SC42150_PythonAssignment1_Template

October 16, 2020

1 SC42150 - Python assignment 1

1.1 Simulating a Stochastic Damped Harmonic Oscillator

This is the template for for the first Phyton assignment given in the course SC42150. Use this template to answer the question.

2 Team members

Name (student number): Weronika Dziarnowska (4551117)

Name (student number): Daniel Varela (5321263)

2.0.1 How to use the template

As you will see there are two types of cells in this Jupyter Notebook, **Python** cells indicated with `In [ ]:` and **Markdown** cells that have no marking to the left of the cell.

1. The **Python** works in the usual way. Enter your code and press `Shift Enter` to run the cell. The output will be displayed below the cell. Variables and functions will be accessible by the other cells after you have run the code.
2. The **Markdown** cells use the Markdown type setting language. This is a very simple type setting system. Check the Cheatsheet for a overview of the commends. Furthermore, **Markdown** works with **LaTeX** math, `$ $` for inline equations and `$$ $$` for centered equations. press `Shift Enter` to compile the text.

After you have answered the questions and are finished with the assignment you can export the **Jupyter Notebook** as a PDF. Go to the top left, click on `File` then `Download As` and Choose **PDF**. If this feature is not supported by your system, you can also choose **LaTeX**. Then you Download a **LaTeX** project. You can then use your preferred **LaTeX** compiler to create the **PDF**. After generation you PDF report, please double check if all answers/plots are correctly displayed.

More information about Jupyter Notebooks can be easily found online.

2.0.2 Useful python packages

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

2.1 Question 1

If the random process $x(k)$ is wide-sense stationary (WSS), what can you say about its mean and variance? And what if the process is not WSS? **1 point**

provide: * *an answer with explanation*

2.1.1 Answer 1

According to Definition 5.6, (some of) the properties of wide sense stationary processes are the following: * $m_x(k) = m_x < \infty$ This property states that the mean of the random process is time invariant, which means that it is a function of the time difference $t_2 - t_1$, and not a function of time t.

Furthermore, because the mean is constant, we can infer that the variance is also not dependent on time. We have:

- $\sigma^2 = E[(x(n) - E[x(n)^2])] = E[x(n)^2] - E[x(n)]^2 = r_x(0) - m_x^2$
- Finally, since $\sigma^2 = c_x(0) < \infty$, we can also say that the variance of a WSS s.p. is finite.

2.2 Question 2

Approximate the derivative operator $\frac{d(\cdot)}{dt}$ and the second order derivative operator $\frac{d^2(\cdot)}{dt^2}$. Subsequently replace the white noise signal $w(t)$ by a discrete white noise sequence as out lined in the theory section. Let $x(k)$ be an approximation of $x(t)$ for $t = k\Delta t$, then we ask you to write your result into the following second order difference equation form

$$x(k) + b_1 x(k-1) + b_2 x(k-2) = b_3 \tilde{w}(k).$$

Provide:

- *derivation for the second order difference equation* **2 points**
- *analytic expression for b_1, b_2, and b_3* **1 point**
- *numerical values for b_1, b_2, and b_3* **1 point**

2.2.1 Answer 2

The equation to discretize is the following: $\frac{dv(t)}{dt} = -\omega^2 x(t) - \gamma v(t) + \beta w(t)$

The first step is to discretize $v(t)$ and $\frac{dv(t)}{dt}$.

$v(t) = \frac{dx(t)}{dt} \approx \frac{x(k)-x(k-1)}{\Delta t} = v(k)$

$\frac{dv(t)}{dt} \approx \frac{v(k)-v(k-1)}{\Delta t} \approx \frac{\frac{x(k)-x(k-1)}{\Delta t} - \frac{x(k-1)-x(k-2)}{\Delta t}}{\Delta t} = \frac{x(k)-2x(k-1)+x(k-2)}{(\Delta t)^2}$

$w(t)$ is discretized as: $w(t) = \frac{w(k)}{\sqrt{\Delta t}}$

This is done to keep the properties of $w(t)$: $E[w(t)] = 0$ and $\sigma^2_{w(t)} = \frac{1}{\Delta t}$.

The discretized noise $w(k)$ is assumed to have $E[w(k)] = 0$ and $\sigma^2_{w(k)} = 1$, because of the python function used to generate it.

Adding the term $\frac{1}{\sqrt{\Delta t}}$ to our discretized noise samples actually keeps has the effect of preserving the properties of the time continuous noise signal:

$E\left[\frac{w(k)}{\sqrt{\Delta t}}\right] = \frac{E[w(k)]}{\sqrt{\Delta t}} = 0$

$\sigma^2_{\frac{w(k)}{\sqrt{\Delta t}}} = E\left[\left(\frac{w(k)}{\sqrt{\Delta t}} - E\left[\frac{w(k)}{\sqrt{\Delta t}}\right]\right)^2\right] = E\left[\left(\frac{w(k)}{\sqrt{\Delta t}}\right)^2\right] = \frac{\sigma^2_{w(k)}}{\Delta t} = \frac{1}{\Delta t}$

Having discretized all terms in the given equation, we now proceed with the substitution:

$\frac{dv(t)}{dt} = -\omega^2 x(t) - \gamma v(t) + \beta w(t) \Leftrightarrow \frac{x(k)-2x(k-1)+x(k-2)}{(\Delta t)^2} = -\omega^2 x(k) - \gamma\frac{x(k)-x(k-1)}{\Delta t} + \beta\frac{w(k)}{\sqrt{\Delta t}}$

$(\frac{1}{(\Delta t)^2} + \omega^2 + \frac{\gamma}{\Delta t})x(k) - (\frac{2}{(\Delta t)^2} + \frac{\gamma}{\delta t})x(k-1) + \frac{x(k-2)}{(\Delta t)^2} = \beta\frac{w(k)}{\sqrt{\Delta t}} \Leftrightarrow$

$(1 + \gamma\Delta t + (\omega\Delta t)^2)x(k) - (2 + \gamma\Delta t)x(k-1) + x(k-2) = (\Delta t)^2\beta\frac{w(k)}{\sqrt{\Delta t}} \Leftrightarrow$

$x(k) - \frac{2+\gamma\Delta t}{1+\gamma\Delta t+(\omega\Delta t)^2}x(k-1) + \frac{1}{1+\gamma\Delta t+(\omega\Delta t)^2}x(k-2) = \frac{(\Delta t)^2\beta}{1+\gamma\Delta t+(\omega\Delta t)^2\sqrt{\Delta t}}w(k)$

From the expression above, the b coefficients are clearly:

$$b_1 = -\frac{2+\gamma\Delta t}{1+\gamma\Delta t+(\omega\Delta t)^2} \qquad b_2 = \frac{1}{1+\gamma\Delta t+(\omega\Delta t)^2} \qquad b_3 = \frac{(\Delta t)^2\beta}{(1+\gamma\Delta t+(\omega\Delta t)^2)\sqrt{\Delta t}}$$

The computation of the values of b_1, b_2 and b_3 follows:

[3]:
```
gamma = 1
Delta_t = pow(10,-3)
omega = 3
beta = 0.5

den = (1 + gamma * Delta_t + pow((omega * Delta_t),2))
b1 = -(2 + gamma * Delta_t) / den
b2 = 1 / den
b3 = (pow(Delta_t , 2) * beta) / (den * math.sqrt(Delta_t))

print('The numerical values of the parameters are:')
print('b1 = ' + str(b1))
print('b2 = ' + str(b2))
print('b3 = ' + str(b3))
```

```
The numerical values of the parameters are:
b1 = -1.9989830261266386
```

```
b2 = 0.9989920170547919
b3 = 1.5795450691094586e-05
```

2.3 Question 3

Determine the Z-transform of the dynamical system described by the difference equation from the previous question. Also determine the poles, both analytically and numerically. Is this system (BIBO) stable?

Provide:

- *The Z-transform of the dynamical system.* $\frac{1}{2}$ **point**
- *The poles analytically.* $\frac{1}{2}$ **point**
- *The poles numerically.* $\frac{1}{2}$ **point**
- *Is the system is BIBO stable? (explain why)* $\frac{1}{2}$ **point**

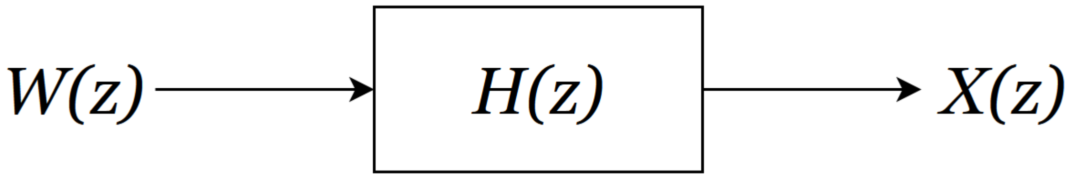
2.3.1 Answer 3

$\mathcal{Z}(x(k) + b_1 x(k-1) + b_2 x(k-2)) = \mathcal{Z}(b_3 w(k))$

The time shift property of the $\mathcal{Z}$-transform states: $\mathcal{Z}(x(k-\alpha)) = z^{-\alpha}X(z)$ for $\alpha \in \mathbb{Z}$. Using this property, we obtain:

$X(z) + b_1 z^{-1}X(z) + b_2 z^{-2}X(z) = b_3 W(z)$

We can now obtain the transfer function of the system, $\mathcal{H}(z)$. Knowing our input is the white noise signal, $W(z)$ and our output will be the position, $X(z)$, we can draw the basic block diagram of our process as:



Here we have $W(z)\mathcal{H}(z) = X(z) \Rightarrow \mathcal{H}(z) = \frac{X(z)}{W(z)}$. Therefore, in our case:

$$\frac{X(z)}{W(z)} = \frac{b_3}{1 + b_1 z^{-1} + b_2 z^{-2}} \Rightarrow \mathcal{H}(z) = \frac{b_3}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

Replacing the constants b_1, b_2 and b_3 by the expressions found in the previous question, we obtain:

$$\mathcal{H}(z) = \frac{b_3}{1 + b_1 z^{-1} + b_2 z^{-2}} \Leftrightarrow \mathcal{H}(z) = \frac{(\Delta t)^2 \beta}{(1 + \gamma\Delta t + (\omega\Delta t)^2) - (2 + \gamma\Delta t)z^{-1} + z^{-2}}$$

The poles can be found by calculating the roots of the denominator of $\mathcal{H}(z)$.

$$z^2 + b1 * z + b2 = 0 \Leftrightarrow z = \frac{-b_1 \pm \sqrt{b_1^2 - 4 \times 1 \times b_2}}{2 \times 1}$$

Replacing b_1 and b_2 by their numeric counterparts, we obtain:

$$z_1 = \frac{1000500}{1001009} + \frac{500\sqrt{35}}{1001009}i \qquad z_2 = \frac{1000500}{1001009} - \frac{500\sqrt{35}}{1001009}i$$

If we approximate the fractions, we get:

$$z_1 = 0.999492 + 0.002955i \qquad z_2 = 0.999492 - 0.002955i$$

[8]:
```python
coeff = [1, b1, b2]
roots = np.roots(coeff)
print('The roots of the system are:')
print('z1 = ' + str(roots[0]))
print('z2 = ' + str(roots[1]))
```

```
The roots of the system are:
z1 = (0.9994915130633193+0.002955058237727406j)
z2 = (0.9994915130633193-0.002955058237727406j)
```

The transfer function can be written in the following form

$ H(z) = b_3 1$_{(1-z_1z^{-1})\frac{1}{(1-z_2z^{-1})}}$$,

which means that it is a multiplication of two geometrical series of the form

$ 1$_{1-r=\sum_\{n=0\}^\{\infty\}r^n}$$.

A geometrical series is convergent for $ | r |_2 < 1 $. Given that both of the poles calculated above satisfy this condition (where $ | r |_2 = $\sqrt{Re(r)^2 + Im(r)^2}$$), itcanbeconcludedthatthetransferfunctionisabsolutesummable.This, accordingtoLemma2.6, isn

2.4 Question 4

Simulate the difference equation of question 2 for $k = 3, \cdots, N$ using the initial conditions $x(1) = x(2) = 0$. To do this you should generate discrete white noise samples using `numpy.random.randn()`.

Provide:

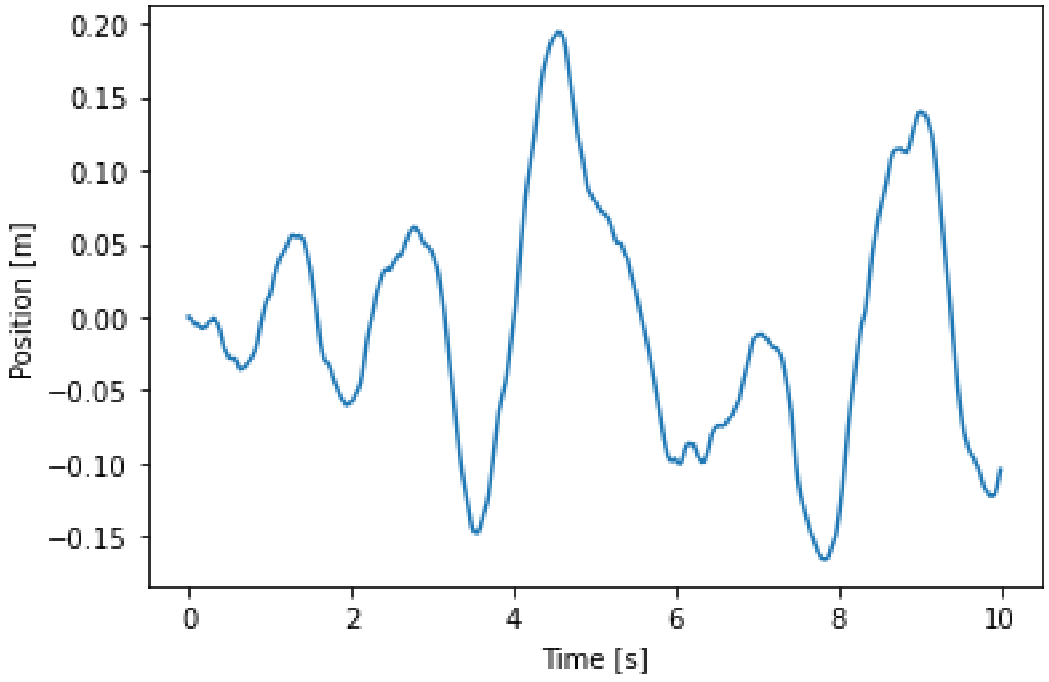
- *the script used to generate N samples of* $x(k)$. **1 point**
- *a plot of one realization of sequence* $x(k)$. **1 point**

2.4.1 Answer 4

Insert the cells with your answer below.

```
[9]: N = 10**4
     x = np.zeros(N+1)
     w = np.zeros(N+1)

     for k in range(2,N+1):
         w_k = np.random.randn()
         x_k = b3 * w_k - b1 * x[k-1] - b2 * x[k-2];
         x[k]=x_k
         w[k]=w_k

     time = np.linspace(0, N*Delta_t, N+1)
     plt.plot(time , x)
     plt.xlabel("Time [s]")
     plt.ylabel("Position [m]")
```

```
[9]: Text(0, 0.5, 'Position [m]')
```



2.5 Question 5

Let us denote one realization of the solution of the difference equation by $x(k, \lambda)$ for $\lambda = 1$ and the used white noise sequence by $\sim w(k, 1)$. Then the task is to 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 $\sim w(k, \lambda)$.

provide:

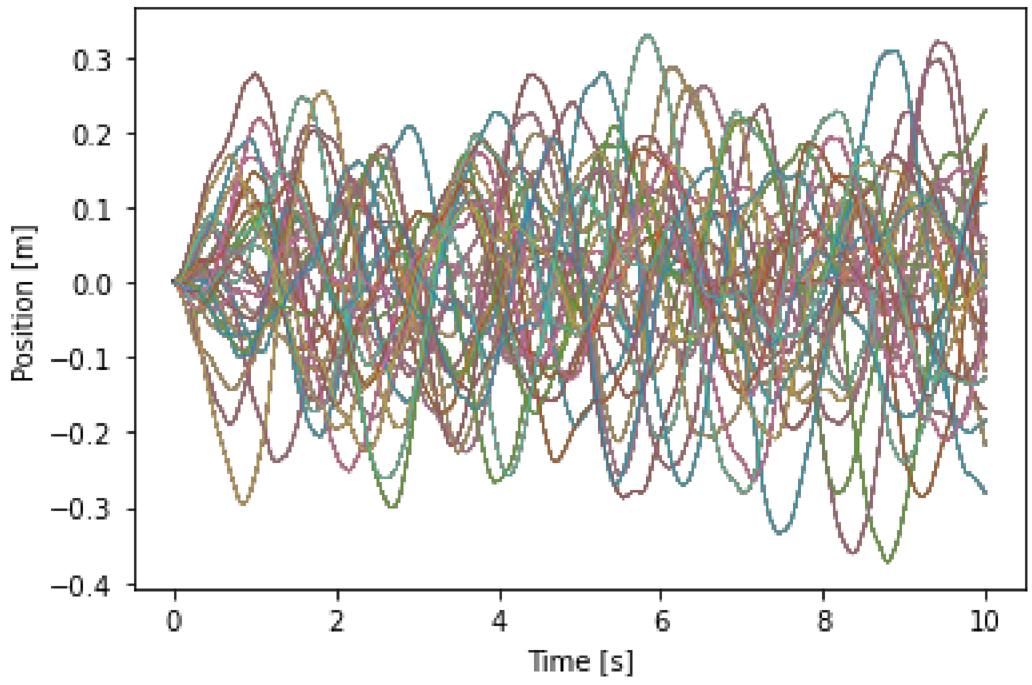
- *script use to generate L realizations of $x(k, \lambda)$.* **1 point**
- *plot of all L realizations for $L = 30$.* **1 point**

2.5.1 Answer 5

Insert the cells with your answer below.

[18]:
```
L = 30
xL = np.zeros((N+1, L+1))
wL = np.zeros((N+1, L+1))

for l in range(0,L+1):
    for k in range(2,N+1):
        wL_k = np.random.randn()
        xL_k = b3 * wL_k - b1 * xL[k-1, l] - b2 * xL[k-2, l];
        xL[k,l]=xL_k
        wL[k,l]=wL_k
```

[15]:
```
for i in range(0,L+1):
    plt.plot(time , xL, linewidth=0.5)
    plt.xlabel("Time [s]")
    plt.ylabel("Position [m]")
```



2.6 Question 6

Using the L realizations of the previous questions, estimate the mean and variance of the stochastic process $x(k)$. Here, you can use the approximation

$$E[x(k)] \sim \frac{1}{L} \sum_{\lambda=1}^{L} x(k,\lambda)$$

and a similar approximation for the variance, or use the available built-in `numpy` commands.

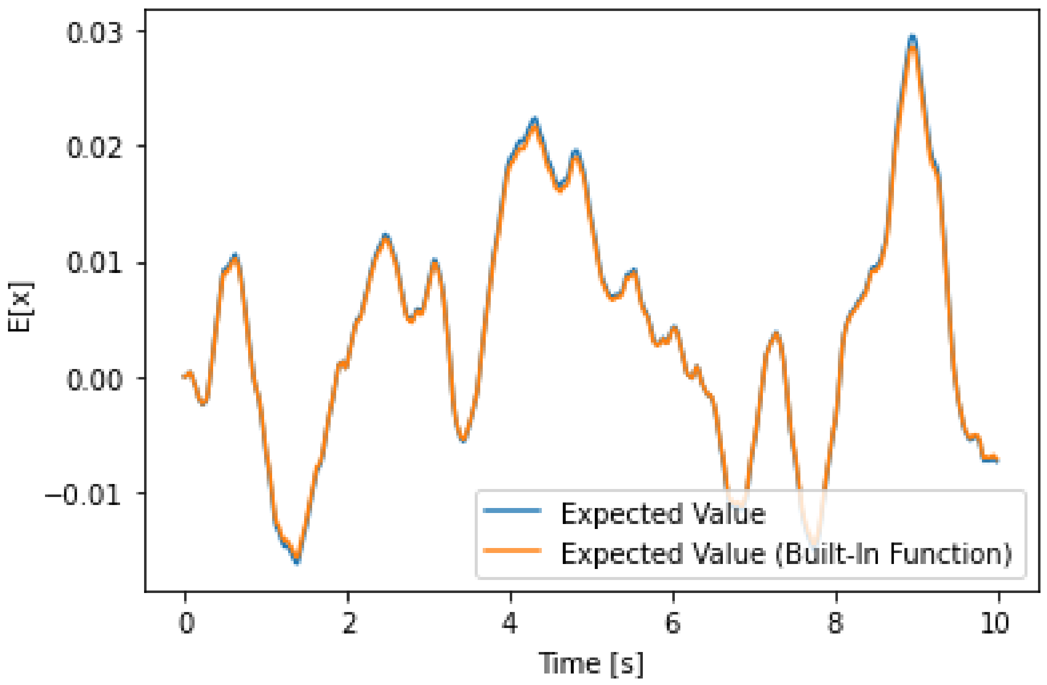
Provide:

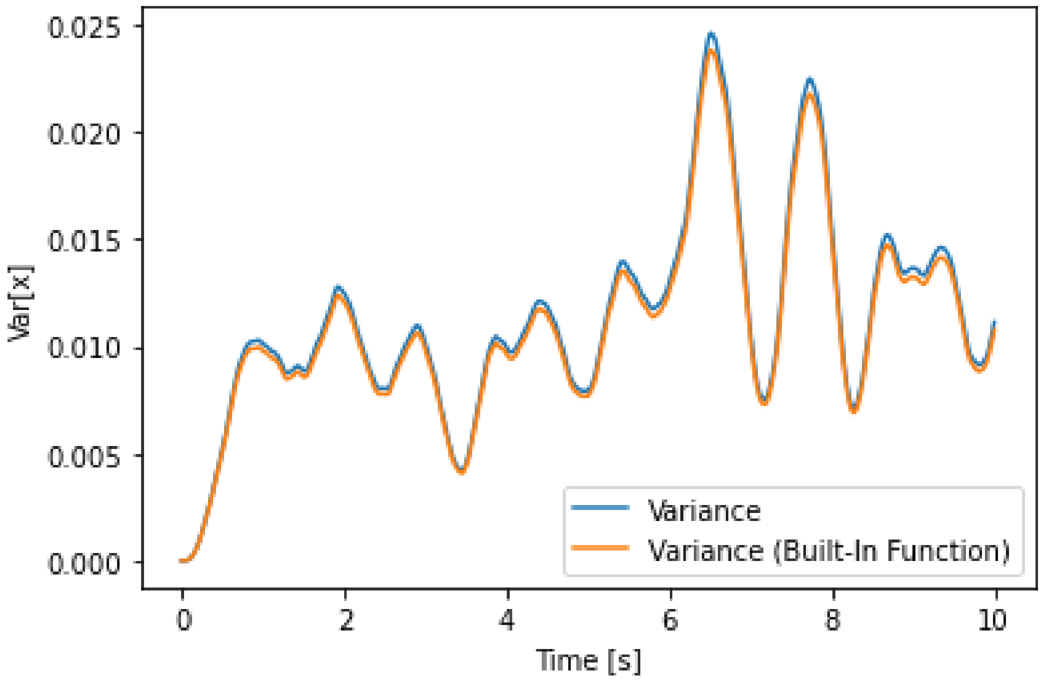
- *A script to calculate the mean and variance.* **1 point**
- *A plot of the mean and the variance over time.* **1 point**

2.6.1 Answer 6

[19]:
```
Ex = 1/L * np.sum(xL, axis=1)
Ex2 = 1/L * np.sum(xL**2, axis=1)
Var = Ex2 - Ex**2

BIVar = np.zeros(N+1)
BIAvg = np.zeros(N+1)

for i in range(0,10001):
    BIVar[i] =  np.var(xL[i,:])
    BIAvg[i] =  np.mean(xL[i,:])

plt.figure(1)
plt.plot(time, Ex, label="Expected Value")
plt.plot(time, BIAvg, label="Expected Value (Built-In Function)")
plt.legend(loc="lower right")
plt.xlabel('Time [s]')
plt.ylabel('E[x]')

plt.figure(2)
plt.plot(time, Var, label="Variance")
plt.plot(time, BIVar, label="Variance (Built-In Function)")
plt.legend(loc="lower right")
plt.xlabel('Time [s]')
plt.ylabel('Var[x]')
```

[19]:
```
Text(0, 0.5, 'Var[x]')
```


E[x]
0.03
0.02
0.01
0.00
−0.01
0
2
4
6
8
10
Time [s]
Expected Value
Expected Value (Built-In Function)


Var[x]
0.025
0.020
0.015
0.010
0.005
0.000
0
2
4
6
8
10
Time [s]
Variance
Variance (Built-In Function)

2.7 Question 7

Based on the results of the previous question, is the the stochastic process $x(k)$ WSS? Why or why not? **1 point**

2.7.1 Answer 7

Recalling the properties of the mean and variance of WSS processes, we have:

- $m(k) = m_x < \infty$
- $\sigma^2 = E[(x(n) - E[x(n)^2])] = E[x(n)^2] - E[x(n)]^2 = r_x(0) - m_x^2$
- $\sigma^2 = c_x(0) < \infty$.

Firstly, we require m(k) to be constant and bounded. If we observe the values obtained for $E[x]$, we can see that they're not always constant. However:

```
[11]: mean_Ex = sum(Ex)/len(Ex)
      deviation_from_mean = Ex-mean_Ex
      percent_deviation = deviation_from_mean / 100
      percent_deviation_str = "{:.5f}".format(max(percent_deviation))
      print('The maximum deviation from the mean value of E[x] is ' +␣
       ↪percent_deviation_str + '%')
```

```
The maximum deviation from the mean value of E[x] is 0.00011%
```

This sort of deviation can be atributted to the errors caused by the approximations used for $\frac{d(\cdot)}{dt}$ and $\frac{d^2(\cdot)}{dt^2}$. Hence one can assume $E[x] \approx cte. \approx 0$. Therefore, the first property is met.

The third bullet point stated that the autocovariance at $t = 0$, which is equal to the variance, should be finite. Looking at the plot, one can conclude that the variance is finite as it does not blow up to infinity anywhere within the 10 seconds of the simulation.

Furthermore, the variance of a WSS process should tend to a constant value for BIBO stable systems. If the system is not BIBO stable, the values for the mean and the variance would actually diverege for different realizations. It has already been proven that the system is BIBO stable, meaning its poles are within the unit circle. However, the poles are very close to the unit circle boundary, a result of which can be seen in the variance plot. The variance does rise above zero rapidly and looks as though it could be oscillating around a constant value but the amplitude is too large to conclude whether that is true. The simulation should be repeated for larger numbers of realizations and/or compared to theoretical values in order to be certain if the variance tends to a constant value.

2.8 Question 8

The mean and variance of the stochastic process $x(t)$ can also be calculated analytically. This leads to the following expressions,

$$\mu_x(t) = e^{-\gamma t/2}\left(x_0 \cos(\omega' t) + \gamma x_0 \frac{\sin(\omega' t)}{2\omega'}\right),$$

$$\sigma_x^2(t) = \frac{\beta^2}{2\gamma\omega^2} + e^{-\gamma t}\left(\frac{\beta^2}{8\gamma\omega'^2\omega^2}\right)\big(-4\omega^2 + \gamma^2\cos(\omega' t) - 2\gamma\omega'\sin(2\omega' t)\big).$$

Here, ω' is another frequency parameter with the value $\omega' = \sqrt{\omega^2 - \gamma^2/4}$, and x_0 is the value of x at time $t = 0$(note that in this case this value greatly simplifies the expression). Compare the theoretical mean and variance with the estimated ones from question 6 for different values of L.

provide:

- *The script used to calculate the theoretical mean and variance.* **1 point**
- *A plot of both the theoretical mean and variance together with the estimated onces from question 6 as a function of time. Plot all four sequences in one plot. make three plots one for $L = 30$, one for $L = 300$, and one for $L = 3000$.* **2 points**

2.8.1 Answer 8

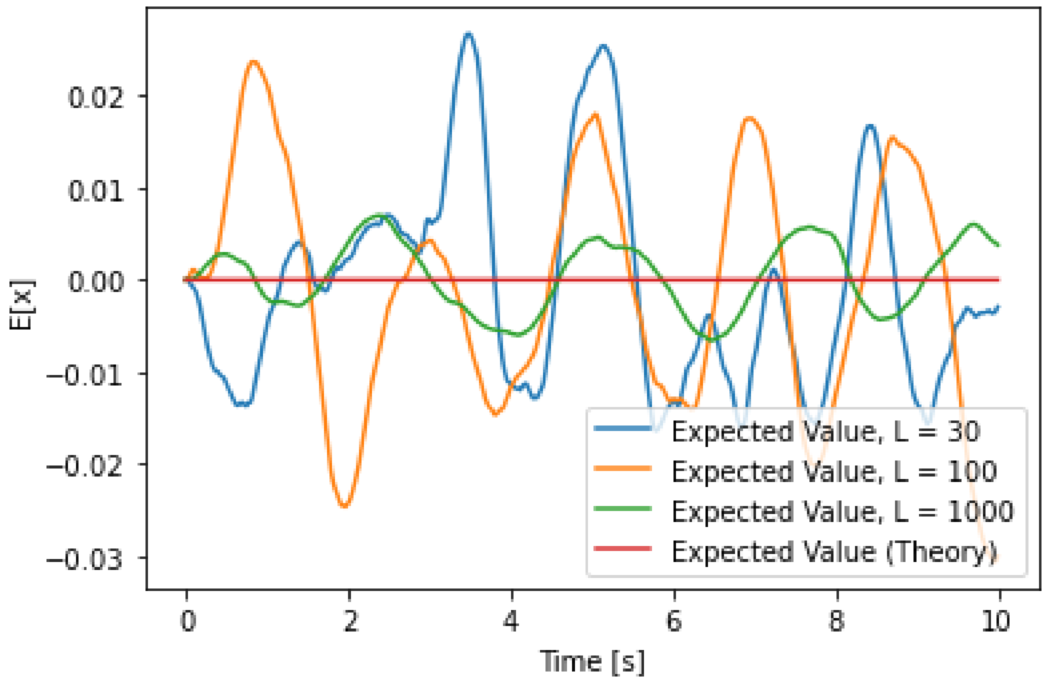
[20]:
```
L_vec = (30, 100, 1000)

Ex = np.zeros((N+1, len(L_vec)))  # Vector to hold the expected values of x for␣
↪each L, calculated with the given formula.
Ex2 = np.zeros((N+1, len(L_vec)))  # Vector to hold the expected values of x^2␣
↪for each L, calculated with the given formula.
Var = np.zeros((N+1, len(L_vec)))  # Vector to hold the variance of x for each␣
↪L, calculated with the given formula.

for i in range (0,len(L_vec)):
    L = L_vec[i]
    xL = np.zeros((N+1, L+1))
    wL = np.zeros((N+1, L+1))

    for l in range(0,L+1):
        for k in range(2,N+1):
            wL_k = np.random.randn()
            xL_k = b3 * wL_k - b1 * xL[k-1, l] - b2 * xL[k-2, l];
            xL[k,l]=xL_k
            wL[k,l]=wL_k

    Ex[:,i] = 1/L * np.sum(xL, axis=1)
    Ex2[:,i] = 1/L * np.sum(xL**2, axis=1)
    Var[:,i] = Ex2[:,i] - Ex[:,i]**2
```

[21]:
```
# Calculation of the Theoretical values of E[x] and \sigma^2.
omega_ = math.sqrt(omega**2 - gamma**2/4)
x0 = 0
avg = np.zeros(N+1)
sigma2 = np.zeros(N+1)
aprox_var = np.zeros(N+1)
```

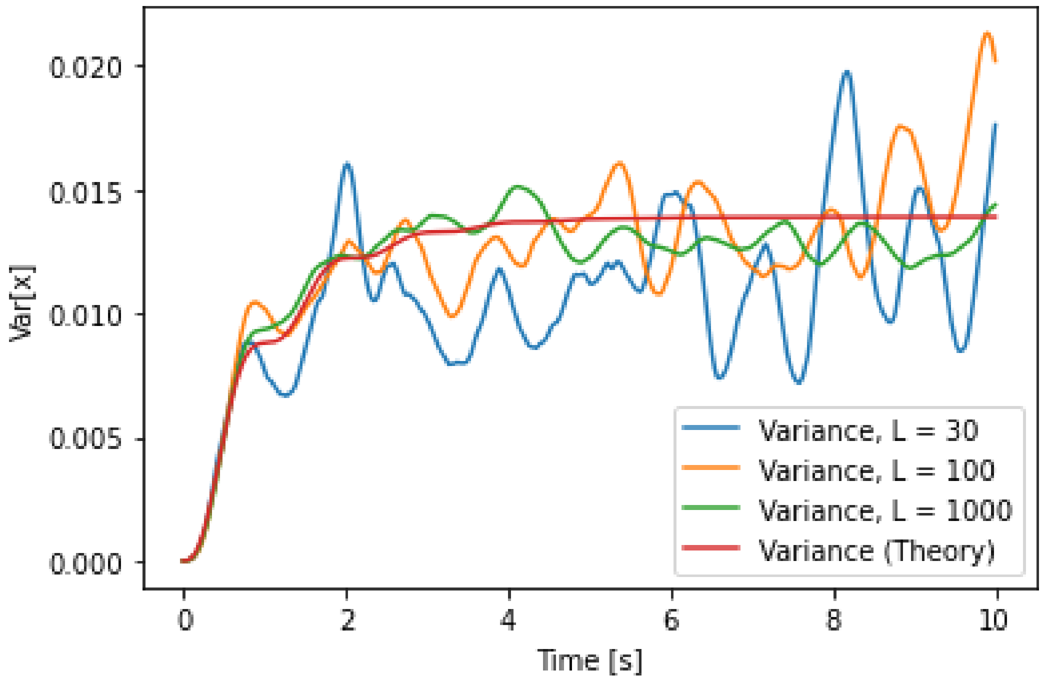
```
for k in range(0,N+1):
    var1 = (beta**2)/(2*gamma*omega**2)
    var2 = math.exp(-gamma*k*Delta_t)*((beta**2)/
 ↪(8*gamma*(omega_**2)*(omega**2)))
    var3 = (-4*omega**2 + (gamma**2)*math.
 ↪cos(omega_*Delta_t*k)-2*gamma*omega_*math.sin(2*omega_*Delta_t*k))

    sigma2[k] = var1 + var2*var3
    avg[k] = math.exp(-gamma*k*Delta_t/2)*(x0*math.cos(omega_*k*Delta_t) +␣
 ↪gamma*x0*(math.sin(omega_*k*Delta_t))/(2*omega_))
```

[22]:
```
plt.figure(1)
plt.plot(time, Ex[:,0], label="Expected Value, L = 30")
plt.plot(time, Ex[:,1], label="Expected Value, L = 100")
plt.plot(time, Ex[:,2], label="Expected Value, L = 1000")

plt.plot(time , avg, label="Expected Value (Theory)")
plt.legend(loc="lower right")
plt.xlabel('Time [s]')
plt.ylabel('E[x]')

plt.figure(2)
plt.plot(time, Var[:,0], label="Variance, L = 30")
plt.plot(time, Var[:,1], label="Variance, L = 100")
plt.plot(time, Var[:,2], label="Variance, L = 1000")


plt.plot(time, sigma2, label="Variance (Theory)")
plt.legend(loc="lower right")
plt.xlabel('Time [s]')
plt.ylabel('Var[x]')
```

[22]:
```
Text(0, 0.5, 'Var[x]')
```



0.02
0.01
0.00
−0.01
−0.02
−0.03
E[x]
0
2
4
6
8
10
Time [s]
Expected Value, L = 30
Expected Value, L = 100
Expected Value, L = 1000
Expected Value (Theory)



0.020
0.015
0.010
0.005
0.000
Var[x]
0
2
4
6
8
10
Time [s]
Variance, L = 30
Variance, L = 100
Variance, L = 1000
Variance (Theory)

2.9 Question 9

How do the estimations of the mean and variance vary with respect to the number of realizations L? **1 point**

2.9.1 Answer 9

As we can see from the plots above, as the number of realizations increases, the plots of the Mean and Variance get closer to the true values.

This result makes sense because, as the number of realizations increases, we also expect the influence of the stochastic process $w(k)$ to trend to the mean of this stochatistic process, $E[x] = 0$. What this implies is that, as the number of realizations increases, the plots of the Mean and Variance will tend to their true values.

2.10 Question 10

Use the results from questions 6-9 and from question 3 to make a final conclusion on whether the stochastic process $x(k)$ is WSS or not, with motivation. **2 points**

2.10.1 Answer 10

WSS stochastic processes have a time invariant and finite mean, and a finite constant variance. The mean has already been proven to be constant in Question 7, which is again confirmed based on the results of Question 8. With more realizations, the plot of the simulation approaches the theoretical function ($m_x = 0$) more and more closely. Regarding the variance of BIBO stable systems, its plot should rapidly rise from zero to the same constant value for each simulation. Such behavior is especially observed for systems with poles well within the unit circle. The system examined in this assignment has poles that are just on the verge of being on the unit circle, which means that the variance may not converge to the constant value immediately. The calculated theoretical function rises to a constant value in about 2s and so do the simulations with large numbers of realizations. It can be therefore concluded that a constant value is indeed achieved. To sum up, based on the mean and the variance behavior of the system, it can be concluded that the system is WSS.