Genpact Smart Decision Services

Risk-Neutral PDs and Default-Adjusted Cash Flows

By: Sadanand Tutakne

Table of Contents

Abstract

This paper attempts to start-off with some well-known methods of deriving market-implied PDs (or, risk-neutral PDs as they are called in the academic literature) and derive credit-adjusted cash flows using the path of these market-implied PDs. The adjusted cash flows can be used for valuation, as an alternative to using credit-adjusted (risky) discount curves.

Key Words

Market-implied default probabilities; risk-neutral probability; implied probability of default, credit spread; discount rates; credit-adjusted discount rates; pricing; re-pricing; valuation; asset pricing;

Paper Type

Technical; R application;

Acknowledgements

I would like to thank our customers, Sid Reddy, Kunal Sengupta, Kaustav Mukherjee, Somnath Mukherjee, Kaushik Das and Sushant Vimal for helpful discussions on the broad area of yield curve modeling and Treasury Analytics in general, of which credit spreads models and implied PDs are sub-parts.

Introduction

This paper attempts to start-off with some well-known methods of deriving market-implied PDs (or, risk-neutral PDs as they are called in the academic literature) and derives credit-adjusted cash flows using the path of these market-implied PDs. The adjusted cash flows can be used for valuation, as an alternative to using credit-adjusted (risky) discount curves. For income simulations, the survival paths (or, conversely, the PD paths) from default-prediction models might be more appropriate, because market-implied probabilities tend to exceed the "statistical" or "actual" PDs by a large margin. This excess is typically attributed to the fact that agents in the market are typically risk-averse and therefore demand a premium larger than the expected loss itself. Yet other explanations of the excess include the presence of other risks like liquidity risk, which also call for premiums over and above the expected loss amounts. As a result, market-implied PDs are typically well above the actual PDs and are useful for valuation, where the focus is on finding values consistent with typical market prices. Where expected cash flows and incomes need to be estimated, the statistical (actual) PDs might be more useful.

A study of different approaches to generating implied PDs is therefore called for. Such a study would help us understand the differences between these approaches in detail, and would pave the way for better decisions in different contexts. Some models might be considered to be "too simplistic", while other models might involve complexities (say, PDs which exceed the 0-1 interval) which might need more careful understanding and interpretation. Studying these details can help us develop the capability to handle these decisions more rationally. This paper attempts to start-off with such a study.

Alternative Approaches to Estimating Market-Implied PDs

The literature on estimating implied PDs is not small any longer and contains applications of the methods to both bond and CDS data. Some famous approaches mentioned frequently in this context are (a) Jarrow and Turnbull (1995), (b) Duffie and Singleton (1999) and (c) Hull and White (2000). In addition to the papers mentioned above, the contributions by Berd, et. al. (2003) from Lehmann Brothers and J.P. Morgan (1999) are also often quoted. MATLAB includes a command called "cdsbootstrap", which is similar to Hull and White (2000) in the sense that it is also a bootstrap procedure using piecewise constant survival intensities. To be specific, it uses an exponential survival function, which makes the equations in the bootstrap nonlinear, instead of the simpler assumption of linear piecewise PDs used in, say, Loffler and Posch (2007). All the above procedures can be applied to a sample of instruments from one single rating class (as long as the risk-free discount curve is available) and to that extent, they can be considered to be different from the transition matrix based approaches, which are typically applied to data on more than one rating category. If the elements of the transition matrix are calibrated to market price data (so that implied bond prices match actual market prices), then the transition matrix approach gives us yet another estimate of market-implied PDs. Using the generator of the matrix, the entire time-path of market-implied PDs can be derived, as done in Arvanitis et. al. (1999).

Since this is a beginning paper, we do not attempt to work with all the above methods here. Here, our objective is to make some headway on a few of these alternative methodologies, and check how the cash-flow paths look like when adjusted by the implied PDs. Specifically, we look closely at

(a) the CDS-bootstrap, which uses the structure of CDS contracts and an exponential survival function with piecewise constant survival intensities to derive risk-neutral PDs from available credit spreads,

(b) a simpler variant of the Lehmann Brothers (2003) methodology, in which recoveries remain a constant fraction of the claimed amounts, but the recoveries are assumed to happen at the usual coupon and maturity dates, not immediately upon default,

(c) the Lehmann Brothers (2003) methodology, in which recoveries are assumed to be simply a fraction of the face value of the bond, but, importantly, they are assumed to happen at the time of default itself, and

(c) two famous methods to derive implied PDs from credit spreads, i.e., the RT-type recoveries assumption which follows Jarrow and Turnbull (1995) and the RM-type recoveries assumption which follows Duffie and Singleton (1999). The main difference between this method and the previous approaches is that here we use market data to estimate credit spreads and then derive the implied PDs from the spreads, whereas in the two earlier approaches, market data is used to estimate implied PDs and the spreads can then be derived (if needed) from cumulative PDs and LGDs.

According to some authors, the main difference between Jarrow and Turnbull (1995) and Duffie and Singleton (1999) is the difference in the assumption on recoveries. While the former assumes that recoveries happen at the maturity dates of the bonds, the latter assumes that recoveries happen at the time of default itself. The Lehmann approach uses the latter assumption, with an exponential spline formulation of the survival function. In our simpler version of the approach, we use the Lehmann exponential spline formulation for the survival function, but ease the assumption on recoveries by assuming that recoveries happen at the usual coupon dates and that a constant fraction "RR" of each cash flow is received at the original agreed date in case of default. The constant recovery rate assumption is a common one in this literature, and it is used in the Lehmann paper as well. In the second approach, we follow Lehmann more closely, except that recoveries are assumed to be a fraction of the face value, and we ignore any accrued interest.

Data and Estimation Methodology

For the CDS bootstrap, CDS spreads for years 1-10 were taken for one or two selected banks from the S&P Capital IQ tool. Going by usual market norms, we assumed that the CDS premiums would be paid twice a year (semi-annually) and that any possible protection payments would be made at the end of a year. This latter assumption is only a simplifying assumption, but one which we find in some free internet sources too.

For the other exercises, data for over 500 US semi-annual corporate coupon bonds (non-callable) from one particular rating category was taken from Bloomberg. The cash flow paths (for the no-default scenario) were derived using a program written in R. Although the ready-made functions in MATLAB for cash-flow generation are somewhat better (in that they also account for holidays and day-count conventions), we have seen in other yield curve modeling exercises that the results we get using our simpler R code are close to the results we get when we generate cash flows using the MATLAB functions. In order to keep an eye on out-of-sample performance issues, we split the data 60-40 into development and validation, and report results for the validation data too.

1. The "CDS Bootstrap" with an Exponential Hazard Function:

As mentioned in the previous section, MATLAB includes a command (called "cdsbootstrap") to derive risk-neutral PDs from CDS spreads. Similar to textbook descriptions of bootstrapping, it is based on sequentially deriving the risk-neutral PDs for higher maturities based on those for lower maturities, and available details suggest that it uses the "usual" exponential hazard function. The equations used within the bootstrap are nonlinear, but otherwise easy to code in other available other software packages like R (which is the tool we have used for the exercises below). Based on some applied exercises we attempted, it seems the problem is typically easy to solve using available nonlinear solvers.

There are several free internet papers which describe the bootstrap procedure based on the details of the CDS contract. An excellent applied tool is the Excel tool called (CreditCurve_Bootstrapping) found on the following website: http://www.quantcode.com/modules/mydownloads/viewallquicklist.php.

The algorithm is based on assuming a functional form for the survival function (which gives the cumulative probability of survival of the customer up to time-point "t") and equating the present value of (certain) premiums paid to the present value of protection received in the (uncertain) event of default. Mathematically,

$$PV(Protection) = (1-R)\sum_{t=1}^{N} d(t)\,[P(t) - P(t-1)],\ and$$

$$PV(Premiums\ paid) = \sum_{t=1}^{M} spread(t)\,.\,d(t)\,.\,[1-P(t)],\ where$$

$d(t)$ refers to the discount factor for time point 't', $P(t)$ to the cumulative probability of default up to time 't', and R to the recovery rate, assumed constant and known at the time of the contract.

If premiums are paid quarterly but the protection is received only at yearly intervals (depending upon whether a default has occurred or not), the cumulative PD function (or, equivalently, cumulative survival probability function) for the premium leg would need to be adjusted for the timing difference in the cash flows. Assuming an exponential hazard function, the cumulative survival functions look like as follows.

$$S(\tau) \equiv Cumulative\ Survival\ Probability\ for\ Premium\ Leg$$

$$= \begin{cases} \exp(-\lambda_{01}\tau),\ for\ year\ 1, \\ exp(-\lambda_{01} - \lambda_{12}\tau),\ for\ year\ 2, \\ ..... \\ \exp(-\lambda_{01} - \lambda_{12} - .. - \lambda_{k-1,k}\tau),\ for\ year\ "k" \end{cases},$$

where τ takes fractional values like $0.25, 0.75, 1.5, 1.75$, etc., for the concerned quarters of a year.

However, since the protection is paid only at the end of the year, the survival function for the protection leg would look like:

$$S(t) \equiv Cumulative\ Survival\ Probability\ for\ Protection\ Leg$$

$$= \begin{cases} exp(-\lambda_{01}),\ for\ year = 1, \\ \exp(-\lambda_{01} - \lambda_{12}),\ for\ year = 2, \\ ..... \\ \exp(-\lambda_{01} - \lambda_{12} - .. - \lambda_{k-1,k}),\ for\ year = "k" \end{cases}$$

The final step in the derivation of the bootstrapped probabilities is equating and solving for the survival / default probabilities for each year sequentially. For example, for year=1, the PV of premiums paid would be a sum of the present values of the premiums paid at the end of each of the 4 quarters. The PV of protection paid for a year would, however, contain only one term – the PV of the possible (uncertain) payment which might need to be made at the end of year 1 (because we assume that protection payments happen only at the end of the year). Solving this nonlinear equation for λ_{01}, we then proceed to obtain the value of the second parameter (λ_{12}) from the second year's data (i.e., payments made and protection promised for the second year). Proceeding sequentially this way, the entire cumulative survival probability curve is derived.

2. The Simplified Lehmann Brothers (2003) Approach:

Assuming Q(t) to be the cumulative survival function, the model in Berd, et. al. (2003) assumes a two state world in which if the customer survives (does not default), then the lender (bondholder) gets the pre-specified cash flow (say, coupon) at any given date. However, if the customer defaults, then a constant fraction RR of the claims are received by the bondholder, at the default date itself, with no further cash flows.

We simplify this approach to an approach where the cash flow dates remain the pre-specified dates (coupon and maturity dates), even in the face of a default. As mentioned above, this seems to be an approach similar to Jarrow and Turnbull (1995) and is therefore a common one in this literature. The expected cash flows for any given date are therefore very easy to derive. Regardless of when default happens, if the borrower has defaulted by time "t", the bondholder gets RR-times the contractual cash flow at time "t", and if the borrower does not default by time "t", the bondholder receives the full contractual cash flow. The expected cash flow at any given date is therefore [Q(t) * CF(t) + (1-Q(t)) * RR * CF(t)], and these expected flows can be added up to arrive at the expected value of the cash flows at the initial point of time. Discounting is done using a risk-free zero coupon curve, which is estimated prior to starting on this part. CF denotes cash flows, RR is a scalar between 0 and 1 denoting the constant recovery rate and (1-Q(t)) is the cumulative probability of default up to the time "t".

The regression equation therefore turns out to be:

Bond Price = Disc. CF Matrix * Survival Function + RR * Disc. CF Matrix * (1-Survival Function) + error,

Nx1 NxT Tx1 1x1 NxT Tx1 Nx1

where:

Survival Function = Exponential Splines * Spline Coefficients

Tx1 Tx3, say 3x1

In the above, T is the cumulative number of cash flow dates for the regression as a whole, N is the number of bonds used for the exercise and the cash flow matrix is discounted using the risk free curve before entering it into the regression. The spline coefficients are then estimated using OLS. Since the splines themselves are of the form exp(-k * alpha * t), where k=1,2,3, so an alpha needs to be chosen before the spline matrix (Nx3) can be created for OLS estimation. Once the regression is run for any given alpha, we optimize over alpha using a simple grid search over some relevant range of values.

For the optimized value of alpha = 0.03, the results were as given in the "Results" section of the paper.

3. The Lehmann Brothers (2003) Approach:

The main difference here is in the assumption on the timing of recovery cash flows. The Lehmann paper assumes that a constant fraction (say, RR) of the claims is paid to the lender at the time of default itself. As for the claim amount, as mentioned earlier, according to J.P. Morgan, we come closest to actual legal practices if we assume that the claim is equal to the principal plus accrued interest at the time of default. However, in the literature, we find other practices too, for example, assuming that the claim is equal to the price of the bond just prior to default, assuming that the claim is equal to the face value of the bond, etc. Part of the reason for the diversity seems to be the fact that some formulations readily lead to closed form solutions for the required functions (say, spreads or the survival curve). However, it also seems to be because different assumptions have led to similar results in some studies. Here, we assume that the claim is simply the face value of the bond. The regression equation therefore changes to:

Bond Price = Disc CF Matrix * Prob Survival (t) + Disc Recovery Matrix * [Prob Survival (t-1) – Prob Survival (t)] + error,

Nx1 NxT Tx1 NxT Tx1 Nx1

where:

Prob Survival (t) = Exponential Splines * Spline Coefficients

Tx1 Tx3, say 3x1

4. Using Spreads from Risky Zero-Coupon Curves:

Using any standard methodology, a risky zero-coupon curve can be estimated much the same way as a risk-free zero coupon curve can, using data from, say, Bloomberg. In what follows, the risky zero-coupon curve was estimated using an exponential spline model for the discount function. Once the two zero-coupon curves are available, credit spreads can be estimated by taking the distance between them for different maturities. The crucial next step is to then convert these spreads into implied PDs, and to do this, we again rely on different recovery assumptions. According to Tobe (2012), two most popular recovery assumptions lead to simple formulae which can be used to derive implied default probabilities from credit spreads or vice-versa. Of course, as Tobe elaborates, we can further improve upon these estimates by explicitly taking into account the errors in the estimation of the spreads themselves. In this paper, we leave that additional refinement for a future exercise, and simply rely on the theoretical relationships to convert spreads to implied PDs. The two recovery assumptions are:

(a) RM type recovery, used by Duffie and Singleton (1999), which gives us the formula:

$$G(t) = \exp(-t.s(t)^{\frac{1}{1-\delta}}) = \left[\frac{P(0,t)}{P^*(0,t)}\right]^{\frac{1}{1-\delta}},$$

where P(0,t) refers to the price of a risky and P*(0,t) refers to the price of a risk-free zero-coupon bond with maturity “t”, s(t) refers to the credit spread applicable to maturity “t”, G(t) is the survival function and delta refers to the constant recovery rate.

(b) RT type recovery, used by Jarrow and Turnbull (1995), which gives us the formula:

$$G(t) = \frac{1}{1-\delta}\exp(-t.s(t) - \delta) = \frac{1}{1-\delta}\left[\frac{P(0,t)}{P^*(0,t)} - \delta\right],$$

where the notation is the same as for the previous case.

Results: Spreads and Adjusted Cash Flows

The main outputs we need from spreads models are (a) the spreads themselves, (b) the implied PDs, given the assumption on LGDs and (c) the resulting credit-adjusted cash flows, which provide us with an alternative way to value risky assets. In evaluating one method over another, the reasonableness of these results would be one factor, but a more basic factor would be the fit of the models in in-sample and out-of-sample data. Therefore, all these four factors are looked at in the results that follow.

A slight modification of the above statement is due when discussing the “CDS bootstrap” approach. There, spreads are already available, and we only need to derive the CDS implied probabilities of default. For that method, we

did not try to derive default-adjusted cash flows because the data for the spreads was taken from a completely different source and was not related to the other bond data we downloaded from Bloomberg. Therefore, for that methodology, only the R code and the results are added below.

1. Results from the "CDS Bootstrap":

As mentioned earlier, the main result for this section is the implied probabilities, which are given below for 3 different banks for which data was available. The R code to generate these results has been added at the end of this paper.

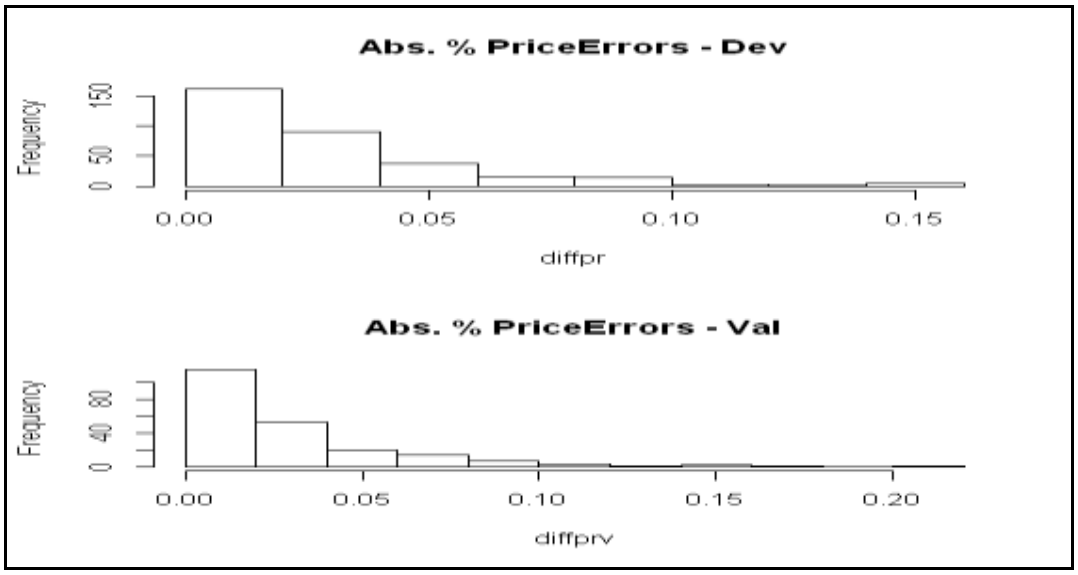
CDS-Bootstrap Results for three Banks (based on Senior CDS Spreads)

Maturity (Years)	LGD Assumption	CDS Spreads for Bank 1 (in bp)	Implied PD for Bank 1	CDS Spreads for Bank 2 (in bp)	Implied PD for Bank 2	CDS Spreads for Bank 3 (in bp)	Implied PD for Bank 3
1	0.45	47.23	0.86%	56.45	1.03%	130.6	2.35%
2	0.45	60.845	2.21%	68.71	2.49%	170	6.04%
3	0.45	80.215	4.36%	84.075	4.55%	213.7	11.19%
4	0.45	101.215	7.30%	103.6	7.44%	250.2	17.09%
5	0.45	123.175	11.04%	124.19	11.08%	277.6	23.10%
6	0.45	141.77	15.09%	139.725	14.81%	304.9	29.63%
7	0.45	154.9	18.98%	150.725	18.40%	323.4	35.57%
8	0.45	161.61	22.25%	158.8	21.82%	336.3	40.93%
9	0.45	166.785	25.39%	165.01	25.12%	345.6	45.79%
10	0.45	170.865	27.67%	169.89	27.40%	353.2	49.14%

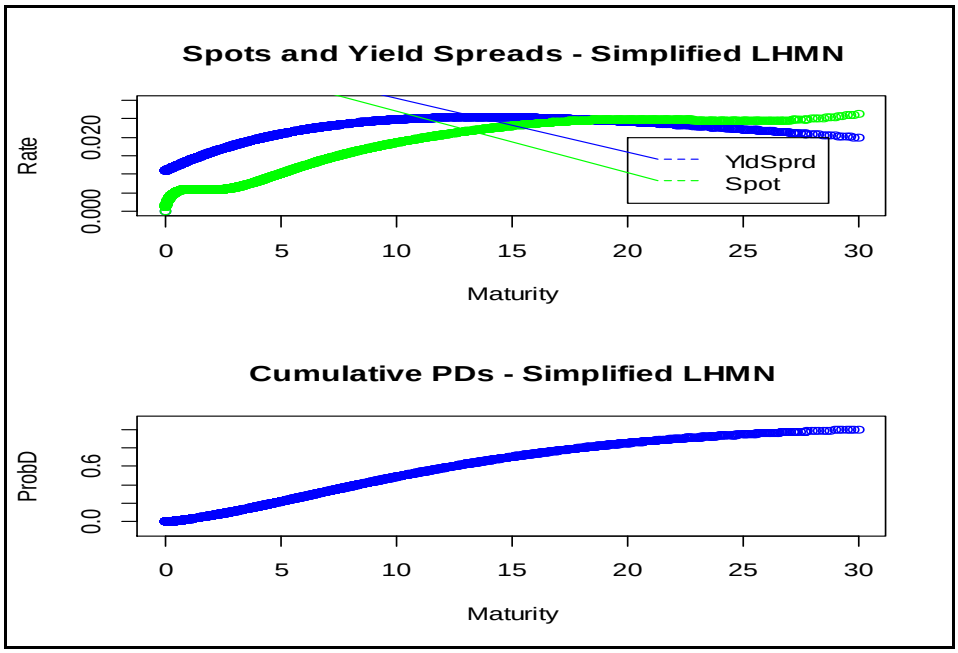
Since default-adjusted cash flows were not generated using these results (being based on a completely different data set), we jump to the results from the next approach, for which default-adjusted cash flows were also generated and checked.

2. Results from the Simplified Lehmann Approach:

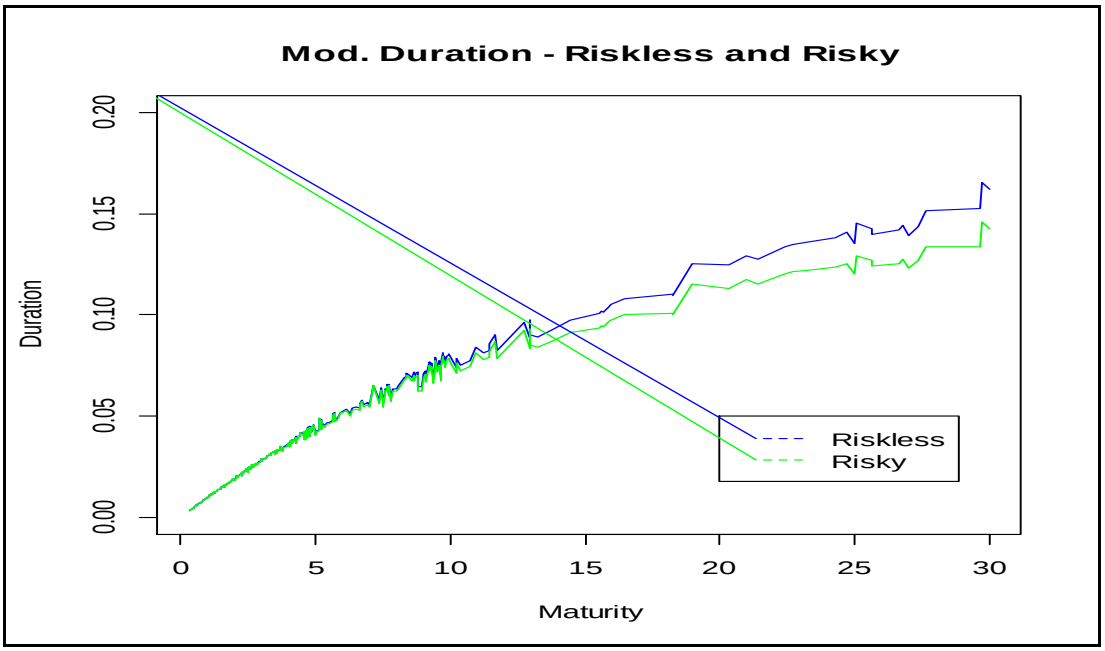
The percentage price-errors in development and validation data were as follows, with medians being 2.04% and 1.84% respectively.



The resulting implied default probabilities and credit spreads were as follows.

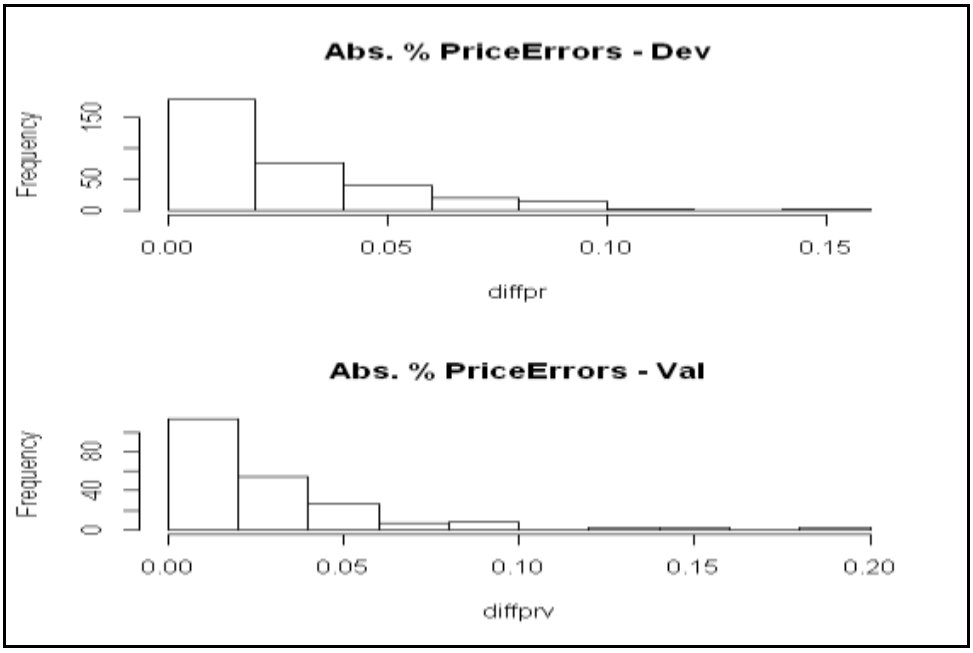


Finally, the riskless and risky modified durations were calculated by shocking the risk-free discount curve by 1 bp and looking at the percentage change in predicted prices due to this change. For the riskless case, the contractual cash flows were used “as-is”, whereas for the risky case, the expected cash flows at each coupon period (given the implied default probabilities) were used instead of the contractual cash flows. As the graph below shows, duration declines, especially for the higher maturity bonds in the data set.

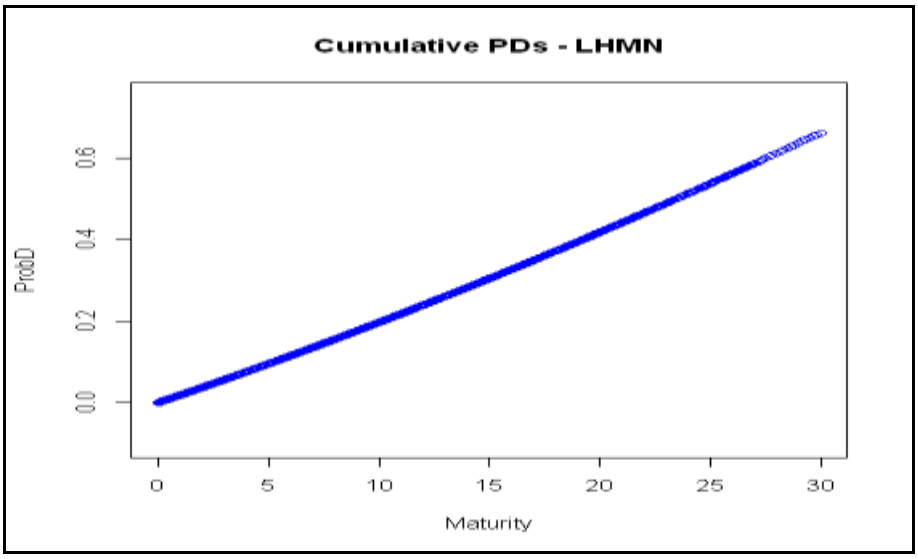


3. Results from the Lehmann Approach:

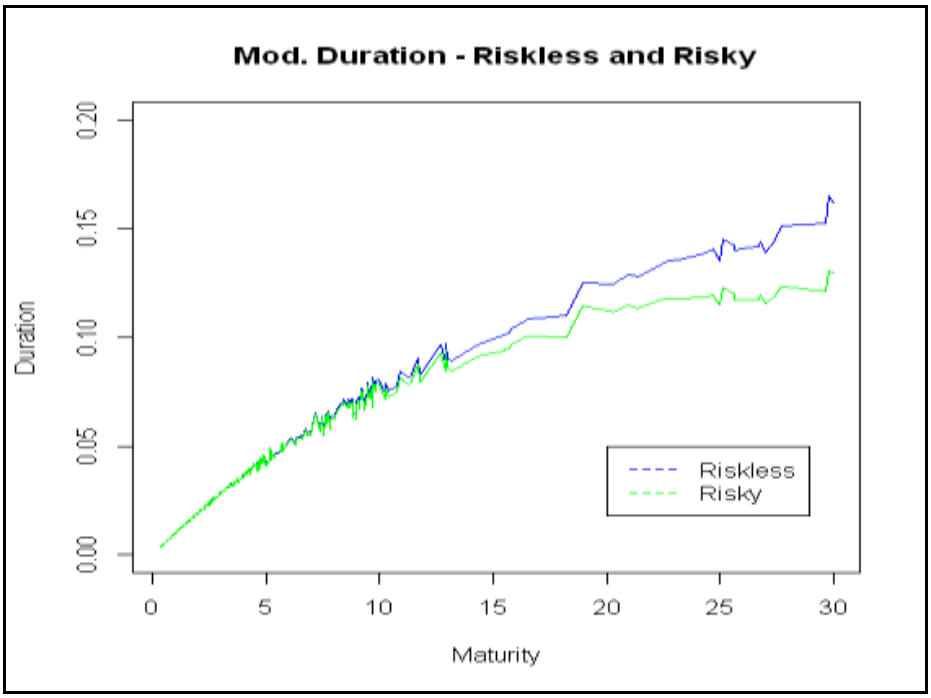
The percentage price-errors in development and validation data were as follows, with medians being 1.82% and 1.79% respectively.



The resulting implied default probabilities were as follows. Spreads were not calculated since the usual formula might not apply here.



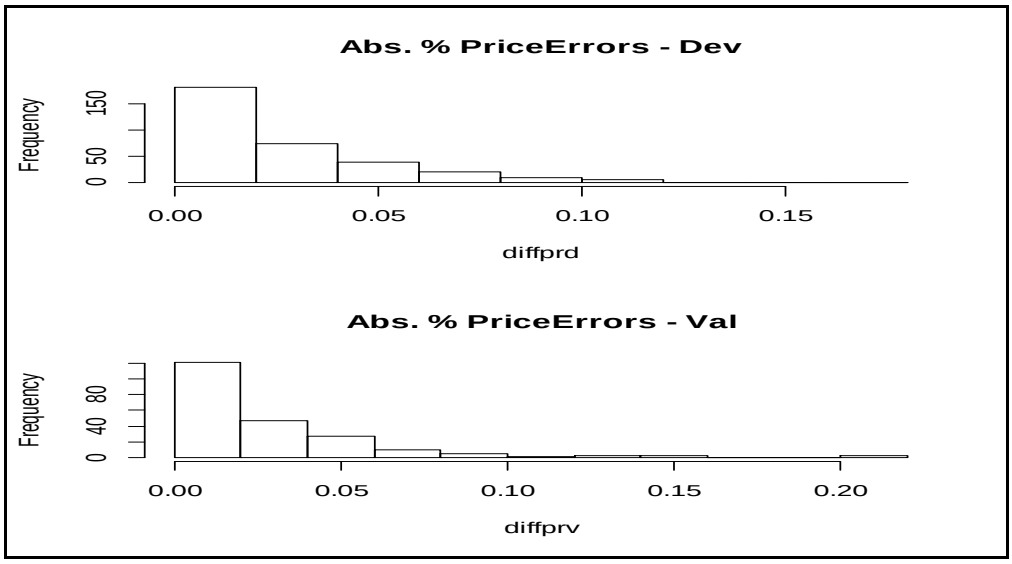
Modified duration – calculated using a 1bp shock to the risk-free interest rate – declines much like in the previous example.



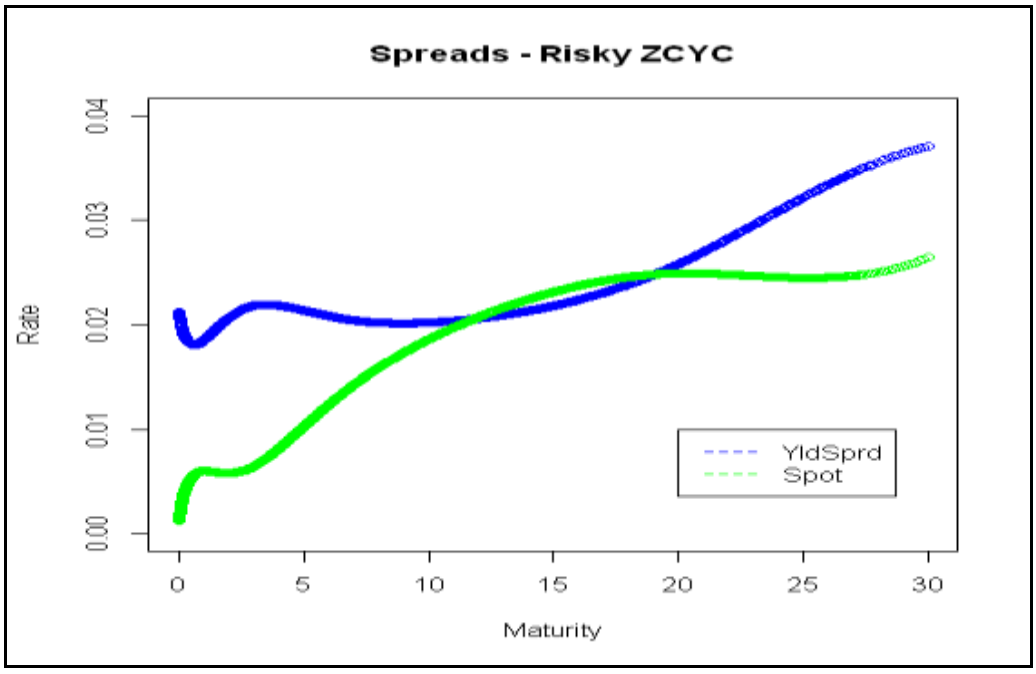
The expected cash flow calculation for this method requires two probabilities for each date. One is the probability of survival up to that time point, and the other is the probability that the customer defaults in that time period, conditional upon survival up to the previous period. All other probabilities are irrelevant under RM-type recovery because all those nodes lead to zero cash flows. Recoveries are a constant fraction of face value, as mentioned earlier.

4. Results from Using Risky Zero-Coupon Curves:

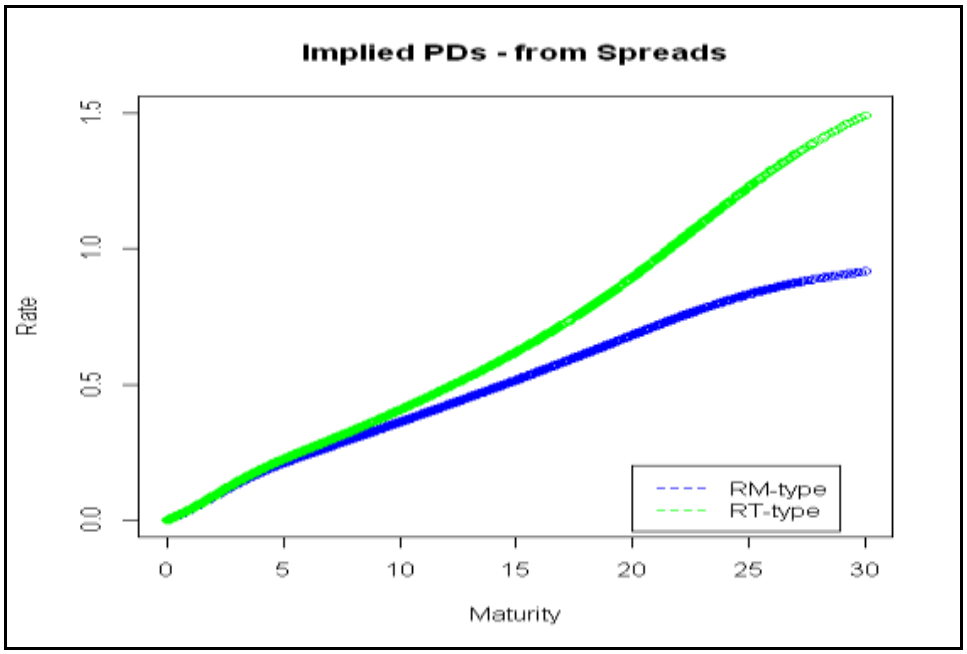
The percentage price-errors in development and validation data were as follows, with medians being 1.69% and 1.68% respectively.



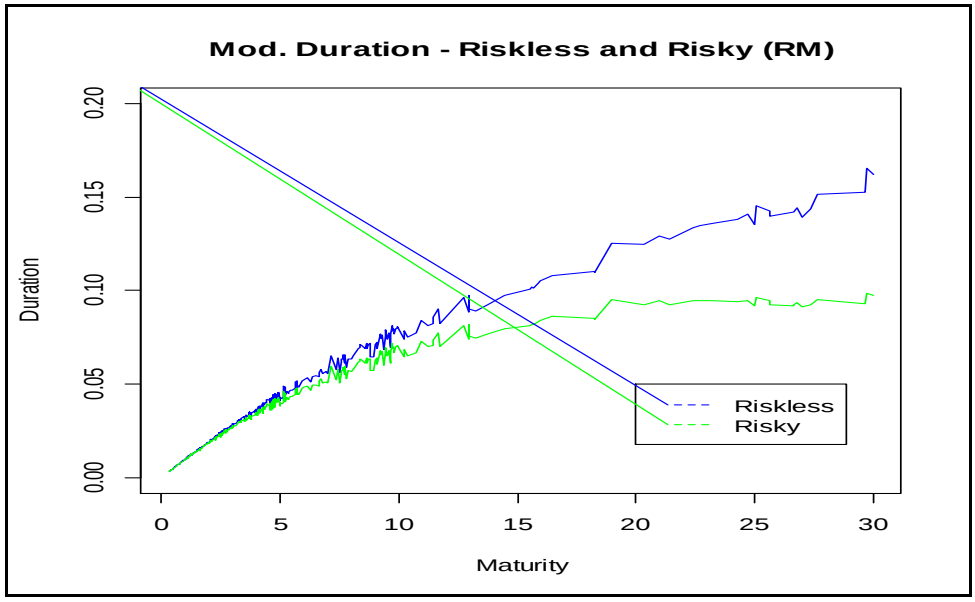
The resulting implied default probabilities and credit spreads were as follows.



The resulting implied default probabilities by the two methods were as follows. As the graph below shows, a simple application of the theoretical formulae leads to implied spreads bounded between 0 and 1 for the RM-type assumption but not for the RT-type assumption. For derivation of default-adjusted cash flows, this does not matter, but in other exercises, we could simply use the implied PD from the RM assumption.



The impact on duration (using the PDs and expected cash flows assuming RM-type recoveries) was as follows.



The impact on duration (using the PDs and expected cash flows assuming RT-type recoveries) is more complicated and is not being given here.

A note on internal consistency checking is due here. Any set of expected cash flows generated using implied PDs, when discounted back using risk-free rates, should give us predicted bond prices which are equal to the predictions from the credit spreads model. Here, when we use the implied PDs from the RT-type assumption and generate cash-flows in a manner similar to our simplified Lehmann approach, we match the predictions of the spreads model perfectly. However, these implied PDs are not bounded between zero and one, as the graph above shows and

therefore, must be used and interpreted with caution. On the other hand, when we use the RM-type recovery assumption, the PDs are bounded between 0 and 1, but it is much more difficult to generate the expected cash flows for this scenario. In this paper, we only provide a sketch of how that could be done for our exercise, but stop short of publishing results here.

In the RM-type approach, we postulate that the claim at the time of default would be the price of the bond just prior to default. To derive this price, we use the forward rates as of each given time point to discount the cash flows, looking at the residual maturity of the remaining cash flows. Next, we use a framework similar to the Lehmann framework in which the expected cash flow calculation requires two probabilities for each date. However, the claim of the lender in this model is completely different – it is equal to the price of the bond just prior to default, and that price needs to be calculated using forward rates and residual maturities at each cash flow date.

The implied PDs were derived from the estimated spreads using the formula given in Tobe (2012) directly, with no refinement for errors in estimation of credit spreads.

Concluding Remarks

In this paper we derive market-implied (i.e., risk-neutral) PDs using several alternative methods, and also look at the impact that credit risk has on the modified duration of bonds using 3 alternative methods. Duration falls unambiguously, and more for higher maturities, regardless of which method of the 3 methods we use to derive the credit-adjusted cash flows.

While there are yet other methods of estimating implied PDs (some more sophisticated like calibration of transition matrix generators and others much simpler like bootstrapping assuming piecewise-constant PDs), we have left some of these methods for future exercises, since this is a beginning paper and since the methods studied here are also very popular in the literature and in the industry.

Sadanand Tutakne
Senior Manager, Financial Services Analytics
GENPACT
e-mail: sadanand.tutakne@genpact.com

About the Author

Sadanand Tutakne, Senior Manager, Financial Services Analytics, GENPACT, has about 11.5 years of corporate experience in behavior and credit scoring, loss forecasting, credit modeling for Basel and econometric modeling for Treasury models, e.g., yield curve modeling, modeling credit spreads and implied PDs and scenarios for yield curves. Prior to joining the corporate sector, he was a doctoral candidate (ABD) at the University of Maryland at College Park, where he worked on vector autoregressive models for quantifying the effects of interest rate policy shocks on macroeconomic variables.

References

1. Arvanitis, Angelo, Jonathon Gregory and Jean-Paul Laurent, "Building Models for Credit Spreads," The Journal of Derivatives 1999.6.3:27-43.

Link: http://www.iijournals.com/doi/abs/10.3905/jod.1999.319117

OR

http://www.ressources-actuarielles.net/ext/isfa/1226.nsf/9c8e3fd4d8874d60c1257052003eced6/7c1d935203184237c1257a4f006b127a/$FILE/building_models_for_credit_spreads.pdf

2. Berd, Arthur M., Roy Mashal and Peili Wang, "Estimating Implied Default Probabilities from Credit Bond Prices," Fixed Income Quantitative Credit Research, Lehmann Brothers, 7th August 2003.
Link: ftp://ftp.idc.ac.il/Faculty/Roy/Pub/QCRQ_Curves.pdf

3. The "CreditCurve_Bootstrapping.xls" tool available freely online at the following website.
Link: http://www.quantcode.com/modules/mydownloads/viewallquicklist.php

4. Duffie, Darrell and Kenneth J. Singleton, "Modeling Term Structures of Defaultable Bonds," Review of Financial Studies,1999, Volume 12: 687-720.
Link: http://www.stanford.edu/~duffie/ds.pdf

5. Hull, John and Alan White, "Valuing Credit Default Swaps I: No Counterparty Default Risk," NYU Working Paper No. FIN-00-021,
Link: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1295226

6. Jarrow, Robert A. and Stuart M. Turnbull, "Pricing Derivatives on Financial Securities Subject to Credit Risk," The Journal of Finance, Vol. 50, No. 1. (Mar., 1995), pp. 53-85.
Link: http://www.econ.hku.hk/~yjtang/CRM/JarrowTurnbull1995JF_ReducedFormModel.pdf

7. J.P. Morgan, "The J.P. Morgan Guide to Credit Derivatives"
Link: http://www.investinginbonds.com/assets/files/intro_to_credit_derivatives.pdf

8. JPMorgan Securities Inc., "Par Credit Default Swap Spread Approximation from Default Probabilities," October 24, 2001.
Link: http://www.scribd.com/doc/19606757/JP-Morgan-Par-Credit-Default-Swap-Spread-Approximation-From-Default-Probabilities

9. Loffler, Gunter and Peter N. Posch, "Credit Risk Modelling using Excel and VBA," Wiley Finance series, John Wiley and Sons, 2007.
Link: http://loeffler-posch.com/?page_id=58

10. Tobe, Reiko, "Estimating Default Probabilities using Corporate Bond Price Data," CFEE Discussion Paper Series No. 2011-1, March 13th, 2012.
Link: http://www.econ.hit-u.ac.jp/~finmodel/papers/DP2011-1Tobe.pdf

R Code for the CDS Bootstrap

```
### Assuming  discrete compounding ###;
implied_pd<- matrix(0,10,16);

for (col in 3:18){
cdsdat<-read.csv("/…path…/CDS.csv",header=TRUE);

rf<- cdsdat$riskfree/100;
RR<- 0.45;
npay<- 2; ### assuming two semi-annual premium payments;
cdsdat$cds<- cdsdat[,col] ; ###Pick Spreads for the Bank for which probs are required##;
cdspay<- (cdsdat$cds/10000) * 100 * (1/npay);  ###Premium payment made per time period;

####################################
####### Lambda for Year 1 ############
####################################

PVdiff1<- function(l01) {
df1<- matrix(0,1,npay);
for (k in 1:npay){
df1[k] <- 1/(1+rf[1])^(k/npay)
};

surv1<- matrix(0,npay,1);
for (k in 1:npay){
surv1[k] <- exp(-l01 * k/npay)
};

return(100*(1-RR)* (1/(1+rf[1])) * (1-exp(-l01)) - cdspay[1] * (df1 %*% surv1) )
}

l01<- uniroot(PVdiff1,c(-1,1))$root ;      ######Solve nonlinear equation to get lambda for first year
e1<- exp(-l01);

df1<- matrix(0,1,npay);
for (k in 1:npay){
df1[k] <- 1/(1+rf[1])^(k/npay)
};

surv1<- matrix(0,npay,1);
for (k in 1:npay){
surv1[k] <- exp(-l01 * k/npay)
};

####################################
####### Lambda for Year 2 ############
####################################
```

```
PVdiff2<- function(l02) {
df2<- matrix(0,1,npay);
for (k in 1:npay){
df2[k] <- 1/(1+rf[2])^(1+k/npay)
};

surv2<- matrix(0,npay,1);
for (k in 1:npay){
surv2[k] <- exp(-l01 - l02 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) ) -
cdspay[2] * (df1 %*% surv1) - cdspay[2] * (df2 %*% surv2))
}

l02<- uniroot(PVdiff2,c(-1,1))$root ;   ######Solve nonlinear equation to get lambda for 2nd year
e2<- exp(-l01-l02);

df2<- matrix(0,1,npay);
for (k in 1:npay){
df2[k] <- 1/(1+rf[2])^(1+k/npay)
};

surv2<- matrix(0,npay,1);
for (k in 1:npay){
surv2[k] <- exp(-l01 - l02 * k/npay)
};

####################################
####### Lambda for Year 3 ############
####################################
PVdiff3<- function(l03) {
df3<- matrix(0,1,npay);
for (k in 1:npay){
df3[k] <- 1/(1+rf[3])^(2+k/npay)
};

surv3<- matrix(0,npay,1);
for (k in 1:npay){
surv3[k] <- exp(-l01-l02 - l03 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) ) -
cdspay[3] * (df1 %*% surv1) - cdspay[3] * (df2 %*% surv2) - cdspay[3] * (df3 %*% surv3) )
}
```

```
l03<- uniroot(PVdiff3,c(-1,1))$root ;
e3<- exp(-l01-l02-l03);

df3<- matrix(0,1,npay);
for (k in 1:npay){
df3[k] <- 1/(1+rf[3])^(2+k/npay)
};

surv3<- matrix(0,npay,1);
for (k in 1:npay){
surv3[k] <- exp(-l01-l02 - l03 * k/npay)
};

####################################
####### Lambda for Year 4 ############
####################################
PVdiff4<- function(l04) {
df4<- matrix(0,1,npay);
for (k in 1:npay){
df4[k] <- 1/(1+rf[4])^(3+k/npay)
};

surv4<- matrix(0,npay,1);
for (k in 1:npay){
surv4[k] <- exp(-l01-l02-l03 - l04 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) ) -
cdspay[4] * (df1 %*% surv1) - cdspay[4] * (df2 %*% surv2) - cdspay[4] * (df3 %*% surv3) -
cdspay[4] * (df4 %*% surv4) )
}

l04<- uniroot(PVdiff4,c(-1,1))$root ;
e4<- exp(-l01-l02-l03-l04);

df4<- matrix(0,1,npay);
for (k in 1:npay){
df4[k] <- 1/(1+rf[4])^(3+k/npay)
};

surv4<- matrix(0,npay,1);
for (k in 1:npay){
surv4[k] <- exp(-l01-l02-l03 - l04 * k/npay)
};
```

```
###################################
####### Lambda for Year 5 ############
###################################

PVdiff5<- function(l05) {
df5<- matrix(0,1,npay);
for (k in 1:npay){
df5[k] <- 1/(1+rf[5])^(4+k/npay)
};

surv5<- matrix(0,npay,1);
for (k in 1:npay){
surv5[k] <- exp(-l01-l02-l03-l04 - l05 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) +
(1/(1+rf[5])^5)*(exp(-l01-l02-l03-l04)-exp(-l01-l02-l03-l04-l05)) ) -
cdspay[5] * (df1 %*% surv1) - cdspay[5] * (df2 %*% surv2) - cdspay[5] * (df3 %*% surv3) -
cdspay[5] * (df4 %*% surv4) - cdspay[5] * (df5 %*% surv5) )
}

l05<- uniroot(PVdiff5,c(-1,1))$root ;
e5<- exp(-l01-l02-l03-l04-l05);

df5<- matrix(0,1,npay);
for (k in 1:npay){
df5[k] <- 1/(1+rf[5])^(4+k/npay)
};

surv5<- matrix(0,npay,1);
for (k in 1:npay){
surv5[k] <- exp(-l01-l02-l03-l04 - l05 * k/npay)
};

###################################
####### Lambda for Year 6 ############
###################################
PVdiff6<- function(l06) {
df6<- matrix(0,1,npay);
for (k in 1:npay){
df6[k] <- 1/(1+rf[6])^(5+k/npay)
};

surv6<- matrix(0,npay,1);
for (k in 1:npay){
surv6[k] <- exp(-l01-l02-l03-l04-l05 - l06 * k/npay)
};
```

```
return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) +
(1/(1+rf[5])^5)*(exp(-l01-l02-l03-l04)-exp(-l01-l02-l03-l04-l05)) +
(1/(1+rf[6])^6)*(exp(-l01-l02-l03-l04-l05)-exp(-l01-l02-l03-l04-l05-l06)) ) -
cdspay[6] * (df1 %*% surv1) - cdspay[6] * (df2 %*% surv2) - cdspay[6] * (df3 %*% surv3) -
cdspay[6] * (df4 %*% surv4) - cdspay[6] * (df5 %*% surv5) - cdspay[6] * (df6 %*% surv6) )
}

l06<- uniroot(PVdiff6,c(-1,1))$root ;
e6<- exp(-l01-l02-l03-l04-l05-l06);

df6<- matrix(0,1,npay);
for (k in 1:npay){
df6[k] <- 1/(1+rf[6])^(5+k/npay)
};

surv6<- matrix(0,npay,1);
for (k in 1:npay){
surv6[k] <- exp(-l01-l02-l03-l04-l05 - l06 * k/npay)
};

####################################
####### Lambda for Year 7 ############
####################################

PVdiff7<- function(l07) {
df7<- matrix(0,1,npay);
for (k in 1:npay){
df7[k] <- 1/(1+rf[7])^(6+k/npay)
};

surv7<- matrix(0,npay,1);
for (k in 1:npay){
surv7[k] <- exp(-l01-l02-l03-l04-l05-l06 - l07 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) +
(1/(1+rf[5])^5)*(exp(-l01-l02-l03-l04)-exp(-l01-l02-l03-l04-l05)) +
(1/(1+rf[6])^6)*(exp(-l01-l02-l03-l04-l05)-exp(-l01-l02-l03-l04-l05-l06)) +
(1/(1+rf[7])^7)*(exp(-l01-l02-l03-l04-l05-l06)-exp(-l01-l02-l03-l04-l05-l06-l07)) ) -
cdspay[7] * (df1 %*% surv1) - cdspay[7] * (df2 %*% surv2) - cdspay[7] * (df3 %*% surv3) -
cdspay[7] * (df4 %*% surv4) - cdspay[7] * (df5 %*% surv5) - cdspay[7] * (df6 %*% surv6) -
cdspay[7] * (df7 %*% surv7) )
}

l07<- uniroot(PVdiff7,c(-1,1))$root ;
```

```
e7<- exp(-l01-l02-l03-l04-l05-l06-l07);

df7<- matrix(0,1,npay);
for (k in 1:npay){
df7[k] <- 1/(1+rf[7])^(6+k/npay)
};

surv7<- matrix(0,npay,1);
for (k in 1:npay){
surv7[k] <- exp(-l01-l02-l03-l04-l05-l06 - l07 * k/npay)
};

####################################
####### Lambda for Year 8 ############
####################################

PVdiff8<- function(l08) {
df8<- matrix(0,1,npay);
for (k in 1:npay){
df8[k] <- 1/(1+rf[8])^(7+k/npay)
};

surv8<- matrix(0,npay,1);
for (k in 1:npay){
surv8[k] <- exp(-l01-l02-l03-l04-l05-l06-l07 - l08 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) +
(1/(1+rf[5])^5)*(exp(-l01-l02-l03-l04)-exp(-l01-l02-l03-l04-l05)) +
(1/(1+rf[6])^6)*(exp(-l01-l02-l03-l04-l05)-exp(-l01-l02-l03-l04-l05-l06)) +
(1/(1+rf[7])^7)*(exp(-l01-l02-l03-l04-l05-l06)-exp(-l01-l02-l03-l04-l05-l06-l07)) +
(1/(1+rf[8])^8)*(exp(-l01-l02-l03-l04-l05-l06-l07)-exp(-l01-l02-l03-l04-l05-l06-l07-l08)) ) -
cdspay[8] * (df1 %*% surv1) - cdspay[8] * (df2 %*% surv2) - cdspay[8] * (df3 %*% surv3) -
cdspay[8] * (df4 %*% surv4) - cdspay[8] * (df5 %*% surv5) - cdspay[8] * (df6 %*% surv6) -
cdspay[8] * (df7 %*% surv7) - cdspay[8] * (df8 %*% surv8))
}

l08<- uniroot(PVdiff8,c(-1,1))$root ;
e8<- exp(-l01-l02-l03-l04-l05-l06-l07-l08);

df8<- matrix(0,1,npay);
for (k in 1:npay){
df8[k] <- 1/(1+rf[8])^(7+k/npay)
};
```

```
surv8<- matrix(0,npay,1);
for (k in 1:npay){
surv8[k] <- exp(-l01-l02-l03-l04-l05-l06-l07 - l08 * k/npay)
};

###################################
####### Lambda for Year 9 ##########
###################################

PVdiff9<- function(l09) {
df9<- matrix(0,1,npay);
for (k in 1:npay){
df9[k] <- 1/(1+rf[9])^(8+k/npay)
};

surv9<- matrix(0,npay,1);
for (k in 1:npay){
surv9[k] <- exp(-l01-l02-l03-l04-l05-l06-l07-l08 - l09 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) +
(1/(1+rf[5])^5)*(exp(-l01-l02-l03-l04)-exp(-l01-l02-l03-l04-l05)) +
(1/(1+rf[6])^6)*(exp(-l01-l02-l03-l04-l05)-exp(-l01-l02-l03-l04-l05-l06)) +
(1/(1+rf[7])^7)*(exp(-l01-l02-l03-l04-l05-l06)-exp(-l01-l02-l03-l04-l05-l06-l07)) +
(1/(1+rf[8])^8)*(exp(-l01-l02-l03-l04-l05-l06-l07)-exp(-l01-l02-l03-l04-l05-l06-l07-l08)) +
(1/(1+rf[9])^9)*(exp(-l01-l02-l03-l04-l05-l06-l07-l08)-exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09)) ) -
cdspay[9] * (df1 %*% surv1) - cdspay[9] * (df2 %*% surv2) - cdspay[9] * (df3 %*% surv3) -
cdspay[9] * (df4 %*% surv4) - cdspay[9] * (df5 %*% surv5) - cdspay[9] * (df6 %*% surv6) -
cdspay[9] * (df7 %*% surv7) - cdspay[9] * (df8 %*% surv8) - cdspay[9] * (df9 %*% surv9) )
}

l09<- uniroot(PVdiff9,c(-1,1))$root ;
e9<- exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09);

df9<- matrix(0,1,npay);
for (k in 1:npay){
df9[k] <- 1/(1+rf[9])^(8+k/npay)
};

surv9<- matrix(0,npay,1);
for (k in 1:npay){
surv9[k] <- exp(-l01-l02-l03-l04-l05-l06-l07-l08 - l09 * k/npay)
};

###################################
####### Lambda for Year 10 ###########
###################################
```

```
PVdiff10<- function(l10) {
df10<- matrix(0,1,npay);
for (k in 1:npay){
df10[k] <- 1/(1+rf[10])^(9+k/npay)
};

surv10<- matrix(0,npay,1);
for (k in 1:npay){
surv10[k] <- exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09 - l10 * k/npay)
};

return(100*(1-RR)*( (1/(1+rf[1]))*(1-exp(-l01)) + (1/(1+rf[2])^2)*(exp(-l01)-exp(-l01-l02)) +
(1/(1+rf[3])^3)*(exp(-l01-l02)-exp(-l01-l02-l03)) + (1/(1+rf[4])^4)*(exp(-l01-l02-l03)-exp(-l01-l02-l03-l04)) +
(1/(1+rf[5])^5)*(exp(-l01-l02-l03-l04)-exp(-l01-l02-l03-l04-l05)) +
(1/(1+rf[6])^6)*(exp(-l01-l02-l03-l04-l05)-exp(-l01-l02-l03-l04-l05-l06)) +
(1/(1+rf[7])^7)*(exp(-l01-l02-l03-l04-l05-l06)-exp(-l01-l02-l03-l04-l05-l06-l07)) +
(1/(1+rf[8])^8)*(exp(-l01-l02-l03-l04-l05-l06-l07)-exp(-l01-l02-l03-l04-l05-l06-l07-l08)) +
(1/(1+rf[9])^9)*(exp(-l01-l02-l03-l04-l05-l06-l07-l08)-exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09)) +
(1/(1+rf[10])^10)*(exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09)-exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09-l10)) ) -
cdspay[9] * (df1 %*% surv1) - cdspay[9] * (df2 %*% surv2) - cdspay[9] * (df3 %*% surv3) -
cdspay[9] * (df4 %*% surv4) - cdspay[9] * (df5 %*% surv5) - cdspay[9] * (df6 %*% surv6) -
cdspay[9] * (df7 %*% surv7) - cdspay[9] * (df8 %*% surv8) - cdspay[9] * (df9 %*% surv9) -
cdspay[10] * (df10 %*% surv10) )
}

l10<- uniroot(PVdiff10,c(-1,1))$root ;
e10<- exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09-l10);

df10<- matrix(0,1,npay);
for (k in 1:npay){
df10[k] <- 1/(1+rf[10])^(9+k/npay)
};

surv10<- matrix(0,npay,1);
for (k in 1:npay){
surv10[k] <- exp(-l01-l02-l03-l04-l05-l06-l07-l08-l09 - l10 * k/npay)
};


####################################
####### Tabulate All Results ###########
####################################
survmat<- as.matrix(c(e1,e2,e3,e4,e5,e6,e7,e8,e9,e10));
implied_pd[,(col-2)]<- 1-survmat;
};
write.csv(implied_pd,"/…path…./imp_pd_hzd.csv");
####################################
```