Finite Elements and Streamline Diffusion for the Pricing of Structured Financial Instruments

Andreas Binder, binder@mathconsult.co.at
Andrea Schatz, schatz@mathconsult.co.at

Abstract

The numerical treatment of partial differential equations in computational finance started with binomial and trinomial trees, with all the drawbacks related to these approaches. In the meanwhile (see, e.g., Duffy 2004, in the July issue of this magazine), finite differences are widely used in modern derivatives pricing. We present how pricing software can be developed on the basis of finite element techniques, which allow more flexibility than finite differences.

Mean reverting models for interest rates tend to become numerically difficult in regions sufficiently far away from the mean-reverting level. The reason is that the convection dominates the diffusion in these regions, and therefore techniques for convection-dominated flows should be applied. We present how streamline diffusion is applied to obtain stable numerical schemes.

We implemented these approaches in a strictly object-oriented software framework. Some software engineering aspects are also highlighted.

Introduction

We consider models for financial instruments which can, after some manipulation, be written in the form of parabolic partial differential equations backwards in time. The manipulation typically requires some Itô calculus, the creation of a risk-free portfolio and self-financing hedging strategies and some assumptions (like zero transaction costs), which are certainly wrong but a good starting point. LIBOR market models typically do not fall into this category, but short rate models do.

For example, let us start with a two-factor Hull-White interest rate model (see Hull-White 1994)

$$dr = [\theta(t) + u(t) - a(t)r(t)]dt + \sigma_1(t)dX_1$$
$$du = -b(t)u(t)dt + \sigma_2(t)dX_2$$

The first factor r denotes the spot rate, the second factor u some kind of long-term development of the interest rates. a is the mean reversion speed of the spot rate r, $(\theta + u)/a$ its reversion level. The stochastic variable u itself reverts to a level of zero at rate b. dX_1 and dX_2 are increments of Wiener processes with instantaneous correlation $\rho(t)$. σ_1 and σ_2 are the volatilities.

No-arbitrage-arguments then lead to the fundamental Hull-White equation

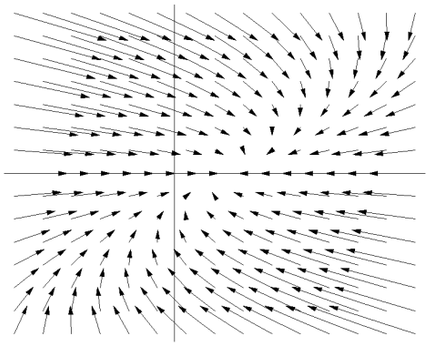
$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_1(t)^2\frac{\partial^2 V}{\partial r^2} + \rho(t)\sigma_1(t)\sigma_2(t)\frac{\partial^2 V}{\partial r \partial u} + \frac{1}{2}\sigma_2(t)^2\frac{\partial^2 V}{\partial u^2}$$
$$+ (\theta(t) + u - a(t)r)\frac{\partial V}{\partial r} - b(t)u\frac{\partial V}{\partial u} - rV = 0,$$

which needs additional end and transition conditions. The calculation domain is, in principle, unbounded. We will discuss the problem of boundary conditions, when restricting ourselves to a bounded calculation domain, below.

The end and transition conditions describe the special shape of a financial contract, like coupons, callabilities and so on.

The given partial differential equation can be interpreted as a diffusion-convection-reaction equation. This type of equation is typically found in applications in continuum mechanics, especially in fluid mechanics. The dissolving of sugar in a cup of coffee, for example, could be described by this type of equation. The dispersion of the sugar due to concentration differences is a diffusion process, described by the second order terms in the equation. The spreading of the sugar driven by a stirring spoon is the

convective part, given by the first order terms, and strongly dominated by the velocity of the coffee. The dissolving of the sugar itself is described by the reactive part, which is the last term at the left hand side of the equation. The following picture, which shows velocity vectors in the *ru*-plane, could give the motion of the coffee forced by the stirring spoon, but, in fact, it gives the deterministic movement of the interest rates in a two-factor Hull-White interest rate model.



This picture demonstrates that in the two-factor Hull-White model the convective part becomes more and more important the larger the considered domain.

It is obvious now that numerical methods used in computational fluid dynamics to solve equations of this type, will work well also for our pricing problems. In computational fluid dynamics it is well known that in the cases of comparatively large or dominating convection standard numerical discretisation techniques lead to instabilities in the numerical solution. These instabilities result in high oscillations. We have to use so-called upwind-strategies, which take into account that in the case of dominating convection the solution in each point is strongly determined by the information transported with the velocity.

Since in the considered pricing problems end conditions for the quantity V are prescribed we have to solve the equation backwards in time. So the information transport due to convection starts in the centre and goes to the boundary.

Numerical Schemes and Finite Elements

Finite volume method

The basic idea of the finite difference method is to approximate the derivatives in the partial differential equation by finite differences. In the case of higher dimensions, especially when including mixed derivatives, a more general formulation is preferred and known under the name finite volume method. The essential idea is to use an integral formulation, integrating the equation over a mesh region and applying Gauss' theorem before carrying out the discretisation:

We start already from the time discretised equation, either fully implicit

$$\frac{V^{n+1}-V^n}{\Delta t}+\frac{1}{2}\sigma_1^{n+1^2}\frac{\partial^2 V^{n+1}}{\partial r^2}+\rho^{n+1}\sigma_1^{n+1}\sigma_2^{n+1}\frac{\partial^2 V^{n+1}}{\partial r\partial u}+\frac{1}{2}\sigma_1^{n+1^2}\frac{\partial^2 V^{n+1}}{\partial u^2}$$
$$+(\theta^{n+1}+u-a^{n+1}r)\frac{\partial V^{n+1}}{\partial r}-b^{n+1}u\frac{\partial V^{n+1}}{\partial u}-rV^{n+1}=0,$$

or, e.g., of Crank-Nicolson type ($\alpha = 0.5$)

$$\frac{V^{n+1}-V^n}{\Delta t}+\left(\frac{1}{2}\sigma_1^{n+1^2}\frac{\partial^2 V^{n+1}}{\partial r^2}+\rho^{n+1}\sigma_1^{n+1}\sigma_2^{n+1}\frac{\partial^2 V^{n+1}}{\partial r\partial u}+\frac{1}{2}\sigma_1^{n+1^2}\frac{\partial^2 V^{n+1}}{\partial u^2}\right.$$
$$\left.+(\theta^{n+1}+u-a^{n+1}r)\frac{\partial V^{n+1}}{\partial r}-b^{n+1}u\frac{\partial V^{n+1}}{\partial u}-rV^{n+1}\right)\alpha$$
$$+\left(\frac{1}{2}\sigma_1^{n^2}\frac{\partial^2 V^{n}}{\partial r^2}+\rho^{n}\sigma_1^{n}\sigma_2^{n}\frac{\partial^2 V^{n}}{\partial r\partial u}+\frac{1}{2}\sigma_1^{n^2}\frac{\partial^2 V^{n}}{\partial u^2}\right.$$
$$\left.+(\theta^{n}+u-a^{n}r)\frac{\partial V^{n}}{\partial r}-b^{n}u\frac{\partial V^{n}}{\partial u}-rV^{n}\right)(1-\alpha)=0.$$

The top indices n and $n+1$ are used for the values at different time levels, where the values at time level n are known and the values at time level $n+1$ are unknown. For ease of readability we will use the fully implicit time discretisation in the following.

The computational domain Ω is discretised into finite volumes $\Omega_i, i = 1, \ldots, N$. The next step is to integrate the equation over these finite sub-domains:

$$\int_{\Omega_i}\frac{V^{n+1}-V^n}{\Delta t}d(r,u)+\int_{\Omega_i}\frac{1}{2}\sigma_1^2\frac{\partial^2 V^{n+1}}{\partial r^2}+\rho\sigma_1\sigma_2\frac{\partial^2 V^{n+1}}{\partial r\partial u}$$
$$+\frac{1}{2}\sigma_2^2\frac{\partial^2 V^{n+1}}{\partial u^2}d(r,u)+\int_{\Omega_i}(\theta+u-ar)\frac{\partial V^{n+1}}{\partial r}$$
$$-bu\frac{\partial V^{n+1}}{\partial u}d(r,u)-\int_{\Omega_i}rV^{n+1}d(r,u)=0\qquad\forall i=1,\ldots,N$$

This is equivalent to

$$\int_{\Omega_i}\frac{V^{n+1}-V^n}{\Delta t}d(r,u)+\int_{\Omega_i}\frac{\partial}{\partial r}\left(\frac{1}{2}\sigma_1^2\frac{\partial V^{n+1}}{\partial r}+\frac{1}{2}\rho\sigma_1\sigma_2\frac{\partial V^{n+1}}{\partial u}\right)$$
$$+\frac{\partial}{\partial u}\left(\frac{1}{2}\rho\sigma_1\sigma_2\frac{\partial V^{n+1}}{\partial r}+\frac{1}{2}\sigma_2^2\frac{\partial^2 V^{n+1}}{\partial u^2}\right)d(r,u)$$
$$+\int_{\Omega_i}\frac{\partial}{\partial r}\left((\theta+u-ar)V^{n+1}\right)-\frac{\partial}{\partial u}\left(buV^{n+1}\right)+aV^{n+1}+bV^{n+1}d(r,u)$$
$$-\int_{\Omega_i}rV^{n+1}d(r,u)=0\qquad\forall i=1,\ldots,N$$

Those volume integrals which contain a divergence term are converted into surface integrals by Gauss' theorem and are evaluated as fluxes across the boundaries Γ_i of each finite volume. (n_r, n_u) denotes the outer

unit normal vector at the boundaries.

$$\int_{\Omega_i} \frac{V^{n+1}-V^n}{\Delta t} d(r,u) + \int_{\Gamma_i} \left(\frac{1}{2}\sigma_1^2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial u}\right) n_r + \left(\frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V^{n+1}}{\partial u^2}\right) n_u \, ds + \int_{\Gamma_i} \left((\theta + u - ar)V^{n+1}\right) n_r - \left(bu\, V^{n+1}\right) n_u \, ds + \int_{\Omega_i} (a+b-r)V^{n+1} \, d(r,u) = 0 \qquad \forall i = 1, \ldots, N$$

The finite dimensional equation is then obtained by the use of quadrature rules for the given integrals. As outlined in the introduction the discretisation of the convection term requires special attention. The flux across the boundaries due to convection has to be treated with special upwind techniques, like Lax-Wendroff or QUICK schemes (see Morton 1996). Detailed analysis of the obtained numerical schemes leads to the conclusion that the introduction of upwind schemes is equivalent to the addition of artificial numerical diffusion.

In the early references the finite volumes are usually rectangular and occasionally quadrilateral, extending to hexahedral volumes in three dimensions. In the case of rectangular finite volumes the obtained discretisation schemes are equivalent to the one obtained using the finite difference approach.

Finite Element Method

The finite volume method itself can be treated as a variant of the finite element method. Starting point for the finite element method is the weak formulation of the given equation. Under the assumption that we are looking for a function V in a function space U we can write the weak form of the already implicitly time discretised problem as:

Find $V^{n+1} \in U$ such that, for all $w \in U$,

$$\int_\Omega \frac{V^{n+1}-V^n}{\Delta t} w \, d(r,u) + \int_\Omega \left(\frac{\partial}{\partial r}\left(\frac{1}{2}\sigma_1^2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial u}\right) + \frac{\partial}{\partial u}\left(\frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V^{n+1}}{\partial u^2}\right)\right) w \, d(r,u) + \int_\Omega \left((\theta + u - ar)\frac{\partial V^{n+1}}{\partial r} - bu\frac{\partial V^{n+1}}{\partial u}\right) w \, d(r,u) - \int_\Omega \left(r V^{n+1}\right) w \, d(r,u) = 0$$

Applying Gauss' theorem in the second order terms lead us to

Find $V^{n+1} \in U$ such that, for all $w \in U$,

$$\int_\Omega \frac{V^{n+1}-V^n}{\Delta t} w \, d(r,u) - \int_\Omega \left(\frac{1}{2}\sigma_1^2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial u}\right) \frac{\partial w}{\partial r} + \left(\frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V^{n+1}}{\partial u^2}\right) \frac{\partial w}{\partial u} d(r,u) + \int_\Gamma \left(\frac{1}{2}\sigma_1^2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial u}\right) w n_r + \left(\frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V^{n+1}}{\partial u^2}\right) w n_u \, ds + \int_\Omega \left((\theta + u - ar)\frac{\partial V^{n+1}}{\partial r} - bu\frac{\partial V^{n+1}}{\partial u}\right) w \, d(r,u) - \int_\Omega \left(r V^{n+1}\right) w \, d(r,u) = 0$$

Consider a discretisation Ω_i, $i = 1, \ldots, N$ of the domain Ω and $U_h \subset U$ a finite element space that consists of piecewice polynomials. Replacing the trial and test space U by this finite dimensional space U_h and approximating the function V by a linear combination of basis functions of the trial space lead to the finite dimensional problem. This would be the standard finite element approach disregarding possible difficulties caused by dominating convection. Up to now the special type of the equation is not taken into account. A rather elegant way to introduce upwind techniques to this scheme is used in the method of streamline diffusion (see also Roos-Stynes-Tobiska 1996).

Streamline Diffusion—Going with the Flow

The fundamental idea of this method is to add extra diffusion in the direction of the streamline – hence the name streamline diffusion. From the technical point of view this is realized by replacing the test function w with a test function of the form $w + \delta_i v \cdot \nabla w$, where $v(= (\theta + u - ar, -bu)^T)$ denotes the velocity, and δ_i is called the SD-parameter.

The weak formulation then reads as:

Find $V^{n+1} \in U$ such that, for all $w \in U$,

$$\int_\Omega \frac{V^{n+1}-V^n}{\Delta t} w d(r,u) - \int_\Omega \left(\frac{1}{2}\sigma_1^2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial u}\right) \frac{\partial w}{\partial r} + \left(\frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V^{n+1}}{\partial u^2}\right) \frac{\partial w}{\partial u} d(r,u) + \int_\Gamma \left(\frac{1}{2}\sigma_1^2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial u}\right) w n_r + \left(\frac{1}{2}\rho\sigma_1\sigma_2 \frac{\partial V^{n+1}}{\partial r} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V^{n+1}}{\partial u^2}\right) w n_u \, ds + \int_\Omega \left((\theta + u - ar)\frac{\partial V^{n+1}}{\partial r} - bu\frac{\partial V^{n+1}}{\partial u}\right) w \, d(r,u) - \int_\Omega (r V^{n+1}) w \, d(r,u) + \sum_{i=1}^{N} \int_{\Omega_i} \delta_i \frac{V^{n+1}-V^n}{\Delta t} v \cdot \nabla w \, d(r,u)$$

$$+\sum_{i=1}^{N}\int_{\Omega_i}\delta_i\left(\frac{\partial}{\partial r}\left(\frac{1}{2}\sigma_1^2\frac{\partial V^{n+1}}{\partial r}+\frac{1}{2}\rho\sigma_1\sigma_2\frac{\partial V^{n+1}}{\partial u}\right)\right.$$

$$\left.+\frac{\partial}{\partial u}\left(\frac{1}{2}\rho\sigma_1\sigma_2\frac{\partial V^{n+1}}{\partial r}+\frac{1}{2}\sigma_2^2\frac{\partial^2 V^{n+1}}{\partial u^2}\right)\right)v\cdot\nabla w\,d(r,u)$$

$$+\sum_{i=1}^{N}\int_{\Omega_i}\delta_i\left((\theta+u-ar)\frac{\partial V^{n+1}}{\partial r}-bu\frac{\partial V^{n+1}}{\partial u}\right)v\cdot\nabla w\,d(r,u)$$

$$+\sum_{i=1}^{N}\int_{\Omega_i}\delta_i\left(rV^{n+1}\right)v\cdot\nabla w\,d(r,u)=0$$

The additional term in the convective part is:

$$\sum_{i=1}^{N}\int_{\Omega_i}\delta_i\left((\theta+u-ar)^2\frac{\partial V^{n+1}}{\partial r}\frac{\partial w}{\partial r}-(bu)^2\frac{\partial V^{n+1}}{\partial u}\frac{\partial w}{\partial u}\right)d(r,u),$$

which has the typical form of a diffusion term. The SD-parameter δ_i depends on the size of the finite elements and on the convection-diffusion ratio, so artificial diffusion is chosen higher in convection dominated regions and smaller in regions where diffusion dominates.

Although the size of the computational domain is, in principle, unbounded, we have to do our calculations on a bounded domain. It is always difficult to find appropriate and realistic boundary conditions for each structured financial instrument considered. We choose the size of the computational domain in a way such that the information of the prescribed boundary condition does not get through to the centre, during the considered time interval. The centre of the domain is determined by the current short rates. So the choice of boundary conditions, which have to be set for solving the partial differential equation, has no influence on the solution. This may be interpreted in such a way that the probability of very high or low, maybe even negative, interest rates is very small.

Therefore it is clear that the size of the computational domain depends on the life time of the considered instrument and on the parameter which form the coefficient functions of the partial differential equation: volatility, drift and mean reversion.

In the method of finite elements we are very flexible concerning the discretisation of Ω. Structured as well as unstructured grids with adaptive refinement in regions where it is necessary can be chosen. The standard setting in our calculation is a structured, two-dimensional, quadrilateral grid with graded higher resolution in both directions near the values of interest of the factors r and u.

Discretisations in time and space (r-u-plane) can be chosen independently in the case that we use implicit time discretisation, either fully implicit or some kind of Crank-Nicolson (see e.g. Duffy 2004).

Solution of the Linear Equations

The discretisation leads then to sparse linear systems with typically thousands of variables for each time steps. These are then solved iteratively by Krylov subspace techniques which typically show very fast convergence.

Comparison to Analytic Solution

In the following we compare the numerical results for the pricing of zero coupon bonds with face amount 1 and different life times obtained by the use of standard finite elements, finite elements with streamline diffusion, and the analytical solution under the two factor Hull-White interest rate model with constant model parameters. The used parameter settings are:

$$a=1.2,\quad b=0.03,\quad \theta=0.05,\quad \sigma_1=0.02,\quad \sigma_2=0.01,\quad \rho=0.5$$

Life times	Analytical solution	Standard finite elements	Finite elements with streamline diffusion
1 year	0.954581	0.95458	0.95458
10 years	0.661886	0.661777	0.661855
20 years	0.461421	0.460902	0.461405
40 years	0.268027	0.265955	0.268311

These results confirm, even for this simple example, that the longer the life time of an instrument, the more important the usage of upwind techniques. We used a discretisation with a time step of 20 days and a space discretisation of 30×30 points (which are quite few points for instruments with such long lifetimes.

If we use a space discretisation which is even coarser, namely 10×10 (obviously too coarse), we still obtain realistic results in the streamline diffusion case, but unacceptable result for long life times in the standard finite element case:

Life times	Analytical solution	Standard finite elements	Finite elements with streamline diffusion
1 year	0.954581	0.954592	0.954582
10 years	0.661886	0.663975	0.661414
20 years	0.461421	0.472521	0.459271
40 years	0.268027	0.464601	0.262467

Software Architecture

We laid special emphasis on implementing these numerical techniques in a strictly object-oriented framework. We used C++ as programming language utilizing the concepts of objects, class hierarchies and polymorphism.

- An *object* has state (data) and behaviour (functions). Each object is created from a class which is a specification of the data and functions. All objects of a class have common behaviour but generally different state.
- Using *class hierarchies*, classes with common components and operations need not be recoded. This mechanism is called inheritance.
- And last but not least, *polymorphism* allows different kind of objects that have common behaviour to be used in code that only uses this common behaviour.

A detailed introduction into the object-oriented programming style with special emphasis on scientific and engineering programs can be found in Barton-Nackman 1994, for a general description and as a reference manual see Stroustrup 2000.

How are these concepts realized in our code?

Each instrument which can be priced with our finite element code consists of the base class *BasisInstrument* and different *AttributeManagers*. These *AttributeManagers* handle different possible attributes of a structured financial instrument, like callability, coupon payments, or discrete dividends. So for example a callable, convertible, fixed rate bond inherits the same class *Callable* as a callable constant maturity floater. So the implementation of a new structured instrument having already existing attributes is rather easy. All attributes which exist already can be combined with new ones to add new instruments.

The core of the two-factor pricing is built by the class FEPricer (FiniteElementPricer). This class knows everything needed about finite elements with streamline diffusion. With the aid of pointers to an object of the class BasisModel and to an object of the class BasisInstrument the information which two-factor model should be used and which instrument should be priced is obtained. In this part of the program, polymorphism is strongly applied.

Going Further

In the previous sections, we have derived the numerical schemes for the solution of the two factor Hull-White differential equation. These methods (finite elements and streamline diffusion) can of course also be applied to different problems like: different interest rate models which can be written as PDEs, quanto swap problems being built from two one-facto interest rate models, callable convertible bonds and many more. The techniques can also be applied for models in more than two space dimensions. Realistically, there will be a performance problem in problems with 4+ space dimensions which would lead to equations with millions of unknowns.

Until now, we have not said too much about end and interface conditions. Consider, e.g., a callable reverse CMS, i.e. a bond, which pays annual coupons of, say, 10% minus the 5 year swap rate, capped at 7% and floored at 2%. These coupons should be set at the beginning of each coupon period. The lifetime of these instruments is typically quite long (10 to 30 years). To make it more complex, the instrument is equipped with a Bermudan callability at each coupon date.

How do we obtain the swap rates at the coupon set dates? The Hull-White equation has a Green's function (the calculations may become quite tedious if the parameters in the model are not constant but, say, piecewise constant). The value $V(r, u, t)$ of a zero coupon bond maturing at a time T requires then the calculation of some integrals only. Swap rates can then be obtained by reverse bootstrapping and taking into account the appropriate day count conventions for the swaps.

At maturity, the bond pays the redemption plus the coupon which was set at the beginning of the last coupon period and which we therefore do not know when propagating backwards from maturity. What we can do, is calculate the different discount factors from maturity to the coupon set date in the different states of r and u at the coupon set date and then multiply them by the coupon rate at the set date.

If the instrument is callable, we have to compare the staying alive value and the call price and take the minimum at the Bermudan call dates. Continuing this propagating backwards, we finally reach the valuation date.

More Software Architecture

Our UnRisk library is not linked to some external C++-code, but is installed within Mathematica as an application package. Therefore we have the following architecture:

UnRisk is called by the Mathematica Kernel, which itself is called either by the Mathematica front end or (via Mathematica Link for Excel) by the Excel front end. Using the Excel front end, the user typically obtains market information like interest rates or volatilities from information providers like Reuters or Bloomberg.

The Mathematica front end, on the other hand, may be used to write additional code, to produce interactive documents or to generate graphics and animations.

The valuation of a callable reverse floater in the Mathematica front end might look like this:

Load the package

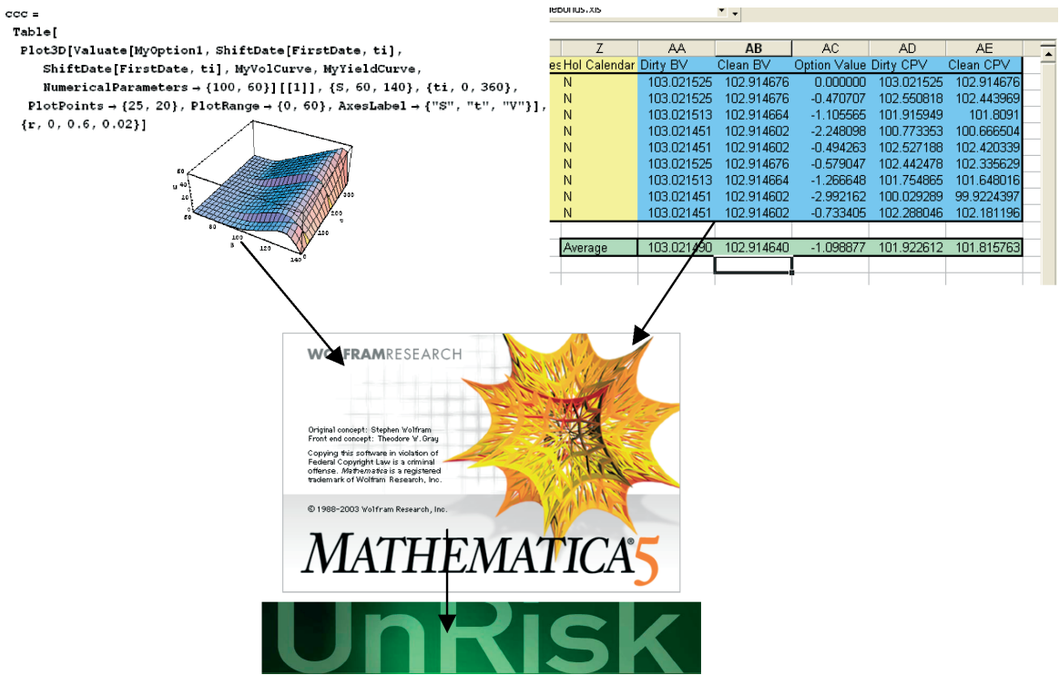
```
Needs["UnRisk`UnRiskFrontEnd`"]
```

Construct a reverse floater (maturity 2024) which pays annual coupons of 12 percent ("Margin") minus ("Reference Weight") the 5 years (=60 months) swap rate set in advance ("RefixAttributes") with caps and floors at 8 and 2 percent, repectively.

```
MyGeneralCMF=MakeGeneralCMFloater[{0.05}, {2024, 10, 10},
{2004, 10, 10}, {2005, 10, 10}, 60, FaceAmount→100,
CouponFrequency→"Annual", CouponBasis->"30/360",
RateFrequency->"Annual", RateBasis->"30/360", Margin→0.12,
ReferenceRate->"Swap", RefixAttributes→{1, 0, 12},
ReferenceWeight→1, Cap→0.08, Floor→0.02];
```

The bond should be callable annually, starting in 2009

```
MyCallSchedule=MakeCallPutSchedule[Table[{{2008+i, 10, 10}, 1.},
{i, 1, 15}]];
MyCPGeneralCMF=MakeCPGeneralCMFloater[MyGeneralCMF,
CallSchedule→MyCallSchedule, CallExercise->"Bermudan",
CallAccrued→True];
```



Construct the Two-Factor-Hull-White model from interest rate curves, cap volatilities and at-the-money swaption volatilities.

```
MyToday={2004, 10, 26;}
MySwapCurve=MakeSwapCurve[MyToday, {{7, .03331 {31, .03162},
{62, .03125}, {92, .03043}, {123, .03011}, {153, .02989}, {184,
.0297}, {274, .02959}, {365, .02973}, {730, .0324}, {1095,
.03525}, {1461, .0378}, {1826, .03995}, {2191, .04185}, {2556,
.0435}, {2921, .0448}, {3286, .0459}, {3651, .0468}, {4380,
.04825}, {5475, .0497}, {7300, .051}, {9125, .05145}, {10950,
.0513}}, MoneyMarketBasis->"ACT/360", SwapBasis->"30/360",
SwapFrequency->"Annual"];
MyYieldCurve=MakeYieldCurve[MySwapCurve];
MyCapStrikes={0.025, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055,
0.06, 0.07, 0.08, 0.09, 0.1};
MyCapMaturities={{2, "30/360", "Quarter-Annual" {3, "30/360",
"Semi-Annual"}, {4, "30/360", "Semi-Annual"}, {5, "30/360",
"Semi-Annual"}, {6, "30/360", "Semi-Annual"}, {7, "30/360",
"Semi-Annual"}, {8, "30/360", "Semi-Annual"}, {9, "30/360",
"Semi-Annual"}, {10, "30/360", "Semi-Annual"}};
MyCapVolas ={{.288, .262, .249, .253, .26, .268, .276, .283,
.296, .31, .324, .339 {.289, .257, .236, .221, .218, .219,
.222, .229, .239, .252, .267, .284}, {.281, .249, .224, .207,
.199, .196, .196, .199, .206, .218, .233, .247}, {.274, .242,
.216, .198, .187, .18, .178, .178, .183, .193, .205, .219},
{.267, .236, .21, .191, .178, .168, .165, .164, .166, .174,
.184, .197}, {.261, .231, .205, .185, .171, .161, .156, .154,
.154, .16, .169, .18}, {.256, .226, .201, .181, .166, .155,
.149, .146, .146, .15, .158, .167}, {.251, .222, .198, .177,
.162, .151, .144, .14, .139, .142, .149, .157}, {.246, .219,
.195, .174, .159, .148, .139, .136, .135, .137, .143, .15}};

MySwaptionExpiries={2, 5, 10;}
MySwaptionEnds={3, 5, 10, 20;}
MySwaptionVolas ={{.179, .156, .131, .118 {.129, .121, .112,
.105}, {.105, .104, .101, .096}};}
MySwapFrequency="Annual";
MySwapBasis="ACT/360";

MyModel=Make2DModel[MyYieldCurve, MyCapMaturities,
MyCapStrikes, MyCapVolas, MySwaptionExpiries, MySwaptionEnds,
MySwaptionVolas, SwapFrequency->MySwapFrequency, SwapBasis-
>MySwapBasis];
```

The calibration problem is an ill-posed problem meaning that small perturbations in the data can lead to arbitrarily large perturbations in the resulting interest rate model parameters if no special stabilizing techniques, so called regularization methods, are applied. We will discuss this aspect in a forthcoming paper.

Our experience shows that one should use as many swaption data as available especially on the long end of lifetimes to obtain good pricing results for bonds with long lifetimes.

Valuate the bond

```
SettlementDay=ShiftByBusinessDays[MyToday, 3];
Valuate[MyCPGeneralCMF, MyToday, SettlementDay, MyModel]
{113.676,113.426,-11.3674,102.309,102.059}
```

The returned vector contains dirty and clean value of the pure reverse CM floater (without callability), the option value of the callability, and dirty and clean value of the callable reverse CMF.

Conclusions

We have presented how finite element techniques can successfully be applied to the pricing of complex structured instruments. Streamline diffusion turns out to be a method which is capable of stabilizing problems with large or dominating convection.

Authors

Andreas Binder got his Ph.D. in Applied Mathematics (University of Linz) in 1991 for the numerical treatment of some problems in continuous casting of steel. He is CEO of MathConsult since 1996. They work on numerical schemes in engineering applications and computational finance since then. In 2001, they released the first version of the UnRisk PRICING ENGINE.

Andrea Schatz worked on the mathematical modelling and numerical treatment of the COREX process for iron production to obtain her Ph.D. in Industrial Mathematics (University of Linz). She is responsible for the finite element development in the UnRisk PRICING ENGINE.

UnRisk is a registered trademark of MathConsult.

REFERENCES

- Barton, J. J. and Nackman, L. R. 1994 *Scientific and Engineering C++, An Introduction with Advanced Techniques and Examples*. Addison-Wesley, New York
- Duffy, D. J., 2004: A critique of the Crank Nicolson scheme strengths and weaknesses for financial instrument pricing., Wilmott magazine, July 2004, 68–76
- Hull, J. and White, A. 1994 *Numerical Procedures for Implementing Term Structure Models II: Two-Factor Models.* Journal of Derivatives, 37–48
- Morton, K. W. 1996 *Numerical Solution of Convection-Diffusion Problems.* Chapman & Hall, London UK
- Stroustrup, B. 2000 *The C++ Programming Language* Addison-Wesley. Reading Massachusetts USA
- Roos, H.-G., Stynes, M., and Tobiska, L. 1996 *Numerical Methods for Singularly Perturbed Differential Equations—Convection-Diffusion and Flow Problems* Springer Verlag, Berlin
- UnRisk Manual: Available from www.unriskderivatives.com

ACKNOWLEDGEMENT

The work of Andrea Schatz has been supported by the Austrian Science Foundation (FWF, www.fwf.ac.at) in the project E67, "Fast numerical methods in computational finance".