Thesis 8

Anna Lucia Lamacchia
Statistics

January 15, 2024

Stochastic Processes and SDE's

1 Introduction

1.1 Brief Historical Context of Stochastic Differential Equations

The roots of Stochastic Differential Equations (SDEs) can be traced back to the early 20th century when the mathematical community began struggling with the challenges of incorporating randomness into differential equations. The work of Norbert Wiener in the 1920s put the foundation for understanding the behavior of Brownian motion, a continuous-time stochastic process. It was later in the 1950s and 1960s that the rigorous mathematical framework for SDEs, particularly Ito's calculus, was developed, marking a significant milestone in the field of stochastic analysis.

1.2 The Significance of SDEs in Capturing Randomness in Dynamic Systems

Dynamic systems in various fields often exhibit inherent randomness or uncertainty due to external influences or inherent variability. Classical deterministic models, described by ordinary differential equations (ODEs), fail in capturing these stochastic elements. SDEs provide a powerful mathematical tool to model systems where randomness is a crucial aspect of their behavior. The significance of SDEs lies in their ability to incorporate stochastic processes, such as Brownian motion, into mathematical models. This enables a more realistic representation of systems evolving over time under the influence of random factors. In finance, for example, SDEs are instrumental in modeling asset prices, capturing the inherent volatility observed in financial markets. In physics, SDEs describe the erratic motion of particles in a fluid. In biology, SDEs are employed to model population dynamics with random variations in birth and death rates. So, we can say that the adoption of SDEs has become very present in diverse scientific and engineering disciplines, reflecting their versatility in modeling real-world phenomena.

2 Basics of Stochastic Processes

2.1 Introduction to Stochastic Processes and their Significance in Modeling Random Phenomena

Stochastic processes form the backbone of modeling systems subjected to randomness. Unlike deterministic processes, stochastic processes incorporate the element of chance, making them indispensable for describing real-world phenomena with inherent uncertainty. A stochastic process is a collection of random variables indexed by time or another parameter, representing the evolution of a system over time.

The significance of stochastic processes lies in their ability to capture the inherent variability observed in dynamic systems. In many natural and engineered systems, random influences or external factors contribute to the unpredictable behavior of the system. Stochastic processes, therefore, serve as a powerful mathematical tool for modeling and understanding these complex and dynamic behaviors. Brownian motion, a fundamental stochastic process, plays a crucial role in the formulation of Stochastic Differential Equations (SDEs) and serves as a cornerstone in various scientific disciplines.

2.2 Definition and Characteristics of Brownian Motion

Brownian motion, first observed by Robert Brown in 1827, refers to the random motion of particles suspended in a fluid. In mathematical terms, it is a continuous-time stochastic process characterized by its key properties:

- **Continuous Paths**: Brownian motion paths are continuous, non-differentiable, and exhibit irregular, erratic movements.
- **Markov Property**: The future behavior of the process is independent of its past given its current state, reflecting the lack of memory in its trajectory.
- **Gaussian Increments**: The increments of Brownian motion over disjoint time intervals are normally distributed, making it a Gaussian process.

- **Stationary Increments**: The statistical properties of Brownian motion remain constant over time, ensuring a consistent stochastic behavior.

Understanding Brownian motion is crucial in comprehending the fundamentals of stochastic processes, laying the groundwork for the formulation of Stochastic Differential Equations.

3 Stochastic Differential Equations

3.1 Formal Definition of SDEs and their Representation

Stochastic Differential Equations (SDEs) provide a mathematical framework to model systems that evolve over time under the influence of random factors. Formally, an SDE is expressed as:

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

where X_t represents the state of the system at time t, $a(t, X_t)$ and $b(t, X_t)$ are deterministic functions, dt denotes the differential of time, and dW_t represents the differential of a Wiener process (Brownian motion).
The inclusion of the stochastic term $b(t, X_t)dW_t$ allows SDEs to incorporate randomness into the evolution of the system, making them suitable for modeling dynamic systems with inherent uncertainty.

3.2 Comparison with Ordinary Differential Equations (ODEs) and Deterministic Models

While Ordinary Differential Equations (ODEs) describe deterministic systems with precise predictions based on initial conditions, SDEs account for the inherent randomness in dynamic processes. In contrast to deterministic models, SDEs offer a more realistic representation of systems influenced by unpredictable external factors. This key distinction is essential in understanding how SDEs enhance our modeling capabilities, enabling the exploration of systems in the presence of randomness, a common feature in many natural and artificial phenomena.

4 Examples of Stochastic Processes and Javascript Simulation

A stochastic process is a collection of random variables indexed by time (or another parameter), where each random variable represents the state of a system at a specific point in time.

4.1 Arithmetic Brownian Motion

Arithmetic Brownian Motion is a stochastic process that describes the linear growth or decay of a variable over time. In finance, it is often used to model the evolution of interest rates or other financial quantities. The equation governing Arithmetic Brownian Motion is given by:

$$dX_t = \mu dt + \sigma dW_t$$

Here, X_t represents the value of the process at time t, μ is the drift or average growth rate, σ is the volatility, dt is the differential of time, and dW_t is the differential of a Wiener process (Brownian motion).

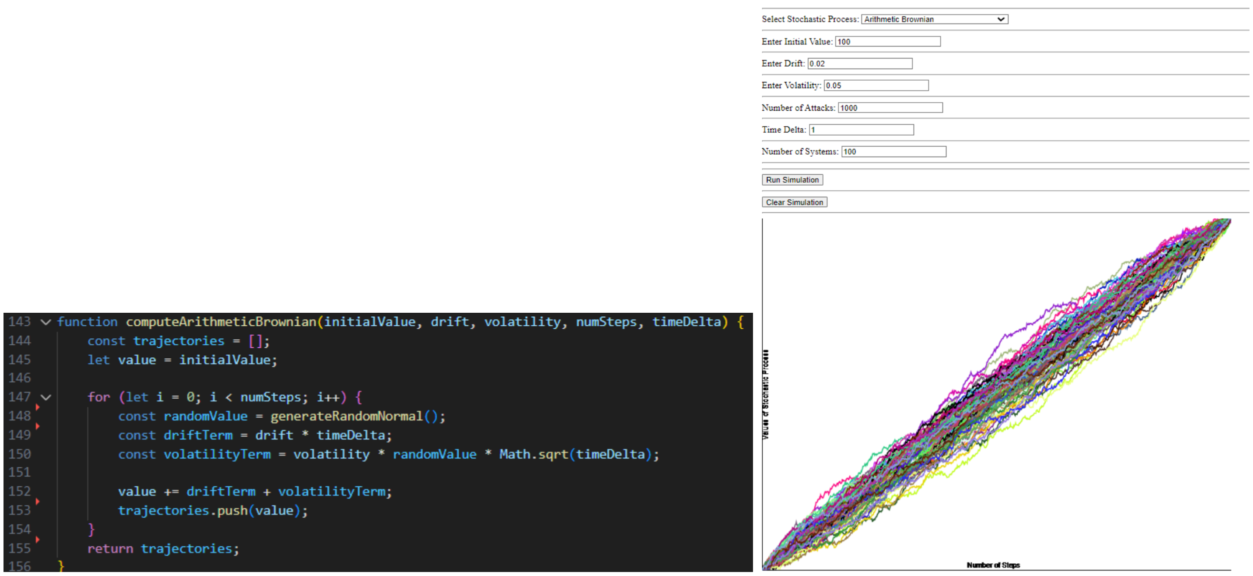


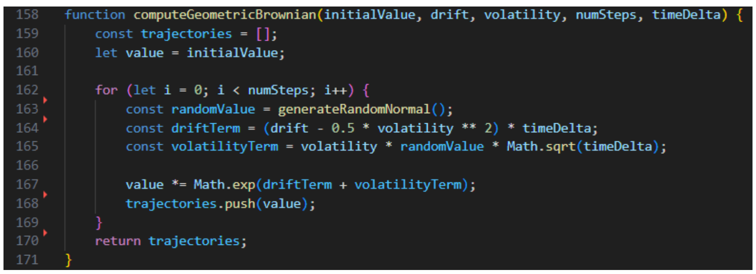
Figure 1: Arithmetic Brownian Motion

4.2 Geometric Brownian Motion (Black–Scholes Model)

Geometric Brownian Motion is a continuous-time stochastic process widely applied in finance, notably in the Black–Scholes option pricing model. This process captures the exponential growth or decay of an asset's value. The equation for Geometric Brownian Motion is:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

Here, S_t denotes the asset price at time t, μ is the average return, σ is the volatility, dt is the differential of time, and dW_t is the differential of a Wiener process.



```
function computeGeometricBrownian(initialValue, drift, volatility, numSteps, timeDelta) {
    const trajectories = [];
    let value = initialValue;

    for (let i = 0; i < numSteps; i++) {
        const randomValue = generateRandomNormal();
        const driftTerm = (drift - 0.5 * volatility ** 2) * timeDelta;
        const volatilityTerm = volatility * randomValue * Math.sqrt(timeDelta);

        value *= Math.exp(driftTerm + volatilityTerm);
        trajectories.push(value);
    }
    return trajectories;
}
```

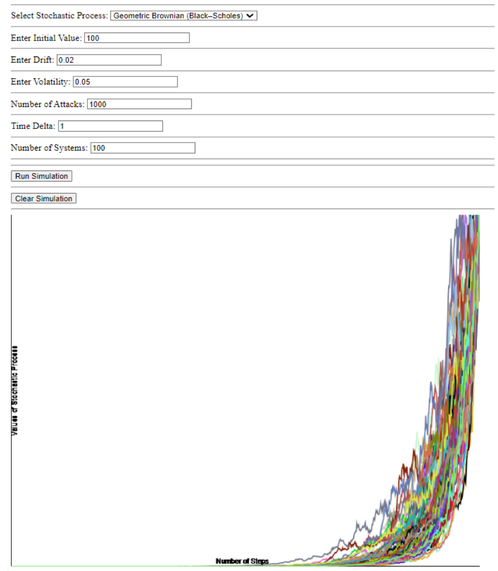


Figure 2: Geometric Brownian Motion

4.3 Ornstein–Uhlenbeck Process (Mean-Reverting)

The Ornstein–Uhlenbeck process models the motion of a particle that tends to return to a central point over time, making it suitable for describing mean-reverting behavior. This process is utilized in physics and finance. The dynamics of the Ornstein–Uhlenbeck process are governed by the equation:

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t$$

Here, θ represents the rate of mean reversion, μ is the mean, σ is the volatility, dt is the differential of time, and dW_t is the differential of a Wiener process.

```javascript
function computeOrnsteinUhlenbeck(meanReversionRate, longTermMean, volatility, numSteps, timeDelta) {
    const trajectories = [];
    let level = 0;

    for (let i = 0; i < numSteps; i++) {
        const randomValue = generateRandomNormal();

        const meanDiff = meanReversionRate * (longTermMean - level) * timeDelta;
        const volatilityDiff = volatility * randomValue * Math.sqrt(timeDelta);
        level += meanDiff + volatilityDiff;
        trajectories.push(level);
    }
    return trajectories;
}
```

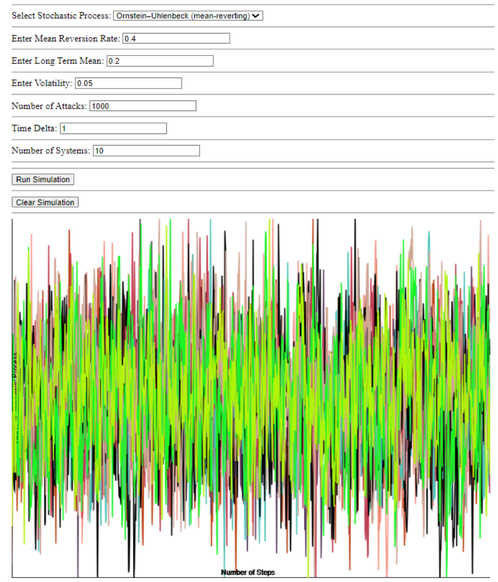


Figure 3: Mean-Reverting

4.4 Poisson Process

The Poisson process is a fundamental counting process used to model the occurrence of events within fixed intervals of time or space. Particularly valuable for rare and independent events, such as arrivals in a queue or phone calls at a call center, the Poisson process is characterized by a constant rate of occurrence (λ). The probability mass function describing the number of events $N(t)$ in a given interval $[0, t]$ is expressed as$P(N(t) = n) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$.

```javascript
function computePoisson(lambda, numIntervals) {
    const trajectories = [];
    let count = 0;

    for (let i = 0; i < numIntervals; i++) {
        const randomValue = Math.random();

        if (randomValue < lambda / numIntervals) {
            count++;
        }

        trajectories.push(count);
    }
    return trajectories;
}
```

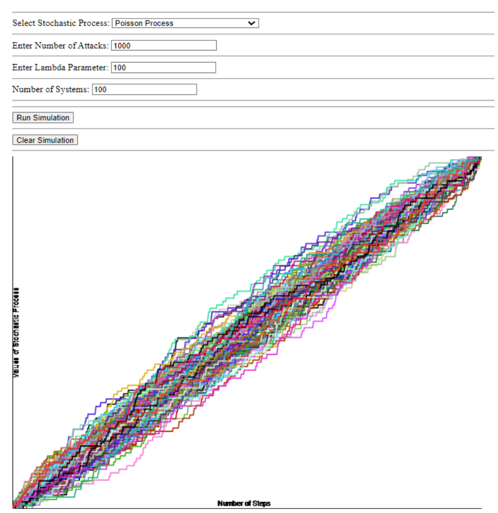


Figure 4: Poisson Process

5 Examples of Stochastic Models and Javascript Simulation

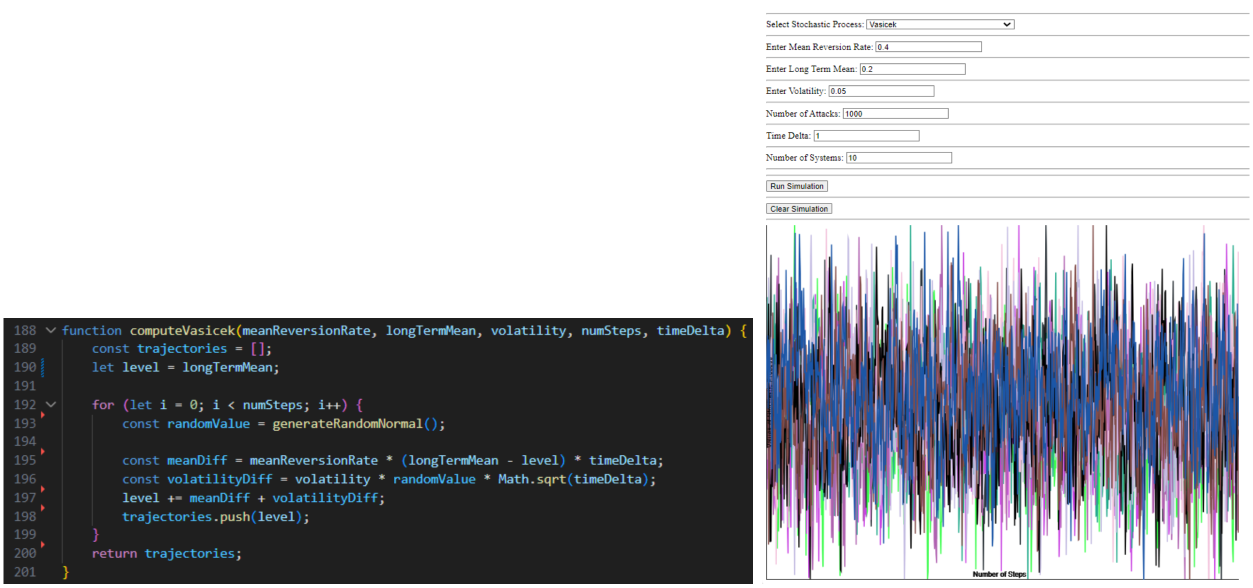
A stochastic model is a mathematical representation of a real-world system that incorporates random elements or uncertainty to describe the system's behavior probabilistically.

5.1 Vasicek Model

The Vasicek model is widely employed in finance for interest rate modeling. It describes the evolution of interest rates in a mean-reverting manner. The stochastic differential equation for the Vasicek model is:

$$dr_t = \alpha(\beta - r_t)dt + \sigma dW_t$$

Here, r_t denotes the interest rate at time t, α is the speed of mean reversion, β is the long-term mean, σ is the volatility, dt is the differential of time, and dW_t is the differential of a Wiener process.



```
function computeVasicek(meanReversionRate, longTermMean, volatility, numSteps, timeDelta) {
    const trajectories = [];
    let level = longTermMean;

    for (let i = 0; i < numSteps; i++) {
        const randomValue = generateRandomNormal();

        const meanDiff = meanReversionRate * (longTermMean - level) * timeDelta;
        const volatilityDiff = volatility * randomValue * Math.sqrt(timeDelta);
        level += meanDiff + volatilityDiff;
        trajectories.push(level);
    }
    return trajectories;
}
```

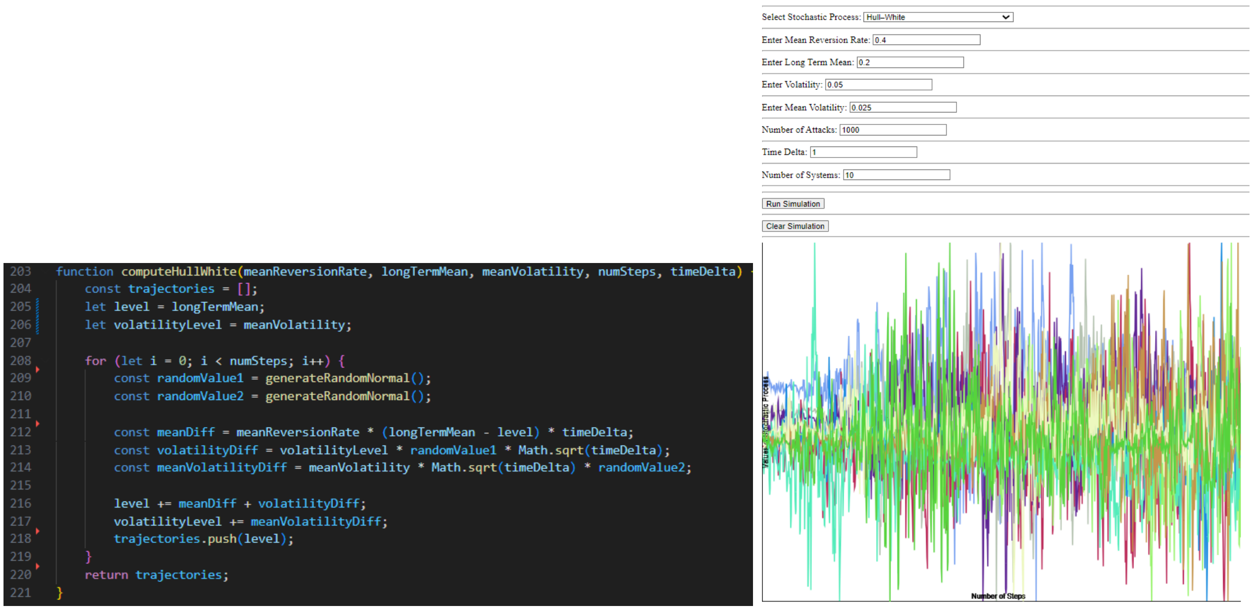

Figure 5: Vasicek Model

5.2 Hull–White Model

The Hull–White model is another interest rate model that extends the Vasicek model by allowing the speed of mean reversion to be time-dependent. The stochastic differential equation for the Hull–White model is:

$$dr_t = (\theta(t) - a(t)r_t)dt + \sigma(t)dW_t$$

Here, $\theta(t)$, $a(t)$, and $\sigma(t)$ are time-dependent parameters.




```javascript
function computeHullWhite(meanReversionRate, longTermMean, meanVolatility, numSteps, timeDelta) {
    const trajectories = [];
    let level = longTermMean;
    let volatilityLevel = meanVolatility;

    for (let i = 0; i < numSteps; i++) {
        const randomValue1 = generateRandomNormal();
        const randomValue2 = generateRandomNormal();

        const meanDiff = meanReversionRate * (longTermMean - level) * timeDelta;
        const volatilityDiff = volatilityLevel * randomValue1 * Math.sqrt(timeDelta);
        const meanVolatilityDiff = meanVolatility * Math.sqrt(timeDelta) * randomValue2;

        level += meanDiff + volatilityDiff;
        volatilityLevel += meanVolatilityDiff;
        trajectories.push(level);
    }
    return trajectories;
}
```

Figure 6: Hull-White Model

5.3 Cox–Ingersoll–Ross (CIR) Model

The Cox–Ingersoll–Ross model is a popular interest rate model that avoids negative interest rates. It introduces mean-reverting behavior with a square root process. The stochastic differential equation for the CIR model is:

$$dr_t = \kappa(\theta - r_t)dt + \sigma\sqrt{r_t}dW_t$$

Here, κ is the speed of mean reversion, θ is the long-term mean, σ is the volatility, dt is the differential of time, and dW_t is the differential of a Wiener process.

```
function computeCIR(meanReversionRate, longTermMean, volatility, numSteps, timeDelta) {
    const trajectories = [];
    let level = longTermMean;

    for (let i = 0; i < numSteps; i++) {
        const randomValue1 = generateRandomNormal();
        const randomValue2 = generateRandomNormal();

        const meanDiff = meanReversionRate * (longTermMean - level) * timeDelta;
        const volatilityDiff = volatility * Math.sqrt(level) * randomValue1 * Math.sqrt(timeDelta);
        const meanVolatilityDiff = 0.25 * volatility ** 2 * timeDelta - (volatility ** 2 / 2) * timeDelta + volatility * Math.sqrt(level) * randomValue2 * Math.sqrt(timeDelta);

        level += meanDiff + volatilityDiff + meanVolatilityDiff;
        trajectories.push(level);
    }
    return trajectories;
}
```

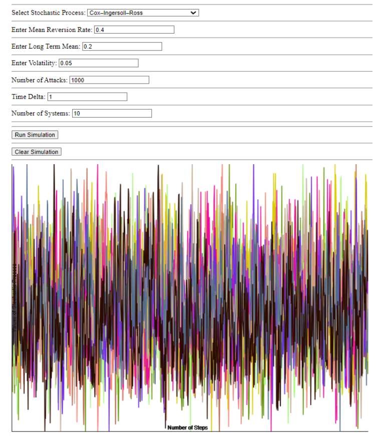


Figure 7: CIR Model

5.4 Black–Karasinski Model

The Black–Karasinski model is an extension of the Vasicek model that allows for time-varying volatility. It is commonly used in interest rate modeling. The stochastic differential equation for the Black–Karasinski model is:

$$dr_t = (\alpha(t) - \beta(t)r_t)dt + \sigma(t)dW_t$$

Here, $\alpha(t)$, $\beta(t)$, and $\sigma(t)$ are time-dependent parameters.

```
function computeBlackKarasinski(meanReversionRate, longTermMean, volatility, eta, lambda, numSteps, timeDelta) {
    const trajectories = [];
    let level = longTermMean;

    for (let i = 0; i < numSteps; i++) {
        const randomValue1 = generateRandomNormal();
        const randomValue2 = generateRandomNormal();

        const meanDiff = meanReversionRate * (longTermMean - level) * timeDelta;
        const volatilityDiff = volatility * Math.sqrt(level) * randomValue1 * Math.sqrt(timeDelta);
        const meanVolatilityDiff = eta * (lambda - level) * timeDelta;
        const volatilityVolatilityDiff = volatility * Math.sqrt(level) * randomValue2 * Math.sqrt(timeDelta);

        level += meanDiff + volatilityDiff + meanVolatilityDiff + volatilityVolatilityDiff;
        trajectories.push(level);
    }
    return trajectories;
}
```

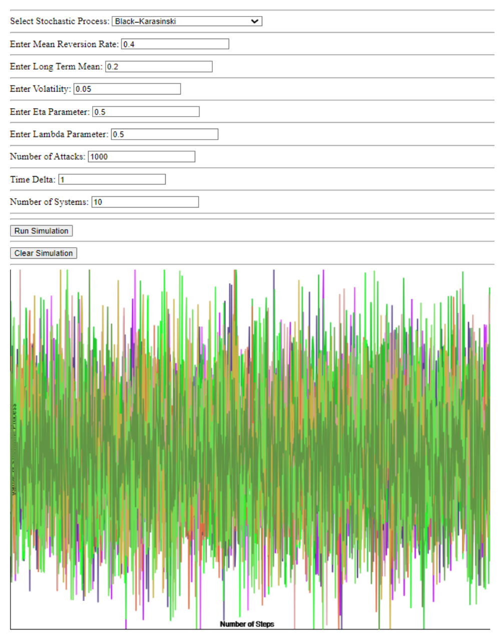


Figure 8: Black–Karasinski Model

5.5 Heston Model

The Heston model is a stochastic volatility model frequently used in financial derivatives pricing. It describes the dynamics of both the asset price and its volatility. The stochastic differential equations for the Heston model are:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} dW_t^v$$

Here, S_t is the asset price, v_t is the volatility, μ is the average return, κ is the speed of mean reversion for volatility, θ is the long-term mean volatility, σ is the volatility of volatility, dt is the differential of time, dW_t^S is the differential of a Wiener process for the asset price, and dW_t^v is the differential of a Wiener process for the volatility.

```
function computeHeston(initialValue, meanReversionRate, longTermVolatility, volatilityOfVolatility, correlation, numSteps, timeDelta) {
    const trajectories = [];
    let value = initialValue;
    let volatility = longTermVolatility;

    for (let i = 0; i < numSteps; i++) {
        const randomValue1 = generateRandomNormal();
        const randomValue2 = generateRandomNormal();

        const priceDiff = meanReversionRate * (longTermVolatility - volatility) * timeDelta;
        const volatilityDiff = volatilityOfVolatility * Math.sqrt(volatility) * randomValue1 * Math.sqrt(timeDelta);
        const priceVolatilityCorrelation = correlation * volatility * randomValue2 * Math.sqrt(timeDelta);

        value *= Math.exp(priceDiff - 0.5 * volatility * timeDelta + priceVolatilityCorrelation);
        volatility += volatilityDiff;
        trajectories.push({ value, volatility });
    }
    return trajectories;
}
```

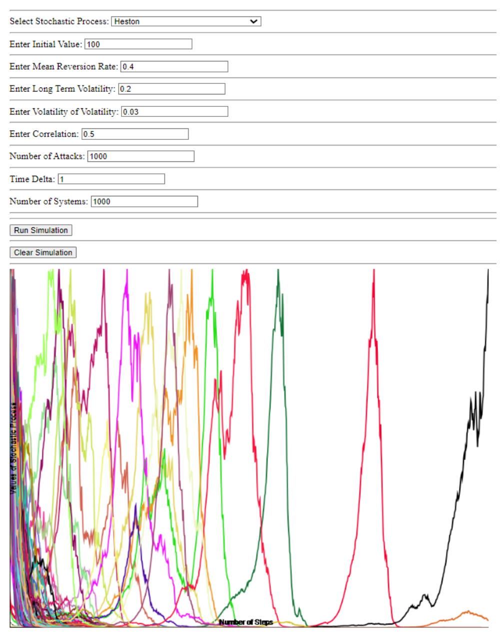


Figure 9: Heston Model

5.6 Chen Model

The Chen model is a stochastic volatility model designed to capture the dynamics of asset prices with jumps. It incorporates both stochastic volatility and jump components. The stochastic differential equations for the Chen model are:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S + dJ_t$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t}dW_t^v$$

Here, S_t is the asset price, v_t is the volatility, μ is the average return, κ is the speed of mean reversion for volatility, θ is the long-term mean volatility, σ is the volatility of volatility, dt is the differential of time, dW_t^S is the differential of a Wiener process for the asset price, dW_t^v is the differential of a Wiener process for the volatility, and dJ_t represents the jump component.

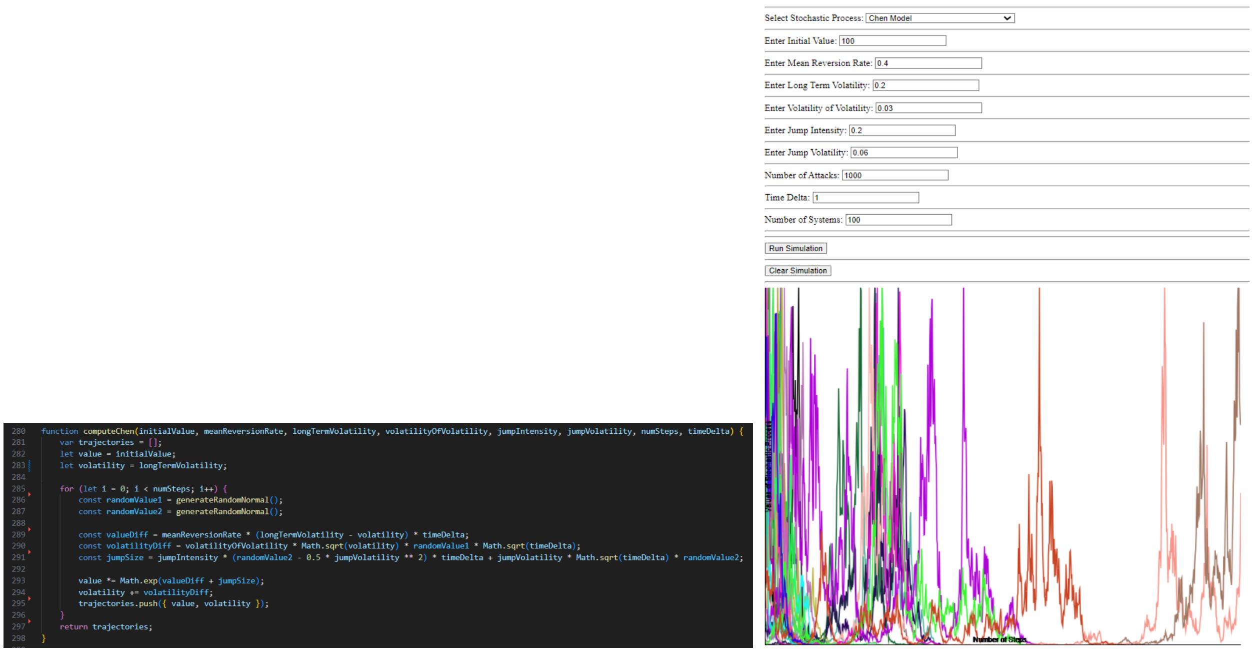


Figure 10: Chen Model

6 Examples of Simulation Schemes

Simulation schemes are numerical methods used to approximate solutions to stochastic differential equations (SDEs). These equations involve random processes, and simulating them requires careful consideration of the stochastic terms. These methods provide a practical means of approximating solutions to SDEs, and the choice among them depends on the specific characteristics of the problem, the desired level of accuracy, and computational considerations.

6.1 Euler-Maruyama Method

The Euler-Maruyama method is a numerical approach used to approximate solutions to stochastic differential equations (SDEs). It extends the classical Euler method for ordinary differential equations (ODEs) to handle the stochastic components in SDEs. For an SDE of the form $dX_t = a(X_t)dt + b(X_t)dW_t$, where X_t is the solution process, $a(X_t)$ is the drift coefficient, $b(X_t)$ is the diffusion coefficient, dt is the differential of time, and dW_t is the Wiener process increment, the Euler-Maruyama method updates the solution as follows:

$$X_{t+\Delta t} = X_t + a(X_t)\Delta t + b(X_t)\sqrt{\Delta t}Z_{t+1},$$

where Δt is the time step, and Z_{t+1} is a random variable sampled from a normal distribution.

```javascript
function computeEulerMaruyama(initialValue, numSteps, timeDelta) {
    const trajectories = [];
    let value = initialValue;

    for (let i = 0; i < numSteps; i++) {
        const randomValue = generateRandomNormal();

        const driftTerm = 0.2 * timeDelta;
        const diffusionTerm = 0.5 * randomValue * Math.sqrt(timeDelta);

        value += driftTerm + diffusionTerm;
        trajectories.push(value);
    }
    return trajectories;
}
```

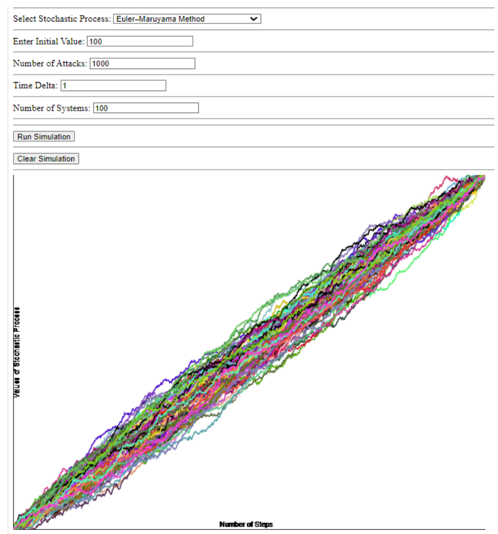


Figure 11: Euler-Maruyama Method

6.2 Milstein Method

The Milstein method is an extension of the Euler-Maruyama method, particularly beneficial when the drift and diffusion coefficients have derivatives with respect to the state variables. This method enhances the accuracy of the approximation. For the same SDE, the Milstein method updates the solution as follows:

$$X_{t+\Delta t} = X_t + a(X_t)\Delta t + b(X_t)\sqrt{\Delta t}Z_{t+1} + \frac{1}{2}b(X_t)b'(X_t)((Z_{t+1})^2 - 1)\Delta t,$$

where $b'(X_t)$ is the derivative of $b(X_t)$ with respect to X_t.

```
function computeMilstein(initialValue, numSteps, timeDelta) {
    const trajectories = [];
    let value = initialValue;

    for (let i = 0; i < numSteps; i++) {
        const randomValue = generateRandomNormal();

        const driftTerm = 0.2 * timeDelta;
        const diffusionTerm = 0.1 * randomValue * Math.sqrt(timeDelta);
        const correctionTerm = 0.5 * 0.2  * ((randomValue * randomValue) - 1) * timeDelta;

        value += driftTerm + diffusionTerm + correctionTerm;
        trajectories.push(value);
    }

    return trajectories;
}
```

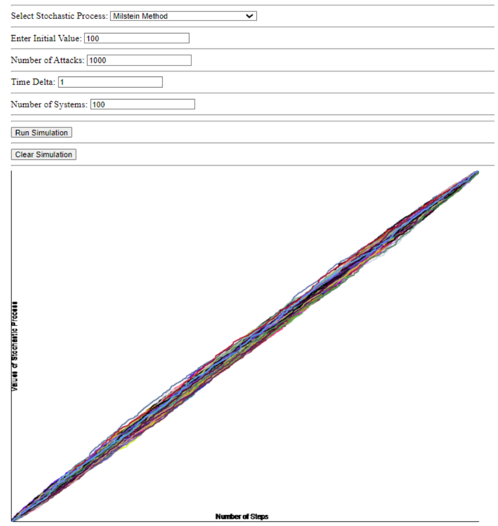


Figure 12: Milstein Method

6.3 Runge-Kutta Methods

Runge-Kutta methods are a family of numerical techniques that offer higher-order accuracy compared to Euler-based methods. For an SDE, Runge-Kutta methods update the solution using a set of coefficients and intermediate steps. The general update formula for a Runge-Kutta method is:

$$X_{t+\Delta t} = X_t + \sum_{i=1}^{s} b_i k_i,$$

where k_i are the increments determined by evaluating the drift and diffusion coefficients at different points.

```
function computeRungeKutta(initialValue, numSteps, timeDelta) {
    const trajectories = [];
    let value = initialValue;

    for (let i = 0; i < numSteps; i++) {
        const random1 = generateRandomNormal();
        const random2 = generateRandomNormal();
        const k1 = 0.1 + 0.2 * random1;  // Arithmetic Brownian Motion increment
        const k2 = 0.1 + 0.2 * (random1 + random2) / 2;
        const k3 = 0.1 + 0.2 * (random1 + random2) / 2;
        const k4 = 0.1 + 0.2 * (random1 + random2);

        const increment = (k1 + 2 * k2 + 2 * k3 + k4) * timeDelta / 6;
        value += increment;
        trajectories.push(value);
    }
    return trajectories;
}
```

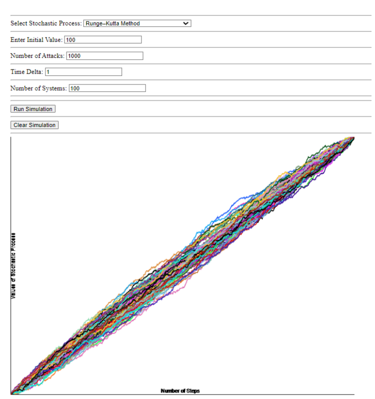


Figure 13: Runge-Kutta Methods

6.4 Heun's Method

Heun's method is a specific Runge-Kutta method that provides second-order accuracy. For an SDE, Heun's method updates the solution as follows:

$$X_{t+\Delta t} = X_t + \frac{1}{2}[a(X_t) + a(X^{(0)}_{t+\Delta t})]\Delta t + \frac{1}{2}[b(X_t) + b(X^{(0)}_{t+\Delta t})]\sqrt{\Delta t} Z_{t+1},$$

where $X^{(0)}_{t+\Delta t} = X_t + a(X_t)\Delta t + b(X_t)\sqrt{\Delta t} Z_{t+1}$.

Figure 14: Heun's Method

7 Interactive Web Application for SDE Simulation

In the realm of stochastic differential equations (SDEs), the ability to visualize and interact with simulations plays a crucial role in enhancing comprehension and practical application. An interactive web application designed for simulating SDEs provides a dynamic platform for users to explore the behavior of stochastic processes in real-time.
Key features of this application are:

1. **Parameter Adjustment**: Users can dynamically adjust parameters such as drift coefficients, diffusion coefficients, and initial conditions.

2. **Multiple SDE Models**: The application can support various SDE models, enabling users to choose and compare different stochastic processes.

3. **Simulation Schemes**: Users can select different simulation schemes.
4. **Visual Representation**: The application provides visual representations of the simulated paths in the form of plots.

The web application can be reached following this link: SDEs Simulator

8 Conclusion

On a final note, with Norbert Wiener's groundbreaking work on Brownian motion, the development of SDEs in the 1950s marked a significant milestone in the field of stochastic analysis.
The significance of SDEs in capturing randomness within dynamic systems is profound. Many real-world phenomena exhibit inherent randomness or uncertainty, which classical deterministic models fail to capture. SDEs, with their ability to incorporate stochastic processes like Brownian motion, offer a more realistic representation of systems evolving over time under the influence of random factors.
As we recapitulate the key findings, it becomes evident that stochastic processes and SDEs offer a powerful and flexible framework for modeling the inherent randomness in dynamic systems. From the foundational principles to practical applications and numerical methods, this thesis has provided a comprehensive exploration of this mathematical landscape.
To conclude, the relevance of stochastic processes and SDEs in the age of data and computing is undeniably paramount. Their ability to model complex, dynamic, and random systems positions them as indispensable tools for researchers and practitioners across diverse disciplines. This understanding forms a necessary foundation for navigating and deciphering the probabilistic nature of the world around us. For this reason, the journey through stochastic processes and SDEs is not merely a mathematical exploration but an essential aspect of comprehending and navigating the intricate dynamics of our ever-changing world.