



# An Introduction to Monte Carlo Methods

Nicholas Mocciolo  
February 2010

# Agenda

---

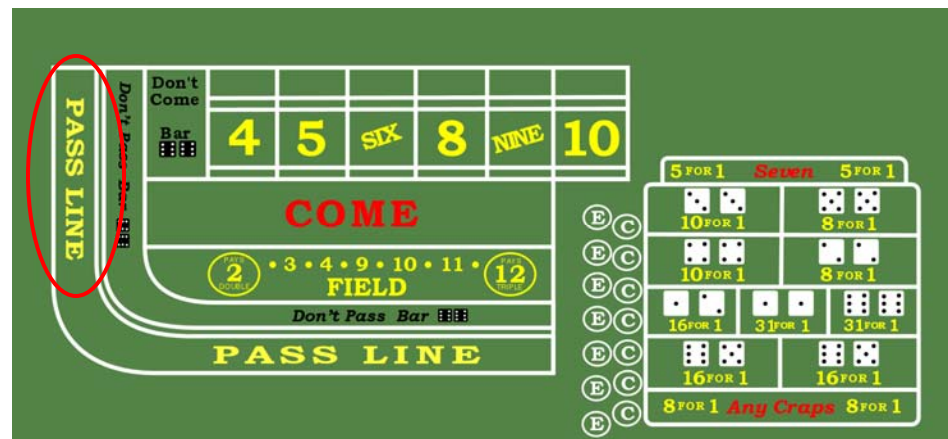
- Background
- Craps and the “Pass Line”
- Random number generation
- Applications of Monte Carlo methods in finance and insurance
- Simulation of stock prices under GBM
- Monte Carlo examples
- Advantages and disadvantages of Monte Carlo methods for option valuation
- Variance reduction techniques
- Monte Carlo methods and American options
- Conclusion

# Monte Carlo Methods – Background<sup>1</sup>

- A class of computational algorithms that rely on repeated sampling to compute their results
  - Generally used when it is infeasible or impossible to compute an exact result
- The term “Monte Carlo method” was coined in the late 1940s by physicists working on nuclear weapons in the Los Alamos National Laboratory
  - The project required a code name to secure its secrecy, and John Von Neumann chose “Monte Carlo” as a reference to the Monte Carlo Casino in Monaco where a fellow mathematician’s uncle had become a regular
- However, random methods of computation and experimentation can be traced back to early work on probability theory in the pre-electronic-computing era, using tables of random numbers for statistical sampling
  - It is thought that the earliest work on probability theory might have been done in the study of games of chance
- Applications are plentiful, and include the physical sciences, design and visuals, finance and business, telecommunications, meteorology, and artificial intelligence

## Example of Monte Carlo Simulation - Craps

- Craps is a popular dice game in which players wager on the outcome of a roll or a series of rolls of two dice
  - Thought to have been developed by the French around the time of the Crusades, and later brought to New Orleans by Bernard Xavier Philippe, a wealthy Louisiana landowner and politician
- Many types of bets can be made on a Craps table, but perhaps the most popular is the “pass line” bet
  - The initial (“come out”) roll is made, at which time a 2, 3, or 12 results in a loss and a 7 or 11 results in a win
  - Any other roll is referred to as the “point,” and the roll continues until either the point number is rolled again (a win), or a 7 is rolled (a loss)



# Example of Monte Carlo Simulation – The Pass Line



- One might wish to estimate the “house edge” on a Pass Line bet; that is, the statistical expectation of this wager

```
Public Function RollPairOfDice(x As Double) _
    As Integer

Select Case x
    Case Is >= 35 / 36
        RollPairOfDice = 12
    Case Is >= 33 / 36
        RollPairOfDice = 11
    Case Is >= 30 / 36
        RollPairOfDice = 10
    Case Is >= 26 / 36
        RollPairOfDice = 9
    Case Is >= 21 / 36
        RollPairOfDice = 8
    Case Is >= 15 / 36
        RollPairOfDice = 7
    Case Is >= 10 / 36
        RollPairOfDice = 6
    Case Is >= 6 / 36
        RollPairOfDice = 5
    Case Is >= 3 / 36
        RollPairOfDice = 4
    Case Is >= 1 / 36
        RollPairOfDice = 3
    Case Else
        RollPairOfDice = 2
End Select

End Function

Public Function PassLineExpectation(Trials As Long, Seed As Long) As Double

Dim i As Long

Dim ComeOutRoll As Integer, Point As Integer, Till As Long, Roll As Integer

'Initialize and Seed the RNG
Dim mt As Object
Set mt = CreateObject("Mersenne.Twister")
mt.Initialize (Seed)

For i = 1 To Trials

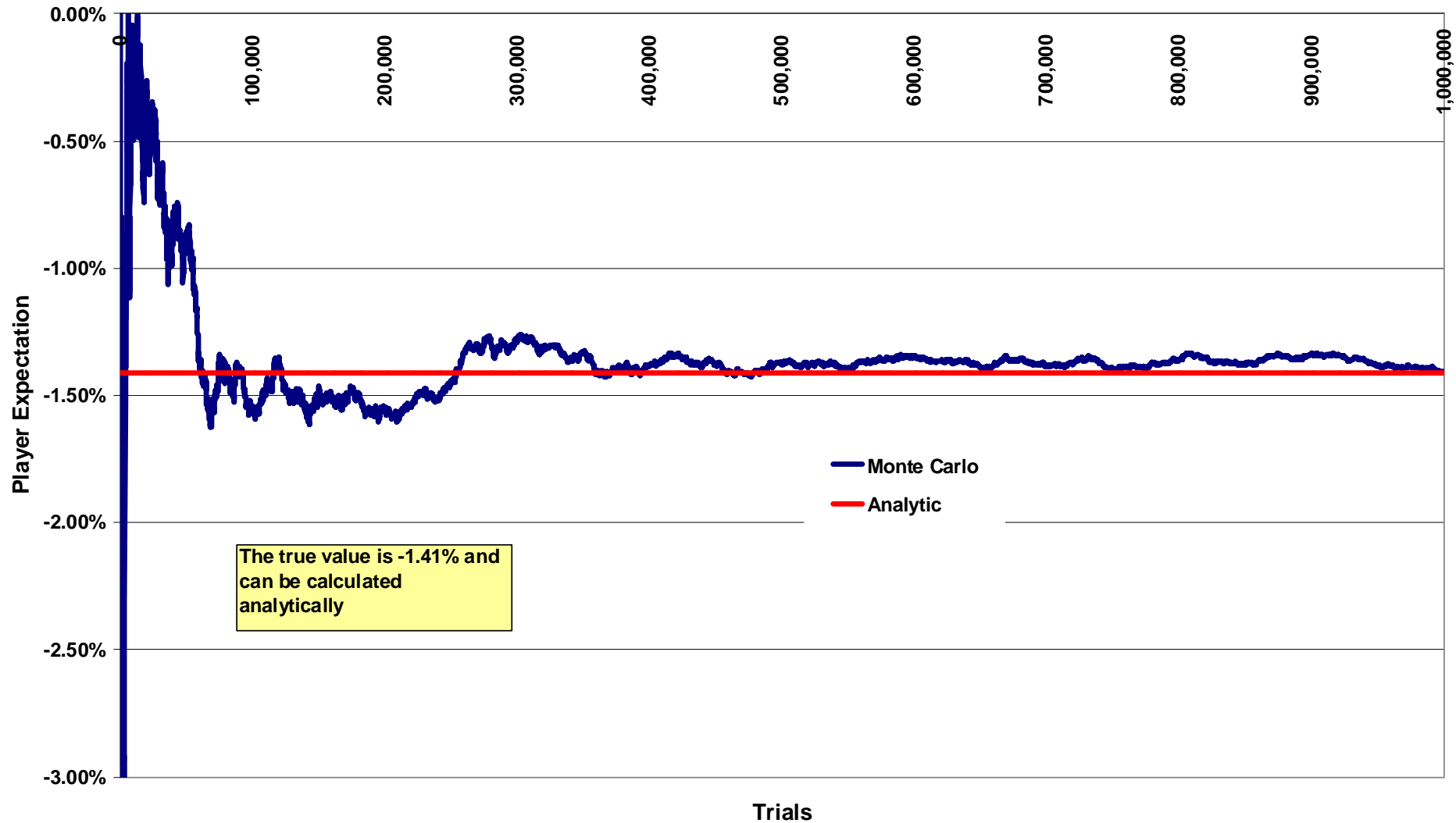
    ComeOutRoll = RollPairOfDice(mt.real3()) ' the come out roll

    Select Case ComeOutRoll
        Case 2, 3, 12 ' 2, 3, and 12 are front line losers
            Till = Till - 1
        Case 7, 11 ' 7 and 11 are front line winner
            Till = Till + 1
        Case Else
            Point = ComeOutRoll ' otherwise establish a point
            Do ' now keep rolling until you re-roll the point (win)
                ' or until a 7 is rolled (loss)
                Roll = RollPairOfDice(mt.real3())
                If Roll = 7 Then
                    Till = Till - 1
                    Exit Do
                End If
                If Roll = Point Then
                    Till = Till + 1
                    Exit Do
                End If
            Loop
        End Select

    PassLineExpectation = Till / Trials ' calculate expectation

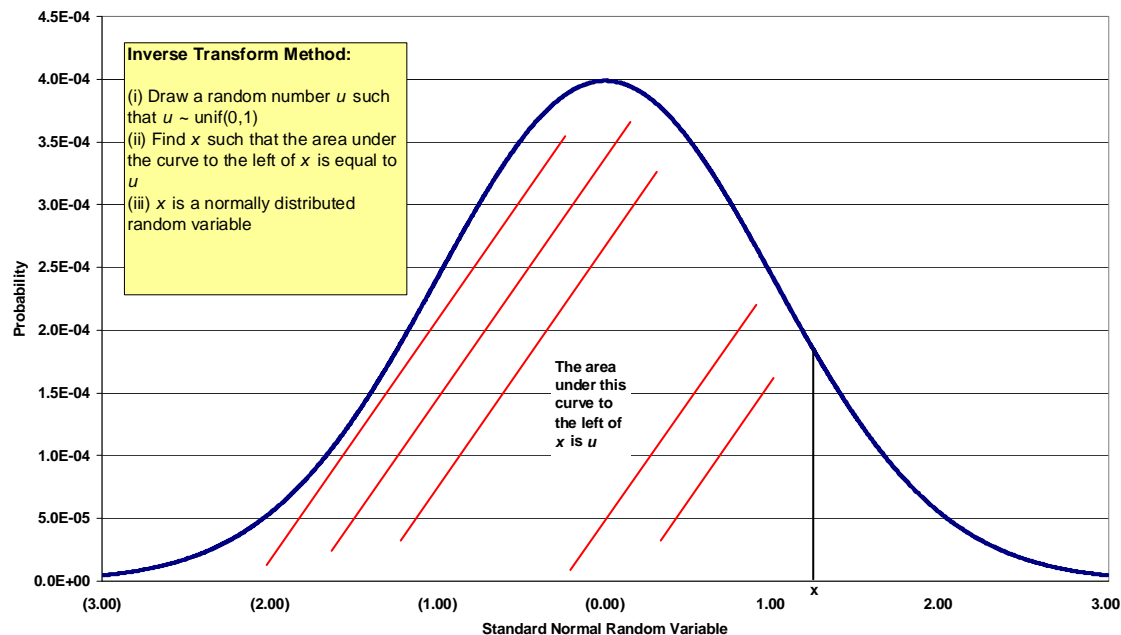
End Function
```

# Example of Monte Carlo Simulation – The Pass Line



# Random Number Generation

- As modern Monte Carlo methods use computers to repeatedly sample the behavior of random phenomena, practitioners need computer algorithms that are capable of producing “random” number streams
  - Such an algorithm is referred to as a random number generator (“RNG”), or more aptly, as a pseudorandom number generator (“PRNG”)
    - Numbers produced by PRNGs are actually not random at all, but deterministic
- Most PRNGs produce values that strive to be uniformly distributed between 0 and 1, since random numbers from any desired distribution can then be obtained by passing the  $\text{unif}(0,1)$  values through the chosen distribution’s CDF



# Random Number Generation

- Researchers have strived to develop PRNGs with the following desirable attributes:
  - Efficient: capable of producing many pseudorandom values quickly
  - Deterministic: results can be reproduced using a “seed”
  - Low periodicity: many values can be generated before the sequence repeats
  - High dimensionality: can draw tuples without distorting pseudorandom qualities
- Ran a series of tests on the PRNGs embedded in Excel, as well as a PRNG called “The Mersenne Twister,” which is widely regarded as one of the best generators available today

Test Name	RNG		
	Excel RAND()	VBA Rnd()	MT 19937
Basic Statistical Test	Fail	Pass	Pass
Monte Carlo Estimate of Pi	Pass	Pass	Pass
1-Dimensional Collision Test	Pass	Fail	Pass
Roulette Test	Pass	Pass	Pass
2-Dimensional Collision Test	Fail	Fail	Pass

- Has a period of  $2^{19937}$  and can generate “bias-free” values in 623 dimensions
- More info available at <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>

“Garbage in, garbage out”

# Monte Carlo Simulation in Finance and Insurance

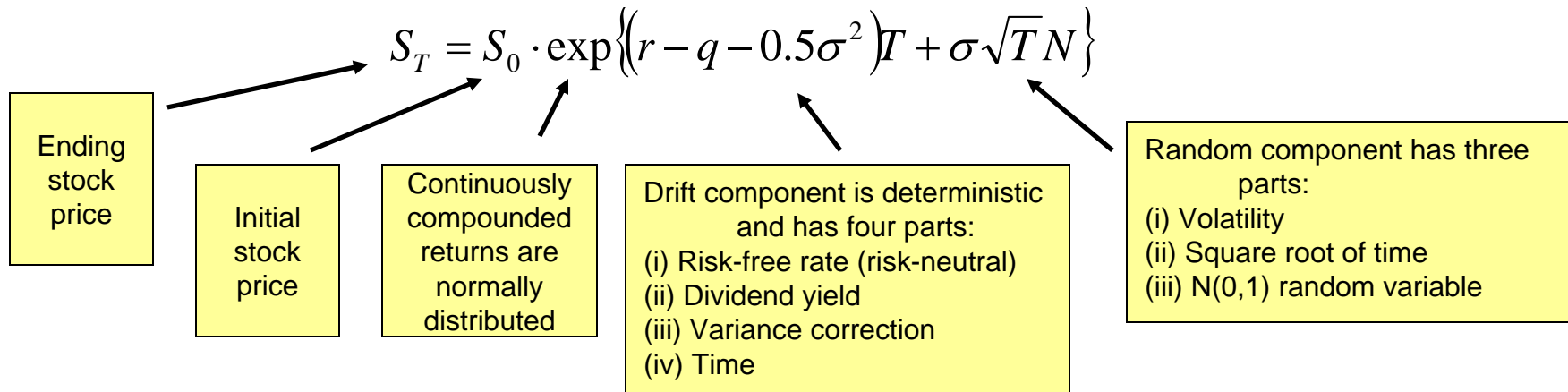
- General applications of Monte Carlo methods in financial services
  - Risk (real-world)
    - VaR, CVaR, CTE, PFE
  - Real options and capital budgeting
    - Contraction, expansion, abandonment
  - Asset allocation
    - Multi-period optimization with constraints
  - Personal financial planning
    - Probability of shortfall
  - Option valuation (risk-neutral)
    - Equity, credit, interest rate, F/X, commodities, cross-asset, GMxB
    - Pioneered by Phelim Boyle (an actuary!) in 1977

To illustrate the methods, we will focus the remainder of the presentation on this application, since analytic solutions are available for comparison and since it is a nice analog for hedging work

# Simulation of Stock Prices – Geometric Brownian Motion (“GBM”)



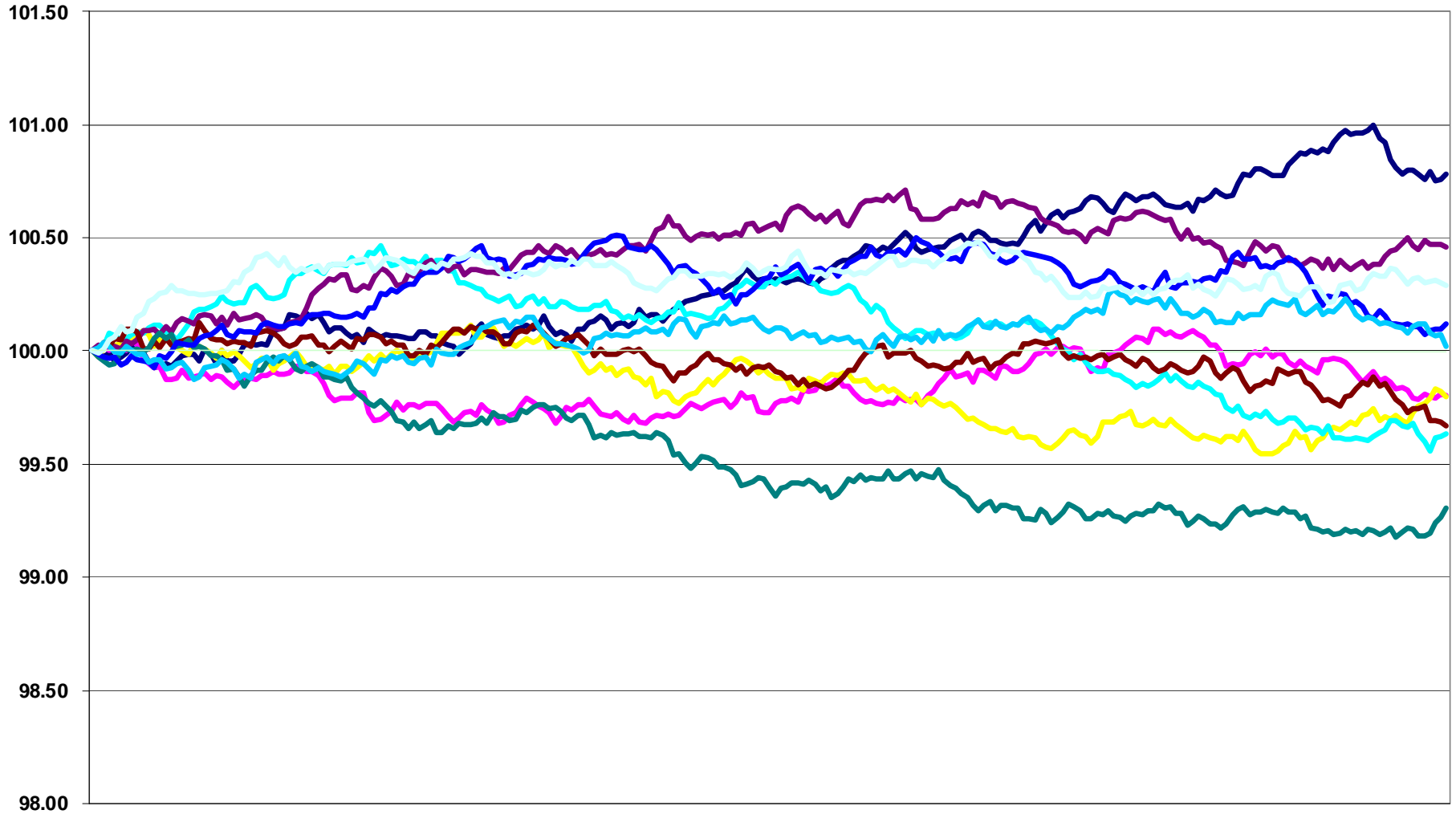
- Let Black-Scholes assumptions govern
  - Deterministic interest rates ( $r$ ), dividend yields ( $q$ ), and volatility ( $\sigma$ )
- We simulate the value of a stock price at a future time  $T$  by using the equation



- Some heuristics (to avoid a lesson on Ito calculus):
  - The volatility correction comes from the continuously compounded nature of returns, which weighs “good” returns more than “bad” ones (i.e.  $\exp(0.10) = 1.1052$  and  $\exp(-0.10) = 0.9048$ , so the average return is actually 0.50%)
  - In GBM, volatility scales with the square root of time (quadratic variation)

# Simulation of Stock Prices – “Spaghetti Graph”

## 10 Paths for a Stock Price Over a 1-hr Interval



# Valuation of a Vanilla Put Option Via Monte Carlo Simulation Under GBM



- Simulate a “sufficient” number of random paths,  $P$ , for the underlying stock
- For each path, calculate the discounted option payoff at maturity,  $f_i$
- Average the payoffs from all paths to estimate the option value,  $f^e$
- Can also compute a standard error for your estimate,  $\text{stdev}(f_i) / \sqrt{P}$

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													

Inputs				
S(0)	100.00			
K	100.00			
T	1			
r	5.00%			
q	2.00%			
σ	30.00%			

Annual Time Steps			
Path	N	S(T=1y)	Put Payoff
0	(0.655)	80.94	19.06
1	1.968	177.77	0.00
2	0.361	109.76	0.00
3	(0.215)	92.36	7.64
4	0.392	110.81	0.00
5	(3.589)	33.57	66.43
6	1.334	146.97	0.00
7	1.913	174.89	0.00
8	0.328	108.70	0.00
9	(0.615)	81.91	18.09
10	0.727	122.52	0.00

Semi-Annual Time Steps					
Path	N1	N2	S(T=6m)	S(T=1y)	Put Payoff
0	0.018	0.078	99.623	100.54	0.00
1	(0.334)	(1.065)	92.461	73.22	26.78
2	1.194	1.189	127.853	163.29	0.00
3	0.789	(2.624)	117.324	66.73	33.27
4	0.282	0.211	105.368	109.36	0.00
5	(0.746)	1.008	84.733	104.15	0.00
6	(0.718)	(0.932)	85.229	69.42	30.58
7	0.268	1.384	105.063	139.85	0.00
8	0.244	(0.443)	104.529	94.45	5.55
9	1.378	1.647	132.963	187.16	0.00
10	0.580	0.572	112.248	125.79	0.00

Simulated Put Value	
9.62*	8.32

Formulas	
=AVERAGE(E12:E22)*EXP(-C6*C5)	=J17*EXP((C6-C7-0.5*C8*C8)*(C5/2)+C8*SQRT(C5/2)*I17)

Can't do this under stochastic discounting!

# Code Snippet for the Valuation of a Vanilla Option Via Monte Carlo Simulation Under GBM - VBA



```
Public Function MonteCarloVanillaOptionValue(Flag As String, _
    S0 As Double, K As Double, T As Double, r As Double, _
    q As Double, v As Double, Paths As Long, Steps As Long, _
    Seed As Long) As Variant

Dim i As Long, j As Long

'Initialize and Seed the RNG
Dim mt As Object
Set mt = CreateObject("Mersenne.Twister")
mt.Initialize (Seed)

Dim dt As Double
dt = T / Steps ' the length of the time step

Dim sqrtdt As Double
sqrtdt = Sqr(dt) ' the square root of the time step

Dim rndrft As Double
' risk-neutral drift
rndrft = (r - q - 0.5 * v * v) * dt

Dim rnshck As Double
' risk-neutral volatility over one time step
rnshck = v * sqrtdt

Dim DF As Double
' discount factor
DF = Exp(-r * T)

Dim f() As Double
ReDim f(1 To Paths)

Dim S() As Double
```

```
Dim N01 As Double, mu As Double
For i = 1 To Paths ' loop through the paths

    ReDim S(0 To Steps)
    S(0) = S0
    For j = 1 To Steps
        ' draw a random number and convert it into a standard normal
        N01 = MoroNormSInv(mt.real3())
        ' simulate the asset price at the new time step
        S(j) = S(j - 1) * Exp(rndrft + rnshck * N01)
    Next j

    ' calculate the payoff
    If Flag = "p" Then
        f(i) = Application.Max(0, K - S(Steps)) * DF
    Else
        f(i) = Application.Max(0, S(Steps) - K) * DF
    End If

    mu = mu + f(i)

Next i
mu = mu / Paths ' estimate option value

Dim sd As Double
' calculate the standard error of the estimates
For i = 1 To Paths
    sd = sd + (mu - f(i)) * (mu - f(i))
Next i
sd = sd / (Paths - 1)
sd = Sqr(sd)
sd = sd / Sqr(Paths) ' divide standard error by square root of paths

Dim lb As Double, ub As Double
' lower bound and upper bound of 90% confidence interval
lb = mu - sd * 1.96
ub = mu + sd * 1.96

' output desired values
MonteCarloVanillaOptionValue = Array(mu, sd, lb, ub)

End Function
```

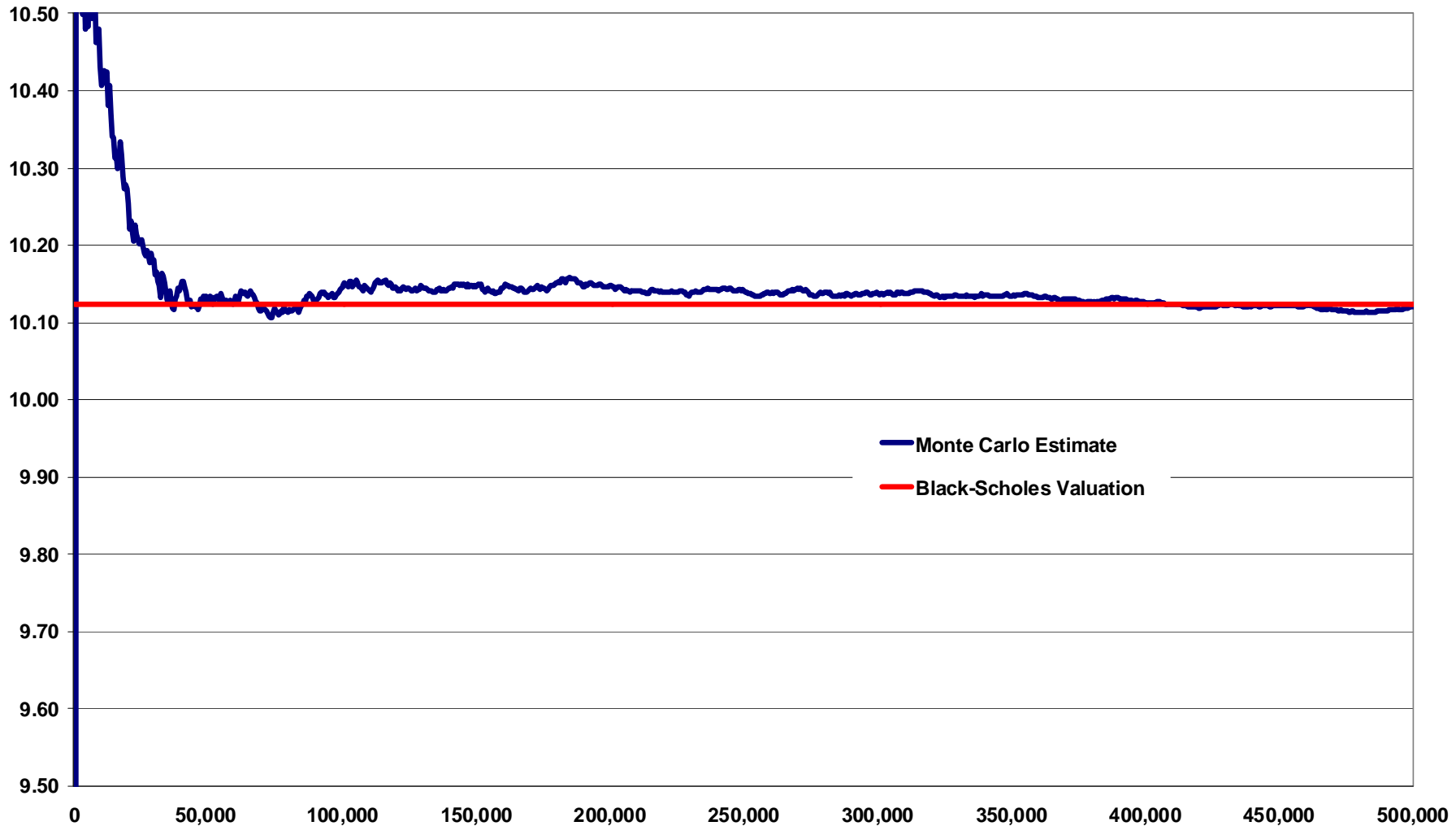
# Put Option Valuation Via Monte Carlo Simulation - Results



- Economic conditions:
  - $S = 100$ ,  $r = 5\%$ ,  $q = 2\%$ ,  $\sigma = 30\%$
- Option features:
  - Put option,  $K = 100$ ,  $T = 1$
- Black-Scholes value = \$10.12
- Monte Carlo results (1 time step per path, seed of 23):

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	10.77	1.55	7.74	13.81
500	10.66	0.63	9.42	11.89
1,000	10.78	0.43	9.93	11.63
5,000	10.48	0.19	10.11	10.86
10,000	10.41	0.14	10.14	10.67
25,000	10.21	0.09	10.04	10.37

# Vanilla Put Option Valuation Via Monte Carlo Simulation – Results for 500,000 Paths



# Down-And-Out Put Option Valuation Via Monte Carlo Simulation - Results



- Economic conditions:
  - $S = 100$ ,  $r = 5\%$ ,  $q = 2\%$ ,  $\sigma = 30\%$
- Barrier option features:
  - Put option,  $K = 100$ ,  $H = 80$ , rebate = 0,  $T = 1$ , monthly observations
- Analytic value = \$1.41
- Monte Carlo results (12 time steps per path, seed of 17):

```
ReDim S(0 To Steps)
S(0) = S0
For j = 1 To Steps
  If RNG = "Twister" Then
    ' draw a random number and convert it into a standard normal
    N01 = MoroNormSInv(mt.real3())
  Else
    ' draw a random number and convert it into a standard normal
    N01 = MoroNormSInv(Rnd())
  End If
  S(j) = S(j - 1) * Exp(rndrft + rnshck * N01)
  If S(j) <= H Then
    f(i) = rb * Exp(-r * (j * dt))
    Exit For
  End If
Next j

If j = Steps + 1 Then
  If Flag = "p" Then
    f(i) = Application.Max(0, K - S(Steps)) * DF
  Else
    f(i) = Application.Max(0, S(Steps) - K) * DF
  End If
End If
```

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	1.76	0.45	0.88	2.63
500	1.68	0.18	1.33	2.04
1,000	1.54	0.12	1.30	1.78
5,000	1.51	0.06	1.40	1.62
10,000	1.49	0.04	1.42	1.57
25,000	1.43	0.02	1.38	1.48

# Multi-Asset Options: A Call on the Max of Two Risky Assets



- Economic conditions:
  - $S1 = 100$ ,  $S2 = 90$ ,  $r = 6\%$ ,  $q1 = 3\%$ ,  $q2 = 1\%$ ,  $\sigma1 = 30\%$ ,  $\sigma2 = 40\%$
- Option features:
  - Call on the max,  $K = 110$ ,  $\rho = 35\%$
- Analytic value = \$15.35
- Monte Carlo results (1 time steps per path, seed of 13):

```

Dim rndrft1 As Double
rndrft1 = (r - q1 - 0.5 * v1 * v1) * dt

Dim rndrft2 As Double
rndrft2 = (r - q2 - 0.5 * v2 * v2) * dt

Dim sqrlmrhorho As Double
sqrlmrhorho = Sqr(1 - rho * rho)

ReDim S1(0 To Steps), S2(0 To Steps)
S1(0) = S01
S2(0) = S02
For j = 1 To Steps
    ReDim N01(1 To 2)
    N01(1) = MoroNormSInv(mt.real3())
    N01(2) = N01(1) * rho + sqrlmrhorho * MoroNormSInv(mt.real3())
    S1(j) = S1(j - 1) * Exp(rndrft1 + rnshck1 * N01(1))
    S2(j) = S2(j - 1) * Exp(rndrft2 + rnshck2 * N01(2))
Next j

f(i) = Application.Max(0, Application.Max(S1(Steps), S2(Steps)) - K) * DF

mu = mu + f(i)
    
```

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	15.83	2.59	10.75	20.91
500	16.51	1.24	14.09	18.94
1,000	16.07	0.87	14.37	17.77
5,000	15.97	0.38	15.22	16.71
10,000	15.70	0.26	15.18	16.21
25,000	15.45	0.16	15.13	15.77
50,000	15.31	0.11	15.09	15.54
100,000	15.40	0.08	15.24	15.56

# Advantages and Disadvantages of Monte Carlo Methods for Option Valuation



- Advantages
  - Complexity is low relative to competing methods (e.g. PDE)
  - Can easily accommodate path-dependency in payoffs
  - More efficient than competing methods for higher-dimensional problems
  - Can calculate a standard error of the valuation estimate
  - Easier to parallelize for grid-computing than competing methods
- Disadvantages
  - Slower than competing methods for low-dimensional problems
  - Difficult to use for the valuation of options with early exercise features
  - Convergence scales with the square root of paths, so that if one wants to reduce simulation error by a factor of 10, one must increase the number of paths by a factor of 100

# Variance Reduction Technique #1: Antithetic Variates



- Heuristic premise:
  - Unbiased samples from a normal distribution should have a mean of 0
  - We should be able to reduce simulation error by enforcing this property explicitly in the sampling
- Procedure:
  - If we are running an odd-numbered path, proceed as usual
  - If we are running an even-numbered path, then take the negative of the normal random variables sampled in the prior path
  - This will automatically force the mean of the sampled variables to zero, and will create a “symmetry” in sampled normal variables
- Can be a very powerful variance reduction technique in many applications
- May actually worsen results when the payoff is a non-monotonic function of the underlying asset

# Variance Reduction Technique #1: Antithetic Variates



- Example of antithetic variates implementation in VBA:

```

If i Mod 2 = 1 Then
  ReDim N01(1 To Steps)
  For j = 1 To Steps
    ' draw a random number and convert it into a standard normal
    N01(j) = MoroNormSInv(mt.real3())
    S(j) = S(j - 1) * Exp(rndrft + rnshck * N01(j))
  Next j
Else
  For j = 1 To Steps
    N01(j) = -N01(j)
    S(j) = S(j - 1) * Exp(rndrft + rnshck * N01(j))
  Next j
End If
  
```

- Vanilla put valuation with antithetic variates:

**Brute Force Monte Carlo**

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	10.77	1.55	7.74	13.81
500	10.66	0.63	9.42	11.89
1,000	10.78	0.43	9.93	11.63
5,000	10.48	0.19	10.11	10.86
10,000	10.41	0.14	10.14	10.67
25,000	10.21	0.09	10.04	10.37

**Antithetic Variates**

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper	S.E. Reduction
100	11.02	1.06	8.95	13.10	32%
500	10.28	0.42	9.46	11.10	33%
1,000	10.34	0.29	9.77	10.91	34%
5,000	10.25	0.13	10.00	10.50	33%
10,000	10.22	0.09	10.05	10.40	34%
25,000	10.20	0.06	10.09	10.31	34%

- Barrier option valuation with antithetic variates:

**Brute Force Monte Carlo**

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	1.76	0.45	0.88	2.63
500	1.68	0.18	1.33	2.04
1,000	1.54	0.12	1.30	1.78
5,000	1.51	0.06	1.40	1.62
10,000	1.49	0.04	1.42	1.57
25,000	1.43	0.02	1.38	1.48

**Antithetic Variates**

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper	S.E. Reduction
100	1.42	0.32	0.79	2.05	28%
500	1.54	0.16	1.23	1.84	14%
1,000	1.55	0.11	1.32	1.77	8%
5,000	1.45	0.05	1.36	1.55	9%
10,000	1.41	0.04	1.34	1.48	9%
25,000	1.41	0.02	1.37	1.46	7%

# Variance Reduction Technique #2: Moment Matching



- Heuristic premise:
  - Why stop at matching the mean of the sampled normal variables?
  - We can explicitly match the first three (or more) moments of the sampled normal variables to the (known) moments of the standard normal distribution
  - Often done in conjunction with antithetic variates since odd moments are automatically matched
- Procedure:
  - Generate and store unadjusted normal random variables using antithetic variates
  - Calculate standard deviation of unadjusted draws and re-normalize all variables
  - Three moments are now matched, the fourth is sometimes matched as well
- Total variance reduction achieved versus the antithetic method alone depends upon the nature of the underlying option being valued
- Consumes significant memory, as all generated normal variables must be retained in order for the re-normalization to be performed

# Variance Reduction Technique #2: Moment Matching



- Example of moment matching implementation in VBA:

```
Dim N01() As Double
ReDim N01(1 To Paths, 1 To Steps)
For i = 1 To Paths
    If i Mod 2 = 1 Then
        For j = 1 To Steps
            N01(i, j) = MoroNormSInv(mt.real3())
        Next j
    Else
        For j = 1 To Steps
            N01(i, j) = -N01(i - 1, j)
        Next j
    End If
Next i
```

```
Dim tsd As Double
For i = 1 To Paths
    For j = 1 To Steps
        tsd = tsd + N01(i, j) * N01(i, j)
    Next j
Next i
tsd = tsd / Paths / Steps
tsd = Sqr(tsd)
```

```
For i = 1 To Paths
    For j = 1 To Steps
        N01(i, j) = N01(i, j) / tsd
    Next j
Next i
```

- Vanilla put valuation with moment matching:

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper	S.E. Reduction
100	9.44	0.93	7.62	11.26	40%
500	9.87	0.40	9.08	10.66	36%
1,000	10.03	0.28	9.48	10.58	35%
5,000	10.06	0.13	9.81	10.30	35%
10,000	10.12	0.09	9.95	10.29	35%
25,000	10.11	0.06	10.00	10.22	35%

- Barrier option valuation with moment matching:

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper	S.E. Reduction
100	1.40	0.32	0.78	2.02	29%
500	1.50	0.15	1.20	1.80	15%
1,000	1.55	0.11	1.33	1.77	8%
5,000	1.44	0.05	1.35	1.54	9%
10,000	1.41	0.04	1.35	1.48	9%
25,000	1.41	0.02	1.37	1.46	7%

# Variance Reduction Technique #3: Importance Sampling



- Heuristic premise:
  - Depending upon the problem, we may be running many simulation paths that do not add value to our understanding of the issue
  - For example, if we are trying to value a deep-out-of-the-money option, most of the paths we sample will result in the option payoff being zero
    - This slows convergence and increases the error of our estimate
  - Why not sample only paths in the “important” region, and multiply the ending answer by the probability of being in this region?
- Procedure:
  - Determine “important” region
  - Adjust samples so they only land in this region
  - Multiply estimated value by the probability of being in the “important” region
- It may be very difficult to determine what the “important” region is, especially for path-dependent options

# Variance Reduction Technique #3: Importance Sampling



- Calculate importance region for sampling →
- Adjust samples so they come from that region →
- Multiply calculated option value by the probability of the “important” region →

```

If Flag = "p" Then
    Dim umax As Double
    umax = Application.WorksheetFunction.NormSDist((Log(K / S0) - rndrft) / rnshck)
Else
    Dim umin As Double
    umin = Application.WorksheetFunction.NormSDist((Log(K / S0) - rndrft) / rnshck)
End If
    
```

```

If Flag = "p" Then
    For j = 1 To Steps
        N01 = MoroNormSInv(mt.real3() * (umax - 0))
        S(j) = S(j - 1) * Exp(rndrft + rnshck * N01)
    Next j
Else
    For j = 1 To Steps
        N01 = MoroNormSInv(umin + mt.real3() * (1 - umin))
        S(j) = S(j - 1) * Exp(rndrft + rnshck * N01)
    Next j
End If
    
```

```

If Flag = "p" Then
    mu = mu * (umax - 0)
Else
    mu = mu * (1 - umin)
End If
    
```

- Out-of-the-money put option (same as before except  $K = 60$ ), BS value \$0.32:

Brute Force Monte Carlo

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	0.59	0.34	(0.09)	1.26
500	0.40	0.10	0.21	0.60
1,000	0.38	0.06	0.26	0.51
5,000	0.30	0.02	0.26	0.35
10,000	0.32	0.02	0.28	0.36
25,000	0.33	0.01	0.31	0.36

Importance Sampling

Simulation Paths	Value Estimate
100	0.32
500	0.33
1,000	0.33
5,000	0.32
10,000	0.31
25,000	0.31

Has converged after only 100 paths!

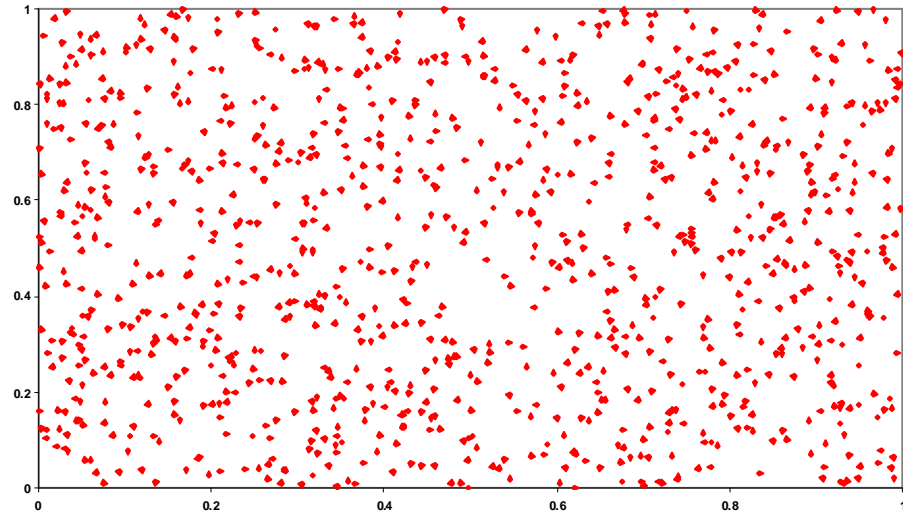
# Variance Reduction Technique #4: Low-Discrepancy Sequences



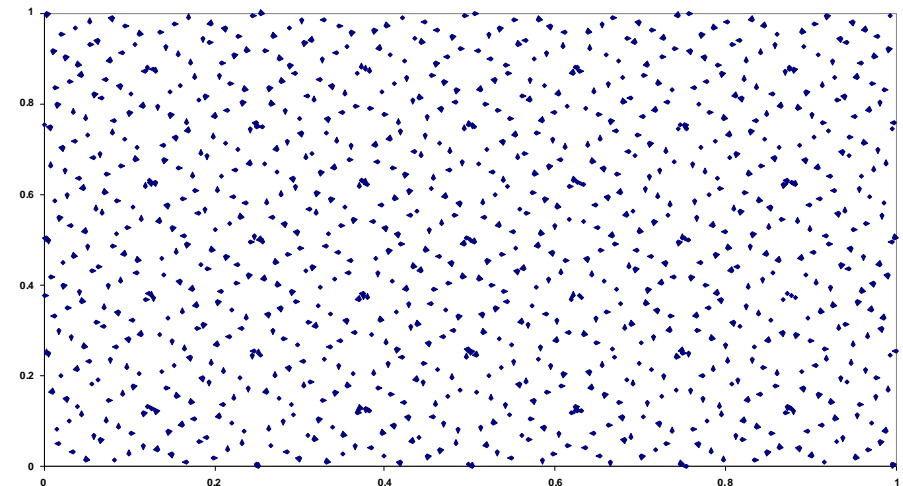
- Heuristic premise:
  - If we can avoid sampling normal variables that are “too clustered” or “too far apart,” we will get a more even spread over the sample space
  - This should give better option values
- Procedure:
  - Produce a sequence of numbers while trying to keep “discrepancy” low
  - “Discrepancy” is a statistical measure of the disparity between the actual number of samples falling into any region and the number that would be expected to fall in that region based upon its size
  - Low-discrepancy (or “quasi-random”) sequences (Halton, Faure, Hammersley, Sobol, Niederreiter, etc) are specially designed to minimize the statistical discrepancy of a sequence of numbers
- When implemented appropriately, low-discrepancy sequences have the property that the standard error of a Monte Carlo estimate will scale with the number of paths, rather than with the square root of the number of paths
- Care must be exercised in the dimensionality of the problem being addressed

# Variance Reduction Technique #4: Low-Discrepancy Sequences

- Two-dimensional sample from typical PRNG →



- Two-dimensional sample from Sobol low-discrepancy sequence →



# Variance Reduction Technique #4: Low-Discrepancy Sequences



- Below illustrates the implementation of a Sobol sequence into the valuation of the call-on-the-max option we illustrated earlier:

```
Dim gsl
Set gsl = CreateObject("GSL.Functions")
Dim sa As Variant, tmpPaths As Variant, tmpStocks As Variant
tmpPaths = Paths
tmpStocks = 2

sa = gsl.getSobolMD(tmpPaths, tmpStocks)

ReDim S1(0 To Steps), S2(0 To Steps)
S1(0) = S01
S2(0) = S02
For j = 1 To Steps
    ReDim N01(1 To 2)
    N01(1) = MoroNormSInv(CDbl(Max(Min(sa(i - 1), 0), 0.999999), 0.000001)))
    N01(2) = N01(1) * rho + sqrt(1 - rho^2) * MoroNormSInv(CDbl(Max(Min(sa(i - 1), 1), 0.999999), 0.000001)))
    S1(j) = S1(j - 1) * Exp(rndrft1 + rnshck1 * N01(1))
    S2(j) = S2(j - 1) * Exp(rndrft2 + rnshck2 * N01(2))
Next j
```

- Call-on-the-max option, analytic value \$15.35:

Brute Force Monte Carlo

Simulation Paths	Value Estimate	Standard Error	90% CI Lower	90% CI Upper
100	15.83	2.59	10.75	20.91
500	16.51	1.24	14.09	18.94
1,000	16.07	0.87	14.37	17.77
5,000	15.97	0.38	15.22	16.71
10,000	15.70	0.26	15.18	16.21
25,000	15.45	0.16	15.13	15.77
50,000	15.31	0.11	15.09	15.54
100,000	15.40	0.08	15.24	15.56

Sobol Sequences

Simulation Paths	Value Estimate
100	14.37
500	15.12
1,000	15.18
5,000	15.32
10,000	15.30
25,000	15.34
50,000	15.34
100,000	15.34

A reasonable answer after just 5,000 paths!

# Variance Reduction Technique #5: Rotating Seed



- Heuristic premise:
  - Suppose I need to value a large number of “similar” options
  - Is there a way to achieve a very good estimate of the aggregate value without achieving a good estimate of the value of any individual option?
    - Can I get the errors to “cancel out” due to the Law of Large Numbers?
- Procedure:
  - Sample paths for the first option
  - Re-seed the PRNG and sample paths for the second option
  - Repeat for each option
- Results in a “tug of war” between the reduction in the number of paths per option that is run (a run-time saver) and a requirement that the generator produce many more sets of paths (a run-time increaser)
- May not yield acceptable results for the individual options

# Variance Reduction Technique #5: Rotating Seed



- Consider a group of 5,000 put options, with maturities ranging from 5 to 15 years, and strike prices ranging from \$50 to \$150, as follows:

Inputs	
Stock	100
Rates	4.00%
Volatilities	30.00%

Put Option	Expiration Time (m)	Strike Price	Analytic Value
1	68	56.16	2.99
2	152	107.73	18.39
3	143	50.00	3.42
.....	.....	.....	.....
.....	.....	.....	.....
.....	.....	.....	.....
4,998	132	148.93	35.33
4,999	84	130.46	29.94
5,000	78	83.93	10.50

- The aggregate Black-Scholes value of the portfolio is \$86,520
- Two Monte Carlo valuations were compared:
  - Brute force Monte Carlo using a seed of 41
  - Rotating seed, in which the first 5,000 prime numbers were chosen to seed the generation of each option's unique set of sample paths

# Variance Reduction Technique #5: Rotating Seed



- Results of the valuation exercise:

METHOD A : BRUTE FORCE

PATHS	64	128	256	512	1,024	2,048	4,096	8,192	16,384	32,768	65,536	131,072
MC Estimate	83,873	86,913	92,925	92,896	87,407	87,709	86,068	86,036	86,599	86,624	87,148	86,842
Standard Error	10,910	7,751	5,478	3,917	2,708	1,908	1,336	940	663	471	333	235
Run Time (m)	0	0	0	1	1	2	4	8	16	32	64	128

For rotating seed, run time is much longer for a fixed number of paths, since 5,000 times as many random paths must be generated ...

... however, this is offset by having to run fewer paths to achieve the same standard error ...

... the winner of this tug-of-war depends upon your specific application.

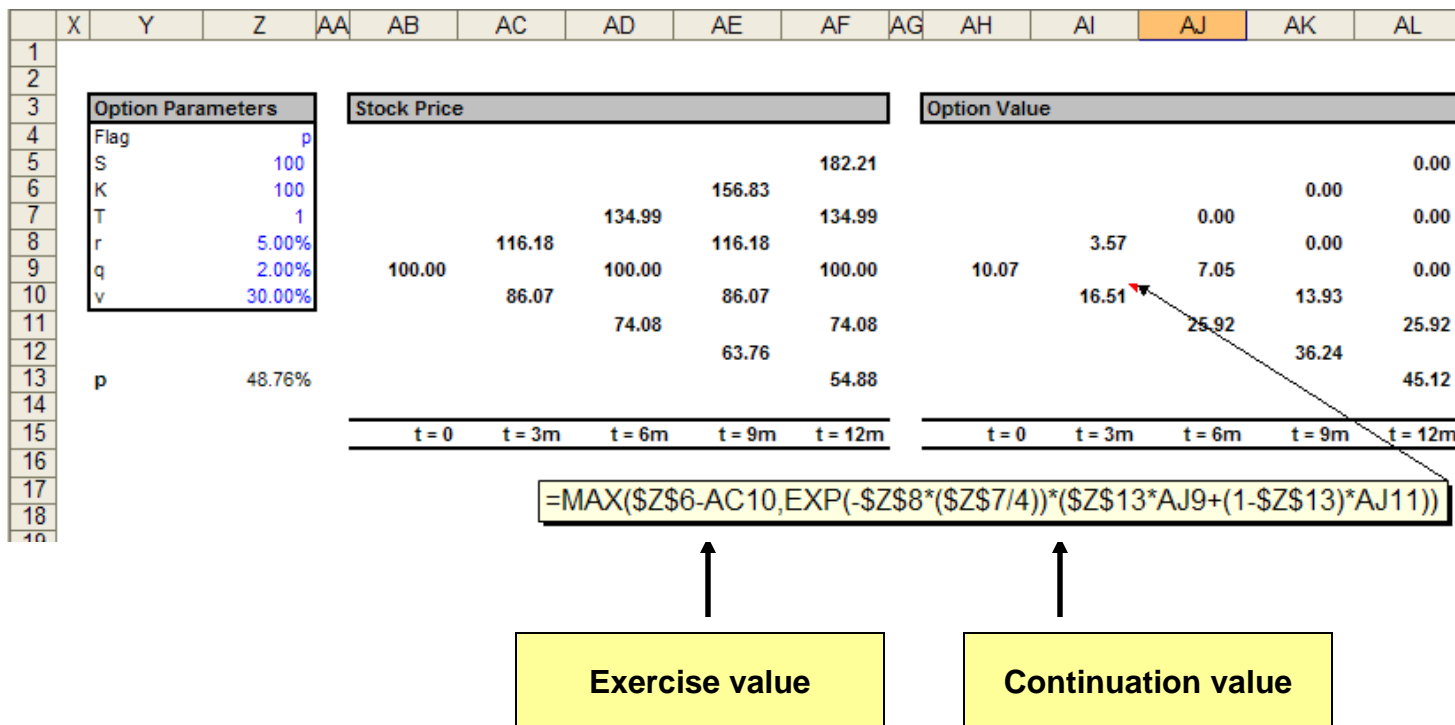
METHOD B : ROTATING SEED

PATHS	2	4	8	16	32	64	128	256	512
MC Estimate	85,470	86,115	86,835	86,459	86,552	86,551	86,736	86,669	86,647
Standard Error	429	519	458	364	270	198	142	93	64
Run Time (m)	0	0	1	1	2	5	9	18	37

# Monte Carlo Simulation and American Options: Overview



- Since they generally rely upon backward induction, it is quite easy to accommodate early-exercise features in PDE- and tree-based option pricing methods



- For Monte Carlo methods, this is much more difficult since they generally proceed forward in time, not backward, meaning that the continuation value of the option is not readily available

# Monte Carlo Simulation and American Options: The Early Exercise Boundary



- Pure brute force (i.e. stopping at every time step and performing an entirely separate (nested) Monte Carlo simulation to determine the continuation value) is simply not practical due to an exponential increase in computation time
- Most research in this area relies on the concept of parameterizing an “early exercise boundary”
  - Critical values for the asset price above or below which early exercise is determined to be optimal
- The Monte Carlo valuation then proceeds in the usual manner, but implements an early exercise at any point along a sample path that satisfies the criterion derived from the early exercise boundary

# Monte Carlo Simulation and American Options: The Early Exercise Boundary



- It might be easiest to walk through an example of one such method
- Suppose we run eight sample paths for the valuation of a 3-year, at-the-money American put option
- The “boundary” at any intermediate point is determined by testing candidate points and choosing the one that maximizes the average value of the option at that point

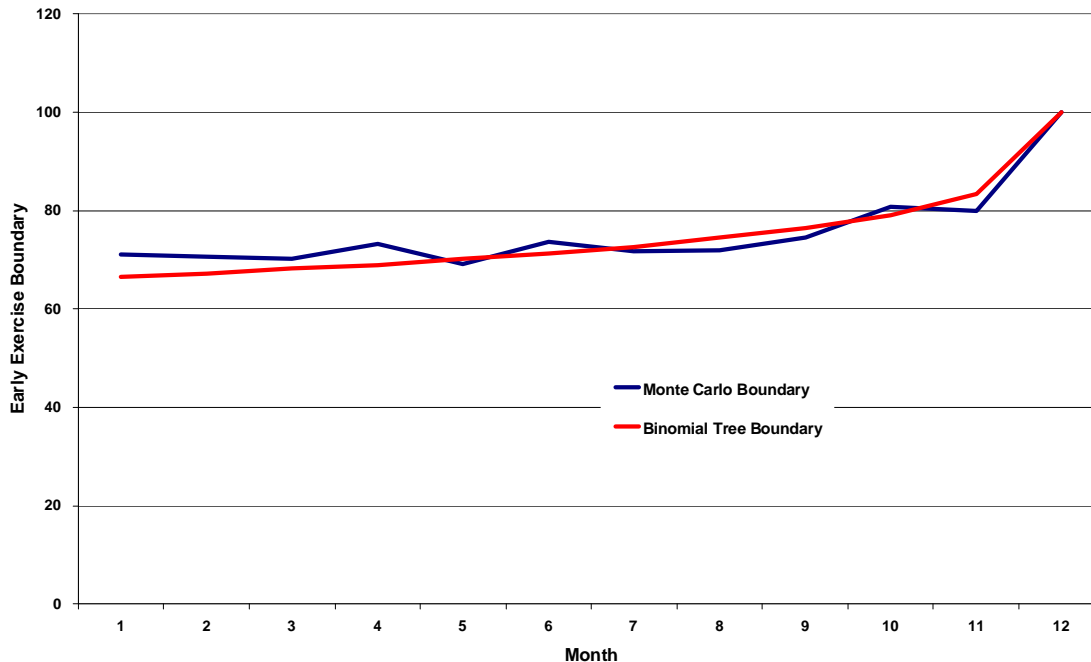
	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY
1										
2	<b>Stock Price Simulations</b>									
3	<b>Path</b>	<b>T = 1</b>	<b>T = 2</b>	<b>T = 3</b>						
4	1	99.41	121.06	84.95						
5	2	106.14	149.03	181.28						
6	3	108.39	92.30	74.59						
7	4	82.40	55.96	32.94						
8	5	105.05	90.63	73.65						
9	6	126.04	143.67	130.21						
10	7	129.43	110.90	171.19						
11	8	101.85	73.17	76.87						
12	<b>Terminal Payoff</b>									
13					<b>T = 3</b>					
14					15.05					
15					0.00					
16					25.41					
17					67.06					
18					26.35					
19					0.00					
20					0.00					
21					23.13					
22	<b>Test Point</b>									
23	<b>Path</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>	<b>T = 2</b>
24	1	0.00	0.00	14.46	14.46	14.46	0.00	14.46	14.46	14.46
25	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
26	3	7.70	7.70	7.70	24.41	24.41	7.70	7.70	7.70	24.41
27	4	44.04	44.04	44.04	44.04	44.04	44.04	44.04	44.04	44.04
28	5	9.37	9.37	9.37	25.32	9.37	9.37	9.37	9.37	25.32
29	6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
31	8	26.83	26.83	26.83	22.22	26.83	26.83	26.83	26.83	26.83
32	<b>Average</b>	<b>10.99</b>	<b>10.99</b>	<b>12.80</b>	<b>16.31</b>	<b>14.89</b>	<b>10.99</b>	<b>12.80</b>	<b>12.80</b>	<b>16.88</b>
33	<b>Early Exercise Boundary @ T = 2</b>									
34										<b>73.17</b>
35	<b>=OFFSET(\$AQ\$15,0,MATCH(MAX(AR25:AY25),AR25:AY25,0))</b>									

- The T = 2 exercise boundary is \$73.17
- We then go back to T = 1 and perform the same procedure, assuming the T = 2 boundary is “locked in”
- Once the boundary is determined, we re-seed the PRNG and run another Monte Carlo simulation, assuming exercise takes place according to the boundary developed
- In practice, many more paths are required to get a suitable boundary

# Monte Carlo Simulation and American Options: The Early Exercise Boundary



- Assume the same vanilla put option as before (1y ATM put), but American
- Using a binomial tree with 12,000 time steps, we get a value of \$10.47
- We then run the early exercise boundary method using monthly time steps, 50,000 paths to compute the boundary, and 50,000 paths to value the option



- The Monte Carlo method produces a value of \$10.74
- This is counterintuitive, as the Monte-Carlo-based early exercise boundary is suboptimal (which should lead to a lower value, all else equal)

# Monte Carlo Simulation and American Options: Control Variates



- Comparing the Monte Carlo estimate of \$10.74 to the binomial estimate of \$10.47 is a flawed comparison, since it is actually a joint test of two features:
  - How well the early exercise boundary is parameterized
  - How well the second Monte Carlo simulation (conditional on the boundary) prices the underlying European option
- In order to render the comparison “fair,” we need to control for the second effect
- We can accomplish this using the control variate technique, as follows:
  - American option value =  $A - (B - C)$ , where:
    - A is the estimated Monte Carlo American option value using the simulated exercise boundary
    - B is the estimated value of the otherwise identical European option using the same sample paths that were used to compute A
    - C is the analytic value of the otherwise identical European option
- This removes some of the simulation bias and provides a better basis for comparison
- In our example this yields  $\$10.74 - (\$10.46 - \$10.12) = \$10.40$ 
  - Closer, and lower, which is a more intuitive outcome

# Monte Carlo Simulation and American Options: Why Would Anyone Do This?

---



- It is unlikely that one would choose the exercise boundary parameterization method via Monte Carlo if the goal were to price a single-underlying American option
- However, there are situations in which this type of method makes sense:
  - If the number of underlyings make a PDE solution unwieldy
  - If the option is simultaneously path-dependent and American-style

## Monte Carlo Methods – Conclusion

---

- Monte Carlo methods rely upon repeated random or pseudorandom sampling
- They are applied in a variety of fields, including finance
  - First applied to option pricing by an actuary
- They have many advantages, including ease of implementation, ability to accommodate path-dependency, and a speed advantage in the presence of many underlyings
- They are slower than competing methods for low-dimensional problems, and have difficulty with American-style payoffs
- Researchers have developed a number of ways to reduce the variance of simulation estimates
- They are likely to gain even more favor in quantitative finance as hardware costs decline and the method is popularized

Thanks for your attention!

# References

---

- Monte Carlo Methods in Financial Engineering, Springer-Verlag
- Options, Futures, and Other Derivatives, 5<sup>th</sup> Edition, Prentice Hall
- Wikipedia