Ito Integration and Calculus, Concept and Didactical Simulations

Pietro Colaguori 1936709

November 2023

1 Concept

Ito calculus is a major topic in the field of stochastic calculus, which is a particular area of mathematics that deals with stochastic differential equations (SDEs).

When we talk about Ito integration we have to keep in mind the following key concepts:

- **Stochastical integral**: The Ito integral extends the classical integral, which is based on the idea of summing up infinitesimally small quantities, to include terms involving the Brownian motion or other stochastic processes. Mathematically:

  $$Y_t = \int_0^t H_s dX_s$$

  where H is a locally square-integrable process and X is a Brownian motion.

- **Stochastic differential equations (SDEs)**: This is where Ito integration comes in, as it is often used to solve SDEs, which describe the evolution of a stochastic process.

- **Ito process**: An Ito process is defined to be an adapted stochastic process that can be expressed as the sum of an integral with respect to Brownian motion and an integral with respect to time. Mathematically:

  $$X_t = X_0 + \int_0^t \sigma_s dB_s + \int_0^t \mu_s ds$$

  Here, B is a Brownian motion and it is required that σ is a predictable B-integrable process, and μ is predictable and integrable.

Talking about Ito's calculus instead, we need to consider these two fundamental properties:

1. **Ito's Lemma**: It's the analogous of the chain rule in classical calculus, allowing us to differentiate functions of stochastic processes. The lemma's statement is the following:

$$df(t, X(t)) = \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\mu(t, X(t)) + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\sigma^2(t, X(t))\right) dt + \frac{\partial f}{\partial x}\sigma(t, X(t))dW(t)$$

   Where $W(t)$ is a Brownian motion, $\mu(t, X(t))$ is the drift coefficient and $\sigma(t, X(t))$ is the diffusion coefficient.

2. **Ito Isometry**: This property expresses the relationship between the square of a stochastic integral and the integral of the square of the integrand. It can also be seen as a consequence of the martingale property of the Brownian motion. Mathematically:

$$E[(\int_0^T X_t dW_t)^2] = E[\int_0^T X_t^2 dt]$$

2 Simulations

I arranged a simulation in Python to compute the Ito's integral of t^3. I create a function to generate some random numbers from a normal distribution and scale them with the square root, then I generate the Brownian motion increments by using the previously mentioned function. After defining the integrand I compute the Ito's integral with the partial sums. I then plot the values of the Ito's integral with respect to the time interval. Of course, because of the randomness brought by the Brownian motion, the graphs are different each time. Note that the number of steps n determines how finely the Ito's integral is approximated.

```python
import streamlit as st
import numpy as np
import matplotlib.pyplot as plt

# Function to generate Brownian increments
def brownian_step(num_steps, dt):
    return np.sqrt(dt) * np.random.normal(size=num_steps)

# Function to simulate Ito integral
def simulate_ito_integral(num_steps, dt):
    t_values = np.linspace(0, num_steps * dt, num_steps + 1)
    brownian_increments = brownian_step(num_steps, dt)
    integrand_values = t_values**3  # Integrand function: t^3
    ito_integral_values = np.cumsum(integrand_values[:-1] * brownian_increments)
```

```python
    return t_values[:-1], ito_integral_values

# Streamlit app
def main():
    st.title("Ito Integration Simulation")

    # User input
    num_steps = st.slider("Number of Steps",
        min_value=10, max_value=1000, value=100,
        step=10)

    # Simulation
    dt = 1.0 / num_steps
    t_values, ito_integral_values =
        simulate_ito_integral(num_steps, dt)

    # Plotting
    fig, ax = plt.subplots()
    ax.plot(t_values, ito_integral_values, label="Ito
        Integral")
    ax.set_xlabel("Time")
    ax.set_ylabel("Value")
    ax.legend()

    # Display the plot using Streamlit
    st.pyplot(fig)

if __name__ == "__main__":
    main()
```

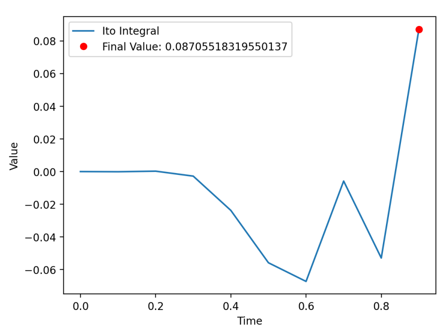

Figure 1: Simulation run with $n = 10$.

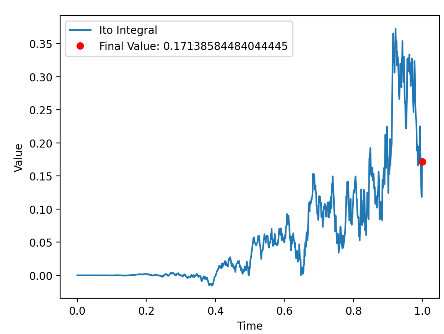

Figure 2: Simulation run with $n = 1000$.