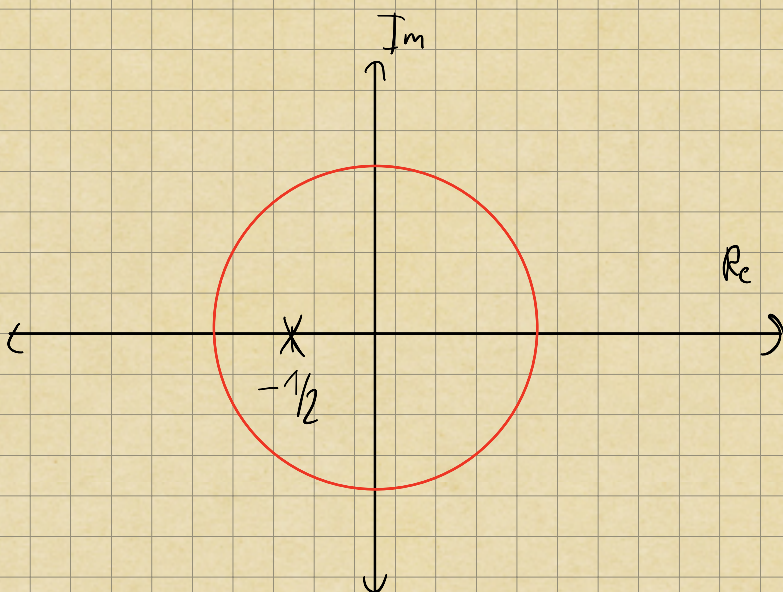
problem 1)

- A randome signal $x(n)$ is generated by filtering white Gaussian noise $w(n)$ with variance $\sigma_w^2 = \frac{3}{4}$ by a causal filter with transfer function

$$H(z) = \frac{1}{1 + \frac{1}{2} z^{-1}}$$

a)



Since the filter $H(z)$ only has poles it is a autoregressive process AR[1]

b) Find the coefficients of the optimal first and second order predictor.

From definition:

$$\hat{X}(n) = -a_1 X(n-1)$$

We can find the prediction coefficient $-a_1$ by minimazing the prediction error:

$$\sigma_f^2 = E\{f^2(n)\}, \quad f(n) = X(n) - \hat{X}(n)$$

$$\sigma_f^2 = E\left(\left(X(n) - \hat{X}(n)\right)^2\right) = E\left(\left(X(n) + a_1 X(n-1)\right)^2\right)$$

Since $\sigma_f^2(a)$, we can minimize by math

$$\frac{\partial \sigma_f^2}{\partial a} = 0$$

$$\frac{\partial \sigma_f^2}{\partial a} = E\left(\left(X(n) + a_1 X(n-1)\right) \cdot 2 \cdot X(n-1)\right)$$

$$= E\left(2\left(X(n)X(n-1) + a_1 X(n-1)X(n-1)\right)\right)$$

$$= E\left(2\left(X(n)X(n-1)\right)\right) + E\left(2a_1 X(n-1)X(n-1)\right)$$

We know from lecture that:

$$r_{xx}(l) = E\left\{X(n)\,X(n-l)\right\}$$

$$\Rightarrow \quad 0 = 2\,r_{xx}(1) + 2a_1 r_{xx}(0)$$

$$\Rightarrow \quad \underline{a_1 = -\frac{r_{xx}(1)}{r_{xx}(0)}}$$

With value we found prev exercise

$$r_{xx}(l) = = \left(-\frac{1}{2}\right)^{|l|}$$

$$\Rightarrow \quad a_1 = -\frac{-\frac{1}{2}}{1}$$

$$\underline{\underline{a_1 = 1/2}}$$

Second order prediction:

$$X(n) = -a_1 X(n-1) - a_2 X(n-2)$$

Would get the same value as using AR(1) process

Proplem 2)

$$X(n) = w(n) - 0.4w(n-1)$$

, $w(n)$ is White Gaussian noise $\sigma^2_w = 1$

a) $X(n)$ has only zeroes and is therefore a MA process.

And since only the current and former value of the input signal are used in forming the output-signal.

b) Find the autocorrelation $r_{xx}(l)$ and the power density spectrum $\Gamma_{xx}(f)$ for this process.

$$r_{xx}(l) = E\{X(n)X(n-l)\}$$

$$= E\{(w(n) - 0.4w(n-1))(w(n-l) - 0.4w(n-1-l))\}$$

$$= E\{w(n)w(n-l) - 0.4w(n)w(n-1-l) - 0.4w(n-1)w(n-l) + 0.4^2 w(n-1)w(n-1-l)\}$$

$$= E\{w(n)w(n-l)\} - 0.4E\{w(n)w(n-1-l)\} - 0.4E\{w(n-1)w(n-l)\} + 0.16E\{w(n-1)w(n-1-l)\}$$

+1 +1

$$\Rightarrow r_{xx}(l) = r_{ww}(l) - 0.4r_{ww}(1+l) - 0.4r_{ww}(l-1) + 0.16r_{ww}(l)$$

$$= 1{,}16r_{ww}(l) - 0.4(r_{ww}(1+l) + r_{ww}(l-1))$$

$\boxed{r_{xx}(l) = E\{X(n)X(n-l)\}}$

Using this.

We know that the auto correlation for white noise is: $\gamma_{ww}(l) = \sigma_w^2 \delta(l)$

$$\Rightarrow \gamma_{xx}(l) = 1.16\,\sigma_w^2 \delta(l) - 0.4\left(\sigma_w^2(1+l) + \sigma_w^2(l-1)\right)$$

$$\gamma_{xx}(l) = \begin{cases} 1.16, & l = 0 \\ -0.4, & l = \pm 1 \\ 0, & \text{else} \end{cases}$$

Power density spectrum:

$$\Gamma_{xx}(f) = \sum_{l=-\infty}^{\infty} \gamma_{xx}(l) e^{-j2\pi f l}$$

$$= \sum_{-1}^{1} \gamma_{xx}(l) e^{-j2\pi f l}$$

$$= -0.4e^{-j2\pi f} + 1.16 - 0.4e^{-j2\pi f \cdot (-1)}$$

$$= -0.4 \cdot 2\cos(2\pi f) + 1.16$$

$$\Gamma_{xx}(f) = -0.8\cos(2\pi f) + 1.16$$

c) The optimal predictor of order p is given by:

$$\hat{x}(n) = -\sum_{k=1}^{p} a_k x(n-k)$$

We find the equations:

First order:

$$-r_{xx}(1) = r_{xx}(0) a_1$$

Second order:

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) \\ r_{xx}(-1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -r_{xx}(-1) \\ -r_{xx}(-2) \end{bmatrix}$$

Third order:

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & r_{xx}(2) \\ r_{xx}(-1) & r_{xx}(0) & r_{xx}(1) \\ r_{xx}(-2) & r_{xx}(-1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} -r_{xx}(-1) \\ -r_{xx}(-2) \\ -r_{xx}(-3) \end{bmatrix}$$

And then find the variance with

$$\underline{\sigma_f^2 = \sum_{k=0}^{p} a_k r_{xx}(k)}, \quad 1 \cdot r_{xx}(0) + a_1 r_{oo}(1)$$

$$1 \cdot r_{xx}(0) + a_1 r_{xx}(1) + a_2 r_{xx}(2)$$

```
First order Yu-Walker equation:

The coefficent of a1 is [0.34482759]

Second order Yu-Walker equation:

The coefficient of a1 is [0.39136302]. The coefficient of a2 is [0.13495277]

Third order Yu-Walker equation:

The coefficient of a1 is [0.39862284]. The coefficient of a2 is [0.15600624]. The coefficient of a3 is [0.05379526]
```

Code:

DIGSIG Øving 8 > Oppgave2Pånytt.py > ...

```python
##Solving the Yule-Walker equations to get the coefficients of the AR model
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy.linalg import solve

#Defining the AR model

#AN np.array with 1.16 as 0 value and -0.4 as 1 and -1 value, and 0 for the rest
#NOTE: For values that is zero in the array I will just type in zero
Yxx = np.array([1.16, -0.4,-0.4])
#Add zeroes to Yxx

##Fist order Yu-Walker equation

#Equation on form Ax = B
B1 = -Yxx[1]
A1 = Yxx[0]

X1 = solve(A1,B1)
#print with line
print("First order Yu-Walker equation: \n")
print("The coefficent of a1 is " + str(X1)+"\n")

#Second order Yu-Walker equation

#Creating a 2x2 matrix with values from Yxx
A2 = np.array([[Yxx[0],Yxx[1]], [Yxx[-1],Yxx[0]]])
#Create a 1x2 matrix with values from B2
B2 = np.array([[-Yxx[-1]],[0]])
X2 = solve (A2,B2)
#print with line
print("Second order Yu-Walker equation: \n")
print("The coefficient of a1 is " + str(X2[0]) + ". The coefficient of a2 is " + str(X2[1]) + "\n")

#Third order Yu-Walker equation
#Creating a 3x3 matrix with values from Yxx
A3 = np.array([[Yxx[0],Yxx[1],0], [Yxx[-1],Yxx[0],Yxx[1]], [0,Yxx[-1],Yxx[0]]])
#Create a 1x3 matrix with values from B3
B3 = np.array([[-Yxx[-1]],[0],[0]])
X3 = solve (A3,B3)
#print with line
print("Third order Yu-Walker equation: \n")
print("The coefficient of a1 is " + str(X3[0]) + ". The coefficient of a2 is " + str(X3[1]) + ". The coefficient of a3 is " + str(X3[2]) + "\n")

```

Variance:

```
The variance of the first order is 1.0220689655172412. The variance of the second order is [1.00345479]. The variance of the theird order is [1.00055086]
```

We see that the variance decreases when we increase the model order. This is a sign that AR model is a better approx then the MA.

d) From lecture notes:

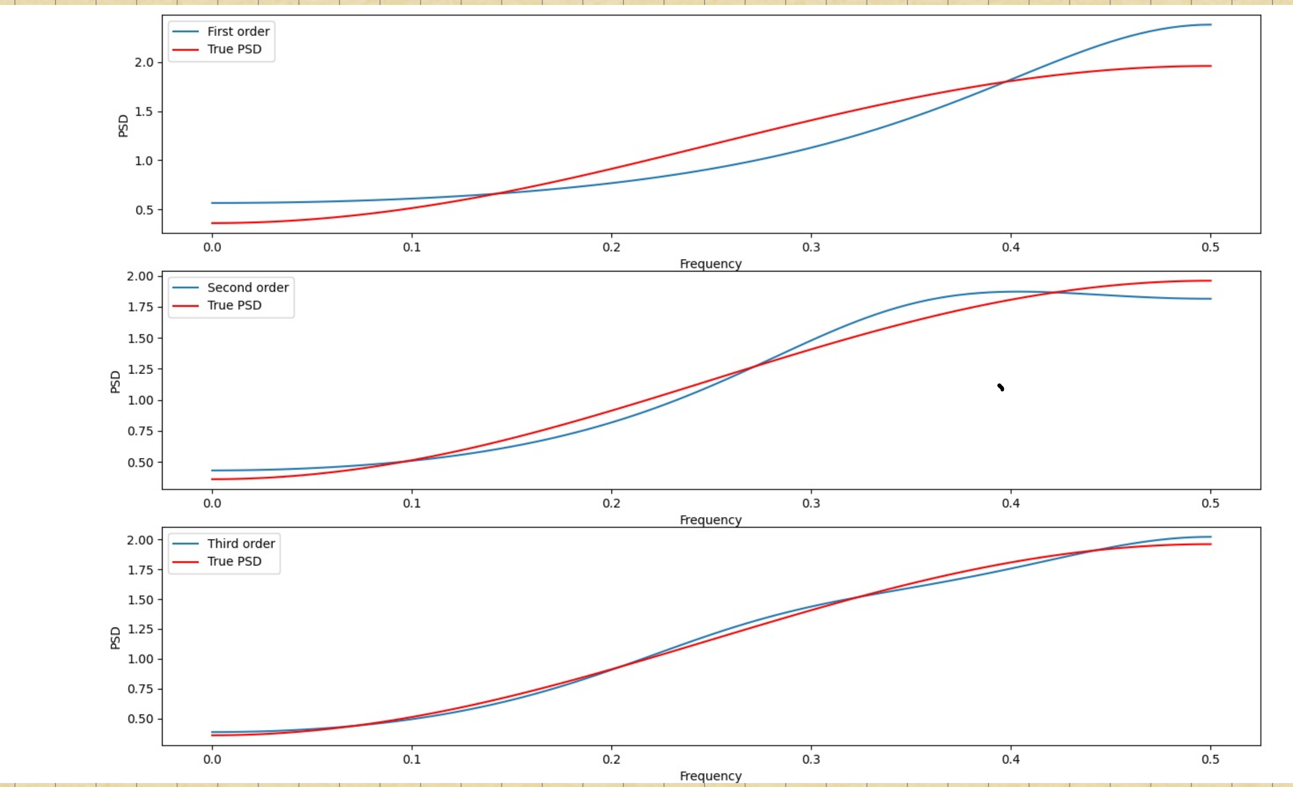
Approximation

$$\hat{\Gamma}_{ff}(f) = \frac{\sigma_f^2}{\left|1 + \sum_{k=1}^{p} a_k e^{-j2\pi f k}\right|^2}$$

Actual found in 2b:

$$\Gamma_{xx}(f) = -0.8\cos(2\pi f) + 1.16$$

Plots:



- As we can see from the plots the best approximation of the MA process is the third order. This is because it has the lowest variance.

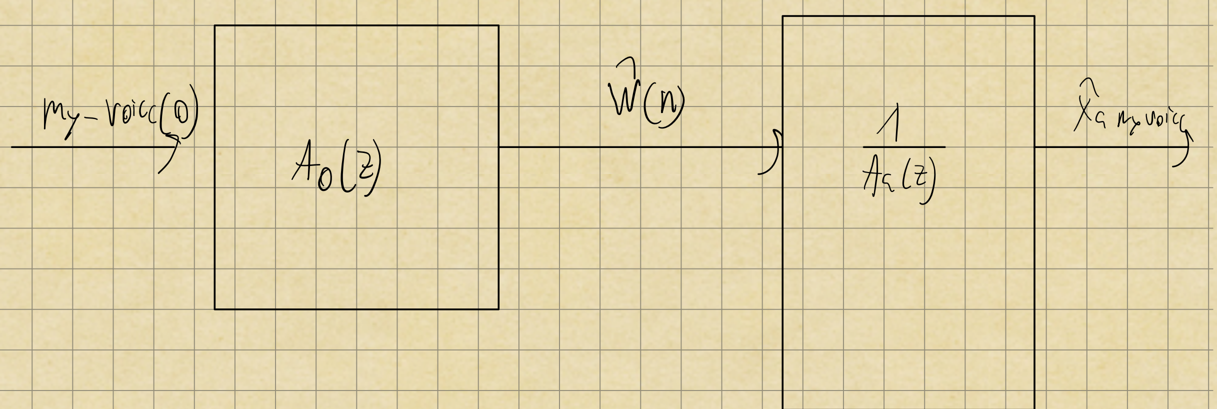
Problem 3)

What we want from System:

AR process: $H(z) = \dfrac{1}{1 + \sum_{k}^{q} p_k z^{-k}}$, order 9

Gameplan:

- We send in a sound signal (My voice) of a vowel
- → We send the recording of the vowel through the inverse filter with coefficients corresponding to the AR(%) process of my vowel
- → The final part is to send this through a filter with coefficients to another vowel, and the result should be the voice of the recorder saying that vowel.



Code:

```python
#First we get the data from the file
data = scipy.io.loadmat('vowels.mat')
norwegianVowels = data['v'][0]
fs = int(data['fs'][0][0])
#We now need a desiered vovel
desiredVowelIndex = 0
desiredNorwegianVowel = norwegianVowels[desiredVowelIndex]
#We now need to get the coefficents of the vovel
#Function the get teh coefficents
def ar_coefficients(signal, order=10):
    a = np.zeros(order + 1)
    e = np.zeros(order + 1)
    r = np.zeros(order + 1)

    for m in range(order + 1):
        r[m] = np.dot(signal[m:], signal[:-m] if m != 0 else signal)

    a[0] = 1.0
    e[0] = r[0]
    for k in range(1, order + 1):
        lambda_val = -np.dot(a[:k], r[k:0:-1]) / e[k-1]
        a[1:k+1] = a[1:k+1] + lambda_val * np.flip(a[:k])
        a[k] = lambda_val
        e[k] = (1 - lambda_val**2) * e[k-1]

    return a[1:]

coeffDesiredVowel = ar_coefficients(desiredNorwegianVowel.ravel())

#We need to get the coefficents of the input vovel
input_vowel, input_fs = sf.read('vowel2.wav')
coeff_input_vowel = ar_coefficients(input_vowel.ravel())
#To get the noice we need to use the recording of the wovel in my voice through the inverse filer
#With coefficents of the input vowel

#We now need to get the noice from the inverse-filter
inverseFilterOfOwnNoice = coeff_input_vowel
#We need to get the noice
noice = scipy.signal.lfilter(inverseFilterOfOwnNoice, [1], input_vowel)
#the final part is to take this noice and filter it with the coefficents of the desired vowel
#This will give us the transformed vowel
desiredNorwegianVowelSound = scipy.signal.lfilter([1],coeffDesiredVowel, noice)
#We need to normalize the sound
desiredNorwegianVowelSound = desiredNorwegianVowelSound / np.abs(desiredNorwegianVowelSound).max()
#We then need to play the sound
```