

Mathematical Models and Numerical Methods

Alonso Peña, PhD, CQF

SDA Professor

SDA Bocconi School of Management

alonso.pena@sdabocconi.it

2

Mathematical Models

In the previous chapter, we described the Bento Box template as a methodology for structuring our approach to price financial derivatives. In the context of the Bento Box template, this chapter corresponds to box 2 – mathematical models. Here we review some of the key mathematical models used in the financial derivatives markets today to describe the behavior of the underlying. In particular, the future evolution of the underlying. The following are the examples of these underlyings:

- An equity or stock
- An exchange rate
- An interest rate
- A credit rating

Equity

In the equity asset class, the underlying is the price of a company stock. For instance, the current price of one share of Vodafone PLC (VOD.L) as quoted in the London Stock Exchange (www.londonstockexchange.com) at some particular time. The price could be £2.32 and the time could be 11:33:24 on May 13, 2013.

In mathematical terms, thus, the price of a stock can be represented as a scalar function of the current time t . We will denote this function as $S(t)$. Note that in technical terms, $S(t)$ is a time series, which even though apparently continuous (with $C[0]$ continuity), is in reality discontinuous (subject to jumps). In addition, it is not a well-behaved function, that is, its first derivative does not exist.

We are going to model $S(t)$ as an stochastic variable. And all the constructions that we build around this value, such as the value of the payoff $H(S_t)$ will be in consequence stochastic functions. In this situation, we are required not to use the standard tools of calculus (such as Taylor series, derivatives, Riemann integral), but are instead required to use the tools from stochastic calculus (such as Ito lemma, Radon-Nykodym derivative, Riemann-Stieltjes integral) to advance our modeling.

In this context, the behavior of the variable $S(t)$ can be described by an SDE. In the case of equities, the standard SDE used to describe the behavior of equities is called **GBM**. Under the so-called real-world probability measure P , GBM is formally represented in continuous time as follows:

$$dS = \mu S dt + \sigma S dW^P$$

Equation 1

However, in literature, this representation is not used for the pricing of financial derivatives. It is substituted by the following representation under the risk-neutral measure Q :

$$dS = rS dt + \sigma S dW^Q$$

Equation 2

In the preceding equation, we have substituted the drift μ by the risk-free rate r of interest, σ is the volatility, and dW is the increment of a Wiener process. Equation 2 can be further represented as follows:

$$\frac{dS}{S} = r dt + \sigma dW^Q$$

In the preceding equation, we can identify the term dS/S on the left hand side (**LHS**) of the equation as the return of the equity. Thus, the two terms on the right hand side (**RHS**) of the equation are a "drift term" and a "volatility term". Each of these terms are "scaled" by parameters μ and σ , which are calibrated to current market prices of traded instruments, such as call and put options.

Note that equation 2 is the fundamental equation used to describe the underlyings in the world of financial derivatives.

For the purpose of pricing derivatives, we require to transform equation 2 from continuous time into discrete time to model the behavior of stocks, say every day ($\Delta t = 1$ days, but in finance we always work in annualized terms, so $\Delta t = 1/365 = 0.00274$ years). We can easily approximate equation 2 by using the Euler-Murayama discretization as follows:

$$\Delta S = rS\Delta t + \sigma S\Delta W$$

$$S_{t+1} - S_t = rS_t\Delta t + \sigma S_t\varepsilon_t\sqrt{\Delta t}$$

$$S_{t+1} = S_t + rS_t\Delta t + \sigma S_t\varepsilon_t\sqrt{\Delta t} = S_t \left(1 + r\Delta t + \sigma\varepsilon_t\sqrt{\Delta t} \right)$$

Equation 3

In the preceding equation, we approximate the differential of the Wiener process as the square root of delta t multiplied by a draw from a Gaussian distribution with zero mean and standard deviation 1 ($N(0,1)$). Equation 3 is a linear iterative equation, which we can compute by having a starting value of S_0 for a number of time steps S_1, S_2, \dots, S_N . We only need the values of the parameters r and σ , and a Gaussian random number generator for the value of ε .

In order to compute the draw from the cumulative standard normal distribution, we will use a method called the **Box-Muller** method. The Box Muller method allows us to convert uniform random numbers into Gaussian random numbers. This is very useful because in many computer libraries we can find standard functions to generate random numbers from a uniform distribution (for example, function `rand()` in C) and through the Box Muller method, we can generate the Gaussian draws we need. We will discuss more about this in *Chapter 4, Equity Derivatives in C++*.

For example, imagine that we would like to simulate the behavior of the stock of company ABC for the next four days. The current value of the stock is 100 EUR. The risk-free interest rate stands at 5 percent per annum (**p.a.**), and the volatility is at 30 percent pa. How shall we proceed?

First we construct a time grid for the business days in which we need the values (note that there are 255 business days in a year). These are t_0, t_1, t_2, t_3 , and t_4 . These correspond respectively to Monday, Tuesday, Wednesday, Thursday, and Friday. In finance, we always work in annualized terms and, therefore, these dates correspond to $\Delta t = 1/255$, so $t_0=0, t_1=1/255, t_2=2/255, t_3=3/255$, and $t_4=4/255$.

We then need the grid of stock prices $S(t)$. These are $S(t_0)$, $S(t_1)$, $S(t_2)$, $S(t_3)$, and $S(t_4)$, which we will assign respectively to Monday, Tuesday, Wednesday, Thursday, and Friday that are represented as S_0 , S_1 , S_2 , S_3 , and S_4 . We already know the value of S_0 , which is the initial price (as observed today), that is, $S_0=100$ on Monday.

Before doing that, we will need a vector of draws from the cumulative standard normal distribution as follows:

$$\varepsilon_1 = +0.4423, \varepsilon_2 = -0.1170, \varepsilon_3 = +0.0291, \varepsilon_4 = +0.6872$$

We can then apply equation 3 iteratively to go from the value of Monday S_0 to the value of Tuesday S_1 as follows:

$$S_1 = S_0 \left(1 + r\Delta t + \sigma \varepsilon_1 \sqrt{\Delta t} \right)$$

Alternatively, we can go from the value of Monday S_0 to the value of Tuesday S_1 with the numerical values as follows:

$$S_1 = (100) \left(1 + (0.05)(1/255) + (0.30)(+0.4423)\sqrt{(1/255)} \right) = 102.12$$

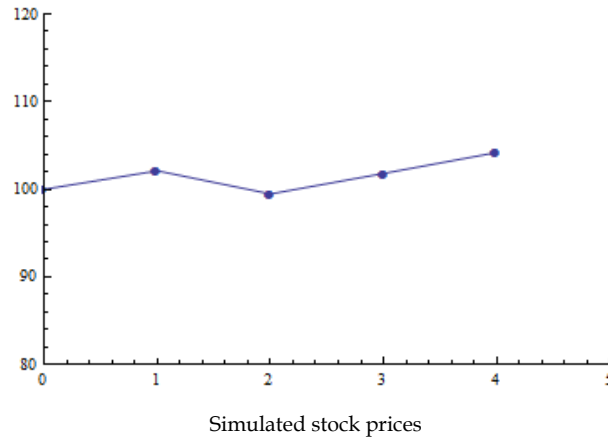
We can then calculate the values for the rest of the days as follows:

$$S_2 = (102.12) \left(1 + (0.05)(1/255) + (0.30)(-0.1170)\sqrt{(1/255)} \right) = 99.47$$

$$S_3 = (99.47) \left(1 + (0.05)(1/255) + (0.30)(+0.0291)\sqrt{(1/255)} \right) = 101.82$$

$$S_4 = (101.82) \left(1 + (0.05)(1/255) + (0.30)(+0.6872)\sqrt{(1/255)} \right) = 104.21$$

If we put together these set of calculated stock prices and plot them against time, we obtain the following graph:



Foreign exchange

In the forex asset class, the underlying is the value of an exchange rate. For example, the current exchange rate between the euro (EUR) and the British pound sterling (GBP) at some particular time. The exchange rate could be $EUR/GBP = 1.31$, meaning that £1 will be exchanged by € 1.31, and the current time could be 11:33:24 on May 13, 2013.

Thus, in mathematical terms, the exchange rate can be represented as a function $X(t)$, which is a scalar function of time, just like in the case of equities. The exchange rate $X(t)$ is thus modeled as an stochastic variable. In mathematical terms, the behavior of $X(t)$ is described using an SDE just like in the case of equities. However, while for equities we used GBM, in the case of forex, we will use a variation that comes from the work of (Garman-Kohlhagen 1983). According to this model, the stochastic differential equation for exchange rates can be expressed as follows:

$$dX = (r_d - r_f)Xdt + \sigma XdW^Q$$

Equation 4

In the preceding equation, r_d and r_f represent the domestic and the foreign risk-free interest rates. The volatility σ is a parameter calibrated to market-quoted instruments.

As we did earlier, before proceeding for the purposes of pricing derivatives, we require to transform equation 1 from continuous time into discrete time to model the behavior of exchange rates say every day ($\Delta t = 1$ day).

We again apply the Euler-Murayama discretization to equation 1 by transforming the differential dX into a difference ΔX , keeping the constants r_d , r_f , and σ unchanged, and approximating the differential of the Wiener process as the square root of delta t multiplied by a draw from a cumulative standard normal distribution, as follows:

$$\Delta X = (r_d - r_f)X\Delta t + \sigma X\Delta W$$

$$X_{t+1} - X_t = (r_d - r_f)X_t\Delta t + \sigma X_t\epsilon_t\sqrt{\Delta t}$$

$$X_{t+1} = X_t + (r_d - r_f)X_t\Delta t + \sigma X_t\epsilon_t\sqrt{\Delta t} = X_t \left(1 + (r_d - r_f)\Delta t + \sigma\epsilon_t\sqrt{\Delta t} \right)$$

Equation 5

As in the case of equation 2, equation 5 is also a linear iterative equation, which we can compute iteratively by having a starting value of X_0 for a number of time steps X_1, X_2, \dots, X_N . We only need the values of the parameters r_d , r_f , and σ and a Gaussian random number generator for the value of ϵ .

For example, imagine that we would like to simulate the behavior of the EUR/USD exchange rate for the next four days at the last quoted value of the business day (known as **end of day (EOD)**). The current value of the exchange rate EUR/USD is 1.33. The domestic risk-free interest rate is 5 percent p.a. and the foreign risk-free interest rate is 3 percent pa, while the volatility is at 30 percent pa. How shall we proceed?

First we construct a time grid for the business days in which we need the values (note that there are 255 business days in a year). These are t_0, t_1, t_2, t_3 , and t_4 . These correspond to Monday, Tuesday, Wednesday, Thursday, and Friday respectively, in turn corresponding to $t_0=0, t_1=1/255, t_2=2/255, t_3=3/255$, and $t_4=4/255$ in annualized terms.

We then need the grid of EOD exchange rates $X(t)$. These are X_0, X_1, X_2, X_3 , and X_4 . We already know the value of X_0 , which is the initial FX rate (as observed today), that is, $X_0=1.33$ on Monday. As we did earlier, before proceeding, we compute a vector of draws from the cumulative standard normal distribution to obtain the following:

$$\epsilon_1 = +0.4423, \epsilon_2 = -0.1170, \epsilon_3 = +0.0291, \epsilon_4 = +0.6872$$

We can then apply equation 3 iteratively to go from the value of Monday X_0 to the value of Tuesday X_1 as follows:

$$X_1 = X_0 \left(1 + (r_d - r_f)\Delta t + \sigma \varepsilon_t \sqrt{\Delta t} \right)$$

Alternatively, we can go from the value of Monday S_0 to the value of Tuesday S_1 with the numerical values as follows:

$$X_1 = (1.33) \left(1 + (0.05 - 0.03)(1/255) + (0.30)(+0.4423)\sqrt{(1/255)} \right) = 1.52$$

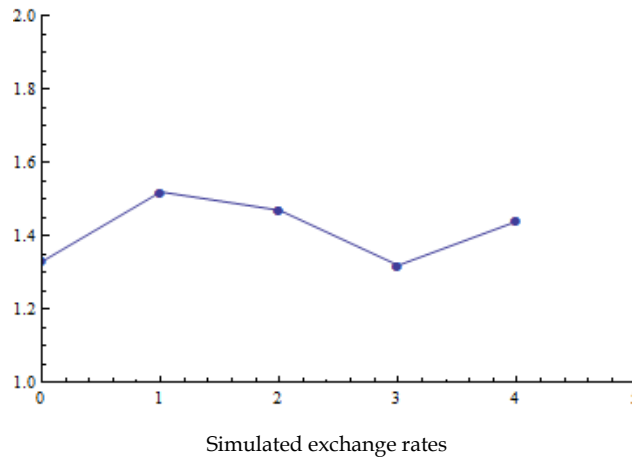
We can then calculate the values for the rest of the days as follows:

$$X_2 = (1.52) \left(1 + (0.05 - 0.03)(1/255) + (0.30)(+0.4423)\sqrt{(1/255)} \right) = 1.47$$

$$X_3 = (1.47) \left(1 + (0.05 - 0.03)(1/255) + (0.30)(+0.4423)\sqrt{(1/255)} \right) = 1.32$$

$$X_4 = (1.32) \left(1 + (0.05 - 0.03)(1/255) + (0.30)(+0.4423)\sqrt{(1/255)} \right) = 1.44$$

If we put together these sets of computed rates and plot them as a function of time, we obtain the following graph:



Interest rates

In the interest rate asset class, the underlying is an interest rate. Interest rates are quite complex. We can see this if we consider the question "what is the interest rate today?" The answer is certainly not simply "5 percent pa" because we also need to specify the maturity of the rates (T) we want to know. So interest has one more dimension than objects like equities or forex. While the currently observed equity value is a scalar quantity, that is, a single number, the current interest rate curve is a vector.

For example, let's consider the spot EURIBOR interest rates observed on May 13, 2013 for the maturities of 1 month, 3 months, 6 months, and 12 months (as published by <http://www.euribor-ebf.eu/>). We denote these spot rates quoted by $R(t, T)$ as $R(0, 3M) = \text{EURIBOR } 3M = 1 \text{ percent pa}$, $R(0, 6M) = \text{EURIBOR } 6M = 2 \text{ percent pa}$, $R(0, 9M) = \text{EURIBOR } 9M = 3 \text{ percent pa}$, and $R(0, 12M) = \text{EURIBOR } 12M = 4 \text{ percent pa}$. Note that $t=0$ because we consider May 13, 2013 as the current date.

How would each of these rates evolve into the future? In other words, how can we model $R(t, T)$? We have the following two choices reflecting the two modelling schools present in literature:

- Short rate models
- Market models

The first is the oldest, while the second is more recent.

In the first model, the key modelling variable is an idealization of the interest rate, the so-called short rate. It is an infinitesimal interest rate dr that applies to a very short time interval. To obtain the interest rate that applies to a full period, we ought to add or integrate the effect of all these small interest rates in the period. In the second model, the key modeling variable is an actual quoted or market interest rate, such as LIBOR. That's why these models are called market models in general, and its most famous version is called **Libor Market Model**.

Short rate models

In continuous time, the short rate can be represented by the following SDE developed by Vasicek as follows:

$$dr = (\theta - \lambda r) dt + \sigma dW$$

Equation 6

The preceding equation is a mean reverting process. The parameter θ is the mean reversion level, λ is the speed of the mean reversion, and σ is the volatility. The parameters θ , λ , and σ control how the stochastic process behaves. The value assigned to θ will be long-term interest rate level to which interest rates will tend, while λ will control how fast interest rates return to the long-term mean level. The volatility σ controls the magnitude of the "jumps" of the process. These parameters can be calibrated to market-quoted instruments, such as options on interest rate swaps (known as **swaptions**).

We can approximate the Vasicek process via the Euler-Murayama methodology described previously to obtain a discretized version of the stochastic process, as follows:

$$\Delta r = (\theta - \lambda r_t) \Delta t + \sigma \Delta W$$

$$r_{t+1} - r_t = (\theta - \lambda r_t) \Delta t + \sigma \Delta W$$

$$r_{t+1} = r_t + (\theta - \lambda r_t) \Delta t + \sigma \epsilon_t \sqrt{\Delta t}$$

Equation 7

The preceding equation is a linear iterative equation, which we can compute iteratively by having a starting value of r_0 for a number of time steps r_1, r_2, \dots, r_N . We only need the values of the parameters *theta*, *lambda*, and *sigma*, and a Gaussian random number generator for the value of *epsilon*.

For example, imagine that we would like to simulate the behavior of the short rate of interest for the next four days. The current value of the interest rate is 5 percent. The parameters $\lambda = 1.0$ and $\theta = 2.0$, while the volatility is at 30 percent pa. How shall we proceed?

First we construct a time grid for the days in which we need the values. These are t_0, t_1, t_2, t_3 , and t_4 in order to correspond to Monday (t_0), Tuesday (t_1), Wednesday (t_2), Thursday (t_3), and Friday (t_4), in turn corresponding to $t_0=0, t_1=1/365, t_2=2/365, t_3=3/365$, and $t_4=4/365$ in annualized terms.

We then need the grid for the short rates $r(t)$. These are r_0, r_1, r_2, r_3 , and r_4 . We will assign them to Monday, Tuesday, Wednesday, Thursday and Friday, respectively. As we already know the value of r_0 , which is the initial interest rate (as observed today), we have that $r_0=5$ percent on Monday.

As we did earlier, before proceeding, we compute a vector of draws from the cumulative standard normal distribution to obtain the following:

$$\varepsilon_1 = +0.4423, \varepsilon_2 = -0.1170, \varepsilon_3 = +0.0291, \varepsilon_4 = +0.6872$$

We can then apply equation 3 iteratively to go from the value of Monday r_0 to the value of Tuesday r_1 as follows:

$$r_{t+1} = r_t + (\theta - \lambda r_t) \Delta t + \sigma \varepsilon_t \sqrt{\Delta t}$$

Alternatively, we can go from the value of Monday S_0 to the value of Tuesday S_1 with the numerical values as follows:

$$r_1 = r_0 + (\theta - \lambda r_0) \Delta t + \sigma \varepsilon_1 \sqrt{\Delta t}$$

$$r_1 = (0.05) + ((1) - (2)(0.05))(1/365) + (0.30)(0.4427)\sqrt{(1/365)}$$

We can then calculate the values for the rest of the days as follows:

$$r_2 = (0.05) + ((1) - (2)(0.05))(1/365) + (0.30)(0.4427)\sqrt{(1/365)}$$

$$r_3 = (0.05) + ((1) - (2)(0.05))(1/365) + (0.30)(0.4427)\sqrt{(1/365)}$$

$$r_4 = (0.05) + ((1) - (2)(0.05))(1/365) + (0.30)(0.4427)\sqrt{(1/365)}$$

Market models

Libor Market Model (LMM) is an advanced mathematical model used to price interest rate derivatives. Also known as the BGM model after its authors (Brace, Gatarek, Musiela, 1997), the LMM has become hegemonic in the financial markets worldwide. Literature offers a wide range of publications about the LMM, mostly its many variants and its complex advanced issues.

The LMM in reality can be understood not as a single model, but rather as a large family of models (Rebonato 1998) and (Brigo and Mercurio 2006). Its many variants include the number of factors considered, the type of volatility modeling used, the type of correlation modeling used, whether stochastic volatility or SABR are used, whether forward LIBOR rates or swap rates are used, and whether semi-analytical or numerical solution methods are used, among others.

Our methodology and notation closely follows that of (Pelsser 2000), which, even though succinct, provides a clear introduction to the LMM. From all the possible variations of the LMM, in this work, we chose the simplest implementation—embodied in the use of lognormal SDEs (GBM) for the forward rates and a single Wiener process driving the volatility in all rates (that is, a one factor case). Under these conditions, we further explore the use of flat volatility.

We first divide the term structure of interest rates N in a set of forward rates L and a set of reset times T as follows:

$$L_1(t), L_2(t), L_3(t), \dots, L_N(t) \quad \text{and} \quad T_0, T_1, T_2, \dots, T_N$$

Each of the preceding forward rates will have its own stochastic process driving them, which will result in N stochastic processes. Following BGM, we use Geometric Brownian Motion (BGM) following (Brace, Gatarek, and Musiela 1997) to describe each of these stochastic processes as follows:

$$dL_i = \mu_i L_i dt + \sigma_i L_i dW_i^Q \quad i = 1, N$$

Equation 8

We now further simplify the model and use a single factor driving all the forward rates. This simplification can be later relaxed into a multifactor LMM. It can be shown that by choosing the last rate as a terminal measure, the drift has the form as follows:

$$dL_i = - \sum_{k=i+1}^N \frac{\alpha_k \sigma_k(T_n) L_k(T_n)}{1 + \alpha_k L_k(T_n)} \sigma_i(t) L_i(T_n) dT + \sigma_i(T_n) dW$$

Note that the drift in the preceding GBM equation is a function of the forward rate, thus not constant but state-dependent.

Given the complexity of the processes in the LMM, it is not possible to obtain a closed-form solution for all the forward rates. This is not a problem, however, as robust numerical methods can be used to solve the discretized version of the forward rate. One important method that is widely used for market models is **Monte Carlo simulation**. The forward rates $L_i(T_n)$ are the realizations of the spot LIBOR rates. In each column, the forward rate $L_i(T_{n+1})$ is updated using the following discretization:

$$L_i(T_{n+1}) = L_i(T_n) - \sum_{k=i+1}^N \frac{\alpha_k \sigma_k(T_n) L_k(T_n)}{1 + \alpha_k L_k(T_n)} \sigma_i(t) L_i(T_n) (T_{n+1} - T_n) + \sigma_i(T_n) (W^{N+1}(T_{n+1}) - W^{N+1}(T_n))$$

Equation 9

The preceding equation, like the ones we have shown for equities and forex, ought to be solved iteratively using simulation. The forward rates can be arranged in an arithmetic table as follows:

$L_1(T_0)$			
$L_2(T_0)$	$L_2(T_1)$		
$L_3(T_0)$	$L_3(T_1)$	$L_3(T_2)$	
$L_4(T_0)$	$L_4(T_1)$	$L_4(T_2)$	$L_4(T_3)$

In the preceding table, the left column represents the term structure of interest rates at time $t=0$. Given a set of LIBOR rates realized on this for T_1, T_2, \dots, T_N , we can extract the future rates that we need for simulation as the ones present in the main diagonal of the preceding table.

For example, consider the following paying fixed-for-floating **Interest Rate Swap (IRS)** with a notional of 1 million EUR. This IRS pays 5 percent every 3 months and receives EURIBOR 3 months' rate every 3 months. The total maturity of the swap is one year, given the following current term structure of interest rates:

$L_1(0)$			
$L_2(0)$	$L_2(3M)$		
$L_3(0)$	$L_3(3M)$	$L_3(6M)$	
$L_4(0)$	$L_4(3M)$	$L_4(6M)$	$L_4(9M)$

Compute the present value of this swap using the LMM. How shall we proceed?

Term structure of spot rates of interest at $t=0$ are *EURIBOR 3M = 1 percent pa*, *EURIBOR 6M = 2 percent pa*, *EURIBOR 9M = 3 percent pa*, and *EURIBOR 12M = 4 percent pa*.

First, we need to compute the initial forward rates by bootstrapping as observed from time $t=0$. We obtain this using the following equation:

$$L_1(0) = 1\%, L_2(0) = 3\%, L_3(0) = 5\%, L_4(0) = 7\%$$

Then using the iterative equation 9, we compute the forward the rates into the future as follows:

$L_1(0)$					1.0%				
$L_2(0)$	$L_2(3M)$				3.0%	2.2%			
$L_3(0)$	$L_3(3M)$	$L_3(6M)$			5.0%	4.3%	5.2%		
$L_4(0)$	$L_4(3M)$	$L_4(6M)$	$L_4(9M)$		7.0%	6.6%	7.2%	6.9%	

corresponds to

We populate the initial forward rates in the left-most column and advance column-by-column to the right until we have all the values we need, as shown on the right. For more details on the calculation, see the book by (Pelsser 2000). Note that even though an IRS can be priced "statically" (that is, without simulation), we use this example to give an idea of what are the steps that the LMM method requires for calculation.

Credit

In credit derivatives modeling, the underlying is credit risk. Modern methodologies of credit risk measurement can be grouped into two alternative approaches – the structural approach pioneered by (Merton 1974) and a reduced form approach utilizing intensity based models to estimate stochastic hazard rates, pioneered by various authors, including (Jarrow and Turnbull 1995), (Jarrow, Lando, and Turnbull 1997), and (Duffie and Singleton 1999).

Structural models

The structural approach to credit risk assumes that a firm defaults when the market value of its assets is less than the obligations or debt it has to pay. Structural models are, therefore, sometimes also referred to as **asset value models**. These models look at a company's balance sheet and its capital structure to assess its creditworthiness. However, one of the key problems with this approach is that the value of a company's assets is hard to observe directly. The annual report only provides an accounting version of the company's real assets and not their market value. For public companies, the equity is normally observable, as is its debt. (Merton 1974) starts with the assumption of an extremely simplified capital structure of the following form:

$$V(t) = E(t) + D(t)$$

Equation 10

In the preceding equation, V represents the value of the firm (the total of the assets of the firm), while E is its equity and D its debt. The equity E is understood as the total value of the equity of the firm, which is equal to the market value of a share (stock) multiplied by the number of shares in the market. For this simplified company, the debt is represented by a single zero coupon bond with maturity T .

At this point Merton asks the question "for a company with the preceding capital structure, when will it go in default?" Well, depends on our definition of default. If we take as default the fact that the company cannot pay its obligations at some specific future time T , then this condition will be satisfied if the value of the company at time T , that is, $V(T)$ is larger than the face value of debt $D(T)$. At this moment in time, the bond holders will request payment and the company will be in position to cover it. On the contrary, if at maturity, the value of the firm is less than the value of the debt it has to pay, it, therefore, will not be able to honor its obligations and will be in default. These two scenarios can be defined mathematically as follows:

if $V(T) > D(T)$ then no default

if $V(T) < D(T)$ then default

What about equity holders? They are in possession of the company's stock in the end. Considering the preceding two scenarios, we then know that if the company goes in default, they receive nothing, while if the company continues to operate, they receive the difference between $V(T)$ and $D(T)$ at maturity, thus giving us the following equation:

$$\left. \begin{array}{l} \text{if } V(T) > D \text{ then } E(T) = V(T) - D \\ \text{if } V(T) < D \text{ then } E(T) = 0 \end{array} \right\} E(T) = \max(V(T) - D, 0)$$

Note that the expression on the left can be written succinctly as the single expression on the right. The expression on the right-hand side represents the payoff to equity holders at maturity. Moreover, this expression has exactly the same form as the payoff of a European Call option with the underlying $V(t)$ and strike D . Following Merton, we further assume that the dynamics of the firm follow a GBM as follows:

$$dV = rVdt + \sigma VdW^Q$$

Equation 11

And by doing this, we establish a bridge between credit risk and the pricing of equity derivatives. In fact, we can now use all the results from pricing equity derivatives to price structural models of credit risk. Note that the volatility σ is the firm's assets volatility and not the equity volatility. The value of the assets of the firm at time $t=0$ can therefore be calculated using the Black-Scholes formula as follows:

$$E_0 = V_0 N(d_1) - D \exp(-rT) N(d_2)$$

Equation 12

$$\text{with } d_1 = \frac{\ln(V_0 / D) + (r + 0.5\sigma^2)T}{\sigma\sqrt{T}} \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{T}$$

In equation 12, $N()$ is the cumulative standard normal distribution. This is a useful expression when we need to calculate problems like **Initial Public Offer (IPO)** of a firm and determine, based on the characteristics of a firm, what should be the fair price of its equity.

For example, imagine that we have a firm that has a total value of its assets of 100 million USD. The risk-free interest rate is 5 percent. The volatility of the firm's assets is assumed to be 20 percent. And the face value of its debt stands at 70 million USD payable as a zero coupon bond in four years. What should be the fair value of its equity at time $t=0$?

We use the framework of structural models of credit risk and proceed as follows. The parameters are $V_0 = 100,000,000$, $D = 70,000,000$, $T = 4$, $\sigma = 20\%$, and $r = 5\%$. Using the preceding formula we obtain the following equation:

$$d_1 = \frac{\ln(100 / 70) + (0.05 + 0.5 \times 0.2^2) 4}{0.2\sqrt{4}} = 1.592$$

$$d_2 = 1.592 - 0.20 \times \sqrt{4} = 1.192$$

$$N(1.592) = 0.944, \quad N(1.192) = 0.883$$

$$E_0 = 100 \times 0.944 - 70 \exp(-0.05 \times 4) \times 0.883 = 43.79$$

Intensity models

The default is a random variable tau (τ) that denotes the time in which the company will go in default. Default can be described as bankruptcy, lack of payment, and so on. To model the arrival risk of a credit event, we need to model an unknown random point in time τ .

Intensity-based models focus directly on describing the conditional probability of default without the definition of the exact default event. Intensity models are based on the concept of survival probability. It is an idea borrowed from actuarial and biological sciences. Survival probability is a decaying exponential function, which describes the probability of how long a firm or a country (sovereign) will survive. This can be expressed mathematically in the simplest terms as follows:

$$P(t, T) = \exp[-\lambda(T - t)]$$

Equation 13

But we want to model default, not survival. In fact, we can compute the probability of default in some future period $[S, T]$ as shown from the current time t , simply as the difference between two consecutive survival probabilities. The probability of default PD in that future period then can be calculated using the following formula:

$$PD(t, S, T) = \exp[-\lambda(S-t)] - \exp[-\lambda(T-t)]$$

Equation 14

The preceding formulas are fundamental for the modeling of credit risk. With them, we can describe the likelihood of default, which is a probabilistic event that will happen in the future. The cashflows associated with this future event are thus contingent on the occurrence of this event. Following the classical framework of financial derivatives, thus credit derivatives can be understood as contingent claims, in which the event that triggers the future cash flow is default.

To be concrete, consider the simplified situation – we are expecting to receive a single cash flow from a company at some future time T . What is the present value of this cash flow?

If we don't consider credit risk, the **Present Value (PV)** of the cash flow is simply the discounted **Future Value (FV)** of the cash flow. If the discount factor in the intervening time is DF , then we can write the simplest formula in finance, which states the following:

$$PV = DF \times FV$$

We can go one step further and assuming continuous compounding with a constant risk-free rate r , we can obtain the following equation:

$$PV = \exp(-rT) \times FV$$

Equation 15

What happens now if we consider the credit risk of the company? The first thing to notice is that the cash flow is not any more certain, but it is uncertain. In mathematical terms, it is not any more deterministic, but it is probabilistic. Therefore, we ought to rewrite the previous equation. But because the future cash flow is unknown, it is in fact a random variable. We ought to write it as an expected value of this random variable as follows:

$$PV = \exp(-rT) \times E[FV]$$

This expectation thus introduces a probability of FV happening or not. But what is this probability? The probability that the company will be there in the future to make the payment. In other words, the probability that the company has survived until then, in other words its survival probability. So we can substitute $E[FV]$ with $FV \times P(0, T)$ in order to obtain the following formula:

$$PV = \exp(-rT) \times FV \times P(0, T)$$

Furthermore, we can explicitly represent the survival probability described previously as follows:

$$PV = \exp(-rT) \times FV \times \exp(-\lambda T)$$

We will receive the future cash flow only if the company is present then in the future (or has survived) to do it. The final expression for the present value of the cashflow, taking into account credit risk, is as follows:

$$PV = \exp(-(r + \lambda)T) \times FV$$

Equation 16

This last expression is interesting as it shows that credit risk is a form of "spread" that is added to the risk free rate that we normally use to discount future cashflows.

For example, imagine that we expect to receive 1 million USD from a counterparty that has a credit risk, quantified by its hazard rate, of $\lambda = 3\%$. Assuming that the risk-free interest rate $r = 5\%$, compute the present value of the cash flow firstly, without credit risk and secondly, with credit risk. Assume continuous time discounting. Assume that the recovery rate is $R = 100\%$. How shall we proceed?

By applying the preceding formulas, we obtain that without credit risk, the PV is as follows:

$$PV = \exp(-rT) \times FV$$

$$PV = \exp(-(0.05)(1)) \times (1,000,000) = 950,000$$

Now, if we consider the credit risk, we have the PV as follows:

$$PV = \exp(-(r + \lambda)T) \times FV$$

$$PV = \exp(-(0.05 + 0.03)(1)) \times (1,000,000) = 880,000$$

Note that the second cash flow is smaller than the first. The effect of the credit risk is to view the value of the cash flow today (its PV) as reduced because not being certain, it has to be multiplied by the chance that it happens (that is, its survival probability).

In the previous analysis, we have assumed that when default occurs, all of the future cash flow FV will be lost. However, in most real situations, some fraction of money is recovered. If we define this fraction as the recovery rate R , the present value PV of the risky future cash flow is still the same, as follows:

$$PV = \exp(-rT) \times E[FV]$$

But the expectation is now resolved in terms of two possible states, default or no default, and each of them is multiplied by its respective probabilities, as follows:

$$E[FV] = \underbrace{FV \times P(0, T)}_{\text{NO DEFAULT}} + \underbrace{FV \times R \times (1 - P(0, T))}_{\text{DEFAULT}}$$

In the preceding equation, we have taken into account the recovery rate R in the case of default. The present value of the risky cash flow PV is now as follows:

$$PV = \exp(-rT) \times FV \times (P(0, T) + R(1 - P(0, T)))$$

Summary

This chapter gave an overview of the fundamental models used to price derivatives in the modern financial markets. We will now take these models as basis to model the underlying of the various asset classes we reviewed and apply a number of numerical methods to implement their calculations efficiently in a computer.

In the next chapter, we will concentrate on numerical methods.

3

Numerical Methods

In the previous chapter, we reviewed some of the key mathematical models used to describe the behavior of the underlying assets of financial derivatives. We saw, in particular, how these models are used to describe the future behavior of these assets based on the information we have today. These models are generally expressed in terms of SDEs and **Partial Differential Equations (PDEs)**.

In this chapter, we are going to describe the three main numerical methods used in the financial markets today in the context of financial derivatives. They are a way to use actual numerical values to the abstract mathematical formulas we saw in the previous chapter. These numerical methods are as follows:

- **Monte Carlo (MC)** simulation
- **Binomial Trees (BT)**
- **Finite Difference Methods (FDM)**

In the context of the Bento Box template, this chapter corresponds to box 3—numerical methods. There is a fourth family of methods, less frequently used, called **quadrature methods**, which are used for numerical integration. These will not be discussed here.

The Monte Carlo simulation method

Monte Carlo simulation is named after the famous casino in the principality of Monaco. It is the most widely used numerical method to price financial derivatives in the industry because of its simplicity, flexibility, and extensibility.

The basic idea of the method is to construct a simulation engine that will allow us to predict a number of possible ways (or trajectories) in which the underlying assets can evolve in the future. These trajectories can be thought of as potential economic or financial scenarios. With MC simulation, we attempt to answer questions such as "given the observed price of Vodafone stock today, what could be the likely prices of the stock each day for the next month?"

As we cannot be certain of the future evolution of prices, our result needs to be based on probability, and, thus, we need large number of samples. Using the stochastic models that we saw in the previous chapter to simulate one possible trajectory, with MC simulation, we are going to simulate many possible trajectories and, for each, compute the payoff that the contract would have had if the prices had followed that specific path in future. Afterwards, we are going to take all these possible payoffs and compute their expected value, that is, the mean or average value. This will give us an estimate of how much this contract will be worth in the future.

MC simulation then allows us to compute the fair price of a financial derivative as its expected discounted payoff. This concept stems from the financial principle of fair pricing, which states that the price that a contract should have if the sum of the cash flows that we expect to receive are the same as the sum of the cash flows that we expect to pay. For more details on MC simulation, you are invited to refer to *Monte Carlo Methods in Financial Engineering*.

In order to have an intuition of why this is the case, consider the following simple example:

Imagine that you have bought a plain vanilla European Call option contract at time $t=0$. This contract will give you a payoff of $H(S_T) = \max(S_T - K, 0)$ at maturity $t=T$. Because the value of the underlying at maturity is uncertain, that is, S_T is a random variable, the payoff function $H(S_T)$ is also uncertain. We can write that the expected value of the payoff function $H(S_T)$ is the expectation $E[H(S_T)]$. In addition, in a European Call contract, we pay a premium today in order to have the right to exercise the option or not at maturity $t=T$. How much should we pay for the premium for this contract today?

As we said before, in a fair value setting, what we expect to receive should be equal to what we expect to pay. By putting all these cashflows together (positive indicating to be received, negative to be paid) we can write the following:

- Amount paid at $t=0$ is written as $-\pi$
- Amount to be received at $t=T$ is written as $+E[H(S_T)]$

If we now compute the present value of these cash flows, we get the following equation:

$$-\pi + DF_T \times E[H(S_T)] = 0$$

In other words, it can be summarized as follows:

$$\pi = DF_T \times E[H(S_T)]$$

The object of the MC simulation method is precisely to help us compute the expectation of the payoff $E[H(S_T)]$; once you compute this, discount this value to obtain the premium of the derivative.

This same idea can be generalized to more complex settings with many complex payoffs and underlyings.

Algorithm of the MC method

For European-type derivatives, MC simulation is composed of the following three steps:

1. The first step is to generate trajectories.

Simulate M trajectories for the evolution of the underlying from $t=0$ to maturity $t=T$. In this step, we use the discretized version of SDE that describes the evolution of the underlying. In our case, we use GBM as SDE, which will allow us to take the value of the stock from its current value S_0 to the value at maturity $t=T$. A discretized version is essentially an approximate version applicable to finite time steps rather than continuous time steps. For more details, please refer to *Monte Carlo Methods in Financial Engineering*. We discretize the life of the option contract in N steps, each of size dt , which can be succinctly written as follows:

$$\{S_i^j\} \quad i = 0 \dots N \quad j = 1 \dots M$$

At the end of this step, we should have a vector of M values for S_T , as follows:

$$\{S_T^i\} \quad i = 1 \dots M$$

These represent a set of possible scenarios for the value of the underlying S at time $t=T$. We use GBM to generate multiple paths that will serve a prediction of where the value of S_T will be at maturity.

2. The next step is to compute the expectation.

Once we have the set of values of the underlying at maturity, we now need to compute the expectation of the payoff at maturity. So we take each of these values and compute the payoff for each value as follows:

$$H(S_T^i) \quad i = 1 \dots M$$

The preceding equation will give us a vector of payoffs. In order to compute the expectation, we need to simply take the average of the payoffs as follows:

$$E[H(S_T^i)] = \frac{1}{M} \sum_{i=1}^M H(S_T^i)$$

3. Now discount the expectation to the present.

The final step is to discount the value of the payoff from maturity to the present time. In order to do this, we will use the following formula:

$$\pi = DF_T \times E[H(S_T)]$$

Alternatively, we can also use continuous compounding, as follows:

$$\pi = \exp(-rT) \times E[H(S_T)]$$

The preceding equation will give us the value of the derivative π . Note that in this case, we have assumed that there is no correlation between the interest rates and the price of stock. That is why we can neatly separate the two effects in the preceding equation. If the interest rates and the price of stock were correlated, then we will not be able to separate the discount factor and the expectation. This no-correlation assumption is standard for simple pricing models.

Example of the MC method

Consider the example where we would like to price a six-month European Call option on Vodafone equity (VOD.L). The current equity price of Vodafone is £100.00, with a volatility of 20 percent and a strike of £100. We assume that the stock pays no dividends. The current risk-free rate is 5 percent pa. How do we proceed to solve this problem using MC simulation? We proceed using the following three phases:

1. The first step is to generate trajectories.

We apply GBM to simulate the value of VOD.L stock from the spot price today $S_0 = £100.00$. For simplicity, we choose to discretize the life of the option from $t=[0,T]$ into $N=5$ time steps and to do $M=5$ simulations using GBM in discrete terms, as follows:

$$S_{i+1} = S_i \left(1 + r\Delta t + \sigma \varepsilon_i \sqrt{\Delta t} \right)$$

The five trajectories will thus be as follows:

$$S_0^1 \rightarrow S_1^1 \rightarrow S_2^1 \rightarrow S_3^1 \rightarrow S_4^1 \rightarrow S_5^1$$

$$S_0^2 \rightarrow S_1^2 \rightarrow S_2^2 \rightarrow S_3^2 \rightarrow S_4^2 \rightarrow S_5^2$$

$$S_0^3 \rightarrow S_1^3 \rightarrow S_2^3 \rightarrow S_3^3 \rightarrow S_4^3 \rightarrow S_5^3$$

$$S_0^4 \rightarrow S_1^4 \rightarrow S_2^4 \rightarrow S_3^4 \rightarrow S_4^4 \rightarrow S_5^4$$

$$S_0^5 \rightarrow S_1^5 \rightarrow S_2^5 \rightarrow S_3^5 \rightarrow S_4^5 \rightarrow S_5^5$$

The prices of the stock at maturity will be as follows:

$$\{S_4^1 = 103.37, S_4^2 = 94.40, S_4^3 = 105.27, S_4^4 = 108.24, S_4^5 = 117.73\}$$

2. The next step is to compute the expectation.

For each of the values of the underlyings, we now compute the payoffs as follows:

$$\{H(S_5^1), H(S_5^2), H(S_5^3), H(S_5^4), H(S_5^5)\}$$

We now use the specific form of the payoff to describe a European Call option as follows:

$$\{\max(S_4^1 - K, 0), \max(S_4^2 - K, 0), \max(S_4^3 - K, 0), \max(S_4^4 - K, 0), \max(S_4^5 - K, 0)\}$$

We apply the following numbers to the preceding equation to get the following result:

$$\{3.37, 0.00, 5.27, 8.24, 17.73\}$$

The expected value of the preceding calculation is as follows:


$$E[H(S_T^i)] = \frac{1}{5}(3.37 + 0.00 + 5.27 + 8.24 + 17.73) = 6.92$$

3. Now discount the expectation to the present.

We now use the following continuous compounding to discount the expected payoff we just calculated in step 2 in order to determine the value of the premium:

$$\pi = \exp(-rT) \times E[H(S_T)] = \exp(-(0.05)(0.5)) \times (6.92) = 6.75$$

In this example, we have used only five scenarios for our MC price. In practice, hundreds or even thousands of scenarios are required in order to obtain an acceptable error. Clearly, the more scenarios you use, the more accurate the approximation. It is possible to derive some error-bound formulas for the MC method and show the speed of convergence. For more details, the reader is invited to refer to *Monte Carlo Methods in Financial Engineering*. Putting together all the trajectories for the five MC scenarios, we obtain the table shown in the following screenshot. Here, we see that all the trajectories start at $S_0=100$ and lead to some final value S_5 . For each trajectory, we compute the payoff H , which is then averaged to compute its expected value. The result is then discounted to obtain the present value of the derivative.

S0	S1	S2	S3	S4	S5	H(S_T)
100,00	101,06	107,10	115,81	113,49	103,37	3,37
	0,0884	0,8656	1,2070	-0,3961	-1,4890	
100,00	105,81	108,18	101,43	103,05	94,40	0,00
	0,8390	0,2757	-1,0661	0,1735	-1,4057	
100,00	100,17	102,71	99,83	105,65	105,27	5,27
	-0,0515	0,3215	-0,5230	0,8436	-0,1360	
100,00	108,82	113,29	112,32	117,29	108,24	8,24
	1,3157	0,5703	-0,2142	0,6198	-1,2992	
100,00	103,79	102,30	101,87	112,90	117,73	17,73
	0,5200	-0,3055	-0,1464	1,6339	0,5973	
6,75						6,92
				mean		

Example of the Monte Carlo simulation

The Binomial Trees method

Binomial Trees (BT) can be traced to the work of (Cox, Ross, and Rubinstein 1979). Like MC methods, they are based on the idea of how the discretization of stock prices can jump up or down. Unlike the MC methods, BT are not based on simulation of many possible paths, but on the construction of a single path of possible future prices that bifurcates at every node. These prices, as well as their associated probabilities, constitute the tree. Once this tree is built, the prices of the underlying at maturity can be determined, and the the payoff at maturity can be then computed and discounted to the present time in order to determine the premium of the derivative.

Algorithm of the BT method

The BT method when applied to price derivatives is composed of three phases: the construction of the tree of prices (forward phase), the computation of the payoffs (maturity phase), and the discounting of the payoffs to the present time (backward phase). We will now explain the BT method in the simplified context of a two-step BT. This can be easily generalized to an N step tree.

To start with, we assume that the underlying can only go up or down in the next time step. So we specify the up factor u to describe how much the value today changes to a higher value and the down factor d to describe how much the value today changes to a lower value, such that the up value is $S(T) = u S(0)$, and the down value is $S(T) = d S(0)$. Furthermore, refer to "Option pricing: A simplified approach". The formula for the up and down values can be shown as follows:

$$u = \exp(\sigma\sqrt{\Delta t})$$

$$d = \exp(-\sigma\sqrt{\Delta t})$$

The following is the formula for the probabilities of going up:

$$p = \frac{\exp(r\Delta t) - d}{u - d}$$

The probability of going down is $1 - p$. We can now proceed to construct our binomial tree in the following three phases:

1. The first phase is the forward phase.

Here we construct the tree. Like in MC simulation, time is discretized in steps dt from $t=0$ to $t=T$. From one step tp , the next price of the underlying can either go up or down by a factor u or d as shown in the following formulas:

$$\text{At } t = 0 : S_0$$

$$\text{At } t = t_1 : uS_0 \text{ or } dS_0$$

$$\text{At } t = t_2 : u^2S_0, udS_0, d^2S_0$$

Thus, the values of the tree at maturity are as follows:

$$S_T^1 = u^2S_0, S_T^2 = udS_0, S_T^3 = d^2S_0$$

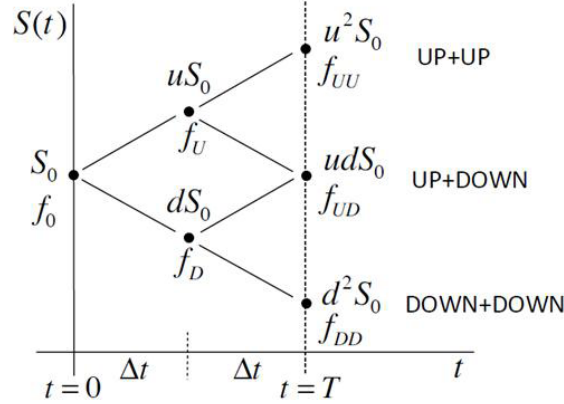
In general case, we proceed in a similar manner until we arrive at the maturity T , and we have $N+1$ values for the variable S . We will calculate these values with the help of the following equation:

$$\{S_T^k\} \quad k = 1 \dots N + 1$$

The preceding equation in our case can be summarized as follows:

$$\{S_T^1, S_T^2, S_T^3\}$$

This entire process is illustrated in the following diagram:



2. The second phase is the payoff phase.

In this phase, we use the values of the underlying at maturity and for each of them, we compute the value of the payoff, as follows:

$$\{H(S_T^k)\} \quad k = 1 \dots N + 1$$

In our case, the equation can be summarized as follows:

$$\{H(S_T^1), H(S_T^2), H(S_T^3)\}$$

The following in turn are the values of the option at maturity T :

$$\{V_T^k = H(S_T^k)\} \quad k = 1 \dots N + 1$$

In our case, the preceding equation can be summarized as follows:

$$\{V_T^1 = H(S_T^1), V_T^2 = H(S_T^2), V_T^3 = H(S_T^3)\}$$

3. The third phase is the backward phase.

In this final phase, we take the values of the payoff at maturity and proceed in a backward manner. We move from the last node to the previous nodes, by computing the option value as the discounted expected payoff in the previous nodes using the weighted probabilities, as follows:

$$V_{T-1}^k = \exp(-r\Delta t) \left[pV_T^k + (1-p)V_T^k \right]$$

In our case, in the second step, the equations are as follows:

$$V_2^1 = \exp(-r\Delta t) \left[pV_3^1 + (1-p)V_3^2 \right]$$

$$V_2^2 = \exp(-r\Delta t) \left[pV_3^2 + (1-p)V_3^3 \right]$$

And, in the first step, the equation is as follows:

$$V_1^1 = \exp(-r\Delta t) \left[pV_2^1 + (1-p)V_2^2 \right]$$

The premium of the derivative, the option price, is the value $\pi = V_1^1$.

Example of the BT method

Consider the example where we would like to price a six-month European Call option on Rolls Royce equity (RR.L). The current equity price of the stock is £100.00, with a volatility of 30 percent p.a. and a strike of £90. We assume the stock pays no dividends. The current risk-free rate is 5 percent pa. How do we proceed to solve this problem using BT?

To start with, we divide the life of the option in two steps, thus $dt=0.25$. The tables in the following screenshot illustrate the numerical values for each of the three steps applied to this problem:

STEP 1				STEP 2				STEP 3			
			134,99				44,99				44,99
		116,18							27,30		
S0	100,00		100,00				10,00	V0	16,04		10,00
		86,07							4,98		
			74,08				0,00				0,00
			ST				H(ST)				
TIME	0,00	0,25	0,50	TIME	0,00	0,25	0,50	TIME	0,00	0,25	0,50

Example of Binomial Trees pricing.

We first compute the up and down factors as well as the up probability p . In numerical terms, these are calculated using the following equations:

$$u = \exp(\sigma\sqrt{\Delta t}) = \exp(0.30\sqrt{0.25}) = 1.16$$

$$d = \exp(-\sigma\sqrt{\Delta t}) = \exp(-0.30\sqrt{0.25}) = 0.86$$

The following are the probabilities of going up and down respectively:

$$p = \frac{\exp(r\Delta t) - d}{u - d} = \frac{\exp(0.05 \times 0.25) - 0.86}{1.16 - 0.86} = 0.50$$

$$(1 - p) = 0.50$$

With all these parameters, we can now proceed to construct our tree in three phases, as follows:

1. The first phase is the forward phase.

We can now construct the two levels of the tree as follows.

$$\text{At } t = 0 : S_0 = 100$$

$$\text{At } t = t_1 : uS_0 = 116.18 \text{ or } dS_0 = 86.07$$

$$\text{At } t = t_2 : u^2S_0 = 134.99, udS_0 = 100, d^2S_0 = 74.08$$

Thus, the values of the tree at maturity are as follows:

$$\{S_T^1 = 134.99, S_T^2 = 100, S_T^3 = 74.08\}$$

2. The second phase is the payoff phase.

In this phase, we use the values of the underlying at maturity, and for each of them, we compute the value of the payoff, as follows:

$$\{H(S_T^k)\} \quad k = 1 \dots N + 1$$

In our case, the equation can be summarized as follows:

$$\{H(S_T^1) = 44.99, H(S_T^2) = 10, H(S_T^3) = 0\}$$

The following in turn are the values of the option at maturity T :

$$\{V_T^k = H(S_T^k)\} \quad k = 1 \dots N + 1$$

In our case, the preceding equation can be summarized as follows:

$$\{V_T^1 = 44.99, V_T^2 = 10, V_T^3 = 0\}$$

3. The third phase is the backward phase.

In this final phase, we take the values of the payoff at maturity and proceed in a backward manner. We move from the last node to the previous nodes, by computing the option value as discounted expected payoff in the previous nodes using the weighted probabilities, as follows:

$$V_{T-1}^k = \exp(-r\Delta t) [pV_T^k + (1-p)V_T^k]$$

In our case, in the second step, the equations are as follows:

$$V_2^1 = \exp(-r\Delta t) [pV_3^1 + (1-p)V_3^2] = 27.30$$

$$V_2^2 = \exp(-r\Delta t) [pV_3^2 + (1-p)V_3^3] = 4.98$$

And in the first step, the equation is as follows:

$$V_1^1 = \exp(-r\Delta t) [pV_2^1 + (1-p)V_2^2] = 16.04$$

The Finite Difference method

The Finite Difference (FD) method is a numerical technique that focuses directly on the approximate solution of a differential equation. As shown by (Black and Scholes 1973) for equity financial derivatives (contingent claims), the problem is expressed in terms of a **Partial Differential Equation (PDE)**.

The basic idea of FDM is to discretize a differential equation. The method transforms the derivatives in the differential equation into quantities or ratios that approximate the derivatives. These quantities are not any more infinitesimal but finite, that is, they have a finite length. This is the origin of the name of finite differences. For more details, the reader can refer to *The Mathematics of Financial Derivatives: A Student Introduction*.

Consider the following illustration where a continuous function $f(X)$ and the first derivative of the function is defined as follows:

$$f'(x) = \frac{df}{dx} \approx \frac{\Delta f}{\Delta x} = \frac{f_{i+1} - f_i}{x_{i+1} - x_i} = \frac{f_{i+1} - f_i}{\Delta x}$$

The preceding function is also known as the **slope**, which is the ratio between the growth (or decrease) in the function with respect to the step size dx . Using the preceding finite difference allows us to calculate the slope of the $f(x)$ function in terms of algebraic quantities.

In quantitative finance, we encounter various types of PDEs. The most important is the Black-Scholes PDE, which is expressed as follows:

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0$$

We now consider solving this equation in a rectangular domain in the S and t axes. In the S axis, the domain is $[a, b]$. In the t axis the domain is $[0, T]$. This can be written mathematically as the domain $\Omega = \{[a, b] \times [0, T]\}$. In the case of a European Call, it has a final condition as follows:

$$V(S, T) = \max(S - K, 0)$$

And the following are the boundary conditions:

$$V(a, t) = 0 \text{ and } V(b, t) = S.$$

Rather than solving the Black-Scholes PDE directly (that is, using variables S and t) we will be following (Wilmott et al. 1995), and we are going to propose a change of variables. This will transform the original PDE into an equivalent PDE, which is easier to solve and in fact is the classical equation of heat diffusion. The change of variables is as follows:

$$S = K \exp(x)$$

$$t = T - \tau / (\frac{1}{2} \sigma^2)$$

The preceding equations transform the Black-Scholes PDE into the classical equation of heat diffusion, as follows:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} \quad -\infty < x < \infty, \quad \tau > 0$$

And the European Call payoff is transformed into the following equation:

$$u(x, 0) = \max \left(\exp\left(\frac{1}{2}(k+1)x\right) - \exp\left(\frac{1}{2}(k-1)x\right), 0 \right)$$

where the parameter k is: $k = r / (\frac{1}{2} \sigma^2)$.

Algorithm of FDM

The application of FDM to the preceding PDE requires the first derivative with respect to time and the second derivative with respect to x , which leads to the following equations:

$$\frac{\partial u}{\partial \tau} \approx \frac{\Delta u}{\Delta \tau} = \frac{u_{i,j+1} - u_{i,j}}{\Delta \tau}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{\Delta}{\Delta x} \left(\frac{\Delta u}{\Delta x} \right) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

The preceding approximations can be derived from a Taylor series expansion. See (Wilmott et al. 1995) as we did in the preceding section.

In order to do this, we need to discretize the domain of the function to a discrete set of nodes. In the case of the BS equation, there will be N division's (or $N+1$) nodes in the spatial dimension and M division's (or $M+1$) nodes in the temporal dimension.

If we now put together our previous approximations, we will obtain the following formula:

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta\tau} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

Solving for the term on the LHS of the preceding equation, we finally obtain the following discretized version of the PDE:

$$u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j}$$

Where in the preceding equation $\alpha = \frac{\Delta\tau}{(\Delta x)^2}$.

The discretized version of the PDE can be solved iteratively in time, using the **explicit** or **forward** finite difference method (FDM) as it's the simplest possible implementation of finite difference techniques for pricing options. We are now ready to follow the next phases to apply the FDM, which are as follows:

1. First, discretize the domain.
Perform this step both in space and time dimensions with time steps $\Delta\tau, \Delta x$.
2. Now approximate each of the derivatives with finite differences.
Just as we have shown in the preceding section, we will apply the principle of transforming the continuous derivatives of the PDE into a finite approximation. This finite approximation will lead to algebraic equations. In literature, this set of equations is called a **stencil**.
3. Next collocate the stencil to all the nodes of the domain.
We now apply the stencil to all the nodes in the domain with the exception of the nodes that represent the initial and boundary conditions. For these nodes, we know the value is a priori, and, hence, it does not need to be computed.

4. Iterate the solution in time with the stencil until we cover the full domain.

In explicit FDM, you simply advance and compute the values for the unknown function u . Note that in other forms of FDM (such as implicit FDM), we need to solve a system of equations via a matrix problem. Please refer to (Wilmott et al. 1995) for further details on implicit methods.

Example of the FD method

Consider the example where we would like to price a six-month European Call option on Barclays equity (BARC.L). The current equity price of BARC is £75, with a volatility of 30 percent p.a. and a strike of £75. We assume the stock pays no dividends. The current risk-free rate is 5 percent pa. How do we proceed to solve this problem using the FDM?

We know that equity financial derivatives satisfy the Black-Scholes PDE when the stock is modelled using GBM. So we solve the heat diffusion equation we described in the previous section. As we did earlier, we apply the following four phases to solve our FDM problem:

1. First discretize the domain.

We divide the domain into N space divisions dS and M time divisions dt , thus $N=5$ and $M=4$. We first apply these values both in space and time dimensions with time steps $\Delta\tau, \Delta x$.

Thus, we obtain six points in time as follows:

$$\tau = \{0.0225, 0.0180, 0.0135, 0.0090, 0.0045, 0.0000\}$$

Five points in space is shown as follows:

$$x = \{-1, -0.5, 0.0, +0.5, +1\}$$

2. Now approximate each of the derivatives with finite differences as follows.

$$u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j}$$

In the preceding equation, $\alpha = 0.018$.

3. Collocate the stencil to all the nodes of the domain.

The following is the initial condition:

$$u(x, 0) = \max \left(\exp\left(\frac{1}{2}(k+1)x\right) - \exp\left(\frac{1}{2}(k-1)x\right), 0 \right)$$

Alternatively, the following is the condition with numerical values:

$$u(x, 0) = \{0.00, 0.00, 0.00, 0.67, 1.82\}$$

In the preceding equation, $k = 1.11$.

The following is the final boundary condition:

$$u(-1, \tau) = 0 \text{ and } u(+1, \tau) = 1.82.$$

4. Iterate the solution in time with the stencil until we cover the full domain.

The following are the internal nodes:

$$u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j}$$

$$i = 2, 3, 4; j = 1 : u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j}$$

$$i = 2, 3, 4; j = 2 : u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j}$$

$$i = 2, 3, 4; j = 3 : u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j}$$

We can arrange the numerical results from our algorithm as shown in the table in the following screenshot, using the transformed variables (upper table) or the original variables (lower table), where we can find that for $S=75$ and $t=0$, the option price is £4,20:

x/tau					
	-1	-0,5	0	0,5	1
0,0225	0,00	0,00	0,06	0,71	1,82
0,0180	0,00	0,00	0,05	0,70	1,82
0,0135	0,00	0,00	0,04	0,69	1,82
0,0090	0,00	0,00	0,02	0,68	1,82
0,0045	0,00	0,00	0,01	0,68	1,82
0,0000	0,00	0,00	0,00	0,67	1,82
s/t					
	27,59	45,49	75,00	123,65	203,87
0,0	0,00	0,15	4,20	50,49	125,93
0,1	0,00	0,09	3,41	50,14	126,56
0,2	0,00	0,05	2,60	49,79	127,20
0,3	0,00	0,02	1,76	49,42	127,84
0,4	0,00	0,00	0,90	49,04	128,48
0,5	0,00	0,00	0,00	48,65	128,87

Example of Finite Difference pricing.

Summary

In this chapter, we reviewed the basics of the three key numerical methods used to price financial derivatives today. For each of them, we have provided an algorithm and a numerical example. Further, more advanced features of these methods can be found in excellent textbooks by (Glasserman 2003), (Kloeden and Platen 1992), and (Wilmott et al. 1995) as mentioned in all the previously discussed sections.

Not all methods are applicable in all situations, just like the tools in a toolbox. Some methods are more effective to solve some specific problems. For example, with a binomial tree, it is simple to evaluate American options also, while for Monte Carlo, it is not so straightforward. Monte Carlo is more powerful in high-dimensional problems, while finite differences can be used effectively for low-dimensional problems.

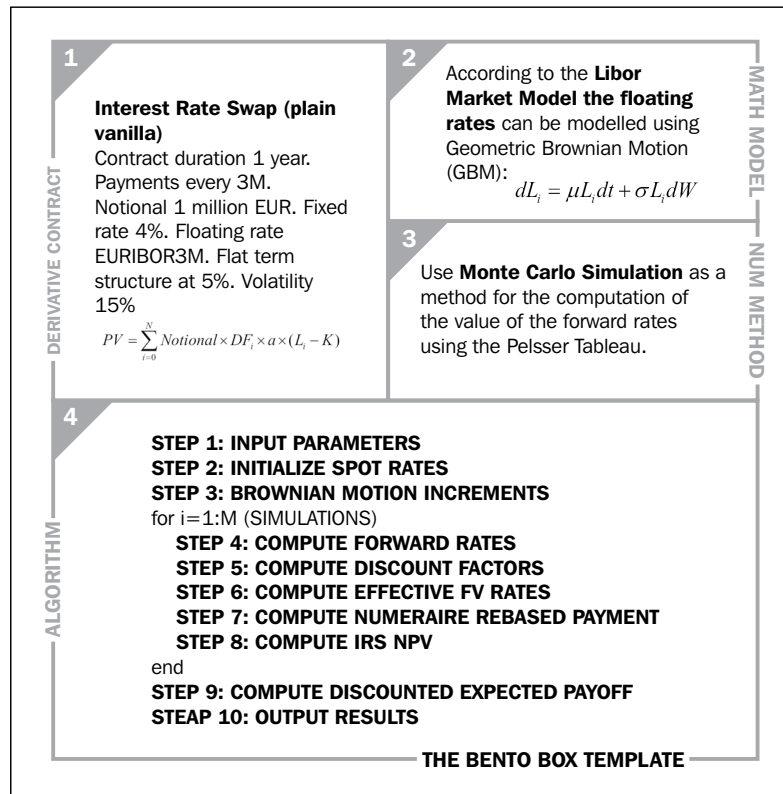
6

Interest Rate Derivatives with C++

This chapter illustrates the application of C++ to the pricing of interest rate derivatives. We will consider two examples: the pricing of a plain vanilla **Interest Rate Swap (IRS)** (basic example) and the pricing of a Cap (advanced example). We provide the full working C++ implementation for both. Both the examples are solved using one factor **Libor Market Model (LMM)** and Monte Carlo simulation. A simpler C implementation (without the OO features) can be found in the accompanying book website. The LMM is described in "The Market Model of Interest Rate Dynamics". An excellent description of the Monte Carlo simulation can be found in "Efficient Methods for Valuing Interest Rate Derivatives".

Basic example – plain vanilla IRS (IR1)

In this example, we will demonstrate the pricing of a plain vanilla IRS. The full details of the contract, including the choice of the mathematical model and its numerical method, are shown in the following Bento Box template:



Bento Box template for basic example (IR1)

Our aim here is to calculate the net present value of this IRS, in particular a paying fixed-for-floating IRS. In this contract, the holder pays the fixed rate and receives the floating rate at regular intervals.

We proceed by completing the contents of the Bento Box in clockwise sense, starting from the top-left corner. First, we will fill all the data of the contract, in particular the payoff function, which in our case is as follows:

$$PV = \sum_{i=0}^N \text{Notional} \times DF_i \times \alpha \times (L_i - K)$$

Equation 1

The present value of the IRS is the sum of the discounted future payments of the IRS. Being a paying IRS, we pay the fixed rate K and we receive the future floating rate L . This rate is fixed (that is, determined) at the beginning of the period and it runs up to the maturity date (when the payment is made). DF_i stands for the respective discount factors. Each payment is multiplied by the notional and day count fraction α .

Second, we ought to select the mathematical model for the underlying. In the case of interest rates, we can choose between short rate models (such as the Vasicek and the Hull and White) or the market models (such as LMM). In this chapter, we select the LMM to solve these problems. Third, we select the numerical method to be used and in this case, we choose the Monte Carlo method. This method will allow us to simulate the random behavior of the forward rates. Fourth, we construct the algorithm that will put together these calculations as a series of calculation steps, which will serve as a blueprint for implementing it in C++.

There are many variations of the LMM, in terms, the number of rates used (multifactor), or the underlying used. (The swap LMM uses the swap rate instead of the forward rate as fundamental unknown.) In this chapter, we will consider only one factor, (lognormal forward) LMM.

The algorithm is shown in box 4 of the Bento Box template. The implementation of the algorithm in C++ is shown in code snippets 16, 17, and 18. Code snippet 16 is the main code block, code snippet 17 is its associated source, while code snippet 18 contains the header file. The algorithm is composed of 10 steps, which take us from the input parameters (**STEP 1**) to the output of the present value of the IRS (**STEP 10**).

Note that Monte Carlo simulation requires a random number generator to operate. We will take advantage of the random number generator, which we developed in *Chapter 3, Numerical Methods*, to price equity derivatives (the Box-Muller algorithm).

We consider an example of a plain vanilla IRS on a notional of one million EUR. The length of the contract is one year and the frequency of payments is every three months. The floating rates are therefore indexed to EURIBOR3M. The fixed rate is 4 percent p.a.

We use LMM with Monte Carlo simulation with 10,000 simulations. We assume an initial flat term structure of interest rates at 5 percent p.a. We also assume a volatility of 15 percent for the forward rates (this value is usually calibrated from observed swaptions in the market).

The upcoming code snippets implement the algorithm from the Bento Box template.

Code 16 – IR1_main.cpp (IRS with Monte Carlo LMM)

The following is the code snippet for IR1_main.cpp file:

```
// IR1_main.cpp
// requires random.cpp IR1_source.cpp

#include "IR.h"
#include <iostream>
using namespace std;

int main()
{
    cout << "\n *** START IR1: IRS Monte Carlo Libor Market Model 1F
        * ** \n\n";

    // Plain Vanilla IRS, pays fixed, receives floating
    // freq payments every 3M, maturity 1 year

    // STEP 1: INPUT PARAMETERS
    double notional = 1000000; // notional
    double K = 0.04; // fixed rate IRS
    double alpha = 0.25; // daycount factor
    double sigma = 0.15; // fwd rates volatility
    double dT = 0.25;
    int N = 3; // number forward rates
    int M = 1000; // number of simulations

    // Construct a IR object from the input parameters:

    IR ir1(notional, K, alpha, sigma, dT, N, M);

    // Obtain the value of premium from member function
    "get_premium()":
```

```

    auto results = ir1.get_simulation_data();

    // STEP 10: OUTPUT RESULTS
    auto sz = results.datapoints.size();
    for (decltype(sz) nsim = 0; nsim < sz; ++nsim)
    {
        cout << "simIRS[" << nsim << "] = " <<
            results.datapoints[nsim] << endl;
    }

    cout << "\n *** IRS PV = " << results.Value << endl;
    cout << "\n *** END IR1: IRS Monte Carlo Libor Market Model 1F
        *** \n";

    return 0;
}

```

Code 17 – IR1_source.cpp (IRS with Monte Carlo LMM)

The following is the code snippet for IR1_source.cpp file:

```

// IR1_source.cpp

#include "IR.h"
#include "random.h"
#include "matrix.h"
#include <algorithm>
#include <iostream>

using namespace std;

IR_results IR::run_LIBOR_simulations() const
{
    matrix<double> L; // forward rates
    matrix_resize(L, N + 1, N + 1);
    matrix<double> D; // discount factors
    matrix_resize(D, N + 2, N + 2);
    vector<double> dW(N + 1); // discount factors
    vector<double> FV(N + 2); // future value payment
    vector<double> FVprime(N + 2); // numeraire-rebased FV payment
    vector<double> V(M); // simulation payoff

    // Composing the SampleBoxMuller class:

    SampleBoxMuller normal;

    double df_prod = 1.0;
    double drift_sum = 0.0;

```

```
double sumPV = 0.0;
double PV = 0.0;

// STEP 2: INITIALISE SPOT RATES
L[0][0] = 0.05;
L[1][0] = 0.05;
L[2][0] = 0.05;
L[3][0] = 0.05;

// start main MC loop

for (int nsim = 0; nsim < M; ++nsim)
{
    // STEP 3: BROWNIAN MOTION INCREMENTS
    dW[1] = sqrt(dT)*normal();
    dW[2] = sqrt(dT)*normal();
    dW[3] = sqrt(dT)*normal();

    // STEP 4: COMPUTE FORWARD RATES TABLEAU
    for (int n = 0; n < N; ++n)
    {
        for (int i = n + 1; i < N + 1; ++i)
        {
            drift_sum = 0.0;
            for (int k = i + 1; k < N + 1; ++k)
            {
                drift_sum += (alpha*sigma*L[k][n]) / (1 +
                    alpha*L[k][n]);
            }
            L[i][n + 1] = L[i][n] * exp((-drift_sum*sigma -
                0.5*sigma*sigma)*dT + sigma*dW[n + 1]); // cout <<"L: i=
                " << i <<"", n+1 = " << n+1 " << L[i][n+1] << "\n";
        }
    }

    // STEP 5: COMPUTE DISCOUNT RATES TABLEAU
    for (int n = 0; n < N + 1; ++n)
    {
        for (int i = n + 1; i < N + 2; ++i)
        {
            df_prod = 1.0;
            for (int k = n; k < i; k++)
            {
                df_prod *= 1 / (1 + alpha*L[k][n]);
            }
            D[i][n] = df_prod;
            // cout <<"D: i = " << i <<"", n = " << n <<"", D[i][n] = "
                << D[i][n] << "\n";
        }
    }
}
```

```

    }
}

// STEP 6: COMPUTE EFFECTIVE FV PAYMENTS
FV[1] = notional*alpha*(L[0][0] - K);
FV[2] = notional*alpha*(L[1][1] - K);
FV[3] = notional*alpha*(L[2][2] - K);
FV[4] = notional*alpha*(L[3][3] - K);

// STEP 7: COMPUTE NUMERAIRE-REBASED PAYMENT
FVprime[1] = FV[1] * D[1][0] / D[4][0];
FVprime[2] = FV[2] * D[2][1] / D[4][1];
FVprime[3] = FV[3] * D[3][2] / D[4][2];
FVprime[4] = FV[4] * D[4][3] / D[4][3];

// STEP 8: COMPUTE IRS NPV

V[nsim] = FVprime[1] * D[1][0] + FVprime[2] * D[2][0] +
    FVprime[3] * D[3][0] + FVprime[4] * D[4][0];
}
// end main MC loop

// STEP 9: COMPUTE DISCOUNTED EXPECTED PAYOFF
sumPV = 0.0;
for (int nsim = 0; nsim < M; nsim++)
{
    sumPV += V[nsim];
}

PV = sumPV / M;

IR_results results(V, PV);

return results;
}

```



For code snippet 18 IR.h, please refer to the code in the code bundle.