
TWO-DIMENSIONAL KRAMERS-MOYAL COEFFICIENT

HW06

Hanie Hatami(99100614)
Stochastic processes course

Contents

1	Abstract	3
2	Results	3
2.1	Data	3
2.2	$D^1(x_1)$	6
2.3	$D^1(x_2)$	7
2.4	$D^2(x_1)$	7
2.5	$D^2(x_2)$	8
2.6	$D^2(x_1, x_2)$	8
3	Code	9

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. When dealing with two-dimensional systems, where two interdependent variables evolve over time, the two-dimensional Kramers-Moyal coefficient becomes a crucial tool for uncovering the intricate details of the stochastic dynamics.

The primary challenge in analyzing two-dimensional stochastic processes lies in accurately characterizing the joint dynamics of the two variables involved. Consider two time series, $X_1(t)$ and $X_2(t)$, which represent the evolution of two interdependent stochastic variables over time.

The goal is to derive the two-dimensional Kramers-Moyal coefficients from these time series. Consider two stochastic variables $x_1(t)$ and $x_2(t)$, satisfy the following coupled Langevin equations,

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} D_1^{(1)}(x_1, x_2) \\ D_2^{(1)}(x_1, x_2) \end{bmatrix} + \begin{bmatrix} g_{11}(x_1, x_2) & g_{12}(x_1, x_2) \\ g_{21}(x_1, x_2) & g_{22}(x_1, x_2) \end{bmatrix} \begin{bmatrix} \Gamma_1(t) \\ \Gamma_2(t) \end{bmatrix} \quad (1)$$

As an example, we integrate the following coupled Langevin equations,

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 + a\Gamma_1(t) \\ \frac{dx_2}{dt} &= 0.02x_1 + 0.03x_2 - x_1^3 - x_1^2x_2 + a\Gamma_2(t) \end{aligned} \quad (2)$$

The main objective is to estimate the first- and second-order Kramers-Moyal coefficients, $D^1(x_1)$, $D^1(x_2)$, $D^2(x_1, x_1)$, $D^2(x_2, x_2)$, and $D^2(x_1, x_2)$, from the given time series data. These coefficients provide insights into:

- **Drift Terms:** $D^1(x_1)$ and $D^1(x_2)$ represent the deterministic components of the system's evolution, indicating the average change in X_1 and X_2 per unit time.
- **Diffusion Terms:** $D^2(x_1, x_1)$ and $D^2(x_2, x_2)$ describe the variance of the fluctuations in X_1 and X_2 , respectively, while $D^2(x_1, x_2)$ captures the covariance of the fluctuations between X_1 and X_2 .

By accurately estimating these coefficients, we can develop a deeper understanding of the system's behavior, enabling better predictions and control. This analysis is particularly useful in fields where noise plays a significant role.

2 Results

2.1 Data

Our data generation assumptions are as follows:

- $x_1(0)$ and $x_2(0)$ are zero.

- Time step of dt is 0.001
- $a=0.05$
- Numbers of generated data is 5000000
- Γ follows gaussian distribution with mean 0 and var 1

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

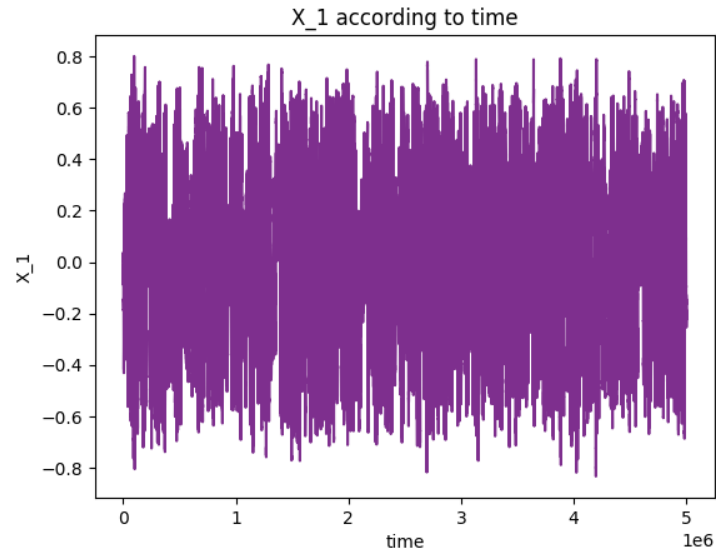


Figure 1: First time serie according to time

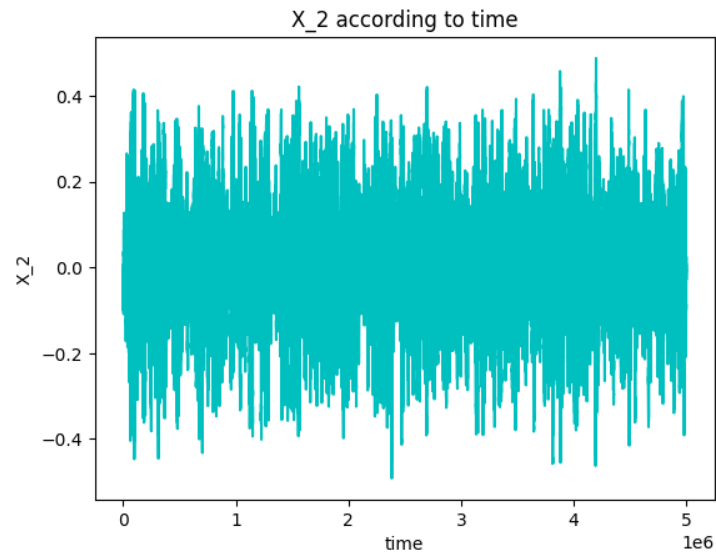


Figure 2: Second time serie according to time

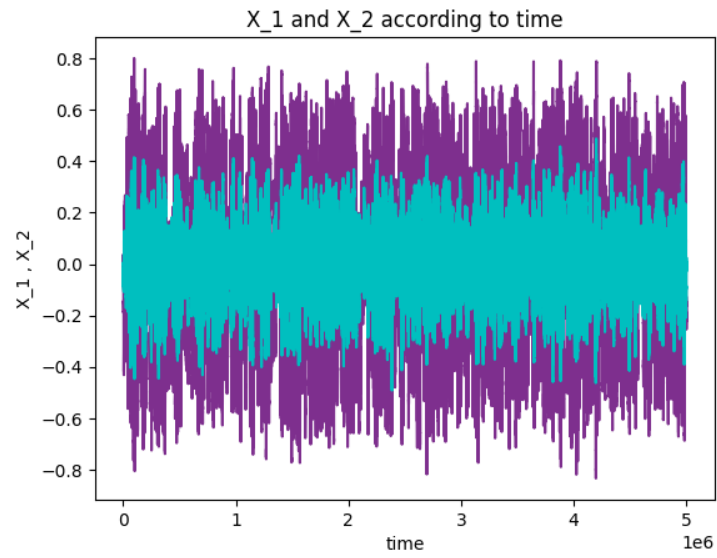


Figure 3: both time series according to time

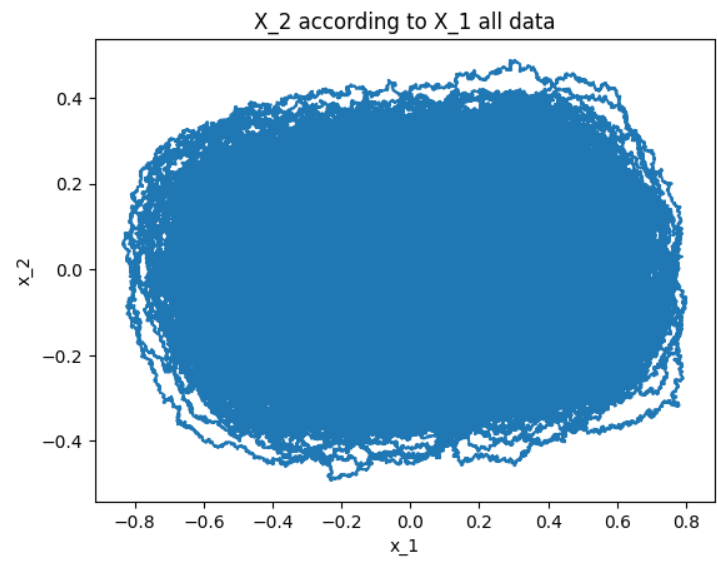


Figure 4: Second time serie according to first time serie

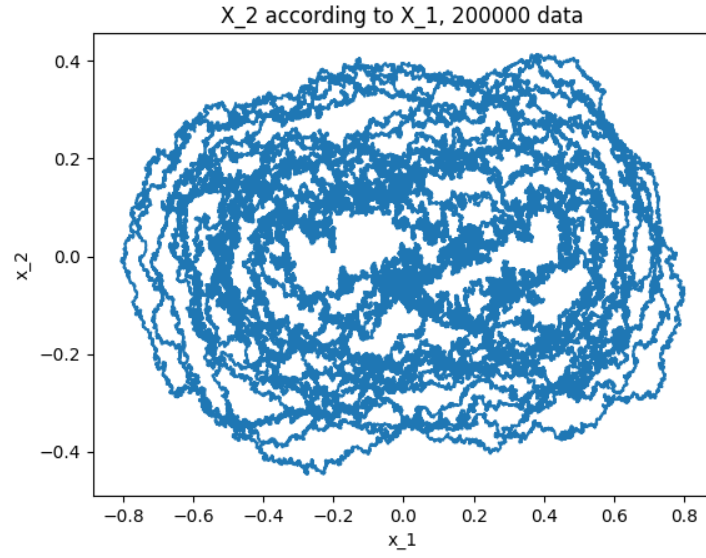


Figure 5: Second time serie according to first time serie in 200000 time step

2.2 $D^1(x_1)$

Drift for first timeserie($D_{\{x_1\}^1}$)

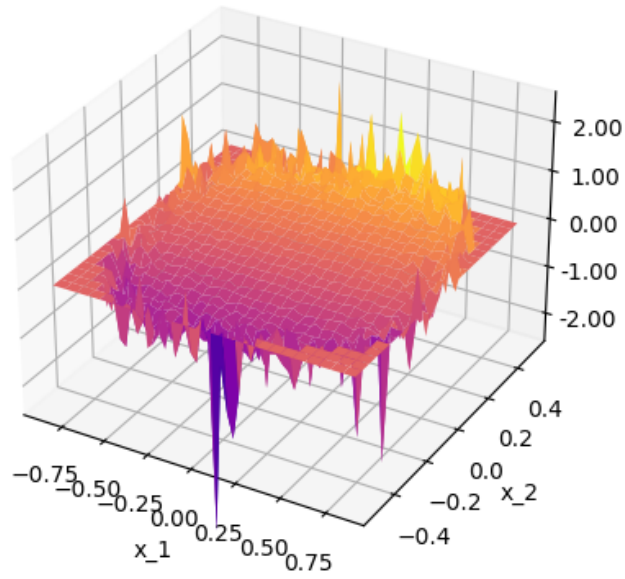


Figure 6: $D^{(1)}(x_1)$

2.3 $D^1(x_2)$

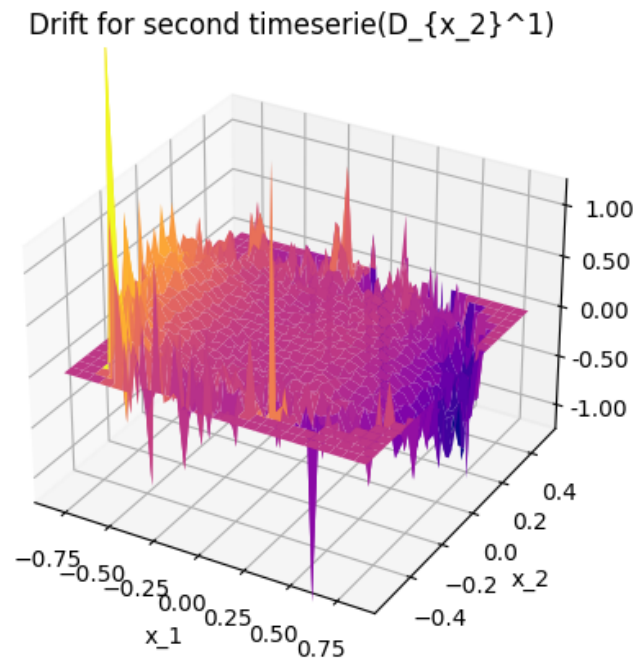


Figure 7: $D^{(1)}(x_2)$

2.4 $D^2(x_1)$

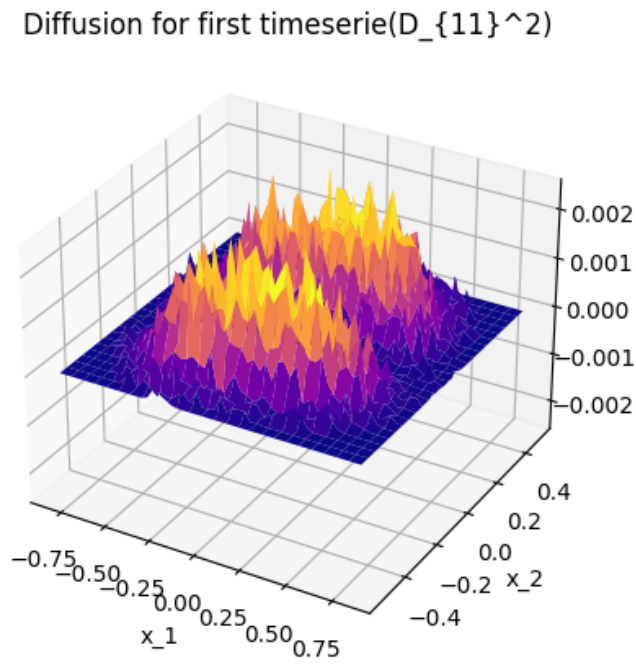


Figure 8: $D^{(2)}(x_1)$

2.5 $D^2(x_2)$

Diffusion for first timeserie(D_{22}^2)

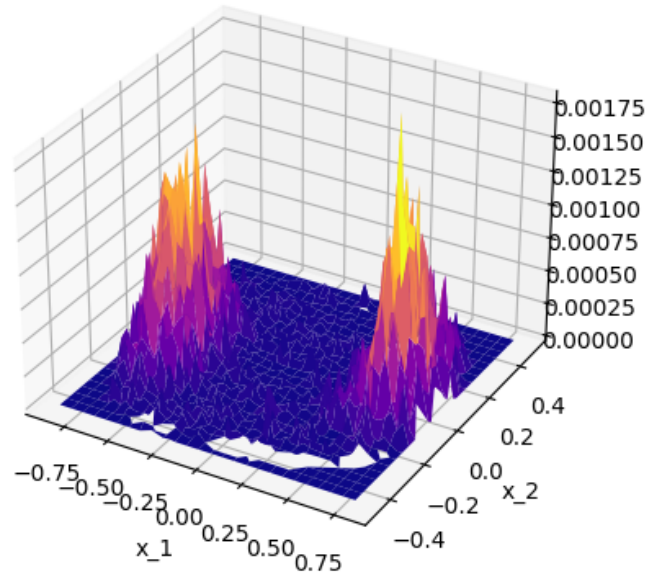


Figure 9: $D^{(2)}(x_2)$

2.6 $D^2(x_1, x_2)$

Diffusion for first timeserie(D_{12}^2)

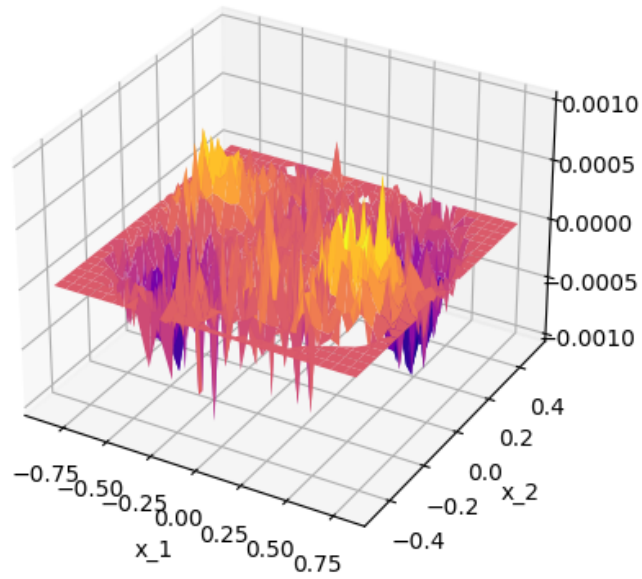


Figure 10: $D^{(2)}(x_1, x_2)$

3 Code

```
1 #Import libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 #Parameters
6 x_1=0
7 x_2=0
8 dt= 0.001
9 alpha = 0.05
10
11 #Generationing noise
12 n = 5000000
13 etha_1 = alpha*np.random.normal(0,1,n)*((dt)**(1/2))
14 etha_2 = alpha*np.random.normal(0,1,n)*((dt)**(1/2))
15
16 #Creating data
17 datax_1 = []
18 datax_2 = []
19 for i in range(n):
20     dx_1 = x_2*dt + etha_1[i]
21     x_1= x_1+dx_1
22     datax_1.append(x_1)
23     dx_2 = (0.02*x_1 + 0.03*x_2 - (x_1**3) - (x_1**2)*x_2)*dt + etha_2[i]
24     x_2 = x_2+dx_2
25     datax_2.append(x_2)
26
27 #Plotting data
28 plt.plot(datax_1,"#7E2F8E")
29 plt.xlabel("time")
30 plt.ylabel("X_1")
31 plt.title("X_1 according to time")
32 plt.show()
33 plt.plot(datax_2,"c")
34 plt.xlabel("time")
35 plt.ylabel("X_2")
36 plt.title("X_2 according to time")
37 plt.show()
38 plt.plot(datax_1,"#7E2F8E")
39 plt.plot(datax_2,"c")
40 plt.xlabel("time")
41 plt.ylabel("X_1 , X_2")
42 plt.title("X_1 and X_2 according to time")
43 plt.show()
44 plt.plot(datax_1,datax_2)
45 plt.xlabel("x_1")
46 plt.ylabel("x_2")
47 plt.title("X_2 according to X_1 all data")
48 plt.show()
49 plt.plot(datax_1[:200000],datax_2[:200000])
50 plt.xlabel("x_1")
51 plt.ylabel("x_2")
52 plt.title("X_2 according to X_1, 200000 data")
53 plt.show()
54
```

```

55 #Binning and locating data
56 bins = 51
57
58 df1 = datax_1
59 df2 = datax_2
60
61 #Hist values and bin edges
62 hist_values_1, bin_edges_1 = np.histogram(df1, bins=bins)
63 hist_values_2, bin_edges_2 = np.histogram(df2, bins=bins)
64
65 #make data between 0 and bins for binning
66 min1 = np.min(df1)
67 min2 = np.min(df2)
68
69 df1 = df1 - min1
70 df2 = df2 - min2
71
72 max1 = np.max(df1)
73 max2 = np.max(df2)
74
75 df1 = df1*(bins)/max1
76 df2 = df2*(bins)/max2
77
78 mask1 = (df1<bins)*(df2<bins)
79 df1 = df1[mask1]
80 df2 = df2[mask1]
81
82 binmid_list11 = []
83 binmid_list21 = []
84
85 #Calculating middle of the bins.
86 for i in range(bins):
87     binmid1 = (bin_edges_1[i] + bin_edges_1[i+1]) / 2
88     binmid_list11.append(binmid1)
89     binmid2 = (bin_edges_2[i] + bin_edges_2[i+1]) / 2
90     binmid_list21.append(binmid2)
91
92 #Calculating  $D_{\{x_1\}}^{-1}$ 
93 DR_1 = np.zeros((bins,bins))
94 counter_1 = np.zeros((bins,bins))
95 A_1 = np.zeros((bins,bins))
96
97 for i in range(len(df1)-1):
98     A_1[int(df1[i]//1)][int(df2[i]//1)] += (df1[i+1]-df1[i])
99     counter_1[int(df1[i]//1)][int(df2[i]//1)] += 1
100
101 for j in range(bins):
102     for k in range(bins):
103         if counter_1[j][k] != 0:
104             DR_1[j][k] = A_1[j][k]/counter_1[j][k]
105             DR_1[j][k] = DR_1[j][k] *max1 / (bins*dt)
106
107 x = np.outer(binmid_list11, np.ones(bins))
108 y = np.outer(binmid_list21, np.ones(bins)).T
109
110 #Plotting with outer
111 Z = np.array(DR_1)

```

```

112
113 fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
114 ax.plot_surface(x, y, Z, cmap='plasma')
115 ax.zaxis.set_major_formatter('{x:.02f}')
116 fig.colorbar(surf, shrink=0.5, aspect=5)
117 ax.set_zlim([-2.5, 2.5])
118 plt.xlabel("x_1")
119 plt.ylabel("x_2")
120 plt.title("Drift for first timeserie(D_{x_1}^1)")
121
122 plt.show()
123
124 #Calculating D_{x_2}^1
125 DR_2 = np.zeros((bins,bins))
126 counter_2 = np.zeros((bins,bins))
127 A_2 = np.zeros((bins,bins))
128 for i in range(len(df1)-1):
129     A_2[int(df1[i]//1)][int(df2[i]//1)] += (df2[i+1]-df2[i])
130     counter_2[int(df1[i]//1)][int(df2[i]//1)] += 1
131
132 for j in range(bins):
133     for k in range(bins):
134         if counter_2[j][k] != 0:
135             DR_2[j][k] = A_2[j][k]/counter_2[j][k]
136             DR_2[j][k] = DR_2[j][k] *max2 / (bins*dt)
137
138 #Plotting with outer
139 Z = np.array(DR_2)
140 ax = plt.axes(projection = '3d')
141
142 ax.plot_surface(x, y, Z, cmap='plasma')
143 ax.zaxis.set_major_formatter('{x:.02f}')
144 ax.set_zlim([-1.2,1.2])
145 fig.colorbar(surf, shrink=0.5, aspect=5)
146 plt.xlabel("x_1")
147 plt.ylabel("x_2")
148 plt.title("Drift for second timeserie(D_{x_2}^1)")
149 plt.show()
150
151 #Calculating D_{11}^2
152 A_2_11 = [[0 for _ in range(bins)] for _ in range(bins)]
153 Dif_11 = [[0 for _ in range(bins)] for _ in range(bins)]
154
155 for j in range(bins):
156     for k in range(bins):
157         A_2_11[j][k] = A_1[j][k]*A_1[j][k]
158         if counter_1[j][k] != 0:
159             Dif_11[j][k] = A_2_11[j][k]/counter_1[j][k]
160             Dif_11[j][k] = Dif_11[j][k]*max1*max1 / (bins*bins*dt)
161 Dif_11 = np.multiply(Dif_11, 0.01)
162 #Plotting with outer
163 Z = np.array(Dif_11)
164 ax = plt.axes(projection = '3d')
165
166 ax.plot_surface(x, y, Z, cmap='plasma')
167 ax.set_zlim([-0.0025,0.0025])
168 fig.colorbar(surf, shrink=0.5, aspect=5)

```

```

169
170 plt.xlabel("x_1")
171 plt.ylabel("x_2")
172 plt.title("Diffusion for first timeserie(D_{11}^2)")
173
174 plt.show()
175
176 A_2_22 = [[0 for _ in range(bins)] for _ in range(bins)]
177 Dif_22 = [[0 for _ in range(bins)] for _ in range(bins)]
178
179 for j in range(bins):
180     for k in range(bins):
181         A_2_22[j][k] = A_2[j][k]*A_2[j][k]
182         if counter_1[j][k] != 0:
183             Dif_22[j][k] = A_2_22[j][k]/counter_2[j][k]
184             Dif_22[j][k] = Dif_22[j][k] *max2*max2 / (bins*bins*dt)
185 Dif_22 = np.multiply(Dif_22, 0.01)
186 #Plotting with outer
187 Z = np.array(Dif_22)
188 ax = plt.axes(projection = '3d')
189
190 ax.plot_surface(x, y, Z,cmap='plasma')
191 ax.set_zlim([0,0.0018])
192 fig.colorbar(surf, shrink=0.5, aspect=5)
193
194 plt.xlabel("x_1")
195 plt.ylabel("x_2")
196 plt.title("Diffusion for first timeserie(D_{22}^2)")
197
198 plt.show()
199
200 A_2_12 = [[0 for _ in range(bins)] for _ in range(bins)]
201 Dif_12 = [[0 for _ in range(bins)] for _ in range(bins)]
202
203 for j in range(bins):
204     for k in range(bins):
205         A_2_12[j][k] = A_1[j][k]*A_2[j][k]
206         if counter_1[j][k] != 0:
207             Dif_12[j][k] = A_2_12[j][k]/counter_2[j][k]
208             Dif_12[j][k] = Dif_12[j][k] *max1*max2 / (bins*bins*dt)
209 Dif_12 = np.multiply(Dif_12, 0.01)
210 #Plotting with outer
211 Z = np.array(Dif_12)
212 ax = plt.axes(projection = '3d')
213
214 ax.plot_surface(x, y, Z,cmap='plasma')
215 fig.colorbar(surf, shrink=0.5, aspect=5)
216 plt.xlabel("x_1")
217 plt.ylabel("x_2")
218 plt.title("Diffusion for first timeserie(D_{12}^2)")
219
220 plt.show()

```