Python1_Template

February 18, 2021

1 SC3011TN - Stochastische Signaalanalyse - 2020/2021

2 Group details

Group number: ...

Student 1:

```
Name: Jeroen Sangers

Student number: 4645197
```

Student 2:

```
Name: Reinaart van Loon

Student number: 4914058
```

Date of completion: February 18th

2.1 Question 1

If the random process $x(k)$ in (1.1) is wide-sense stationary (WSS), what can you say about its mean and variance? And what if the process is not WSS?

2.1.1 Answer 1

If WSS:

Its mean is constant and finite : $m_x (k) = m_x < \infty$ $The variance is finite and for a complex stochastic process non-negative for : 1. $c_x(0) < \infty$ 2. $ r_x(0) \geq 0$

If not WSS:

Its mean depends on the index: so $m_x(k)$ not always $m_x(k+l)$ Variance can be infinite

[1]:
```
# Import packages
import numpy as np
import matplotlib.pyplot as plt
import scipy as sci
from tqdm import tqdm
from scipy.optimize import least_squares
```

```
# Initiate random seed for reproducible results
np.random.seed(2021)

# Constants (Table 2.1 of the assignment text)
N = 1e5                          # Number of samples
dt = 1e-3                        # Sampling period
m = 0.01                         # Mass of the black hole
a = 3*np.pi/16                   # Plummer radius
b_max = a                        # Maximum impact parameter
G = 1                            # Normalized gravitational constant
M = 1                            # Normalized total cluster mass
V_0 = np.sqrt(G*M/(2*a))         # Relative velocity with interacting stars
Lambda = b_max*V_0**2/(G*m)      # Coulomb logarithm factor
c = G*M*m/a**3                   # Spring constant
gamma = 128*np.sqrt(2)/(7*np.pi) * np.sqrt(G/(M*a**3)) * m**2 * np.log(Lambda) #␣
 ↪Friction coefficient
```

As the number of interacting stars N_{star}, fill in the first 5 digits of the student number of Student 1

[2]:
```
N_star = 46451   # First 5 digits of student number of Student 1; number of␣
 ↪interacting stars
m_star = M/N_star # Mass of an individual interacting star
R = 4*G*M*m_star*gamma/(9*a) # Noise factor
```

2.2 Question 2

Let $x(k)$ be the discrete-time representation of the position of the black hole, with $t = k\Delta t$. Derive the discrete-time dynamics of $x(k)$. That is, find the parameters $\beta_1, \beta_2, \beta_3$ in the following second order difference equation:

$$x(k) + \beta_1 \cdot x(k-1) + \beta_2 \cdot x(k-2) = \beta_3 \cdot \tilde{w}(k)$$

Use Euler's backward approximation to approximate the derivative operator $\frac{d(.)}{dt}$ and second order derivative operator $\frac{d^2(.)}{d^2t}$ in (1.1). Subsequently replace the white noise signal $w(t)$ by a discrete white noise sequence as outlined in Section 1.3.

2.2.1 Provide:

1. Analytical expressions for the parameters $\beta_1, \beta_2, \beta_3$
2. Their numerical values, by making use of the data given in Table 2.1

2.2.2 Answer 2

$\beta_1 = -\frac{2m+\gamma\Delta t}{m+\gamma\Delta t+c\Delta t^2}$

$\beta_2 = \frac{m}{m+\gamma\Delta t+c\Delta t^2}$

$\beta\_3 = \frac{\sqrt{R}\Delta t^{\frac{3}{2}}}{m+\gamma\Delta t+c\Delta t^2}$

[3]:
```
b1 = -(2*m + gamma*dt)/(m+gamma*dt+c*dt**2)
b2 = m/(m+gamma*dt+c*dt**2)
b3 = (np.sqrt(R)*dt**(1.5))/(m+gamma*dt+c*dt**2)
```

2.3 Question 3

Analyze the stochastic discrete-time dynamics in the z-domain. Specifically, answer the following questions: 1. Using the z-transform, determine the transfer function $H(z)$ from $\tilde{w}$ to x of the discrete-time dynamical system described by difference equation (2.1). 2. Determine the poles of $H(z)$. 3. Is this system (BIBO) stable? Use Definition 2.5 or Lemma 2.6 on page 24 of the reader.

2.3.1 Answer 3

1. $ H(z) = $\beta_3 \frac{}{1+\beta_1 z^{-1}+\beta_2 z^{-2}}$$p_1=0,99963...+0.00218...j$$p_2=0,99963...-0.00218...j$
2. We are looking at a causal system (only past and present values are used in difference function 2.1) and both poles are inside the unit circle, so R.O.C. includes the unit circle. Therefore, the system is stable and therefore BIBO stable.

[4]:
```
### Calculation cell
import cmath
z_1 = 1j+1
z_2 = 1j+1

z_1 = (-b1+cmath.sqrt(b1**2-4*b2))/2
z_2 = (-b1-cmath.sqrt(b1**2-4*b2))/2

r = np.sqrt(np.real(z_2)**2 + np.imag(z_2)**2)
```

2.4 Question 4

Simulate a single realization of the trajectory of the black hole. That is, simulate the difference equation (2.1) for $k = 3, \cdots, N$ with $N = 5000$ using the initial conditions $x(1) = x(2) = 0$ $. Hereby you should generate discrete white noise samples using the **Numpy** command `np.random.normal(size=N)`

2.4.1 Provide:

1. Python code to generate N samples of $x(k)$
2. Plot a realization of one sequence $x(k)$
3. Explain the results. Do the results agree with your analysis in Question 3?

2.4.2 Answer 4

Yes, it seems stable since the displacement doesn't diverge. Also, it seems to keep oscilalting around zero, as if there is a restoring force.

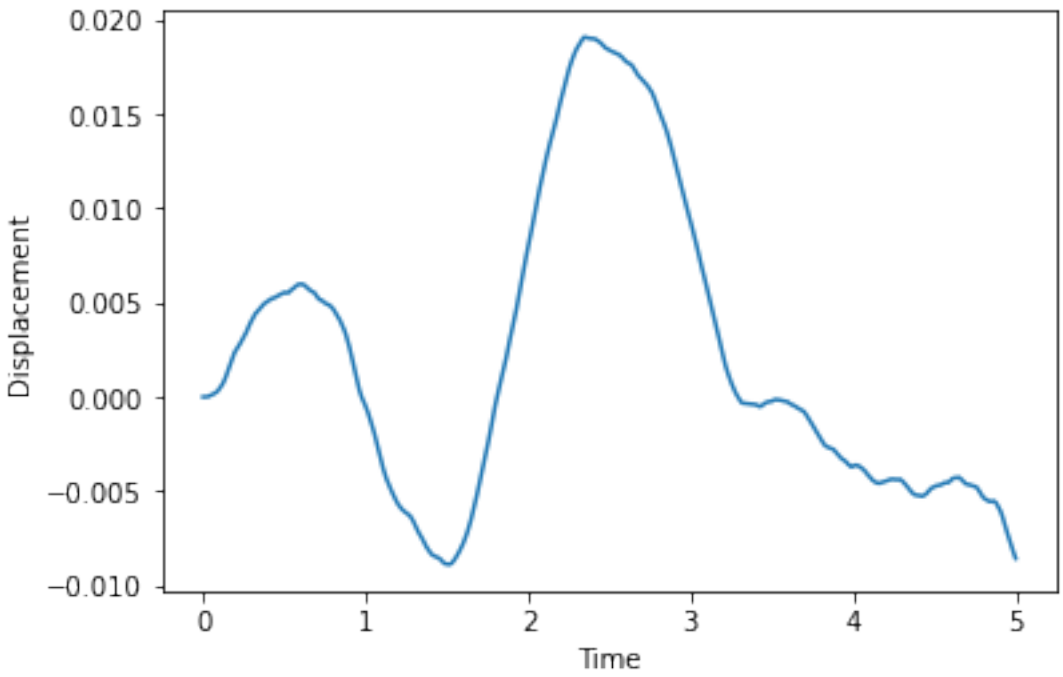
[5]:
```
N = 5000

x = np.empty(N)
x[0]=0
x[1]=0

w = np.random.normal(size=N)
```

Simulate the difference equation:

[6]:
```
for k in range(N-2):
    x[k+2] = b3*w[k+2]-b2*x[k]-b1*x[k+1]
```

Plot the results:

[7]:
```
plt.figure(1)
plt.plot(np.arange(N)*dt,x)
plt.xlabel('Time')
plt.ylabel('Displacement')
plt.show()
```



2.5 Question 5

In Question 4, you generated one realization of the trajectory of the black hole. To formalize this, let us denote the realization generated in Question 4 as $x(k, \lambda)$ for $\lambda = 1$ and the corresponding white noise sequence as $\tilde{w}(k, 1)$.

Now, simulate multiple realizations of the trajectory of the black hole. Specifically, generate L realizations $x(k,\lambda)$ for $\lambda = 1, \cdots, L$, with each realization generated for a different realization of the discrete time white noise sequence $\tilde{w}(k,\lambda)$.

NB. You may overwrite the realization generated in Question 4.

2.5.1 Provide:

1. Python script used to generate L realizations $x(k,\lambda)$ sequences
2. Plot of all L realizations for $L = 50$
3. Explain the results. Do the results agree with your analysis in Question 3?

2.5.2 Answer 5

Again it seems stable since the cluster of realizations is around the horizontal axis with a displacement of 0. Therefore, the system doesn't seem to diverge.

Also, there seems to be a restoring force that stimulates oscillation.

[8]:
```
L = 50
x = np.zeros((N,L))
w = np.random.normal(size=(N,L))
```

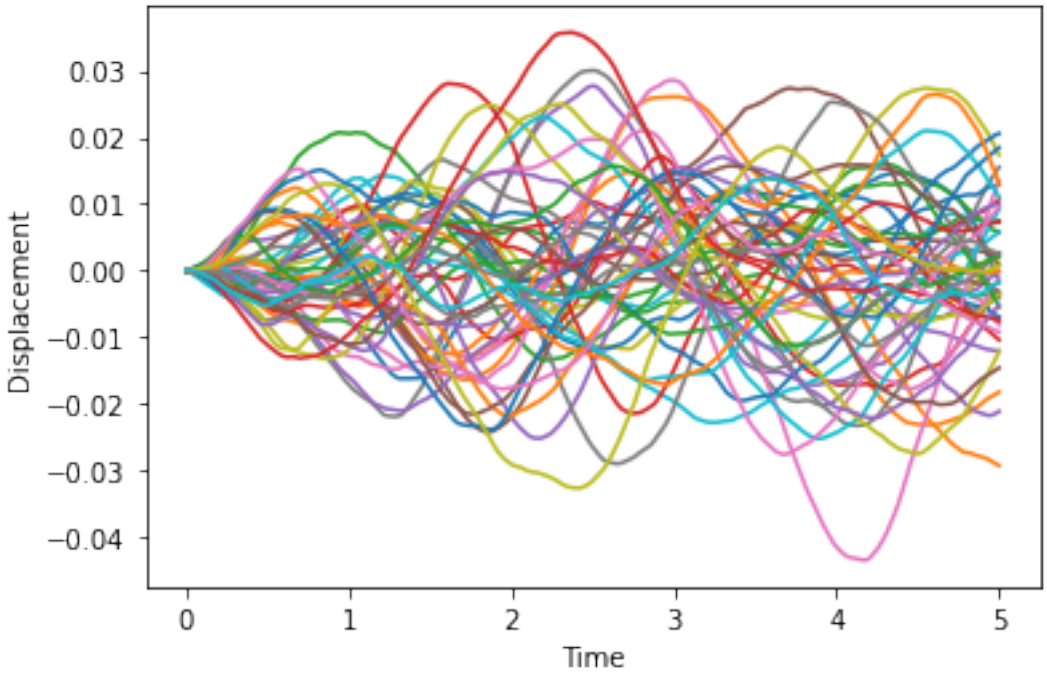
Perform L realizations of the difference equation

[9]:
```
for k in range(N-2):
    x[k+2,:] = b3*w[k+2,:]-b2*x[k,:]-b1*x[k+1,:]
```

Plot the results

[10]:
```
plt.figure(2)
plt.plot(np.arange(N)*dt,x)
plt.xlabel('Time')
plt.ylabel('Displacement')
plt.show
```

[10]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



2.6 Question 6

NB: Make sure your computer has enough memory / try small L values first!

Now, simulate even more realizations of the trajectory of the black hole.

Increase L, as defined in Question 5, to respectively $L = 100,\ 500,\ 2500$. For each value of L, obtain all realizations at time steps $k = 10^3$, $k = 10^4$ and $k = 10^5$. That is, find $\{x(10^3, \lambda)\}_{\lambda=1}^{L}$, $\{x(10^4, \lambda)\}_{\lambda=1}^{L}$, $\{x(10^5, \lambda)\}_{\lambda=1}^{L}$.

Using this data, generate histograms with **python** command `matplotlib.pyplot.hist()`, for each combination of time $k = 10^3, 10^4, 10^5$ and amount of realizations $L = 100,\ 500,\ 2500$. Your answer should thus consist of 9 histograms. The number of bins in each histogram should be equal to $\sqrt{L}$.

2.6.1 Provide:

1. One figure containing the 9 histograms. Make sure the figure has no overlapping text!
2. Comment on your results.

2.6.2 Answer 6

The further we go in time (graphs further to the RHS), the variance becomes larger. This is seen by a more spreaded histogram.

The more realizations we use (graphs further down), the histograms converge to a smoother curve.

[11]:
```
Ls =  np.asarray([100,500,2500]) # Array containing realization instances (i.e. 100, 500, 2500)
Ns =  np.asarray([1e3,1e4,1e5])   # Array containing time instances (i.e. 1e3, 1e4, 1e5)
```

Take L the maximum number of simulations we need to compare (We can take subsets for the lower values of L). Recall that N is the maximum number of time steps we need to simulate.

[12]:
```
L = Ls[-1]
N = Ns[-1]
x = np.zeros((int(N),L))
w = np.random.normal(size=(int(N),L))
```

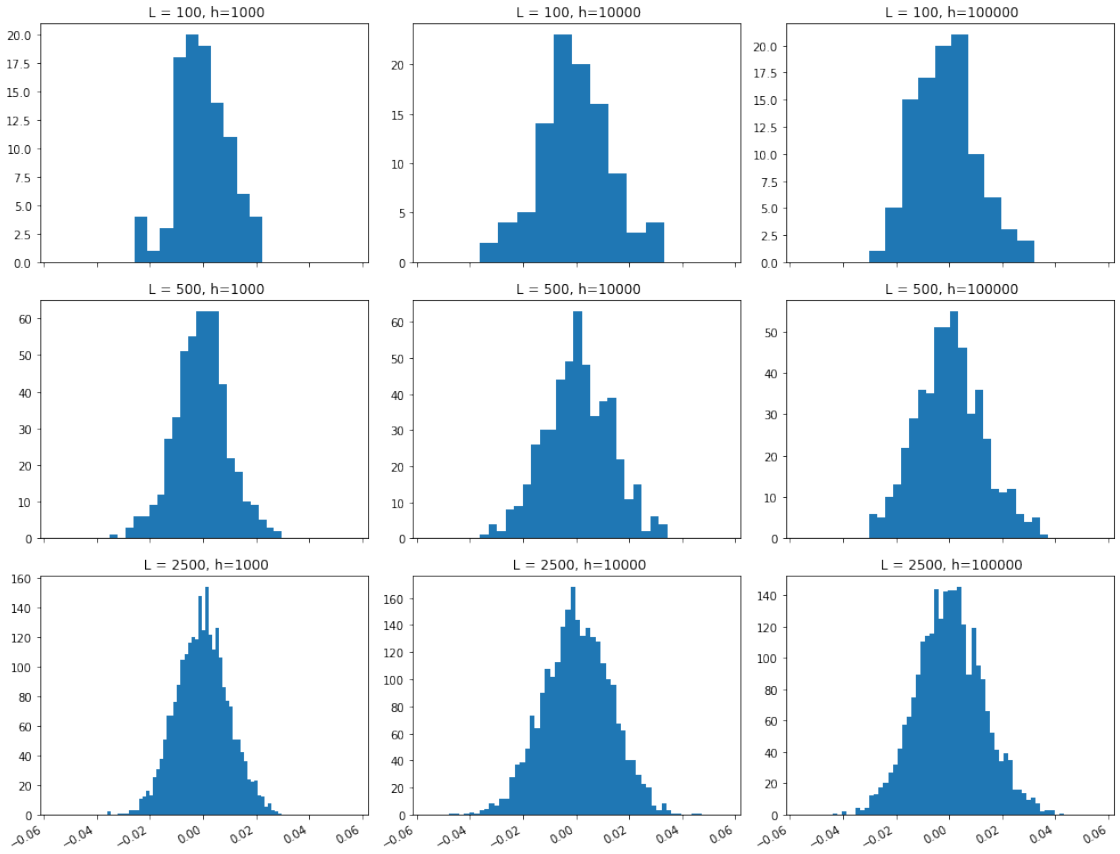
Simulate the difference equation L times. Depending on your hardware and code, this step could take 10-15 min.

NB. You can also vectorize the below computations instead of using a loop and use numpy operations to significantly speed up computations on arrays.

[13]:
```
for k in range(int(N)-2):
    x[k+2,:] = b3*w[k+2,:]-b2*x[k,:]-b1*x[k+1,:]
```

Take different subsets of the data for each combination of k and L and plot the results as histograms:

[14]:
```
fig, axs = plt.subplots(3, 3, figsize = [13.8, 10.6], tight_layout = 3.0)
fig.autofmt_xdate()

for i in range(len(Ls)):
    L = Ls[i]
    for j in range(len(Ns)):
        h = Ns[j]
        axs[i, j].hist(x[int(h)-1,0:L], bins=int(np.sqrt(L)))
        axs[i, j].set_title(' L = %d, h=%d' % (L,h))
        axs[i, j].set_xlim([np.min(x), np.max(x)])
```



2.7 Question 7

Fit a Gaussian function, given by $f(\alpha) = \kappa e^{-\frac{\alpha^2}{2\sigma^2}}$, to the 9 histograms Question 6. For that purpose, denote the center of bin i of the histogram by α_i and the corresponding height of the histogram as $h(\alpha_i)$. Then solve the following least squares optimization problem using the **scipy** function `scipy.optimize.least_squares()`

$$\hat{\kappa},\ \hat{\sigma} = arg\ min_{\kappa,\sigma} \sum_{i=1}^{P} \left(h(\alpha_i) - \kappa e^{-\frac{\alpha_i^2}{2\sigma^2}} \right)^2$$

Hint: You can use the numpy function `numpy.std` to get an estimate of the standard deviation of the random samples $\{x(k,\lambda)\}$. Together with the number of elements in the largest bin, you can use this to derive initial estimates of κ and σ in the Gaussian function.

2.7.1 Provide:

1. Python script reading the random samples and the samples $h(\alpha_i)$ of the histograms
2. 9 plots of the histograms and the Gaussian fits in one figure. Make sure the figure has no overlapping text!

3. Comment on your results.

2.7.2 Answer 7

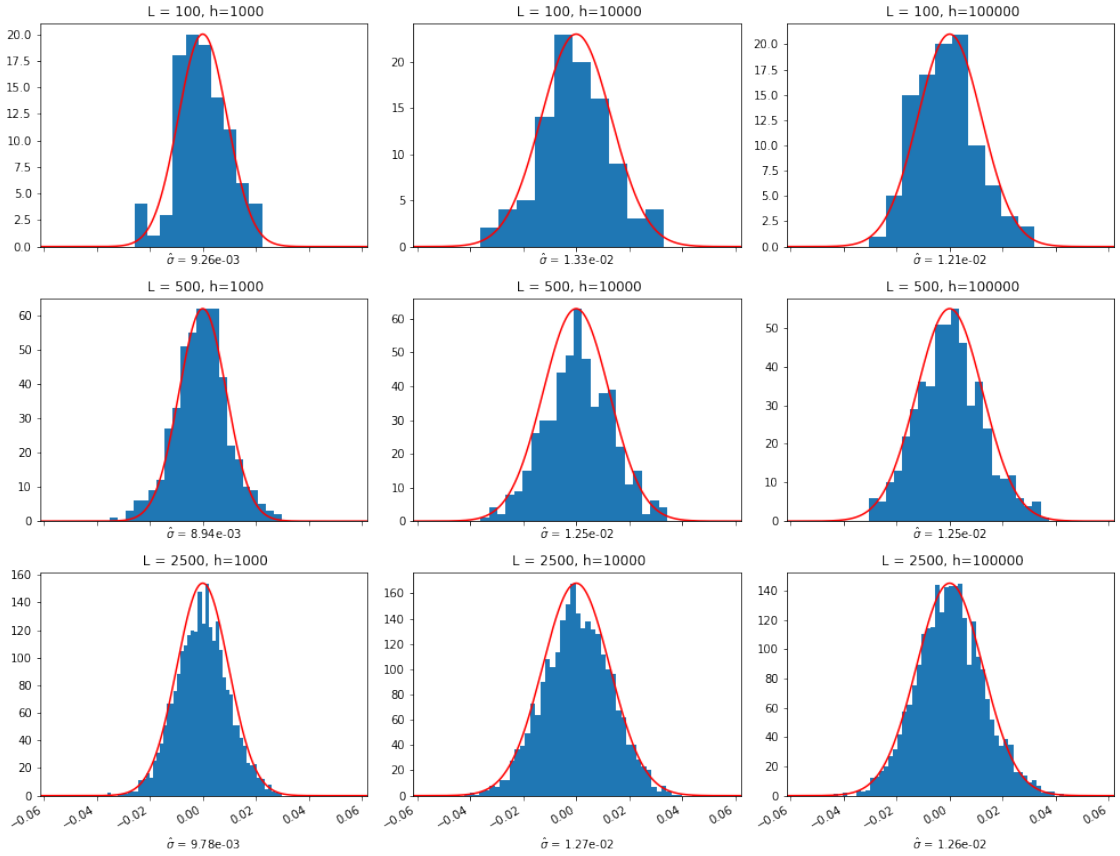
The histograms converge to gaussian shapes for more realizations.

The left-bottom graph is closer to the best gaussian fit than the right-bottom graph.

[15]:
```
# define functions
def model(x, alpha):
    '''
    Define the Gaussian function

    Inputs:
        x: array of fitting parameters (i.e. kappa, sigma)
        alpha: coordinate to describe the center of a bin of a histogram
    '''
    gaussian = x[0]*np.exp(-alpha**2/(2*x[1]**2))
    return gaussian

def error(x, alpha, h):
    '''
    Define the difference between the simulated data and the Gaussian function.
    That is, the difference between the center of each bin of the histogram and
    the corresponding coordinate of the Gaussian density function.

    Inputs:
        x: array of fitting parameters (i.e. kappa, sigma)
        alpha: coordinate to describe the center of a bin of a histogram
        h: height of the histogram at coordinate alpha
    '''
    difference = h-model(x,alpha)
    return difference

def jac(x, alpha, h):
    '''
    Define Jacobian, i.e. the partial derivatives of the difference between the␣
 ↪simulated data
    and the Gaussian function, with respect to the parameters contained in x.

    Inputs:
        x: array of fitting parameters (i.e. kappa, sigma)
        alpha: coordinate to describe the center of a bin of a histogram
        h: height of the histogram at coordinate alpha
    '''
    J = np.empty((alpha.size, x.size))
    J[:,1] = np.exp(-alpha**2/(2*x[1]**2))
    J[:,0] = alpha**2*x[0]/(x[1]**3)*np.exp(-alpha**2/(2*x[1]**2))
```

```
    return J

# Make figure that shows histograms
fit_parameters = np.zeros([2, len(Ls), len(Ns)])

for i in range(len(Ls)):
    L = Ls[i]

    for j in range(len(Ns)):
        h = Ns[j]

        # Repeat hist command to get the data, the command gives bin edges
        # But you need the centers
        [counts, edges] = np.histogram(x[int(h)-1,0:L], bins=int(np.sqrt(L)))
        centers = (edges[1:]+edges[:-1])/2

        # Calculate initial estimates of sigma and kappa
        sigma0 = np.std(x[int(h-1),:L])
        kappa0 = np.amax(counts)
        x0 = np.array([kappa0, sigma0])

        # Fit the Gaussian through the histogram data and save into array
        p_opt = least_squares(error,
                              x0,
                              jac = jac,
                              args = (centers, counts));
        fit_parameters[:,i,j] = p_opt.x

        #Plot the Gaussian on top of the histogram
        xg = np.linspace(np.min(x), np.max(x), 1000)
        yg = model(p_opt.x,xg)

        #Add the Gaussian fits to the histograms
        axs[i, j].plot(xg,yg,'r')
        axs[i, j].set_xlabel('$\hat{\sigma}$ = %.2e' % p_opt.x[1])

fig
```

[15]:



3 Question 8

Summarize the results Question 7 in the following table.

The results of Gaussian fits for the random samples $\{x(k,\lambda)\}$ for $k = 10^3, 10^4, 10^5$ and $L = 100, 500, 2500$:

Your answer should contain 3 tables (for different values of k) in the style of the above table template.

3.0.1 Answer 8

$k = 10^3$

	L	$\hat{\kappa}$	$\hat{\sigma}$
50	20	$9.26 \cdot 10^{-3}$	
500	62	$8.94 \cdot 10^{-3}$	
5000	154	$9.78 \cdot 10^{-3}$	

$k = 10^4$

	L	$\hat{\kappa}$	$\hat{\sigma}$
50	23	$1.33 \cdot 10^{-2}$	
500	63	$1.25 \cdot 10^{-2}$	
5000	168	$1.27 \cdot 10^{-2}$	

$k = 10^5$

	L	$\hat{\kappa}$	$\hat{\sigma}$
50	21	$1.21 \cdot 10^{-2}$	
500	55	$1.25 \cdot 10^{-2}$	
5000	145	$1.26 \cdot 10^{-2}$	

3.1 Question 9

Analyze how the standard deviation of the realizations and Gaussian fits is related to time and to the amount of realizations.

3.1.1 Provide:

1. Plot $\hat{\sigma}$ as a function of k, for $k = \{0, 1, ..., 10^5 - 1\}$ (i.e. $N = 10^5$ time steps).
2. How does $\hat{\sigma}$ depend on k?
3. How does the Gaussian fit change with respect to k and L?

3.1.2 Answer 9

Below we plotted the variance $\hat{\sigma}$ for every timestep k and multiple amounts of realisations L.

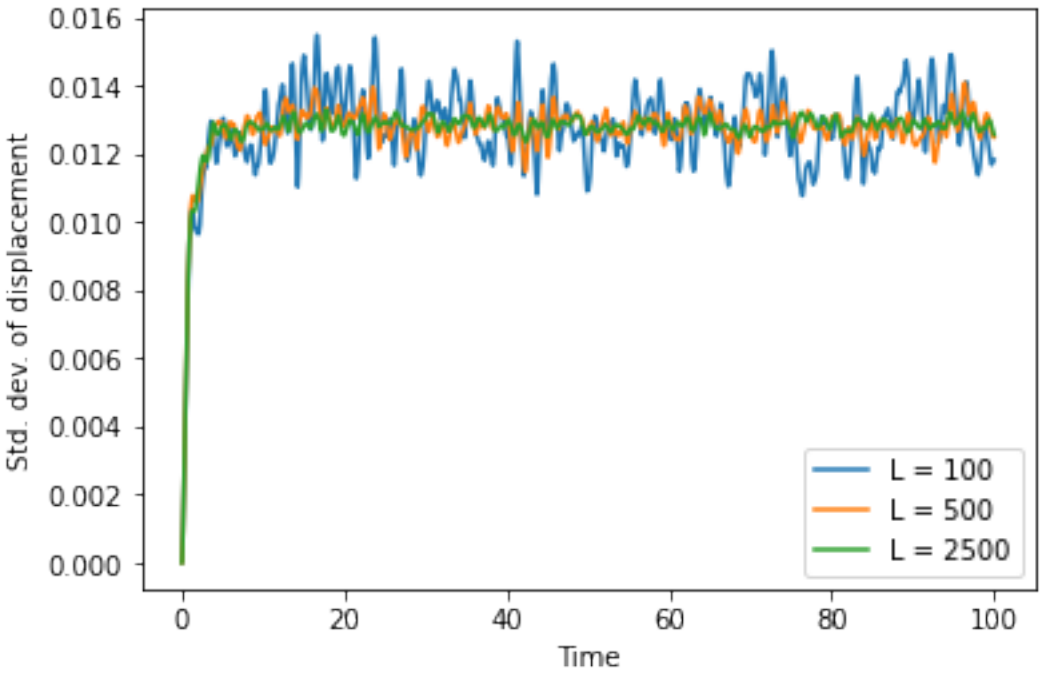
$\hat{\sigma}$ starts of at zero which is to be expected since the first values are set to zero, but after quickly rises to a deviation and stays approximately constant. Thus after a short amount of time the displacements do resemble a Gaussian distribution, which matches what we see in the histograms above.

For different amounts of realistation we see that "spikeyness" of the graph changes, for more realisations the standard deviation of the standard deviation of the displacement becomes smaller. THis is why the histograms at the top, whith the smaller L-values, deviate more from an ideal Gaussian distribution

[16]:
```
N = 10**5

t = np.linspace(0, dt*N, N)

print(x.shape)


sigma_x = [np.std(x[:,:Ls[0]-1], axis=1), np.std(x[:,:Ls[1]-1], axis=1), np.
 ↪std(x[:,:Ls[2]-1], axis=1)]
```

```
realisation_average = np.average(x, axis=1)



difference = x - (realisation_average*np.ones((1,N))).T

std_dev_sq = np.average(difference**2, axis = 1)



plt.figure(6)
plt.plot(t,sigma_x[0], label='L = 100')
plt.plot(t,sigma_x[1], label='L = 500')
plt.plot(t,sigma_x[2], label='L = 2500')
plt.legend()

plt.xlabel('Time')
plt.ylabel('Std. dev. of displacement')
plt.show()
```

```
(100000, 2500)
```



3.2 Question 10

Analyze the autocorrelation of the process.

3.2.1 Provide:

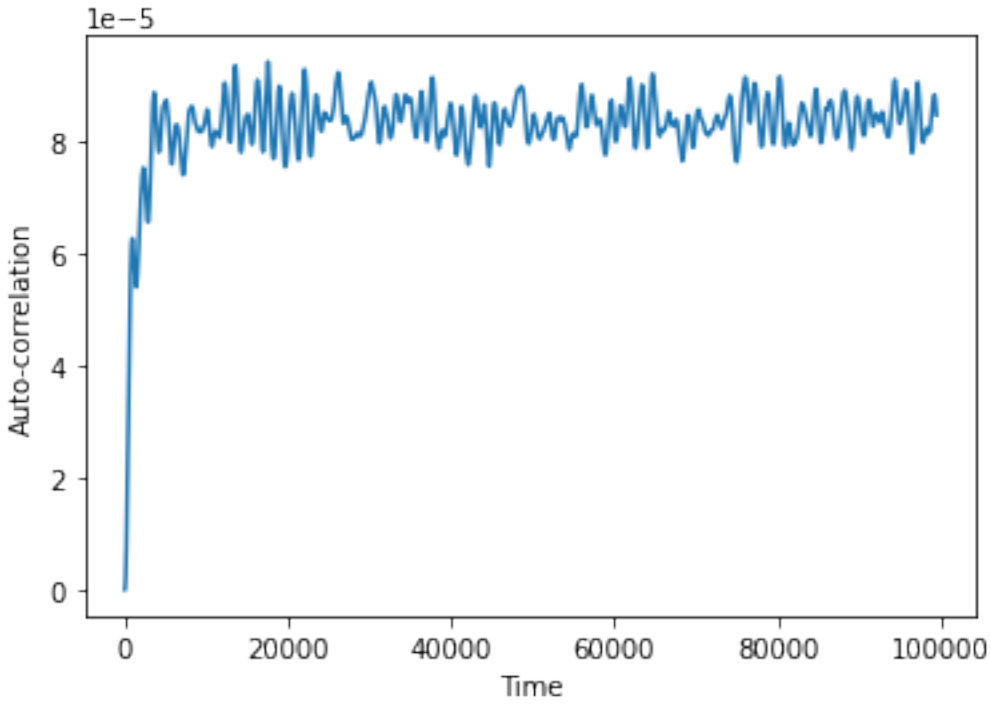
1. What is the relationship between the standard deviation $\hat{\sigma}(k)$, autocorrelation $r_x(k,k)$ and auto-covariance $c_x(k,k)$?
2. Now, calculate the autocorrelation $r_x(k,k+500)$ for $k = \{0,1,2,\cdots,10^5-501\}$ using $L = 2500$ realizations. Plot the results.
3. How does $r_x(k,k+500)$ depend on k?

3.2.2 Answer 10

The following equation was derived: $ (k)^2 = c_x(k,k) = r_x(k,k) + m_x $\cdot m_x\blacksquare * $and since we think the mean of our signal is 0 they would all be equal. Thus $\hat{\sigma}(k)^2 = c_x(k,k) = r_x(k,k)$

We see that the autocorrelation very rapidly converges to a virtually constant value (it still has minor deviations) . Therefore, it seems the autocorrelation does not really depend on the time (the value k).

[17]:
```
L = 2500
h2 = 500
h = 10**5 - h2
start = 0

Rx = np.zeros((L,h))

for i in range(0,L):
    Rx[i] = x[0:h,i]*np.conj(x[h2:h+h2,i])

Rx_mean = np.mean(Rx, axis=0)


plt.figure(11)
plt.plot(Rx_mean)
plt.xlabel('Time')
plt.ylabel('Auto-correlation')
plt.show()
```



3.3 Question 11

Using the results obtained so far, what can you conclude regarding wide sense stationarity (WSS) of the random process?

3.3.1 Answer 11

Since our data-set is finite we can not be certain but from the graphs we draw the following conclusions: 1. Our mean is virtually zero since our deviation squared is equal to the autocorrelation $\hat{\sigma}(k)^2 = r_x(k,k)$. 2. As can be seen in the graph relating to question 10, our autocorrelation is (virtually) constant after a time. 3. In question 9 we see that the variance is (virtually) constant and finite after a time.

So we think that our signal $x(n)$ is WSS.