Pricing Rates Derivatives under HJM Model

Plamen Stilyianov June 2009

Table of Contents

Introduction

The pricing of interest rate derivatives under the Heath-Jarrow-Morton (HJM) model is the main topic of this paper.

The HJM framework refers to a class of models that are derived by directly modelling the dynamics of the instantaneous forward rates. The basic insight of this framework is to recognise that there exists a specific relationship between the drift and volatility parameters of the forward rate dynamics in a no arbitrage world. HJM were instrumental in allowing the forward rates become the state variables in the model as opposed to quantities derived from the spot rates.

With historical data of UK yield curves, I estimate the model parameters of the HJM model, using an important technique called Principal Component Analysis (PCA).

The purpose of this paper is to apply the HJM framework using Monte Carlo simulations in the programming language VBA for the estimation of the future spot rate. This rate will then be used for the pricing of caps. The volatility structure is chosen to follow an exponential model.

Extract from the VBA code algorithms is enclosed in the sections discussing the implemented models. The attached CD contains the complete code and software application discussed in this paper.

Historical Data

This study requires the use of historical data on the UK forward curve from Bank of England. Which can be found at;

http://www.bankofengland.co.uk/statistics/yieldcurve/archive.htm

The Excel program stores all the forward rates used during its price computation in “Rates” sheet. Modifications to this sheet will impact generated prices.

The data is in matrix format with columns of forward years and rows indicating the daily changes.

UK instantaneous nominal forward curve

Maturity years:	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5
03 Jan 00													
04 Jan 00	6.44	6.71	6.65	6.50	6.33	6.15	5.99	5.84	5.71	5.57	5.44	5.30	5.16
05 Jan 00	6.45	6.75	6.68	6.54	6.39	6.23	6.08	5.95	5.82	5.69	5.56	5.43	5.28
06 Jan 00	6.44	6.74	6.68	6.56	6.41	6.26	6.12	5.98	5.84	5.71	5.57	5.43	5.28
07 Jan 00	6.41	6.69	6.62	6.49	6.35	6.20	6.06	5.93	5.79	5.66	5.52	5.38	5.23
10 Jan 00	6.40	6.64	6.55	6.42	6.27	6.13	5.98	5.85	5.72	5.58	5.44	5.30	5.15
11 Jan 00	6.45	6.70	6.62	6.50	6.36	6.22	6.08	5.95	5.82	5.69	5.55	5.40	5.25
12 Jan 00	6.47	6.70	6.64	6.52	6.40	6.26	6.13	6.00	5.88	5.75	5.61	5.47	5.31
13 Jan 00	6.45	6.66	6.59	6.47	6.33	6.19	6.05	5.91	5.77	5.63	5.49	5.35	5.19
14 Jan 00	6.44	6.67	6.61	6.49	6.36	6.22	6.09	5.96	5.83	5.69	5.56	5.42	5.26
17 Jan 00	6.47	6.73	6.67	6.55	6.42	6.28	6.14	6.01	5.88	5.75	5.61	5.47	5.32
18 Jan 00	6.49	6.78	6.75	6.65	6.53	6.40	6.27	6.14	6.01	5.87	5.73	5.58	5.43
19 Jan 00	6.43	6.72	6.70	6.61	6.48	6.34	6.20	6.07	5.93	5.79	5.65	5.50	5.35
20 Jan 00	6.44	6.74	6.73	6.65	6.53	6.40	6.27	6.13	6.00	5.86	5.71	5.56	5.40
21 Jan 00	6.42	6.72	6.71	6.62	6.50	6.36	6.23	6.09	5.95	5.81	5.67	5.52	5.36
24 Jan 00	6.41	6.71	6.70	6.61	6.49	6.36	6.22	6.08	5.94	5.80	5.65	5.50	5.34
25 Jan 00	6.39	6.67	6.65	6.56	6.43	6.29	6.14	6.00	5.86	5.71	5.56	5.40	5.25
26 Jan 00	6.34	6.61	6.60	6.52	6.40	6.27	6.14	6.00	5.86	5.72	5.58	5.43	5.27
27 Jan 00	6.41	6.70	6.69	6.61	6.49	6.37	6.24	6.11	5.98	5.84	5.70	5.54	5.39
28 Jan 00	6.43	6.72	6.72	6.64	6.53	6.40	6.27	6.14	6.01	5.87	5.72	5.57	5.41
31 Jan 00	6.39	6.66	6.64	6.55	6.43	6.30	6.16	6.03	5.89	5.74	5.59	5.44	5.28
01 Feb 00	6.34	6.58	6.56	6.45	6.32	6.17	6.02	5.87	5.72	5.57	5.41	5.25	5.09
02 Feb 00	6.29	6.55	6.55	6.46	6.34	6.20	6.05	5.90	5.75	5.60	5.44	5.28	5.12
03 Feb 00	6.28	6.53	6.54	6.45	6.31	6.16	6.01	5.85	5.69	5.53	5.37	5.20	5.03
04 Feb 00	6.33	6.59	6.59	6.50	6.38	6.24	6.10	5.95	5.80	5.64	5.48	5.32	5.15
07 Feb 00	6.36	6.61	6.61	6.52	6.41	6.27	6.13	5.98	5.83	5.67	5.50	5.33	5.16
08 Feb 00	6.32	6.56	6.57	6.49	6.37	6.24	6.09	5.94	5.79	5.63	5.46	5.29	5.11

Principal Components Analysis

Principal Component Analysis (PCA) is a multivariate procedure which combines two or more correlated variables into a smaller number of factors or principal components. Essentially, a set of correlated variables are transformed into a set of uncorrelated factors which are ordered by reducing variability. The main use of PCA is to reduce the dimensionality of a data set while retaining as much information as is possible. The procedure is explained in three processes.

Step1: Data Normalization.

The way to normalize the data is to create a row-by-row difference on the M x N matrix. This would result us (M-1) x N matrix.

Pseudocode for difference matrixes:

```
For i = 2 To 1306
  If Sheets("Rates").Cells(i, 2).Value <> "ND" Then
    For j = 1 To 50
      tmpMat(counter, j) = Sheets("Rates").Cells(i, j + 1).Value
    Next j
    counter = counter + 1
  End If
Next i
Dim diffMat As Variant
ReDim diffMat(1 To counter - 2, 1 To 50)
For i = 1 To UBound(diffMat, 1)
  For j = 1 To UBound(diffMat, 2)
    diffMat(i, j) = tmpMat(i + 1, j) - tmpMat(i, j)
  Next j
Next i
```

Step2: Co-variance calculation.

Next procedure is to create a symmetric N x N matrix which is used in computation of Eigen values.

$$Cov(X,Y) = \frac{\sum(x - \bar{x})(y - \bar{y})}{n}$$

Pseudopodia for covariance calculation:

```
  For i = 1 To n
    covMat(i, i) = Application.Covar(seriesArr(i), seriesArr(i)) * 10000
    For j = i + 1 To n
      tmpVal = Application.Covar(seriesArr(i), seriesArr(j)) * 10000
      covMat(i, j) = tmpVal
      covMat(j, i) = tmpVal
    Next j
  Next i
  CalculateEigenValues covMat
End Sub
```

Step3: Eigen Value & Eigen Vector Computation

Finding Eigen values/vector is fundamental to the PCA analysis since the key Eigen values/vectors become the key principal components. The chosen method of finding Eigen values in this paper is Jacobi Method.

Jacobi is an iterative method for finding Eigen values/vector in a symmetric matrix. The logic is simple and composed of the following steps:

Pseudocode for calculating eigenvalues:

```
Do While ((offSum > tol_) And (counter < maxIterations_))
  rotMatrix = GetRotationMatrix(tmpSmat)
  'rotMatrix = GetRotationThetaMatrix(tmpSmat)
  tmpSmat = Application.MMult(Application.Transpose(rotMatrix), Application.MMult(tmpSmat, rotMatrix))
  tmpEigMat = Application.MMult(tmpEigMat, rotMatrix)
  offSum = GetOffDiagonalSum(tmpSmat)
  counter = counter + 1
Loop
ReDim eigenValuesVec_(1 To n)
Dim i As Single
For i = 1 To n
  eigenValuesVec_(i) = tmpSmat(i, i)
Next i
```

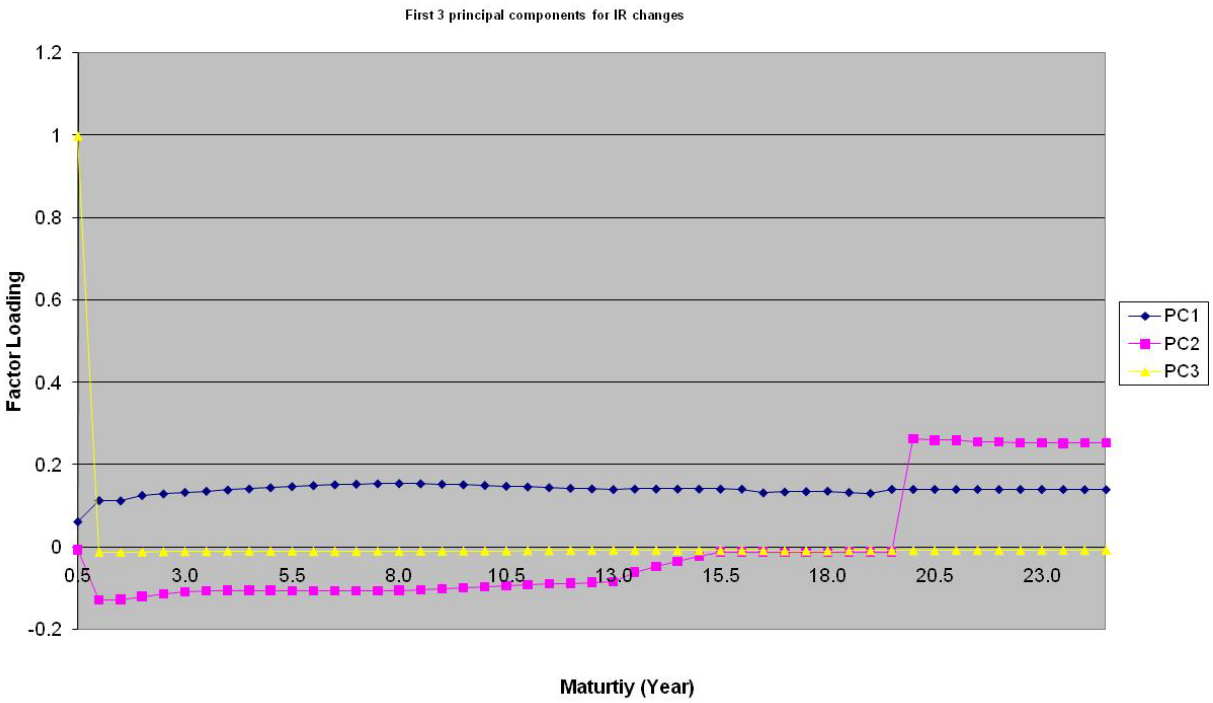
Screenshot of PCA calculation

Press the "Start PCA" button to start PCA analysis [Start PCA]

		0.08039	0.008204	0.002217	0.000786	0.000784	0.00077	0.000769	0.000758	0.000737	0.000714	0.000695
		PC1	PC2	PC3								
	EigenValues	803.8987	82.03532	22.16621	7.856426	7.844168	7.700445	7.689302	7.584317	7.373517	7.137351	6.954269
Maturity (Years)												
0.5	EigenVectots	0.063128	-0.00488	0.997952	-0.00137	-0.00159	-0.00113	-0.00178	-0.00201	-0.00088	-0.00072	-0.00201
1.0		0.114866	-0.12656	-0.0125	-0.03581	-0.03815	-0.03391	-0.04146	-0.04611	-0.03211	-0.03103	-0.04792
1.5		0.114081	-0.12569	-0.01241	-0.03556	-0.03789	-0.03367	-0.04117	-0.04579	-0.03189	-0.03082	-0.0476
2.0		0.127315	-0.11854	-0.01169	-0.03349	-0.03568	-0.03171	-0.03877	-0.04312	-0.03031	-0.02925	-0.04482
2.5		0.131205	-0.11208	-0.011	-0.03205	-0.03358	-0.03028	-0.03649	-0.04058	-0.02868	-0.02766	-0.04218
3.0		0.134815	-0.10768	-0.01051	-0.03081	-0.03277	-0.02909	-0.03485	-0.03876	-0.02752	-0.02652	-0.04028
3.5		0.137739	-0.10502	-0.0102	-0.03005	-0.03198	-0.02835	-0.03382	-0.03761	-0.02681	-0.02582	-0.0391
4.0		0.140568	-0.10383	-0.01003	-0.02968	-0.03158	-0.02799	-0.03413	-0.037	-0.02644	-0.02546	-0.03846
4.5		0.143208	-0.10339	-0.00997	-0.02957	-0.03148	-0.02787	-0.03401	-0.03677	-0.02633	-0.02534	-0.03822
5.0		0.146174	-0.10359	-0.00997	-0.02964	-0.03156	-0.02794	-0.03411	-0.03679	-0.02638	-0.02538	-0.03824

Results

The results of our PCA analysis on the 25 year Forward Data (see Historical Data section) has been plotted below:



I would like to highlight some facts here about the PCA of the yield curve. As we can see from the above graph, the leading component PC1 is relatively flat, with all elements positive. PC1 being the leading component, will cause roughly a parallel move of the yields in response to a random shock. The second principal component PC2 tilts: its elements change from positive values to negative values. Given a positive shock, short maturity yields will increase, while long maturity yields will decrease. The third principal component PC3 bends, it starts in negative territory and ends up in positive territory. A positive shock to this will move both short and long–term yields lower, yet such shock will move the median-term yields higher, thus “bending” the yield curve. Knowing that the aggregated weight of the first three eigenvalues is 93%, we may say that deformation of the forward-rate curve is caused by parallel shifts, tilting, and bending, corresponding to the shapes of the three leading eigenvectors

HJM Model

The multi-factor HJM model for the risk neutral random walk for the forward curve is:

$$d\bar{F}(t,\tau) = \bar{m}(t,\tau)dt + \sum_{i=1}^{N} \bar{v}_i(t,\tau)dX$$

where

$$\bar{m}(t,\tau) = \frac{\partial}{\partial\tau}\bar{F}(t,\tau) + \sum_{i=1}^{N} \bar{v}_i(t,\tau)\int_t^\tau \bar{v}_i(t,s)\,ds$$

The **Mu** and **Volatility** is explained in detail in the below sections.

Key features of HJM Model are:

- It relies on simulation and modelling the entire forward curve and not just the spot rate. This makes calibration pretty much automatic.
- In a risk neutral framework, $dF(t,\tau)$ in above equation provides the evolution of a continuous instantaneous forward curve.
- HJM is non-Markov model due to the persistence of dX involved in the volatility parameters.

The above equation is a version that already factors in Musiela Parameterization. What this implies is that model for the volatility of the curve will be of the form;

$$V(t,T) = V(t,T-t)$$

Meaning we would model volatility at each maturity and not at each maturity dates. This is the reason why there is a rolling adjustment to the forward curve which is represented by the following in the above equation:

$$\frac{\partial}{\partial\tau}\bar{F}(t,\tau)$$

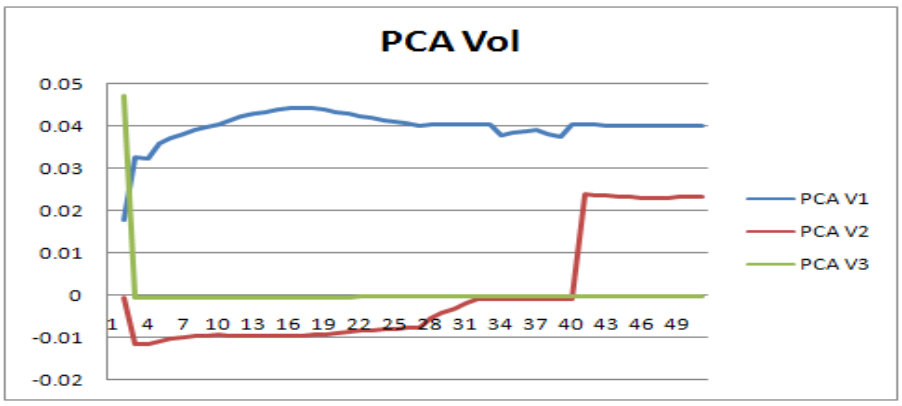
For pricing with the HJM model, Monte Carlo simulation is the prefered framework discussed in this paper.

Volatilities

The volatility (V_i) is one of the most important parameter in the HJM pricing framework. This is because it determines how the forward curve is going to be simulated under the risk-neutral process. In this context, since we are in a multi-factor HJM environment, there are multiple volatility factors which inflict its effect upon the simulated forward curve.

Deriving the volatilities

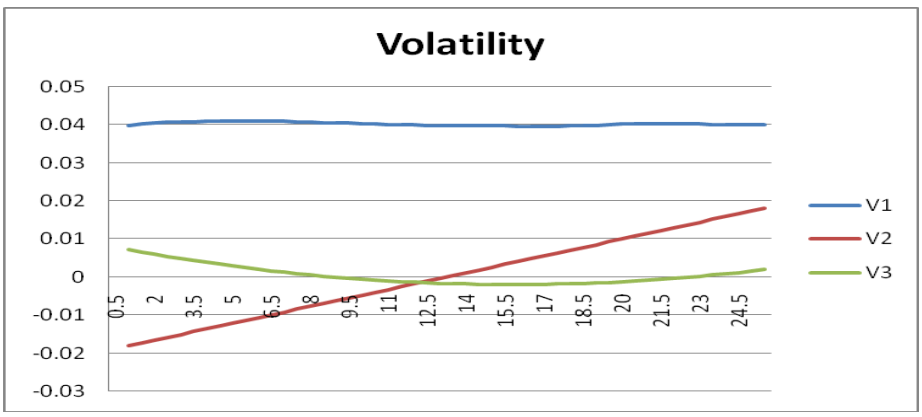
The output of PCA analysis provides us with top three principal components plotted bellow.



We use the top three principal components (PC) to derive the respective volatilities.

$$V_{i,j} = EigenVector_{i,j} * \sqrt{EigenValue_{i,j}}$$

Applying the above equation results us with the vector of volatilities that are used in HJM pricing. We also typically normalize the PC's before deriving volatilities to offset any specific data adjustment that was done during PCA analysis.



Mu

$$\bar{m}(t,\tau) = \frac{\partial}{\partial\tau}\bar{F}(t,\tau) + \sum_{i=1}^{N}\bar{v}_i(t,\tau)\int_t^{\tau}\bar{v}_i(t,s)\,ds$$

The Mu parameter is the deterministic piece of the $dF(t,\tau)$ in the HJM model. The above equation has the Musiela parameterization and thus the extra parameter of $\frac{\partial}{\partial\tau}\bar{F}(t,\tau)$ is prefixed to emphasis the rolling down the yield curve effect.

A simple implementation of Mu is shown below.

```
Function m(tau As Double, vl1, vl2, vl3)
   Dim dtau, n, m1, m2, m3, i As Variant

   If tau = 0 Then
      m = 0
   Else
      dtau = 0.01
      n = Int(tau / dtau)
      dtau = tau / n

      m1 = 0.5 * vol1(vl1)
      For i = 1 To n - 1
         m1 = m1 + vol1(vl1)
      Next i

      m1 = m1 + 0.5 * vol1(vl1)
      m1 = m1 * dtau
      m1 = vol1(vl1) * m1
      m2 = 0.5 * vol2(vl2)

      For i = 1 To n - 1
         m2 = m2 + vol2(vl2)
      Next i

      m2 = m2 + 0.5 * vol2(vl2)
      m2 = m2 * dtau
      m2 = vol2(vl2) * m2
      m3 = 0.5 * vol3(vl3)

      For i = 1 To n - 1
         m3 = m3 + vol3(vl3)
      Next i

      m3 = m3 + 0.5 * vol3(vl3)
      m3 = m3 * dtau
      m3 = vol3(vl3) * m3
      m = m1 + m2 + m3
   End If
End Function
```

Simulating HJM

To simulate curves for various forward time-intervals, the following equations are used.

$$d\bar{F}(t,\tau) = \bar{m}(t,\tau)dt + \sum_{i=1}^{N} \bar{v}_i(t,\tau)dX$$

where

$$\bar{m}(t,\tau) = \frac{\partial}{\partial\tau}\bar{F}(t,\tau) + \sum_{i=1}^{N} \bar{v}_i(t,\tau)\int_t^{\tau} \bar{v}_i(t,s)\,ds$$

We could visualize the whole thing as a matrix with columns as various forward periods (1year forward, 3-year forward, etc) and timeline as the rows. And we would be simulating curves for each forward period across different timelines.

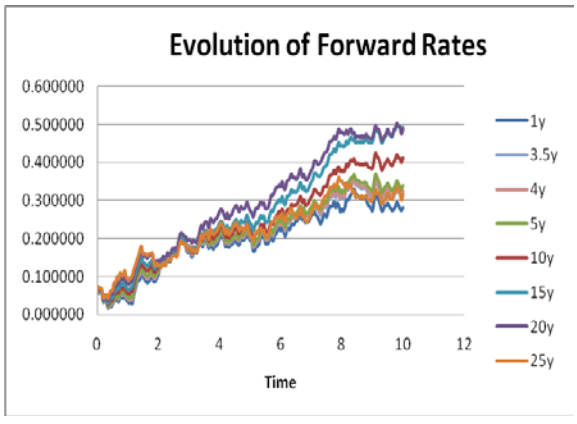
Monte Carlo Simulation (HJM Model)

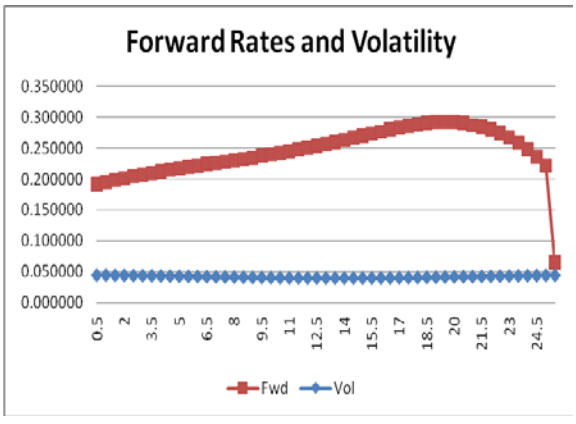
m	0.000987	0.001975	0.002932	0.003876	0.004794	0.005691
v1	0.039898	0.040347	0.040509	0.040683	0.040782	0.040862
v2	-0.0181	-0.01736	-0.01662	-0.01588	-0.01514	-0.0144
v3	0.007349	0.00673	0.006132	0.005555	0.005	0.004465
Vol	0.044425	0.044437	0.044214	0.044024	0.043789	0.043555

TimeStep 0.05

Maturities and Forward Rates

Time	*0.5*	*1*	*1.5*	*2*	*2.5*	*3*
0	0.064382	0.064382	0.064382	0.064382	0.064382	0.064382
0.05	0.064843	0.065170	0.065464	0.065750	0.066019	0.066277
0.1	0.057299	0.057661	0.058035	0.058398	0.058756	0.059106
0.15	0.045351	0.046288	0.047239	0.048168	0.049085	0.049983
0.2	0.049928	0.050867	0.051795	0.052703	0.053593	0.054464
0.25	0.048425	0.049584	0.050719	0.051831	0.052917	0.053979
0.3	0.059402	0.060510	0.061543	0.062557	0.063532	0.064480
0.35	0.054390	0.055352	0.056296	0.057218	0.058117	0.058992
0.4	0.043524	0.044526	0.045569	0.046589	0.047601	0.048597





Pricing -Caps & Floors

Concept

An interest rate cap is a derivative product with an interest rate as the underlying. A cap may be extending over one or more time periods. Each time period (called reset period), whose length is agreed by the two parts of the contract, is covered by a caplet. For each caplet, there is a pre-specified interest rate (cap rate). This rate is set in relation to a major interest floating rate index (e.g. Libor). In case where the latter exceeds the cap rate, the buyer of the cap gains the difference. The caplet therefore is providing protection against movements of the rate index over a pre-specified maximum level within a certain period.

Cap's value

The cap value consists of two parts:

- **Intrinsic Value**

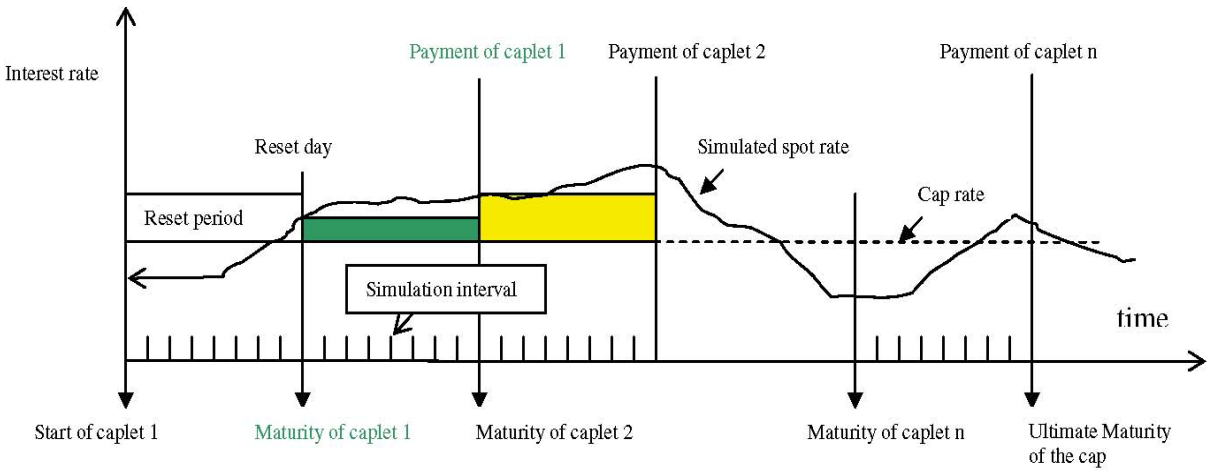
If the cap rate is smaller than the implied forward rate, the cap has intrinsic value. We can get the implied forward rate directly by estimating the forward rate curve from the zero-coupon bond prices in the market. If the cap has positive intrinsic value, the cap is said to be 'in the money'. Where the cap rate is greater than the implied forward rate and where it is equal to the implied forward rate, is said to be 'out of the money' and 'at the money' respectively.

- **Time Value**

As a hedge the cap provides a guarantee that the future rate which the debt-issuer has to pay will not exceed a certain level. The level of the interest rate changes according to the policy followed by the central governments and their understanding of the needs of the economy. Changes in the condition of the markets lead to changes in the interest rate. There are periods (years) with many changes and other, relatively, flatter periods. The volatility of the interest rate is responsible for the time value of the cap. The impact of the volatility on the cap price becomes smaller as time approaches the maturity of the cap. Therefore, in a market with no changes, the passing of time will simply reduce the cap's value.

To aid the explanation of a cap the following figure is displayed

Graphical description of a cap with n caplets



How to price a cap

The price of a cap is the sum of the prices of its caplets. The price of a caplet can be estimated by finding the present value of the expected gain from the caplet.

$$Cap = \sum_{i=0}^{n} Caplet_i$$

And for

$$Floor = \sum_{i=0}^{n} Floorlet_i$$

To find the expected gain we need to find primarily the expected value of the interest rate index at the maturity of the caplet. Then the difference (if positive) between this interest rate index and the cap rate should be multiplied with the principal amount and the reset period. This product should be discounted to find the present value of the caplet.

Caps / Floors as Bond Options

A cap can be viewed as a portfolio of puts on zero-coupon bonds. The payoff at time t_i in the amount of

$$\tau N max(R(t_{i-1}, t_i) - \bar{R}, 0)$$

$R(t_{i-1}, t_i)$ is LIBOR rate,
$\bar{R}$ is the strike price

is equivalent to the payoff at time t_{i-1} :

$$\frac{\tau N}{1 + \tau R(t_{i-1}, t_i)} max(R(t_{i-1}, t_i) - \bar{R}, 0)$$

where

$$\frac{\tau N}{1 + \tau R(t_{i-1}, t_i)}$$

is the discount factor for the time interval $[t_{i-1}, t_i]$, that is, a zero-coupon bond $P(t_{i-1}, t_i)$ paying $1 at time t_{i-1}

$$P(t_{i-1}, t_i) = \frac{\tau N}{1 + \tau R(t_{i-1}, t_i)} = e^{-\int_t^T F(t,s)ds}$$

We can rewrite the caplet payoff as follows

$$\frac{\tau N}{1 + \tau R(t_{i-1}, t_i)} max(R(t_{i-1}, t_i) - \bar{R}, 0) = \frac{N}{\bar{P}} \max(\bar{P} - P(t_{t-1}, t_i), 0)$$

A caplet can be viewed as quantity $\frac{N}{\bar{P}}$ of put option on one-period zero coupon bond $P(t_{i-1}, t_i)$ with strike of

$$\bar{P} = \frac{1}{1 + \tau \bar{R}}$$

If assume the rate $R(t_{i-1}, t_i)$ is lognormally distributed with some known standard deviation of $lnR(t_{i-1}, t_i)$ equal to

$$\sigma(t_i) = \sigma(t_i; t_{i-1}, t_i) = \sigma\sqrt{t_{i-1}}$$

where σ is the common volatility parameter that is retrieved from market quotes at time t, then we can use Black's formula for a caplet C_i that pays at time t_i:

$$C_i = \tau_i N P(t, t_i)(f(t; t_{i-1}, t_i)\Phi(d_+) - \bar{R}\Phi(d_-))$$

where $P(t, t_i)$ is the one-period discount bond, $f(t; t_{i-1}, t_i)$ is the forward LIBOR rate for the period, $\Phi(.)$ is the cumulative normal distribution, and

$$d_{\pm} = \frac{1}{\sigma(t; t_{i-1}, t_i)} \left(\ln\left(\frac{f(t; t_{i-1}, t_i)}{\bar{R}}\right) \pm \frac{\sigma^2(t; t_{i-1}, t_i)}{2} \right)$$

Summary

A Caplet is equivalent to a put option expiring at time t_{i-1}on a bond maturing at time t_i. Similarly, a Floorlet is equivalent to a call option expiring at time t_{i-1}on a bond maturing at time t_i.

Note the forward rate can be expressed through bond prices, which is the case below:

$$P(t,t_i) = P(t,t_{i-1})\frac{1}{1+\ \tau f(t;t_{i-1},t_i)}$$

The psedocode on caplet computation is shown below.

```
Public Function CapletBlack(valPeriod As Variant)
  Dim Niters As Single
  Dim startpeg As Single


  L = Sheets("CapFloor").Range("G8").Value 'principal

  tau = Sheets("CapFloor").Range("G5").Value  'accrual period in Yrs
  Rk = Sheets("CapFloor").Range("G6").Value 'strike rate for caplet

  startpeg = Application.Floor(valPeriod / tau, 1) 'start peg for caplet
  'eg., startpeg=4 implies caplet begins from 4th time period and ends in 5th period
  'discretized by tau

  nF = startpeg + 1
  ReDim F(0 To nF)
  ReDim Fvol(0 To nF)
  Dim rngstr As String
  For i = 0 To nF
    tmp = i + 5
    rngstr = "B" & tmp
    F(i) = Sheets("CapFloor").Range(rngstr).Value
    tmp = i + 6
    rngstr = "C" & tmp
    Fvol(i) = Sheets("CapFloor").Range(rngstr).Value
  Next i

  oldF = F
  originalF = F
  blackvol = 0
  For i = 0 To startpeg - 1
    blackvol = blackvol + Fvol(i) * Fvol(i)
  Next i
  blackvol = (blackvol / startpeg) ^ 0.5

  df = 1
  For i = 0 To startpeg
    df = df / (1 + tau * originalF(i))
  Next i
  Fk = originalF(startpeg)
  oMaturity = startpeg * tau

  d1 = (Log(Fk / Rk) + blackvol * blackvol * oMaturity * 0.5) / (blackvol * oMaturity ^ 0.5)
  'd2 = (Log(Fk / Rk) - blackvol * blackvol * oMaturity * 0.5) / (blackvol * oMaturity ^ 0.5)
  d2 = d1 - (blackvol * oMaturity ^ 0.5)

  blackPrice = L * tau * df * (Fk * Application.NormSDist(d1) - Rk * Application.NormSDist(d2))

  CapletBlack = blackPrice
End Function
```

Application User Guide

This section is a quick summary of the features of the 'Cap/Floor Pricer' program that implements the models discussed in this paper.

The main screen (shown below) has 3 main components:

- Instrument to price
- Input Parameters: such as maturity, strike, etc.
- Input Term Structure for Forward Rate & Volatility calculated by MC and PCA

Screenshot of the main screen

Input Term Structure		
Time	Forward rate	Forward Vol
0	0.014089	0.040000
0.5	0.015089	0.044425
1	0.009745	0.044437
1.5	0.004430	0.044214
2	0.000818	0.044024
2.5	0.005999	0.043789
3	0.011113	0.043555
3.5	0.016160	0.043322
4	0.021144	0.043094
4.5	0.026068	0.042867
5	0.030938	0.042643
5.5	0.035757	0.042416
6	0.040532	0.042186
6.5	0.045268	0.041953
7	0.049975	0.041718
7.5	0.054663	0.041482
8	0.059342	0.041250
8.5	0.064026	0.041023
9	0.068726	0.040808
9.5	0.073457	0.040611
10	0.078229	0.040432

Setlement period (Yrs)	0.5	**Cap/Floor Specifications**
Strike rate	0.014089	
Maturity	3	
Principal	100,000	

Cap	**2,088.37**	**Cap/Floor price output**
Floor	**268.41**	

Compute Price

Steps in pricing a product:

1. Enter input parameters
   a. Maturity
   b. Strike in decimals
   c. Notional of the product
   d. Settlement Interval
   e. Term Structure*
2. Press "Compute Price" to start the computation.

* for obtaining the term structure from the historical data can be found in "Rates" spreadsheet use the "PCA" spreadsheet (below screenshot)

Press the "Start PCA" button to start PCA analysis Start PCA

		0.08039	0.008204	0.002217	0.000786	0.000784	0.00077	0.000769	0.000758	0.000737	0.000714
		PC1	PC2	PC3							
	EigenValues	803.8987	82.03532	22.16621	7.856426	7.844168	7.700445	7.689302	7.584317	7.373517	7.137351
Maturity (Years)											
0.5		0.063128	-0.00488	0.997952	-0.00137	-0.00159	-0.00113	-0.00178	-0.00201	-0.00088	-0.00072
1.0		0.114866	-0.12656	-0.0125	-0.03581	-0.03815	-0.03391	-0.04146	-0.04611	-0.03211	-0.03103
1.5		0.114081	-0.12569	-0.01241	-0.03556	-0.03789	-0.03367	-0.04117	-0.04579	-0.03189	-0.03082
2.0		0.127315	-0.11854	-0.01169	-0.03349	-0.03568	-0.03171	-0.03877	-0.04312	-0.03031	-0.02925
2.5		0.131205	-0.11208	-0.011	-0.03205	-0.03358	-0.03028	-0.03649	-0.04058	-0.02868	-0.02766
3.0		0.134815	-0.10768	-0.01051	-0.03081	-0.03277	-0.02909	-0.03485	-0.03876	-0.02752	-0.02652
3.5		0.137739	-0.10502	-0.0102	-0.03005	-0.03198	-0.02835	-0.03382	-0.03761	-0.02681	-0.02582
4.0		0.140568	-0.10383	-0.01003	-0.02968	-0.03158	-0.02799	-0.03413	-0.037	-0.02644	-0.02546
4.5		0.143208	-0.10339	-0.00997	-0.02957	-0.03148	-0.02787	-0.03401	-0.03677	-0.02633	-0.02534
5.0		0.146174	-0.10359	-0.00997	-0.02964	-0.03156	-0.02794	-0.03411	-0.03679	-0.02638	-0.02538

EigenVectots

Future Improvements

There are a number of improvements that can be done to the above design & implementation. They include:

1 Support for sophisticated interpolation. The two preferred approaches for this are – the monotone convex or the Nelsen Siegel Yield curve smoothing.

2 Support of Control Variate (via Black '76 model) would be ideal for increased accuracy and reduced Monte-Carlo simulation count.

3 Support for Quasi-Monte-Carlo random number generation.

4 The HJM model is very heavily continuous time based while the markets operate on discrete time. Moving in the direction of BGM style of modelling would help attain higher order of reality/practicality to the prices.

5 Building the program in high performance language like C++ and leveraging its features on multi-threading/multi-core could substantially improve performance.

References

Paul Wilmott, Paul Wilmott on Quantitative Finance I/II/II, 2nd Ed, 2006, John Wiley & Sons, Ltd
John Hull, Options, Futures and Other Derivatives, 5th Ed, 2003, Prentice Hall
Lixin Wu, Interest Rate Modeling, Theory and Practice,2009, Chapman & Hall
Sanjay Nawalkha, Gloria Soto, Natalia Beliaeva, Interest Rate Risk Modeling, 2005, John Wiley & Sons, Ltd
Justin London, Modeling Derivatives in C++, 2007, John Wiley & Sons, Ltd