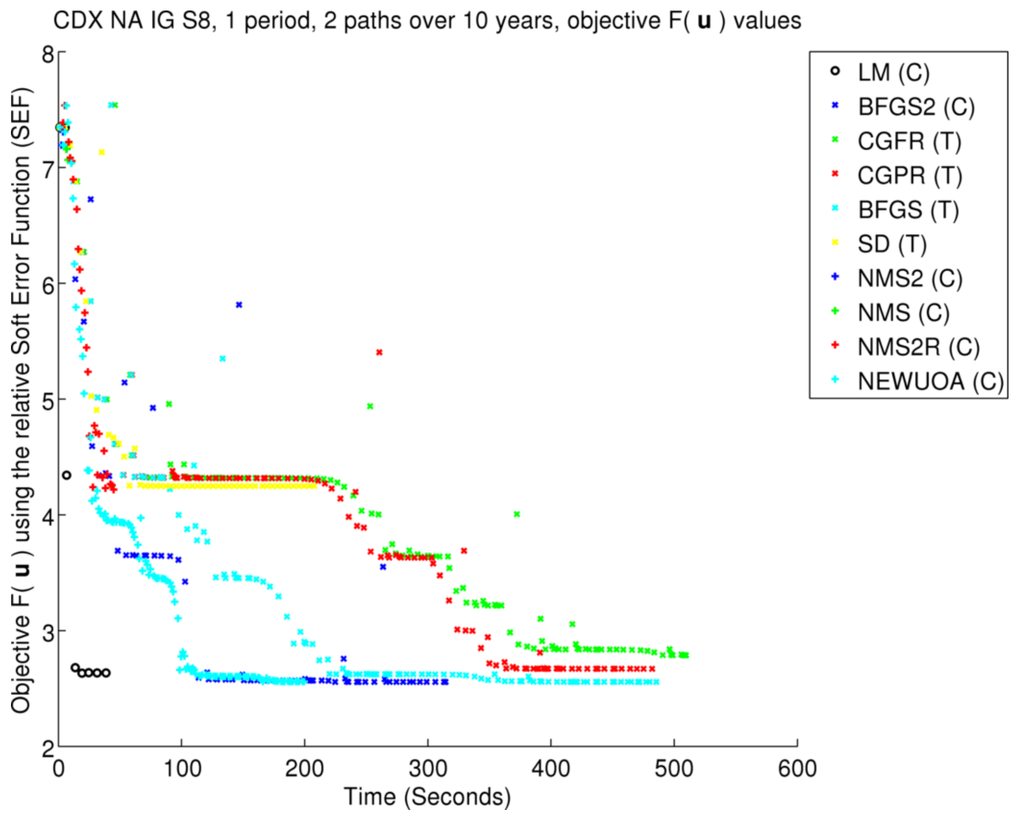

Figure A.6: CDX NA IG S8 data set from March 23, 2007 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with two paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

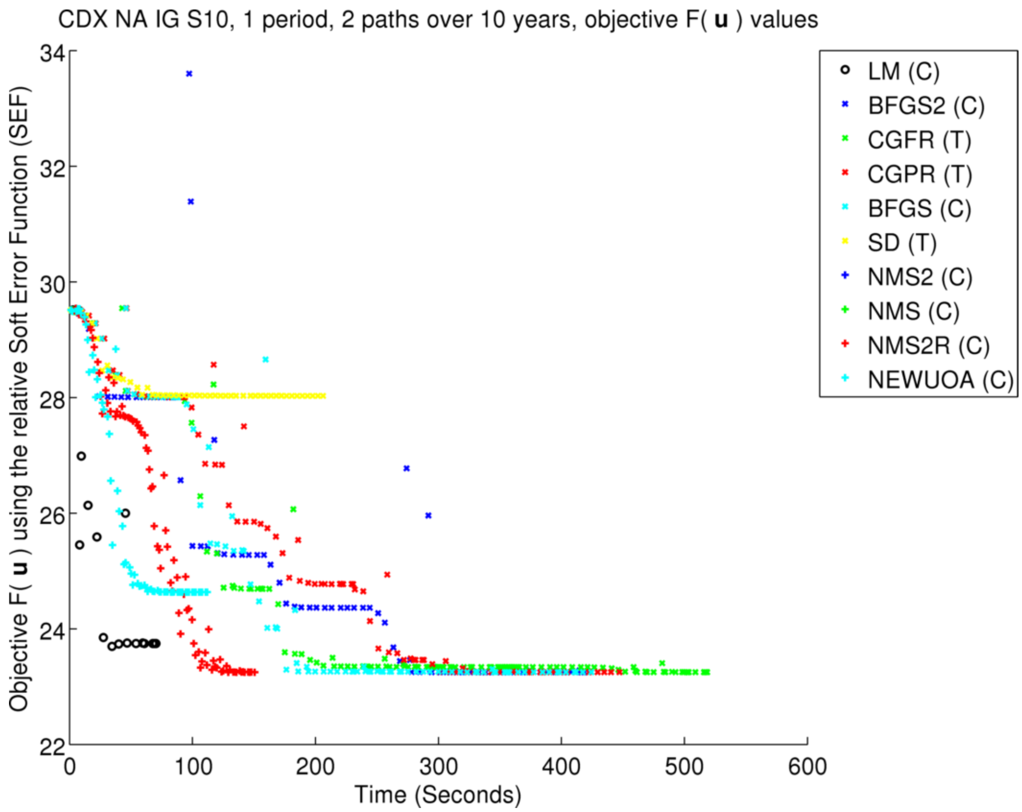

Figure A.7: CDX NA IG S10 data set from December 8, 2008 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with two paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

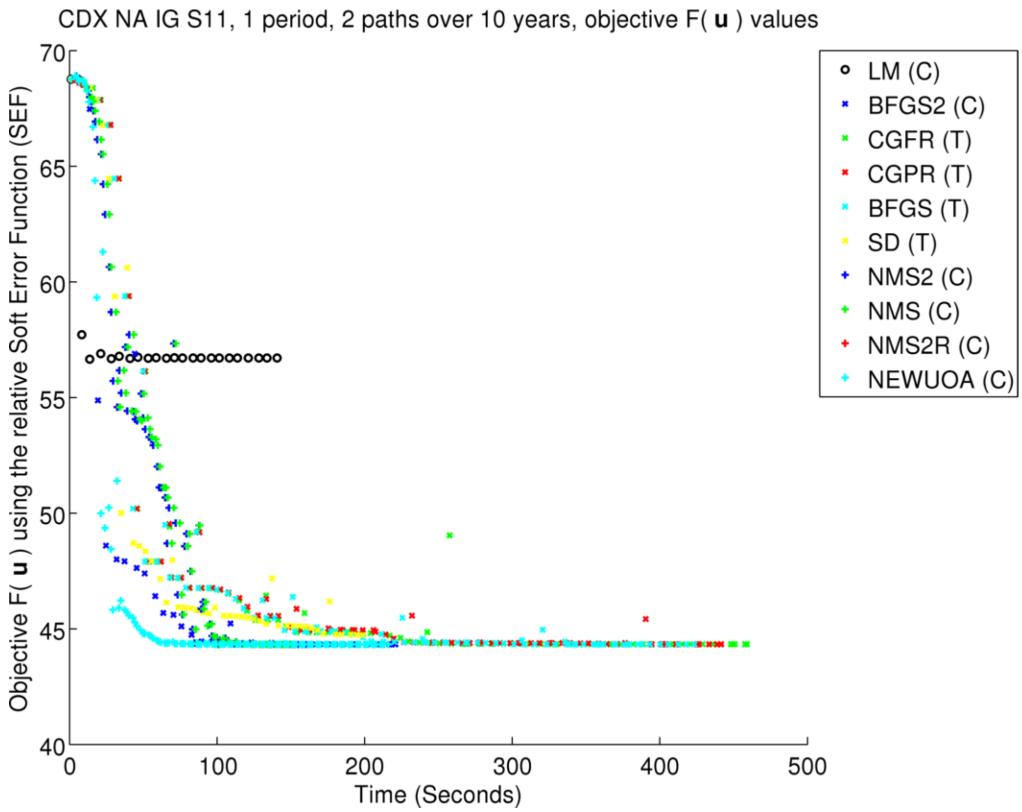

Figure A.8: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with two paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

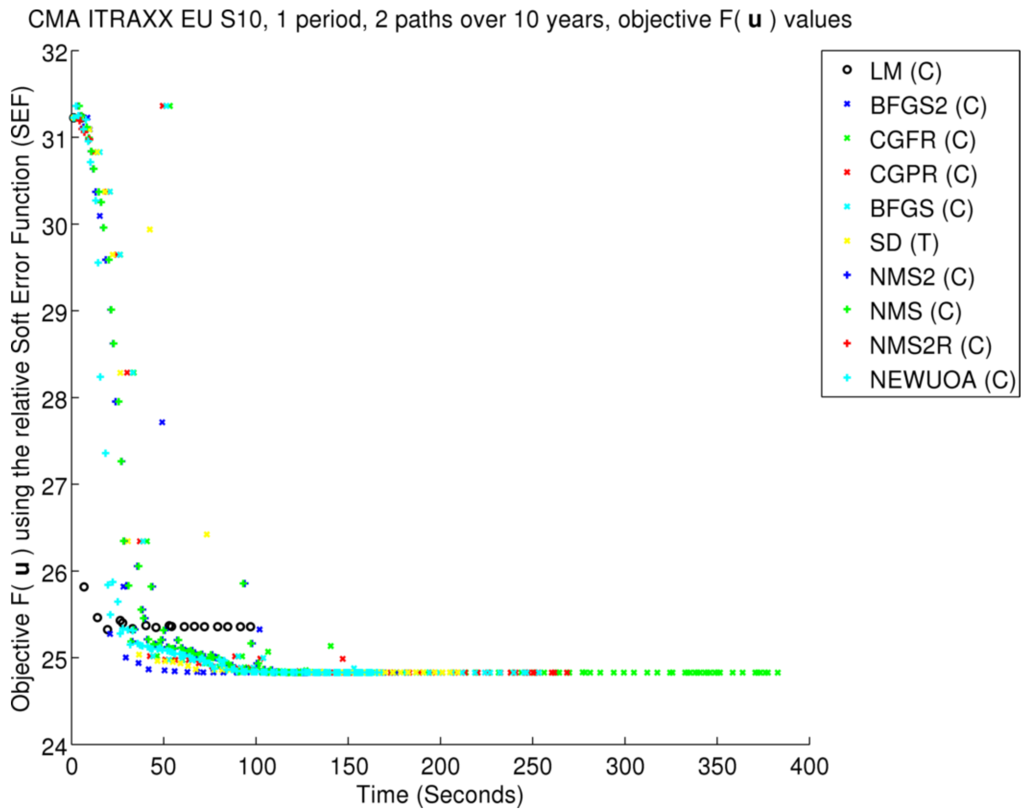

Figure A.9: CMA ITRAXX EU S10 data set from September 30, 2008 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with two paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

A.2.1.2 Single Period Multi-Path Parameterization With Four Paths Per Period

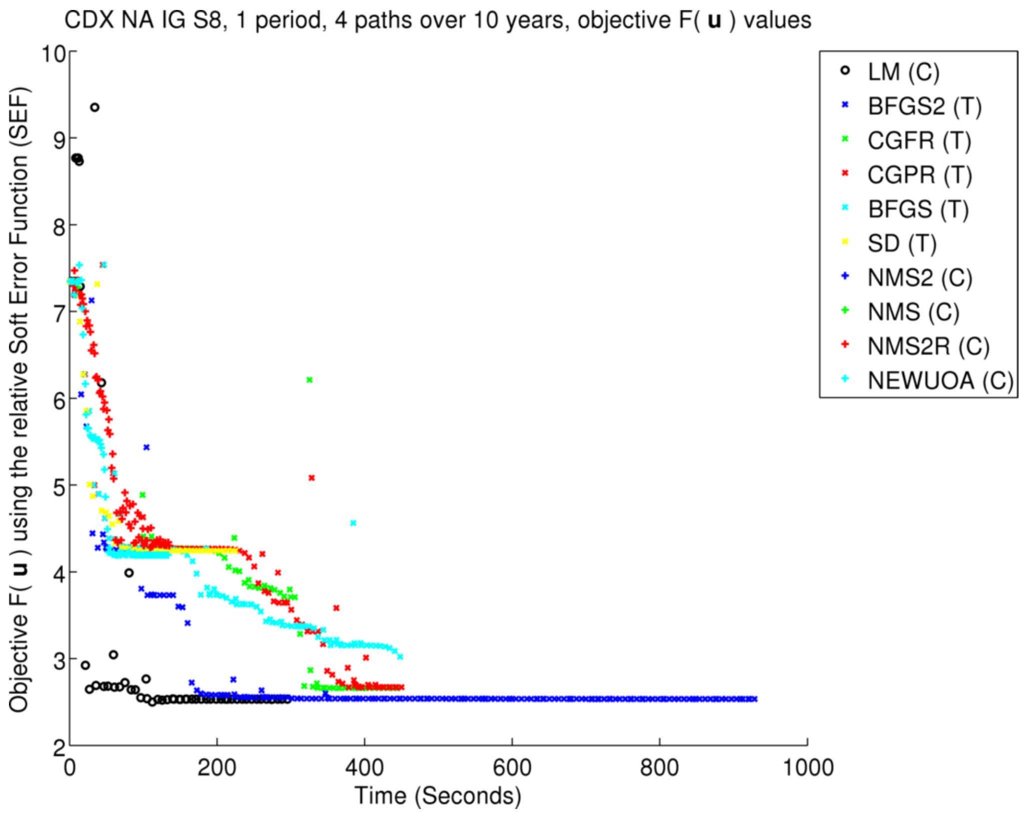

Figure A.10: CDX NA IG S8 data set from March 23, 2007 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with four paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

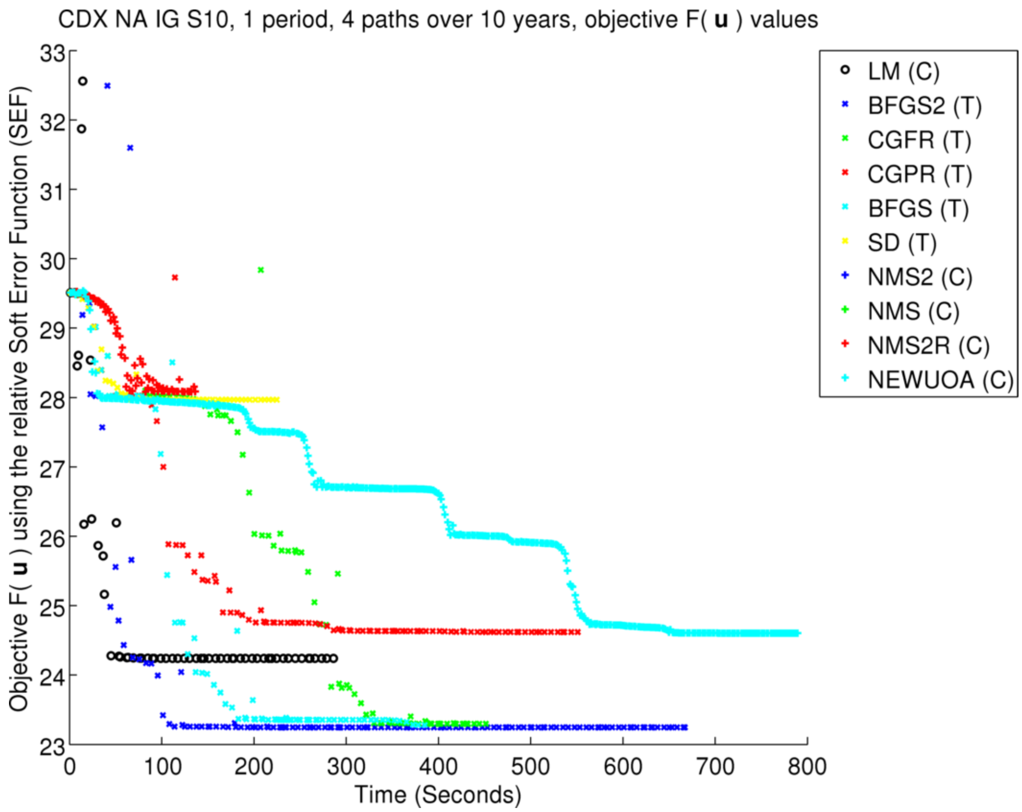

Figure A.11: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with four paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

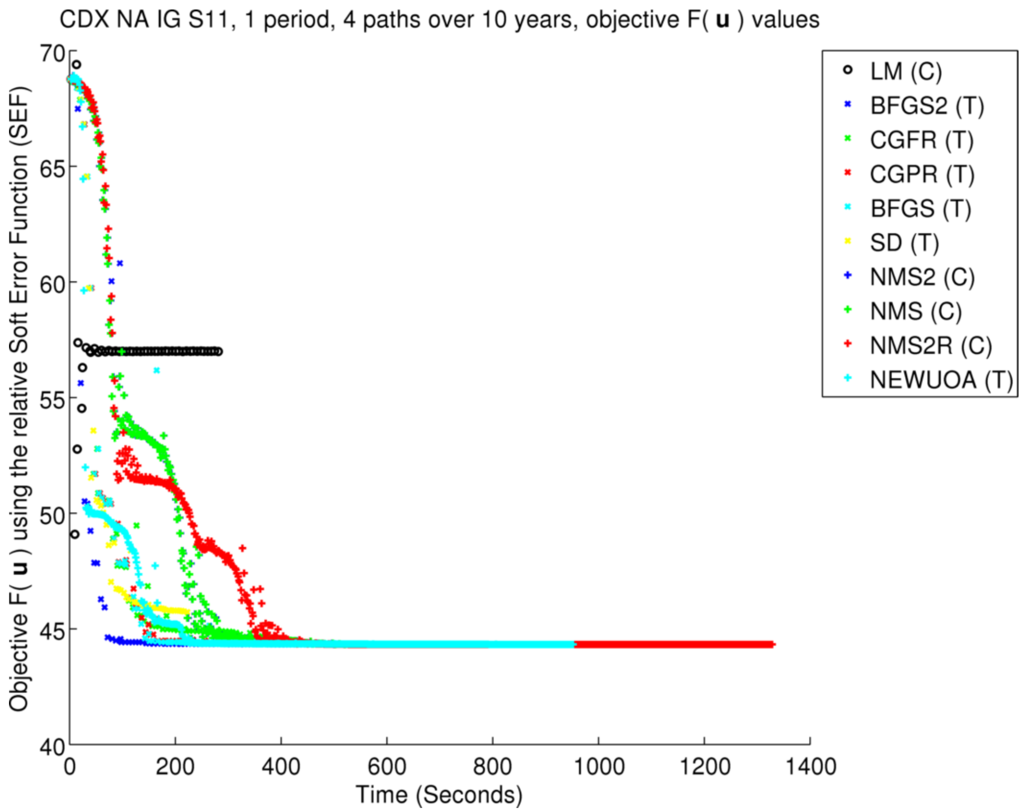

Figure A.12: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with four paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

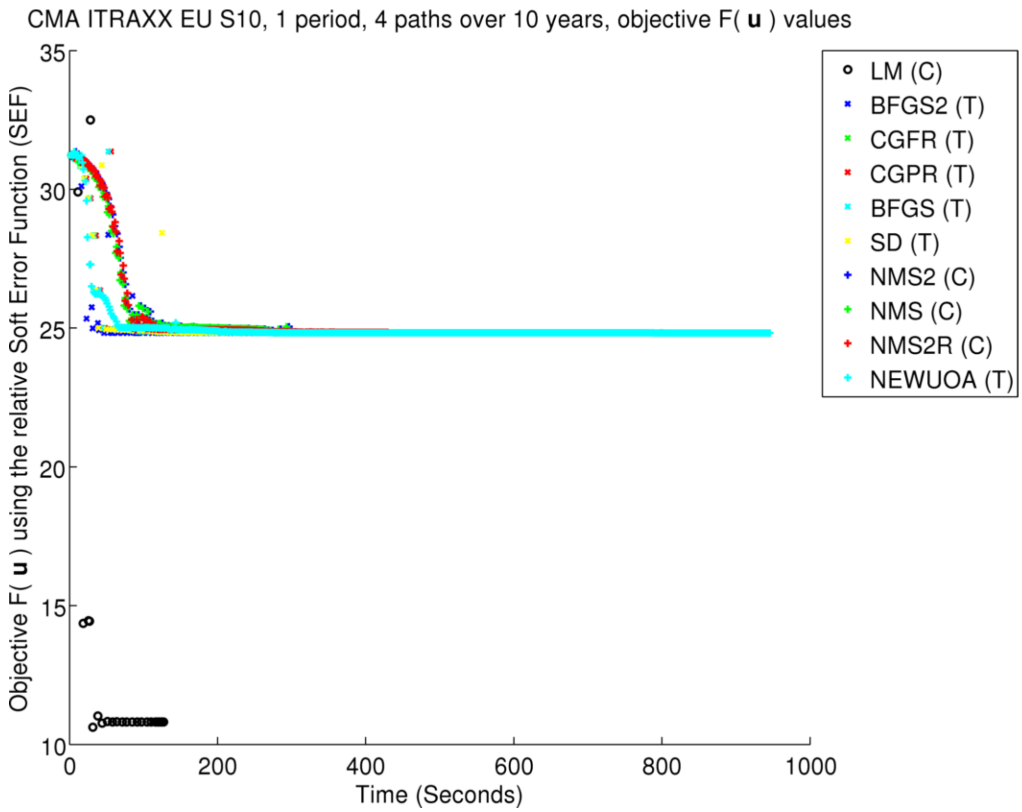

Figure A.13: CMA ITRAXX EU S10 data set from September 30, 2008 calibrated with optimization algorithms from Section 2.8 with a single period multi-path parameterization with four paths. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

A.2.1.3 Two Period Multi-Path Parameterization With Three Paths Per Period (T=5,10)

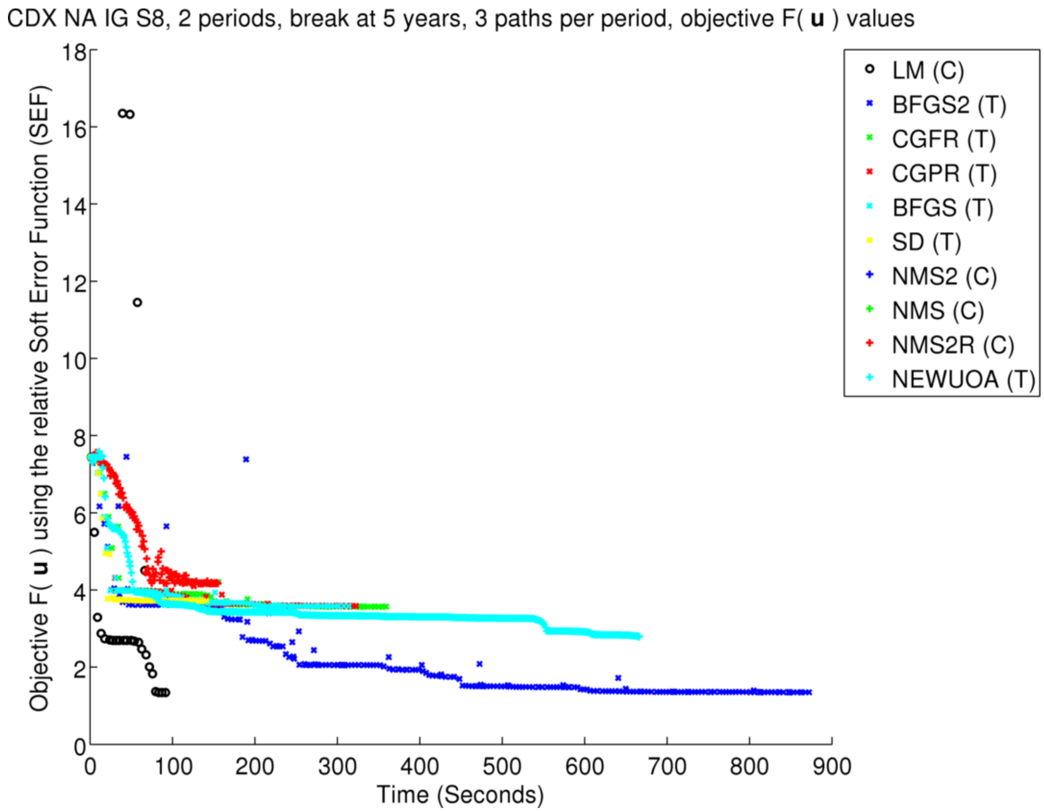

Figure A.14: CDX NA IG S8 data set from March 23, 2007 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

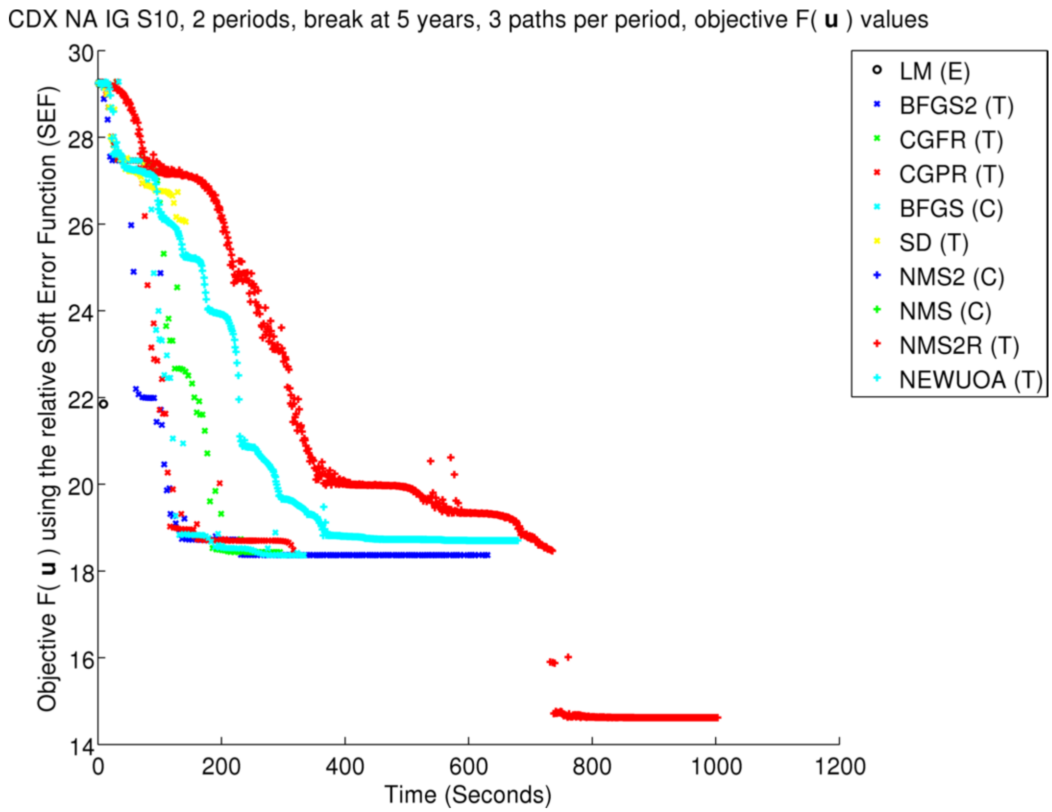

Figure A.15: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

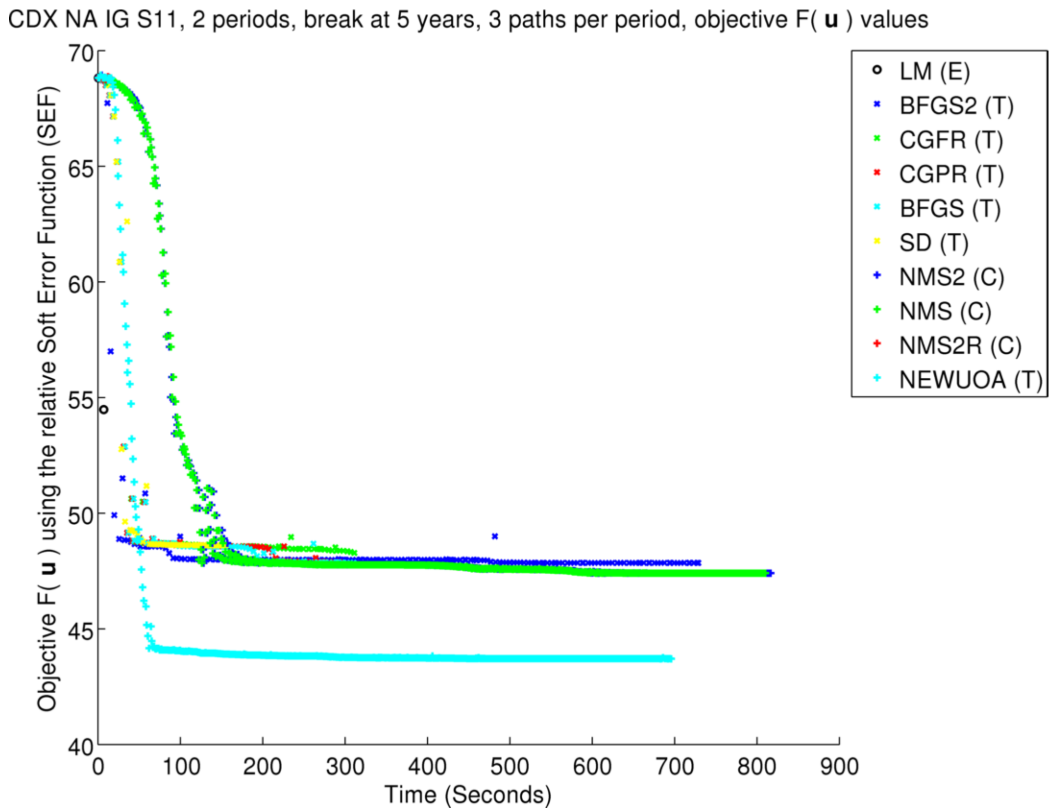

Figure A.16: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

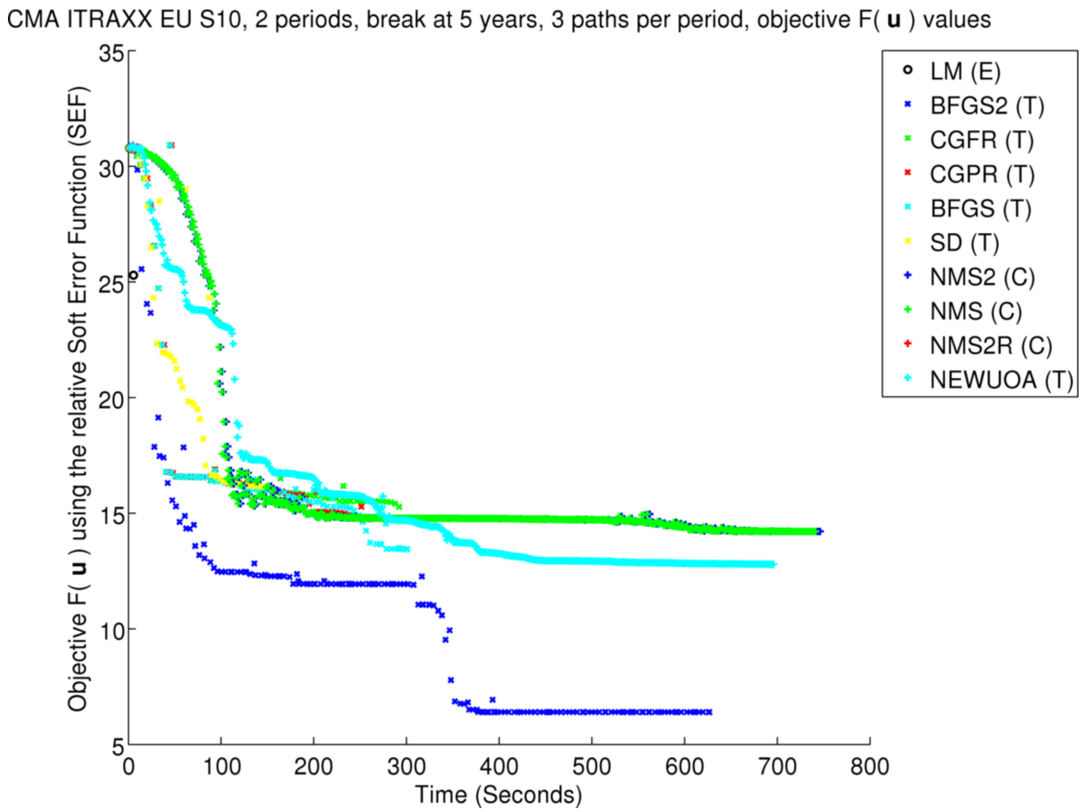

Figure A.17: CMA ITRAXX EU S10 data set from September 30, 2008 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

A.2.1.4 Four Period Multi-Path Parameterization With Two Paths Per Period (T=2.5,5,7.5,10)

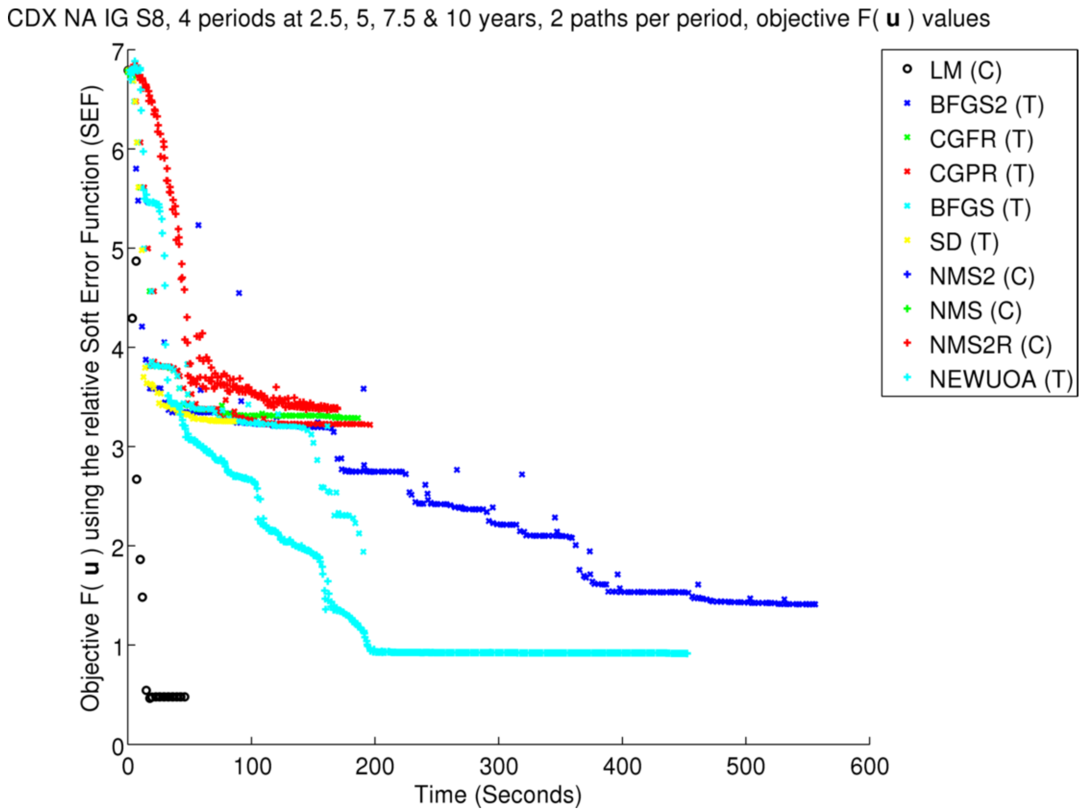

Figure A.18: CDX NA IG S8 data set from March 23, 2007 calibrated with optimization algorithms from Section 2.8 with a four period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

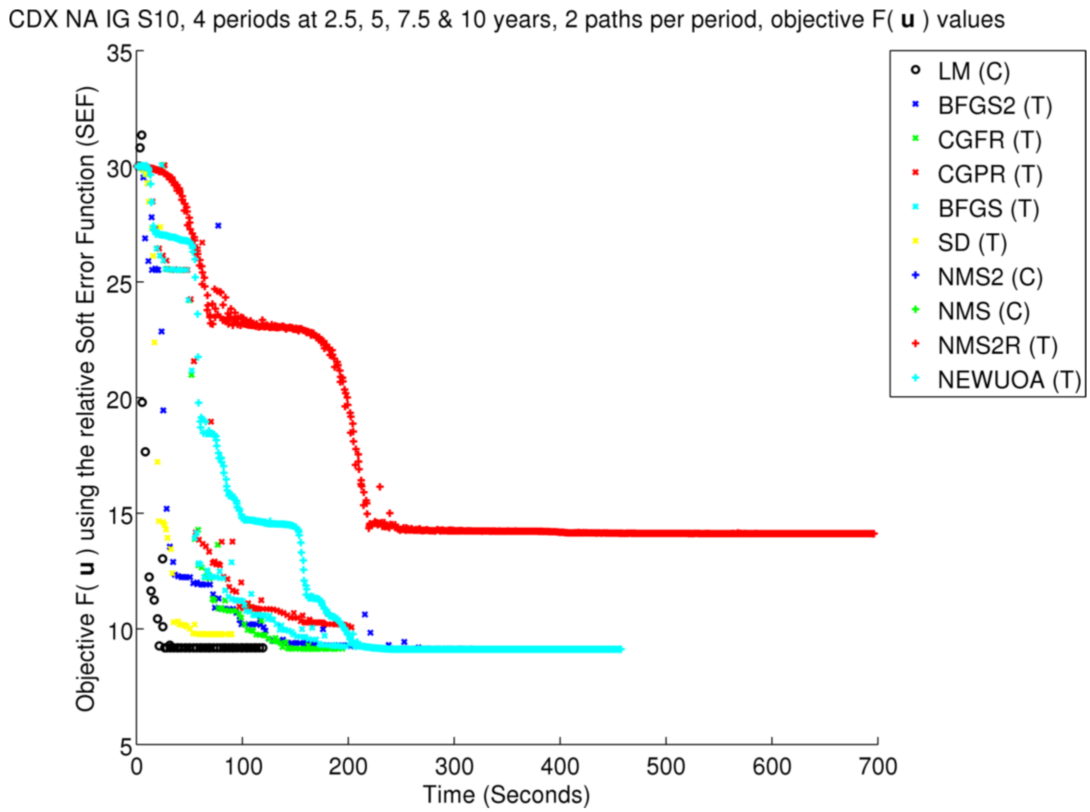

Figure A.19: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a four period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

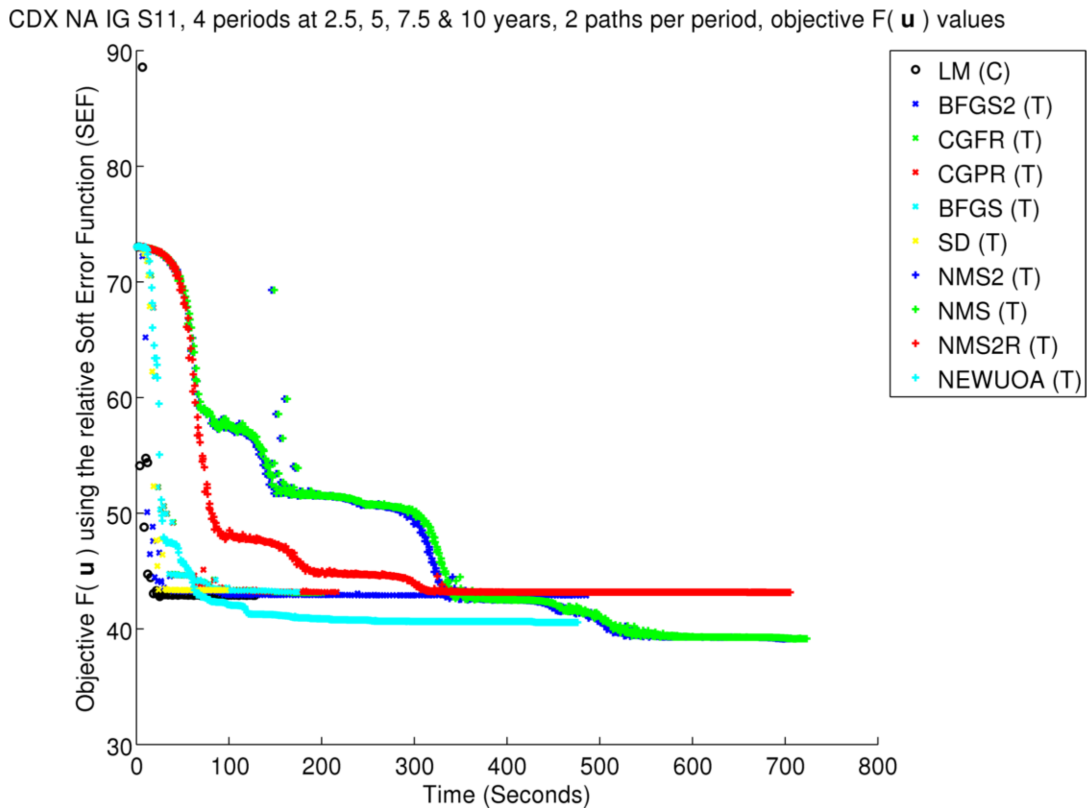

Figure A.20: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a four period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

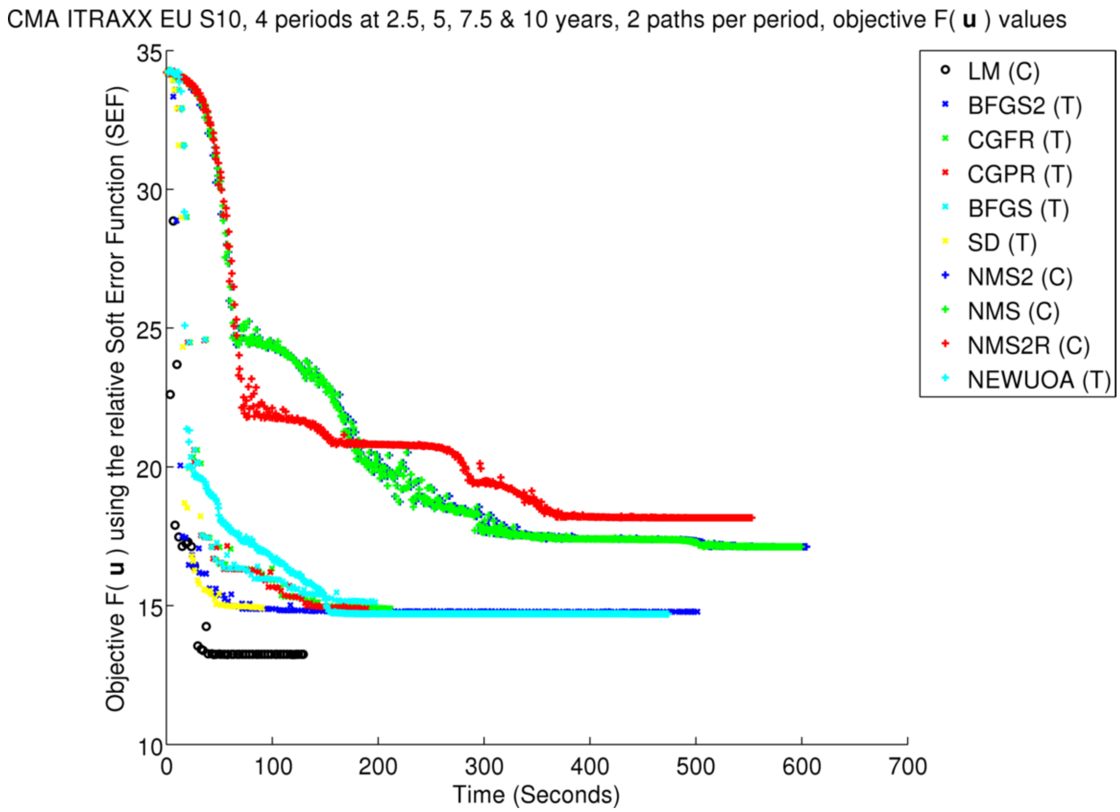

Figure A.21: CMA ITRAXX EU S10 data set from September 30, 2008 calibrated with optimization algorithms from Section 2.8 with a four period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

A.2.1.5 Three Period Multi-Path Parameterization With Two Paths Per Period (T=5,7,10)

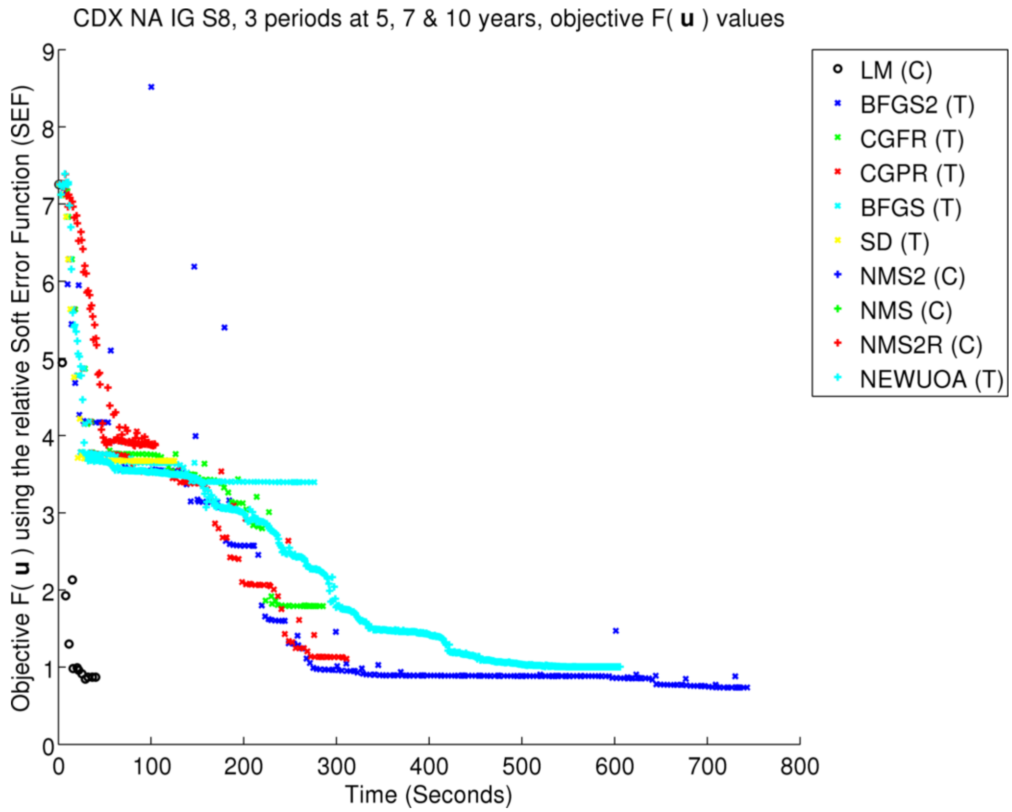

Figure A.22: CDX NA IG S8 data set from March 23, 2007 calibrated with optimization algorithms from Section 2.8 with a three period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

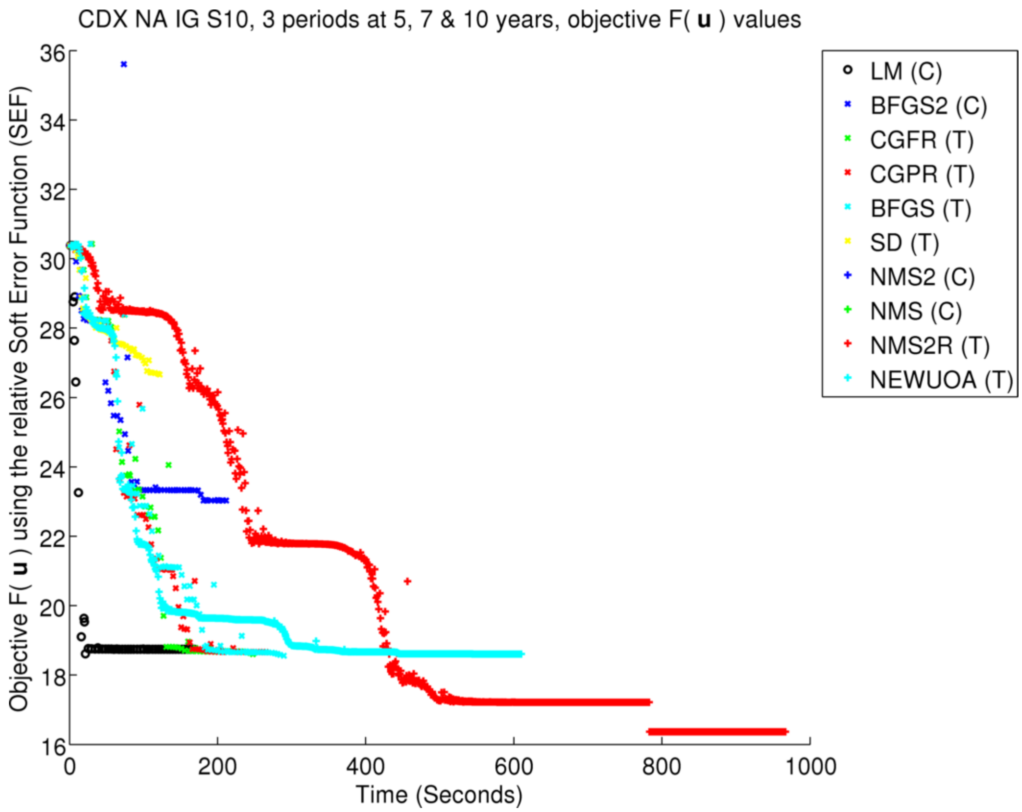

Figure A.23: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a three period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

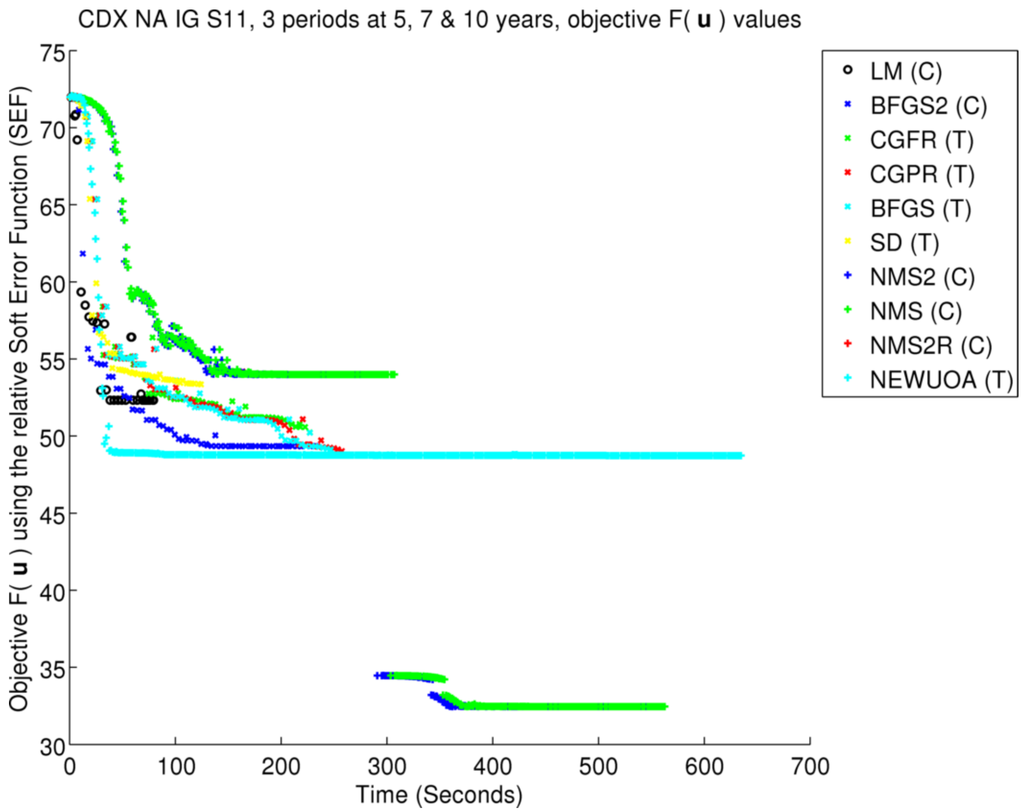

Figure A.24: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a three period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

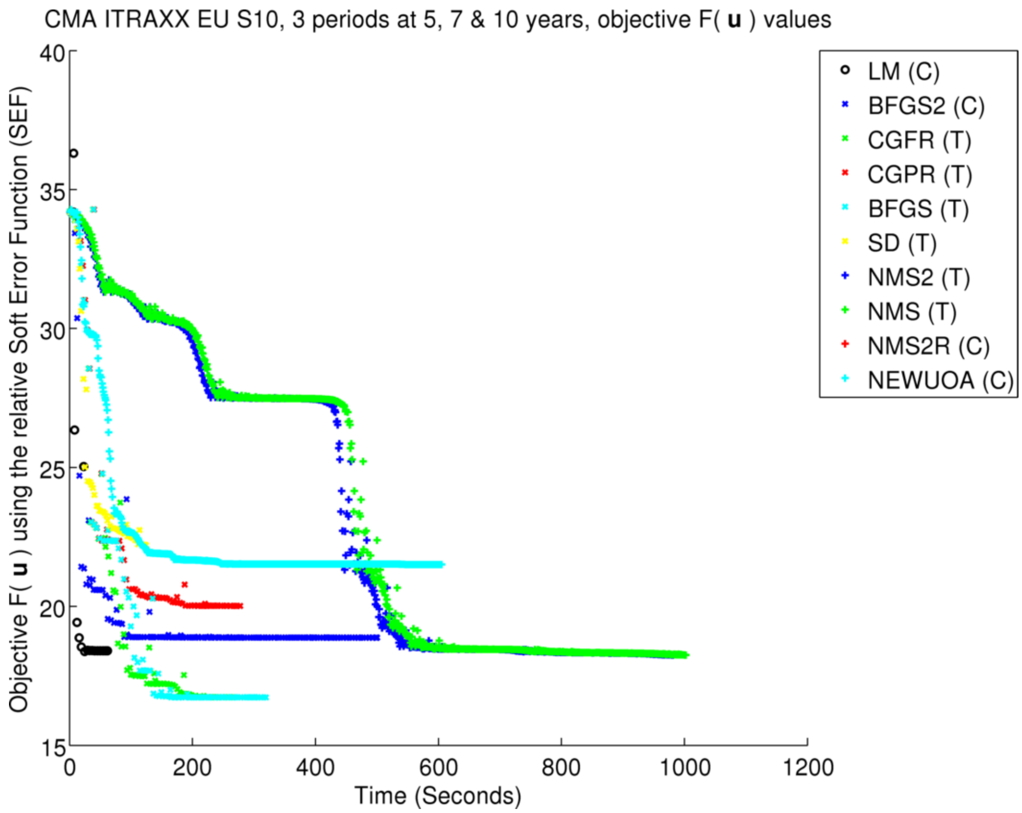

Figure A.25: CMA ITRAXX EU S10 data set from September 30, 2008 calibrated with optimization algorithms from Section 2.8 with a three period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

A.2.1.6 Two Period Multi-Path Parameterization With Two Paths Per Period (T=5,10)

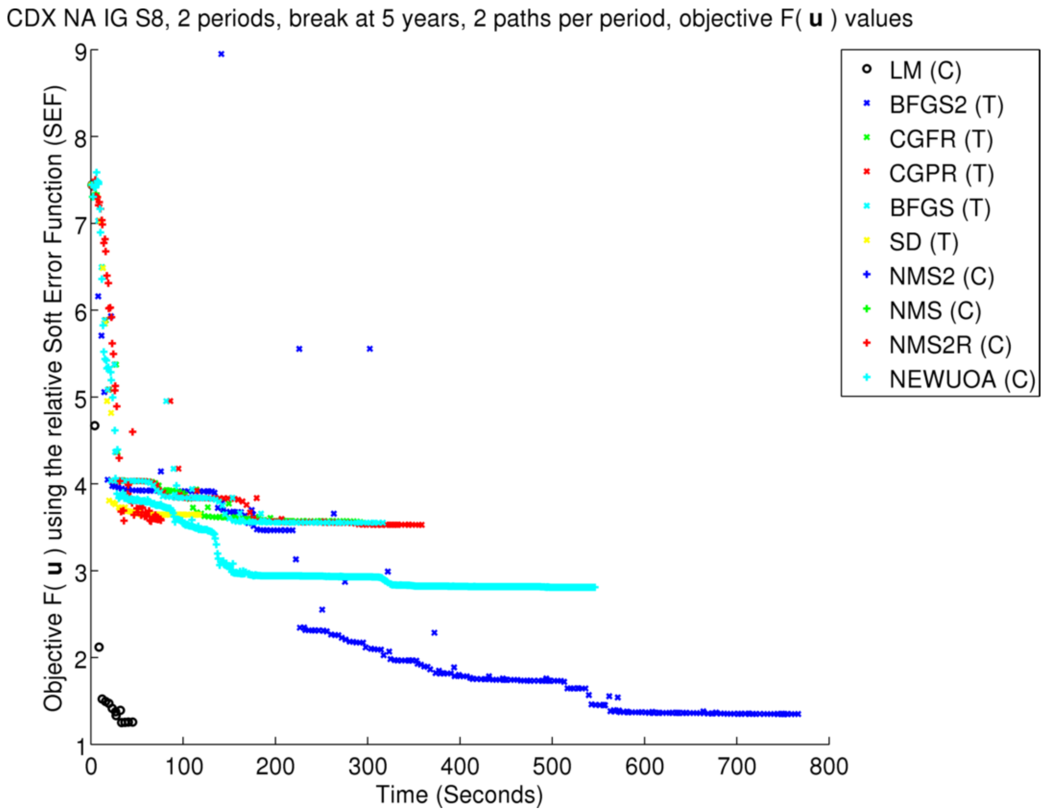

Figure A.26: CDX NA IG S8 data set from March 23, 2007 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

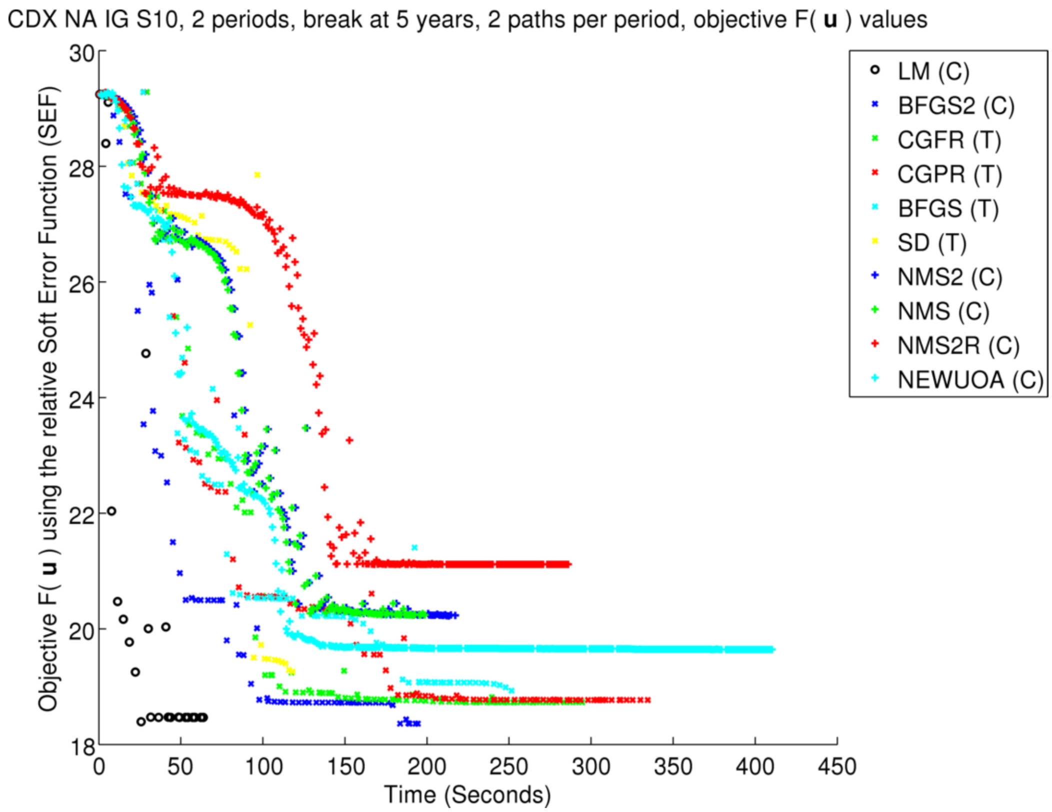

Figure A.27: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

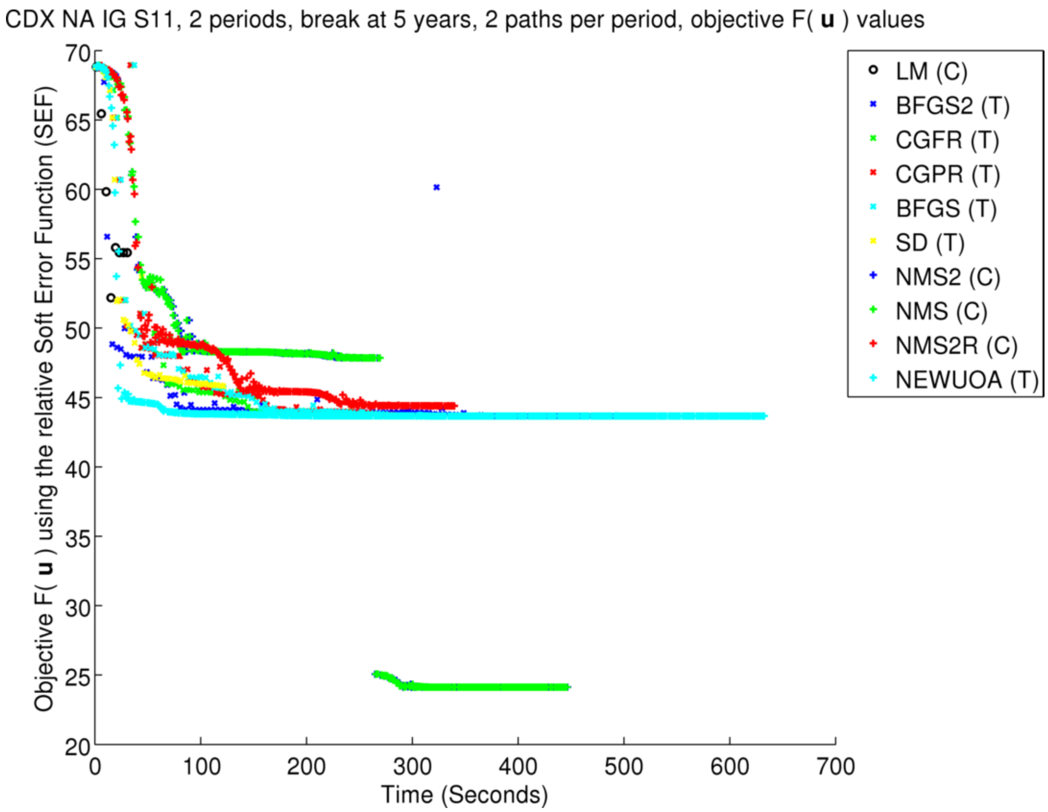

Figure A.28: CDX NA IG S11 data set from November 9, 2008 calibrated with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

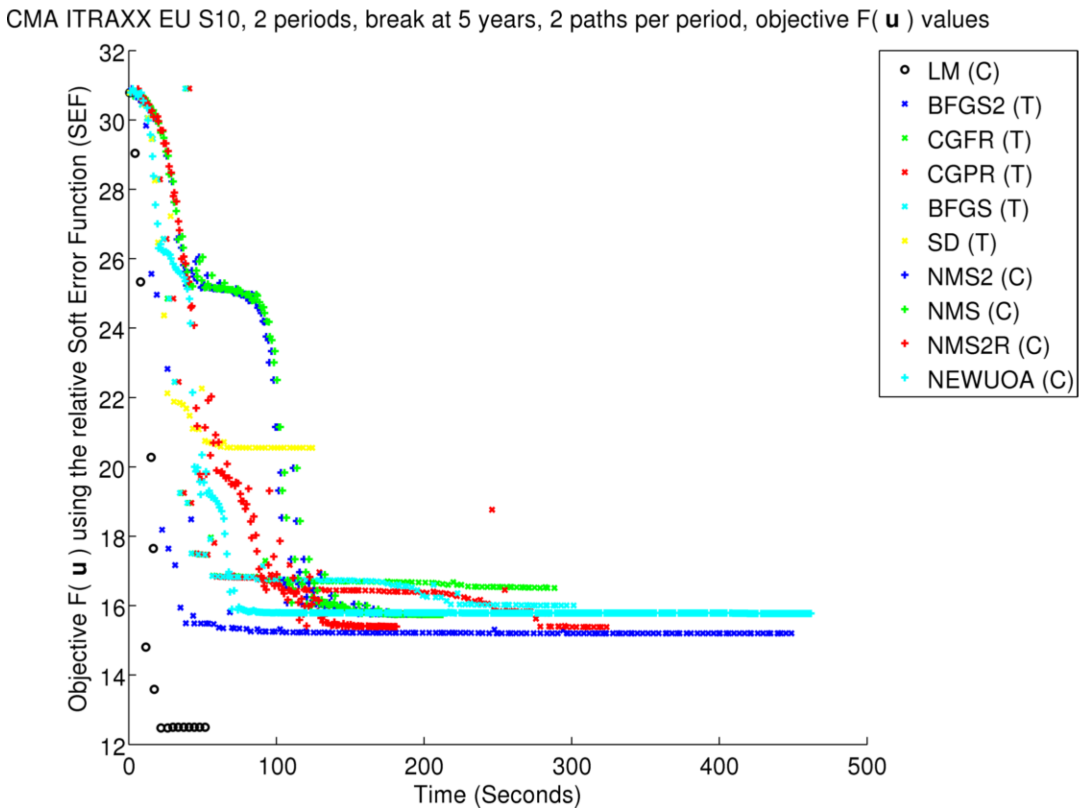

Figure A.29: CMA ITRAXX EU S10 data set calibrated on September 30, 2008 with optimization algorithms from Section 2.8 with a two period multi-path parameterization with two paths per period. Derivative-free algorithms were limited to 500 iterations, and derivative-based algorithms were limited to 40 iterations. Letters in brackets denote convergence: (C) denotes that the algorithm has converged on its own with the user-specified convergence test, (T) denotes that the algorithm has exceeded the aforementioned number of iterations before convergence and was terminated by the user, and (E) denotes for derivative-based algorithms only that the optimization algorithm encountered the value $\beta_i = 1$ during optimization and signaled for termination.

A.3 Tables

Tables A.2 to A.19 on pages 128–145 show the Multi-period Single-factor Copula Model (MSCM) calibration results. Each table shows the MSCM expected spreads (2.37) and market spreads, respectively, for each tranche and maturity. These tables also show the calibrated multi-path parameters γ_j and probabilities ρ_j, as discussed in Section 3.1. For completeness, we also calibrated the Hull Copula [1] on the same market data, and quote the Hull Copula tranche implied copula correlation parameter β next to γ_j and ρ_j.

Data set	Maturities (years)	Tranches (%)	Date range	# of CDO quotes
CDX NA IG S8	5, 7, 10	3-7-10-15-30	23/3/07-22/9/08	12
CDX NA IG S10	2, 3, 4, 5, 6, 7, 8, 9, 10	7-10-15-30	8/12/08-9/1/09	27
CDX NA IG S11	2, 3, 4, 5, 6, 7, 8, 9, 10	3-4-5-6-7-10-15-30	9/10/08-5/11/08	63
CMA ITRAXX EU S10	2, 3, 4, 5, 6, 7, 8, 9, 10	3-4-5-6-9-12-22	30/9/08-15/10/08	54

Table A.1: Data sets used for model calibration.

Tranche (%)	$T = 5$	$T = 7$	$T = 10$
$3 - 7$	$6.963469e+01, 9.59300e+01$	$2.319174e+02, 2.272500e+02$	$5.552026e+02, 4.596000e+02$
$7 - 10$	$1.781014e+01, 1.781000e+01$	$4.753623e+01, 4.754000e+01$	$1.265733e+02, 1.152900e+02$
$10 - 15$	$8.391069e+00, 8.390000e+00$	$2.451807e+01, 2.219000e+01$	$4.634550e+01, 5.442000e+01$
$15 - 30$	$2.062190e+01, 3.700000e+00$	$8.056089e+00, 8.010000e+00$	$1.558307e+01, 1.799000e+01$

γ_j	$2.446253e-01$	$1.651777e-01$
ρ_j	$7.690422e-01$	$6.404671e-01$

β	$3.472154e-01$

Table A.2: CDX NA IG S8 CDO quotes from March 23, 2007 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$7 - 10$	$4.171151e+02, 3.434000e+02$	$1.242569e+03, 5.366000e+02$	$1.995952e+03, 6.914000e+02$
$10 - 15$	$1.672082e+02, 1.672000e+02$	$2.333103e+02, 2.644000e+02$	$4.855102e+02, 3.240000e+02$
$15 - 30$	$1.228789e+02, 8.540000e+01$	$1.172311e+02, 1.004000e+02$	$1.149202e+02, 1.230000e+02$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$7 - 10$	$2.321260e+03, 1.095000e+03$	$2.277870e+03, 8.624000e+02$	$2.240827e+03, 1.110000e+03$
$10 - 15$	$8.441865e+02, 6.000000e+02$	$7.889452e+02, 4.282000e+02$	$7.466786e+02, 6.000000e+02$
$15 - 30$	$1.205727e+02, 1.765000e+02$	$1.472234e+02, 1.238000e+02$	$1.657890e+02, 1.615000e+02$

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$7 - 10$	$2.208661e+03, 9.230000e+02$	$2.180385e+03, 9.192000e+02$	$2.155283e+03, 1.123000e+03$
$10 - 15$	$7.133238e+02, 4.448000e+02$	$6.864255e+02, 4.388000e+02$	$6.643147e+02, 6.100000e+02$
$15 - 30$	$1.794281e+02, 1.356000e+02$	$1.898622e+02, 1.368000e+02$	$1.980794e+02, 1.640000e+02$

γ_j	$4.216326e-02$	$2.620157e-04$
ρ_j	$7.755412e-01$	$2.878358e-03$

β	$2.462884e-01$

Table A.3: CDX NA IG S10 CDO quotes from December 8, 2008 calibrated with the NEWUOA algorithm with at most 500 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$3 - 4$	$9.112495e + 02, 1.166200e + 03$	$1.027632e + 03, 1.392000e + 03$	$1.043791e + 03, 1.513000e + 03$
$4 - 5$	$6.874172e + 02, 7.760000e + 02$	$8.228950e + 02, 9.638000e + 02$	$8.610125e + 02, 1.101400e + 03$
$5 - 6$	$5.119236e + 02, 5.510000e + 02$	$6.246075e + 02, 7.176000e + 02$	$6.863784e + 02, 8.258000e + 02$
$6 - 7$	$3.932699e + 02, 3.828000e + 02$	$4.644138e + 02, 5.306000e + 02$	$5.366568e + 02, 6.210000e + 02$
$7 - 10$	$3.039951e + 02, 2.452000e + 02$	$3.240323e + 02, 3.370000e + 02$	$3.661498e + 02, 4.092000e + 02$
$10 - 15$	$2.769845e + 02, 1.372000e + 02$	$2.788507e + 02, 1.750000e + 02$	$2.820164e + 02, 2.048000e + 02$
$15 - 30$	$2.738477e + 02, 6.840000e + 01$	$2.738823e + 02, 8.200000e + 01$	$2.739071e + 02, 9.140000e + 01$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$3 - 4$	$1.038387e + 03, 1.569800e + 03$	$1.071043e + 03, 1.555200e + 03$	$1.154009e + 03, 1.533600e + 03$
$4 - 5$	$8.591407e + 02, 1.204800e + 03$	$8.894737e + 02, 1.211600e + 03$	$9.312440e + 02, 1.193800e + 03$
$5 - 6$	$7.002374e + 02, 9.482000e + 02$	$7.424589e + 02, 9.818000e + 02$	$7.725156e + 02, 9.778000e + 02$
$6 - 7$	$5.726113e + 02, 7.312000e + 02$	$6.218155e + 02, 7.648000e + 02$	$6.531244e + 02, 7.718000e + 02$
$7 - 10$	$4.139102e + 02, 5.411700e + 02$	$4.546729e + 02, 5.602000e + 02$	$4.832576e + 02, 5.991700e + 02$
$10 - 15$	$2.918409e + 02, 2.093350e + 02$	$3.065332e + 02, 2.276000e + 02$	$3.187859e + 02, 2.423350e + 02$
$15 - 30$	$2.739398e + 02, 9.116500e + 01$	$2.608399e + 02, 9.220000e + 01$	$2.513010e + 02, 9.450000e + 01$

Table A.4: CDX NA IG S11 CDO quotes from October 9, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (6 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.5 for the rest of the calibration results.

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$3 - 4$	$1.264872e+03, 1.531600e+03$	$1.319561e+03, 1.529800e+03$	$1.345049e+03, 1.526800e+03$
$4 - 5$	$1.027184e+03, 1.193600e+03$	$1.099740e+03, 1.195000e+03$	$1.135618e+03, 1.198200e+03$
$5 - 6$	$8.390280e+02, 9.828000e+02$	$9.159837e+02, 9.838000e+02$	$9.660636e+02, 9.866000e+02$
$6 - 7$	$6.981747e+02, 7.936000e+02$	$7.634865e+02, 8.052000e+02$	$8.222338e+02, 8.130000e+02$
$7 - 10$	$5.175872e+02, 5.972000e+02$	$5.549328e+02, 6.106000e+02$	$6.002549e+02, 6.225000e+02$
$10 - 15$	$3.389540e+02, 2.460000e+02$	$3.580832e+02, 2.580000e+02$	$3.769865e+02, 2.646650e+02$
$15 - 30$	$2.483687e+02, 9.860000e+01$	$2.471975e+02, 1.018000e+02$	$2.472653e+02, 1.006650e+02$

γ_j	$9.177589e-03$	$1.080036e-01$
ρ_j	$7.163328e-02$	$2.147276e-01$

β	$7.849448e-01$

Table A.5: CDX NA IG S11 CDO quotes from October 9, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.4 for the rest of the calibration results. Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$3 - 4$	$1.971312e+02, 5.338000e+02$	$4.699358e+02, 7.306000e+02$	$7.510163e+02, 8.108000e+02$
$4 - 5$	$1.515959e+02, 3.580000e+02$	$2.665896e+02, 5.088000e+02$	$4.898323e+02, 6.142000e+02$
$5 - 6$	$1.432932e+02, 2.694000e+02$	$1.893628e+02, 3.760000e+02$	$3.140653e+02, 4.658000e+02$
$6 - 9$	$1.356931e+02, 1.804000e+02$	$1.623888e+02, 2.538000e+02$	$1.939902e+02, 3.044000e+02$
$9 - 12$	$1.284883e+02, 1.112000e+02$	$1.558976e+02, 1.490000e+02$	$1.696817e+02, 1.836000e+02$
$12 - 22$	$1.141126e+02, 7.380000e+01$	$1.460608e+02, 8.800000e+01$	$1.619296e+02, 1.002000e+02$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$3 - 4$	$8.987232e+02, 8.398000e+02$	$9.033317e+02, 8.870000e+02$	$9.885819e+02, 9.552000e+02$
$4 - 5$	$6.950106e+02, 6.492000e+02$	$7.142095e+02, 6.812000e+02$	$7.389687e+02, 7.278000e+02$
$5 - 6$	$5.116953e+02, 5.118000e+02$	$5.507475e+02, 5.508000e+02$	$5.727475e+02, 5.852000e+02$
$6 - 9$	$2.765212e+02, 3.816300e+02$	$3.109065e+02, 4.056000e+02$	$3.490481e+02, 4.222700e+02$
$9 - 12$	$1.813876e+02, 2.057500e+02$	$1.806878e+02, 2.210000e+02$	$1.918755e+02, 2.300900e+02$
$12 - 22$	$1.714079e+02, 9.554000e+01$	$1.644456e+02, 1.090000e+02$	$1.595086e+02, 1.118400e+02$

Table A.6: CMA ITRAXX EU S10 CDO quotes from September 30, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years on (4 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.7 for the rest of the calibration results.

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$3 - 4$	$1.154747e+03, 1.006800e+03$	$1.267242e+03, 1.026400e+03$	$1.313285e+03, 1.038800e+03$
$4 - 5$	$8.381511e+02, 7.780000e+02$	$9.636112e+02, 8.138000e+02$	$1.047491e+03, 8.422000e+02$
$5 - 6$	$6.171524e+02, 6.194000e+02$	$7.108161e+02, 6.576000e+02$	$8.116069e+02, 6.936000e+02$
$6 - 9$	$3.816019e+02, 4.394000e+02$	$4.211501e+02, 4.594000e+02$	$4.817304e+02, 4.817950e+02$
$9 - 12$	$2.133170e+02, 2.422000e+02$	$2.386380e+02, 2.540000e+02$	$2.652346e+02, 2.610900e+02$
$12 - 22$	$1.565227e+02, 1.222000e+02$	$1.557733e+02, 1.280000e+02$	$1.575275e+02, 1.300750e+02$

γ_j	$2.541558e-03$	$1.215005e-02$
ρ_j	$2.730079e-01$	$6.024393e-01$

β	$7.678160e-01$

Table A.7: CMA ITRAXX EU S10 CDO quotes from September 30, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years on (4 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.6 for the rest of the calibration results. Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 5$	$T = 7$	$T = 10$
$3 - 7$	$9.583329e + 01, 9.59300e + 01$	$2.296175e + 02, 2.272500e + 02$	$4.645405e + 02, 4.596000e + 02$
$7 - 10$	$1.781181e + 01, 1.781000e + 01$	$5.084513e + 01, 4.754000e + 01$	$1.326335e + 02, 1.152900e + 02$
$10 - 15$	$8.389689e + 00, 8.390000e + 00$	$2.219097e + 01, 2.219000e + 01$	$6.865014e + 01, 5.442000e + 01$
$15 - 30$	$1.228459e + 00, 3.700000e + 00$	$6.604137e + 00, 8.010000e + 00$	$1.799226e + 01, 1.799000e + 01$

γ_j	$1.746384e - 01$	$5.073824e - 01$	$4.330975e - 02$	$9.300647e - 02$
ρ_j	$6.470438e - 01$	$2.216772e - 01$	$6.556171e - 01$	$5.099330e - 01$

β	$3.472154e - 01$

Table A.8: CDX NA IG S8 CDO quotes from March 23, 2007 calibrated with the BFGS2 algorithm with at most 120 iterations and a 4-period 2-branch multi-path parameterization with periods every 10 years (8 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
7 – 10	$3.293171e+02, 3.434000e+02$	$9.369600e+02, 5.366000e+02$	$1.223851e+03, 6.914000e+02$
10 – 15	$4.919937e+01, 1.672000e+02$	$1.332248e+02, 2.644000e+02$	$2.917535e+02, 3.240000e+02$
15 – 30	$4.610989e+01, 8.540000e+01$	$9.297116e+01, 1.004000e+02$	$1.503951e+02, 1.230000e+02$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
7 – 10	$1.349598e+03, 1.095000e+03$	$7.183892e+02, 8.624000e+02$	$7.257858e+02, 1.110000e+03$
10 – 15	$4.730546e+02, 6.000000e+02$	$4.140835e+02, 4.282000e+02$	$4.257796e+02, 6.000000e+02$
15 – 30	$1.871029e+02, 1.765000e+02$	$3.816940e+02, 1.238000e+02$	$3.863675e+02, 1.615000e+02$

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
7 – 10	$9.098090e+02, 9.230000e+02$	$9.857235e+02, 9.192000e+02$	$1.044495e+03, 1.123000e+03$
10 – 15	$4.101139e+02, 4.448000e+02$	$4.441552e+02, 4.388000e+02$	$4.782026e+02, 6.100000e+02$
15 – 30	$3.001410e+02, 1.356000e+02$	$2.908705e+02, 1.368000e+02$	$2.845296e+02, 1.640000e+02$

γ_j	$9.592274e-01$	$1.901607e-02$	$2.016594e-02$	$1.606518e-01$
ρ_j	$1.600940e-01$	$2.479793e-01$	$1.579903e-01$	$2.455560e-01$

β	$2.462884e-01$

Table A.9: CDX NA IG S10 CDO quotes from December 8, 2008 calibrated with the BFGS2 algorithm with at most 120 iterations and a 4-period 2-branch multi-path parameterization with periods every 2.5 years (8 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$3 - 4$	$2.757893e+03, 1.166200e+03$	$3.343906e+03, 1.392000e+03$	$3.137049e+03, 1.513000e+03$
$4 - 5$	$1.317884e+03, 7.760000e+02$	$1.936606e+03, 9.638000e+02$	$1.778813e+03, 1.101400e+03$
$5 - 6$	$5.596324e+02, 5.510000e+02$	$1.027065e+03, 7.176000e+02$	$9.820061e+02, 8.258000e+02$
$6 - 7$	$2.532014e+02, 3.828000e+02$	$5.254714e+02, 5.306000e+02$	$5.687855e+02, 6.210000e+02$
$7 - 10$	$1.467174e+02, 2.452000e+02$	$2.410680e+02, 3.370000e+02$	$3.220927e+02, 4.092000e+02$
$10 - 15$	$1.372158e+02, 1.372000e+02$	$1.947249e+02, 1.750000e+02$	$2.657044e+02, 2.048000e+02$
$15 - 30$	$1.372060e+02, 6.840000e+01$	$1.944973e+02, 8.200000e+01$	$2.643459e+02, 9.140000e+01$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$3 - 4$	$2.976777e+03, 1.569800e+03$	$3.130798e+03, 1.555200e+03$	$3.011114e+03, 1.533600e+03$
$4 - 5$	$1.667029e+03, 1.204800e+03$	$1.761781e+03, 1.211600e+03$	$1.681060e+03, 1.193800e+03$
$5 - 6$	$9.483095e+02, 9.482000e+02$	$9.990641e+02, 9.818000e+02$	$9.670925e+02, 9.778000e+02$
$6 - 7$	$5.943033e+02, 7.312000e+02$	$6.147725e+02, 7.648000e+02$	$6.152655e+02, 7.718000e+02$
$7 - 10$	$3.814526e+02, 5.411700e+02$	$3.753859e+02, 5.602000e+02$	$3.949337e+02, 5.991700e+02$
$10 - 15$	$3.112086e+02, 2.093350e+02$	$2.937197e+02, 2.276000e+02$	$3.202826e+02, 2.423350e+02$
$15 - 30$	$3.052287e+02, 9.116500e+01$	$2.867295e+02, 9.220000e+01$	$3.127765e+02, 9.450000e+01$

Table A.10: CDX NA IG S11 CDO quotes from October 9, 2008 calibrated with the BFGS2 algorithm with at most 120 iterations and a 4-period 2-branch multi-path parameterization with periods every 2.5 years (8 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.11 for the rest of the calibration results.

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$3 - 4$	$3.355151e+03, 1.531600e+03$	$3.354564e+03, 1.529800e+03$	$3.354027e+03, 1.526800e+03$
$4 - 5$	$2.257550e+03, 1.193600e+03$	$2.255805e+03, 1.195000e+03$	$2.254208e+03, 1.198200e+03$
$5 - 6$	$1.691191e+03, 9.828000e+02$	$1.686988e+03, 9.838000e+02$	$1.683145e+03, 9.866000e+02$
$6 - 7$	$1.368216e+03, 7.936000e+02$	$1.359894e+03, 8.052000e+02$	$1.352346e+03, 8.130000e+02$
$7 - 10$	$8.767251e+02, 5.972000e+02$	$8.640425e+02, 6.106000e+02$	$8.530512e+02, 6.225000e+02$
$10 - 15$	$2.960900e+02, 2.460000e+02$	$3.157284e+02, 2.580000e+02$	$3.312424e+02, 2.646650e+02$
$15 - 30$	$2.125427e+02, 9.860000e+01$	$2.400985e+02, 1.018000e+02$	$2.613760e+02, 1.006650e+02$

γ_j	$6.153181e-04$	$9.844330e-05$	$1.768339e-02$	$9.344992e-03$
ρ_j	$7.428151e-01$	$5.411367e-02$	$1.728121e-03$	$4.427084e-03$

β	$7.849448e-01$

Table A.11: CDX NA IG S11 CDO quotes from October 9, 2008 calibrated with the BFGS2 algorithm with at most 120 iterations and a 4-period 2-branch multi-path parameterization with periods every 2.5 years (8 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.10 for the rest of the calibration results. Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$3 - 4$	$5.277848e + 02, 5.338000e + 02$	$7.676456e + 02, 7.306000e + 02$	$8.806903e + 02, 8.108000e + 02$
$4 - 5$	$3.574063e + 02, 3.580000e + 02$	$4.825509e + 02, 5.088000e + 02$	$5.841954e + 02, 6.142000e + 02$
$5 - 6$	$2.826125e + 02, 2.694000e + 02$	$3.575465e + 02, 3.760000e + 02$	$4.319646e + 02, 4.658000e + 02$
$6 - 9$	$1.803211e + 02, 1.804000e + 02$	$2.517703e + 02, 2.538000e + 02$	$3.053002e + 02, 3.044000e + 02$
$9 - 12$	$7.967881e + 01, 1.112000e + 02$	$1.564955e + 02, 1.490000e + 02$	$2.196506e + 02, 1.836000e + 02$
$12 - 22$	$3.600491e + 01, 7.380000e + 01$	$9.916554e + 01, 8.800000e + 01$	$1.700307e + 02, 1.002000e + 02$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$3 - 4$	$1.006077e + 03, 8.398000e + 02$	$1.033219e + 03, 8.870000e + 02$	$1.028869e + 03, 9.552000e + 02$
$4 - 5$	$7.259694e + 02, 6.492000e + 02$	$7.190715e + 02, 6.812000e + 02$	$7.276984e + 02, 7.278000e + 02$
$5 - 6$	$5.510413e + 02, 5.118000e + 02$	$5.267492e + 02, 5.508000e + 02$	$5.404951e + 02, 5.852000e + 02$
$6 - 9$	$3.683028e + 02, 3.816300e + 02$	$3.313008e + 02, 4.056000e + 02$	$3.557167e + 02, 4.222700e + 02$
$9 - 12$	$2.603291e + 02, 2.057500e + 02$	$2.493101e + 02, 2.210000e + 02$	$2.805370e + 02, 2.300900e + 02$
$12 - 22$	$2.128160e + 02, 9.554000e + 01$	$2.331219e + 02, 1.090000e + 02$	$2.643448e + 02, 1.118400e + 02$

Table A.12: CMA ITRAXX EU S10 CDO quotes from September 30, 2008 calibrated with the BFGS2 algorithm with at most 120 iterations and a 4-period 2-branch multi-path parameterization with periods every 2.5 years (8 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.13 for the rest of the calibration results.

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$3 - 4$	$1.638052e+03, 1.006800e+03$	$1.629291e+03, 1.026400e+03$	$1.620436e+03, 1.038800e+03$
$4 - 5$	$1.268411e+03, 7.780000e+02$	$1.255841e+03, 8.138000e+02$	$1.242691e+03, 8.422000e+02$
$5 - 6$	$9.233313e+02, 6.194000e+02$	$9.155231e+02, 6.576000e+02$	$9.062021e+02, 6.936000e+02$
$6 - 9$	$4.450421e+02, 4.394000e+02$	$4.601337e+02, 4.594000e+02$	$4.726590e+02, 4.817950e+02$
$9 - 12$	$2.176089e+02, 2.422000e+02$	$2.449213e+02, 2.540000e+02$	$2.684539e+02, 2.610900e+02$
$12 - 22$	$1.923580e+02, 1.222000e+02$	$2.192575e+02, 1.280000e+02$	$2.402296e+02, 1.300750e+02$

γ_j	$2.488802e-01$	$2.267539e-02$	$1.375008e-02$	$8.320620e-03$
ρ_j	$3.871771e-01$	$2.246750e-01$	$2.854114e-02$	$4.513781e-02$

β	$7.678160e-01$

Table A.13: CMA ITRAXX EU S10 CDO quotes from September 30, 2008 calibrated with the BFGS2 algorithm with at most 120 iterations and a 4-period 2-branch multi-path parameterization with periods every 2.5 years (8 model parameters) using the relative soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.12 for the rest of the calibration results. Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 5$	$T = 7$	$T = 10$
$3 - 7$	$9.593012e+01, 9.59300e+01$	$2.050810e+02, 2.272500e+02$	$4.596002e+02, 4.596000e+02$
$7 - 10$	$1.781235e+01, 1.781000e+01$	$4.753623e+01, 4.754000e+01$	$1.123107e+02, 1.152900e+02$
$10 - 15$	$5.770887e+00, 8.390000e+00$	$4.764192e+01, 2.219000e+01$	$5.183622e+01, 5.442000e+01$
$15 - 30$	$7.727291e-01, 3.700000e+00$	$1.376227e+01, 8.010000e+00$	$2.710064e+01, 1.799000e+01$

γ_j	$6.514573e-01$	$1.418288e-02$
ρ_j	$4.297662e-01$	$5.797056e-01$

β	$2.763489e-01$

Table A.14: CDX NA IG S8 CDO quotes from March 23, 2007 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the absolute soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T=2$	$T=3$	$T=4$
$7-10$	$4.556497e+02, 3.434000e+02$	$6.841100e+02, 5.366000e+02$	$8.968089e+02, 6.914000e+02$
$10-15$	$3.869937e+02, 1.672000e+02$	$3.909647e+02, 2.644000e+02$	$4.548640e+02, 3.240000e+02$
$15-30$	$3.862776e+02, 8.540000e+01$	$3.712907e+02, 1.004000e+02$	$3.614672e+02, 1.230000e+02$

Tranche (%)	$T=5$	$T=6$	$T=7$
$7-10$	$9.836096e+02, 1.095000e+03$	$9.597196e+02, 8.624000e+02$	$9.311738e+02, 1.110000e+03$
$10-15$	$5.521351e+02, 6.000000e+02$	$5.888028e+02, 4.282000e+02$	$5.999999e+02, 6.000000e+02$
$15-30$	$3.551107e+02, 1.765000e+02$	$3.137035e+02, 1.238000e+02$	$2.879493e+02, 1.615000e+02$

Tranche (%)	$T=8$	$T=9$	$T=10$
$7-10$	$9.278171e+02, 9.230000e+02$	$9.821710e+02, 9.192000e+02$	$1.083525e+03, 1.123000e+03$
$10-15$	$5.956490e+02, 4.448000e+02$	$5.893906e+02, 4.388000e+02$	$5.882335e+02, 6.100000e+02$
$15-30$	$2.700925e+02, 1.356000e+02$	$2.583050e+02, 1.368000e+02$	$2.496544e+02, 1.640000e+02$

γ_j	$2.394868e-35$	$8.784070e-11$
ρ_j	$2.278566e-01$	$1.000000e+00$

β	$9.840018e-01$

Table A.15: CDX NA IG S10 CDO quotes from December 8, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the absolute soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$3 - 4$	$1.164185e+03, 1.166200e+03$	$1.450361e+03, 1.392000e+03$	$1.497013e+03, 1.513000e+03$
$4 - 5$	$6.897159e+02, 7.760000e+02$	$1.023107e+03, 9.638000e+02$	$1.137437e+03, 1.101400e+03$
$5 - 6$	$4.166094e+02, 5.510000e+02$	$6.922639e+02, 7.176000e+02$	$8.687605e+02, 8.258000e+02$
$6 - 7$	$2.970853e+02, 3.828000e+02$	$4.666572e+02, 5.306000e+02$	$6.593318e+02, 6.210000e+02$
$7 - 10$	$2.483249e+02, 2.452000e+02$	$2.877277e+02, 3.370000e+02$	$3.887927e+02, 4.092000e+02$
$10 - 15$	$2.395681e+02, 1.372000e+02$	$2.402819e+02, 1.750000e+02$	$2.461296e+02, 2.048000e+02$
$15 - 30$	$2.391297e+02, 6.840000e+01$	$2.391319e+02, 8.200000e+01$	$2.391343e+02, 9.140000e+01$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$3 - 4$	$1.493007e+03, 1.569800e+03$	$1.529784e+03, 1.555200e+03$	$1.589534e+03, 1.533600e+03$
$4 - 5$	$1.154767e+03, 1.204800e+03$	$1.195998e+03, 1.211600e+03$	$1.235787e+03, 1.193800e+03$
$5 - 6$	$9.256304e+02, 9.482000e+02$	$9.698784e+02, 9.818000e+02$	$1.002350e+03, 9.778000e+02$
$6 - 7$	$7.598598e+02, 7.312000e+02$	$8.024890e+02, 7.648000e+02$	$8.315154e+02, 7.718000e+02$
$7 - 10$	$5.039130e+02, 5.411700e+02$	$5.443232e+02, 5.602000e+02$	$5.691327e+02, 5.991700e+02$
$10 - 15$	$2.689591e+02, 2.093350e+02$	$2.963137e+02, 2.276000e+02$	$3.162397e+02, 2.423350e+02$
$15 - 30$	$2.391640e+02, 9.116500e+01$	$2.384379e+02, 9.220000e+01$	$2.369609e+02, 9.450000e+01$

Table A.16: CDX NA IG S11 CDO quotes from October 9, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (6 model parameters) using the absolute soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.17 for the rest of the calibration results. Parameter β is the Hull Copula correlation parameter, included for reference.

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$3-4$	$1.663901e+03, 1.531600e+03$	$1.701085e+03, 1.529800e+03$	$1.719346e+03, 1.526800e+03$
$4-5$	$1.303337e+03, 1.193600e+03$	$1.353127e+03, 1.195000e+03$	$1.378514e+03, 1.198200e+03$
$5-6$	$1.052917e+03, 9.828000e+02$	$1.106331e+03, 9.838000e+02$	$1.140784e+03, 9.866000e+02$
$6-7$	$8.683257e+02, 7.936000e+02$	$9.146419e+02, 8.052000e+02$	$9.546136e+02, 8.130000e+02$
$7-10$	$5.993899e+02, 5.972000e+02$	$6.295074e+02, 6.106000e+02$	$6.627360e+02, 6.225000e+02$
$10-15$	$3.419136e+02, 2.460000e+02$	$3.634305e+02, 2.580000e+02$	$3.823784e+02, 2.646650e+02$
$15-30$	$2.402161e+02, 9.860000e+01$	$2.435280e+02, 1.018000e+02$	$2.468853e+02, 1.006650e+02$

γ_j	$5.213397e-03$	$7.759241e-02$
ρ_j	$1.966539e-01$	$1.565168e-01$

β	$7.845863e-01$

Table A.17: CDX NA IG S11 CDO quotes from October 9, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the absolute soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.16 for the rest of the calibration results. Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Tranche (%)	$T = 2$	$T = 3$	$T = 4$
$3 - 4$	$3.649178e+02, 5.338000e+02$	$5.918990e+02, 7.306000e+02$	$7.945163e+02, 8.108000e+02$
$4 - 5$	$2.977460e+02, 3.580000e+02$	$4.307943e+02, 5.088000e+02$	$6.035501e+02, 6.142000e+02$
$5 - 6$	$2.688212e+02, 2.694000e+02$	$3.487232e+02, 3.760000e+02$	$4.623612e+02, 4.658000e+02$
$6 - 9$	$2.433807e+02, 1.804000e+02$	$2.790195e+02, 2.538000e+02$	$3.212325e+02, 3.044000e+02$
$9 - 12$	$2.028563e+02, 1.112000e+02$	$2.238648e+02, 1.490000e+02$	$2.396540e+02, 1.836000e+02$
$12 - 22$	$1.428382e+02, 7.380000e+01$	$1.763577e+02, 8.800000e+01$	$1.946254e+02, 1.002000e+02$

Tranche (%)	$T = 5$	$T = 6$	$T = 7$
$3 - 4$	$9.071278e+02, 8.398000e+02$	$8.870050e+02, 8.870000e+02$	$9.225998e+02, 9.552000e+02$
$4 - 5$	$7.540201e+02, 6.492000e+02$	$7.330559e+02, 6.812000e+02$	$7.333918e+02, 7.278000e+02$
$5 - 6$	$6.132585e+02, 5.118000e+02$	$6.009504e+02, 5.508000e+02$	$5.944005e+02, 5.852000e+02$
$6 - 9$	$4.064122e+02, 3.816300e+02$	$4.060854e+02, 4.056000e+02$	$4.109076e+02, 4.222700e+02$
$9 - 12$	$2.675229e+02, 2.057500e+02$	$2.680513e+02, 2.210000e+02$	$2.734537e+02, 2.300900e+02$
$12 - 22$	$2.082299e+02, 9.554000e+01$	$2.096565e+02, 1.090000e+02$	$2.110067e+02, 1.118400e+02$

Table A.18: CMA ITRAXX EU S10 CDO quotes from September 30, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the absolute soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.19 for the rest of the calibration results.

Tranche (%)	$T = 8$	$T = 9$	$T = 10$
$3 - 4$	$9.926419e+02, 1.006800e+03$	$1.034922e+03, 1.026400e+03$	$1.047900e+03, 1.038800e+03$
$4 - 5$	$7.756009e+02, 7.780000e+02$	$8.273395e+02, 8.138000e+02$	$8.586250e+02, 8.422000e+02$
$5 - 6$	$6.099564e+02, 6.194000e+02$	$6.508451e+02, 6.576000e+02$	$6.934282e+02, 6.936000e+02$
$6 - 9$	$4.181544e+02, 4.394000e+02$	$4.337267e+02, 4.594000e+02$	$4.618652e+02, 4.817950e+02$
$9 - 12$	$2.820926e+02, 2.422000e+02$	$2.917585e+02, 2.540000e+02$	$3.027845e+02, 2.610900e+02$
$12 - 22$	$2.126034e+02, 1.222000e+02$	$2.146728e+02, 1.280000e+02$	$2.174399e+02, 1.300750e+02$

γ_j	$4.582050e-02$	$1.224857e-02$
ρ_j	$1.626674e-01$	$3.030195e-01$

β	$7.673730e-01$

Table A.19: CMA ITRAXX EU S10 CDO quotes from September 30, 2008 calibrated with the BFGS2 algorithm with at most 40 iterations and a 2-period 2-branch multi-path parameterization with periods between 5 and 10 years (4 model parameters) using the absolute soft error function (2.61) with $\epsilon = 0.0001$ and $\delta = 0.5$. CDO quote format is (model spread, market spread). See Table A.18 for the rest of the calibration results. Calibrated MSCM parameters are γ_j and ρ_j, as discussed in Section 3.1. Calibrated Hull Copula parameter β is included for reference.

Bibliography

[1] J. Hull. *Options, Futures, and Other Derivatives.* Prentice Hall, New Jersey, sixth edition, 2005.

[2] A. Kakodkar, S. Galiani, J. J′onsson, and A. Gallo. *Credit derivatives handbook 2006.* Technical report, Merrill Lynch, 2006.

[3] Vanderbilt Capital Advisors. *Calculating implied default rates from CDS spreads.* Available from http://www.vcallc.com/mailings/additions/cdsspreads2.htm.

[4] F. Abid, N. Naifar. *CDO Parameters Estimation Using Market Prices.* International Research Journal of Finance and Economics, ISSN 1450-2887 Issue 18 (2008).

[5] L. S. J. Luo. *Bootstrapping Default Probability Curves.* Journal of Credit Risk, Volume 1/Number 4, 2005.

[6] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration: Second Edition.* Dover, San Diego, 2007.

[7] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables.* Dover, New York, 1965.

[8] W. Zhang. *On Computational Methods for the Valuation of Credit Derivatives.* Doctor of Philosophy Thesis, University of Toronto, Department of Compuer Science, Canada, 2010.

[9] X. Ma. *Numerical Methods for the Valuation of Synthetic Collateralized Debt Obligations.* Doctor of Philosophy Thesis, University of Toronto, Department of Compuer Science, Canada, 2007.

[10] R. Fletcher. *Practical Methods of Optimization.* John Wiley & Sons, Toronto, 1980.

[11] K. R. Jackson, A. Kreinin, W. Zhang. *Dynamic Factor Copula Model.* Submitted to the Canadian Applied Mathematics Quarterly (CAMQ), 2010.

[12] T. V. Mikosh, S. I. Resnick and S. M. Robinson. *Numerical Optimization.* Springer, U.S.A., second edition, 2006.

[13] M. J. D. Powell. *The NEWUOA software for unconstrained optimization without derivatives.* Presented at the 40-th workshop on Large Scale Nonlinear Optimization, Erice, Italy, 2004.

[14] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, New York, third edition, 2007.

[16] GSL Team. *The GNU Scientific Library - a free numerical library licensed under the GNU GPL.* Online reference, http://www.gnu.org/software/gsl/manual/html_node, 2010.

[17] W. Chu, S. S. Keerthi and C. J. Ong. *A Unified Loss Function in Bayesian Framework for Support Vector Regression.* In Proceeding of the 18th International Conference on Machine Learning, 2001.

[18] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, NY, 1995.

[19] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* The MIT Press, London, 2002.

[20] H. Shekarforoush, M. Berthod and J. Zerubia. *Direct Search Generalized Simplex Algorithm for Optimizing Non-linear Functions.* Technical report #2535, Unit de recherche INRIA Sophia-Antipolis, France, 1995.

[21] A. R. Conn, K. Scheinberg and L. N. Vicente. *Introduction to Derivative-free Optimization.* S.I.A.M., U.S.A., 2009.

[22] A. Bultheel and R. Cools. *The Birth of Numerical Analysis.* World Scientific Publishing Co. Pte. Ltd., Singapore, 2010.

[23] M. A. Epelman. *Continuous Optimization Methods, Section 1 (IOE511/MATH562).* Course notes, proof available online http://www.personal.umich.edu/~mepelman/teaching/IOE511/Handouts/511notes07-7.pdf, 2007.

[24] M. T. Heath. *Scientific Computing: An Introductory Survey.* McGraw-Hill, New York, second edition, 2002.

[25] N. Yamashita and M. Fukushima. *On the rate of convergence of the Levenberg-Marquardt method.* Computing (Suppl. 15): 237-249 (2001).

[26] Boost development team. *Boost 1.43.0 Library Documentation.* Online reference, http://www.boost.org/doc/libs/1_43_0, 2010.

[27] OpenMP Architecture Review Board. *OpenMP Application Program Interface.* Online reference, http://www.openmp.org/mp-documents/spec30.pdf, 2008.

[28] J. Hull and A. White. *Forwards and European Options on CDO Tranches.* J. L. Rotman School of Management, Toronto, 2007.

[29] A. Kalemanova, B. Schmid and R. Werner. *The Normal Inverse Gaussian distribution for synthetic CDO pricing.* Journal of Derivatives, 14(3):80-93, 2007.

[30] G. Van Damme. *Legendre, Laguerre and Hermite - Gauss Quadrature.* Online source code, http://www.mathworks.com/matlabcentral/fileexchange/26737-legendre-laguerre-and-hermite-gauss-quadrature, 2010.

[31] G. Van Damme, private communication, 2010.

[32] Boost development team. *Examples - how to extend UBLAS.* Online reference, http://www.crystalclearsoftware.com/cgi-bin/boost_wiki/wiki.pl?Examples_-_How_To_Extend_UBLAS, 2010.

[33] C. Fingers. *A Comparison of Stochastic Default Rate Models.* RiskMetrices Journal, 1(2):4973, 2000.

[34] L. Andersen. *Portfolio Losses in Factor Models: Term Structures and Intertemporal Loss Dependence.* Working paper, available at www.defaultrisk.com, 2006.

[35] J. Sidenius. *On the Term Structure of Loss Distributions - a Forward Model Approach.* International Journal of Theoretical and Applied Finance, 10(4):749761, 2007.

[36] J. Turc, D. Benhamou, B. Herzog and M. Teyssier. *Pricing Bespoke CDOs: Latest Developments.* Cross Asset Research, Section Report, 2006.

[37] D. O′Kane and M. Livesey. *Base Correlation Explained.* Lehman Brothers, Fixed Income Quantitative Credit Research, 2004.

[38] P. Baheti and S. Morgan. *Quantitative Credit Research Quarterly.* Lehman Brothers, Fixed Income Quantitative Credit Research, Volume 2007-Q1, 2007.

[39] C.P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, U.S.A., 2004.

[40] R. Merton. *On the pricing of corporate debt: The risk structure of interest rates.* Journal of Finance, 29(2):449-470, 1974.

[41] O. Vasicek. *Probability of loss distribution.* Technical report, KMV Corporation, 1987.

[42] D. Li. *On default correlation: A copula approach.* Journal of Fixed Income, 9(4):43-54, 2000.

[43] L. Andersen, J. Sidenius, and S. Basu. *All your hedges in one basket.* Risk, 16(11):67-72, 2003.

[44] J. Hull and A. White. *Valuation of a CDO and an nth to default CDS without Monte Carlo simulation.* Journal of Derivatives, 12(2):8-23, Winter 2004.

[45] J. Laurent and J. Gregory. *Basket default swaps, CDOs and factor copulas.* Journal of Risk, 7(4):103-122, 2005.

[46] D. Duffie and K. Singleton. *Modeling term structures of defaultable bonds.* Review of Financial Studies, 12(4):687-720, 1999.

[47] R. Jarrow and S. Turnbull. *Pricing derivatives on financial securities subject to credit risk.* Journal of Finance, 50(1):53-85, 1995.

[48] R. Bartle. *Introduction to Measure Theory.* Wiley, U.S.A., 1966.

[49] M. Spivak. *Calculus.* Publish or Perish Inc., Houston, 1994.