STOCHASTIC PROCESSES

HW04

Hanie Hatami(99100614)
Stochastic processes course

Contents

1 Abstract

In this exercise, we want to plot $D^{(1)}$, $D^{(2)}$ and $D^{(4)}$ in terms of x once with binning and with error bars, and once with interaction matrix.

2 Results

2.1 Binning:

2.1.1 $D^{(1)}$

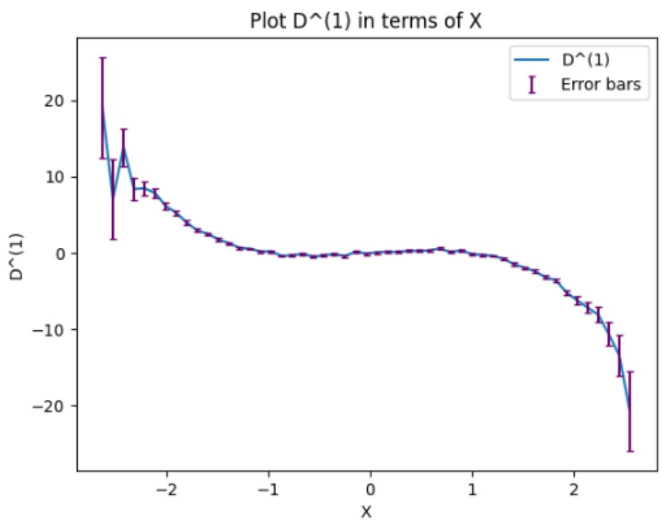

Figure 1: $D^{(1)}$ in terms of X

2.1.2 $D^{(2)}$

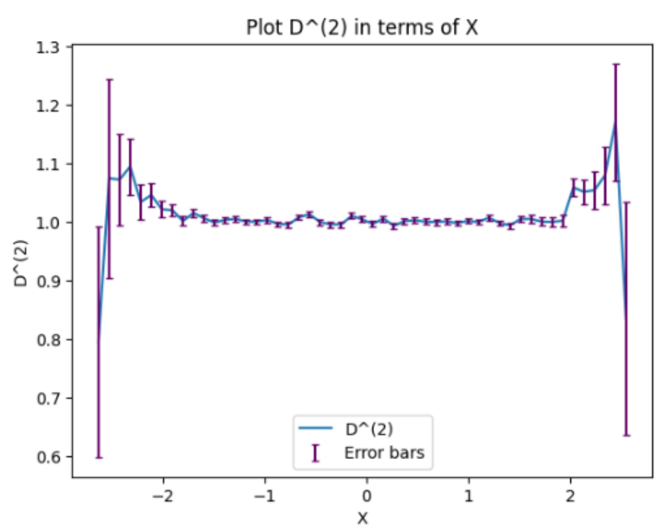

Figure 2: $D^{(2)}$ in terms of X

2.1.3 $D^{(4)}$

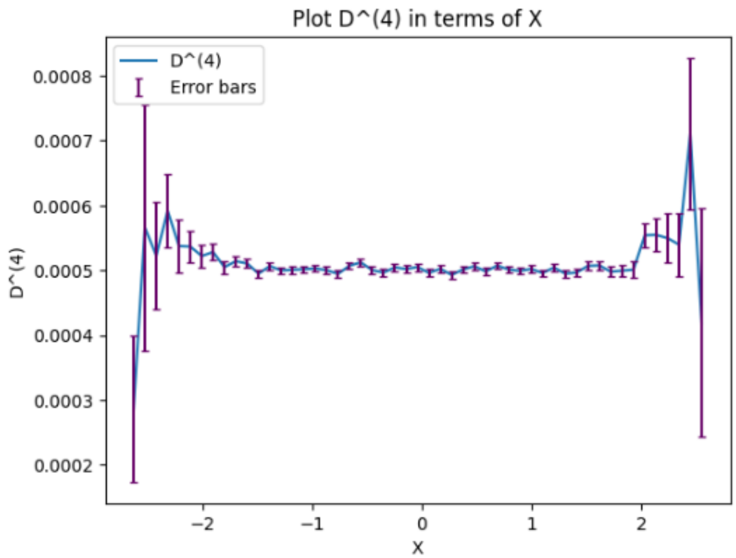

Figure 3: $D^{(4)}$ in terms of X

2.2 Interaction Matrix:

2.2.1 $D^{(1)}$

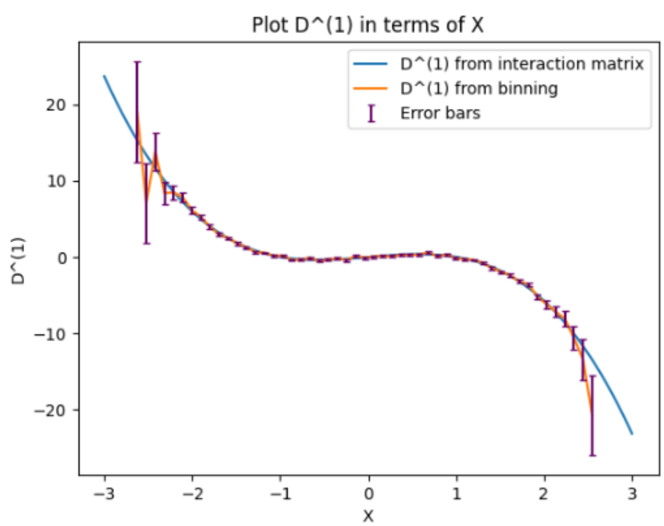

Figure 4: $D^{(1)}$ in terms of X

2.2.2 $D^{(2)}$

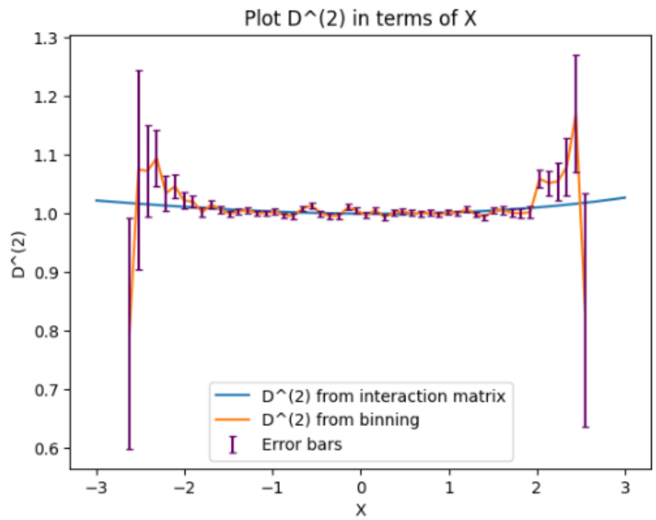

Figure 5: $D^{(2)}$ in terms of X

2.2.3 $D^{(4)}$

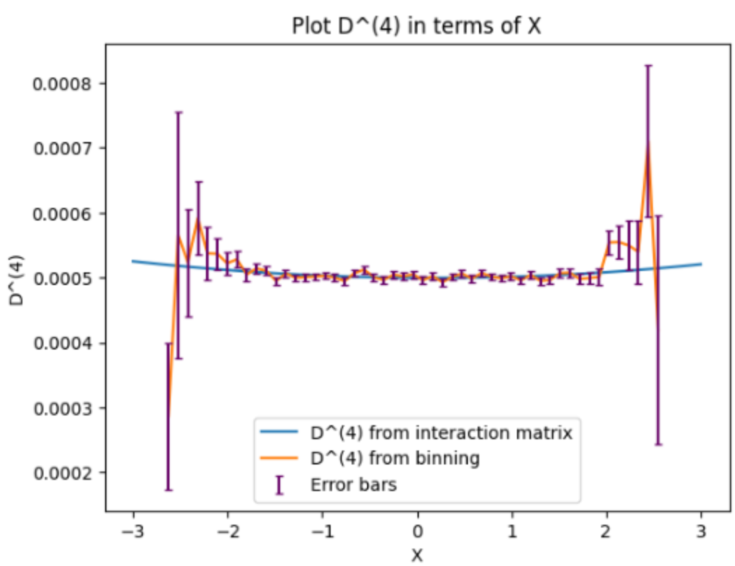

Figure 6: $D^{(4)}$ in terms of X

3 Conclusion

$D^{(1)}$ represents our trend. Plot $D^{(1)}$ shows us that the data tends to go up in very negative X's and go down in very positive X's, but in small and medium X's, the probability of increasing and decreasing X is the same, and the data between There has no difference between going up and down.

$D^{(2)}$ shows the intensity of these differences and movements. As if it shows the slope of the data. As we can see in the $D^{(2)}$ plot, in very large and very small X's, the value of $D^{(2)}$ is lower, which indicates that the intensity of the differences is less, but in the middle X's, we see the highest value of $D^{(2)}$ and in the value of 2, which indicates that Jumps are more intense.

Since our $D^{(4)}$ is not zero, we have jump in the data and it is not continuous.

As we can see, the results we obtained from the interaction matrix part are very consistent and similar to the results we obtained by binning, and we can conclude from.

4 Code

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

# Mount drive to Read data and plot it
# Load get data
from google.colab import drive
drive.mount('/content/drive')

# Read data and plot it
# Load data
df = np.loadtxt('/content/drive/MyDrive/data/0-001.txt')

#Plot the data
plt.plot(df)
bins = 51

# Calculating needed values
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()

# Calculating D^(4)
D4_dt = []
errors_dt =[]
for j in range(bins):
  power4_list = np.power(diffrences[j], 4)
  d1 = (sum(power4_list))/len(diffrences[j])
  res = pd.Series(power4_list).var()
  sem = (res/len(power4_list))**(1/2)
  errors_dt.append(sem)
  D4_dt.append(d1)

D4 = [i * (1000/24) for i in D4_dt]
errors = [i * (1000/24) for i in errors_dt]
plt.plot(binmid_list,D4, label='D^(4)')
plt.errorbar(binmid_list, D4, yerr=errors, fmt='none', color='#660066',
    capsize=2, label='Error bars')
plt.legend()
plt.xlabel("X")
plt.ylabel("D^(4)")
plt.title('Plot D^(4) in terms of X')
plt.show()

data = df
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)")
```

```python
plt.title('Plot D^(2) in terms of X')
plt.show()

tau=1
y = data[tau:] - data[:-tau]
y = y**4
ys4 = np.zeros(4)
for i in range(4):
  ys4[i] = np.mean(y)
  y *= data[:-1]


phis4 = A_inv@ys4
phis4 = phis4/0.024
x = np.linspace(-3,3, 10001)

plt.plot(x,np.array([phis4[i]*x**i for i in range(4)]).sum(0), label='D^(4) from interaction matrix')
plt.plot(binmid_list,D4, label='D^(4) from binning')
plt.errorbar(binmid_list, D4, yerr=errors, fmt='none', color='#660066', capsize=2, label='Error bars')
plt.legend()
plt.xlabel("X")
plt.ylabel("D^(4)")
plt.title('Plot D^(4) in terms of X')
plt.show()
```