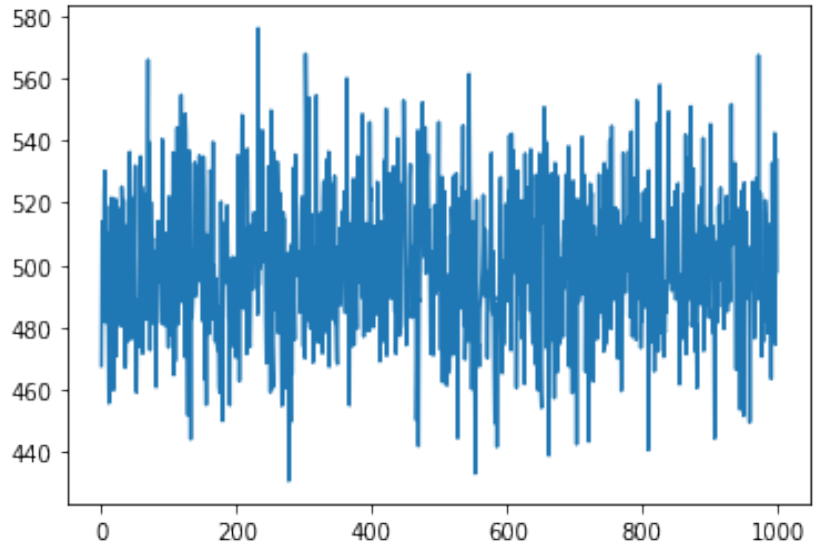
CT303 - Digital Communications
Autumn 2020

Lab *3:Noise and PSD* Group: *A9*

1. Function myawgn takes in PSD, Bandwidth, sampling frequency,and length of the sequence as the input. To generate an AWGN, we create a random signal which has mean zero and variance one and we create N realizations for every signal. To keep the variance of the noise signal equal to a signal having PSD equal to the value which is fed in input, we multiply it with square root of 2*PSD*Bandwidth. Then we take the fourier transform of the signal by multiplying the signal with its tranpose followed by taking the mean and dividing it by T. The PSD obtained is equal to the value of PSD which has been fed in the input.

   (a) fs=200 Bandwidth=100 PSD=500 Length=1000



2. The goal was to generate a Gaussian distribution using a function mygauss which takes in a mean vector, covariance matrix and number of samples as input. This function works similar to randn.m function but the randn.m function gives an output considering the mean to be 0 and variance to be 1. In mygauss function, we generate the sigma matrix by the cholesky decomposition of the covariance matrix. cholesky function used from scipy library is used. The covariance matrix is passed in the function and the upper triangle matrix returned can be used a variance.

   To get our desired matrix as output, a matrix of random numbers is generated using random.randn function. Dot product of this randomly generated matrix with mean 1 and variance 0 is taken with the matrix obtained after cholesky decomposition of covariance matrix. Mean is added to the resultant matrix and is returned by the function mygauss.

```
samples = myguss(mean,cov,1500)
samples
```

```
array([[ 1.92984449,  2.63108794],
       [ 1.5199997 ,  3.29491214],
       [ 0.24508389, -0.12880876],
       ...,
       [ 1.61021555,  2.30634823],
       [ 0.47238874,  0.38403069],
       [ 2.80865606,  5.28752793]])
```

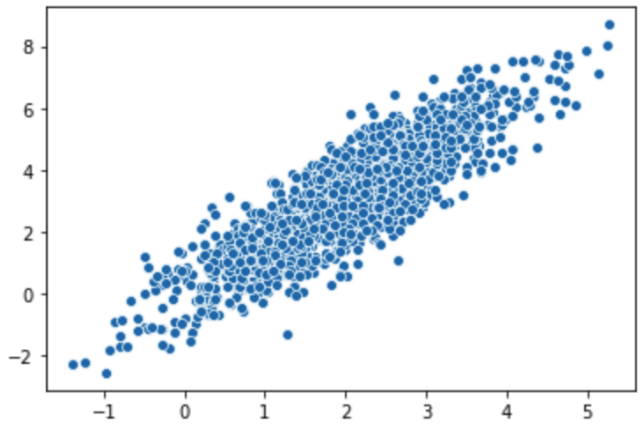
To check if the mygauss function is working correctly it is tested later in the code by providing dummy values and checking if the output matches with it.

```
[41] samples.mean(0)
```

```
array([1.97820729, 2.99137825])
```

```
sns.scatterplot(x=samples[:,0],y=samples[:,1])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f182d2cb160>
```




```
[40] np.cov(samples[:,0],samples[:,1])
```

```
array([[1.05115178, 1.58053546],
       [1.58053546, 3.15022191]])
```

The mean and covariance of the result obtained after running the function are consistent with the mean vector and covariance matrix provided as input.