
ESTIMATION OF KM CONDITIONAL MOMENTS USING STATISTICAL MOMENTS

HW07

Hanie Hatami(99100614)
Stochastic processes course

Contents

1	Abstract	3
2	Results	3
2.1	Data	3
2.2	$D^1(x)$	4
2.3	$D(x)$	4
2.4	$D^4(x)$	5
2.5	$D^6(x)$	5
2.6	ξ	6
2.7	λ	6
3	Code	7

1 Abstract

In the study of stochastic processes, understanding the underlying dynamics of systems influenced by random forces is of paramount importance. One powerful method to analyze such systems is through the Kramers-Moyal (KM) expansion, which provides a way to describe the evolution of the probability distribution of a stochastic variable. In this exercise we want to estimate KM coefficient with moments.

2 Results

2.1 Data

We generate data by jump-diffusion process, which is given by a (It^o) dynamical stochastic equation:

$$dx(t) = D^{(1)}(x, t)dt + \sqrt{D(x, t)}dW(t) + \xi dJ(t) \quad (1)$$

- $x_1(0)$ is 0.1.
- Time step of dt is 0.001
- Numbers of generated data is 1000000
- $dW(t)$ is wiener.
- ξ is the size of jump with gaussian distribution with mean 0 and variance 1.
- $J(t)$ is a Poisson jump process with jump rate λ

After generating the data, we draw the data graphs and it is as follows.

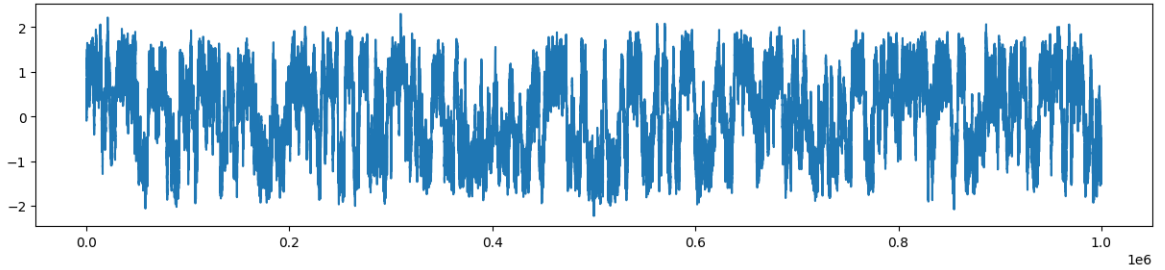


Figure 1: Data according to time

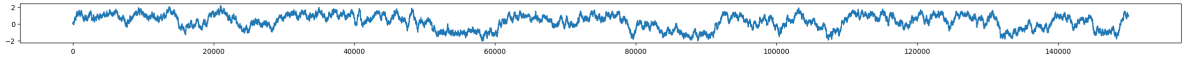


Figure 2: First 150000 data according to time

2.2 $D^1(x)$

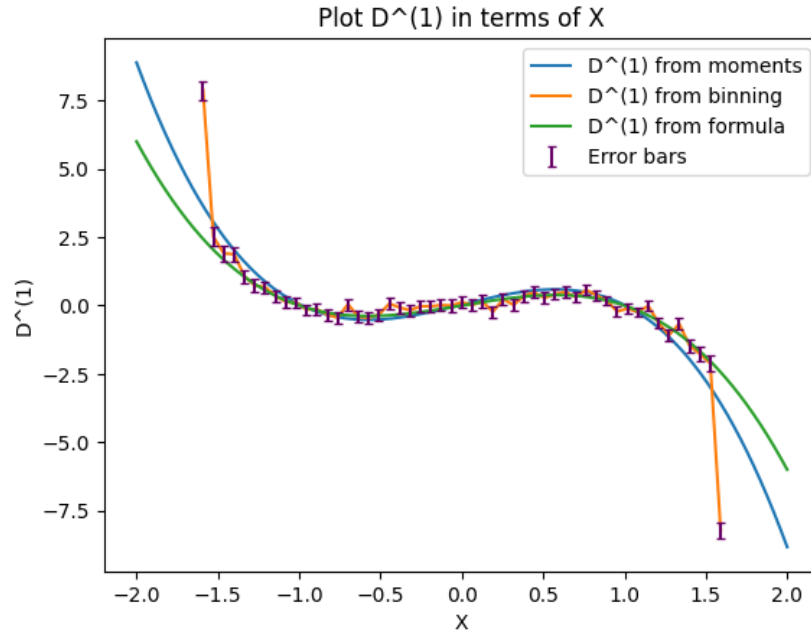


Figure 3: $D^{(1)}(x)$

2.3 $D(x)$

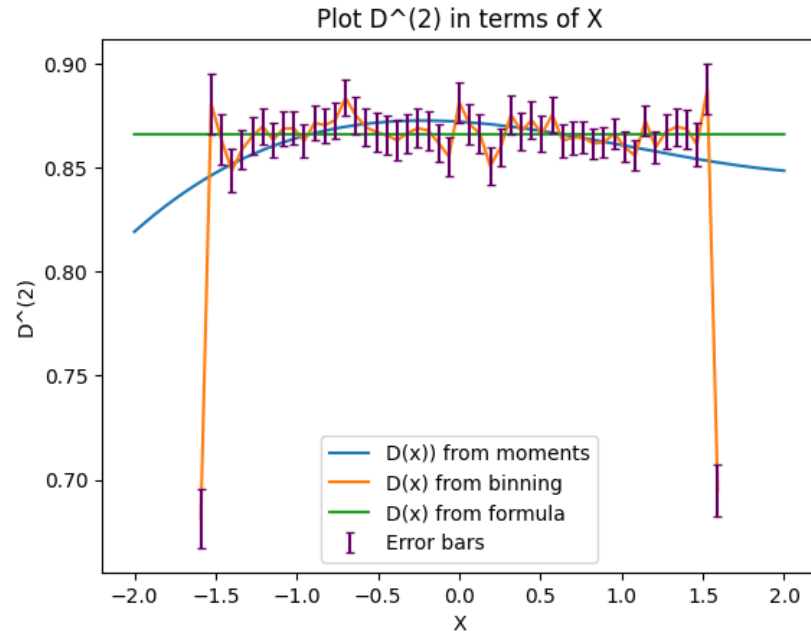


Figure 4: $D(x)$

2.4 $D^4(x)$

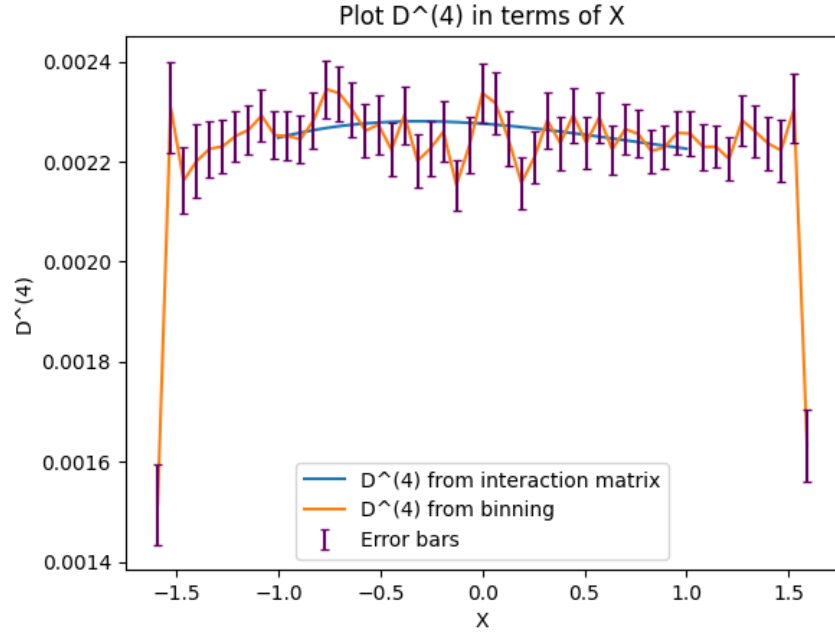


Figure 5: $D^{(4)}(x)$

2.5 $D^6(x)$

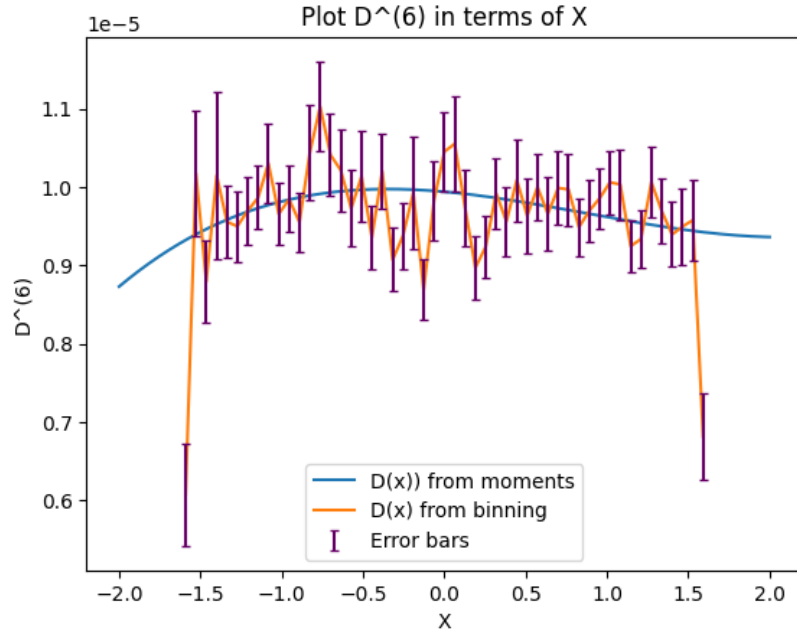


Figure 6: $D^{(6)}(x)$

2.6 ξ

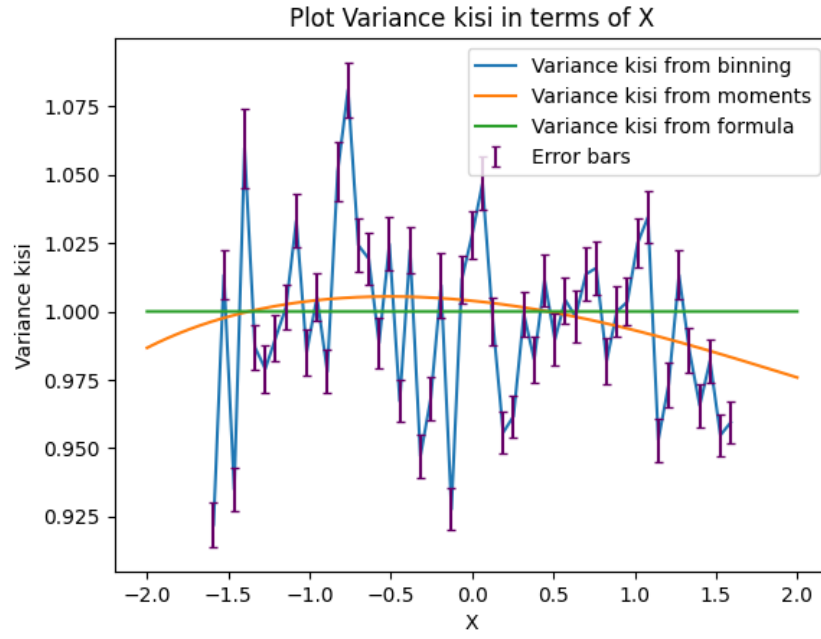


Figure 7: ξ

2.7 λ

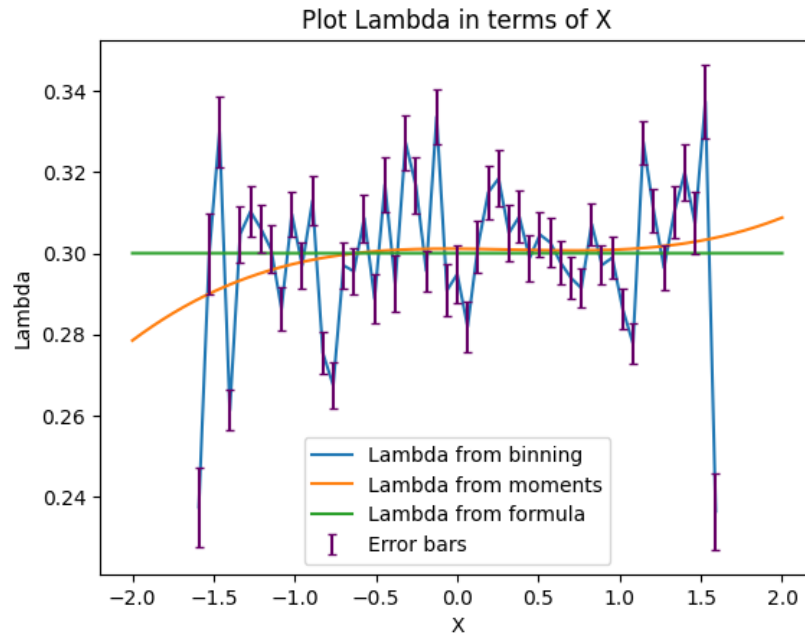


Figure 8: λ

3 Code

```
1 # Import needed libraries
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from numpy import random
6
7 #Generationg noise
8 n = 1000000
9 etha = np.random.normal(0,1,n)
10 jump_rate = random.poisson(lam=0.3, size=n)
11 kisi = np.random.normal(0,1,n)
12
13 #Generating data:
14 x=0.1
15 dt= 0.001
16 data = []
17 for i in range(n):
18     dx = (x-x**3)*dt + (((0.75)**(1/2))*dt)**(1/2)*etha[i] + kisi[i]*
19         jump_rate[i]*dt
20     x= x+dx
21     data.append(x)
22
23 print(data)
24 df = np.array(data)
25
26 plt.figure(figsize=(15,3))
27 plt.plot(df)
28 plt.show()
29 plt.figure(figsize=(30,1))
30 plt.plot(df[:150000])
31 plt.show()
32
33 vari = np.var(df)
34 df = df[df> -2*vari]
35 df = df[df< 2*vari]
36
37 # Calculating needed values
38 bins = 51
39 hist_values, bin_edges = np.histogram(df, bins=bins)
40 max = np.max(df)
41 min = np.min(df)
42
43 # Calculation diffrences
44 tool = len(df)
45 differences = []
46 binmid_list = []
47 for i in range(bins):
48     binmid = (bin_edges[i] + bin_edges[i+1]) /2
49     binmid_list.append(binmid)
50     diff = []
51     for j in range(tool):
52         if df[j]>= bin_edges[i] and df[j] < bin_edges[i+1] and j != tool-1:
53             ekhtelaf = df[j+1] - df[j]
54             diff.append(ekhtelaf)
```

```

54     differences.append(diff)
55
56 data = df
57 diffr = []
58 for i in range(len(data)-1):
59     diff = data[i+1]-data[i]
60     diffr.append(diff)
61
62 y1 = [map(lambda x, y: x * y**i, diffr, data) for i in range(4)]
63
64
65 # Calculating D^(1)
66 D1_dt = []
67 errors_dt = []
68 for j in range(bins):
69     d1 = (sum(differences[j]))/len(differences[j])
70     res = pd.Series(differences[j]).var()
71     sem = (res/len(differences[j]))**(1/2)
72     errors_dt.append(sem)
73     D1_dt.append(d1)
74
75 D1 = [i * 1000 for i in D1_dt]
76 errors1 = [i * 1000 for i in errors_dt]
77
78 tau=1
79 y = data[tau:] - data[:-tau]
80 data1 = np.ones(len(data))
81 ys = np.zeros(4)
82 for i in range(4):
83     ys[i] = np.mean(y)
84     y *= data[:-1]
85
86
87 mmnts = np.ones(7)
88 xk = np.ones(len(data))
89 for i in range(7):
90     mmnts[i] = xk.mean()
91     xk *= data
92
93 A = np.zeros((4,4))
94 for i in range(4):
95     A[i] = np.roll(mmnts, -i)[:4]
96
97 A_inv = np.linalg.inv(A)
98 phis = A_inv@ys
99 phis = (phis)/0.001
100 x = np.linspace(-2,2, 10001)
101
102 Y = x - x**3
103 plt.plot(x,np.array([phis[i]*x**i for i in range(4)]).sum(0), label='D^(1)
    from moments')
104 plt.plot(binmid_list,D1, label='D^(1) from binning')
105 plt.plot(x,Y, label='D^(1) from formula')
106 plt.errorbar(binmid_list, D1, yerr=errors1, fmt='none', color='#660066',
    capsize=2, label='Error bars')
107 plt.legend()
108 plt.xlabel("X")

```



```

109 plt.ylabel("D^(1)")
110 plt.title('Plot D^(1) in terms of X')
111 plt.show()
112
113 # Calculating D^(2)
114 D2_dt = []
115 errors_dt = []
116 for j in range(bins):
117     double_list = np.power(differences[j], 2)
118     d1 = (sum(double_list))/len(differences[j])
119     res = pd.Series(double_list).var()
120     sem = (res/len(double_list))**(1/2)
121     errors_dt.append(sem)
122     D2_dt.append(d1)
123
124 D2 = [(i) * (1000) for i in D2_dt]
125 errors2 = [i * (1000) for i in errors_dt]
126
127 tau=1
128 y = data[tau:] - data[:-tau]
129 y = y**2
130 ys2 = np.zeros(4)
131 for i in range(4):
132     ys2[i] = np.mean(y)
133     y *= data[:-1]
134
135
136 phis2 = A_inv@ys2
137 phis2 = phis2/0.001
138 x = np.linspace(-2,2, 10001)
139 Y = (0.75)**(1/2) * (x**0)
140 plt.plot(x,np.array([phis2[i]*x**i for i in range(4)]).sum(0), label='D(x)
    from moments')
141 plt.plot(binmid_list,D2, label='D(x) from binning')
142 plt.plot(x,Y, label='D(x) from formula')
143 plt.errorbar(binmid_list, D2, yerr=errors2, fmt='none', color='#660066',
    capsize=2, label='Error bars')
144 plt.legend()
145 plt.xlabel("X")
146 plt.ylabel("D^(2)")
147 plt.title('Plot D^(2) in terms of X')
148 plt.show()
149
150 # Calculating D^(4)
151 D4_dt = []
152 errors_dt = []
153 for j in range(bins):
154     power4_list = np.power(differences[j], 4)
155     d1 = (sum(power4_list))/len(differences[j])
156     res = pd.Series(power4_list).var()
157     sem = (res/len(power4_list))**(1/2)
158     errors_dt.append(sem)
159     D4_dt.append(d1)
160
161 D4 = [(i) * (1000) for i in D4_dt]
162 errors4 = [i * (1000) for i in errors_dt]
163

```

```

164 tau=1
165 y = data[tau:] - data[:-tau]
166 y = y**4
167 ys4 = np.zeros(4)
168 for i in range(4):
169     ys4[i] = np.mean(y)
170     y *= data[:-1]
171
172
173 phis4 = A_inv@ys4
174 phis4 = phis4/0.001
175 x = np.linspace(-1,1, 10001)
176 plt.plot(x,np.array([phis4[i]*x**i for i in range(4)]).sum(0), label='D^(4)
    from interaction matrix')
177 plt.plot(binmid_list,D4, label='D^(4) from binning')
178 plt.errorbar(binmid_list, D4, yerr=errors4, fmt='none', color='#660066',
    capsize=2, label='Error bars')
179 plt.legend()
180 plt.xlabel("X")
181 plt.ylabel("D^(4)")
182 plt.title('Plot D^(4) in terms of X')
183 plt.show()
184
185 totvar = np.var(sum(differences, []))
186 landa=[sum(differences[i]>totvar) / len(differences[i]) for i in range(len(
    differences))]
187
188 D6_dt = []
189 errors_dt = []
190 for j in range(bins):
191     power6_list = np.power(differences[j], 6)
192     d6 = (sum(power6_list))/len(differences[j])
193     res = pd.Series(power6_list).var()
194     sem = (res/len(power6_list))**(1/2)
195     errors_dt.append(sem)
196     D6_dt.append(d6)
197
198 D6 = [i * (1000) for i in D6_dt]
199 errors6 = [i * (1000) for i in errors_dt]
200
201 tau=1
202 y = data[tau:] - data[:-tau]
203 y = y**6
204 ys6 = np.zeros(4)
205 for i in range(4):
206     ys6[i] = np.mean(y)
207     y *= data[:-1]
208
209
210 phis6 = A_inv@ys6
211 phis6 = phis6/0.001
212 x = np.linspace(-2,2, 10001)
213
214 plt.plot(x,np.array([phis6[i]*x**i for i in range(4)]).sum(0), label='D(x)
    from moments')
215 plt.plot(binmid_list,D6, label='D(x) from binning')
216 plt.errorbar(binmid_list, D6, yerr=errors6, fmt='none', color='#660066',

```

```

    capsize=2, label='Error bars')
217 plt.legend()
218 plt.xlabel("X")
219 plt.ylabel("D^(6)")
220 plt.title('Plot D^(6) in terms of X')
221 plt.show()
222
223 varkisi_bin = np.array(D6) / (5 * np.array(D4))*(1000/0.87)
224 varkisi_mmnts = np.array([phis6[i]*x**i for i in range(4)]).sum(0)/(5*np.
    array([phis4[i]*x**i for i in range(4)]).sum(0))*(1000/0.87)
225 Y = 1 *(x**0)
226 errorskisi = np.array(errors6) / np.array(errors4)
227 plt.plot(binmid_list, varkisi_bin, label='Variance kisi from binning')
228 plt.plot(x, varkisi_mmnts, label='Variance kisi from moments')
229 plt.plot(x, Y, label='Variance kisi from formula')
230 plt.errorbar(binmid_list, varkisi_bin, yerr=errorskisi, fmt='none', color
    ='#660066', capsize=2, label='Error bars')
231 plt.legend()
232 plt.xlabel("X")
233 plt.ylabel("Variance kisi")
234 plt.title('Plot Variance kisi in terms of X')
235 plt.show()
236
237 landa_bin = np.array(D4) / (3*(varkisi_bin)**2)*(9000/22.5)
238 landa_mmnts = np.array([phis4[i]*x**i for i in range(4)]).sum(0) / (3*(
    varkisi_mmnts)**2)*(9000/22.5)
239 Y = 0.3*(x**0)
240 errorslanda = np.array(errors4) / errorskisi
241 plt.plot(binmid_list, landa_bin, label='Lambda from binning')
242 plt.plot(x, landa_mmnts, label='Lambda from moments')
243 plt.plot(x, Y, label='Lambda from formula')
244 plt.errorbar(binmid_list, landa_bin, yerr=errorslanda, fmt='none', color
    ='#660066', capsize=2, label='Error bars')
245 plt.legend()
246 plt.xlabel("X")
247 plt.ylabel("Lambda")
248 plt.title('Plot Lambda in terms of X')
249 plt.show()

```