Credit Derivatives – Pricing and Bootstrapping

Author: Mi Sun

University College London

MSc Financial Computing Project 2009-2010

Project Supervisor: Christopher Clack

Submission Date: 15 September 2010

Abstract

This project explores different pricing models for credit derivatives and implements the bootstrapping method for the survival curve and base correlation. The credit derivatives discussed in this report include credit default swap (CDS) and collateralized debt obligation (CDO). The main aim of the project is to make sure that there is no arbitrage opportunity between quoted (CDS and index CDO tranches) instruments and unquoted (bespoke CDO tranches) instruments. In doing so, the objectives include obtaining the CDS survival curve from market spreads. Then together with the standardized index CDO tranche spreads, the CDS survival curve is an input parameter for obtaining the base correlation. Finally using the base correlation, the bespoke (i.e. customized) CDO tranche can be priced consistently.

There are seven chapters. The first one introduces the background and underlying logic of the whole project. The second chapter starts to introduce the pricing of credit default swaps (CDSs), and discusses the bootstrapping and interpolation of the CDS survival curve. The third chapter moves on to introduce the pricing of collateralized debt obligations (CDOs), and investigates two models for building the CDO tranche survival curve. The fourth chapter discusses the latest development in credit derivative pricing in the market, namely the stochastic recovery rate. The fifth chapter looks into the correlation between credits and implements the bootstrapping method for base correlation. The sixth chapter briefly discusses Matlab programming techniques. The last chapter provides a summary and critical evaluation of the project.

Acknowledgements

First of all, I am very grateful to Afzal Tarar at PwC China for providing me the opportunity to meet Colin Lawrence at the Financial Service Authority.

I want to particularly thank Colin Lawrence for introducing me to the Quantitative Modeling Team (formerly the Analytics and Risk Technology Team) at the FSA. This team is definitely the perfect place to hold the project for my degree in financial computing.

I would like to thank all the people from the Quantitative Modeling Team. Each of them has provided me with a lot of help and advice.

I also want to say millions of thanks to my dear friend Fatema and my parents for providing me immense support at the most difficult time of my personal life.

Finally, I would like to reserve my biggest thanks for Igor Tsatskis, Don Stewart, and Dan Oprescu at the Quantitative Modeling Team for their very generous support and great encouragement during the project.

Table of Contents

Chapter 1. Background

This project explores different pricing models for credit derivatives and implements the bootstrapping method for the survival curve and base correlation. The credit derivatives discussed in this report include credit default swap (CDS) and collateralized debt obligation (CDO). The main aim of the project is to make sure that there is no arbitrage opportunity between quoted (CDS and index CDO tranches) instruments and unquoted (bespoke CDO tranches) instruments. In doing so, the objectives include obtaining the CDS survival curve from market spreads. Then together with the standardized index CDO tranche spreads, the CDS survival curve is an input parameter for obtaining the base correlation. Finally using the base correlation, the bespoke (i.e. customized) CDO tranche can be priced consistently.

All models and algorithms in this project implemented in Matlab[1]. It is a programming language developed by MathWorks that is designed specifically for technical computing. It has very strong abilities in handling numerical calculations, data visualization, and algorithm development. Hence it is very suitable for this project.

1.1 Credit Derivatives

Credit derivatives were introduced to the market in the mid-1990s. Since then, the size of the credit derivative market has been dramatically growing. Initially, it was primarily used by banks to hedge their credit risk of bonds or loans. Nowadays, the uses of credit derivatives include increasing asset liquidity, diversifying credit risk, and diversifying investment portfolios (O'Kane 2008, p. 2). It is important to price all the instruments consistently so that the market is fair and arbitrage-free.

The credit default swap (CDS) is the simplest yet the most important single-name credit derivative. The expression 'single-name' means that one CDS contract is only exposed to the default risk of one credit. A CDS is a bilateral over-the-counter contract between the protection seller and the protection buyer. During the contract, the protection buyer will make periodical payments (referred to as the premium leg), usually quarterly, to the protection

[1] Matlab stands for Matrix Laboratory

seller until the occurrence of a credit event[2] or the contract maturity date, whichever comes sooner. In return for the stream of payments, the protection seller will compensate the protection buyer (the payment is referred to as the protection leg) in the case of a credit event.

Following the single-name credit derivatives come the multi-name credit derivatives, where there are more than one credit risky assets involved in the contract. One of the most important multi-name credit derivatives is the collateralized debt obligation (CDO). It is a security whose payments are linked to the default of credits in the underlying portfolio (O'Kane 2008, p. 230). Since the various forms of CDOs have the same structure and pricing formula, this document will not differentiate them by name.

The most remarkable achievement of multi-name credit derivatives is to introduce a liquid market of default correlation (O'Kane 2008, p. 219). The payment of a CDO is determined by the amount of default in the portfolio, which in turn is affected by the default correlation. On the one hand, when default correlation is positive, the risk of two or more credits defaulting together is higher than if they were independent. This is common with credits who share some factors such as being complements in the business or being in the same economy. On the other hand, when correlation is negative, joint default has a lower probability than the independent case. However, current pricing convention usually ignores the case of negative correlation because its influence is negligible (O'Kane 2008, p. 219).

Having mentioned the payment, it is time to introduce the CDO waterfall rule[3] for the scheduled coupon and principal payments. A CDO can have multiple tranches. Each tranche is assigned an attachment and a detachment point, which stand for the percentage portfolio loss. The attachment point indicates the amount of loss a portfolio can suffer before the tranche starts to lose its notional. Therefore, the lower the attachment point, the more vulnerable the tranche is in the occurrence of defaults. The tranche with attachment at zero is referred to as the equity tranche.

[2] Credit events include bankruptcy, failure to pay, obligation acceleration, obligation default, repudiation/moratorium, and restructuring (O'Kane 2001). In general terms, credit events are referred to as default.

[3] Although the waterfall rule was explicitly designed for traditional cash CDOs, the same mechanism applies to single-tranche CDOs as well. Hence the same name is used here.

1.2 The Pricing Logical Flow

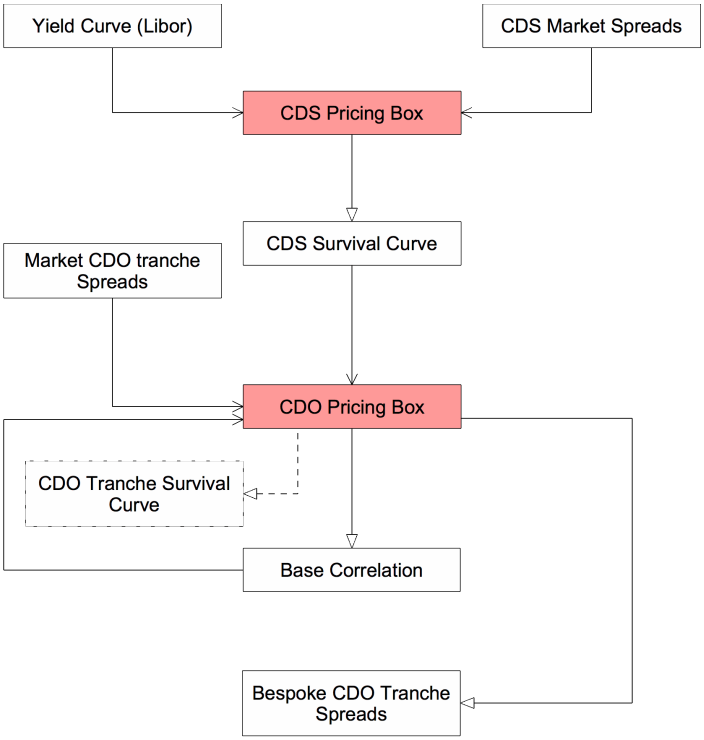


Figure 1.1 The Pricing Logical Flow

Figure 1.1 presents the logic behind this project. The coloured boxes are the pricing functions that are programmed with Matlab. The solid arrow indicates that the quantity is an input, and empty arrow indicates that the quantity is an output. Proceeding in such a way, the resulting bespoke CDO tranche spreads are guaranteed to be consistent with the CDS spreads and index CDO spreads. Note that the pricing boxes can work both ways – for example, given the CDS survival curve and yield curve, CDS spreads can be calculated.

The CDS survival curve is the fundamental element in the pricing of credit derivatives. With the yield curve and the CDS spreads, which are obtainable from the market, the CDS survival curve can be bootstrapped. It is then used as an input to the CDO pricing box.

With CDO pricing, it is essential to first obtain the tranche survival curve from the CDS survival curve. Because the leg mechanic is the same for both CDS and CDO, but CDO pricing uses the tranche survival curve instead of the CDS survival curve. Two models for calculating the tranche survival curve are implemented and tested. They are the finite homogeneous portfolio (FHP) model and the large homogeneous portfolio (LHP) model.

Both models are based on the one factor Gaussian copula model, which adopted some ideas from Merton's structural model[4]. Notice that the tranche survival curve is part of the CDO pricing box; the ultimate aim of building the CDO pricing box is to obtain the base correlation.

From 2003, the concept of implied correlation was brought to the market. It is the correlation computed by taking the market tranche spreads and inverting the CDO pricing box. The implied correlation for the generic tranches is called compound correlation. However, the resulting correlation curve was not flat as it should have been with the assumption made in the calculation of the tranche survival curve.

Then the concept of base correlation was first introduced by McGinty *et al.* in 2004 (O'Kane 2008, p. 375). The fundamental concept behind base correlation is that any generic tranche price is a combination of two equity tranche prices[5]. Base correlation is the correlation bootstrapped for each equity tranche. The price of a generic tranche can therefore be calculated as a combination of two base tranches prices using their base correlations, and this is how bespoke tranches can be priced in an arbitrage-free approach.

Before the 2007 crisis, the recovery rate[6] was assumed to be constant. During the crisis, the spreads of the super-senior tranches, which were considered to be immune to portfolio losses, increased to the point where base correlation values cannot be bootstrapped anymore (Amraoui, Cousot, Hitier, and Laurent 2009). Hence researchers proposed the stochastic recovery rate model for CDO pricing, among which the Amraoui-Hitier (AH) model is widely accepted in the market (Merrill Lynch 2009). The model specifies that recovery rate is a deterministic function of a stochastic variable, the market factor, and it is therefore indirectly stochastic. It has been proved that the stochastic recovery assumption drives the base correlation down as well as flattens it to some extent, which is a desirable feature for credit derivatives pricing (Amraoui, Cousot, Hitier, and Laurent 2009).

---

[4] This model will not be discussed in this report. Interested readers can refer to Merton, R. (1974) "On the Pricing of Corporate Debt: The Risk Structure of Interest Rates", *Journal of Finance*, 29, 449-470.
[5] Equity tranche is also referred to as base tranche.
[6] Recovery rate is the amount that can be recovered from par after a credit event. It is a parameter in both CDS and CDO pricing formula.

Chapter 2 The CDS Pricing Box

This chapter will first introduce the pricing of credit default swaps (CDSs), and then discuss the bootstrapping and interpolation of the CDS survival curve.

As formerly stated, the pricing box is a function programmed with Matlab. This function is invertible, meaning that it is possible to input the CDS survival curve and the yield curve to obtain the CDS spreads (the direct problem), or to input the CDS spreads and the yield curve to obtain the CDS survival curve (the inverse problem). Before solving the inverse problem, the functions to solve the direct problem are programmed first.

2.1 The Direct Problem

The premium leg of a CDS is the present value of all the future cash flows. Since the regular payments are quarterly distributed into the future, they must be weighed by the credit survival probability at the payment date and then discounted by the yield curve to obtain the present value. Therefore the regular coupon part of a CDS has the following form:

$$L^R = S\sum_{i=1}^{N}\Delta_i Q_i Z_i \tag{2.1}$$

where S refers to the contractual spread of a contract, Δ_i refers to the day count fraction between time i-1and time i, Q_i denotes the survival probability at time i, and Z_i refers to the Libor discount factor at time i.

Apart from the regular coupon part, the premium leg also needs to take into the account the accrued effect from the last coupon payment date to the time of default. This payment needs to be made after a credit event to the protection seller. Therefore it is determined by the default time. Given that time is continuous, the present value of the accrued coupon is calculated as:

$$L^A = S\sum_{i=1}^{N}\int_{t_{i-1}}^{t_i}\Delta(t_{n-1},s)Z(t,s)(-dQ(t,s)) \approx \frac{S}{2}\sum_{i=1}^{N}\Delta_i Z_i(Q_{i-1}-Q_i), \tag{2.2}$$

where s is the time of default.

This approximation is based on the observation that a default occurs, on average, halfway through a coupon period and so the accrued premium then will be

$$S\frac{\Delta(t_{i-1},t_i)}{2}.$$

Following the same assumption, the time interval and the discount factor is fixed, then the remaining integral in Equation 2.2 gives $Q_{i-1}-Q_i$. Finally using the discount factor to time $(t_{i-1}+t_i)/2$, the accrued coupon can be discounted to today's value. The result is the second expression in Equation 2.2.

Therefore, the present value of the premium leg is the sum of the regular coupon and the accrued coupon:

$$\text{Premium Leg PV} = S\cdot\left(\sum_{i=1}^{N}\Delta_i Q_i Z_i + \frac{1}{2}\sum_{i=1}^{N}\Delta_i Z_i(Q_{i-1}-Q_i)\right) \tag{2.3}$$

where the expression in the parenthesis is called the risky annuity.

The protection leg is only triggered by a credit event, and so it is uncertain. Such a quantity is modeled as the expectation of the recovered amount at the time of default:

$$\hat{D}(t,T) = E_t\left[\exp\left(-\int_t^{\tau} r(s)\,ds\right)\cdot\Phi(\tau)\cdot 1_{\tau\le T}\right]. \tag{2.4}$$

Here, $r(s)$ is the continuously compounded risk-free rate. Note that the risk of default is captured by the default indicator function $1_{\tau\le T}$ where τ is the time of default and T is the contract maturity date. The indicator function has the form:

$$1_{\tau\le T} = \begin{cases} 1, \text{ if } \tau \le \text{T and the credit defaults before T} \\ 0, \text{ if } \tau > \text{T and the credit survives} \end{cases}.$$

Another assumption made in Equation 2.4 is that the recovery rate, the risk-free rate and the default time are independent (O'Kane 2008, p. 105). Since the expectation of the recovered amount is equal to $E\left[\Phi(\tau)\right] = (1-R)$, Equation 2.4 can be written as

$$P = (1-R)\sum_{i=1}^{N}\int_{t_0}^{t_i} Z_i(-dQ_i). \tag{2.5}$$

The derivation of Equation 2.5 is the same as that for Equation 2.2. Therefore the present value of the protection leg is:

$$\text{Protection Leg PV} = (1-R)\sum_{i=1}^{N} Z_i(Q_{i-1} - Q_i). \tag{2.6}$$

The definition of CDS spread is that it is the value that makes the initial value of the CDS contract equal to zero (O'Kane 2008, p. 98). It is calculated as the protection leg subtracted by the premium leg:

$$\text{CDS Present Value} = P - (L^R + L^A). \tag{2.7}$$

By equating the present value to zero, the CDS spread is calculated as the protection leg divided by the risky annuity:

$$S = \frac{(1-R)\sum_{i=1}^{N} Z_i(Q_{i-1} - Q_i)}{\sum_{i=1}^{N} \Delta_i Q_i Z_i + \frac{1}{2}\sum_{i=1}^{N} \Delta_i Z_i (Q_{i-1} - Q_i)}. \tag{2.8}$$

The direct problem is solved in Excel before programmed with Matlab, and the results are compared with the JP Morgan (2001) paper to ensure that the formulas are correct, as showed in Figure 2.1.

Sheets Charts

	A	B	C	D	E	F	G	H
1	Recovery	30%						
2								
3	Period(i)	Yld	DF	Survival Prob	Regular Coupons	Accrued Coupons	Protection	Spread
4	0.25	2.35%	99.41%	96.43%	0.240	0.004	0.025	1,018
5	0.5	2.33%	98.84%	93.05%	0.470	0.009	0.048	1,009
6	0.75	2.39%	98.25%	89.76%	0.690	0.013	0.071	1,008
7	1	2.52%	97.63%	86.56%	0.901	0.017	0.093	1,010
8	1.25	2.70%	96.98%	83.91%	1.105	0.020	0.111	985
9	1.5	2.87%	96.28%	81.51%	1.301	0.023	0.127	959
10	1.75	3.05%	95.55%	79.41%	1.491	0.025	0.141	930
11	2	3.22%	94.78%	77.30%	1.674	0.028	0.155	911
12								

Figure 2.1 CDS Spread Calculations in Excel

Note that in the calculation, the quarters are expressed as 0.25. This is only a rough approximation to the true period. The market convention is to follow the Actual 360 rule[7]. This issue will be further discussed in the CDS survival curve bootstrapping section.

[7] Actual 360 is used with US dollars, Japanese yen and euros. For UK Sterling, Actual 365 is used (O'Kane 2008, p. 11).

2.2 The Inverse Problem and the Bootstrap

The inverse problem involves calculating the CDS survival curve given the market CDS spreads. From equation 2.8, it can be seen that the only unknown variable is the survival curve, which makes inverting the formula possible. There are three quantities that are closely related. They are the zero default rate, the survival probability, and the forward default rate. The relationship can be expressed as:

$$Q(T) = \exp\left(-\int_0^T h(t)\,dt\right) = \exp\left(-y(T)\cdot T\right) \tag{2.9}$$

where $h(t)$ is the forward default rate and $y(T)$ is the zero default rate. In this project, it is the zero default curve that is bootstrapped. Because of their one-to-one correspondence, for simplicity, this document will also refer to the zero default curve as the survival curve.

As was mentioned in the beginning, the survival curve is built using the bootstrapping method. It is one of the curve building methods that works from the shortest maturity towards the longest maturity. At each step, there is a one-dimensional equation to solve. The result of this equation is combined with all the previous results, and they are used to solve for the next point. Proceeding in such a way, the bootstrap method is highly efficient and produces very stable results (O'Kane 2008, p. 114).

Given the full set of CDS spreads and the yield curve, the bootstrapping of the survival curve starts with the value at the first coupon payment date. In Equation 2.8, if $i = 1$, then the spread will be a function of Q_1 because the survival probability at time zero must be 1, i.e. $Q_0 = 1$. Then the value of Q_1 can be found. The next step is to let $i = 2$. Since Q_1 is already known, Q_2 can be found with a one-dimensional root search algorithm. Notice that here the survival probabilities are expressed in terms of the zero default rate, and so it is the zero default rate that is actually being solved. The reason for solving for the zero default curve instead of solving for the survival probability directly is that the former quantity is more suitable for interpolation. This issue will be discussed in the next section. Figure 2.2 presents the whole bootstrapped curve together with the original survival curve from the JP Morgan (2001) paper.

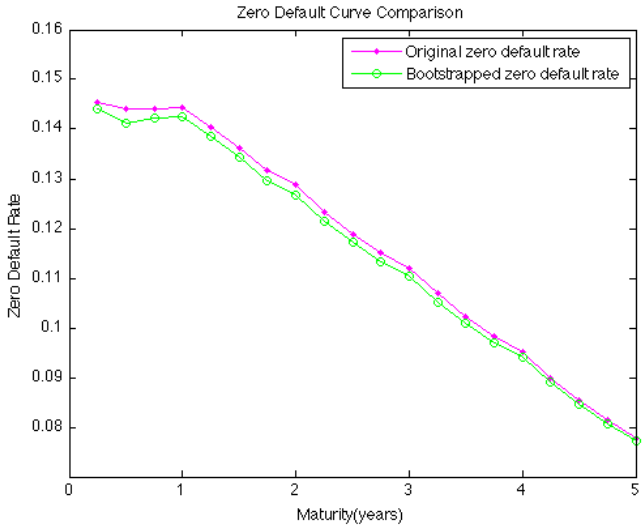


Figure 2.2 Bootstrapped Survival Curve

It is obvious that there is a small discrepancy between the two curves. This is the result of using hard-coded time period length of 0.25 instead of following the Actual 360 convention. Under the Actual 360 convention, the day count fraction is calculated as:

$$\Delta = \frac{ActualNoDays}{360}.$$

Therefore a better approximation for delta is:

$$\Delta = \frac{365.25}{360} \times \frac{1}{4}.$$

Now delta is approximately 0.254. Now, Figure 2.3 presents the result from bootstrapping with the new set of maturities:

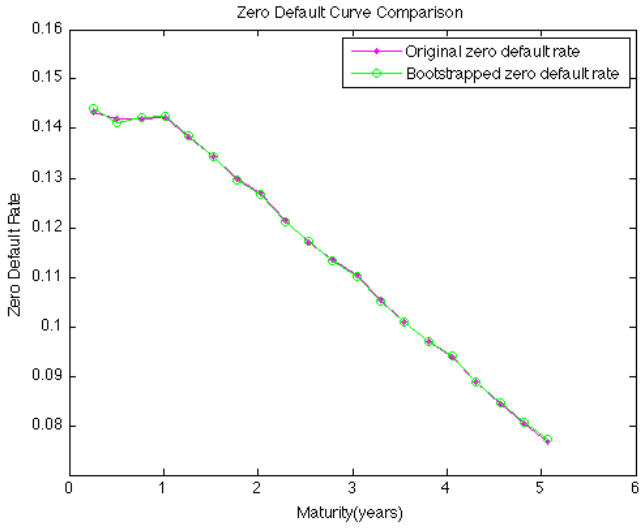


Figure 2.3 Bootstrapped Survival Curve under the Actual 360 Convention

The new curve is almost identical to the original curve. The small differences may be due to weekends and holidays[8], which means that the real maturities used to generate the original spreads may not strictly follow the previous equation. However this issue is not further investigated. It can be seen from the calculation that the Actual 360 convention makes each quarter longer than 0.25.

In order to check the accuracy of the bootstrapping function, it is advisable to recalculate the spreads with the bootstrapped survival curve and the yield curve. Figure 2.4 validates the results.

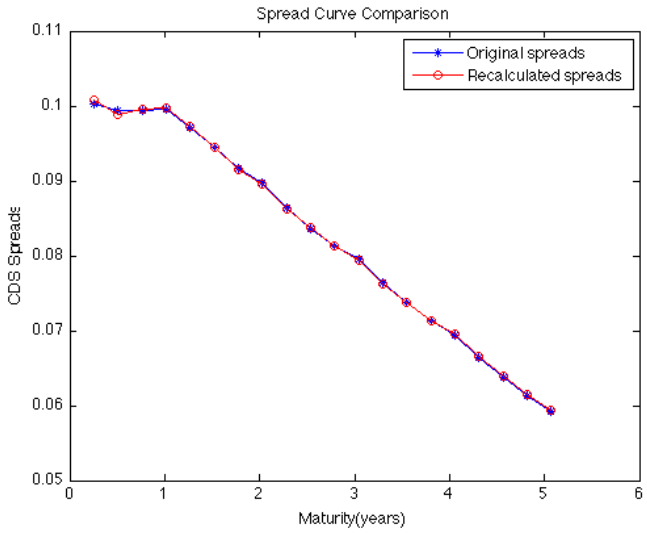


Figure 2.4 CDS Spreads Recalculated

However, the above bootstrap is based on the full set of maturity-spread pairs. This is an artificial scenario and is only implemented to validate the calculation. In reality, market spreads are not available for each maturity. For example, we may only have spreads for contract that ends in 6M, 1Y, 2Y, 3Y, 5Y, 7Y and 10Y, where M and Y stands for months and years respectively. Therefore, an interpolation scheme is required to fill in the gaps between the available maturities.

[8] Weekends and holidays cannot be set as a coupon payment day. Usually the next day will be the coupon payment day if such a day is encountered.

2.3 Interpolation

Interpolation is a method for constructing new data points within the range of a discrete set of given data points. When producing the zero default curve, bootstrap should be performed with interpolation at each step. The reason is that bootstrap produces a set of discrete points, and interpolation is used to produce a smooth and continuous curve.

There are certain criteria regarding how to select the interpolation scheme and quantity so that results are acceptable. For example, it is desirable that the interpolated curve is smooth and local, the latter meaning that a change in one part of the curve will not bring big changes in other part of the curve (Hagan & West 2006). In the case of the survival curve, the most important criterion is that precise results should be obtainable given a set of reference maturity-rate pairs (O'Kane 2008, p. 113).

In order to achieve such accuracy, it is necessary to compare the various interpolation schemes applied to different quantities. O'Kane (2008, p. 115-118) discussed in his book the results from linear interpolation of three different quantities, namely the log of survival probability, the zero default rate, and the forward default rate. In the end, only the result from the first method was acceptable because the other two produce saw-like curves that are not true representations of the market environment.

Sometimes it is necessary to perform extrapolation as well. It is the act of estimating values outside the range of given maturity. In the case of the zero default curve, the values before the shortest maturity point available should be horizontal to that point, and the values exceeding the last interpolation point should exhibit the same slope as that point.

Now it is time to move onto the mathematical construct of log-linear interpolation of survival probability. Having mentioned above that zero default curve cannot be linearly interpolated, it is sensible to interpolate the log of survival probability (which is equivalent to $-y(T)\cdot T$ in Equation 2.9) and then divide each result by the corresponding maturity to obtain the set of zero default rate (i.e. $y(T)$).

For any linear function $f(t)$ as presented in Figure 2.5 below, the standard interpolation formula to some time $t^* \in [t_{n-1}, t_n]$ is given by Equation 2.10:

$$f(t^*) = \frac{(t_n - t^*) f(t_{n-1}) + (t^* - t_{n-1}) f(t_n)}{t_n - t_{n-1}} \tag{2.10}$$

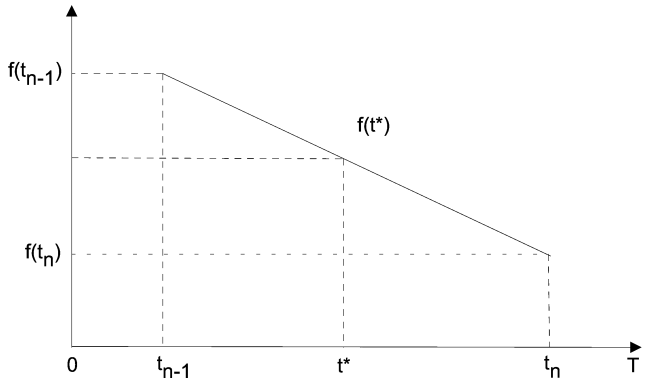


Figure 2.5 Linear Interpolation Scheme for f(t)

The log-linear interpolation of the survival probability starts with letting

$$f(t) = -\ln(Q(t)). \tag{2.11}$$

From Equation 2.9, it can be derived that

$$f(t) = \int_0^t h(s)\, ds \quad \Rightarrow \quad h(t) = \frac{\partial f(t)}{\partial t}.$$

Hence, by differentiating Equation 2.10

$$h(t^*) = \frac{\partial f(t^*)}{\partial t^*} = \frac{f(t_n) - f(t_{n-1})}{t_n - t_{n-1}}. \tag{2.12}$$

From Equation 2.12, it can be seen that $h(t^*)$ does not depend on t^*. So the forward default curve is piecewise constant between the given maturity limits. Therefore by substituting Equation 2.11 to Equation 2.12, the constant forward default probability for t^* where $t_{n-1} < t^* < t_n$ can be defined as:

$$h(t^*) = h(t_{n-1}) = \frac{1}{t_n - t_{n-1}} \ln\left(\frac{Q(t_{n-1})}{Q(t_n)}\right),$$

and hence obtaining the formula for the survival probability to some time t^* as:

$$Q(t^*) = Q(t_{n-1}) \exp\left(-(t^* - t_{n-1}) h(t_{n-1})\right). \tag{2.13}$$

Equipped with the formulas, it is time to start the implementation in Matlab.

2.4 Implement Interpolation and Bootstrap with Matlab

In Matlab, there is a built-in function for linear interpolation (and extrapolation), the `interp1` function. This is the method to obtain the result with the least effort. In order to interpolate zero default rates, the parameters that need to be provided are: the reference zero default rates, their corresponding maturities, and a set of maturities that needs to be interpolated.

After creating the interpolation function, a test file and some data is required to check the function's correctness. Given a set of maturity-rate pairs, the interpolated zero default curve with markers at the reference points are presented in Figure 2.6.

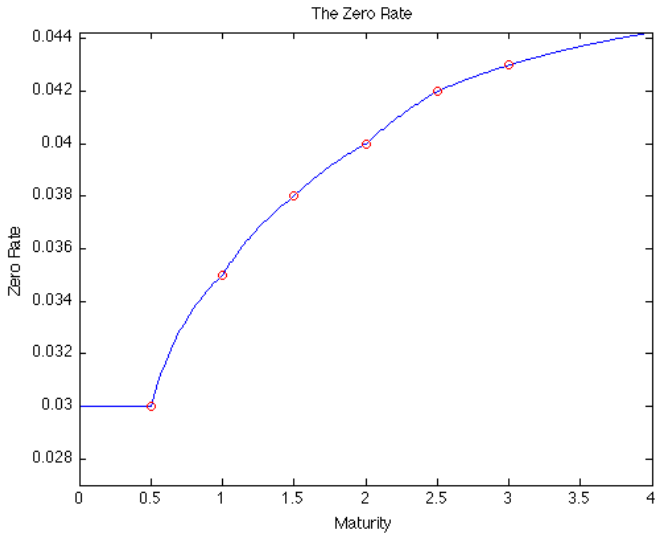


Figure 2.6 Interpolation of the Zero Default Curve

Once the interpolation function is tested and validated, it is used in bootstrapping the zero default curve. The implementation of the bootstrapping method is in the form of a Matlab function. To save steps further, it is useful to create separate functions to calculate the risky annuity and the protection leg as in Equation 2.8. These two functions should work under the fully determined conditions so that they can be called in the bootstrap function.

Given an incomplete set of maturity-spread pairs, the idea of this bootstrap is to create a function handle of the CDS present value (`presentValueHandle`) as defined by Equation 2.7; express it in terms of the zero default rate and then use the Matlab `fzero` function to solve for the one-dimensional root.

As with the bootstrap with the full set of data, each step in this bootstrap should also be one-dimensional. Vectorization of the unknown parameter (i.e. express the whole set of zero default rates as an array) will violate the concept of bootstrapping because it enables a function to take account of all the values at once. Furthermore, the `fzero` function does not work with vectors.

In order to avoid such vectorization, it is convenient to introduce another parameter, the last zero default rate (`lastZeroDefaultRate`), letting it be the unknown variable of the present value function, and make the already obtained zero default rate array a parameter. After solving for the `lastZeroDefaultRate`, equate the end of the zero default rate array to it. The solved rates, together with the interpolated ones, are then used for the construction of the next `presentValueHandle`.

The `presentValueHandle` is not the only sub-function in this bootstrap function. Interpolation should be performed after obtaining a bootstrapped value so that the next `presentValueHandle` can be calculated with all the previous zero default rates. Hence another trick is to let the bootstrap function to output a handle of the zero rate interpolation function (`zeroRateLogLinear`) instead of an array of interpolated values. This function handle enables the function to accept any maturity periods later in a test file. Therefore the bootstrap proceeds along the maturity timeline, with each step being one-dimensional.

Again, in order to test the functions built above, a test file needs to be created and data need to be provided for the function. The data used in this test file is from the same JP Morgan (2001) paper. The resulted zero default curve is illustrated with Figure 2.7. The corresponding forward default curve and survival probability curve is presented in Figure 2.8 and 2.9 respectively.

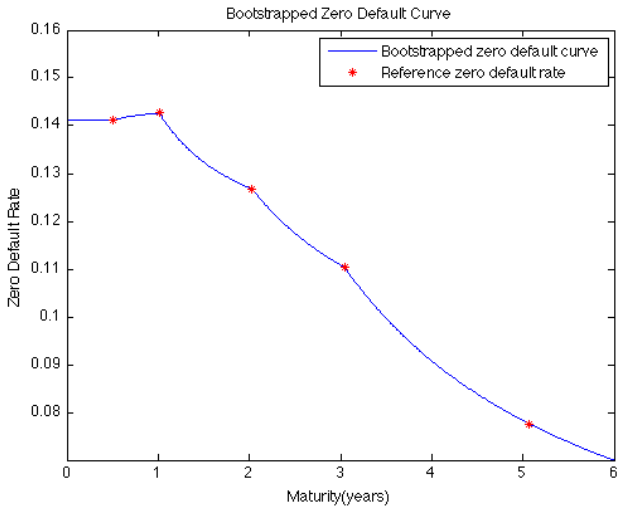


Figure 2.7 Bootstrapped Survival Curve

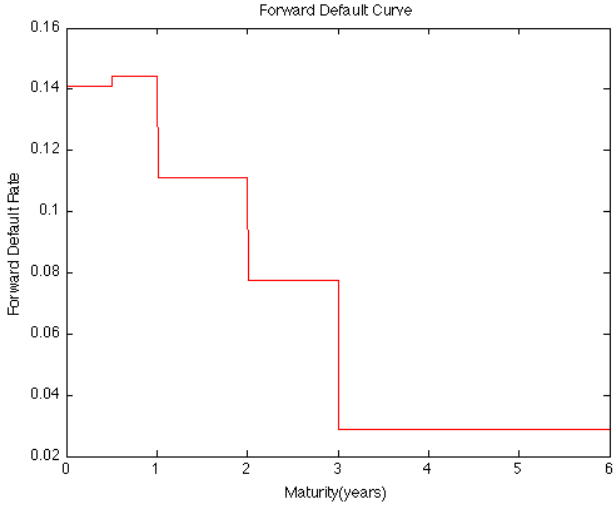


Figure 2.8 The Forward Default Curve

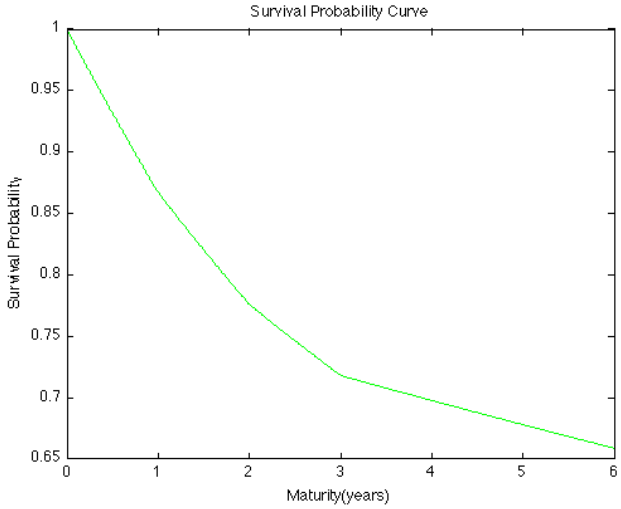


Figure 2.9 The Survival Probability Curve

Chapter 3 The CDO Pricing Box

This chapter will present the pricing formulas for CDOs and discuss two models of building the tranche survival curve.

One of the ways that standard index CDO tranches are quoted is by setting the attachment-detachment points at 0-3%, 3-6%, 6-9%, 9-12%, and 12-22%. The premium payment for each tranche is based on its notional, the loss on which is proportional to the loss on the underlying portfolio, and the proportion is pre-determined by the attachment and detachment point. So each tranche has its own survival curve that is a function of tranche loss and is dependent on the portfolio loss. Similar to CDS, each tranche has a premium leg and a protection leg, and they are used to calculate the tranche present value.

Regarding the survival curve, all the models studied in this project focus on the equity tranche survival curve. Indeed, any generic tranche can be expressed as a combination of two equity tranches. Moreover, using the equity tranche enables the pricing of bespoke tranches and the bootstrap of base correlation.

3.1 The Tranche Loss and Tranche Survival Curve

The CDO tranche loss has a simple mechanism, as illustrated in Figure 3.1, where $L(T,K_1,K_2)$ is the tranche loss, and $L(T)$ is the portfolio loss.

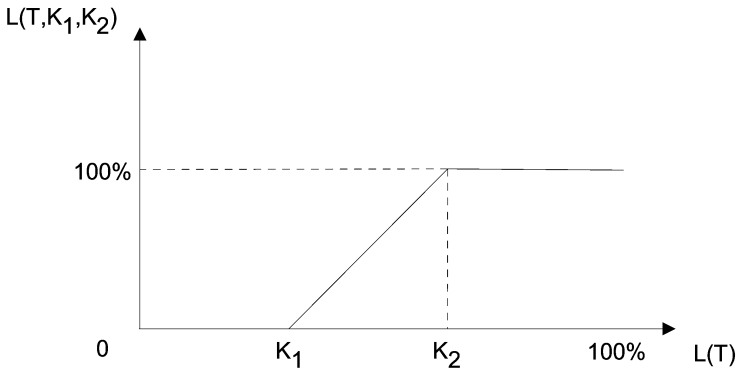


Figure 3.1 The Tranche Loss Mechanism

There are three scenarios:

1. When the percentage loss of the portfolio is below the attachment point K_1, the tranche incurs zero loss. The probability of this situation is $\Pr(L(T) \le K_1)$ and this increases as K_1 increases.
2. The probability of tranche loss between 0% and 100% is linearly correspondent to the portfolio loss $L(T)$.
3. The probability of 100% loss is equivalent to the situation where the percentage portfolio loss exceeds K_2, i.e. $\Pr(L(T) \le K_2)$. This probability decreases as K_2 increases.

Therefore the mathematical expression for the tranche loss is:

$$L(T,K_1,K_2) = \frac{\max(L(T) - K_1,0) - \max(L(T) - K_2,0)}{K_2 - K_1}. \tag{3.1}$$

The tranche survival probability at time t is defined to be the expectation of the tranche outstanding notion at that time. So it can be written as:

$$Q(T,K_1,K_2) = 1 - E[L(T,K_1,K_2)] = 1 - \frac{E[\min(L(T),K_2) - \min(L(T),K_1)]}{K_2 - K_1} \tag{3.2}$$

As discussed at the beginning of this chapter, any generic tranche can be calculated with two base tranches. To see this, first of all two concepts needs to be clarified, the absolute value and the percentage value. Intuitively, an absolute value refers to the value in terms of the currency, and a percentage value is expressed in percentage format.

Now let A and B be the absolute attachment and detachment point respectively. Let L_{AB} denote the absolute tranche loss; L_P denote the absolute portfolio loss; and L_{OA}, L_{OB} denote the absolute equity tranche loss. Then according to Figure 3.1, the following relations hold:

$$L_{AB} = \begin{cases} 0, & L_P < A \\ L_P, & A < L_P < B \\ B - A, & L_P > B \end{cases} \qquad L_{OA} = \begin{cases} L_P, & L_P < A \\ A, & L_P > A \end{cases} \qquad L_{OB} = \begin{cases} L_P, & L_P < B \\ B, & L_P > B \end{cases}$$

Therefore, there is a relationship between the losses, namely

$L_{AB} = L_{OB} - L_{OA}$.

Now introduce another two variables, N_{AB} and N_{TR} which stand to the absolute outstanding tranche notional and the absolute total tranche notional. Then we can write:

$L_{AB} + N_{AB} = N_{TR} = B - A$.

Consider the percentage tranche loss l_{ab} and percentage equity tranche loss l_{oa} and l_{ob} who have the form:

$$l_{ab} = \frac{L_{AB}}{N_{TR}} = \frac{L_{AB}}{B - A} \qquad l_{oa} = \frac{L_{OA}}{A} \qquad l_{ob} = \frac{L_{OB}}{B}$$

Denote the percentage attachment and detachment point by a and b, and denote the absolute portfolio notional by N_P, then we can write:

$$a = \frac{A}{N_P} \qquad b = \frac{B}{N_P}$$

Therefore according to the above, the following derivation can be justified:

$$(B - A)\cdot l_{ab} = L_{AB} = L_{OB} - L_{OA} = B\cdot l_{ob} - A\cdot l_{oa}$$

$$\Rightarrow \quad l_{ab} = \frac{B}{B - A} l_{ob} - \frac{A}{B - A} l_{oa} = \frac{b}{b - a} l_{ob} - \frac{a}{b - a} l_{oa}$$

$$\Rightarrow \quad l_{ab} = x\cdot l_{ob} + (1 - x)\cdot l_{oa}$$

where $x = \frac{b}{b - a}$.

In terms of percentage outstanding tranche notional n_{ab} and percentage outstanding equity tranche notional n_{oa} and n_{ob} who are calculated as:

$$n_{ab} = 1 - l_{ab} \qquad n_{oa} = 1 - l_{oa} \qquad n_{ob} = 1 - l_{ob}$$

We can derive:

$$1 - n_{ab} = x(1 - n_{ob}) + (1 - x)(1 - n_{oa}) \quad \Rightarrow \quad n_{ab} = xn_{ob} + (1 - x)n_{oa}$$

Finally, the survival probability is the expectation of the outstanding notional:

$$Q_{ab} = E[n_{ab}] = xE[n_{ob}] + (1 - x)E[n_{oa}] = xQ_{ob} + (1 - x)Q_{oa}. \qquad (3.3)$$

3.2 The Legs

Same as the CDS premium payments, the CDO tranche premium leg contains a series of cash flows made by the tranche protection buyer to the tranche protection seller. The size of the payments is determined by the contractual tranche spread. Here we denote this spread as $S(K_1,K_2)$ where K_1 and K_2 are the attachment and detachment points.

Following the Actual 360 convention, the coupons on the premium leg are usually paid quarterly on the survived notional of the tranche. So at time t_i, the size of a single premium payment for each \$1 face value is:
$\Delta(t_{i-1},t_i)S(K_1,K_2)(1-L(t_i,K_1,K_2))$.
Therefore the present value of the premium leg at time zero is:

$$\text{Premium Leg PV} = S(K_1,K_2)\sum_{i=1}^{N_T}\Delta(t_{i-1},t_i)Z(t_i)E[1-L(t_i,K_1,K_2)] \tag{3.4}$$

As with CDS pricing, the premium leg should take into account the accrued payment from the last coupon payment date to the default date. Due to the fact that STCDO premium payment is only a function of the tranche loss, there are three scenarios to be considered:

1. If the default occurs immediately before a coupon payment date, it will have no effect on the premium paid.
2. If the default occurs at the start of the accrual period, then the next coupon is calculated on the prorated notional over the coupon period.
3. For defaults that occur midway through the period, we can approximate the tranche notional as the average of the start and the end notional.

(O'Kane 2008, p. 234)

Therefore, a useful and more accurate approximation is to assume that the spread is paid on the average tranche notional since the previous coupon payment date (O'Kane 2008, p. 236):

$$\text{Premium Leg PV} = S(K_1,K_2)\sum_{i=1}^{N_T}\Delta(t_{i-1},t_i)Z(t_i)E[1-\frac{L(t_{i-1},K_1,K_2)+L(t_i,K_1,K_2)}{2}] \tag{3.5}$$

According to Equation 3.2, the premium leg present value can be written as:

$$\text{Premium Leg PV} = \frac{S(K_1,K_2)}{2}\sum_{i=1}^{N_T}\Delta(t_{i-1},t_i)Z(t_i)[Q(t_{i-1},K_1,K_2)+Q(t_i,K_1,K_2)] \tag{3.6}$$

Slightly different in concept than the CDS protection payment, the CDO tranche protection leg consists of a series of loss payments made by the investor to the dealer to cover default loss on the tranche. The size of the payment is simply the change in the value of the tranche loss.

Suppose that the reference portfolio has N_c credits all with the same exposure and recovery rate R, then the percentage portfolio loss for each default is:

$$l = \frac{(1-R)}{N_c}. \tag{3.7}$$

Therefore, the maximum number of defaults that a portfolio can suffer before incurring a loss on the tranche is:

$$n_1 = ceil\left(\frac{K_1}{l}\right). \tag{3.8}$$

Similarly, the number of defaults required to reduce the tranche notional to zero is given by:

$$n_2 = ceil\left(\frac{K_2}{l}\right). \tag{3.9}$$

To calculate the present value of the protection leg, consider tranche loss over a small period of time dt : $dt = dL(t,K_1,K_2)$. The present value of the protection leg is therefore:

$$\text{Protection Leg PV} = \int_0^T Z(s)E[dL(s,K_1,K_2)]. \tag{3.10}$$

According to Equation 3.2, we can write:

$$E[dL(s,K_1,K_2)] = dE[L(s,K_1,K_2)] = -dQ(T,K_1,K_2). \tag{3.11}$$

Substituting Equation 3.11 to Equation 3.10, the protection leg has the form:

$$\text{Protection Leg PV} = -\int_0^T Z(s)\, dQ(s,K_1,K_2). \tag{3.12}$$

Comparing Equation 3.12 to Equation 2.5, it is clear that calculation of the protection leg of a CDO tranche is analogical to the protection leg of a CDS, only without the consideration of the recovery rate. Therefore the protection leg of a tranche is equal to:

$$\text{Protection Leg PV} = \sum_{i=1}^{N} Z_i\left(Q(t_{i-1},K_1,K_2) - Q(t_i,K_1,K_2)\right) \tag{3.13}$$

Comparing Equation 3.6 to Equation 2.3, and Equation 3.13 to Equation 2.6, it is clear that the CDO pricing function is almost identical to the CDS pricing function. The essential difference is that CDO pricing box accept tranche survival curve instead of CDS survival probability. Therefore a study on the tranche survival is necessary in order to build the CDO pricing box.

3.3 The Portfolio Loss Distribution and the Gaussian Copula Model

From Equation 3.2, it is obvious that the tranche survival curve is a function of the portfolio loss $L(T)$. Therefore in order to build the tranche survival curve, it is necessary to obtain the portfolio loss distribution first. The fundamental model on which this distribution is build is the Gaussian Copula Model.

The Gaussian copular model, or the Gaussian latent variable model, was introduced in the late 1990s, and has since become the standard pricing model for most correlation products (Merrill Lynch 2009). It assumes a random Gaussian variable A_i for each credit i in the portfolio with mean zero and unit standard deviation, and specifies that default will occur if the value of A_i is less than a time dependent threshold $C_i(T)$. More specifically,

$$\Pr(\tau_i \le T) = \Pr\left(A_i \le C_i(T)\right) = 1 - Q_i(T), \tag{3.14}$$

where $Q_i(T)$ is the survival probability for credit i at time T.

Since $A_i \sim N(0,1)$, $1 - Q_i(T) = \Phi\left(C_i(T)\right)$ and therefore the threshold is expressed as:

$$C_i(T) = \Phi^{-1}\left(1 - Q_i(T)\right). \tag{3.15}$$

For correlation products, introduce an A_i for each of the $i = 1,\ldots,N_c$ credits in the portfolio. In order to correlate default, we just need to correlate A_i and A_j. Hence specify a one-factor correlation structure as:

$$A_i = \beta_i Z + \sqrt{1-\beta_i^2}\,\varepsilon_i, \tag{3.16}$$

where Z is called the market factor and is common to all the credits in the portfolio, and ε_i is the idiosyncratic factor that is specific to each credit i. Both are Gaussian distributed with mean zero and unit standard deviation. They are also independent.

So what is β_i? In this project, all the models used to build the tranche survival curve assume homogeneity for the underlying credits, meaning that each credit has the same survival probability, recovery rate, and pair-wise correlation. So β_i here is the square root of the pair-wise correlation between the underlying credits in the portfolio.

Now according to Equation 3.13:

$$A_i < C_i(T) \quad \Rightarrow \quad \varepsilon_i < \frac{C_i(T) - \beta_i Z}{\sqrt{1-\beta_i^2}}. \tag{3.17}$$

Therefore conditioning on the market factor Z and substituting Equation 3.17 back into Equation 3.14 will give the conditional default probability for credit i before time T:

$$p_i(T|Z) = 1 - Q_i(T|Z) = \Phi\left(\frac{C_i(T) - \beta_i Z}{\sqrt{1-\beta_i^2}}\right). \tag{3.18}$$

Since the portfolio is assumed to be homogeneous, $p_i(T|Z) = p(T|Z)$ for each i. Therefore the conditional loss distribution of the portfolio is a binomial distribution:

$$f(L(T)|Z) = \Pr\left(L(T) = \frac{n(1-R)}{N}\middle| Z\right) = \binom{N}{n} p(T|Z)^n \left(1 - p(T|Z)\right)^{N-n}, \tag{3.19}$$

where N is the total number of credits and n is the number of defaulted credits. Note that the expression for $L(T)$ stems from Equation 3.7 for the percentage portfolio loss.

Once the conditional distribution is obtained, unconditional distribution can be computed by integration over the density of the market factor:

$$f(L(T)) = \int_{-\infty}^{+\infty} f\left(L(T)|Z\right) \cdot \phi(Z)\, dZ. \tag{3.20}$$

Assuming that each credit in the portfolio has the same exposure, then the percentage portfolio loss with k defaults is

$$l = \frac{k(1-R)}{N_c}.$$

The unconditional loss distribution against the percentage portfolio loss is plotted with Matlab and is illustrated with Figure 3.2 (here the number of credits is 125, correlation is 20% and the recovery rate is 40%). The result is validated with the Dresdner Kleinwort (2008) spreadsheet model "*Opening the Black Box Part 2 – The Finite Homogeneous Pool Model*".

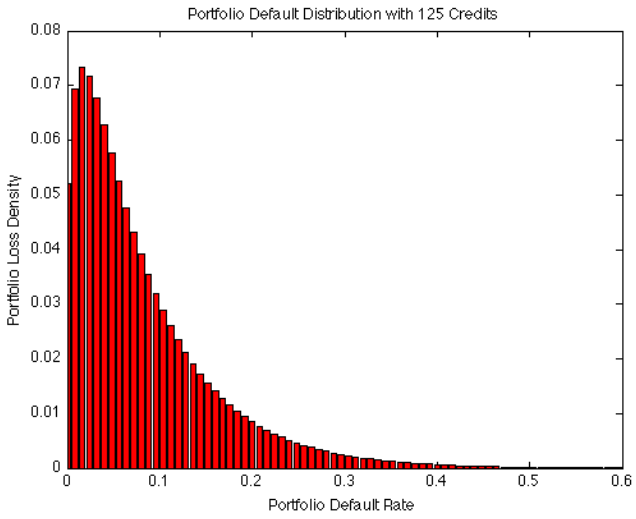


Figure 3.2 Unconditional Portfolio Loss Distribution

Once the portfolio distribution function is tested and validated, the next step is to build the tranche survival curve. The curve has different values under different models because of their different assumptions. The most important two are the finite homogeneous portfolio (FHP) model and the large homogeneous portfolio (LHP) model. Although the latter has been the standard market model for building tranche survival curve for a long time, it is logically more appropriate to start with the FHP model.

3.4 The Tranche Survival Curve under the FHP Model

The FHP model assumes that the number of credits in the portfolio is homogenous and finite[9]. Such assumptions allow its survival curve to be modeled with the binomial distribution expressed in Equation 3.18.

If we denote the percentage portfolio loss at time t by $l_p(t)$, then

$$l_p(t) = \frac{(1-R)k}{N_c}$$

where k is the number of defaulted credits. So for an equity tranche, the outstanding notional at time t is expressed as:

[9] In the iTraxx index there are 125 credits.

$$n_{oa}(t) = \begin{cases} 1 - \dfrac{l_p(t)}{a}, & l_p(t) < a \\ 0, & l_p(t) > a \end{cases} = \left[1 - \frac{l_p(t)}{a}\right] \cdot 1_{l_p(t) < a}$$

where a is the tranche detachment point.

Now, the tranche survives as long as the portfolio loss does not exceed the detachment point, i.e. $l_p(t) < a$. So we have the following derivation:

$$\frac{(1-R)k}{N_c} < a \quad \Rightarrow \quad k < \frac{N_c a}{1-R} \quad \Rightarrow \quad n_{oa}(t) = \begin{cases} 1 - \dfrac{k(1-R)}{N_c a}, & if\ k < \dfrac{N_c a}{1-R} \\ 0, & if\ k > \dfrac{N_c a}{1-R} \end{cases}$$

Letting $g = \dfrac{k(1-R)}{N_c a}$, the expectation of conditional outstanding notional is calculated as:

$$\begin{aligned} E[n_{oa}(t) \mid Z] &= E\left[\left(1 - \frac{k}{g}\right) \cdot 1_{k<g} \middle| Z\right] \\ &= \sum_{k=0}^{\lfloor g \rfloor} \left(1 - \frac{k}{g}\right) \cdot E[1_{k<g} \mid Z] \\ &= \sum_{k=0}^{\lfloor g \rfloor} \left(1 - \frac{k}{g}\right) \cdot \binom{N_c}{n} p(T \mid Z)^k (1 - p(T \mid Z))^{N_c - k} \end{aligned}$$

and hence the equity trance survival curve, i.e. the unconditional tranche outstanding notional, is calculated as:

$$Q_{oa} = E[n_{oa}(t)] = \int_{-\infty}^{+\infty} E[n_{oa}(t) \mid Z] \phi(Z)\, dZ.$$

Following Equation 3.3, the tranche survival curve for a generic tranche is:

$$Q_{ab} = E[n_{ab}(t)] = xE[n_{ob}(t)] + (1-x)E[n_{oa}(t)] = xQ_{ob} + (1-x)Q_{oa} \tag{3.21}$$

where $x = \dfrac{b}{b-a}$.

Figure 3.3 displays the equity tranche survival curve with different detachment points, and Figure 3.4 displays the tranche survival curve for standard index tranches, both under the assumption of 125 credits, 40% recovery rate, and 20% correlation. The tranche survival curves are then used to calculate the tranche spreads, the result of which is validated with the same Dresdner Kleinwort (2008) model.

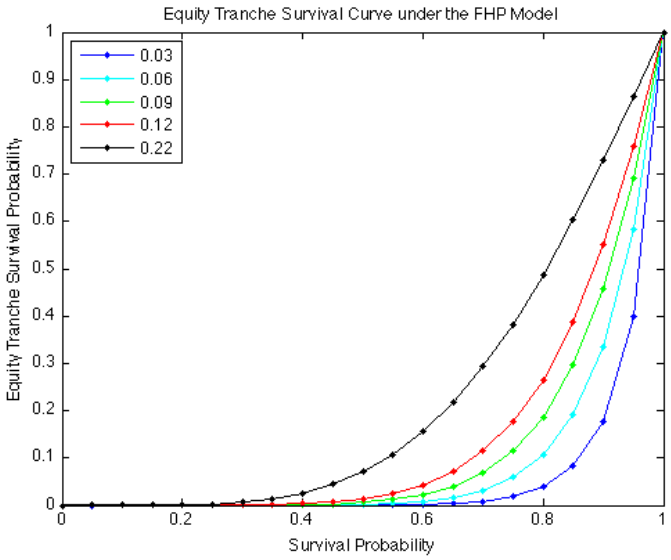


Figure 3.3 Equity Tranche Survival Curve under the FHP Model

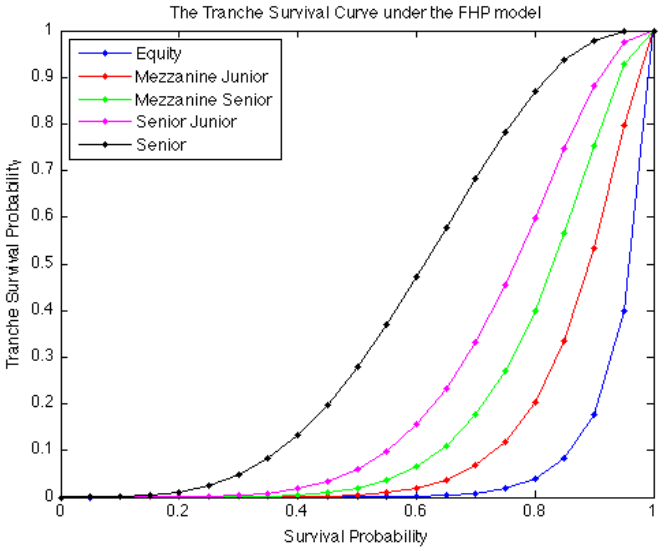


Figure 3.4 Tranche Survival Curve under the FHP Model

3.5 The Tranche Survival Curve under the LHP Model

The LHP model is a limiting case of the Gaussian copula model because it assumes that there are infinitely many credits in a homogeneous portfolio. Based on the assumption, the model does not require binomial distribution so the calculation is greatly simplified. O'Kane (2008, p. 303) pointed that the implementation of LHP model is typically 100 times faster than that of a model who calculates the exact portfolio loss distribution, and the error is within 5%. Therefore it has become the standard market model for credit derivative pricing.

Recall that in Section 3.1 we have specified that the tranche survival curve is a function of the portfolio loss. In Equation 3.2, the only variable is the expectation of the minimal of the portfolio loss and the tranche attachment/detachment point. Deriving this expectation is the primary objective of this section.

We start with the conditional loss distribution that has been specified with Equation 3.19. It can be shown that when the number of credits tends to infinity, this density function tends to a unit point mass of probability located at the conditional expectation of the portfolio loss (O'Kane 2008, p. 304). That is to say, with infinite number of credits, the conditional loss distribution function becomes a Dirac delta function that has the property:

$$\delta(x) = 0 \quad \forall\, x \neq 0 \qquad \int_{-\infty}^{+\infty} a \cdot \delta(x)\, dx = a \qquad \text{and} \qquad \int_{-\infty}^{+\infty} \delta(x-a) f(x)\, dx = f(a)$$

where a is a constant. Following this property, the conditional expectation of portfolio loss is:

$$E[L(T \mid Z)] = (1-R)\, p(T \mid Z) = (1-R)\Phi\left(\frac{C_i(T) - \beta_i Z}{\sqrt{1-\beta_i^2}}\right). \tag{3.22}$$

Now if we denote the cumulative distribution function of the conditional portfolio loss distribution with $F(K)$ and where K is the equity detachment point, and define it to be: $F(K) = \Pr\left(L(T|Z) \le K\right)$. Then given that the portfolio is homogeneous, $\beta_i = \beta$, so

$$F(K) = \Pr\left((1-R)\Phi\left(\frac{C(T) - \beta Z}{\sqrt{1-\beta^2}}\right) \le K\right).$$

Now since

$$(1-R)\Phi\left(\frac{C_i(T) - \beta_i Z}{\sqrt{1-\beta_i^2}}\right) \le K \quad \Rightarrow \quad \frac{C_i(T) - \beta_i Z}{\sqrt{1-\beta_i^2}} \le \Phi^{-1}\left(\frac{K}{1-R}\right)$$

$$\Rightarrow Z \ge \frac{1}{\beta_i}\left(C(T) - \sqrt{1-\beta_i^2}\,\Phi^{-1}\left(\frac{K}{1-R}\right)\right)$$

Letting $A(K) = \frac{1}{\beta}\left(C(T) - \sqrt{1-\beta^2}\,\Phi^{-1}\left(\frac{K}{1-R}\right)\right)$, the cumulative distribution function $F(K)$ can be rewritten with a condition on the market factor Z:

$$F(K) = \Pr(Z \ge A(K)) = 1 - \Pr(Z \le A(K)) = 1 - \Phi(A(K)).$$

Now we can write the expectation in Equation 3.2 as:

$$E\left[\min\left(L(T),K\right)\right] = E\left[L(T)\cdot 1_{L(T)<K}\right] + K\cdot E\left[1_{L(T)\geq K}\right] \tag{3.23}$$

where the indicator function is defined to capture the risk of incurring a loss more than K:

$$1_{L(T)\geq K} = \begin{cases} 1 & if\ L(T)\geq K \\ 0 & if\ L(T)<K \end{cases}. \tag{3.24}$$

The expectation of Equation 3.24 is

$$E[1_{L(T)\geq K}] = \Pr(L(T)\geq K) = 1 - F(K) = \Phi(A(K)). \tag{3.25}$$

Note that the conditional and unconditional expectation of the loss distribution function is:

$$E\left[L(T)|Z\right] = \int_{-\infty}^{+\infty} L(T)\cdot f\left(L(T)|Z\right) dt$$

$$E\left[L(T)\right] = \int_{-\infty}^{+\infty} E\left[L(T)|Z\right]\cdot \phi(Z)\, dZ$$

Since $f\left(L(T)|Z\right)$ is a Dirac delta function, the first integration has the result:

$E(L(T)\,|\,Z) = L_0(T) = (1-R)\Phi\left(\dfrac{C_i(T)-\beta_i Z}{\sqrt{1-\beta_i^2}}\right)$ where $L_0(T)$ is the point of expectation of the loss distribution function. Hence the unconditional expectation of the loss distribution function is:

$$E[L(T)] = (1-R)\int_{A(K)}^{\infty}\Phi\left(\frac{C(T)-\beta Z}{\sqrt{1-\beta^2}}\right)\phi(Z)\, dZ.$$

Therefore the expectation of incurring a loss that is less than K is:

$$\begin{aligned} E[L(T)\cdot 1_{L(T)<K}] &= (1-R)\int_{A(K)}^{\infty}\Phi\left(\frac{C(T)-\beta Z}{\sqrt{1-\beta^2}}\right)\phi(Z)\, dZ \\ &= (1-R)\left(\int_{-\infty}^{+\infty} - \int_{-\infty}^{A(K)}\right)\Phi\left(\frac{C(T)-\beta Z}{\sqrt{1-\beta^2}}\right)\phi(Z)\, dZ \end{aligned}. \tag{3.26}$$

Let $a = A(K)$, we first solve the following the integral:

$$\int_{-\infty}^{a}\phi(x)\Phi\left(\frac{b-\beta x}{\sqrt{1-\beta^2}}\right)dx = \int_{-\infty}^{a}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\int_{-\infty}^{(b-\beta x)/\sqrt{1-\beta^2}}\frac{1}{\sqrt{2\pi}}e^{-y^2/2}\, dx\, dy.$$

Let $y = \dfrac{Z-\beta x}{\sqrt{1-\beta^2}}$, so $dy = \dfrac{dZ}{\sqrt{1-\beta^2}}$, so we have:

$$\int_{-\infty}^{a}\phi(x)\Phi\left(\frac{b-\beta x}{\sqrt{1-\beta^2}}\right)dx = \int_{-\infty}^{a}\frac{1}{\sqrt{2\pi}}e^{-x^2/2}\int_{-\infty}^{(b-\beta x)/\sqrt{1-\beta^2}}\frac{1}{\sqrt{2\pi}}e^{-y^2/2}\,dx\,dy$$

$$= \frac{1}{2\pi}\int_{-\infty}^{a}dx\int_{-\infty}^{b}\exp\left[-\frac{x^2}{2}-\frac{1}{2}\left(\frac{Z-\beta x}{\sqrt{1-\beta^2}}\right)^2\right]\frac{dZ}{\sqrt{1-\beta^2}}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{a}dx\int_{-\infty}^{b}\exp\left[-\frac{x^2-\beta xZ+Z^2}{2(1-\beta^2)}\right]\frac{dZ}{\sqrt{1-\beta^2}}$$

$$= \frac{1}{2\pi\sqrt{1-\beta^2}}\int_{-\infty}^{a}dx\int_{-\infty}^{b}dZ\exp\left[-\frac{x^2-\beta xZ+Z^2}{2(1-\beta^2)}\right]$$

$$= \Phi_2(a,b,\beta)$$

Here Φ_2 is the bi-variate normal cumulative distribution function that has the form:

$$\Phi_2(a,b,\rho) = \frac{1}{2\pi\sqrt{1-\rho^2}}\int_{-\infty}^{a}dx\int_{-\infty}^{b}dy\exp\left(-\frac{x^2-2\rho xy+y^2}{2(1-\rho^2)}\right).$$

Therefore Equation 3.26 equals:

$$E[L(T)\cdot 1_{L(T)<K}] = (1-R)\int_{A(K)}^{\infty}\Phi\left(\frac{C(T)-\beta Z}{\sqrt{1-\beta^2}}\right)\phi(Z)\,dZ$$

$$= (1-R)\left(\int_{-\infty}^{+\infty}-\int_{-\infty}^{A(K)}\right)\Phi\left(\frac{C(T)-\beta Z}{\sqrt{1-\beta^2}}\right)\phi(Z)\,dZ.$$

$$= (1-R)[\Phi_2(C(T),\infty,\beta)-\Phi_2(C(T),A(K),\beta)]$$

Since $\Phi_2(a,b,\rho)+\Phi_2(a,-b,-\rho)=\Phi(a)$, Equation 3.26 equals:

$$E[L(T)\cdot 1_{L(T)<K}] = (1-R)[\Phi_2(C(T),\infty,\beta)-\Phi_2(C(T),A(K),\beta)]$$
$$= (1-R)\Phi_2(C(T),-A(K),-\beta).$$

Now substitute Equation 3.27 into Equation 3.23 then the tranche survival curve can be obtained:

$$E[\min(L(T),K)] = E[L(T)\cdot 1_{L(T)<K}]+K\cdot E[1_{L(T)\geq K}]$$
$$= (1-R)\Phi_2(C(T),-A(K),-\beta)+K\cdot\Phi(A(K)). \qquad (3.27)$$

However, the equity tranche survival curve still remains to be found. Recall in Section 3.1 we have introduced the method to decompose the generic tranche into two equity tranches. The same can be applied with LHP model. Using the same notation, we have:

$$n_{oa} = \frac{N_{OA}}{A} = \frac{A - L_{OA}}{A} = 1 - \frac{L_{OA}}{A} = 1 - \frac{1}{A} \cdot \begin{cases} L_p, L_p < A \\ A, L_p > A \end{cases} = \begin{cases} 1 - \frac{L_p}{A}, L_p < A \\ 0, L_p > A \end{cases} = \begin{cases} 1 - \frac{l_p}{a}, l_p < a \\ 0, l_p > a \end{cases} = (1 - \frac{l_p}{a})^+$$

$$l_{oa} = \frac{L_{OA}}{A} = \begin{cases} \frac{L_p}{A}, L_p < A \\ 1, \ L_p > A \end{cases} = \begin{cases} \frac{l_p}{a}, l_p < a \\ 1, \ l_p > a \end{cases} = \min(\frac{l_p}{a}, 1) = \frac{1}{a}\min(l_p, a)$$

So the expectation of the equity tranche outstanding notional, i.e. the equity tranche survival curve, is:

$$Q_{oa} = E[n_{oa}] = E[1 - l_{oa}] = 1 - \frac{1}{a} E[\min(l_p, a)].$$

It can be seen that $E[\min(l_p, a)]$ is equivalent to $E[\min(L(T), K)]$ as defined by Equation 3.27. Therefore,

$$\begin{aligned} Q_{oa} &= 1 - \frac{1}{a}[(1 - R)\Phi_2(C(a), -A(a), -\beta) + a\Phi(A(a))] \\ &= 1 - \frac{1 - R}{a}\Phi_2(C(a), -A(a), -\beta) - \Phi(A(a)) \\ &= \Phi(-A(a)) - \frac{1 - R}{a}\Phi_2(C(a), -A(a), -\beta) \quad \text{since } \Phi(A) + \Phi(-A) = 1 \end{aligned} \quad . \qquad (3.27)$$

Substitute Equation 3.27 to Equation 3.3 with attachment and detachment points a and b, the tranche survival curve in terms of the equity tranches is expressed as:

$$Q_{ab} = \frac{b}{b - a}\left[\Phi(-A(b)) - \frac{1 - R}{b}\Phi_2(C(b), -A(b), -\beta)\right] - \frac{a}{b - a}\left[\Phi(-A(a)) - \frac{1 - R}{a}\Phi_2(C(a), -A(a), -\beta)\right]$$

Figure 3.5 displays the equity tranche survival curve with different detachment points, and Figure 3.6 displays the tranche survival curve for standard index tranches, both under the assumption of 40% recovery rate and 20% correlation. The tranche survival curves are then used to calculate the tranche spreads, the result of which is validated with the Dresdner Kleinwort (2008) spreadsheet model "*Opening the Black Box Part 2 – The Large Homogeneous Pool Model*".

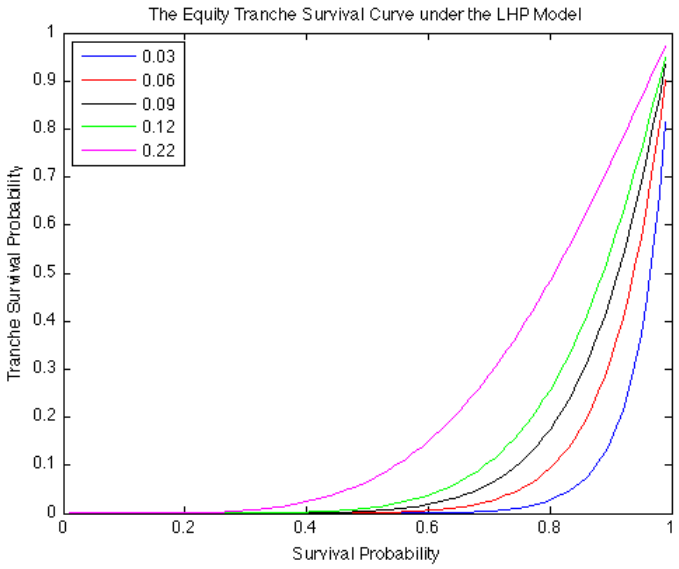


Figure 3.5 Equity Tranche Survival Curve under the LHP Model

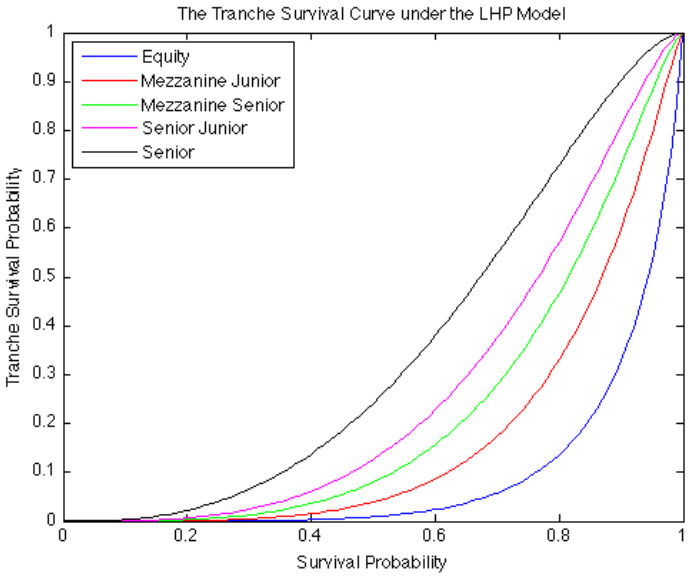


Figure 3.6 Tranche Survival Curve under the LHP Model

It is worth noticing that as the number of credits in the portfolio increases, the FHP tranche survival curve converges to the LHP tranche survival curve. As illustrated in Figure 3.7, a portfolio with 50 credits has a survival curve that is very close to the one built under the LHP model.

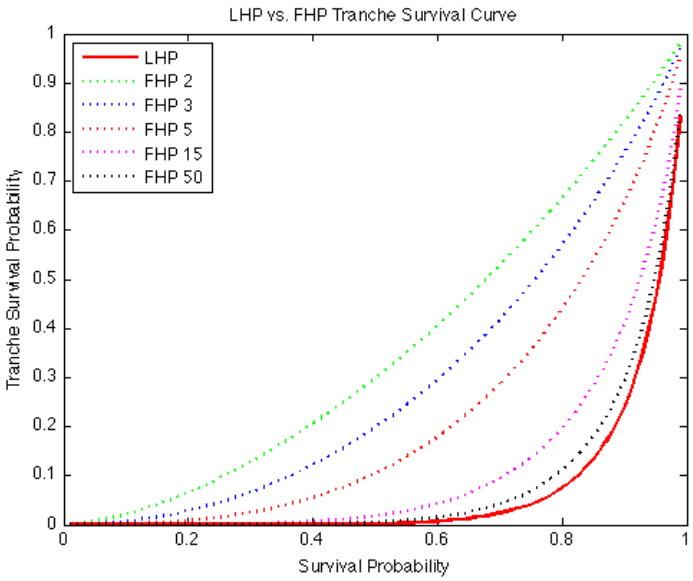


Figure 3.7 FHP Convergence to LHP

Chapter 4 The Stochastic Recovery Rate

Until this point, the discussion of models has assumed constant recovery rate. This was also the standard practice in credit derivatives modeling before the 2007 crisis. During the crisis, the CDX investment-grade index widened so much that the standard Gaussian copula model could not handle anymore. Indeed, under the assumption of 40% recovery rate, even using a default correlation level at 100% could only give a tranche spread of 39 bps[10] where the real market price was 55 bps (Amraoui & Hitier 2008). Moreover, the spreads of the super senior tranche (with attachment and detachment point at 60% and 100%), who used to have no value because under 40% recovery rate even if all the credits in the portfolio have defaulted, the loss would not exceed 60%, started to rise. Therefore researchers started to re-consider the deterministic recovery rate assumption.

It is very difficult to forecast a company's recovery rate because it is driven by many factors. Empirical recovery rates present a high variance, making the confidence of forecasting very low (Andersen & Sidenius 2004). Among all the models, Amraoui and Hitier proposed the one that now has become most widely approved (Merrill Lynch 2009). They specified that recovery rate should be a deterministic function of the market factor. Since the market factor, which reveals the systematic risk, is stochastic, the recovery rate is indirectly stochastic.

In the Amraoui-Hitier (AH) stochastic recovery model, a new parameter was introduced – the recovery markdown denoted by R_0. The recovery rate is modeled to range from R_0 to 100% while maintaining the single name CDS valuation properties (Amraoui & Hitier 2008). That is, the following condition must be maintained:

$$E\left[\left(1-\tilde{R}(Z)\right)1_{\tau<t}\right]=(1-R)\,p(t) \tag{4.1}$$

where $\tilde{R}$ is the stochastic recovery rate, R is the constant recovery rate for CDS, and $p(t)$ is the conditional default probability specified in Equation 3.18.

Analytically we have:

[10] bps stands for basis points. 1 basis point is a unit equals to 0.0001.

$$\begin{aligned} E\left[\left(1-\tilde{R}(Z)\right)1_{\tau<t}\right] &= \int_{-\infty}^{+\infty} E\left[\left(1-\tilde{R}(Z)\right)1_{\tau<t}\middle|Z\right]\phi(Z)\,dZ \\ &= \int_{-\infty}^{+\infty}\left(1-\tilde{R}(Z)\right)E\left[1_{\tau<t}\middle|Z\right]\phi(Z)\,dZ \\ &= \int_{-\infty}^{+\infty}\left(1-\tilde{R}(Z)\right)p(t|Z)\phi(Z)\,dZ \end{aligned} \tag{4.2}$$

Introducing R_0 and define the following relationship:

$$\left(1-\tilde{R}(Z)\right)p(t|Z) = (1-R_0)\,p_0(t|Z) \text{ where } p_0(t|Z) = \Phi\left(\frac{\Phi^{-1}(p_0(t)) - \beta Z}{\sqrt{1-\beta^2}}\right).$$

Therefore Equation 4.2 becomes:

$$\begin{aligned} E\left[\left(1-\tilde{R}(Z)\right)1_{\tau<t}\right] &= \int_{-\infty}^{+\infty}\left(1-\tilde{R}(Z)\right)F(t|Z)\phi(Z)\,dZ \\ &= (1-R_0)\int_{-\infty}^{+\infty}F_0(t|Z)\phi(Z)\,dZ \quad . \\ &= (1-R_0)F_0(t) \end{aligned}$$

Now the condition specified in Equation 4.1 becomes

$(1-R_0)F_0(t) = (1-R)F(t)$,

from which the following equations hold:

$$F_0(t) = \frac{1-R}{1-R_0}F(t) \qquad \text{and} \qquad \tilde{R}(Z) = 1-(1-R_0)\frac{F_0(t|Z)}{F(t|Z)}.$$

Now to illustrate the relationship between the recovery rate and the market factor, we can plot the conditional default probability and the stochastic recovery rate with different correlation levels. The results are presented from Figure 4.1 to Figure 4.3. The underlying survival probability is set at 80%.

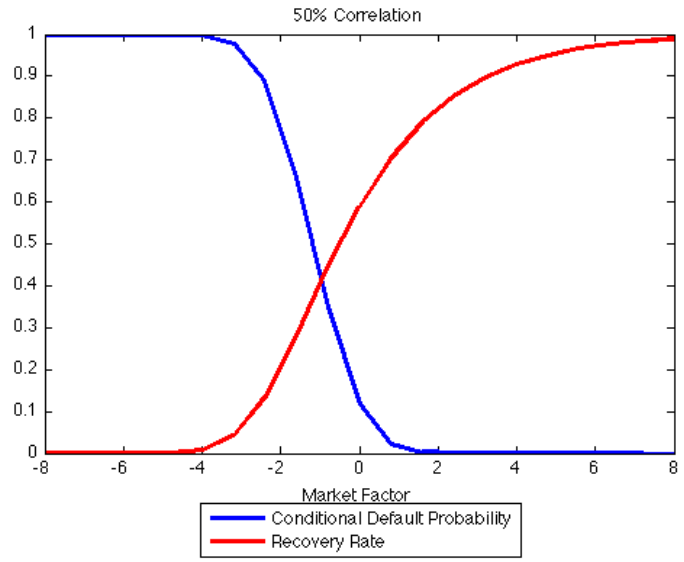


Figure 4.1 Conditional Probability vs. Recovery Rate with 50% Correlation

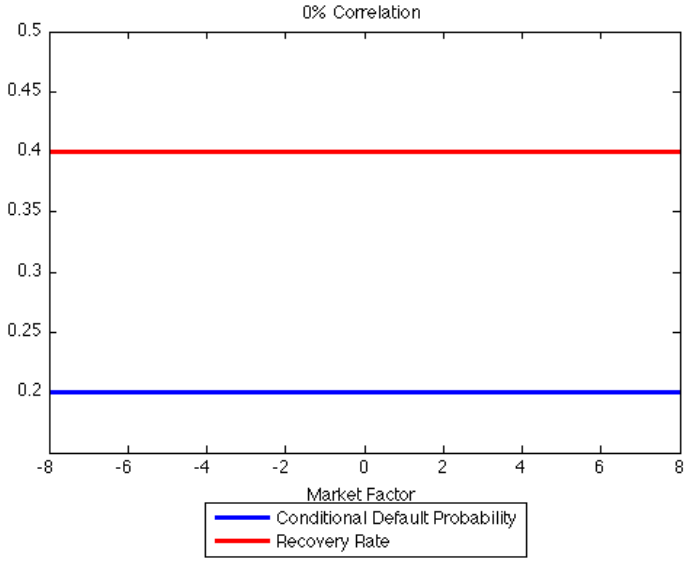


Figure 4.2 Conditional Probability vs. Recovery Rate with 0 Correlation

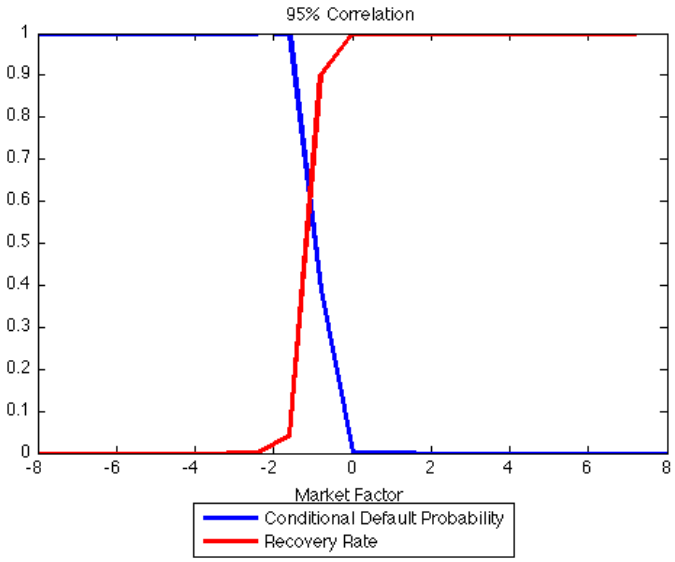


Figure 4.3 Conditional Probability vs. Recovery Rate with 95% Correlation

The stochastic recovery rate is negatively determined by the systematic risk, meaning that when the risk is high, the recovery rate is low. Under 50% correlation, it can be seen that the recovery rate increases smoothly when the default probability decreases, meaning that the market condition is good, and vice versa. When the credits are independent, i.e. when correlation is zero, the recovery rate behaves as the CDS constant recovery, which is in accordance with the Gaussian copula model. When the correlation is very high at 95%, the recovery rate behaves like a step function where it increases sharply to 100% when the market factor is high, and decreases steeply to 0 when the market factor is low. These results are consistent with the ones Merrill Lynch (2009) paper "*Coping with the Copula*".

The idea of stochastic recovery rate can be implemented with both the LHP and the FHP model. In this project it is implemented with the later. Recall that in Section 3.1, the expected tranche outstanding notional is specified as

$$E[n_{oa}(t) \mid Z] = \sum_{k=0}^{\lfloor g \rfloor} \left(1 - \frac{k}{g}\right) \cdot \binom{N_c}{n} p(T \mid Z)^k (1 - p(T \mid Z))^{N_c - k}.$$

With the stochastic recovery rate assumption, the only change is to make the variable g as a function of Z. That is, the conditional survival probability is:

$$Q_{oa}(t \mid Z) = \sum_{k=0}^{\lfloor g(Z) \rfloor} \left(1 - \frac{k}{g(Z)}\right) \cdot \binom{N_c}{n} p(T \mid Z)^k (1 - p(T \mid Z))^{N_c - k}$$

where $g(Z) = \dfrac{N_c a}{1 - \tilde{R}(Z)}$ with $\tilde{R}(Z) = 1 - (1 - R_0)\dfrac{F_0(t|Z)}{F(t|Z)}$ as specified above.

Therefore, the unconditional tranche survival curve is obtained by integration over the density of the market factor:

$$Q_{oa} = \int_{-\infty}^{+\infty} Q_{oa}[t \mid Z]\phi(Z)\, dZ.$$

Similar to the LHP model, the generic tranche survival curve under the FHP-AH model is also calculated as in Equation 3.21. The equity and generic tranche survival curves generated with Matlab are shown in Figure 4.4 and Figure 4.5 respectively.

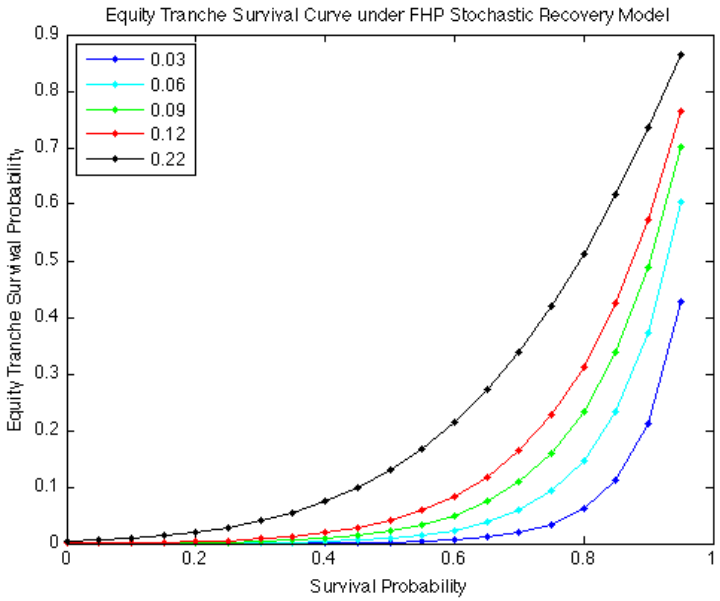


Figure 4.4 Equity Tranche Survival Curve under the FHP-AH Model

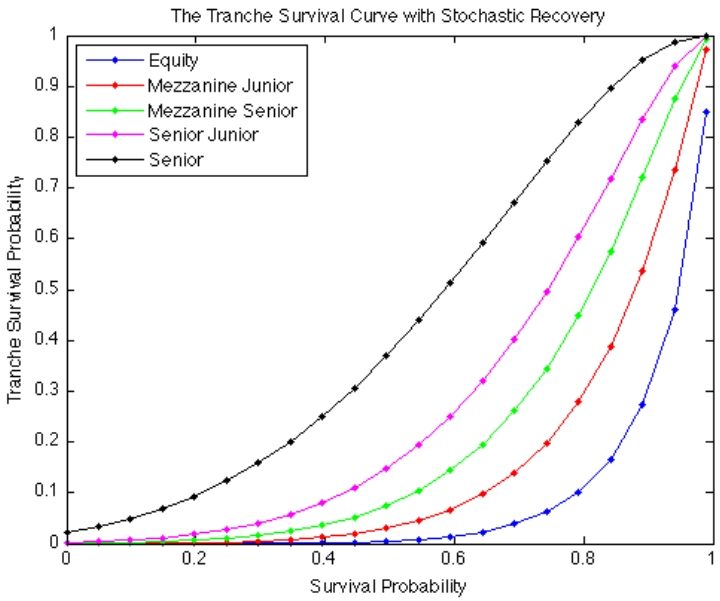


Figure 4.5 Tranche Survival Curve under the FHP-AH Model

Notice that all the survival curves under the FHP-AH model end at 0.99. This is done on purpose to avoid the case when survival probability equals to one. The reason is that the stochastic recovery rate has conditional default probability on the denominator, which equals to zero when survival probability is one, resulting in computation error.

Although there is no data source to justify the results, one observation can be done to show that the model is correct. With the definition of recovery markdown, when it equals to the constant CDS recovery rate, the survival curve with stochastic recovery assumption should converge to the FHP survival curve. Indeed this is true:

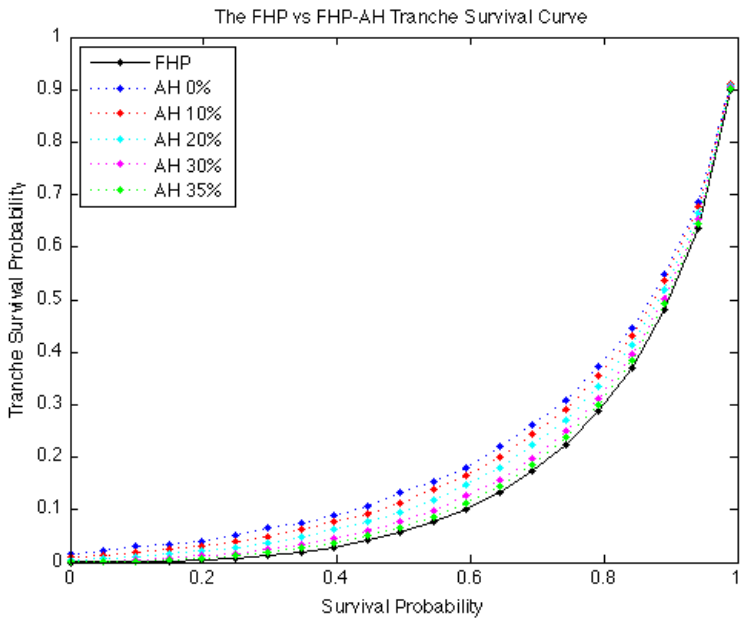


Figure 4.6 AH Convergence to FHP

As illustrated with Figure 4.6, when the recovery markdown equals to 35%, the resulting survival curve is sufficiently close to the FHP curve with the recovery rate set at 40%.

With stochastic recovery rate, the spread of the super senior tranches can be explained. More importantly, it drives down and flattens the base correlation skew, which will be the topic of the next chapter.

Chapter 5 A Discussion on Default Correlation

This chapter first discusses the concepts of compound correlation and base correlation. Then an implementation of base correlation bootstrap is presented. The ultimate goal of bootstrapping the base correlation is to price off-market tranches consistently with the market tranches. The previous set up of this report has made it possible by decomposing generic tranches into equity tranches.

5.1 Implied Correlation and Correlation Smile

So far, the discussion of models has based on the assumption of a flat correlation curve. This value can be calculated by observing the market price of standard tranches and then reverse the CDO pricing box based on the Gaussian copula model, and hence it is called implied correlation. The implied correlation for generic tranches is called compound correlation.

Compound correlation is extensively used in the modeling of correlation products. However, this is not a true representation of the assets in the underlying portfolio. Simply taking the market tranche spreads from the CDX index, O'Kane (2008, p. 368) showed that the implied correlation for different tranches exhibits a 'smile' like shape when plotted as an increasing function of the tranche detachment point. Hager and Schöbel (2005) also pointed out that some market spreads might not be obtainable by choices of correlation.

There are several reasons for the existence of correlation smile. First of all, the pricing models are built under the following assumptions: the Gaussian copula dependency structure, constant recovery rate, and flat correlation curve. Second, the spread is also driven by the force of demand and supply, with mezzanine tranche being extremely popular among investors (Hager and Schöbel 2005).

Although simple to use, compound correlation has some serious shortcomings because of the smile shape. First, it fails to preserve the no-arbitrage condition. This is an important condition that means the sum of tranche expected loss across the whole capital structure

should equal to expected portfolio loss. According to Equation 3.2, the expected loss of a tranche in face value is given by:

$$\text{Expected Loss} = E_{\rho(K_1,K_2)}[\min(L(T),K_2) - \min(L(T),K_1)],$$

which is calculated according to Equation 3.27 with correlation $\rho(K_1,K_2)$. Now assume that there is a set of contiguous tranches. The total expected loss will be:

$$\begin{aligned}\text{Total Expected Loss} = {} & E_{\rho(0,K_1)}[\min(L(T),K_1) - \min(L(T),0)] \\ & + E_{\rho(K_1,K_2)}[\min(L(T),K_2) - \min(L(T),K_1)] \\ & + \ldots \\ & + E_{\rho(K_{M-1},K_M)}[\min(L(T),K_M) - \min(L(T),K_{M-1})] \\ & + E_{\rho(K_M,1)}[\min(L(T),1) - \min(L(T),K_M)]\end{aligned} \quad . \tag{5.1}$$

The expected portfolio loss is

$$E_\rho[L(T)] = \frac{1}{N}\sum_{i=1}^{N} E[L_i(T)] \tag{5.2}$$

where $E\left[L_i(T)\right]$ is the expected loss of credit i to time T.

Equation 5.1 can only reduce to Equation 5.2 if the correlation curve is flat so that the terms in Equation 5.1 will cancel. With correlation smile, this condition cannot be preserved.

Another shortcoming of compound correlation is that it cannot be used to price bespoke tranches who has the same maturity and reference portfolio, but with different attachment/detachment points. If compound correlation is to be used to price the non-standard tranche, then an interpolation scheme is needed for the correlation curve. However this is very difficult to achieve because compound correlation is a function of two variables (O'Kane 2008, p. 371).

Given the above, nowadays compound correlation has been overtaken by base correlations when it comes to CDO pricing. Base correlation refers to the correlation for the base tranches, i.e. the equity tranches transformed from a generic tranche. Consider the expected loss for a K_1-K_2 tranche again:

$$\text{Expected Loss} = E_{\rho(K_2)}[\min(L(T),K_2)] - E_{\rho(K_1)}[\min(L(T),K_1)]$$

Therefore the total expected loss for contiguous tranches is:

$$\begin{aligned}\text{Total Expected Loss} = & E_{\rho(K_1)}[\min(L(T),K_1)] - E_{\rho(K_0)}[\min(L(T),0)] \\ & + E_{\rho(K_2)}[\min(L(T),K_2)] - E_{\rho(K_1)}[\min(L(T),K_1)] \\ & + \ldots \\ & + E_{\rho(K_{M-1})}[\min(L(T),K_{M-1})] - E_{\rho(K_{M-2})}[\min(L(T),K_{M-2})] \\ & + E_{\rho(1)}[\min(L(T),1)] - E_{\rho(K_{M-1})}[\min(L(T),K_{M-1})]\end{aligned}$$

This equation can be reduced to the expected portfolio loss so that the no-arbitrage condition is preserved.

5.2 Base Correlation Bootstrap

Similar to the bootstrap of zero rates, this bootstrap works from the lowest attachment point to the highest attachment point. Recall that in Section 3.1, each generic tranche survival curve is shown to be a combination of two equity tranche survival curves. In the bootstrapping process, each equity tranche is associated with its own base correlation, the value of which is found by a one-dimensional root search algorithm.

For example, for tranche with attachment and detachment points at 3% and 7%, the survival curves for tranches 0-3% and 0-7% are first expressed in terms of correlation ρ_1 and ρ_2. Note that ρ_1 is the already calculated compound correlation for the equity tranche 0-3%. So the tranche survival curve is in fact a linear function in terms of ρ_2. Finally using the fact that the present value of a tranche should equal to zero, ρ_2 can be solved. This scheme is implemented with the LHP, the FHP, and the FHP-AH pricing model. The final result is shown in Figure 5.1:

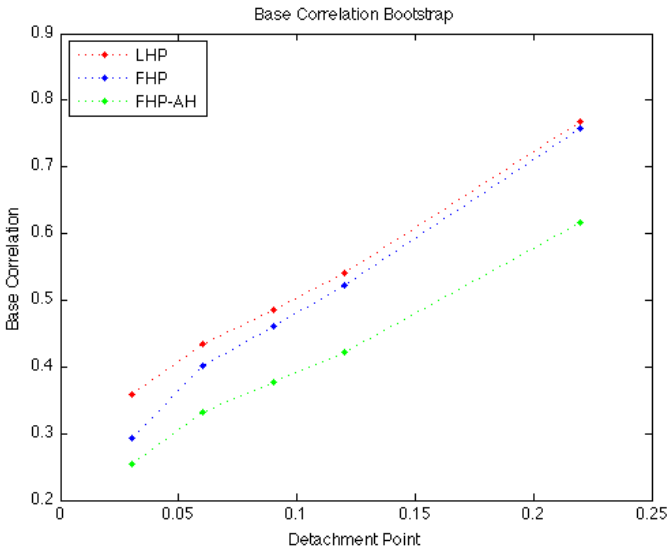


Figure 5.1 Base Correlation Bootstrap under Three Models

The result shows that with the stochastic recovery assumption, the base correlation skew is lower and flatter than that produced with the LHP and the FHP model. This is a desirable feature because it reduces the dependency between the priced tranches and the implied parameters and eases the computation of bespoke tranche pricing (Amraoui, Cousot, Hitier & Laurent 2009).

Chapter 6 Programming with Matlab

As stated at the beginning of Chapter 1, Matlab is extensively used throughout the project. Quoted from the MathWorks website:

> *'MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.'*

Matlab is especially useful for technical computing and computational finance. Indeed, it is very popular within the financial service industry – many top investment banks, commercial banks, asset management firms, and hedge funds use products from the Matlab family to perform research, modeling, analysis, etc. (MathWorks 2009).

In this project where mathematics and graphics are heavily involved, Matlab can perform matrix computation and produce graphs with the expressions written in the mathematical form. This has dramatically raised the efficiency with programming. The efficiency in performing analysis and obtaining results is an important factor in financial computing because the market is always on the change and computational delay is the least desirable feature.

When programming with Matlab, there are a few advices that should be followed to ensure efficiency.

1. Same as programming with Java, comment is important in Matlab programming. It helps to clarify the content for the programmer as well as for other people who will read the code.
2. Be consistent with naming conventions. This is also to help the programmer as well as other people.
3. Do not compute the same value twice. That is to say, any value that is going to be used more than once should be calculated in advance and assigned a name. For example, in the function `LHPEquityTrancheSurvivalCurve`, the quantity β is used in several expressions. Since computing the square root is computation-time consuming, it should be calculated beforehand.

4. It is not advisable to use nested functions in one file unless the second function exclusively written for the main function. For example, many functions for the CDO pricing box requires integration. Since the trapezoidal rule is used with all of them, a function that explicitly implements trapezoidal integration should be separately programmed.
5. Try to avoid loops. Matlab is a vectorized language, meaning that each variable is presented as a matrix. For example, a variable $x = 2$ is seen as a 1-by-1 matrix by Matlab. Therefore, it is much more timesaving to perform vector computations than to perform the computation in a for loop.
6. Use built-in functions wherever possible. Since Matlab is designed for technical computing, there are plenty of useful built-in functions that can speed up the computation. For example, the `interp1` function can perform linear interpolation in the same way as Equation 2.13 does, but using the build-in function can avoid any programmer mistakes and is much faster to implement.
7. Do not forget the debugging tool. The debugging tool is a very useful feature in Matlab. It allows the program to stop in the middle of execution for the programmer to check the variables and values so as to find any mistakes.
8. Publish the M-file after testing it. The scripts produced with the Matlab editor are called M-files. The publish tool allows any M-file to be output to a specific file format and then to be used elsewhere for illustration or explanation purposes. Figure 6.1 is a published HTML file showing the result of the

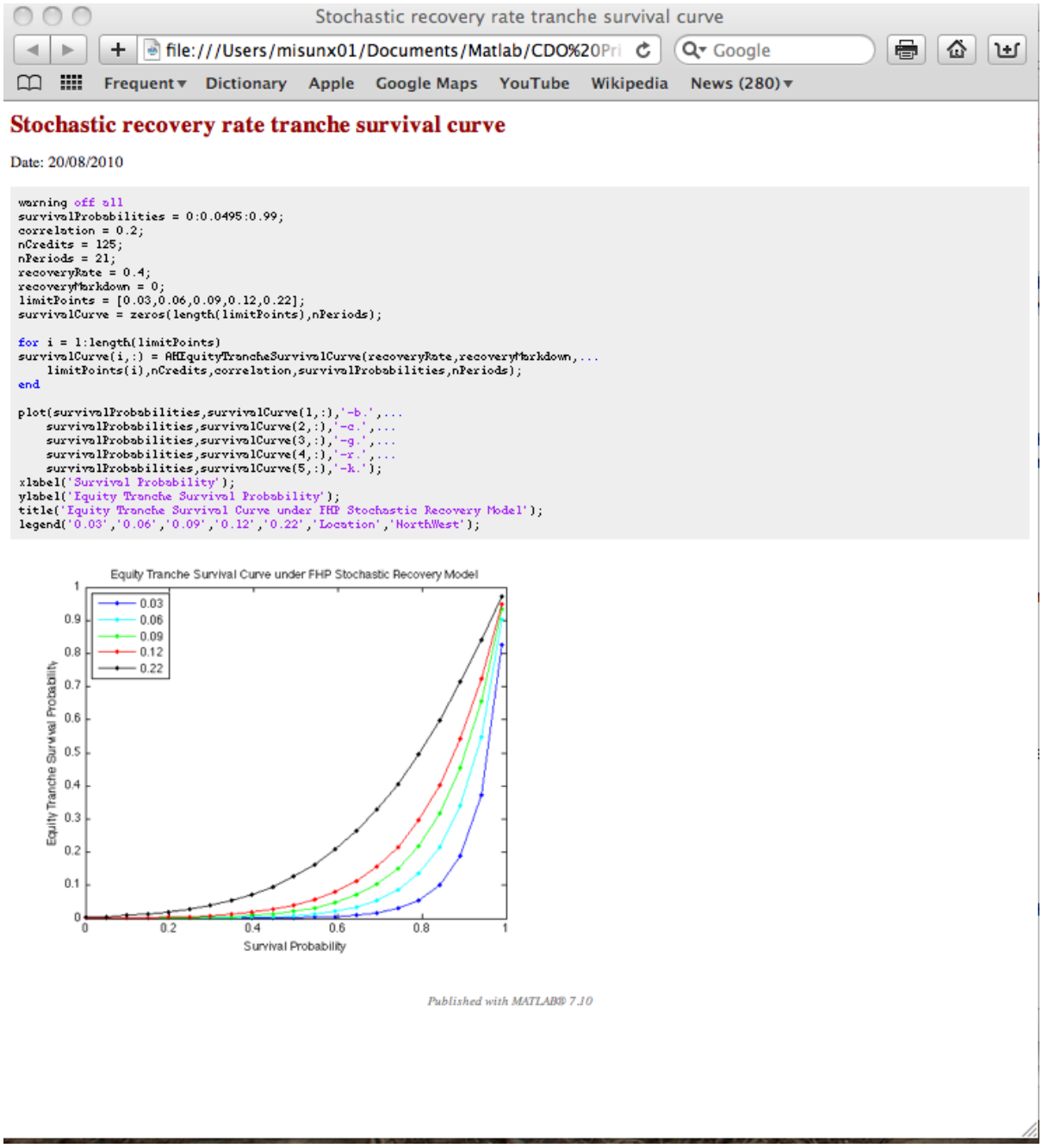


Figure 6.1 The Published M-File in HTML Format

Chapter 7 Summary and Evaluation

This chapter first provides a summary of the project. Then a critical evaluation in the author's point of view is presented.

7.1 Summary

The main goal of this project is to explore the pricing models for credit derivatives and program them with Matlab. In the mean time, the bootstrapping method is implemented with the CDS survival curve and base correlation.

The starting point is credit default swaps, which is the backbone of the credit derivative market. CDS survival probability is the fundamental element for the pricing of CDOs. For efficiency and accuracy, the bootstrapping method is used together with interpolation to determine the CDS survival curve.

Collateralized debt obligation is one of the most important correlation products in the credit derivative market. Its premium and protection payments are determined by the amount of defaults in the underlying portfolio, which in turn is a function of default correlation.

In order to obtain the CDO survival curve, some assumptions are made with the Gaussian copula model. Based on these assumptions the finite homogeneous portfolio model and the large homogeneous portfolio model are implemented to calculate the CDO survival curve. The results from Chapter 3 showed that the FHP model converges to the LHP model as the number of credits in the portfolio increases above 50. Since usually a CDO index is based on 125 credits, using the LHP model to build the tranche survival curve can produce acceptable results with less computational time.

The recent market trend is to model CDO survival curve with stochastic recovery rate. Among the others, the Amraoui-Hitier model is studied and then implemented with the FHP model. The results showed that the FHP model is a limiting case of the AH stochastic recovery model.

The aim of the whole process is to bootstrap base correlation so that non-standard CDO tranches can be priced consistently with the standard index CDO tranches. Under the three models, the FHP-AH model produces the most desirable correlation skew.

However, in order to price the bespoke CDO tranches, the base correlation skew needs to be interpolated so that the corresponding base correlation can be determined for the bespoke tranche. This is not implemented in this project due to limited time.

7.2 Critical Evaluation

This project is not about software engineering. Therefore the outcome of this project is not a software system, but a system of Matlab functions. The usage of the functions is to provide numerical or graphical results that can be used in the pricing of credit derivatives and the analysis of the market.

A major shortcoming of the project is that there is no user-friendly GUI built. It is unrealistic to expect end-users to write an M-file and call a function to obtain the result. Therefore the system is more suitable for professionals to make use of. Given more time, a GUI should be built for the functions so that they can serve common investors as an analytical tool.

Although time is very limited for this amount of work, all of the functions are tested to be valid. The results are either compared with banks' papers, or with the model built by my colleague at the FSA.

Because FSA only has a limited number of licenses available for Matlab, I had to build all the models on my own laptop. Again, since time was constrained, I did not run my codes on the FSA computer so I cannot assess their efficiency. For example, the bootstrap of base correlation under the FHP-AH model can take up to 20 minutes. This may due to inefficient function design or low computational power of my laptop.

However, I have followed the team's internal standard for code writing, and I have commented the code wherever necessary. Therefore I believe that the codes are clear and

understandable by other professionals. If anyone is interested in extending and improving the functions, there should not be any major difficulties.

If I had the chance to do this project again, I will put more effort on studying the default correlation between the credits. As cited by Elizalde (2005) in his paper,

> "*Default correlation and default dependency modeling is probably the most interesting and also the most demanding open problem in the pricing of credit derivatives.*"

While doing the project, I have focused on building the model, but I did not consider in depth the assumed structure of default correlation that underlies all the models. The usual practice is to use equity return correlations, but some agencies use historical implied default correlations (Elizalde 2005). Clearly, since default correlation is a variable in all models, different measure of it can lead to different shape of the survival curve.

Nevertheless, although there is still a lot to explore in the credit derivatives field, I am satisfied by my achievement of building all these models and doing all the analyses in 10 weeks. I have learned much knowledge about credit derivatives and Matlab programming, which I am sure will benefit me in the future.

Appendix A User Manual

To obtain the results that are presented in this report, it is the test file that should be run. Although the name of the M-file generally indicates the functionality, the following table further specifies the usage of each function.

CDS Pricing Box	
CDSLegs	Calculates the CDS risky annuity and the protection leg.
CDSLegsTest	Test the CDSLegs function.
coupontest	Testing the calculation of the premium and the protection legs.
discountFactors	Calculates the discount factors given the yields and maturities.
discountFactorsTest	Test the discountFactors function.
forwardDefaultRate	Calculates the forward default given the zero default rates and maturities.
survivalProbability	Calculates the CDS survival probability given recovery rate, discount factors, maturities, and spreads.
survivalProbabilityTest	Test the survivalProbability function.
zeroDefaultRateBootstrap	Bootstrap the zero default rates.
zeroDefaultRateBootstrapTest	Test the zeroDefaultRateBootstrap function.
zeroRateLogLinear	Function that performs the log-linear interpolation of the zero default rate.
zeroRateLogLinearTest	Test the zeroRateLogLinear function.
CDO Pricing Box	
AHEquityTrancheSurvivalCurve	Calculates the equity tranche survival curve under the FHP-AH model.
AHEquityTrancheSurvivalCurveTest	Test the AHEquityTrancheSurvivalCurve function.

AHTrancheSpread	Calculates the tranche spread using the tranche survival curve produced with the FHP-AH model.
AHTrancheSpreadTest	Test the AHTrancheSpread function.
AHTrancheSurvivalCurve	Calculates the tranche survival curve under the FHP-AH model.
AHTrancheSurvivalCurveTest	Test the AHTrancheSurvivalCurve function.
baseCorrelationBootstrap	Bootstrap the base correlation under any specified model.
baseCorrelationBootstrapTest	Test the baseCorrelationBootstrap function.
bivariateCDF	Calculates the bivariate normal cumulative distribution function.
bivariateCDFTest	Test the bivariateCDF function.
equityTrancheImpliedCorrelation	Calculates the compound correlation of equity tranches.
equityTrancheImpliedCorrelationTest	Test the equityTrancheImpliedCorrelation function.
FHPDefaultCreditDistribution	Model the unconditional portfolio loss distribution under the FHP model. The result is the loss distribution for the number defaults.
FHPDefaultCreditDistributionTest	Test the FHPDefaultCreditDistribution function.
FHPDefaultDistributionTest	Plot the portfolio loss distribution across different maturities.
FHPDefaultTimeDistribution	Model the unconditional portfolio loss distribution under the FHP model. The result is the loss distribution across different maturities.
FHPDefaultTimeDistributionTest	Test the FHPDefaultTimeDistribution function.
FHPEquityTrancheSurvivalCurve	Calculates the equity tranche survival curve under the FHP model.
FHPEquityTrancheSurvivalCurveTest	Test the FHPEquityTrancheSurvivalCurve function.
FHPTrancheSpread	Calculates the tranche spreads under the FHP model.

FHPTrancheSpreadTest	Test the FHPTrancheSpread function.
FHPTrancheSurvivalCurve	Calculates the tranche survival curve under the FHP model.
FHPTrancheSurvivalCurveTest	Test the FHPTrancheSurvivalCurve function.
LHPEquityTrancheSurvivalCurve	Calculates the equity tranche survival curve under the FHP model.
LHPEquityTrancheSurvivalCurveTest	Test the LHPEquityTrancheSurvivalCurve function.
LHPTrancheSpread	Calculates the tranche spreads under the LHP model.
LHPTrancheSpreadTest	Test the LHPTrancheSpread function.
LHPTrancheSurvivalCurve	Calculates the tranche survival curve under the LHP model.
LHPTrancheSurvivalCurveTest	Test the LHPTrancheSurvivalCurve function.
LHPTrancheSurvivalProb	Calculates the tranche survival curve under the LHP model using Equation 3.2
marketCondition	Calculates the conditional default probability and the stochastic recovery rate.
marketConditionTest	Plot the conditional default probability and the stochastic recovery rate to illustrate the influence of market condition on recovery rate.
tranchePresentValue	Calculate the present value of a tranche.
tranchePresentValueTest	Test the tranchePresentValue function.
trapezium	Calculates any integral under the trapezoidal rule.
trapeziumTest	Test the trapezium function.

Appendix B Code Listing

Due to the large amount of codes, only the most important ones are listed here. Not all M-files that are test functions will be included. Moreover, support functions such as the `trapezium` and the `bivariateCDF` will not be listed either. All the codes can be found in the CD.

CDSLegs.m

```
% This function calculates the risky annuity and the
protection leg of a
% CDS.
% Date: 27/07/2010

function [riskyAnnuity,protectionLeg] =
CDSLegs(recoveryRate,discountFactors,...
    survivalProbabilities,maturities)

    m = length(discountFactors);
    n = length(survivalProbabilities);
    k = length(maturities);

    if(m ~= n || m ~= k || n ~= k)
        error('The lengths of the survival probability array,
maturity array, and the discount factor array must be equal');
    end

    lgd = 1 - recoveryRate;
    delta = diff([0,maturities]);
    survivalProbabilityDiff = -
diff([1,survivalProbabilities]);

%   Obtain the products
    regularCouponProducts =
discountFactors.*survivalProbabilities.*delta;
    accruedCouponProducts =
discountFactors.*survivalProbabilityDiff.*delta/2;
    protectionLegProducts =
lgd*discountFactors.*survivalProbabilityDiff;

%   Do cumulative sum along the product array and obtain the
array of
%   coupons.
%     premiumArray =
cumsum(regularProdArray+accruedProdArray);
%     protectionArray = cumsum(protectionProdArray);

%   Obtain the final risk annuity and protection leg
    riskyAnnuity =
sum([regularCouponProducts,accruedCouponProducts]);
    protectionLeg = sum(protectionLegProducts);
end
```

zeroDefaultRateBootstrap.m

```
% Bootstrap the zero default rate.
% Date: 20/07/2010-21/07/2010

function zeroDefaultRateHandle =
zeroDefaultRateBootstrap(recoveryRate,...
    discountFactors,maturities,refMaturities,refSpreads)

    nMaturities = length(refMaturities);

%   Initial estimate of the first zero default rate.
    initZeroDefaultRate = refSpreads(1)/(1-recoveryRate);

    maturityPosition = zeros(1,nMaturities);
    zeroDefaultRates = zeros(1,nMaturities);

    for i = 1:nMaturities
%       Find the index of the reference maturiy at the whole
maturity
%       array.
        index = find(maturities == refMaturities(i));

        if length(index) ~= 1
            error('Wrong reference maturities provided.');
        end

        maturityPosition(i) = index;
        currentMaturities = maturities(1:maturityPosition(i));

        presentValueHandle =
@(lastZeroDefaultRate)presentValue(lastZeroDefaultRate,...

zeroDefaultRates(1:i),recoveryRate,discountFactors,...

currentMaturities,refMaturities(1:i),refSpreads(i));

        zeroDefaultRates(i) =
fzero(presentValueHandle,initZeroDefaultRate);

%       Set the new initial value for the fzero function.
        initZeroDefaultRate = zeroDefaultRates(i);
    end
%   Output the handle so that calculation can be done after
each iteration.
    zeroDefaultRateHandle =
@(interpMaturities)zeroRateLogLinear(interpMaturities,...
        zeroDefaultRates,refMaturities);

end

function PV = presentValue(lastZeroDefaultRate,
```

```
zeroDefaultRates,...

recoveryRate,discountFactors,currentMaturities,refMaturities,r
efSpread)

    zeroDefaultRates(end) = lastZeroDefaultRate;

    zeroDefaultRateHandle =
@(interpMaturities)zeroRateLogLinear(interpMaturities,...
        zeroDefaultRates,refMaturities);

    currentZeroDefaultRates =
zeroDefaultRateHandle(currentMaturities);

    survivalProbabilities = exp(-
currentZeroDefaultRates.*currentMaturities);

%   Find the index of the discount factors.
    discountFactorUsed =
discountFactors(1:length(currentMaturities));

%   Calculate the PV using existing functions.
    [riskAnnuity,protectionLeg] =
CDSLegs(recoveryRate,discountFactorUsed,...
        survivalProbabilities,currentMaturities);

    PV = protectionLeg-refSpread*riskAnnuity;
end
```

zeroDefaultRateBootstrapTest.m

```
%% Bootstrap the zero default rate
% This file tests the bootstrapzerodefaultrate function. Using
the
% bootstrapped zero default curve, I can derive the survival
probability
% curve and the forward default curve.
%
% Some comparasons have also been done. One is for the
spreads, and another
% is for the zero default rates.

%% Values of the parameters
recoveryRate = 0.30;
% refSpreadArray = [0.0988,0.0998,0.0896,0.0794,0.0594];

refSpreadArray =
[0.1008,0.0988,0.0995,0.0998,0.0972,0.0945,0.0915,0.0896,...

0.0863,0.0837,0.0813,0.0794,0.0763,0.0737,0.0714,0.0695,0.0665
,0.0639,...
    0.0615,0.0594];

discountFactorArray =
[0.9941,0.9884,0.9825,0.9763,0.9698,0.9628,0.9555,...

0.9478,0.9399,0.9317,0.9231,0.9144,0.9055,0.8964,0.8872,0.8779
,0.8685,...
    0.8591,0.8496,0.8400];

maturityArray = (1:20)*365.25/360/4;
% refMaturityArray = maturityArray([2,4,8,12,20]);
refMaturityArray = maturityArray(1:20);

%% Bootstrap the zero default rates.
% _The output of this function is a function handle._
zeroDefaultRateHandle =
zeroDefaultRateBootstrap(recoveryRate,...
    discountFactorArray, maturityArray, refMaturityArray,
refSpreadArray);
zeroDefaultRateArray = zeroDefaultRateHandle(maturityArray);


%% Plot and compare the boostrapped zero default rate
% _The aim is to compare the bootstrapped rates to the
original rates obtained from
% the original survival probability._
origSurvivalProbArr =
[0.9643,0.9305,0.8976,0.8656,0.8391,0.8151,0.7941,...

0.7730,0.7579,0.7432,0.7287,0.7145,0.7066,0.6990,0.6914,0.6837
```

```
,0.6822,...
    0.6806,0.6791,0.6776];
origZeroDefaultRateArr = -
log(origSurvivalProbArr)./maturityArray;
figure(1);
plot(maturityArray,origZeroDefaultRateArr,'m.-
',maturityArray,zeroDefaultRateArray,'go-');
xlabel('Maturity(years)');
ylabel('Zero Default Rate');
legend('Original zero default rate','Bootstrapped zero default
rate');
title('Zero Default Curve Comparison');

%% Plotting the curves.
% _The plot function do not accept function handles._
maturities = 0:0.01:6;
%   outputcurves(zeroDefaultRateHandle,maturities);
%   Output zero default rate from the handle
zeroDefaultRateArr = zeroDefaultRateHandle(maturities);
refZeroDefaultRateArr =
zeroDefaultRateHandle(refMaturityArray);
figure(2);
plot(maturities,zeroDefaultRateArr,'b-
',refMaturityArray,refZeroDefaultRateArr,'r*');
xlabel('Maturity(years)');
ylabel('Zero Default Rate');
legend('Bootstrapped zero default curve','Reference zero
default rate');
title('Bootstrapped Zero Default Curve');

%   Forward default rate can be obtained from the zero default
rates.
fwdDefaultRateArr =
forwardDefaultRates(zeroDefaultRateArr,maturities);
figure(3);
plot(maturities(2:end), fwdDefaultRateArr, 'r-');
% stairs(maturities(2:end), fwdDefaultRateArr, 'r-');
xlabel('Maturity(years)');
ylabel('Forward Default Rate');
title('Forward Default Curve');

%   Survival probabilities can be obtained from zero default
rates as well.
survivalProbArr = exp(-zeroDefaultRateArr.*maturities);
figure(4);
plot(maturities, survivalProbArr, 'g-');
xlabel('Maturity(years)');
ylabel('Survival Probability');
title('Survival Probability Curve');

%% Recalculate the CDS spreads to check the results.
nMaturities = length(refMaturityArray);
```

```
maturityPosition = zeros(1,nMaturities);
recalculatedSpreadArray = zeros(1,nMaturities);

% Solve the direct problem of finding the CDS spreads using
the JP Morgan
% given discount factors and survival probabilities.
origSpreadArray = zeros(1,length(maturityArray));
for j = 1:length(maturityArray)
    [riskyAnnuity,protectionLeg] =
CDSLegs(recoveryRate,discountFactorArray(1:j),...
        origSurvivalProbArr(1:j),maturityArray(1:j));
    origSpreadArray(j) = protectionLeg/riskyAnnuity;
end

% Recalculate the spreads
recalculatedSpreadArray = zeros(1,length(maturityArray));
for i = 1:nMaturities

    index = find(maturityArray == refMaturityArray(i));
    maturityPosition(i) = index;
    currentMaturityArray =
maturityArray(1:maturityPosition(i));

    currentZeroDefaultRateArray =
zeroDefaultRateHandle(currentMaturityArray);

    survivalProbArray = exp(-
currentZeroDefaultRateArray.*currentMaturityArray);

    discountFactors =
discountFactorArray(1:length(currentMaturityArray));

    [riskyAnnuity,protectionLeg] =
CDSLegs(recoveryRate,discountFactors,...
        survivalProbArray,currentMaturityArray);

    recalculatedSpreadArray(i) = protectionLeg./riskyAnnuity;
end

% Plot and compare the spreads
figure(5);
plot(maturityArray,origSpreadArray,'b*-
',maturityArray,recalculatedSpreadArray,'ro-');
xlabel('Maturity(years)');
ylabel('CDS Spreads');
legend('Original spreads','Recalculated spreads');
title('Spread Curve Comparison');

%% Compare the bootstrapped zero default rates with the
estimated rates,
%% which are obtained from the original spreads and the
recovery rate.
```

```
estimatedZeroDefaultRateArr = origSpreadArray./(1-
recoveryRate);
figure(6);
plot(maturityArray,zeroDefaultRateArray,'bo-
',maturityArray,estimatedZeroDefaultRateArr,'rs-');
xlabel('Maturity');
ylabel('Zero Default Rate');
title('Compare the Survival Curve Again');
legend('Bootstrapped Curve','Estimated Curve');
```

zeroRateLogLinear.m

```
% This function is to calculate the zero rate of a CDS given a
set of
% yields and maturities.
% Date: 07/07/2010
function zeroRates = zeroRateLogLinear(maturities, refYields,
refMaturities)
%   Calculate the lengths of the yield array and the maturity
array.
    nYields = length(refYields);
    nMaturities = length(refMaturities);

%   The lengths of the yield array and the maturity array must
be equal.
    if (nYields ~= nMaturities)
        error('The number of yields must be equal to the
number of maturities');

%   Special case: the lengths of the arrays are 1.
    elseif (nYields == 1)
        zeroRates = refYields(1)*ones(length(maturities));

%   The general case.
    else
%        Interpolation, with extrapolation selected as an
option.
         refLogs = refYields.*refMaturities;
         logs =
interp1([0,refMaturities],[0,refLogs],maturities, 'linear',
'extrap');
         zeroRates = logs./maturities;
%       Extrapolation only before the shortest reference
maturity.
%         zeroRates(maturities < min(refMaturities)) =
refYields(1);
    end
end
```

AHEquityTrancheSurvivalCurve.m

```
% This function calculates the equity tranche survival curve
under the FHP model
% with stochastic recovery rate.
% Date: 20/08/2010

function survivalCurve =
AHEquityTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,limitPoint,nCredits,correlation,survivalProba
bilities,...
    nPeriods)

    survivalCurve = zeros(1,nPeriods);
    for t = 1:nPeriods
        integrandHandle =
@(marketFactor)integration(recoveryRate,recoveryMarkdown,...

limitPoint,nCredits,correlation,survivalProbabilities(t),marke
tFactor);

        survivalCurve(t) = trapezium(integrandHandle,6,-6,30);
    end
end

function integrand =
integration(recoveryRate,recoveryMarkdown,limitPoint,...
    nCredits,correlation,survivalProbabilities,marketFactor)

    beta = sqrt(correlation);
    biCoeff = zeros(1,nCredits+1);

    for k = 0:nCredits
        biCoeff(k+1) = nchoosek(nCredits,k);
    end

    C = norminv(1-survivalProbabilities,0,1);

    conditionalDefaultProb = normcdf((C-
beta*marketFactor)/sqrt(1-correlation));
    I = @(marketFactor)(normcdf((C-beta*marketFactor)/sqrt(1-
correlation)))*...
        normpdf(marketFactor);
    unconditionalDefaultProb = trapezium(I,6,-6,30);

    unconditionalDefaultProbZero = (1-
recoveryRate)*unconditionalDefaultProb/...
        (1-recoveryMarkdown);

    conditionalDefaultProbZero =
normcdf((norminv(unconditionalDefaultProbZero,0,1)-...
```

```
        beta*marketFactor)/sqrt(1-correlation));

    stochasticRecoveryRate = 1-(1-
recoveryMarkdown)*conditionalDefaultProbZero/...
        conditionalDefaultProb;

    outstandingNotional = (1-(0:nCredits).*(1-
stochasticRecoveryRate)/...

(nCredits*limitPoint)).*((0:nCredits)<nCredits*limitPoint/...
        (1-stochasticRecoveryRate));

    integrand =
normpdf(marketFactor,0,1)*sum(outstandingNotional.*...
        biCoeff.*(conditionalDefaultProb.^(0:nCredits)).*...
        ((1-conditionalDefaultProb).^(nCredits-
(0:nCredits))));
end
```

AHTrancheSpread.m

```
% This function calculates the tranche spread under the FHP-AH
model
% Date: 24/08/2010

function
[riskyAnnuities,protections,upfrontFees,trancheSpreads] = ...

AHTrancheSpread(recoveryRate,recoveryMarkdown,attachmentPoints
,...

detachmentPoints,discountFactors,survivalProbabilities,maturit
ies,...
    correlation,nCredits,nPeriods)

    riskyAnnuities = zeros(1,length(attachmentPoints));
    protections = zeros(1,length(attachmentPoints));
    trancheSpreads = zeros(1,length(attachmentPoints));
    upfrontFees = zeros(1,length(attachmentPoints));

    for i = 1:length(attachmentPoints)
        trancheSurvCurve =
AHTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,attachmentPoints(i),detachmentPoints(i),...

nCredits,correlation,survivalProbabilities,nPeriods);

        [riskyAnnuities(i),protections(i)] =
CDSLegs(0,discountFactors,...
            trancheSurvCurve,maturities);

        trancheSpreads(i) = protections(i)/riskyAnnuities(i);

        if attachmentPoints(i) == 0
            runningSpread = 0.05;
            upfrontFees(i) = (trancheSpreads(i)-
runningSpread)*riskyAnnuities(i);
            trancheSpreads(i) = runningSpread;
        else
            upfrontFees(i) = 0;
        end
    end
end
```

AHTrancheSurvivalCurve.m

```
% This function calculates the generic tranche survival curve
under the FHP model
% with stochastic recovery rate.
% Date: 20/08/2010

function survivalCurve =
AHTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,attachmentPoint,detachmentPoint,nCredits,corr
elation,...
    survivalProbabilities,nPeriods)

    if attachmentPoint == 0
        survivalCurve =
AHEquityTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,detachmentPoint,nCredits,correlation,...
            survivalProbabilities,nPeriods);
    else
        equityCurve1 =
AHEquityTrancheSurvivalCurve(recoveryRate,recoveryMarkdown,...

attachmentPoint,nCredits,correlation,survivalProbabilities,nPe
riods);
        equityCurve2 =
AHEquityTrancheSurvivalCurve(recoveryRate,recoveryMarkdown,...

detachmentPoint,nCredits,correlation,survivalProbabilities,nPe
riods);

        constant = detachmentPoint/(detachmentPoint-
attachmentPoint);

        survivalCurve = constant*equityCurve2+(1-
constant)*equityCurve1;
    end
end
```

baseCorrelationBootstrap.m

```
% Bootstrap the base correlation based on market spreads using
the LHP
% model.
% Date: 24/08/2010

function baseCorrelation =
baseCorrelationBootstrap(recoveryRate,...

discountFactors,maturities,attachmentPoints,detachmentPoints,.
..

survivalProbabilities,marketSpreads,pricingModel,nCredits,upfr
ontFee)

    baseCorrelation = zeros(1,length(attachmentPoints));

    baseCorrelation(1) =
equityTrancheImpliedCorrelation(recoveryRate,...

detachmentPoints(1),nCredits,upfrontFee,survivalProbabilities,
maturities,...
        pricingModel,discountFactors,marketSpreads(1));

    initValue = baseCorrelation(1);
    for i = 2:length(attachmentPoints)
        correlationAtK1 = baseCorrelation(i-1);
        presentValueHandle =
@(correlationAtK2)presentValue(recoveryRate,...

attachmentPoints(i),detachmentPoints(i),discountFactors,...

survivalProbabilities,maturities,marketSpreads(i),correlationA
tK1,...
            correlationAtK2,pricingModel,nCredits,upfrontFee);

        baseCorrelation(i) =
fzero(presentValueHandle,initValue);
        initValue = baseCorrelation(i);
    end
end

function PV =
presentValue(recoveryRate,attachmentPoint,detachmentPoint,...

discountFactors,survivalProbabilities,maturities,marketSpread,
...

correlationAtK1,correlationAtK2,pricingModel,nCredits,upfrontF
ee)
```

```
    PV =
trancheePresentValue(recoveryRate,attachmentPoint,detachmentPoi
nt,...

nCredits,upfrontFee,correlationAtK1,correlationAtK2,...

survivalProbabilities,maturities,pricingModel,discountFactors,
...
        marketSpread);
end
```

baseCorrelationBootstrapTest.m

```
%% Base Correlation Boostrap under All Models
% The base correlation is bootstrapped under the LHP, FHP, and
FHP-AH
% models.

%% Data
recoveryRate = 0.40;
discountFactors =
[0.9978,0.9852,0.9730,0.9607,0.9485,0.9365,0.9249,0.9132,...

0.9016,0.8903,0.8792,0.8680,0.8570,0.8462,0.8356,0.8250,0.8145
,0.8083,...
    0.7943,0.7842,0.7743];
survivalProbabilities =
[0.9993,0.9951,0.9909,0.9867,0.9825,0.9784,0.9743,...

0.9702,0.9661,0.9620,0.9580,0.9539,0.9499,0.9459,0.9419,0.9379
,0.9339,...
    0.9300,0.9261,0.9222,0.9183];
maturities =
[0.044,0.297,0.547,0.803,1.058,1.311,1.561,1.817,2.072,2.325,.
..

2.575,2.831,3.086,3.339,3.592,3.847,4.103,4.356,4.606,4.861,5.
117];
nCredits = 50;
attachmentPoints = [0,0.03,0.06,0.09,0.12];
detachmentPoints = [0.03,0.06,0.09,0.12,0.22];
% marketSpreads = [0.05,0.02595,0.0101,0.00385,0.00315];
marketSpreads = [0.05,0.0566,0.03075,0.0169,0.0027];
upfrontFee = 0.45;
warning off all

%% The LHP Base Correlation

pricingModel = 'LHP';
LHPBaseCorrelation = baseCorrelationBootstrap(recoveryRate,...

discountFactors,maturities,attachmentPoints,detachmentPoints,.
..

survivalProbabilities,marketSpreads,pricingModel,nCredits,upfr
ontFee);
disp('LHP Base Correlation:');disp(LHPBaseCorrelation);

%% The FHP Base Correlation

pricingModel = 'FHP';
FHPBaseCorrelation = baseCorrelationBootstrap(recoveryRate,...
```

```
discountFactors,maturities,attachmentPoints,detachmentPoints,.
..

survivalProbabilities,marketSpreads,pricingModel,nCredits,upfr
ontFee);
disp('FHP Base Correlation');disp(FHPBaseCorrelation);

%% The FHP-AH Base Correlation

pricingModel = 'FHP-AH';
AHBaseCorrelation = baseCorrelationBootstrap(recoveryRate,...

discountFactors,maturities,attachmentPoints,detachmentPoints,.
..

survivalProbabilities,marketSpreads,pricingModel,nCredits,upfr
ontFee);
disp('FHP-AH Base Correlation');disp(AHBaseCorrelation);

%% Plot the Curves
plot(detachmentPoints,LHPBaseCorrelation,':r.',detachmentPoint
s,...

FHPBaseCorrelation,':b.',detachmentPoints,AHBaseCorrelation,':
g.');
xlabel('Detachment Point');
ylabel('Base Correlation');
title('Base Correlation Bootstrap');
legend('LHP','FHP','FHP-AH','Location','NorthWest');
```

FHPDefaultTimeDistribution.m

```
 Model the the unconditional portfoliio loss distribution
under the FHP
% model. The result is the loss distribution across the time
periods.
% Date: 18/08/2010

function portfolioLoss =
FHPDefaultTimeDistribution(nCredits,nDefaults,correlation,...
    defaultProbabilities,nPeriods)

    beta = sqrt(correlation);
    portfolioLoss = zeros(1,nPeriods);
    format long
    biCoeff = nchoosek(nCredits,nDefaults);

    for t = 1:nPeriods
        C = norminv(defaultProbabilities(t),0,1);
        integrand = @(Z)normpdf(Z,0,1)*...
            (normcdf((C-beta*Z)/sqrt(1-
correlation))^nDefaults)*...
            ((1-normcdf((C-beta*Z)/sqrt(1-
correlation)))^(nCredits-nDefaults));

        portfolioLoss(t) = biCoeff*trapezium(integrand,6,-
6,30);
    end
end
```

FHPEquityTrancheSurvivalCurve.m

```
% This function calculates the tranche survival curve under
the FHP model.
% Date: 18/08/2010

function survivalCurve =
FHPEquityTrancheSurvivalCurve(recoveryRate,...

limitPoint,nCredits,correlation,survivalProbabilities,nPeriods
)

    survivalCurve = zeros(1,nPeriods);
    for t = 1:nPeriods
        integrandHandle =
@(marketFactor)integration(recoveryRate,limitPoint,...

nCredits,correlation,survivalProbabilities(t),marketFactor);

        survivalCurve(t) = trapezium(integrandHandle,5,-5,30);
    end
end

function integrand =
integration(recoveryRate,limitPoint,nCredits,...
    correlation,survivalProbabilities,marketFactor)

    beta = sqrt(correlation);
    maxNoDefaults = floor(nCredits*limitPoint/(1-
recoveryRate));

    biCoeff = zeros(1,maxNoDefaults+1);
    outstandingNotional = zeros(1,maxNoDefaults+1);

    for i = 0:maxNoDefaults
        biCoeff(i+1) = nchoosek(nCredits,i);
        outstandingNotional(i+1) = 1-i*(1-
recoveryRate)/(nCredits*limitPoint);
    end

    C = norminv(1-survivalProbabilities,0,1);
    conditionalDefaultProb = normcdf((C-
beta*marketFactor)/sqrt(1-correlation));
    integrand =
normpdf(marketFactor,0,1).*sum(biCoeff.*outstandingNotional.*.
..
        (conditionalDefaultProb.^(0:maxNoDefaults)).*...
        (1-conditionalDefaultProb).^(nCredits-
(0:maxNoDefaults)));
end
```

LHPEquityTrancheSurvivalCurve.m

```
% This function calculates the equity tranche survival curve.
% Date: 10/08/2010

function equitySurvProbabilities = ...
    LHPEquityTrancheSurvivalCurve(recoveryRate,limitPoint,...
    survivalProbabilities,correlation)

    beta = sqrt(correlation);

%   The threshold C(T)
    C = norminv(1-survivalProbabilities,0,1);

%   The market factor condition
    A = (1/beta)*(C-sqrt(1-correlation)*...
        norminv(limitPoint/(1-recoveryRate),0,1));
    mean = 0;
    bivariateNormal = bivariateCDF(C,-A,mean,-beta);

    equitySurvProbabilities = normcdf(-A)-(1-recoveryRate)*...
        bivariateNormal'/limitPoint;
end
```

TranchePresentValue.m

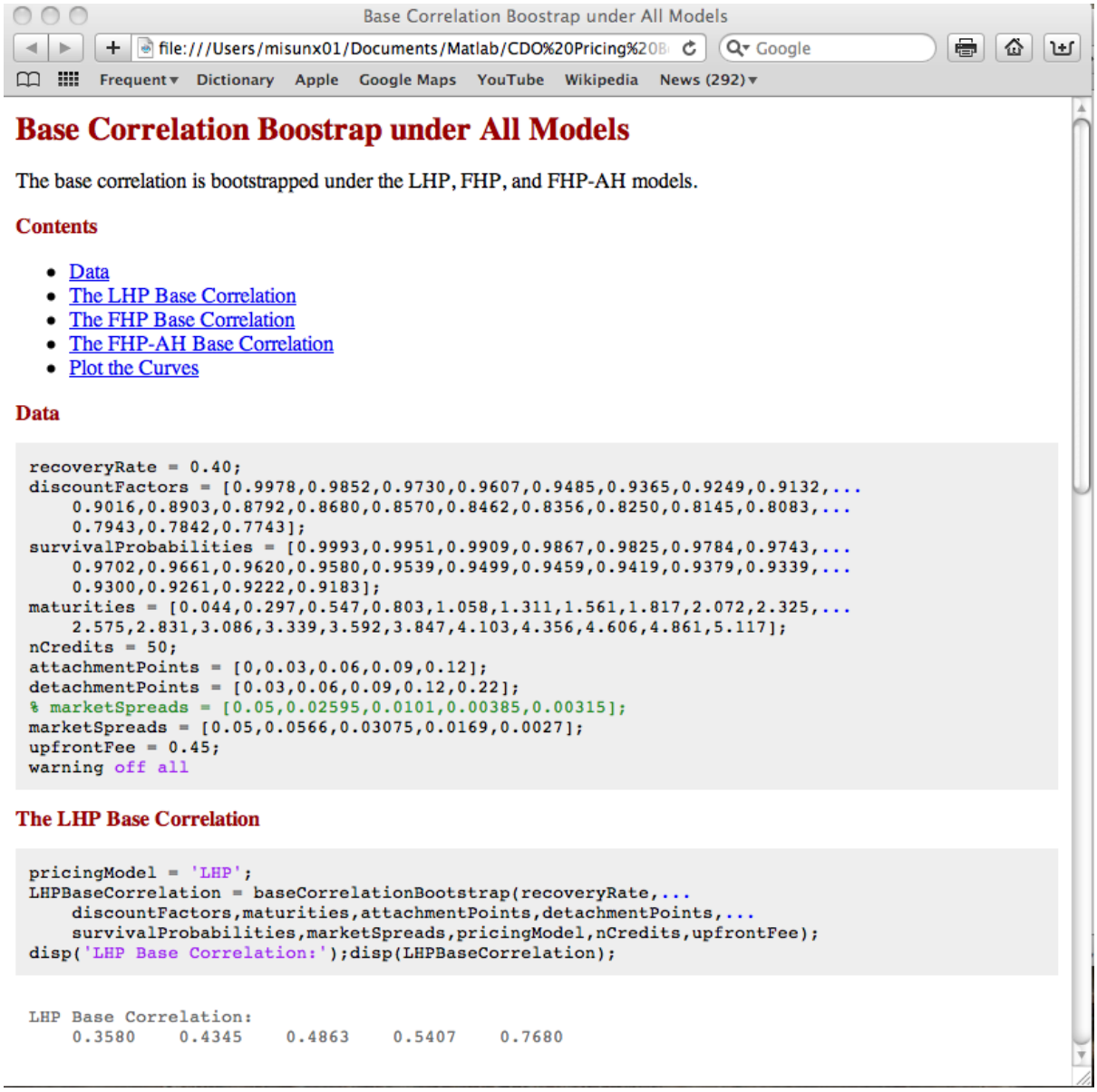
```
% This function returns the tranche PV
% Date: 25/08/2010

function tranchePV =
tranchePresentValue(recoveryRate,attachmentPoint,...

detachmentPoint,nCredits,upfrontFee,correlationAtK1,correlatio
nAtK2,...

survivalProbabilities,maturities,pricingModel,discountFactors,
...
    marketTrancheSpread)

    nPeriods = length(maturities);
    recoveryMarkdown = 0;
    constant = detachmentPoint/(detachmentPoint-
attachmentPoint);
    if attachmentPoint == 0
        switch pricingModel
            case 'LHP'
                survivalCurve =
LHPEquityTrancheSurvivalCurve(recoveryRate,...

detachmentPoint,survivalProbabilities,correlationAtK2);
            case 'FHP'
                survivalCurve =
FHPEquityTrancheSurvivalCurve(recoveryRate,...

detachmentPoint,nCredits,correlationAtK2,survivalProbabilities
,...
                    nPeriods);
            case 'FHP-AH'
                survivalCurve =
AHEquityTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,detachmentPoint,nCredits,correlationAtK2,...
                    survivalProbabilities,nPeriods);
            otherwise
                error('Wrong model name entered.');
        end
    else
        upfrontFee = 0;
        switch pricingModel
            case 'LHP'
                survivalCurve1 =
LHPEquityTrancheSurvivalCurve(recoveryRate,...

attachmentPoint,survivalProbabilities,correlationAtK1);
                survivalCurve2 =
LHPEquityTrancheSurvivalCurve(recoveryRate,...
```

```
detachmentPoint,survivalProbabilities,correlationAtK2);
                survivalCurve = constant*survivalCurve2+(1-
constant)*...
                    survivalCurve1;
            case 'FHP'
                survivalCurve1 =
FHPEquityTrancheSurvivalCurve(recoveryRate,...

attachmentPoint,nCredits,correlationAtK1,survivalProbabilities
,...
                    nPeriods);
                survivalCurve2 =
FHPEquityTrancheSurvivalCurve(recoveryRate,...

detachmentPoint,nCredits,correlationAtK2,survivalProbabilities
,...
                    nPeriods);
                survivalCurve = constant*survivalCurve2+(1-
constant)*...
                    survivalCurve1;
            case 'FHP-AH'
                survivalCurve1 =
AHEquityTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,attachmentPoint,nCredits,correlationAtK1,...
                    survivalProbabilities,nPeriods);
                survivalCurve2 =
AHEquityTrancheSurvivalCurve(recoveryRate,...

recoveryMarkdown,detachmentPoint,nCredits,correlationAtK2,...
                    survivalProbabilities,nPeriods);
                survivalCurve = constant*survivalCurve2+(1-
constant)*...
                    survivalCurve1;
            otherwise
                error('Wrong model name entered.');
        end
    end

    [riskyAnnuity,protection] = CDSLegs(0,discountFactors,...
            survivalCurve,maturities);

    tranchePV = upfrontFee+marketTrancheSpread*riskyAnnuity-
protection;
end
```

Appendix C Published M-Files

As mentioned in Chapter 6, the publish function in Matlab is very useful for demonstrating the code and its output. Therefore I have selected the most important test files and attached the published results here. The rest of the published results can be found in the html folder in the CD.

baseCorrelationBootstrapTest.m



Base Correlation Boostrap under All Models

The base correlation is bootstrapped under the LHP, FHP, and FHP-AH models.

Contents

Data

```
recoveryRate = 0.40;
discountFactors = [0.9978,0.9852,0.9730,0.9607,0.9485,0.9365,0.9249,0.9132,...
    0.9016,0.8903,0.8792,0.8680,0.8570,0.8462,0.8356,0.8250,0.8145,0.8083,...
    0.7943,0.7842,0.7743];
survivalProbabilities = [0.9993,0.9951,0.9909,0.9867,0.9825,0.9784,0.9743,...
    0.9702,0.9661,0.9620,0.9580,0.9539,0.9499,0.9459,0.9419,0.9379,0.9339,...
    0.9300,0.9261,0.9222,0.9183];
maturities = [0.044,0.297,0.547,0.803,1.058,1.311,1.561,1.817,2.072,2.325,...
    2.575,2.831,3.086,3.339,3.592,3.847,4.103,4.356,4.606,4.861,5.117];
nCredits = 50;
attachmentPoints = [0,0.03,0.06,0.09,0.12];
detachmentPoints = [0.03,0.06,0.09,0.12,0.22];
% marketSpreads = [0.05,0.02595,0.0101,0.00385,0.00315];
marketSpreads = [0.05,0.0566,0.03075,0.0169,0.0027];
upfrontFee = 0.45;
warning off all
```

The LHP Base Correlation

```
pricingModel = 'LHP';
LHPBaseCorrelation = baseCorrelationBootstrap(recoveryRate,...
    discountFactors,maturities,attachmentPoints,detachmentPoints,...
    survivalProbabilities,marketSpreads,pricingModel,nCredits,upfrontFee);
disp('LHP Base Correlation:');disp(LHPBaseCorrelation);
```

```
LHP Base Correlation:
    0.3580    0.4345    0.4863    0.5407    0.7680
```

The FHP Base Correlation

```
pricingModel = 'FHP';
FHPBaseCorrelation = baseCorrelationBootstrap(recoveryRate,...
    discountFactors,maturities,attachmentPoints,detachmentPoints,...
    survivalProbabilities,marketSpreads,pricingModel,nCredits,upfrontFee);
disp('FHP Base Correlation');disp(FHPBaseCorrelation);
```

```
FHP Base Correlation
    0.2936    0.4008    0.4607    0.5216    0.7587
```

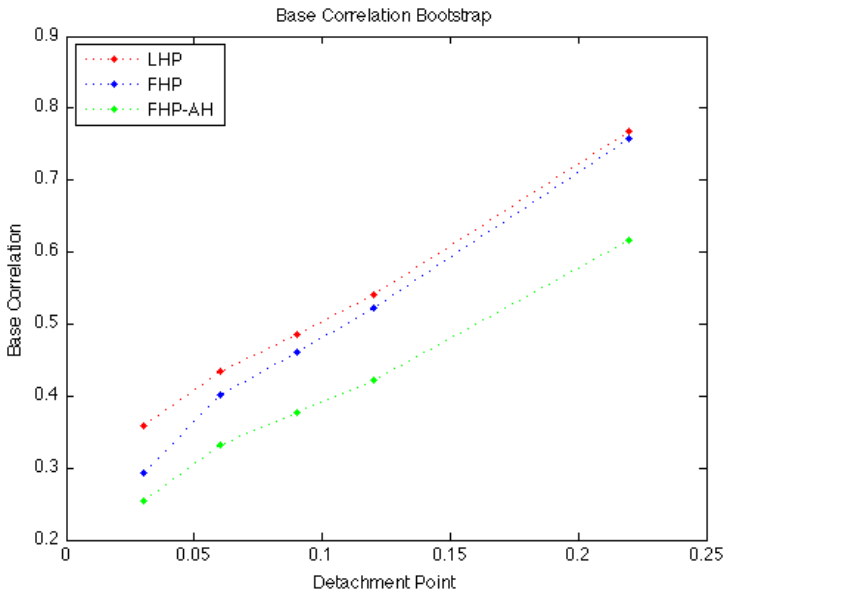
The FHP-AH Base Correlation

```
pricingModel = 'FHP-AH';
AHBaseCorrelation = baseCorrelationBootstrap(recoveryRate,...
    discountFactors,maturities,attachmentPoints,detachmentPoints,...
    survivalProbabilities,marketSpreads,pricingModel,nCredits,upfrontFee);
disp('FHP-AH Base Correlation');disp(AHBaseCorrelation);
```

```
FHP-AH Base Correlation
    0.2540    0.3325    0.3778    0.4226    0.6175
```

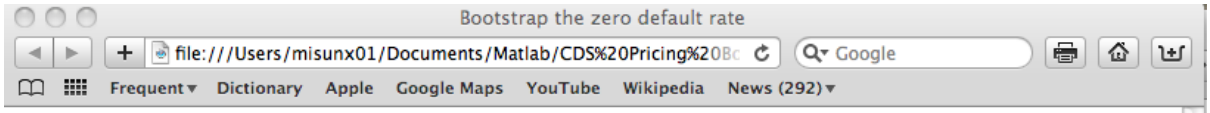
Plot the Curves

```
plot(detachmentPoints,LHPBaseCorrelation,':r.',detachmentPoints,...
    FHPBaseCorrelation,':b.',detachmentPoints,AHBaseCorrelation,':g.');
xlabel('Detachment Point');
ylabel('Base Correlation');
title('Base Correlation Bootstrap');
legend('LHP','FHP','FHP-AH','Location','NorthWest');
```



Published with MATLAB® 7.10

zeroDefaultRateBootstrapTest.m



Bootstrap the zero default rate

This file tests the bootstrapzerodefaultrate function. Using the bootstrapped zero default curve, I can derive the survival probability curve and the forward default curve.

Some comparasons have also been done. One is for the spreads, and another is for the zero default rates.

Contents

Values of the parameters

```
recoveryRate = 0.30;
% refSpreadArray = [0.0988,0.0998,0.0896,0.0794,0.0594];

refSpreadArray = [0.1008,0.0988,0.0995,0.0998,0.0972,0.0945,0.0915,0.0896,...
    0.0863,0.0837,0.0813,0.0794,0.0763,0.0737,0.0714,0.0695,0.0665,0.0639,...
    0.0615,0.0594];

discountFactorArray = [0.9941,0.9884,0.9825,0.9763,0.9698,0.9628,0.9555,...
    0.9478,0.9399,0.9317,0.9231,0.9144,0.9055,0.8964,0.8872,0.8779,0.8685,...
    0.8591,0.8496,0.8400];

maturityArray = (1:20)*365.25/360/4;
% refMaturityArray = maturityArray([2,4,8,12,20]);
refMaturityArray = maturityArray(1:20);
```

Bootstrap the zero default rates.

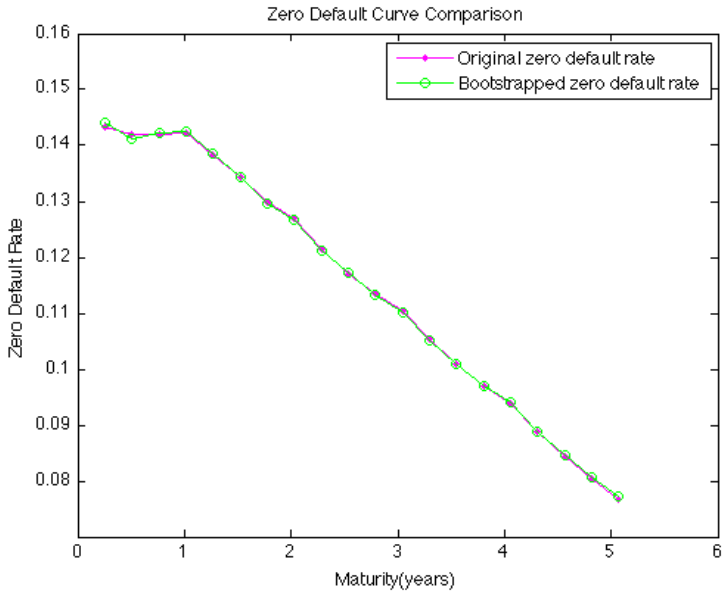
The output of this function is a function handle.

```
zeroDefaultRateHandle = zeroDefaultRateBootstrap(recoveryRate,...
    discountFactorArray, maturityArray, refMaturityArray, refSpreadArray);
zeroDefaultRateArray = zeroDefaultRateHandle(maturityArray);
```

Plot and compare the boostrapped zero default rate

The aim is to compare the bootstrapped rates to the original rates obtained from the original survival probability.

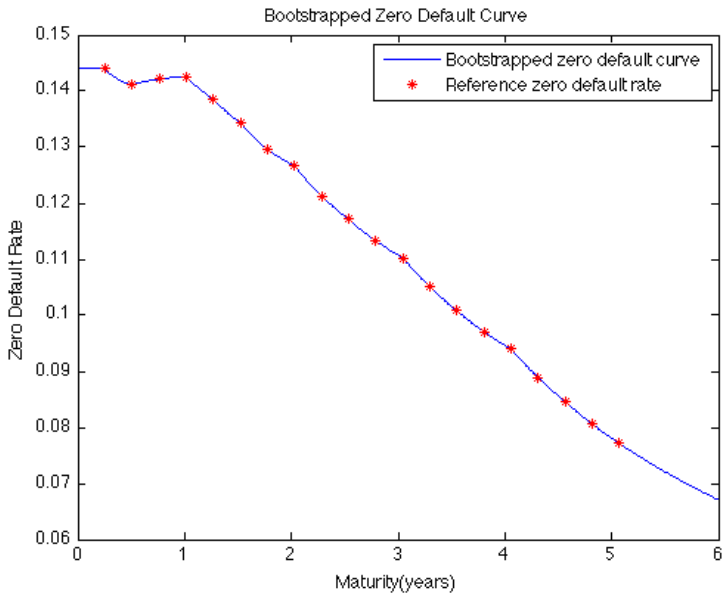
```
origSurvivalProbArr = [0.9643,0.9305,0.8976,0.8656,0.8391,0.8151,0.7941,...
    0.7730,0.7579,0.7432,0.7287,0.7145,0.7066,0.6990,0.6914,0.6837,0.6822,...
    0.6806,0.6791,0.6776];
origZeroDefaultRateArr = -log(origSurvivalProbArr)./maturityArray;
figure(1);
plot(maturityArray,origZeroDefaultRateArr,'m.-',maturityArray,zeroDefaultRateArray,'go-');
xlabel('Maturity(years)');
ylabel('Zero Default Rate');
legend('Original zero default rate','Bootstrapped zero default rate');
title('Zero Default Curve Comparison');
```

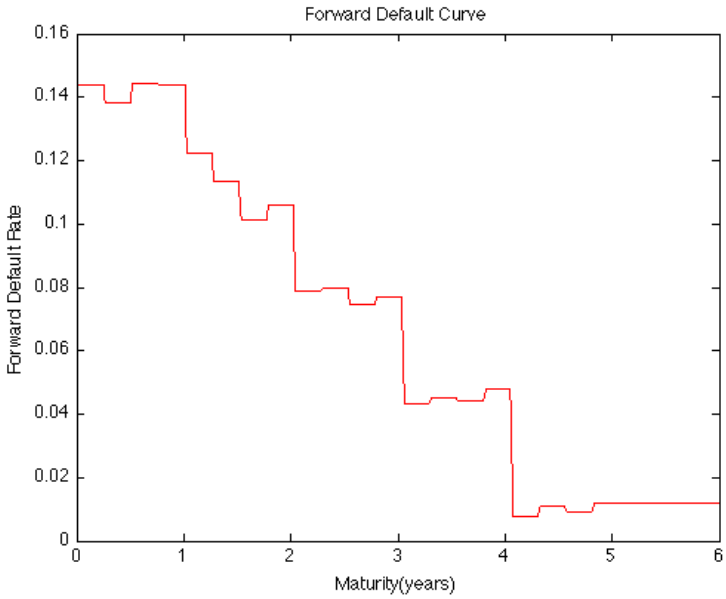


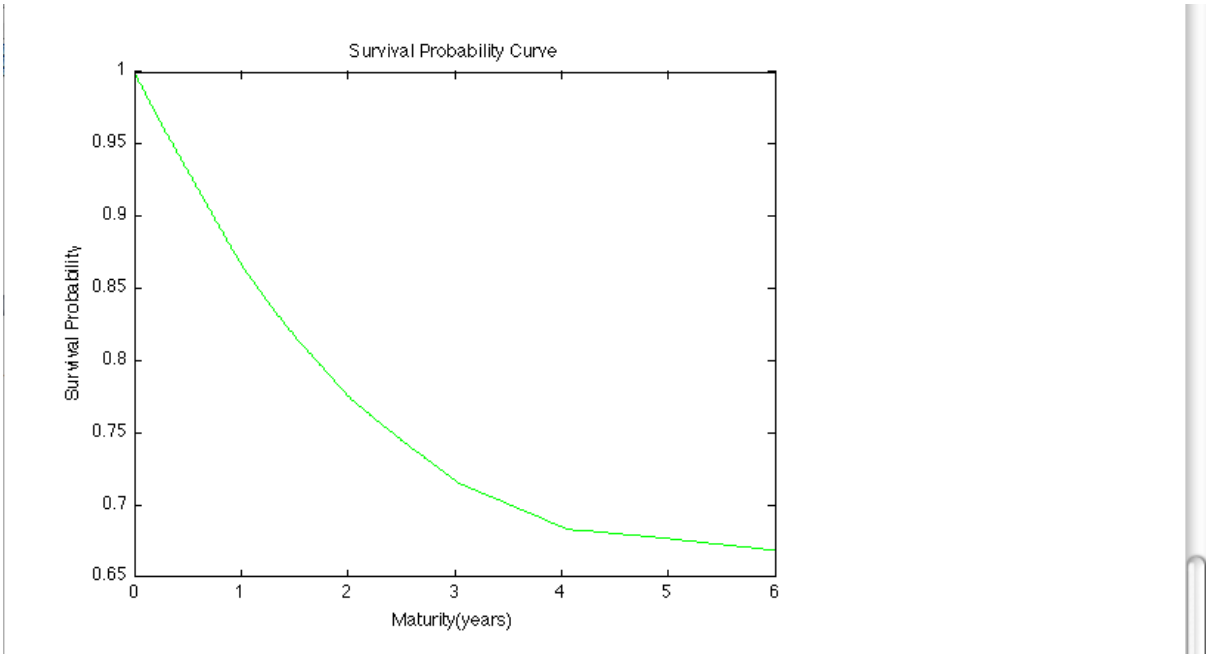
Plotting the curves.

The plot function do not accept function handles.

```
maturities = 0:0.01:6;
%   outputcurves(zeroDefaultRateHandle,maturities);
%   Output zero default rate from the handle
zeroDefaultRateArr = zeroDefaultRateHandle(maturities);
refZeroDefaultRateArr = zeroDefaultRateHandle(refMaturityArray);
figure(2);
plot(maturities,zeroDefaultRateArr,'b-',refMaturityArray,refZeroDefaultRateArr,'r*');
xlabel('Maturity(years)');
ylabel('Zero Default Rate');
legend('Bootstrapped zero default curve','Reference zero default rate');
title('Bootstrapped Zero Default Curve');

%   Forward default rate can be obtained from the zero default rates.
fwdDefaultRateArr = forwardDefaultRates(zeroDefaultRateArr,maturities);
figure(3);
plot(maturities(2:end), fwdDefaultRateArr, 'r-');
% stairs(maturities(2:end), fwdDefaultRateArr, 'r-');
xlabel('Maturity(years)');
ylabel('Forward Default Rate');
title('Forward Default Curve');

%   Survival probabilities can be obtained from zero default rates as well.
survivalProbArr = exp(-zeroDefaultRateArr.*maturities);
figure(4);
plot(maturities, survivalProbArr, 'g-');
xlabel('Maturity(years)');
ylabel('Survival Probability');
title('Survival Probability Curve');
```

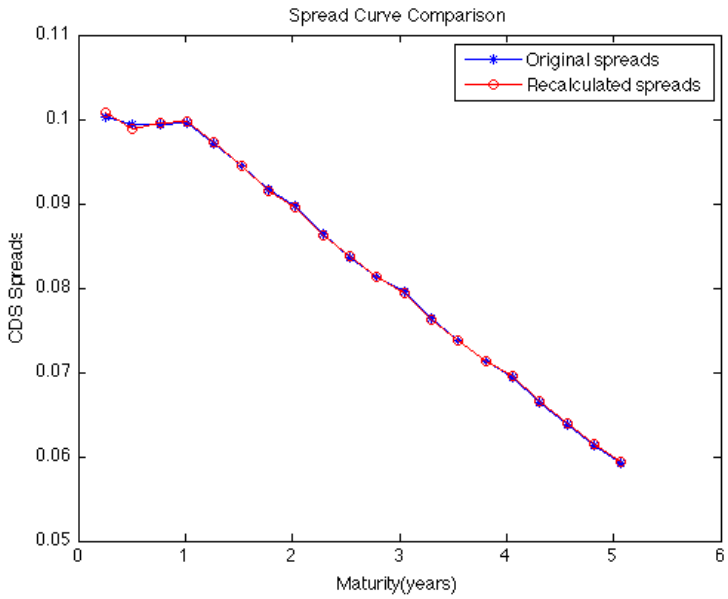

Bootstrapped Zero Default Curve
Bootstrapped zero default curve
Reference zero default rate
Zero Default Rate
Maturity(years)
0.15
0.14
0.13
0.12
0.11
0.1
0.09
0.08
0.07
0.06
0
1
2
3
4
5
6


Forward Default Curve
Forward Default Rate
Maturity(years)
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
1
2
3
4
5
6



Recalculate the CDS spreads to check the results.

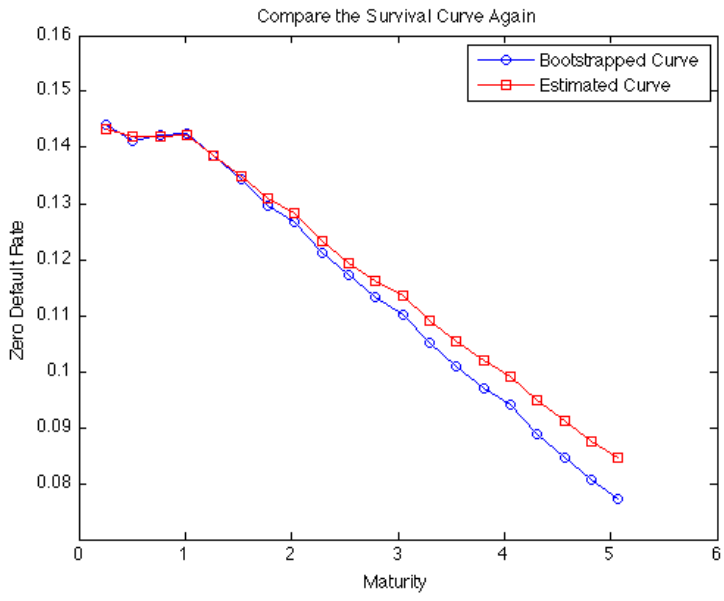
```
nMaturities = length(refMaturityArray);
maturityPosition = zeros(1,nMaturities);
recalculatedSpreadArray = zeros(1,nMaturities);

% Solve the direct problem of finding the CDS spreads using the JP Morgan
% given discount factors and survival probabilities.
origSpreadArray = zeros(1,length(maturityArray));
for j = 1:length(maturityArray)
    [riskyAnnuity,protectionLeg] = CDSLegs(recoveryRate,discountFactorArray(1:j),...
        origSurvivalProbArr(1:j),maturityArray(1:j));
    origSpreadArray(j) = protectionLeg/riskyAnnuity;
end

% Recalculate the spreads
recalculatedSpreadArray = zeros(1,length(maturityArray));
for i = 1:nMaturities

    index = find(maturityArray == refMaturityArray(i));
    maturityPosition(i) = index;
    currentMaturityArray = maturityArray(1:maturityPosition(i));

    currentZeroDefaultRateArray = zeroDefaultRateHandle(currentMaturityArray);

    survivalProbArray = exp(-currentZeroDefaultRateArray.*currentMaturityArray);

    discountFactors = discountFactorArray(1:length(currentMaturityArray));

    [riskyAnnuity,protectionLeg] = CDSLegs(recoveryRate,discountFactors,...
        survivalProbArray,currentMaturityArray);

    recalculatedSpreadArray(i) = protectionLeg./riskyAnnuity;
end

% Plot and compare the spreads
figure(5);
plot(maturityArray,origSpreadArray,'b*-',maturityArray,recalculatedSpreadArray,'ro-');
xlabel('Maturity(years)');
ylabel('CDS Spreads');
legend('Original spreads','Recalculated spreads');
title('Spread Curve Comparison');
```



Compare the bootstrapped zero default rates with the estimated rates,

which are obtained from the original spreads and the recovery rate.

```
estimatedZeroDefaultRateArr = origSpreadArray./(1-recoveryRate);
figure(6);
plot(maturityArray,zeroDefaultRateArray,'bo-',maturityArray,estimatedZeroDefaultRateArr,'rs-');
xlabel('Maturity');
ylabel('Zero Default Rate');
title('Compare the Survival Curve Again');
legend('Bootstrapped Curve','Estimated Curve');
```



Published with MATLAB® 7.10

Bibliography

Amraoui, S. and Hitier, S (2008). Optimal Stochastic Recovery for Base Correlation, Working paper.

Amraoui, S., Cousot, L, Hitier, S. and Laurent, J.-P. (2009). Pricing CDOs with State Dependent Stochastic Recovery Rates, Working paper.

Andersen, L. and Sidenius, J. (2004). Extensions to the Gaussian Copula: Random Recovery and Random Factor Loadings, *Journal of Credit Risk*, 1(1), Winter 2004/05

Beinstein, E. and Scott, A. (2006). Credit Derivatives Handbook, JP Morgan Corporate Quantitative Research.

Burtschell, X., Gregory, J. and Laurent, J.-P. (2008). A Comparative Analysis of CDO Pricing Models, Working paper.

Choros, B., Härdle, W. and Okhrin, O. (2009). CDO Pricing with Copulae, *SFB 649 Discussion Paper 2009-013*, Humboldt-Univerisität zu Berlin.

Ech-Chatbi, C. (2008). CDS and CDO Pricing with Stochastic Recovery, Working paper.

Elizalde, A. (2005). Credit Risk Models IV: Understanding and Pricing CDOs, Working paper.

Hagan, P.S. (2005). Credit Derivatives, Working paper.

Hagan, P.S. and West, G. (2005). Interpolation Methods for Curve Construction, *Applied Mathematical Finance*, 13(2): 89-129, June 2006.

Hager, S. and Schöbel, R. (2005). A Note on the Correlation Smile, Working paper.

Hull, J. (2008). *Options, Futures, and Other Derivatives*, Prentice Hall.

JP Morgan, (2001). Par Credit Default Swap Spread Approximation from Default Probabilities, Report.

Kakodkar, A., Galiani, S. and Shchetkovskiy M. (2004). Base Correlations – Overcoming the Limitations of Tranche Implied Correlations, Working paper, Merrill Lynch Global Securities Research & Economic Group.

Kakodkar, A., Martin, B. and Galiani, S. (2003). Correlation Trading, Research report, Merrill Lynch Global Securities Research & Economics Group.

Kakodkar, A. et al (2009). Coping with the Copula, Research report, Merrill Lynch Credit Derivatives Strategy Global.

O'Kane, D. (2001). Credit Derivatives Explained – Markets, Products, and Regulations, Lehman Brothers Structured Credit Research.

O'Kane, D. (2008). *Modelling Single-name and Multi-name Credit Derivatives*, John Wiley & Sons.

O'Kane, D. and Livesey, M. (2004). Base Correlation Explained, *Lehman Brothers Quantitative Credit Research Quarterly*, 2004 Q3/4.

Picone, D., Shah, P., Stoeckle, M. and Loddo, A. (2008). CDO Models: Opening the Black Box Part I – Large Homogeneous Pool Model, Working paper, Dresdner Kleinwort Structured Credit Research.

Picone, D., Shah, P., Stoeckle, M. and Loddo, A. (2008). CDO Models: Opening the Black Box Part II – Large Homogeneous Pool Model, Working paper, Dresdner Kleinwort Structured Credit Research.

The MathWorks, (2009). The MathWorks in Financial Services, http://www.mathworks.com/financial-services/

Wikipedia, (2010). Matlab, http://en.wikipedia.org/wiki/MATLAB