JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Harnessing Excel's VBA for Simulating Fundamental Stochastic Processes: A Deep Dive into GBM and CIR Implementations

[1]**Debasree Goswami**
[1]Associate Professor, Department of Statistics, Hindu College, University of Delhi, Delhi, India-7

Abstract : This research delves into the application of stochastic processes, specifically the Geometric Brownian Motion (GBM) and the Cox-Ingersoll-Ross (CIR) process, in the realm of quantitative finance. Recognizing the ubiquity of Microsoft Excel in the finance sector, we introduce a hands-on approach to simulate these processes using Visual Basic for Applications (VBA). Our paper highlights the implementation details, ensuring the adaptability of the developed code for various financial stochastic processes. By bridging theoretical finance with practical Excel-based solutions, this research aims to provide a valuable tool for financial analysts and researchers.

Keywords: *Stochastic Processes, Geometric Brownian Motion (GBM), Cox-Ingersoll-Ross (CIR) Process, Visual Basic for Applications (VBA), Financial Modeling in Excel, Numerical Simulation, Milstein Scheme.*

1. INTRODUCTION

Financial markets are inherently uncertain, with asset prices and interest rates exhibiting random fluctuations over time. To capture and understand these dynamics, stochastic processes, which describe systems evolving over time with a probabilistic element, have become fundamental in the realm of quantitative finance. Among these, the Geometric Brownian Motion (GBM) and the Cox-Ingersoll-Ross (CIR) process stand out due to their widespread applications in modeling stock prices and interest rates, respectively.

While there are numerous software platforms tailored for financial modeling, Microsoft Excel remains a staple in the finance industry, offering accessibility and ease of use. Its embedded programming environment, Visual Basic for Applications (VBA), allows for the creation of custom functions and simulations, making it a versatile tool for financial analysts and researchers. This paper delves into the implementation of the GBM and CIR processes in Excel using VBA, providing a hands-on approach to understanding and simulating these fundamental stochastic processes. Furthermore, we discuss the adaptability of the developed code to model other related processes, exemplified by the ease of transitioning from the CIR to the Ornstein–Uhlenbeck (OU) process.

By bridging the gap between theoretical finance and practical implementation, this paper aims to serve as a valuable resource for both academia and industry professionals looking to harness the power of Excel for stochastic financial modeling.

2. METHODOLOGY

2.1. Stochastic Differential Equation and Milstein Scheme

A general one-dimensional SDE can be written as:

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t \qquad (1)$$

where dX_t is the change in the process X_t, $a(X_t, t)$ is the drift term, $b(X_t, t)$ is the diffusion term, and dW_t is a Wiener process or Brownian motion [1].
The Milstein scheme [1], [2] approximates the solution of the SDE by discretizing time into small intervals of length Δt, and the approximation at time t_{n+1} is given by:

$$X_{t_{n+1}} = X_{t_n} + a\left(X_{t_n}, t_n\right)\Delta t + b\left(X_{t_n}, t_n\right)\Delta W_n + \frac{1}{2}b\left(X_{t_n}, t_n\right)b'\left(X_{t_n}, t_n\right)\left((\Delta W_n)^2 - \Delta t\right) \quad (2)$$

Here $\Delta W_n = W_{t_{n+1}} - W_{t_n}$ is the increment of the Wiener process, which is normally distributed with mean 0 and variance Δt. Further, $b'\left(X_{t_n}, t_n\right)$ is the derivative of the diffusion term with respect to X.

2.2. Geometric Brownian Motion (GBM) and the Milstein Scheme

Geometric Brownian Motion (GBM) is a continuous-time stochastic process [3] widely used in mathematical finance to model stock prices, interest rates, exchange rates, and other financial variables. The process assumes that the percentage changes in the stock price are normally distributed, resulting in a log-normal distribution of the stock price over time. Mathematically, it has the form

$$dS_t = \mu S_t dt + \sigma S_t dW_t. \tag{3}$$

where S_t is the value of the stock price at time t, μ is the drift rate, often interpreted as the expected return of the stock, σ is the volatility coefficient, and W_t is a standard Brownian motion or Wiener process.
The Milstein approximation for GBM is:

$$\Delta S_t = \mu\, S_t\, \Delta t + \sigma\, S_t\, \Delta W_t + \frac{1}{2}\sigma^2\, S_t\, ((\Delta W_t)^2 - \Delta t) \tag{4}$$

Here, ΔS_t is the change in stock price, Δt is the time step, and ΔW_t is the increment of the Wiener process, which is normally distributed with mean 0 and variance Δt.

The additional term $\frac{1}{2}\sigma^2\, S_t((\Delta W_t)^2 - \Delta t)$ in the Milstein scheme is a correction term which provides an improvement over the Euler-Maruyama method in approximating GBM.

2.3. Cox-Ingersoll-Ross (CIR) Process and the Milstein Scheme

The Cox-Ingersoll-Ross (CIR) process [3] is a type of mean-reverting stochastic process primarily used in finance to model interest rates. The CIR process ensures that interest rates remain positive, making it a more realistic model for certain financial applications compared to other interest rate models.

The stochastic differential equation (SDE) representing the CIR process is given by:

$$dX_t = \alpha(\theta - X_t)\, dt + \sigma\sqrt{X_t}\, dW_t. \tag{5}$$

where X_t is the process value at time t, α is the speed of mean reversion, θ is the long-term mean (interest rate), σ is the volatility and dW_t is the increment of a Wiener process.
The Milstein scheme is an extension of the Euler-Maruyama scheme that includes a correction term to account for the second moment of the Wiener process increment. For the CIR process, the Milstein scheme can be written as:

$$\Delta X_t = \alpha(\theta - X_t)\, \Delta t + \sigma\sqrt{X_t}\, \Delta W_t + \frac{1}{2}\sigma^2\, ((\Delta W_t)^2 - \Delta t) \tag{6}$$

where $\Delta X_t = X_t - X_{t-1}$ is the change in the process value, Δt is the time increment and ΔW_t is a Wiener process increment over the interval Δt , which is normally distributed with mean 0 and variance Δt.
The additional term $\frac{1}{2}\sigma^2((\Delta W_t)^2 - \Delta t)$ in the Milstein scheme corrects for the approximation error in the Euler-Maruyama method, making the Milstein scheme a more accurate method for approximating the CIR process.

2.4. Implementation in Excel using VBA

Visual Basic for Applications (VBA) is an event-driven programming language developed by Microsoft. It is primarily used for automating tasks in Microsoft Office applications, with Excel being one of its most popular platforms. VBA allows users to create custom functions, automate repetitive tasks, and develop user-defined models and simulations, making it a powerful tool suitable for financial modeling and analysis.
Further, opting for ($Application.WorksheetFunction.RandBetween(0, 2^{32})\ /\ (2^{32})$) over VBA's native $Rnd()$ function provided a more granular distribution of random numbers in the interval [0,1], due to its ability to generate a larger set of distinct values, enhancing the resolution of the random number generation process.

Description of the functions:
$MilsteinGBMandEXACT$: This function approximates the Geometric Brownian Motion using the Milstein scheme and provides the exact solution values. Depending on the user's selection, it can return either the approximate values (upon selecting one column) or the exact values and approximate values (upon selecting two columns). The function requires the initial stock price (So), drift coefficient (Mu), volatility coefficient (Sigma), and a range of time values (tRange) as its arguments. (Appendix II)

$MilsteinCIR$: Designed to approximate the Cox-Ingersoll-Ross (CIR) process using the Milstein scheme, this function takes initial parameters such as the initial value (Xo), mean reversion rate (Alpha), long-term mean (Mu), and volatility (Sigma), along with a range of time values (tRange). It then produces an array of approximated values over the specified time frame. (Appendix III)

$ValidateParameters$: Serving as a utility function, "ValidateParameters" ensures that the input parameters for the above functions are valid. It checks for numeric values, the structure of the time range, and the order of time values, raising errors if any inconsistencies are found. (Appendix I)

3. RESULTS AND DISCUSSION

Using the developed VBA functions, we generated sample paths for both the Geometric Brownian Motion and the Cox-Ingersoll-Ross processes. For a given set of parameters, the functions returned arrays of values representing the evolution of the processes over time. The results were plotted directly within Excel, demonstrating the ease of integration and visualization when using VBA in conjunction with Excel's native charting capabilities. (Appendix IV and V)

Upon comparing the results obtained from our VBA functions with those from other platforms, such as R and Python, we observed a high degree of similarity. This attests to the accuracy of the Milstein scheme's implementation in our VBA functions. While minor discrepancies can arise due to differences in random number generation or numerical methods across platforms, the overall trajectories of the generated paths were consistent.

The Milstein scheme, as implemented in our VBA functions, offers a balance between accuracy and computational efficiency. While there are more sophisticated numerical methods available, [2] the Milstein scheme provides a good trade-off, especially for applications within Excel where rapid computation is often desired.

Furthermore, the flexibility of VBA allows for easy modifications and extensions of these functions, catering to more specific needs or advanced financial models. The integration with Excel also means that users can leverage Excel's data analysis and visualization tools, making the entire modeling and analysis process seamless and efficient.

Further, the Cox-Ingersoll-Ross (CIR) process and the Ornstein–Uhlenbeck (OU) process share structural similarities, with the primary difference being in the volatility term. By adjusting the line 54 in the VBA function "$MilsteinCIR$" (Appendix-III) for the CIR process, one can easily adapt it to model the OU process, demonstrating the flexibility and adaptability of the developed code for various stochastic processes in finance.

4. REFERENCES

[1] T. Mikosch, "Elementary Stochastic Calculus, with Finance in View," vol. 6, Oct. 1998, doi: 10.1142/3856.
[2] P. E. Kloeden and E. Platen, "Numerical Solution of Stochastic Differential Equations," *Numerical Solution of Stochastic Differential Equations*, 1992, doi: 10.1007/978-3-662-12616-5.
[3] J. Franke, W. K. Härdle, and C. M. Hafner, "Statistics of Financial Markets," 2019, doi: 10.1007/978-3-030-13751-9.

ACKNOWLEDGMENT

The author would like to extend heartfelt gratitude to Ms. Jasvinder Goswami, Associate Professor in the Department of Statistics at PGDAV(M) College, University of Delhi, and his wife, for her invaluable discussions and insightful comments on the manuscript.

APPENDIX - I

```
' =================== Helper Functions ===================
Function ValidateParameters(inputs As Variant, tRange As Range) As Boolean
    ' Check if all inputs are numeric
    Dim i As Integer
    For i = LBound(inputs) To UBound(inputs)
        If Not IsNumeric(inputs(i)) Then
            Err.Raise 9999, "ValidateParameters", "All input parameters must be numeric values."
            Exit Function
        End If
    Next i

    ' Check if tRange is a single column
    If tRange.Columns.Count > 1 Then
        Err.Raise 9999, "ValidateParameters", "tRange must be a single column of values."
        Exit Function
    End If

    ' Convert the range to an array
    Dim t() As Variant
    t = tRange.Value

    ' Check if tRange contains only numeric values
    For i = LBound(t, 1) To UBound(t, 1)
        If Not IsNumeric(t(i, 1)) Then
            Err.Raise 9999, "ValidateParameters", "tRange must contain only numeric values."
            Exit Function
        End If
    Next i

    ' Check if tRange values are in non-decreasing order
    For i = LBound(t, 1) + 1 To UBound(t, 1)
        If t(i, 1) <= t(i - 1, 1) Then
            Err.Raise 9999, "ValidateParameters", "Values in tRange must be in >= order."
            Exit Function
        End If
    Next i

    ValidateParameters = True
End Function
```

APPENDIX - II

```
' ==================== Main Functions ====================
' Function: ========== MilsteinGBMandEXACT ===============
' Purpose: Approximate the GBM using the Milstein scheme and provide the exact solution values.
'         The function's behavior changes based on the number of selected columns:
'         - 2 columns: Returns exact values in the first column and approximate values in the second.
'         - 1 column: Returns only the approximate values.
' Parameters:
'   - So: Initial stock price
'   - Mu: Drift coefficient
'   - Sigma: Volatility coefficient
'   - tRange: Range of time values
' Returns: An array of exact and/or approximate stock prices based on the number of selected columns.

Function MilsteinGBMandEXACT(So As Variant, Mu As Variant, Sigma As Variant, tRange As Range) As Variant()

    On Error GoTo ErrorHandler

    Dim DeltaT As Double
    Dim CumDeltaW As Double
    Dim DeltaW As Double
    Dim PrevSt As Double
    Dim ForNextSt As Double
    Dim NextSt As Double
    Dim ExactSt As Double
    Dim St() As Variant
    Dim t() As Variant
    Dim i As Integer
    Dim n As Integer
    Dim no_of_column_selected As Integer

    ' Convert the range to an array
    t = tRange.Value

    ' Determine the number of time steps based on the length of the input time array
    n = UBound(t, 1) - LBound(t, 1) + 1

    ' Validate the parameters using the helper function
    If Not ValidateParameters(Array(So, Mu, Sigma), tRange) Then Exit Function

    ' Determine the number of columns selected by the user
    no_of_column_selected = Application.Caller.Columns.Count

    ' Check the number of columns selected by the user
    If no_of_column_selected = 2 Then
        ' User selected two columns
        ReDim St(1 To n, 1 To 2) As Variant
    Else
        ' User selected one column
        ReDim St(1 To n, 1 To 1) As Variant
    End If

    ' Initialize the first value
    St(1, 1) = So
    If no_of_column_selected = 2 Then
        St(1, 2) = So
    End If

' Initialize for the first value Wiener Process
    CumDeltaW = 0

    ' Loop through each time step
    For i = 2 To n
        ' Compute the time increment
        DeltaT = t(i, 1) - t(i - 1, 1)

        ' Generate a random number for the Wiener process increment
        DeltaW = (Sqr(DeltaT) * Application.NormSInv(Application.WorksheetFunction.RandBetween(0, 2 ^ 32) / (2 ^ 32)))

        CumDeltaW = CumDeltaW + DeltaW

        ' Exact solution for GBM
        ExactSt = So * Exp((Mu - 0.5 * Sigma ^ 2) * t(i, 1) + Sigma * CumDeltaW)

        ' Milstein scheme for GBM approximation
        PrevSt = St(i - 1, 1)
        ForNextSt = Mu * PrevSt * DeltaT + Sigma * PrevSt * DeltaW + 0.5 * (Sigma ^ 2) * PrevSt * ((DeltaW ^ 2) - DeltaT)
        NextSt = PrevSt + ForNextSt

        ' Assign values to the output array based on the number of selected columns
        If no_of_column_selected = 2 Then
            St(i, 1) = ExactSt
            St(i, 2) = NextSt
        Else
```

```
            St(i, 1) = NextSt
        End If
    Next i

    ' Return the array of values
    MilsteinGBMandEXACT = St

    Exit Function

ErrorHandler:
    MsgBox "An error occurred in MilsteinGBMandEXACT: " & Err.Description, vbExclamation, "Error"
End Function
```

APPENDIX - III

```
' ==================== Main Functions ====================
' Function: ========== MilsteinCIR =======================
' Purpose: Approximate the Cox-Ingersoll-Ross (CIR) process using the Milstein scheme.
' Parameters:
'   - Xo: Initial value
'   - Alpha: Speed of mean reversion
'   - Mu: Long-term mean
'   - Sigma: Volatility coefficient
'   - tRange: Range of time values
' Returns: An array of values representing the CIR process.

Function MilsteinCIR(Xo As Variant, Alpha As Variant, Mu As Variant, Sigma As Variant, tRange As Range) As Variant()

    On Error GoTo ErrorHandler

    Dim n As Integer
    Dim DeltaW As Double
    Dim DeltaT As Double
    Dim PrevSt As Double
    Dim NextXt0 As Double
    Dim NextXt1 As Double
    Dim NextXt2 As Double
    Dim NextXt3 As Double
    Dim NextXt As Double
    Dim Xt() As Variant
    Dim t() As Variant
    Dim i As Integer

    ' Convert the range to an array
    t = tRange.Value

    ' Determine the number of time steps based on the length of the input time array
    n = UBound(t, 1) - LBound(t, 1) + 1

    ' Validate the parameters using the helper function
    If Not ValidateParameters(Array(Xo, Alpha, Mu, Sigma), tRange) Then Exit Function

    ' Initialize the St array to store the stock prices
    ReDim Xt(1 To n, 1 To 1) As Variant
    Xt(1, 1) = Xo

    ' Loop through each time step to compute the stock price using the Milstein scheme
    For i = 2 To n
        ' Compute the time increment
        DeltaT = t(i, 1) - t(i - 1, 1)

        ' Generate a random number from a standard normal distribution for the Wiener process increment
        DeltaW = (Sqr(DeltaT) * Application.WorksheetFunction.NormSInv(Application.WorksheetFunction.RandBetween(0, 2 ^ 32) / (2 ^ 32)))

        ' Apply the Milstein scheme formula
        PrevXt = Xt(i - 1, 1)
        NextXt = PrevXt + (Alpha * (Mu - PrevXt) * DeltaT) + (Sigma * Sqr(PrevXt) * DeltaW) + (0.25 * (Sigma ^ 2) * DeltaT * ((DeltaW ^ 2) - DeltaT))
        Xt(i, 1) = NextXt
    Next i

    ' Return the array of stock prices
    MilsteinCIR = Xt

    Exit Function

ErrorHandler:
    MsgBox "An error occurred in MilsteinCIR: " & Err.Description, vbExclamation, "Error"
End Function
```

APPENDIX - IV

Table - 1			GBM	
		Time (t)	Exact Solution	Approximate Solution
n =	256	0	1.000	1.000
DeltaT: $\Delta t =$	0.00391	0.0039	0.836	0.836
		0.0078	0.924	0.925
mu: $\mu =$	0.01	0.0117	0.904	0.904
Sg: $\sigma =$	2	0.0156	0.734	0.734
S_o =	1	0.0195	0.733	0.733
		0.0234	0.765	0.765
		0.0273	0.690	0.690
		0.0313	0.644	0.643
…	…	…	…	…
		0.9727	1.273	1.272
		0.9766	1.329	1.329
		0.9805	1.335	1.336
		0.9844	1.378	1.378
		0.9883	1.700	1.700
		0.9922	1.841	1.842
		0.9961	1.684	1.683
		1.0000	1.554	1.554

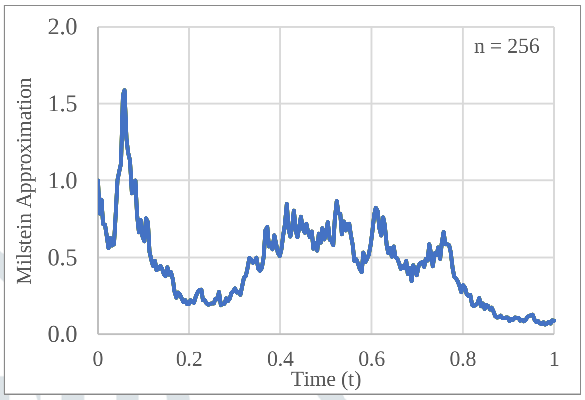

Fig 1. Equidistant Milstein scheme approximating the Geometric Brownian Motion process for parameter choices in the Table 1.

APPENDIX - V

Table - 2			CIR
		Time (t)	Approximate Solution
n =	128	0	0.5
DeltaT: $\Delta t =$	0.00781	0.0078	0.5191
		0.0156	0.5110
mu: $\mu =$	1	0.0234	0.6123
Sg: $\sigma =$	0.8	0.0313	0.5065
X_o =	0.5	0.0391	0.5785
Alpha: $\alpha =$	5	0.0469	0.4677
		0.0547	0.4184
		0.0625	0.3857
…	…	…	…
		0.9453	1.3250
		0.9531	1.3522
		0.9609	1.3025
		0.9688	1.2873
		0.9766	1.3052
		0.9844	1.3930
		0.9922	1.3613
		1.0000	1.3001

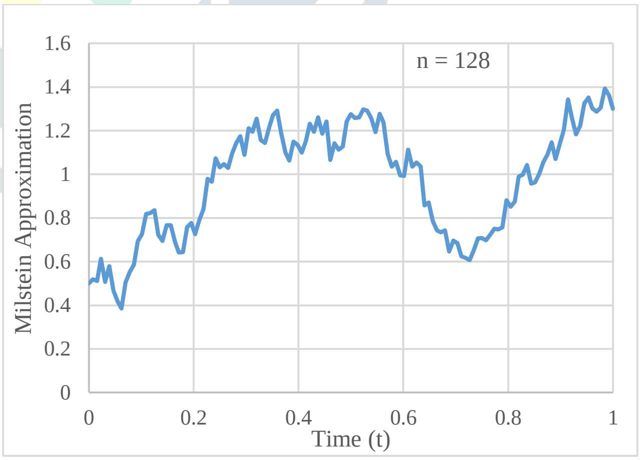

Fig 2. Equidistant Milstein scheme to approximate the Cox-Ingersoll-Ross (CIR) process for parameter choices in the Table 2.