## 1. Slide 1

Show the slide

Hello everyone, today, I give a presentation about two topics: the pricing of autocallable contracts and the methods for fast computing multivariate normal probability

## 2. Slide 2

Show the slide

Here is our agenda. Because of the shortage of the presentation time, I can only present briefly

- first, the obtained results

- second, share interesting findings, key ideas, and interesting methods that I used for the two studies and

- last, inform the progression of the project

For the sake of clarity, slides contain on purpose as few mathematical formulas as possible. For those who are math lovers, please read my three documents I sent you last week for the pleasure.

I intended to talk in detail about the MNP methods in this presentation. But I doubt we have enough time for this, so, I put all the methodology in Appendix. I will send you later this PowerPoint presentation.

## 3. Slide 3 – Cartography

I know you all are familiar with the autocallable. Behnaz said to me that there is a first presentation about this product which was made last year by Yousra and Behnaz. So, I allow myself to focus only on the technical aspect of the pricing, to make a cartography for this purpose, to show how we decompose an autocallable into several components and what are their important features.

An autocallable can be decomposed into 3 components and its first important feature is the cancellation condition. If at a date before its maturity, a cancellation event is triggered, the contract pays to the investor a rebate amount and stops. If not, the contract pays coupons and at the maturity, it pays an option.

The second important feature of autocallables is on the underlying. Underlying can be “mono asset”, “Best-of /worst-of asset” or “Basket of assets”.

We go now to the three components

Concerning the Rebate component, nothing is special here.

Concerning the coupon component, there are phoenix coupon, snowball coupon and range accrual coupon that are the three most important features.

For the option component, there are a very interesting feature, that is the Knock-in barrier monitoring. Barrier monitoring can be of type European, Bermuda or American.

This cartography covers all important features of autocallables. We have now the closed form exact formulas for all of them.

Concerning the implementation in Gator, only 4 cases in grey have not been implemented yet in Gator, because of lacking of time, but I will implement them in Gator later. (Best-of/Worst-of + Basket + Floating coupon + Continuous barrier monitoring). We remind that for the 4 cases, we had already the methodology.

## 4. Slide 4 – Closed form exact pricing formulas

For the pricing, as you all know there are two steps in general, first we must be able to write the payoff. Second, we find the expectation of this payoff.

For the pricing of autocallables, as there are many variants (in theory, not less that 1500 variants), we add a supplementary step: “payoff decomposition”. And we use one single formula, that I named : Base formula 1.

All payoff of all type of the three components can be easily decomposed into this Base formula 1 and have closed form exact formula.

Besides, the final payoff is experimentally compact and irreducible and this method is already optimal regarding to the computation complexity. Up to now, for all payoff I have decomposed, there is no other exact formula which is more compact than the final payoff decomposition by Base formula 1.

## 5. Slide 5 – Option decomposition

So, we just show we can price optimally all variants of autocallable with the Base formula 1. But for the option component, it is more complex than other components.

Currently, there are many option types (put, call, digital,…) , and we may have new option with unknown yet payoff (perhaps butterfly? Condor? Or a payoff more complex). So, we need a generic solution.

We don't directly price the option as we did for rebate and coupon components. We use a method, named "Option decomposition".

Instead of pricing directly option by option, we price the 4 base options. And from the 4 base options, we can construct any option payoff we want.

I already show some examples of "Option decomposition" in the document I sent you last week.

With this method, we can price, always with closed form exact expression, almost all possible options.

For any new options, we need only less than 3 code lines. So, this method facilitates clearly further implementation of new options.

## 6. Slide 6 – Best-of/Worst-of

Come back to the Cartography slide

At this stage, we can price now all the variants of mono-autocallables. What remains now is the two features: *Best-of/ Worst-of* underlying and *Basket underlying*.

Return to slide 6

For Worst-of underlying, the underlying is the worst of performing asset, its formula is as follows …

If we can use the same method as for the case of mono autocallable, we can price analytically all Best-of/ Worst-of autocallable.

Here is the **Base formula 2,** can we compute the expectation of this formula? Yes we can, and this finding have not existed yet in the literature, to the best of my knowledge.

The key technique is called "Integration region split". The proof is in the appendix of my document.

## 7. Slide 7 –

The expectation of the Base formula 2 contain many Multivariate Normal Probabilities. In CCR scope, as we must compute a million prices for each product, this exact formula is difficult to be used as it is highly time consuming.

In fact, I tried to find this exact formula to be able to compute fast autocallables in CCR scope, as for the mono underlying case. It turns out that our quest is only half achieved (we have exact method, but not fast). So, our formula may be used for other purposes, but not in CCR scope.

We have no choice but to use approximation methods. But we dispose a back-up solution.

Approximation $X_{\min}$ by $X$

This approximation method was already used for Best-of/Worst-of vanilla option in 2015.

## 8. Slide 8 –

We did talk about the theoretical formulas. We need now some numerical results to make sure that our formulas work in practice.

I recall that, our objective here is to implement and to test as many autocallable variants as possible. (Graphic) As you see in this graphic, concerning the implementation in Gator, there are 4 layers:

- Current CCR scope (around 2000 autocallables), current BNPP scope (around 30 thousands autocallable), GPrime pricer and Gprime documentation.

The implementation in Gator that I made is at this level (GPrime documentation). However, strangely, not all variant described in Gprime documentation are implemented in Gprime pricer. So, for the test, we test at the Gprime pricer level.

--

We do the PV check over a generic autocallable with maturity of 2 and have 4 cancellation dates. Indeed, I re-used the same configuration as Yousra and Behnaz used last year. There are two reasons

1. The first one is that, the generic autocallable here seems a typical deal that may represent the autocallable portfolio
2. The second reason is that, I can check my results, not only with GPRIME pricer, but also with 2 Python pricers made by Yousra and Behnaz. The first one is by MC, the second one is closed form as mine.
   For information, for the case of In fine European barrier monitoring and phoenix coupon, I obtained the same results as the ones from the two Python pricers. The error is less that 0.01%

Now, for other variants of autocallables, we compared our results with GPRIME. The error metric used is the following (The formula)

## 9. Slide 9 –

As there are many variants of autocallable, I decomposed the pricer into 3 components Rebate, Coupon and Option. Each component, for each type of variant, I take one.

In total, we can have: 10 variant for Rebate, 12 variant for coupon and 13 variant for options, in total, we have 1560 autocallables.

The price of these autocallable is computed by our Gator pricer, and is compared with price from Gprime.

We have the following statistics

(Describe the error)

Given the fact that, the error tolerance for the pricer in CCR scope, is of 10% if the maturity less than 1 year and 10% multiplied by the maturity if the maturity is higher than 1 year. For our case, the error tolerance  is 20%.

## 10. Slide 10 –

We discussed about the exact formulas for autocallables and the accuracy test via PV check.

But what about the runtime? We need to add a second dimension “computation time” in the problem. And there are two answer: short and long.

For the short answer (read the slide)

For information, my personal desktop at home is at least twice faster than the laptop. I have 3.2Ghz, RAM 64 Gb and 8 cores. I bought it 3 years ago and it costed me 800€. I believe the server of BNPP for running the CCR calculation must at least 5 times faster than my personal desktop.

For the long answer, it is really long. And even is the main subject of the second topic “Fast computation of Multivariate Normal Probabilities”, This second topic, I’m pretty sure that it is not less interesting than the first topic of autocallables.

## 11. Slide 11 –

But before going to the second topic, we need to add a final slide to discuss more about the accuracy test via PV check, and then to emphasize the importance of the second topic

We observed some anomalies,

- Option component: with the feature Out of money put spread option (currently, on the CCR portfolio, 80 over 2000 autocallables have put spread option. Whether these put spread is out of money or not, I don't have information yet)

- Coupon component: Range accrual coupon with bearish cancellation (currently, there is no autocallable with this feature)

For the cases, errors may rise up to 10% or even 20% if we are really in out-of-money cases. For information, the test I did in previous slide, I didn't use out-of-money put spread option, that's why the errors are still small, in the order of 1%.

Indeed, it is not really a big problem **now**, but it may be a problem in the future if we don't know why these anomalies occurs.

--

We look at the error metric, this error depends on 2 prices: Gator price and GPRIME price.

The Gator price depend on 2 components

1. The first one is the exact formulas, based on the Base Formula 1. Normally , there is no computation error here
2. The second one is the Multivariate Normal Probabilities that we will talk in detail in the next slide. It depends on constraint on the computation time

The accuracy of GPRIME price depend no doubt on the number of MC scenarios used.

--

A better understanding of the GPRIME pricer for autocallables and MNP methods is necessary

## 12. Slide 12 –

Easy

## 13. Slide 13 –

The 5 method can be classified into 3 categories: Monte Carlo simulation, Machine Learning and Integration.

We begin with the first method that I called MinMatrixGenz. This method was invented by Genz in 1986. This method used to be fast, compared to other methods at the time. Its complexity is $\mathcal{O}(nK^2)$ But later, it becomes relatively slow compared to other later methods.

In 1992, Alan Genz invented a method, which I called Genz' method. This method is fast and exact, and it is the default choice for most of programming languages like Python, Matlab,…

In 2011, Cunningham invented a method, named EPMGP, which is based on a Machine Learning technique called Expectation Propagation. Like most of Machine Learning methods, this method is not proven. It is even a biased method. However, in practice, it can compute MNP extremely fast in most general cases.

In 2017, Botev improved the Genz 1992 method, by adding an optimization program. This improvement increase the speed of Genz method by around twice times.

In August 2021, in occasion of searching an adequate method for computing MNP for autocallable, I found some interesting properties of Wiener covariance matrices.

(Go to the next slide)

- These properties enable me to transform the multivariate probability in the form that I can used the Genz 1986 method.
- However, the Genz 1986 method is quite slow, with complexity of $\mathrm{O}(nK^2)$. This method can not compete with Genz, Botev or EPMGP methods.
- I improved the algorithm by applying the Fast Fourier Transform technique for the integration, that allows me to reduce the complexity to $\mathcal{O}(nK \ln K)$ and speed up at least 100 times the initial method, with a discretization of 0.5%. And the more accuracy is required, the higher speed gain is.
- I called this method MinMatrixFFT. This method is able to compete with Genz, Botev and EPMGP method.

## 14. Slide 14 –

To make things simple, these methods compute MNP from fast to very fast regarding to the computation speed and from sufficient to high regarding to the accuracy

The reality is much more complex. The order in term of accuracy performance and speed performance is not clear, it depends on many factors.

Each method can be illustrated by a curve in the plane  Accuracy-Speed

Show the graphic

The curve can be adjusted by parameter. For example, for Genz and Botev method, by increasing or decreasing the MC simulations, we can go from “fast but less accurate” to  “accurate but slow”. Same for MinMatrixFFT method.

The EPMGP is a particular method, that we can not adjust the accuracy.

## 15. Slide 15 –

As we can see now in the illustration, no method is uniformly the best for all cases. It depends on the scenarios.

For information, after the first study of MNP methods. Until end of July, EPMGP was still my first choice at it is very fast and sufficiently accurate for most of general cases. But after working on the autocallables, I discover that the covariance matrices of Brownian motion have a particular form that may reduce the accuracy of the EPMGP method.

In September, I run many tests to accessing the performance of the 5 methods (you can take a look at my second document relating to this topic). With error tolerance of $10^{-3}$, testing in Python, the MinMatrixFFT method is the best method in this scenario.

Indeed, from the tests I did up to now, I think that, methods have initial fixed cost and variable cost (which depend on the parameter of accuracy)

- For initial fixed cost: Genz method costs the least, after that MinMatrixFFT (because I need some setting for the Fast Fourier Transform)  and Botev (because of 1 optimization at the beginning of the calculation). I don’t know whether the fixed cost of MinMatrixFFT is higher or lower the Botev method
- For variable cost: I believe that MinMatrix variable cost is lowest.

This allows me to imply that, the higher accuracy we seek, the more likely the MinMatrixFFT method is the best.

## 16. Slide 16 –

Now, in the context of pricing of autocallable in CCR scope. Do we need an accuracy less than $10^{-3}$ for each MNP computation? I don't know, as the error metric on autocallable price is the cumulative sum of error of the MNP of this autocallable.

There is more likely that we can have higher error tolerance, as the error tolerance in CCR scope is high (I recall that the error tolerance is 10%, and may be higher for longer maturity contract).

By consequence, we may have other choice.

Explain the graphic

In the case 2, Botev's method will be the winner.

In the case 3: Gen' method is the best method, as its initial cost is lowest.

In the case 4: EMPGP method is the best method. As it is the fastest ever MNP method.

## 17. Slide 17 –

In this conclusion, I summarize the current state of the project and what we may need to do for the next steps.

Read the conclusion