STOCHASTIC PROCESSES

HW05

Hanie Hatami(99100614)
Stochastic processes course

Contents

1 Abstract

In this exercise, we want to generate the data that we used in the previous exercise and get the Kramer's coefficients, this time we will get the coefficients again and make a small comparison.

2 Results

2.1 Binning:

2.1.1 $D^{(1)}$

Calculated data:

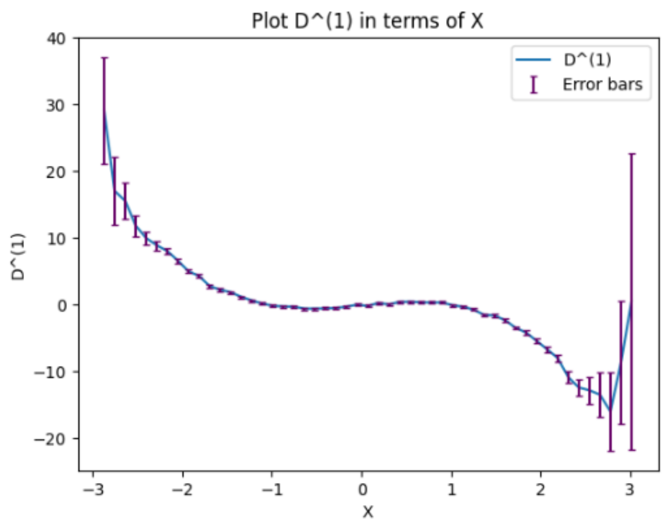

Figure 1: $D^{(1)}$ in terms of X

Given data(previous exercise):

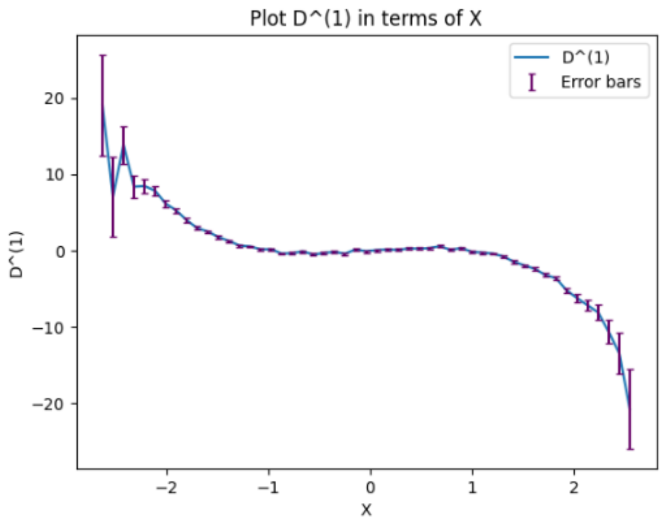

Figure 2: $D^{(1)}$ in terms of X

2.1.2 $D^{(2)}$

Calculated data:

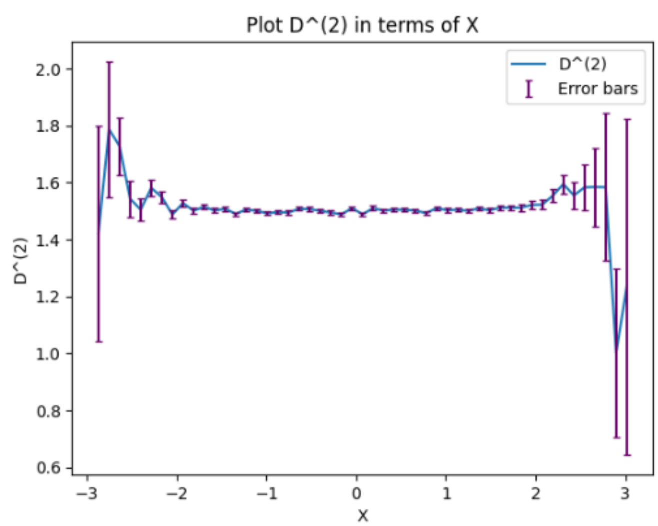

Figure 3: $D^{(2)}$ in terms of X

Given data(previous exercise):

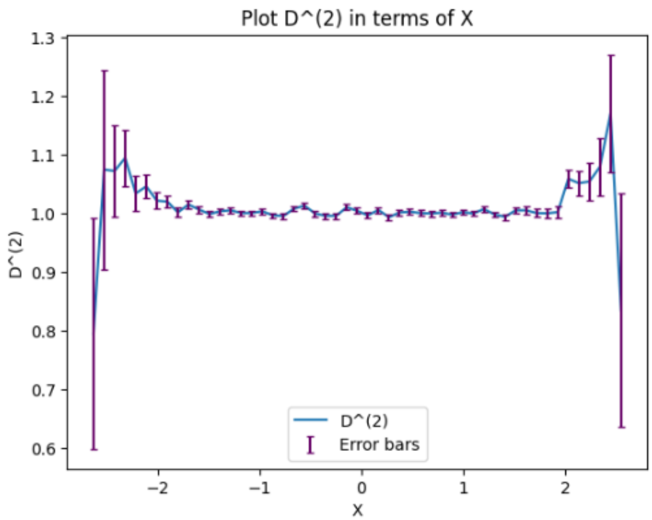

Figure 4: $D^{(2)}$ in terms of X

2.2 Interaction Matrix:

2.2.1 $D^{(1)}$

Calculated data:

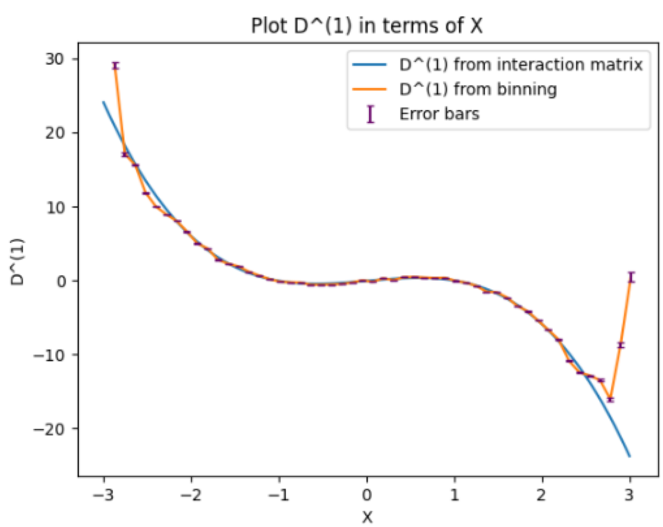

Figure 5: $D^{(1)}$ in terms of X

Given data(previous exercise):

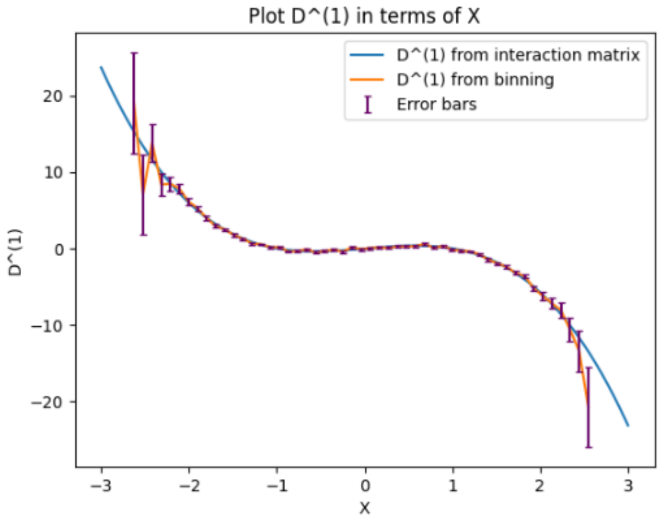

Figure 6: $D^{(1)}$ in terms of X

2.2.2 $D^{(2)}$

Calculated data:

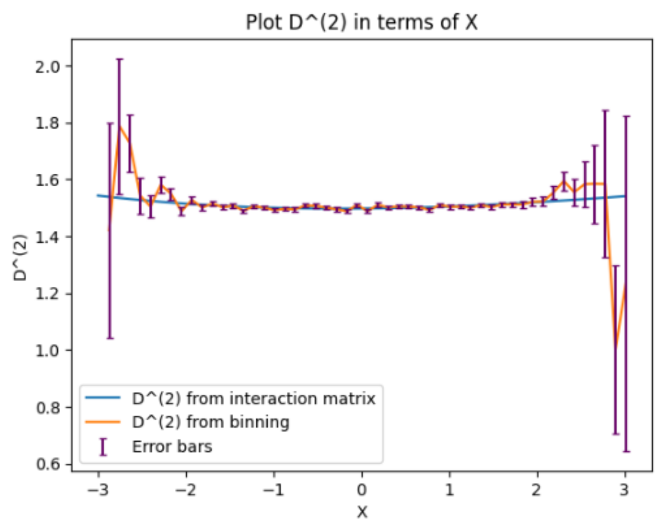

Figure 7: $D^{(2)}$ in terms of X

Given data(previous exercise):

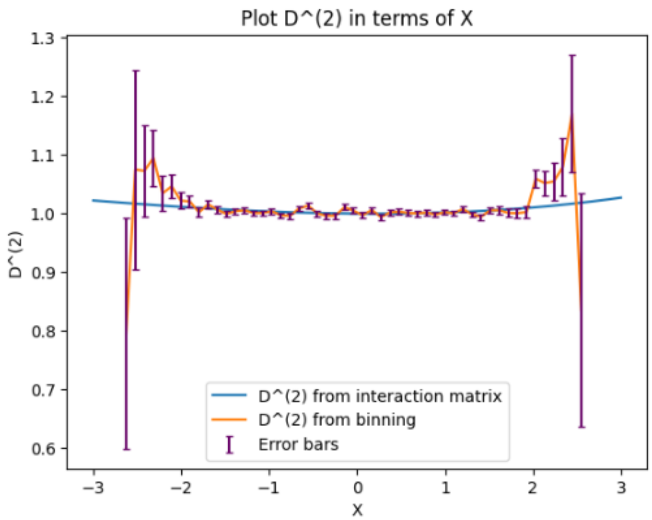

Figure 8: $D^{(2)}$ in terms of X

3 Conclusion

As it is clear, the generated data results are very close to the data we received.(The formula used to produce the same data has only slightly changed the noise.)
As a result, the explanation of this section is the same as the previous exercise.

4 Code

```
# Import needed libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Generationg noise
n = 3000000
etha = np.random.normal(0,1,n)

#Generating data:
x=0.1
dt= 0.001
data = []
for i in range(n):
  dx = (x-x**3)*dt + ((3*dt)**(1/2))*etha[i]
  x= x+dx
  data.append(x)

#Binning
# Calculating drift and diffusion
df = data
bins = 51
hist_values, bin_edges = np.histogram(df, bins=bins)
max = np.max(df)
min = np.min(df)
# Calculation diffrences
tool = len(df)
diffrences = []
binmid_list = []
for i in range(bins):
  binmid = (bin_edges[i] + bin_edges[i+1]) /2
  binmid_list.append(binmid)
  diff = []
  for j in range(tool):
    if df[j]>= bin_edges[i] and df[j] < bin_edges[i+1] and j != tool-1:
      ekhtelaf = df[j+1] - df[j]
      diff.append(ekhtelaf)
  diffrences.append(diff)

# Calculating D^(1)
D1_dt = []
errors_dt =[]
for j in range(bins):
  d1 = (sum(diffrences[j]))/len(diffrences[j])
  res = pd.Series(diffrences[j]).var()
  sem = (res/len(diffrences[j]))**(1/2)
  errors_dt.append(sem)
  D1_dt.append(d1)

D1 = [i * 1000 for i in D1_dt]
errors = [i * 1000 for i in errors_dt]
plt.plot(binmid_list,D1, label='D^(1)')
plt.errorbar(binmid_list, D1, yerr=errors, fmt='none', color='#660066',
    capsize=2, label='Error bars')
```

```
plt.legend()
plt.xlabel("X")
plt.ylabel("D^(1)")
plt.title('Plot D^(1) in terms of X')
plt.show()

# Calculating D^(2)
D2_dt = []
errors_dt =[]
for j in range(bins):
  double_list = np.power(diffrences[j], 2)
  d1 = (sum(double_list))/len(diffrences[j])
  res = pd.Series(double_list).var()
  sem = (res/len(double_list))**(1/2)
  errors_dt.append(sem)
  D2_dt.append(d1)

D2 = [i * (1000/2) for i in D2_dt]
errors = [i * (1000/2) for i in errors_dt]
plt.plot(binmid_list,D2, label='D^(2)')
plt.errorbar(binmid_list, D2, yerr=errors, fmt='none', color='#660066',
    capsize=2, label='Error bars')
plt.legend()
plt.xlabel("X")
plt.ylabel("D^(2)")
plt.title('Plot D^(2) in terms of X')
plt.show()

#Interaction matrix
data = np.array(data)
diffr = []
for i in range(len(data)-1):
  diff = data[i+1]-data[i]
  diffr.append(diff)



y1 = [map(lambda x, y: x * y**i, diffr, data) for i in range(4)]

tau=1
y = data[tau:] - data[:-tau]
data1 = np.ones(len(data))
ys = np.zeros(4)
for i in range(4):
  ys[i] = np.mean(y)
  y *= data[:-1]


mmnts = np.ones(7)
xk = np.ones(len(data))
for i in range(7):
  mmnts[i] = xk.mean()
  xk *= data

A = np.zeros((4,4))
for i in range(4):
  A[i] = np.roll(mmnts, -i)[:4]
```

```

A_inv = np.linalg.inv(A)
phis = A_inv@ys
phis = phis/0.001
x = np.linspace(-3,3, 10001)

plt.plot(x,np.array([phis[i]*x**i for i in range(4)]).sum(0), label='D^(1)
    from interaction matrix')
plt.plot(binmid_list,D1, label='D^(1) from binning')
plt.errorbar(binmid_list, D1, yerr=errors, fmt='none', color='#660066',
    capsize=2, label='Error bars')
plt.legend()
plt.xlabel("X")
plt.ylabel("D^(1)")
plt.title('Plot D^(1) in terms of X')
plt.show()

tau=1
y = data[tau:] - data[:-tau]
y = y**2
ys2 = np.zeros(4)
for i in range(4):
  ys2[i] = np.mean(y)
  y *= data[:-1]


phis2 = A_inv@ys2
phis2 = phis2/0.002
x = np.linspace(-3,3, 10001)

plt.plot(x,np.array([phis2[i]*x**i for i in range(4)]).sum(0), label='D^(2)
    from interaction matrix')
plt.plot(binmid_list,D2, label='D^(2) from binning')
plt.errorbar(binmid_list, D2, yerr=errors, fmt='none', color='#660066',
    capsize=2, label='Error bars')
plt.legend()
plt.xlabel("X")
plt.ylabel("D^(2)")
plt.title('Plot D^(2) in terms of X')
plt.show()
```