Monte Carlo in Esperanto

A simple parser addition to your MC armory.

Krishna Kumar
Citigroup, New York, NY.

Abstract: This article shows how a simple parser environment in Excel/VBA could be used to perform single and multi-dimensional Monte Carlo. The clsMathParser is a class for math expression evaluation in Excel/VBA. We show that a host of vanilla and exotic derivatives can be priced easily in this framework. The parser allows separation of concern and natural language expression of structured product payoffs. We provide test results for 1-dimensional and multi-dimensional options and sample spreadsheets.

Keywords: *Monte Carlo, Code Reuse, Parser, Separation of Concerns, VBA, clsMathParser, Pathwise Derivatives.*

"Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one's subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects.

We know that a program must be correct and we can study it from that viewpoint only; we also know that is should be efficient and we can study its efficiency on another day [. . .] But nothing is gained—on the contrary—by tackling these various aspects simultaneously. It is what I sometimes have called "the separation of concerns"

Edsger W. Dijkstra

1 Introduction

Every few months there is a new thread on the Wilmott forum with a new spin on the age-old computer language debate. Although the thread title is different each time, C++ vs Fortran, Ada vs Java, the discussions are typical of the debate: heated and inconclusive.

Language preferences and superiority of one environment over the other are hard to decide. There has been an attempt in the past at benchmarking languages using standardized test suites [1], and more recently Bagley's shootout[1].

Today, there are about a dozen active programming languages in use, of which C, C++ and Java seem to dominate the Street. But in terms of reuse and rapid development tools there is much work still to be done.

2 Domain Specific Languages

Computer Scientists spend a significant amount of their time creating languages that are specific to a problem domain (Domain Specific Languages, or DSL's). A DSL is a specific formal language that is formed using a limited set of symbols with a set of predefined rules.

DSL's are similar to high-level languages, they often allow us to express in the operating lexicon of the problem domain, "mathematics". Although DSL remains an active area of research and it has been repeatedly shown to improve productivity and speed, DSL use remains rare. Although many banks appear to have made custom ports of high-level languages there is a disconnect between the model abstraction, conceptual basis and actual implementation.

In terms of high-level tools there are commercial products that provide a high-level language environment and "*automagically*" generate and compile code for a target language environment based on a domain-specific notation[2]. Although very promising, the average Quant developer/Financial Engineer is not likely to use these tools.

3 Monte Carlo Method and Separation of Concerns:

In this article we consider the Monte Carlo method for pricing Options, and using a simple parser show that a host of different options can be priced easily and with extensive reuse of our code. The approach utilizes the **clsMath** parser environment in Excel.

*I would like to thank Aaron Brown, Credit Risk Architecture, Morgan Stanley for his generous comments on a draft version of this article. I would also like to thank my colleague Tom Gladd for his comments.

Monte Carlo methods are extensively used in today's environment, and their popularity grows from their relative ease of use and an increase in computational power or clock speed. Any earlier reluctance to use it due to its slow convergence or a perceived inability to handle early exercise has by now been overturned.[3]

The term "Monte Carlo" was first coined by Metropolis, Phelem Boyle [2], was amongst the pioneers in applying it towards option pricing. The recent books of Jackel [3] and Glasserman [4] are excellent references to the method, besides Wilmott's earlier tome [5].

Apart from its ease of implementation, Monte Carlo can be categorized as an *"embarrassingly parallel problem"*, thus adding CPU's or moving the computation to a cluster will dramatically reduce computation time. We also show here that computationally it allows us to do separation of concerns[4] in a natural way.

3.1 Monte carlo method in Brief

The Monte Carlo method works by generating random samples from a domain and evaluating the function at these randomly generated points to obtain a numerical integral estimate of the function. It is convenient when the function is hard to evaluate analytically. The MC estimate of the integral will then converge to the true value as the sample size increases (this follows from the strong law of large numbers). We know from Martingale Pricing theory that the price of a derivative security is the expected value of the discounted payoff of the instrument under a uniquely determined martingale measure, and for a payoff that is a function of the terminal value of a single underlying we have:

$$C(T) = e^{-rT} \int \psi(z) f(z) dz$$

Here $f(z)$ *and* $\psi(z)$ are respectively the density of the underlying security and Payoff of the derivative. And so in terms of implementation of the Monte Carlo there are two aspects, the evolution of the underlying asset price and the application of the payoff on this evolution. And this naturally leads to a simple separation in the problem.

Consider the case of a simple European option that is exercised at fixed time "T" we have the payoff as $\{S(T)\text{-}K\}^+$.

To price this option we have the evolution of the underlying governed by the following SDE

$$\frac{dS(t)}{S(t)} = r\,dt + \sigma\,dW(t)$$

Now the solution of the above is

$$S(T) = S(0)\exp\left(\left[r - \frac{1}{2}\sigma^2\right]T + \sigma W(T)\right)$$

As $W(T)$ is normally distributed with mean 0 and variance T this can be written as

$$S(T) = S(0)\exp\left(\left[r - \frac{1}{2}\sigma^2\right]T + \sigma\sqrt{T}Z\right)$$

A high level Pseudo code for Monte Carlo can then be written as the following:

```
Step (1)  Generate Random Paths for the underlying Assets
Step (2)  Apply the Payoff on the Random Paths
Step (3)  Average the Payoff over the sample
```

For Step (1) we have choices. From amongst different RNG algorithms one could use Pseudo Random sequences or Quasi-Random sequences. Also one could then construct paths with these sequences using a simple Weiner Path or using a Brownian Bridge. Also the underlying paths could be from Variance Gamma process with Levy density [13].

Next we have to specify the Payoff function, which is typically a mathematical expression. By using a simple parser environment, we show that Step (2) can be defined and evaluated at runtime. This leads to code-reuse and flexible implementation.

But more importantly Step (1) and Step (2) are independent of each other. Step (1) could also be a simple Jump Diffusion model like the one proposed by Merton [8] and detailed in [4].

Merton's Jump process can be written as the following:

$$\frac{dS(t)}{S(t-)} = r\,dt + \sigma\,dW(t) + dJ(t)$$

In this model jumps are Compound Poisson Process and their inter arrival times are exponential. The Jumps themselves are lognormally distributed and are independent of the diffusion and the Jump process.

The process in Step (1) could also be one where the underlying is driven by a stochastic volatility model or a Garch type model.

3.2 Greeks by Pathwise Method[5]

The Pathwise derivatives have been proposed as an alternative to typical finite-difference methods that are used to compute the Greeks in the Monte Carlo framework.

Finite difference based Greeks are notorious for being noisy. Morrison suggests the following in GSL. (Gnu-Scientific Library)

> *"Construct a divided difference table with a fairly large step size to get a very rough estimate of f''. Use this to estimate the step size which will minimize the error in calculating f'"*

More recently Martins [12] illustrates the use of a complex step derivative to reduce the finite difference noise when computing derivatives[6].

In the context of Monte-Carlo, the optimal step-size "h" to compute the Greeks by bumping remains a difficult problem. This has been discussed in [4] and [9]. A large "h" introduces errors due to a non-linear payoff function and a small "h" induces a higher variance. This is further complicated with bias and computational cost from doing two separate runs to estimate the finite difference Greeks in Monte Carlo.

Pathwise Derivatives has been proposed as an alternative in [10] and details on this can also be found in [4] and [11].

>

The method works for smooth payoffs by using the payoff derivatives to estimate the price derivatives.

$$
\begin{aligned}
C(T) &= E[f(S(\theta))] \\
\frac{dC(T)}{d\theta} &= \frac{d}{d\theta}E[f(S(\theta))] \\
\frac{dC(T)}{d\theta} &= E\left[\frac{df(S(\theta))}{d\theta}\right]
\end{aligned}
$$

Essentially to compute the derivative with respect to an underlying parameter (asset price, volatility, interest rates) we simply move the derivative under the Payoff inside the expectation.

And the Delta and Vega for a vanilla European option can be derived as:

Delta:

$$\Delta = e^{-rT} 1_{\{S(T)\geq K\}} \frac{S(T)}{S(0)}$$

Vega:

$$\nu = e^{-rT} 1_{\{S(T)\geq K\}} S(T)(-\sigma T + \sqrt{T}Z)$$

Where $1_{\{S(T)\geq K\}}$ is an indicator function.

Further the Pathwise derivatives can be computed independent of the asset price process S(T). This allows us to do separation of concerns. We can now generate the S(T) using our favorite volatility model and/or Jump diffusion model and still use the above expressions to compute the Greeks.

Exhibit - 1

```
Function MC_parse1d(payoffstring,S0,v,t,r,d) As Double

Dim ST, volTime As Double
volTime = v * Sqr(t)
'##loop counter
 Dim indx As Long
 '##running sum holder
 Dim avgP,drift As Double
 Const NUM_SIMS As Double = 10000
 drift = (r * t - 0.5 * volTime ^ 2)
Dim Fun As New clsMathParser
 If Not Fun.StoreExpression(payoffstring) Then GoTo Error_Handler

On Error GoTo Error_Handler

For indx = 1 To NUM_SIMS
 dr_ = myGauss(0, 1) '##a normal RNG
 ST = S0 * Exp(drift + volTime * dr_)
 Fun.VarSymb("S")= ST
 Value_F = Fun.Eval
  avgP = avgP + Value_F
 Next indx

 MC_parse1d = avgP / NUM_SIMS
 Exit Function

 Error_Handler:
 Debug.Print Err.Source, Err.Description

End Function
```

3.3 Math Parser environment for Excel

The clsMathParser is a numeric evaluator that can be embedded within Excel/VBA that allows us to input and evaluate any numerical expression at runtime. The parser recognizes physical expressions, constants and international units of measure. For details on the parser c.f. [6].

As we described above the payoff function is typically a mathematical expression. Again a European call option is **Max(S-X, 0)** and a put option is **Max(X-S, 0).** And a Call spread is defined as

$$\text{if } S_T >= X, \min\left(U, \max\left(\frac{S_T - S_0}{S_0}, 0\right)\right) else\ 0$$

A host of 1-dim options can be priced using the code in Exhibit-1.

Similarly for a 2-d option we have a simple spread option defined as: **Max([S1-S2-X], 0)**

A weighted 2-asset basket option payoff defined as: **Max([w1*S1+ w2*S2], 0)** etc.

The 1-d parser code that can be used to price several 1-d payoff types. The payoff expression is passed to the function as an argument.

In Exhibit-1 the 1-dim paths are generated using a Gaussian random number generator in line 17, the parser is initialized in line 11 and evaluated in lines 19,20.

4 Numerical Test Results

In this section we present some test results using the above framework.

4.1 1-dim Options

4.1.1 Marks

	Payoff String	MC value	Analytical Value
Vanilla Call	(S<=110)*0+(S>110)*(S-110)	6.25	6.19
Vanilla Put	(S>=110)*0+(S<110)*(110-S)	16.15	16.19
Vanilla Call	max(S-110,0)	6.20	6.19
Call Spread	10000*min(10,max(S/100-1, 0))	995.89	
Quadratic option?	max(S^2-S-10000,0)	2315.79	
Asset or nothing call	(S<110)*0+(S>=110)*S	40.19	39.8882
Cash or nothing put	(S<110)*110+(S>=110)*0	76.41	76.3022
Gap Call Option	(S>110)*(S-120)	2.27	
Gap Put Option	(S<110)*(120-S)	28.19	
Supershares	(100<=S<=125)*S/90	0.39	

4.1.2 Greeks by Pathwise Method[7]

European call	Payoff String	MC Greeks
Delta	(S>90)*(S/100)	**0.71**
Vega	AND(((log(S/100)-(0.5*0.25^2))/0.25)*S,S>110)	**0.37**

4.2 Barriers

Stock price	100.0	S	H	120
Strike	110.0	X	L	80
Years to maturity	1.00	T	**K**	0
Risk-free rate	0.00%	r		
Volatility	25.0%	v		
Div yield	0	d		
Time steps	100		nstep	1000

>

4.2.1 Marks

	Payoff String	Barrier Indicator	MC value	Closed Form
up and out	max(S-110,0)	AND(S<120,1)	**0.16**	**0.13**
down and out	max(S-110,0)	AND(S>80,1)	**6.04**	**6.09**
up and in	max(S-110,0)	OR(S>80,0)	**5.71**	**6.06**
down and in	max(S-110,0)	OR(S<80,0)	**0.09**	**0.10**

4.3 2-dim Options

Asset Price 1	100.0	div1	—
Asset Price 2	90.0	div2	—
Strike		Rho	0.50
Years to maturity	1.00		
Risk-free rate	0.00%		
Volatility 1	20.00%		
Volatility 2	25.0%		

4.3.1 Marks

Option Type	Payoff String	Monte carlo price
Spread Option	`max(S1-S2-5,0)`	4.996640747
Multi-strike call/max	`max(0,max(S1-80,S2-70))`	37.25773123
Best of	`(S1/100>S2/90)*(S1-90)+(S1/100<S2/90)*(S2-80)`	29.64422334
Worst of	`(S1/100<S2/100)*(S1-90)+(S1/100>S2/90)*(S2-80)`	6.979293785
Basket Call	`max(0,(S1+S2)-90)`	125.5420128
Basket Put	`max(0,140-(S1+S2))`	0.381792605
Weighted Basket	`max(0,(0.7*S1+0.3*S2)-90)`	14.13227
Multi Strike Put/Min	`max(0,max(120-S1, 130-S2))`	33.53047215
Quantos[8]	`max(S1-1000/S2,0)`	93.83028718

4.3.2 Greeks

	Payoff String	Greeks
Spread Option		
Delta 2	`-(S1/100)*(S2-S1>5)`	−0.491338145
Delta 1	`(S2/90)*(S2-S1>5)`	0.800314928
Max Option		
Delta 1	`(S1/100)*AND(S1>S2,S1>80)`	0.311265346
Delta 2	`(S2/100)*AND(S2>S1,S2>80)`	0.754868842

5 Summary

In this paper we have shown that using a simple math parser one can price and compute Greeks for a host of different payoff functions using a single piece of code. Further as the MC method is amenable for separation of concerns this leads to a natural framework, with reuse and the possibility of rapid development for new payoff functions. It is possible to handle path dependent options (lookbacks, barriers) with a little effort and alternate processes (VG) in this framework.

FOOTNOTES & REFERENCES

1. *http://shootout.alioth.debian.org/*
2. See Scicomp (*http://www.scicomp.com*) for this. The claim for the earliest MC parser tool used in finance appears to be "Derivatool" from FEA *http://www.fea.com*
3. In an interesting aside the earlier edition of a popular book claimed that it was not possible to price American-style options using MC and a more recent edition has updated and discusses the Least Squares approach of Longstaff and Schwartz.
4. "**Separation of concerns** (SoC) is the process of breaking a program into distinct features with no overlap in functionality. Typically, concerns are synonymous with features or behaviours or stages", Refer [7] for additional details.
5. This is known as infinitesimal perturbation analysis.
6. Excerpted from GSL source code file "diff.c", Axel Voight pointed this to me.
7. Note that Pathwise derivatives are not calculable when the payoff is discontinuous in the underlying parameter. (Digitals and Barriers)
8. S2 here is any exchange rate.

■ http://www.cowell-shah.com/research/benchmark/code
■ Boyle, Phelem P. (1977). "Options: A Monte Carlo Approach." Journal of Financial Economics.
■ Jackel, Peter (2003). Monte Carlo Methods in Finance, Wiley.
■ Glasserman, Paul (2004). Monte Carlo Methods in Financial Engineering, Springer Verlag.
■ Wilmott, Paul (2000). Paul Wilmott on Derivatives, Wiley.
■ Volpi, Leonardo (2005). ClsMathParser—A Class for Math Expressions Evaluation in Visual Basic.
■ http://en.wikipedia.org/wiki/Separation_of_concerns
■ Merton, R.C. (1976). Option pricing when underlying stock returns are discontinuous, Journal of Financial Economics 3:125–144.
■ Glynn, P.W, Optimization of stochastic systems via simulation, Proceedings of the 21st conference on Winter simulation, p. 90–105.
■ Ho, Y. C., Cao, X. Perturbation analysis and optimization of queueing networks. J. Optim. Theory Appl. 40, 4 (Aug. 1983), 559–582.
■ Broadie, M, Glasserman, P. Estimating Security Price Derivatives Using Simulation, Management Science, 1996, Vol 42, No 2, 269–285.
■ J. R. R. A. Martins, I. M. Kroo, and J. J. Alonso. An automated method for sensitivity analysis using complex variables. AIAA Paper 2000–0689, Jan. 2000.
■ Webber, Nick, Riberio, Claudia. Valuing Path Dependent Options in the Variance-Gamma Model by Monte Carlo with a Gamma Bridge.