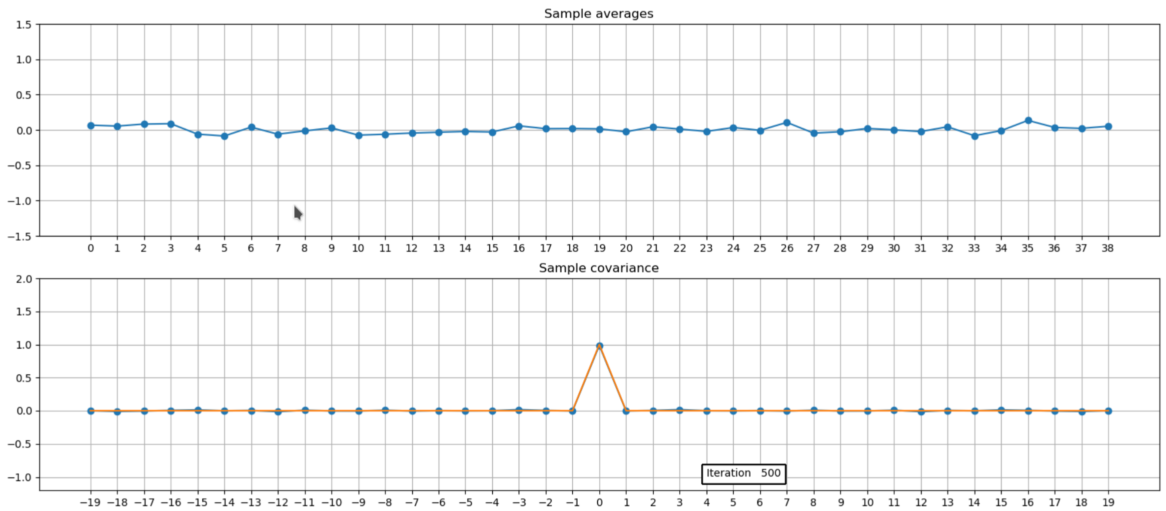
BAILLIEUL Alix
Dulout Eugénie

TP DE TSIA-202a

EXERCICE 1

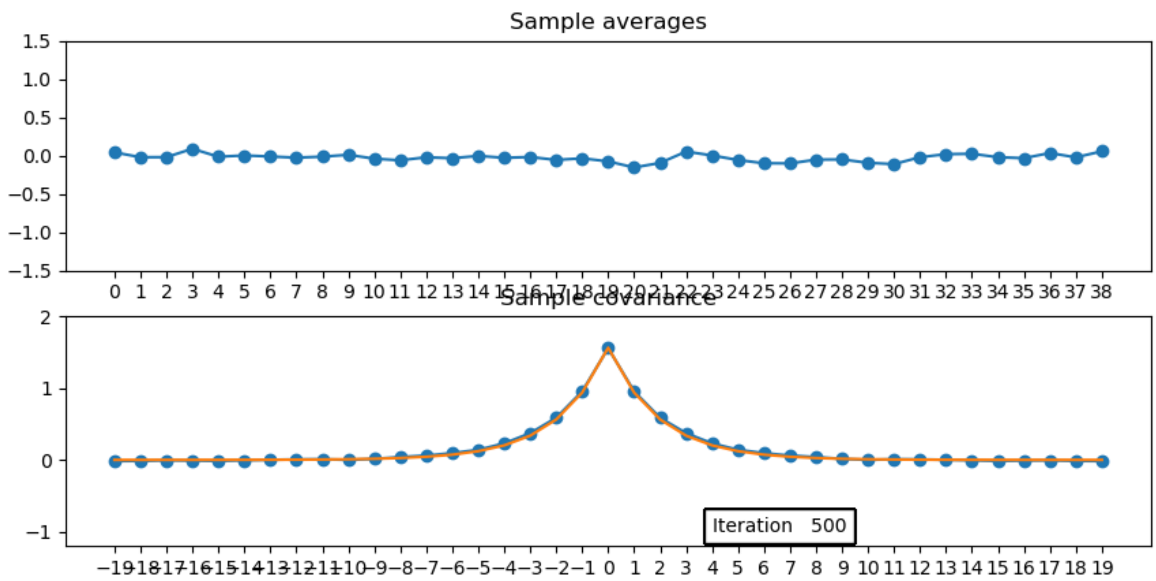
1) White Noise:

Fonction d'autocovariance théorique en orange et pratique en bleu sur le deuxième graphe.
en théorie, on a Gamma(t)={sigma² si t=0 et 0 sinon}



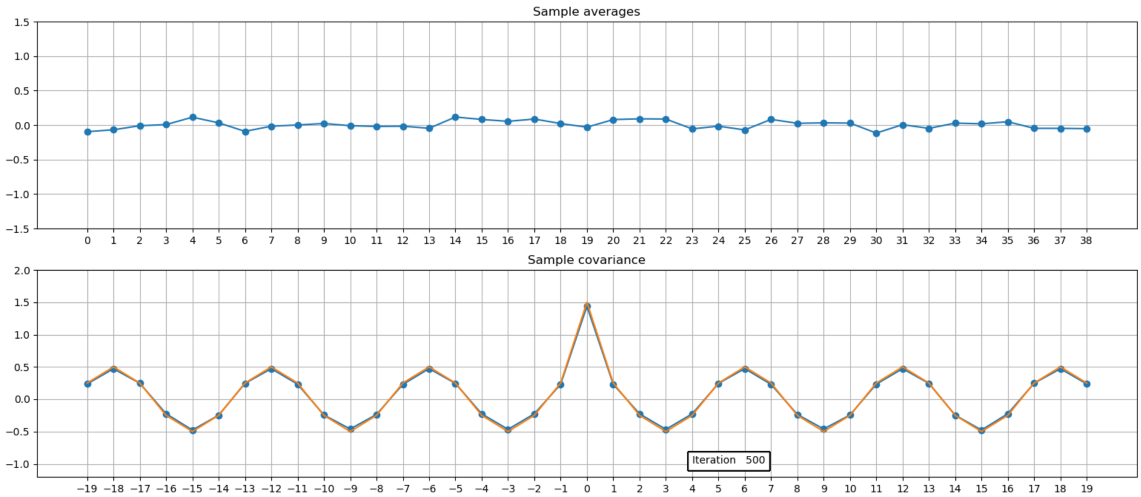
2) AR process:

Fonction d'autocovariance théorique en orange et pratique en bleu sur le deuxième graphe.
en théorie, on a Gamma(t)=ϕ**|t| / (1 - ϕ**2)



3) Sinusoidal process:

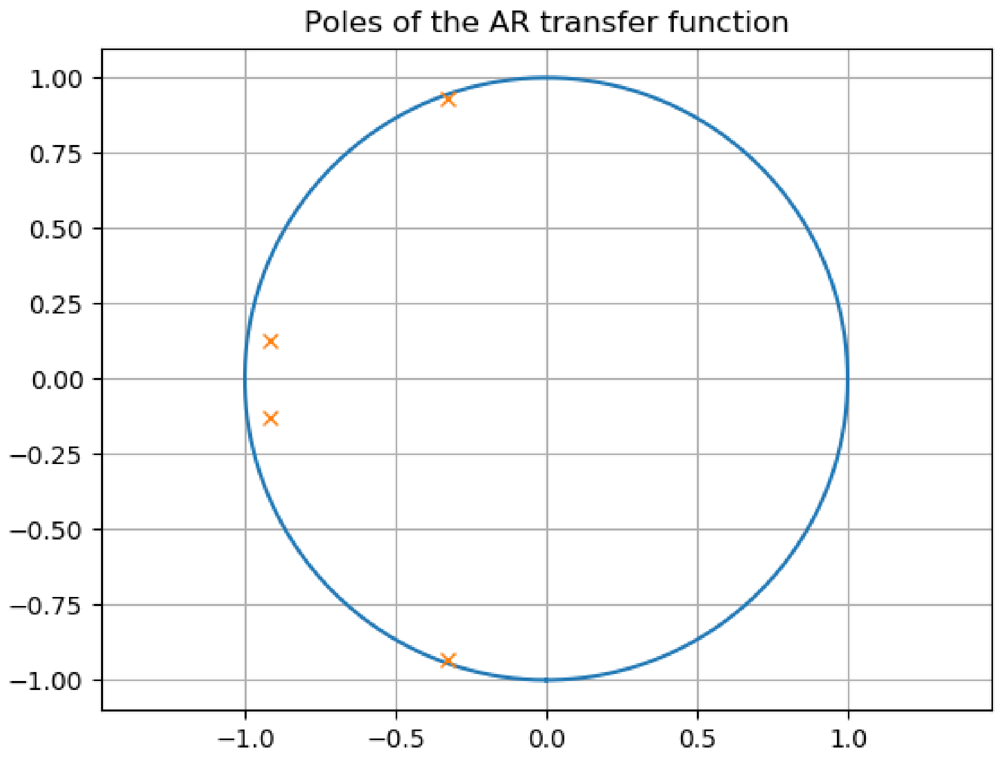
Fonction d'autocovariance théorique en orange et pratique en bleu sur le deuxième graphe. en théorie, on a Gamma(t)={(A²/2)*cos(omega0*t)+sigma**2 si t=0, (A²/2)*cos(omega0*t) sinon)

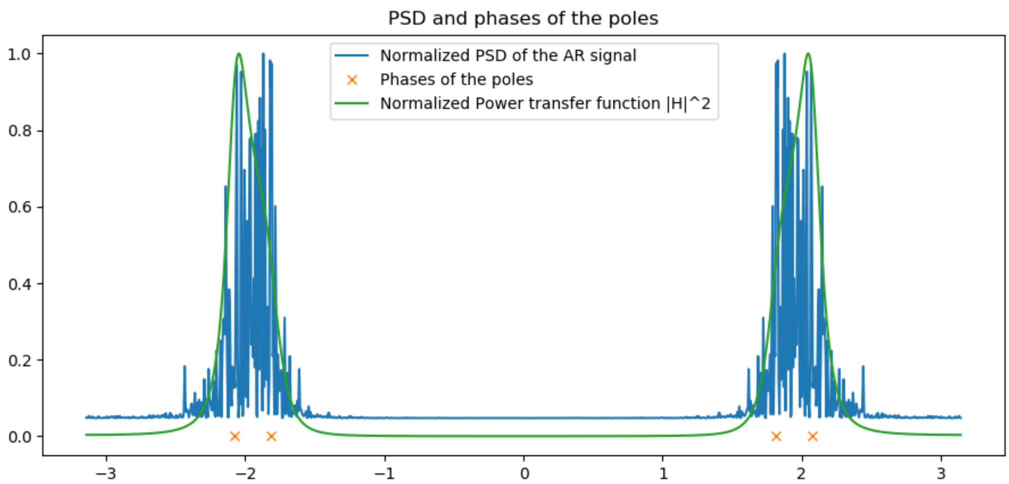


Pour les valeurs théoriques, on a utilisé le code suivant:

```
import numpy as np
import matplotlib.pyplot as plt



H=20
axe=np.arange(-H+1,H)
Y=np.zeros(len(axe))


sigma=1
A=1.0
w=np.pi/3
phi=0.6

signal='SIN' #AR ou SIN ou WN

if signal=='SIN':
    for t in range(len(Y)):
        Y[t]=(A**2/2)*np.cos(w*(t-H+1))
    Y[H-1]+=sigma**2

elif signal=='AR':
    for t in range(len(Y)):
        Y[t]=phi**abs(t-H+1)/(1-phi**2)

else:
    Y[H-1]=sigma**2

plt.plot(axe,Y)
```

Pour le filtre AR, on obtient avec la fonction rp.drawZ_DTFT_AR :





EXERCICE 2

Q1.

1) On a : $I_n(\lambda) = \frac{1}{2\pi} \sum_{k=-n+1}^{n-1} \hat{\gamma}_n(k) e^{-i\lambda h}$

$$= \frac{1}{2\pi} \sum_{k=-n+1}^{n-1} \left(\sum_{k=0}^{n-1-k} \frac{1}{n} (X_k - \hat{\mu}_n)(X_{k+h} - \hat{\mu}_n) \right) e^{-i\lambda h}$$

$$= \frac{1}{2\pi n} \times \left(\sum_{k=0}^{n-1} \sum_{j=0}^{n-1} (X_k - \hat{\mu}_n)(X_j - \hat{\mu}_n) e^{-i\lambda(k-j)} \right)$$

$$= \frac{1}{2\pi n} \sum_{k=0}^{n-1} (X_k - \hat{\mu}_n) e^{-i\lambda k} \sum_{j=0}^{n-1} (X_j - \hat{\mu}_n) e^{+i\lambda j}$$

On note $\hat{X}$ la DTFT de $X_0 - \hat{\mu}_n \cdots X_{n-1} - \hat{\mu}_n$

On a $\hat{X}(\lambda) = \frac{1}{2\pi} \sum_{k=0}^{n-1} (X_k - \hat{\mu}_n) e^{-i\lambda h}$

Donc $I_n(\lambda) = \frac{2\pi}{n} \times \hat{X}(\lambda) \times \overline{\hat{X}(\lambda)}$

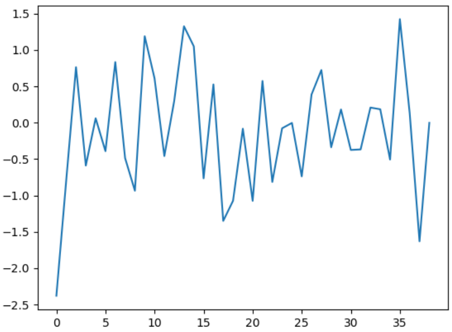
Donc $I_n(\lambda) = \frac{2\pi}{n} |\hat{X}(\lambda)|^2$

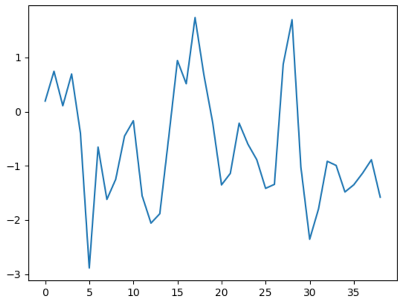
Q2.

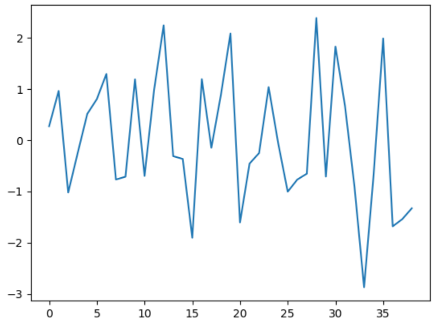
on utilise le code suivant pour la fonction I:

```
def I(X,m):
    L=np.arange(m)
    n=len(X)
    moy=0
    for x in X:
        moy+=x
    moy=moy/n
    for k in range(m):
        tftd=0
        for i in range(n):
            tftd+=(X[k]-moy)*np.exp(-2j*np.pi*k*i/m)
        tftd=tftd/(2*np.pi)
        L[k]=(2*np.pi/n)*abs(tftd)**2
    return L
```

On obtient en testant la fonction:

pour WN: 

pour AR: 

pour SIN: 

Q3.

On a $I_n(2\pi \frac{k}{m}) = \frac{1}{2\pi} \sum_{h=-n+1}^{n-1} \hat{\gamma}_n(h) e^{-2i\pi \frac{kh}{m}}$

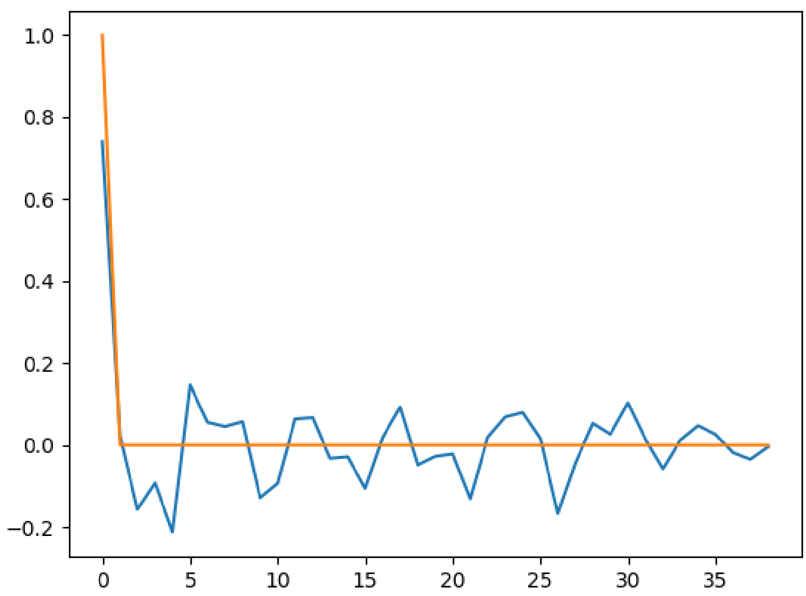
Donc si $m = 2n-1$, en appliquant la TFTD inverse, on obtient:

$$\hat{\gamma}_n(h) = \frac{2\pi}{2n-1} \sum_{k=-n+1}^{n-1} I_n\left(2\pi \frac{k}{2n-1}\right) e^{2i\pi \frac{kh}{2n-1}}$$

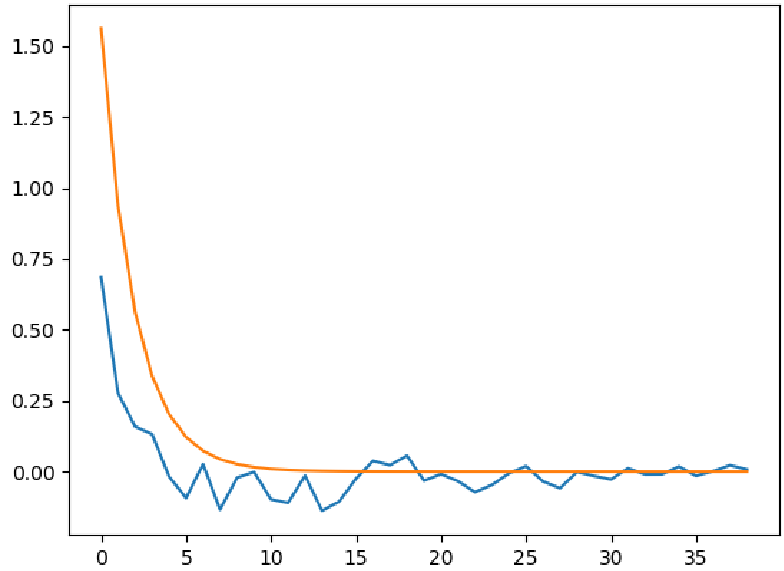
Avec le code suivant, on affiche le résultat de l'autocovariance obtenue avec la fonction acovb que l'on compare avec l'autocovariance théorique, on remarque que sur un échantillon de 20 valeurs, les résultats sont déjà relativement proche de ce qui est attendu:

```python

cov=acovb(X)
axe=np.arange(len(cov))
plt.plot(axe,cov)

Y=np.zeros(len(cov))

if signal=='SIN':
    for t in range(len(Y)):
        Y[t]=(A**2/2)*np.cos(w*(t))
    Y[0]+=sigma**2

elif signal=='AR':
    for t in range(len(Y)):
        Y[t]=phi**abs(t)/(1-phi**2)

else:
    Y[0]=sigma**2

plt.plot(axe,Y)


plt.show

```

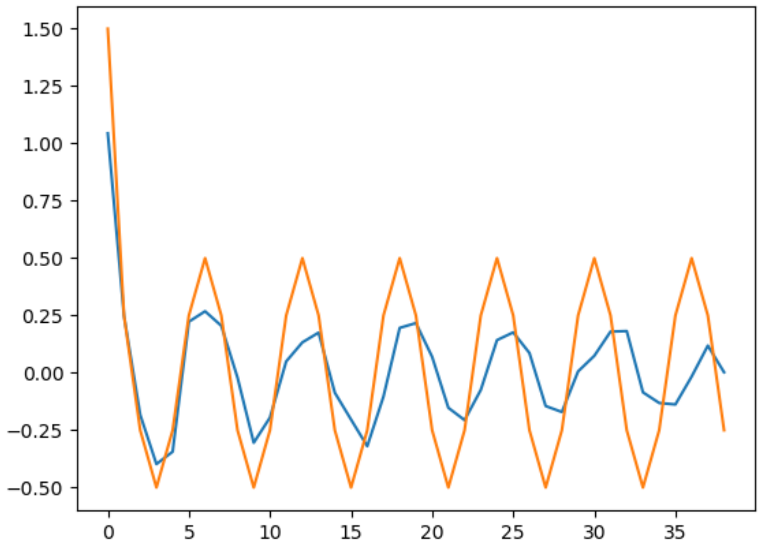
pour WN:



pour AR:



pour SIN:



Q4.

EXERCICE 3

Partie 1 - Yule-Walker equation

Q1.

$X_t = \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \dots + \varphi_p X_{t-p} + Z_t$, Z_t WN$(0,\sigma^2)$

$\mathbb{E}(X_{t-h} Z_t) = \mathbb{E}(\varphi_1 X_{t-h-1} Z_t + \varphi_2 X_{t-h-2} Z_t + \dots + \varphi_p X_{t-h-p} Z_t + Z_{t-h} Z_t)$

$= \sum_{i=1}^{p} \varphi_i \mathbb{E}(X_{t-h-i} Z_t) + \underbrace{\mathbb{E}(Z_{t-h} Z_t)}_{=0 \text{ because white noise}}$

Let's compute X_{t-i} for $i \leq p$.

$X_{t-(p-1)} = \varphi_p X_{t-p} + Z_t$

$X_{t-(p-2)} = (\varphi_{p-1}\varphi_p + \varphi_p) X_{t-p} + Z_t$

$\vdots$

$X_{t-i} = \varphi_i' X_{t-p} + Z_t$.

Then $X_{t-p} = \alpha_{t-p} X_0 + Z_t$. Therefore we can rewrite all X_{t-i} as $\alpha_{t-i} X_0 + Z_t$.

Then $\mathbb{E}(X_{t-h} Z_t) = \alpha_{t-h} \mathbb{E}(X_0 Z_t) + 0$.

Since X_t is causal, $X_0 = 0 + 0 + \dots + 0 + Z_0$.

Finally, $\mathbb{E}(X_{t-h} Z_t) = \alpha_{t-h} \mathbb{E}(Z_0 Z_t) = 0$.

Q2.

We have $\gamma(h) = \mathbb{E}(X_{t-h} X_t) - \underbrace{\mathbb{E}(X_{t-h})}_{=0} \underbrace{\mathbb{E}(X_t)}_{=0}$

$\gamma(h) = \mathbb{E}\left(\sum_{i=1}^{p} \varphi_i X_{t-h} \cdot X_{t-i} + Z_t X_t\right) = \sum_{i=1}^{p} \varphi_i \underbrace{\mathbb{E}(X_{t+h-i} X_t)}_{=\gamma(h-i)} + \underbrace{\mathbb{E}(X_t Z_t)}_{=0}$

$\Rightarrow \boxed{\gamma(h) = \sum_{i=1}^{p} \varphi_i \gamma(h-i)}$

Q3.

For $h=0$, $\gamma(0) = \mathbb{E}(X_t X_t) = \mathbb{E}(X_t^2) = \mathbb{E}(X_0^2) = \mathbb{E}\left(\sum_{i=1}^{p} \varphi_i X_{-i} X_0 + Z_0\right)$

$$= \sum_{i=1}^{p} \varphi_i \, \mathbb{E}(X_{-i} X_0) + \mathbb{E}(X_0 Z_0)$$

$$\boxed{\gamma(0) = \sum_{i=1}^{p} \varphi_i \gamma(-i) + \sigma^2.}$$

Q4.

$$\begin{cases} \gamma(h) = \sum_{i=1}^{p} \varphi_i \gamma(h-i) & \forall h \geq 1 \\ \gamma(0) = \sigma^2 + \sum_{i=1}^{p} \varphi_i \gamma(-i) \end{cases}$$

Then

$$\Gamma_{p+1} = \begin{pmatrix} \gamma(0) & \gamma(-1) & \cdots & \gamma(-p) \\ \gamma(1) & \gamma(0) & \ddots & \gamma(1-p) \\ \vdots & & \ddots & \\ \gamma(p) & \gamma(p-1) & \cdots & \gamma(0) \end{pmatrix} \begin{pmatrix} 1 \\ -\varphi_1 \\ \vdots \\ -\varphi_p \end{pmatrix} = \begin{pmatrix} \gamma(0) - \sum_{i=1}^{p} \varphi_i \gamma(-i) \\ \gamma(1) - \sum_{i=1}^{p} \varphi_i \gamma(1-i) \\ \vdots \\ \gamma(p) - \sum_{i=1}^{p} \varphi_i \gamma(p-i) \end{pmatrix} = \begin{pmatrix} \sigma^2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Partie 2 - Estimation

Q1 & Q2.

On génère un processus AR-4 de longueur N = 1000 à l'aide de la fonction [X,a] = genAR(p,N). On utilise la fonction acovb pour générer la séquence des autocovariances biaisées et on déduit une estimation Gamma,n,p+1 de Gamma,p+1.

```
[X,a] =genAR(4,1000)
np.mean(X)

gamma =  rp.acovb(X)

Gamma_hat = la.toeplitz(gamma[:p+1])
print(Gamma_hat)
```

Q3.

On a $\hat{\Gamma}_{n,p+1} \begin{bmatrix} 1 \\ -\varphi_1 \\ \vdots \\ -\varphi_p \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \sigma^2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$

$$\Rightarrow \frac{1}{\sigma^2} \begin{bmatrix} 1 \\ -\varphi_1 \\ \vdots \\ -\varphi_p \end{bmatrix} = \hat{\Gamma}_{n,p+1}^{-1} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$\Rightarrow \frac{1}{\sigma^2} = v[0] \Rightarrow \sigma^2 = \frac{1}{v[0]}$$

Puis $(\varphi_i)_{i \in (1,p)} = v[i] \times \sigma^2$.

Voici le code correspondant:

```
[X,a] =genAR(4,1000)


gamma =  rp.acovb(X)

Gamma_hat = la.toeplitz(gamma[:p+1])
#print(Gamma_hat)
Gamma_hat_inv=la.inv(Gamma_hat)

v1 = np.zeros(p+1)
v1[0]=1

v=np.dot(Gamma_hat_inv,v1)



sigma2Est = 1/v[0]
estimated_coeff= [v[i]*sigma2Est for i in range(1,p+1)]


print(estimated_coeff )
print(sigma2Est)
```

```
[-1.3110978654547587, 0.9235659209371755, -0.7763309708007593, 0.3127765341065186]
0.9861260320945275
```

L’erreur relative est (1-0.986)=1,4%.

#Compare to the poles ?

Partie 3- Application to speech signal