tr_lrs1d_zcbond

Technical manuel for the implementation in C language of a lattice pricing model

Marwan Younes ENSTA

Table des matières

1 Introduction

As part of the Personel Project in Laboratory, I have implemented in C language the algorithm describe in the Li, Ritchken, and Sankarasubramanian 1995 article entitled "Lattice Models for Pricing American Interest Rate Claims". The authors use a specific volatility structure in the Hearth-Jarrow-Morton (hereafter HJM) paradigm to price - using a lattice technique - options on interest rate products.

This report describes the source code used in the C program.

2 The mathematical equations

The purpose of this section is not to provide a detailed mathematical study of the model developped by Li, Ritchken, and Sankarasubramanian (hereafter LRS), but to list the main equations they derived, and that I shall use in the algorithm.

2.1 Introduction : calculating the bond price

We want to calculate the price of a zero-coupon bond at date t, with maturity T, in the HJM paradigm. By definition, the price $P(t,T)$ is given by

$$P(t,T) = e^{-\int_t^T f(t,s)ds}$$

where $f(t,s)$ is the forward rate, and represents the interest rate at time s that the market "sees" at time t. The evolution of forward rates for any maturity T is given by a diffusion process of the form :

$$df(t,T) = \mu_f(t,T)dt + \sigma_f(t,T)dw(t),\ f(0,T) \text{ given } \forall\ T \geq t$$

The choice of the volatility $\sigma_f(t,T)$ determins completely the contract price, since for every choice the drift is determined in a unique way, using the no-arbitrage condition :

$$\mu_f(t,T) = \sigma_f(t,T) \int_t^T \sigma_f(t,s)ds$$

HJM showed that unless one severely restrains the volatility structure, the evolution of the term structure may depend upon the entire path taken since its initialization. But in that case, the evolution of the term structure

isn't Markovian in respect to a finite dimensionnal state. The implementation of pricing models for interest rate derivatives is faced with a number of difficulties. If, for example, one uses a classic lattice model, the paths cannot recombine (due to the non-markovian term structure). The information has to be manipulated along the entire path, and the lattice size grows exponentially (see for example HJM 1990).

Ritchken and Sankarasubramanian (RS) have determined the necessary and sufficient conditions on the volatility structure to resolve these difficulties : these conditions do not suppress the path-dependance of the volatility structures, but summarize it in a single statistic. Thus for volatility structures in this class, the evolution of the term structure can be written in a Markovian form in respect to two state variables.

In the RS model, the class of the volatility structure that allows the trm structure to be represented by a two-state Markovian model is caracterized by :

$$\sigma_f(t,T) = \sigma_f(t,t)k(t,T) \tag{1}$$

with

$$k(t,T) = e^{-\int_t^T \kappa(x)dx}$$

If the volatility structure is of the form (1), then the prices of the zero-coupons can be expressed using the available prices at $t = 0$, the spot interest rate at time t, $r(t)$, and by a second statistic, $\phi(t)$, representing the accumulated variance of the forward rate at date t. If we denote :

$$\beta(t,T) = \int_t^T k(t,u)du$$

and

$$\begin{aligned} \phi(t) &= \int_0^t \sigma_f^2(u,t)du \\ &= \int_0^t \sigma_f^2(u,u)k^2(u,t)du \end{aligned}$$

The theoretical price of a zero-coupon at date t and maturity T is given by (see RS 1995 for the details) :

$$P(t,T) = \frac{P(0,T)}{P(0,t)} e^{-\beta(t,T)(r(t)-f(0,t))-\frac{1}{2}\beta^2(t,T)\phi(t)} \tag{2}$$

The dynamic of the two-state Markovian process is given by :

$$dr(t) = \mu(r, \phi, t)dt + \sigma_f(t, t)dw(t) \tag{3a}$$
$$d\phi(t) = (\sigma_f^2(t, t) - 2\kappa(t)\phi(t))dt \tag{3b}$$

2.2 Dynamic of the state variables

The class of volatilities that we considered is very large, and doesn't impose any particular constraints on the spot volatility structure, $\sigma_f(t, t)$. In particular, the volatilities can depend on the two state variables, $r(t)$ et $\phi(t)$:

$$\sigma_f(t, t) = \sigma(r(t), \phi(t), t) \tag{4}$$

Of particular interest if the following model :

$$\sigma_f(t, t) = \sigma[r(t)]^\gamma \tag{5}$$

RLS study the case $\gamma = 0.5$ (square root model), and the case $\gamma = 1$ (proportional model, which will be implemented)

RLS start by putting the interest rate process of equation (3) under a constant volatility form using the following transformation (see Nelson and Ramaswamy, 1990) :

$$Y(t) = \int \frac{1}{\sigma[r(t), \phi(t), t]} dr(t) \tag{6}$$

Then by denoting $r(t) = h(Y(t))$ the inverse function, we obtain :

$$dY(t) = m(Y, \phi, t)dt + dw(t) \tag{7a}$$
$$d\phi(t) = (\sigma^2[r(t), \phi(t), t] - 2\kappa(t)\phi(t))dt \tag{7b}$$

where :

$$m(Y, \phi, t) = \frac{\partial Y}{\partial t} + \mu(r, \phi, t)\frac{\partial Y}{\partial r(t)} + \frac{1}{2}\sigma^2[r(t), \phi(t), t]\frac{\partial^2 Y(t)}{\partial r(t)^2}$$

To illustrate the transform, let us consider the proportional model where $\gamma = 1$. For $r(t) > 0$, the specific structure of equation (7) results in :

$$Y(t) = \frac{ln[r(t)]}{\sigma} \tag{8a}$$

$$v(Y, \phi, t) = \left[\kappa[f(0,t) - e^{\sigma Y(t)}] + \phi(t) + \frac{d}{dt} f(0,t)\right] \tag{8b}$$

$$m(Y, \phi, t) = \frac{1}{\sigma}\left[v(Y, \phi, t) - \frac{1}{\sigma^2}\right] \tag{8c}$$

For each time increment Δt, the state variables (y^a, ϕ^a) move either to à (y^{a+}, ϕ^{a+}), or to à (y^{a-}, ϕ^{a-}). The evolution of y^a is caracterized by :

$$y^{a+} = y^a + (J+1)\sqrt{\Delta t} \geq y^a + m(y^a, \phi^a, t)\sqrt{\Delta t} \geq y^a + (J-1)\sqrt{\Delta t} = y^{a-} \tag{9}$$

In order to define J, we denote $Z = E[m(y^a, \phi^a, t)\sqrt{\Delta t}]$, $\delta = Sign(Z)$ and we have :

$$J = \begin{cases} \delta Z & \text{if } Z \text{ pair} \\ \delta |Z| + 1 & \text{otherwise} \end{cases} \tag{10}$$

J is chosen in such a way that the two terms y^{a-} et y^{a+} are the upper and lower limit of the interest rate expectation in the next time increment. Since the process for ϕ is locally deterministic, the values ϕ^{a+} and ϕ^{a-} are equal and completely determined by the current state variables (y^a, ϕ^a). If we denote by ϕ^{a*} their common value, we get :

$$\phi^{a+} = \phi^{a-} = \phi^{a*} = \phi^a + \left[\sigma^2(h[y^a], \phi^a, t) - 2\kappa(t)\phi^a\right]\Delta t \tag{11}$$

2.3 Jump Probability

The choice of J is made in a way that the expectation of the interest rate lies between the two successor points y^{a-} and y^{a+}, which constrains the upjump probability to be between 0 and 1. If we denote p this upjump probability $p(y^a, \phi^a)$, and taking the expectation of $(y - y^a)$, we have :

$$p(y^{a+} - y^a) + (1-p)(y^{a-} - y^a) = m(y^a, \phi^a, t)\Delta t \tag{12}$$

thus :

$$p = \frac{m(y^a, \phi^a, t)\Delta t + (y^a - y^{a-})}{y^{a+} - y^{a-}} \tag{13}$$

This methods insures that locally, the expectation of the interest rate r coincides with the drift, and that the variance of the approximated processus converges towards the true variance when taking an increasingly precise time partition.

2.4 Recursion and linear interpolation

In order to calculate recursively the price of an option at time i, knowing the option price at time $i+1$ that we denote by $g_{i+1}(y^a, \phi^a)$, we using the following formula :

$$g_i(y^a, \phi^a) = [pg_{i+1}(y^{a+}, \phi^{a*}) + (1-p)g_{i+1}(y^{a-}, \phi^{a*})]e^{-r_i^a \Delta t} \tag{14}$$

In reality, since ϕ^{a*} is completely determined by (y^a, ϕ^a) the values of the bond at the successor nodes can be unavailable.
For example, we can imagin that the price of the bond for (y^{a+}, ϕ^{a*}) isn't available. Through our construction method, we know that in this case we shall have the prices for the states (y^{a+}, ϕ_+^{a*}) and (y^{a+}, ϕ_-^{a*}), where $\phi_-^{a*} \leq \phi^{a*} \leq \phi_+^{a*}$.
All we need to do to derive the correct price is a linear interpolation :

$$g_{i+1}(y^{a+}, \phi^{a*}) = g_{i+1}(y^{a+}, \phi_-^{a*}) + \left(\frac{\phi^{a*} - \phi_-^{a*}}{\phi_+^{a*} - \phi_-^{a*}}\right)(g_{i+1}(y^{a+}, \phi_+^{a*}) - g_{i+1}(y^{a+}, \phi_-^{a*})) \tag{15}$$

2.5 Summary : the essential equations

The algorithm won't use, of course, all the equations we established in the previous section.

— We shall need the following equations (8a) and (9) to calculate the new interest rate, thus obtaining :

$$r^{a+} = e^{\ln[r(t)] + \sigma(J+1)\sqrt{\Delta t}} \tag{16}$$

$$r^{a-} = e^{\ln[r(t)] + \sigma(J-1)\sqrt{\Delta t}} \tag{17}$$

J is calculated using equation (10).

— The successor of $\phi(a)$ is calculated using (11).

— The up-jump probability is derived of equation (13).

— The bond price is given by equation (2)

— Finally, the linear recursion on option prices is obtained using (14) and interpolation (15)

3 General presentation of the lattice method

The lattice method used here is very conventionnal : we first build a descending tree, meaning that from the first node we build two sons, and each of these two nodes will give birth to two new nodes, etc. Each generation represents one time increment Δt, the total number of generations multiplied by this time increment is equal to the time to maturity of the du contrat. This lattice has the important property of having its branches recombine, property that results from the mathematical choice of constraining the class of admissible volatilities, in such a way that the terme structure is represented by a two-state Markovian structure.
After building the descending lattice, we calculate the price of the bonds at the terminal nodes. Knowing the strike, we can calculate the price of the option at maturity for each node. We shall then climb back the lattice, using the upjump probability, to calculate recursively at each generation the price of the option, until finally reaching the initial node.

4 The constraints of the model

This algorithm demands important constraints in terms of memory. Let us examin what information we shall need to store in order to first build the lattice, then climb it recursively.

— We use a two-state Markovian process, which are the interest rale r and the accumulated variance of the forward rate ϕ. We start by storing the interest rate r in each node of the tree.

— There are as many values of ϕ at each node as paths taken in the lattice that arrive in that node, since equation (11) clearly shows that the successor ϕ^{a*} of element ϕ^a depends on the latter and thus, from node to node, depends on the path taken. Keeping in line with the article, rather that storing all the ϕ's obtained in the node, we shall only keep the minimum and the maximum.

We can thus defined using those two elements an interval $[\phi_{min}, \phi_{max}]$ that we partition into m equidistant points ($\phi_{min} = \phi^a(1) \leq \phi^a(2) \leq ... \leq \phi^a(m) = \phi_{max}$), that we use to calculate the successor points in the nodes below, and from those successor points we can extract the minimum and the maximum to define an interval that we partition, we then use this partition to calculate the successor points, etc...

— To avoid repeating the same calculations, we've chosen to store in each node the partition step of the interval $[\phi_{min}, \phi_{max}]$ as well as the array of ϕ successors : for each element of the $\phi(k)$ partition, we calculate the $\phi(k)^*$ that will succede it and we store it in a new array (in reality comparing it to the minimum and maximum of a new array, to keep only those two elements).
— To avoid visiting the same node twice while building the lattice, we shall embed the nodes with a flog that is toggled from 0 to 1 when the node is visited.

— During the recursion (while climbing back the lattice from the terminal nodes), we need to know the prices of the option at each node, as well as the upjump probability ; since these two values depend on ϕ (and since we have m different values for ϕ at each node) , we shall represent them them as arrays of m elements. While handling an american option we shall need to know the value of the bond price at each node and for the different ϕ's (since it's calculated using equation (2)) : we thus need to store the bond price in the form of an array of size m, which is the number of partition points of the interval $[\phi_{min}, \phi_{max}]$.

5 Simplifying hypotheses

— We consider the term structure to be flat initially. That is we take $f(0,t)$ constant.
— This strike is taken at-the-money, meaning that the strike is equal to the forward price of the bond.
— We only consider the case of a proportionnel volatility structure.

6 Source code description

We shall use the type *double* for all the real numbers used. We need three auxiliary functions, that are given in the beginning of our code, as well as a new structure, well-suited for storing the information contained in each node.

6.1 The auxiliary functions

They are the `min` function and the `max` function, and they calculate the minimum and maximum of a couple.
The function `calcule J` calculates the coefficient J according to the rules defined in (10).

6.2 The structure BOX :

We shall need to store in each node specific information relative to ϕ, r, the upjump probability p, etc. To that end, we use the **BOX** structure. It includes the following elements :

- a **flag** of type *int* to detect if the node has already been visited during the lattice exploration.
- An array containing two elements **phi**[**2**] of type *double*, where we shall store the maximum and the minimum of the ϕ that we've calculated using the ϕ's of the node in the precedent generation.
- A pointer **bondprice** on an element of type *double*, that we shall use to represent the price of the bond. This information is only necessary in the case of an american option, where one has to calculate the maximum between the option price calculated through recursion and the payoff if the option is exercised immediatement, in order to derive the option price.
- A pointer **optionprice** on an element of type *double*, that we shall sue to store the option price
- A pointer **upjump** of type *double*, where we shall store the upjump probability.

6.3 The main functions

There is one main function, called "pricer", which calculates the call/put european/american option. To distinguish between the different cases, two boolean operators specify which case is to be considered. The function is of

type *void*, and takes in input the following arguments suivants :

— The number of elements in the time partition, given by **time_partition**, of type *int*.

— The number of elements in the partition of ϕ (the accumulated variance of the forward rate), given by **nb_partition_phi**, of type *int*.

— The maturity of the option, given by **maturity** of type *int*.

— The initial interest rate r and ϕ, given by **r_0** and **phi_0**, of type *double*.

— The duration of the contract, given by **contract_duration**.

— The face value of the bond considered, **face_value** of type *double*.

— The strike of the option : **strike_price**, of type *double*.

— The volatility **sigma**, of type *double*.
— The exogenous factor **k**, of type *double*.

— Finally **result** the result of the calculation passed as address to the function.

The function uses several local variables of which :

— **time_increment** is the time increment $\Delta t = \frac{maturity}{(nb\_partition\_phi-1)}$.

— **sqrt_time_increment** is simply the square root of the precedent variable, calculated once to avoid repeating it at each loop.

— **forward_price** is the bond's forward price.

These functions store the option price they calculate in the memory space indexed by the **result** pointer .

We shall use a triangular inferior matrix ro represent the lattice. The first node will be the first element of the first line and the first column, the next two nodes will be the first two elements of the second line, etc...
The operations performed by the function are the following :

1. We start by dynamically allocatin the necessary memory : in order to do that, we do a loop on the lines while allocating the necessary memory to a pointer of type **BOX**. We then perform a second loop on the elements of the matrix (those under the diagonal), to allocate in each element of type **BOX** - actually each node of the lattice - the necessart memory for the different arrays. We are referring of course to the arrays **option_price** and **up_jump** (if we were handling an american option, the array **bond_price**) of size **nb_partition_phi**.

2. We then initialize the first node of the lattice, since we know all the necessary information to fill this box : the initial interest rate **r_0**, and the accumulated variance of the forward rate **phi_0**.

3. We then loop on the elements of the triangular matrix (using, like above, two loops of type *for*, the first on the lines and the second on the columns), except the last line, to calculate for each node its successor, that is calculating the interest rate and the ϕ elements of the successor nodes.

   At the node (i,j) (line i, column j), we calculate the ϕ partition step (that we allocate in the variable **pas**), since we know the extremeties of the interval and the number of elements of the partition. We also calculate **v** and **m** using equations (8b) and (8c), as well as the successor of **phi[0]**, and the variation of the successor (**accroissement_phi_successeur**) when we move from $\phi(k)$ to $\phi(k+1)$ in the partition : this allows us, while performing a loop on all elements of the partition and adding after each loop to the variable **phi_successeur** the value **accroissement_phi_successeur**, to derive the successor **phi_successeur[n]** of the partition element (**phi[n]**) on which we are looping.

4. Inside the previous loop on the matrix elements, that is at the level of each node being processed, we loop on all the elements of the ϕ partition of the node : we calculate **J** using **m** and **sqrt_time_increment** (see (10)) which allows us to calculate the interest rate **r** of the successor, and depending on whether the son has allready been visited or not, we store that interest rate.
   In the same manner, if the successor has not been accessed, we store the value of **phi_successeur** in the two elements of the array, and if it

has already been visited we take the maximum of the previous maximum and **phi_successeur**, and we attribute that value to **phi[1]**. We perform a similar operation for the minimum, that we attribute to **phi[0]**. The calcultion of m and J allows us, using equation (13), to derive the upjump probability that we attribute to the proper array location (**up_jump[n]**).

5. We then loop on the terminal nodes to allocate the partition step to the according variable **pas**.

6. We loop again on the terminal nodes : for each node, we loop on the ϕ partition to calculate the option price at the maturity date and allocate it in the corresponding variable **option_price[n]**. In order to do that it is sufficient to calculate the bond price using (2),then using the call or put formula depending on the case : $((S_T - K)_+$ for a call or $(K - S_T)_+$ for a put).

7. We shall finally climb back the lattice : this is done through an upward loop on the lines (from the bottom of the matrix to the top), which includes a loop on the columns, which itself includes a loop on the ϕ partition : in this last loop, we calculate using the linear interpolation of equation (15) the termes $g_{i+1}(y^{a+}, \phi^{a*})$ and $g_{i+1}(y^{a-}, \phi^{a*})$ that are then stored in the variables **gplus** and **gmoins**.

   In order to determin which terms surround ϕ^*, we verify if the latter is the maximum element of the partition : if it is, then we don't need to do a linear interpolation, and if not we calculate the number of the elements that surrounds ϕ^* in the partition de ϕ (we take the integer value of the distance between ϕ^* and the minimum of the partition divided by the partition step). Finally, we use equation (14) for the upward recursion, which concludes the loop.

7 Limitations

Limited memory available : in the case of a european option, a time partition of 800, with a phi partition of 300 is impossible to calculate with 256 MB RAM computers.

8 Typing errors in the LRS article

Page 728, Figure 2 : “computed using équation 6a” instead of “computed using équation 8b” Page 730, Figure 3 : “The strike price is 22.3130” instead of “The strike price is 21.3130”