18.4 Micro-structure noise

Faraz Khaleghi

In this exercise we are gonna check micro-structure properties and perform a self-consistency check on our process

# Introduction

We have two process: $x\left(t\right)=−x\left(t\right)dt + \sqrt{2}dw$ $y\left(t\right)=x\left(t\right)+σζ\left(t\right)$ we have x(t=0)=0, dt=0.01 for $10^{6}$ data points, $σ$=0.2 and $ζ$(t) is uncorrelated zero mean, unit variance Gaussian noise. Our time series amplitude will be between 4 and -4. So I have one hundred bins over the range of -3 to 3. $α$



# (a) Kramers-Moyal coefficients for y and estimate $σ$

I have estimated the Kramers-Moyal coefficients the normal way and with the equation, $K^{\left(2\right)}\left(y, τ\right)=K^{\left(2\right)}\left(x, τ\right) + γ\_{2}\left(y\right)$ and also, $γ\_{2}\left(y\right) = 2<η^{2}> =2σ^{2}$ we can estimate $σ$ as seen in the figure below:





# (b) Drift and diffusion coefficients of x(t)

We will estimate these coefficients the classic way again. Here are some figures confirming the precision of the estimated coefficients:





# $K^{\left(2\right)}\left(y, τ\right)$ plot

In the figure below we see the dependence of the first Kramers-Moyal coefficient for process y(t) to $τ$, confirming the existence of micro-structure noise:



# Code

The python code file is attached to the zip file and here we have a summary of it:

dt = 0.01
final\_t = 10000
t\_steps = int(final\_t/dt)
t = np.arange(t\_steps)\*dt
x = np.zeros(t\_steps)
y = np.zeros(t\_steps)
sigma = 0.2

for i in tqdm(range(0, t\_steps-1)):
 x[i+1] = x[i] - x[i]\*dt + np.sqrt(2)\*np.random.normal(loc=0.0, scale=1.0)\*np.sqrt(dt)
 y[i] = x[i] + sigma\*np.random.normal(loc=0.0, scale=1.0)

def K(n, x, start, end, binsize, \*args, \*\*kwargs):
 ''' returns Kramers-Moyal coefficients for different bins
 default for tau is dt
 '''
 binrange = [int(start/binsize), int(end/binsize)]
 bincount = (binrange[1] - binrange[0]) + 1
 global bins
 bins = np.linspace(start, end, num=bincount)
 K = []

 if 'tau' in kwargs:
 tau = kwargs['tau']
 elif len(args) > 0:
 tau = args[0]
 else:
 tau = dt

 for i in range(binrange[0], binrange[1]+1):
 Bin = []
 for j in range(x.size-int(tau/dt)):
 if ((x[j] < float(i)\*binsize + binsize/2) & (x[j] > float(i)\*binsize - binsize/2)):
 Bin.append(x[j+int(tau/dt)] - x[j])
 Bin = np.array(Bin)
 K.append(np.mean(Bin\*\*n))
 K = np.array(K)
 return K

K\_2x = K(n=2, x=x, start=-3, end=3, binsize=0.05, tau=0.01)
K\_2y = K(n=2, x=y, start=-3, end=3, binsize=0.05, tau=0.01)
K\_1x = K(n=1, x=x, start=-3, end=3, binsize=0.05, tau=0.01)