Monte Carlo

November 26, 2022

[1]:

```
# numerical libraries
import numpy as np
import scipy
import matplotlib.pyplot as plt
%matplotlib inline
```

[3]:

```
np.random.seed(10) # reproducibility
Delta = 1
dt = 0.001
t = np.arange(0, Delta+dt, dt)
Nt = len(t)
# simulate two Wiener processes
dW1 = np.random.normal(size=Nt)
dW2 = np.random.normal(size=Nt)
W1 = np.cumsum(dW1)
W2 = np.cumsum(dW2)
W1_Delta = np.random.randn()
W2_Delta = np.random.randn()
# terms in approximation
Xi1 = W1_Delta
Xi2 = W2_Delta
# order of approximation
p = 50
mean = 0
nmc = 10**4
for _ in range(nmc):
    # generate all coefficients
    a_jr = np.zeros([2, p])
    b_jr = np.zeros([2, p])
    zeta_jr = np.zeros([2, p])
    eta_jr = np.zeros([2, p])
    for idx in range(p):
        r = idx + 1
        a_jr[:, idx] = np.random.normal(0, 1/(2*((np.pi)**2) * (r**2)), size=2)
        b_jr[:, idx] = np.random.normal(0, 1/(2*((np.pi)**2) * (r**2)), size=2)
        zeta_jr[:, idx] = np.sqrt(2)*np.pi*r*a_jr[:, idx]
        eta_jr[:, idx] = np.sqrt(2)*np.pi*r*b_jr[:, idx]
```

```
    a10 = -np.sqrt(2)/np.pi * np.sum((1 / np.arange(1, p+1)) * zeta_jr[0, :]) #␣
↪ignore tail series
    a20 = -np.sqrt(2)/np.pi * np.sum((1 / np.arange(1, p+1)) * zeta_jr[1, :])
    Ap_12 = (0.5/np.pi) * np.sum(( (1 / np.arange(1, p+1) ) * (zeta_jr[0, :] *␣
↪eta_jr[1, :] - \
                                                          eta_jr[0, :] *␣
↪zeta_jr[1, :])))
    Jp_12 = 0.5*Xi1*Xi2 - 0.5*(a20*Xi1 - a10*Xi2) + Ap_12
    result = W1_Delta * Jp_12
    mean = mean + result / nmc
mean
```

[3]: -0.0009368538586441822