**Chromatic dispersion compensation using complex-valued all-pass filter**

In this project the principle of Chromatic Dispersion (CD) in optical channel is described, focusing on the part that it has an all-pass behavior, which means that the amplitude of the signal that passes through the channel is left unchanged, whereas the phase of the signal is changed due to CD. This CD can be compensated by using IIR or FIR filters. In this project I implemented an IIR compensation filter with complex coefficients for compensating CD.

**3. Practical examples**

**Abel-Smith algorithm**

The Abel-Smith algorithm is a method to extract the filter coefficients for an all-pass design with an arbitrary group delay. The main principle is to divide the area below τdesired(ω) into 2π area frequency bands. The area below τdesired(ω) is guaranteed to be an integer multiple of 2π because it is equal to 2πNIIR.

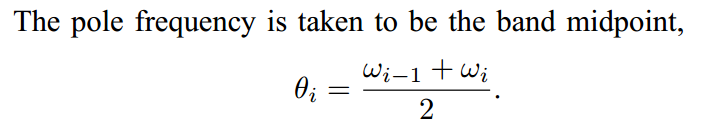
With the given parameters a value for alpha equal to α=0.7360 is obtained and a value for NIIR = 5. This two parameters are input parameters for the *abel\_smith* function, which returns the ρ and θ vectors. In this function, I chose ζ=0.8 (at the middle of the interval [0.75, 0.85]), after which the function *abel\_smith\_divide* is called, which takes as input parameters also α and NIIR and outputs the vector called *omega\_seg* which contains the pulsations between [–π,π] for which the area under τdesired(ω) is 2π.

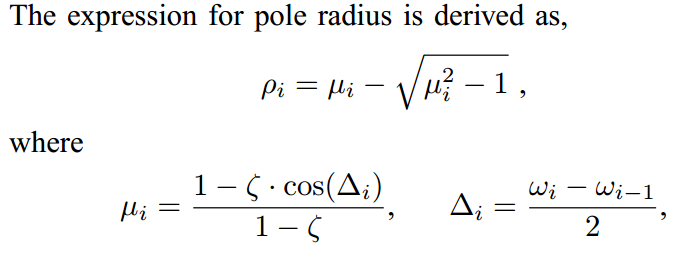
Within this function, in the first step β is computed and is equal to β = NIIR = 5. Afterwards, the first and last value of omega\_seg are initialized with –π and π respectively (omega\_seg has NIIR elements). In order to find the other values of ω in-between I did the following: I took the equation of the line describing τdesired(ω) and I chose two points on it: at the first iteration I chose ω0=-π and ω1 which is to be determined. The two points, together with the OX axis describe a trapezoid, whose area can be expressed in terms of ω1 and ω0, and this area is also equal to 2π. Doing this, we obtain a second order equation, which yields two solutions. From these two solutions, we choose only the one who is smaller in absolute value than the one previously determined. The algorithm is repeated until all points are calculated. A detailed derivation of the equation is included In the Appendix, page 3.

The values obtained for omega-seg are:

[-3.1415926535,-2.4524196594,-1.6703670709, -0.7431304218,0.4638963171,3.1415926535]

These values are then used to compute the ρ and θ vectors according to the formulas:





At the end, the values for ρ and θ are:

ρ = [0.5101779789,0.4682523645,0.4109212203,0.3236712115,0.1244310801]

θ = [-2.7970061565,-2.0613933652,-1.2067487464,-0.1396170523,1.8027444853]

**Bit error rate**

In this exercise a CD channel and CD equalization are implemented and a BER curve is obtained.

In the first task I generate the channel impulse response. For this I created a function called impulse\_response\_channel which takes as inputs α and the number of frequency points and outputs h\_CD. Within this function, after computing the omega values in the interval [-π,π] (namely the frequency points where the transfer function is going to be estimated), I compute:

H\_CD = exp(-1i\*alpha\*omega.^2)

After this, I apply the ***ifftshift*** to H\_CD because this function of Matlab works on the domain from [0,2π], and the frequency range for H\_CD is [-π,π]. Thus the domain between [0, π] coincides, whereas the interval from [π,2π] has to be shifted at the beginning of the H\_CD vector.

This being done, then the impulse response h\_CD is computed by taking the ***IFFT*** of H\_CD. After assuring that h\_CD is a column vector (through the command h\_CD = h\_CD(:)), the command ***fftshift*** is applied to h\_CD for a similar reason with the one described above. Finally,the channel response is plotted by making use of the ***stem*** command. The result can be seen Figure 1.

After loading the values for ρ,θ and Φ another function is called, namely *conv\_anyinput\_allpass\_equalizer* which takes as input arguments the previous variables, together with h\_CD and outputs the result of the convolution between the IIR filter described by ρ,θ,Φ and the input signal. Within this function the ***filter*** function of Matlab is used, together with the cascading of NIIR IIR order 1 filters, for which the corresponding b and a vectors are shown below:

for k=1:length(rho)

b = [-rho(k)\*exp(-1i\*theta(k)), 1];

a = [1, -rho(k)\*exp(1i\*theta(k))];

output = filter(b,a,input);

input = output;

end

output = exp(-1i\*phi).\*output;



**Figure 1: Impulse response of the CD channel**

After computing the output of one such filter, this signal becomes the input for the following order 1 IIR filter, because they are cascaded. In the end, the output is multiplied by e-jΦ to obtain the desired phase response. The output of this function is shown in Figure 2.

**Figure 2: Output after CD equalization**

As it can be easily determined from Figure 2, the output after the equalization is nearly a Dirac impulse of amplitude 0.9892 at sample number 518.

After this the BER vs. SNR curve is plotted and is shown in Figure 3.



**c) Frequency sampling**

In the frequency sampling exercise we are required to compute the channel response for the first 256 samples.

Given the input values, I obtained α=0.7355. Then I created a function called impulse\_response\_channel which takes as input arguments α and the number of equally spaced frequency points in the interval [-π,π]. In the function, after computing the omega values (namely the frequency points where the transfer function is going to be estimated), I compute

H\_CD = exp(-1i\*alpha\*omega.^2)

After this, I apply the ***ifftshift*** to H\_CD because this function of Matlab works on the domain from [0,2π], and my frequency range for H\_CD is [-π,π]. Thus the domain between [0, π] coincides, whereas the interval from [π,2π] has to be shifted at the beginning of the H\_CD vector.

This being done, I then compute the impulse response h\_CD by taking the ***IFFT*** of H\_CD. After assuring that h\_CD is a column vector (through the command h\_CD = h\_CD(:)), I use the command ***fftshift*** applied to h\_CD for a similar reason with the one described above.

Finally, I plot the channel response of the channel by making use of the ***stem*** command. The result can be seen below.



**Channel response of the channel for the first 256 samples**