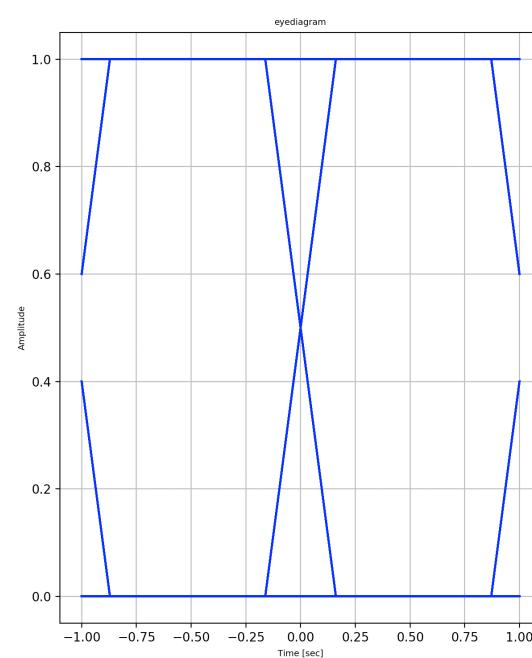
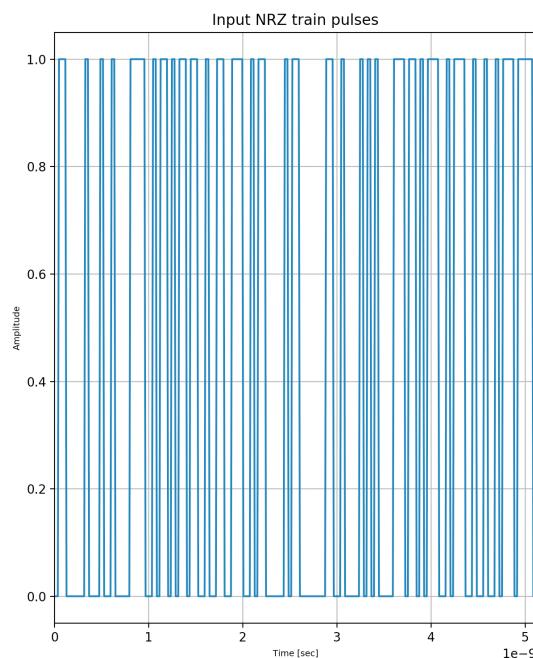


## Σκοπός του αλγορίθμου

Σκόπος της εργασίας είναι η ανάπτυξη αλγορίθμου για την αντιστάθμιση της χρωματικής διασποράς, μέσω ενός FIR φίλτρου με κατάλληλη κρουστική συνάρτηση ώστε να αναιρείται η χρωματική διασπορά.

### Βήμα 1: Δημιουργία διαμορφωμένου σήματος προς μετάδοση

- Αρχικά δημιουργούμε μια τυχαία ακολουθία ψηφίων-συμβόλων.
- Δημιουργούμε παλμούς χρησιμοποιώντας 16 δειγματα/σύμβολο και κωδικοποιώντας την παραχθείσα ακολουθία με Unipolar-NRZ. Οι παλμοί αυτοί αντιστοιχούν στην περιβάλλουσα του διαμορφωμένου σήματος, καθώς αυτή είναι που περνάει από την φωτοδίοδο.
- Unipolar: για bit '0' στέλνουμε 0 αλλιώς για bit '1' στέλνουμε +A V (διαλέγουμε A=1).
- NRZ: ο παλμός διαρκεί ολόκληρη την περίοδο του συμβόλου.
- Διαμορφώνουμε τους παλμούς ώστε να έχουν rise και fall time. (Οι παλμοί με την διαμόρφωση αυτή που έγινε μέσω συνέλιξης διαπλατύνθηκαν γι' αυτό κόβονται στο eyediagram).



Σχήμα 1: (α) Διαμορφωμένο κατά πλάτος σήμα με ρυθμό μετάδοσης συμβόλων  $25 * 10^9$  symbols/sec .  
(β) Το αντίστοιχο διάγραμμα οφθαλμού για υπέρθεση των παλμών στο διάστημα δύο  $T_{symb}$ .

## Βήμα 2: Εισαγωγή της χρωματικής διασποράς

Υπολογίζουμε το αρχικό σήμα στο πεδίο της συχνότητας με τη βοήθεια του μετασχηματισμού Fourier διακριτού χρόνου (DFT). Εάν  $s_n$  η αρχική ακολουθία παλμών μήκους  $M$  δειγμάτων, τότε στο πεδίο της συχνότητας προκύπτει από την σχέση:

$$S_k = \sum_{n=0}^{M-1} s_n e^{-j \frac{2\pi}{M} kn}, \quad k = 0, 1, \dots, M-1$$

Εφαρμόζοντας στο πεδίο του χρόνου στο σήμα μας, την χρωματική διασπορά  $H$  (για δεδομένο μήκος οπτικής ίνας) παίρνουμε το Hadamard product:

$$SS = H \odot S$$

Μετατρέπουμε το σήμα με τη χρωματική διασπορά στο πεδίο του χρόνου, με Inverse DFT:

$$ss_n = \frac{1}{M} \sum_{k=0}^{M-1} X_k e^{j 2\pi kn/M}, \quad n = 0, 1, \dots, M-1$$

Η χρωματική διασπορά περιγράφεται στο πεδίο του χρόνου από την κρουστική απόκριση:

$$h(z, t) = \sqrt{\frac{c}{jD\lambda^2 z}} e^{j \frac{\pi c}{D\lambda^2 z} t^2}$$

Και στο πεδίο των συχνοτήτων από τη συνάρτηση μεταφοράς:

$$H(z, \omega) = e^{-j \frac{\beta_2 z}{2} \omega^2} = e^{j \frac{D\lambda^2 z}{4\pi c} \omega^2}$$

Όπου :

$\beta_2$ : παράμετρος διασποράς της ταχύτητας της ομάδας

D: παράμετρος διασποράς

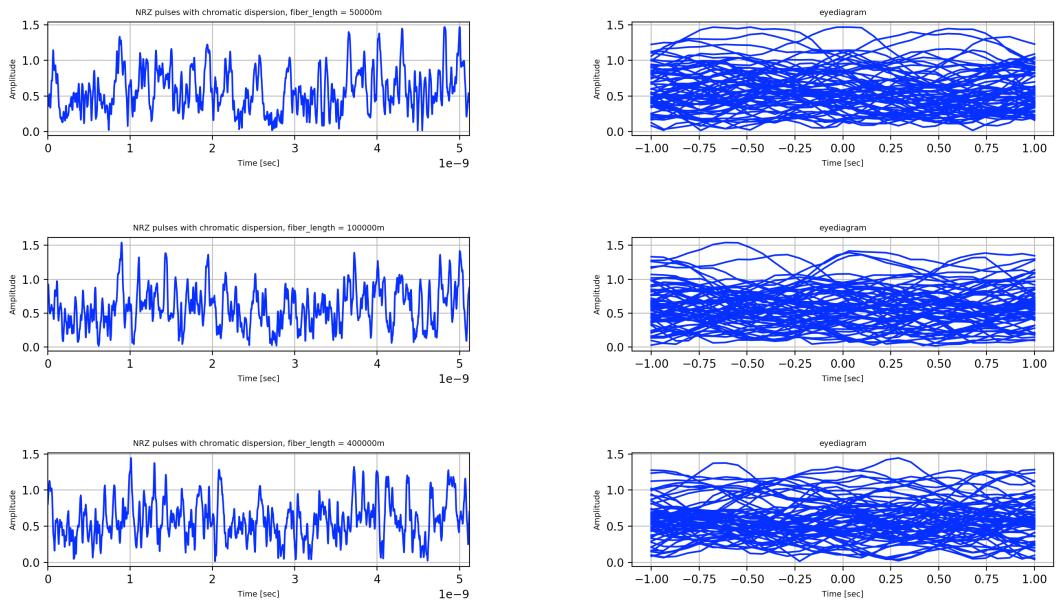
$\lambda$ : το μήκος κύματος

$\omega$ : η γωνιακή ταχύτητα

c: η ταχύτητα του φωτός στο κενό

z: το μήκος της οπτικής ίνας

Διατηρώντας τις παραπάνω παραμέτρους σταθερές και μεταβάλοντας μόνο το μήκος της οπτικής ίνας, z, παραμετρούμε ότι όσο αυξάνεται το μήκος της οπτικής ίνας τόσο περισσότερο αλλοιώνεται το αρχικό σήμα στο πεδίο του χρόνου.

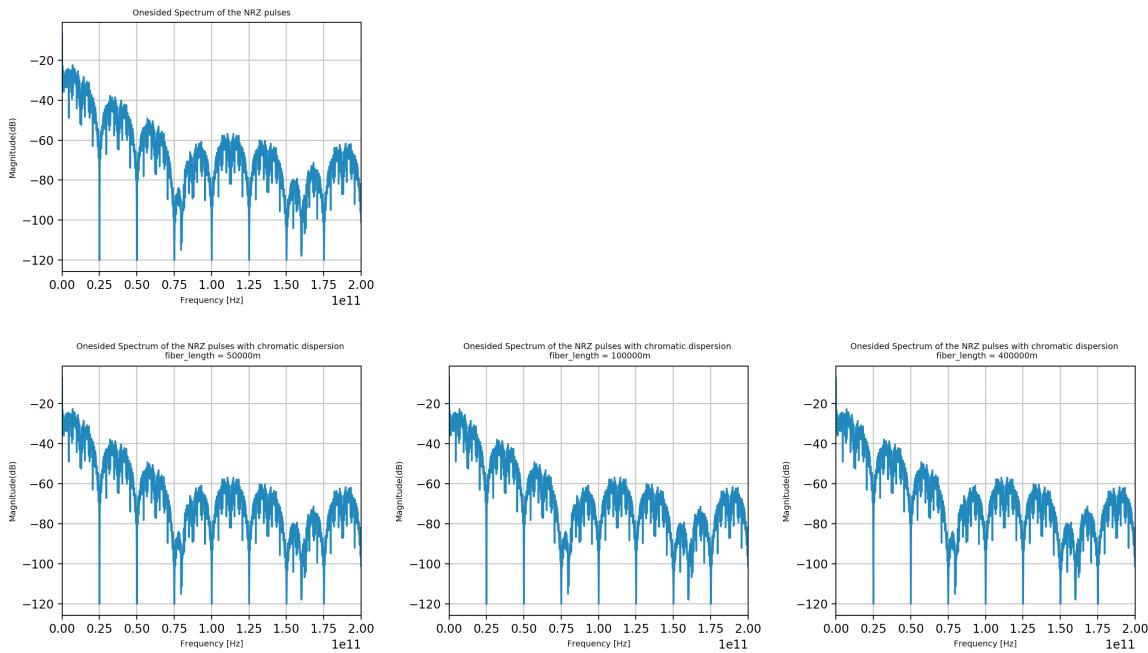


Σχήμα 2: (a) Τα σήμα που λαμβάνουμε από την οπτική ίνα στο πεδίο του χρόνου, το οποίο έχει επηρεαστεί από το φαινόμενο της χρωματικής διασποράς (β) Το αντίστοιχο διάγραμμα οφθαλμού.

Θα υπολογίσουμε την εκτίμηση της πυκνότητας φάσματος ισχύος (PSD) για το σήμα πρίν και μετά την μετάδοση, με τη βοήθεια του περιοδογράμματος.

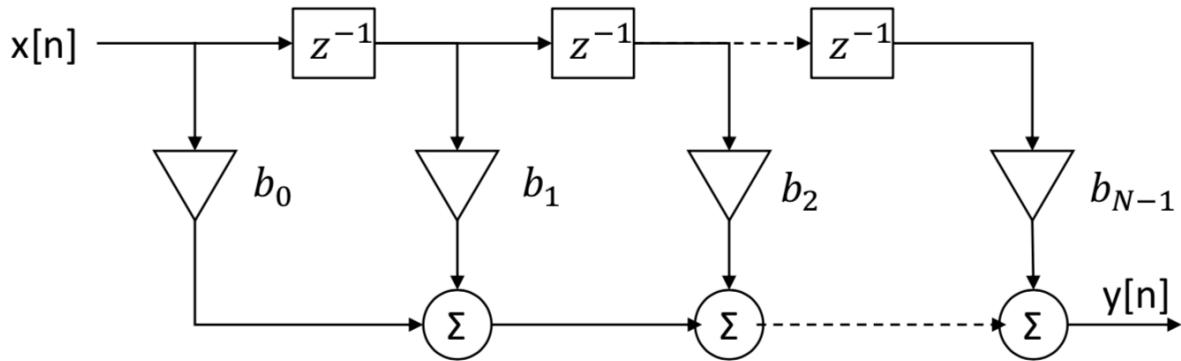
$$P = \frac{|S|^2}{M^2} \quad \text{και} \quad P_{CD} = \frac{|SS|^2}{M^2}$$

Όπως φαίνεται από το Σχήμα 3, το PSD δεν επηρεάζεται από τη χρωματική διασπορά, καθώς η συνάρτηση μεταφοράς της χρωματικής διασποράς έχει  $|H(z, \omega)| = 1$ , για κάθε τιμές των  $z$  και  $\omega$ .



Σχήμα 3: (a) Το φάσμα του σήματος προς μετάδοση. (β) Το φάσμα του σήματος που λήφθηκε μέσω της οπτικής ίνας.

### Βήμα 3: Εφαρμογή του φίλτρου για την αντιστάθμιση της χρωματικής διασποράς

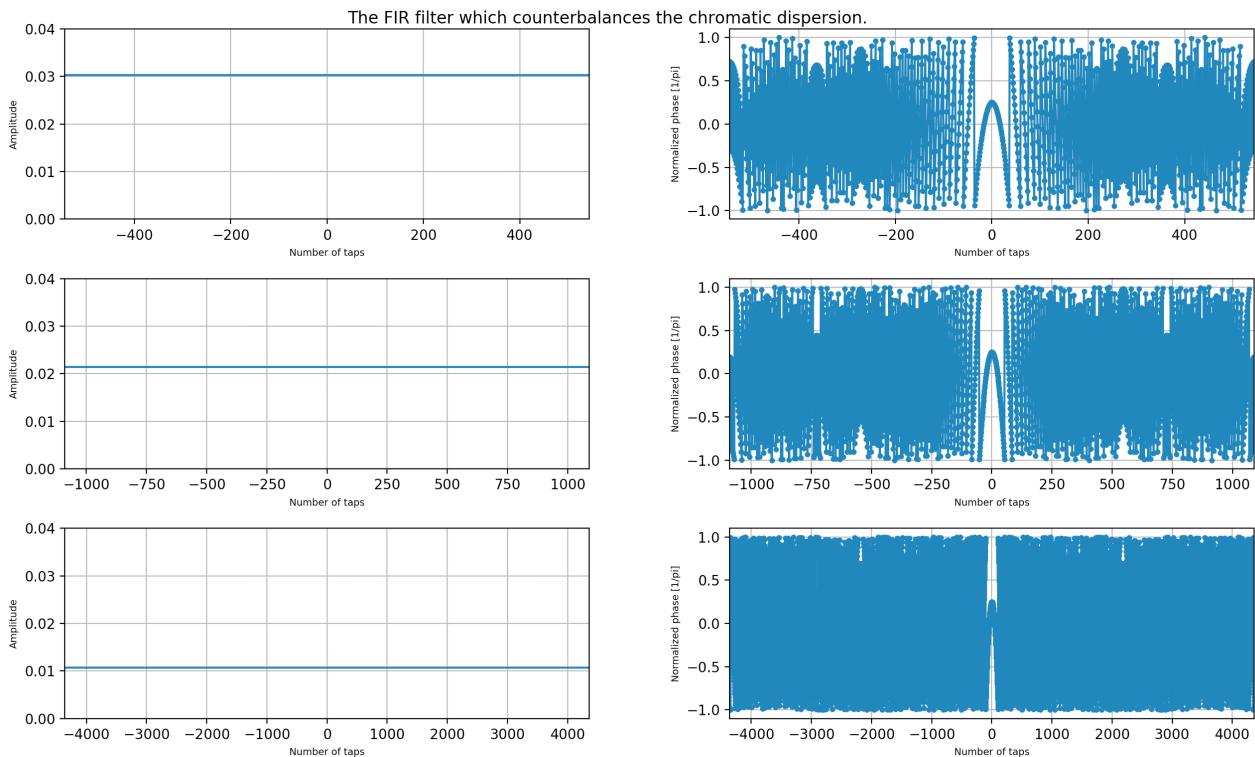


Υλοποιούμε ένα FIR φίλτρο με κρουστική συνάρτηση ίση με αυτή της χρωματικής διασποράς αλλά με αντίθετο πρόσημο της παραμέτρου της διασποράς  $D$ . Συγκεκριμένα, οι τιμές των βαρών (taps)  $b_k$  του FIR φίλτρου δίνονται από τη σχέση:

$$b_k = \sqrt{\frac{jcT_s^2}{D\lambda^2z}} e^{-j\frac{\pi cT_s^2}{D\lambda^2z}k^2}$$

όπου περιορίζουμε χρονικά την κρουστική συνάρτηση του φίλτρου, κατά το θεώρημα Nyquist για την αποφυγή του aliasing, σύμφωνα με την σχέση:

$$\left\lfloor -\frac{N}{2} \right\rfloor \leq k \leq \left\lfloor \frac{N}{2} \right\rfloor, \quad N = 2 * \left\lfloor \frac{|D|\lambda^2 z}{2cT_s^2} \right\rfloor + 1$$

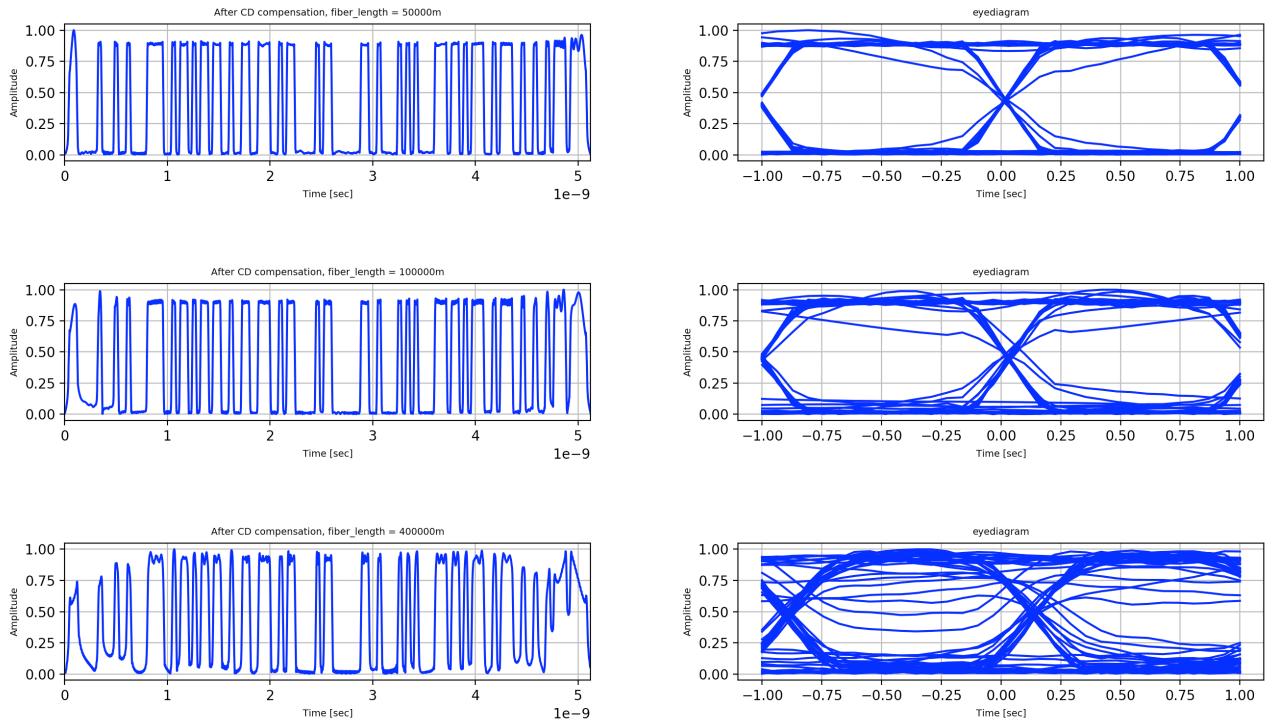


Σχήμα 4: (α)Το πλάτος του FIR φίλτρου. (β) Η κανονικοποιημένη φάση του FIR φίλτρου.

Εφαρμόζουμε το φίλτρο με συνέλιξη στο πεδίο του χρόνου στο σήμα που λάβαμε από την μετάδοση μέσω της οπτικής ίνας.

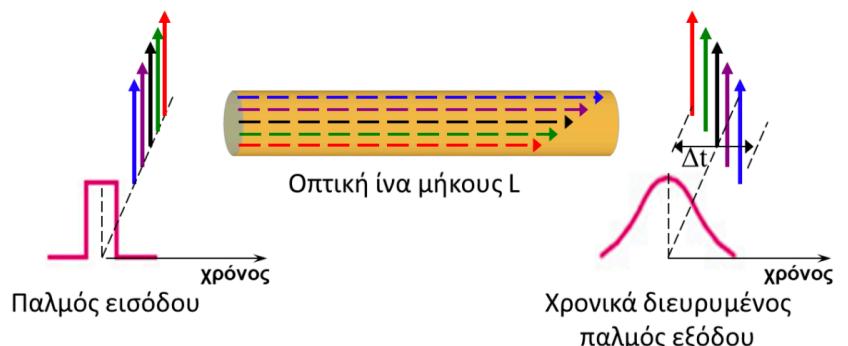
$$filtered_{ss}[n] = (ss * b)[n] = \sum_{m=0}^{N-1} ss[n-m]b[m], \quad n = N + M - 1$$

Όπου από αυτά τα  $N+M-1$  δείγματα κρατάμε μόνο τα  $M$ , σύμφωνα με το mode='same' της συνάρτησης `scipy.signal.convolve()`.



Σχήμα 5: (a) Το ανακτημένο σήμα, που πήραμε με την βοήθεια του φίλτρου αντιστάθμισης χρωματικής διασποράς, στο πεδίο του χρόνου. (β) Το αντίστοιχο διάγραμμα οφθαλμού.

Παρατηρούμε ότι όσο αυξάνεται το μήκος της οπτικής ίνας τόσο περισσότερο το φαινόμενο της χρωματικής διασποράς επηρεάζει το σήμα που στέλνουμε. Από το διάγραμμα οφθαλμού παρατηρούμε την ύπαρξη θορύβου που δεν εξαλειφθηκε από το φίλτρο, τόσο στη στάθμη του '1' όσο και στου '0' και ότι το "άνοιγμα" του οφθαλμού έχει μικρύνει καθώς η στάθμη του '1' εχει μικρότερο πλάτος, αφού η ενέργεια "απλώθηκε" αφού ο παλμός διαπλατύνθηκε. Επιπλέον, στο διάγραμμα οφθαλμού παρατηρούμαι την μετατόπιση των παλμών πιο δεξιά, δηλαδή καθυστέρηση στο πεδίο του χρόνου που οφείλεται στο ότι η χρωματική διασπορά εισάγει μια αλλαγή στη φάση. Τέλος παρατηρούμε χρονική διαπλάτυνση των παλμών που οφείλεται στο ότι η χρωματική διασπορά οδηγεί τις διαφορετικές φασματικές συνιστώσες να μεταδοθούν με ελαφρώς διαφορετικές ταχύτητες ομάδας.



## ΠΑΡΑΡΤΗΜΑ: Πηγαίος Κώδικας

```
import numpy as np
import matplotlib.pyplot as plt
import cmath
import math
import scipy.signal

def NRZgenerator (bitsInput ,length, samplesPerSymbol):
    # Input
    #   bitsInput: a sequence of '0' and '1'
    #   length: signal's length
    #   samplesPerSymbol
    # Output
    #   NRZ train pulses.
    outNRZ = np.zeros((length))
    for i in range (0,len(bitsInput)):
        for j in range(0,samplesPerSymbol):
            outNRZ[i*samplesPerSymbol+j] = bitsInput[i]
    return outNRZ

def eyediagram (signal,samplesPerSymbol,totalSamples,Tsymb,ax):
    t = np.linspace(-1,1,2*samplesPerSymbol)
    for i in range(0,totalSamples-2*samplesPerSymbol+1,2*samplesPerSymbol):
        plt.plot(t,signal[i:i+2*samplesPerSymbol], 'b')
    ax.set_title("eyediagram",fontsize=7)
    ax.set_xlabel("Time [sec]",fontsize=7)
    ax.set_ylabel("Amplitude",fontsize=7)
    ax.grid()

noOfSamples = 2048
noSymb = noOfSamples/16
z = np.array([150000,100000,400000])
N = np.zeros((len(z)))
D = 17/1000000.
lamda = 1550/1000000000.
symbolRate = 25 * 1000000000.
samplingRate = 16*symbolRate
T = 1/samplingRate
c = 29792458
F, I = np.modf(abs(D)*lamda*lamda*z/(2*c*T*T))
N = 2*I+1
N = N.astype(int)

squareNRZ = np.zeros((noOfSamples))
inputNRZ = np.zeros((noOfSamples))
S = np.zeros((noOfSamples), dtype=complex)
t = np.zeros((noOfSamples))
f = np.zeros((noOfSamples))
k = np.zeros((len(z),np.amax(N)))
b = np.zeros((len(z),np.amax(N)), dtype=complex)
H = np.zeros((len(z),noOfSamples), dtype=complex)
ss = np.zeros((len(z),noOfSamples), dtype=complex)
SS = np.zeros((len(z),noOfSamples), dtype=complex)
filtered_ss = np.zeros((len(z),noOfSamples), dtype=complex)

bitsSeq = np.random.randint(2, size = noSymb)          #generates a random sequence of bits
t = np.linspace(0,noOfSamples*T,num=noOfSamples)
f = np.linspace(-samplingRate/2, samplingRate/2, num = noOfSamples)
#Computation of input NRZ train pulses
squareNRZ = NRZgenerator(bitsSeq, noOfSamples, int(samplingRate/symbolRate))
win = [0,0,1,1,1,1,1,0,0]
inputNRZ = scipy.signal.convolve(np.pad(squareNRZ,(4,4), 'edge'), win, mode='valid')/sum(win)
#Fourier Transform of the input NRZ signal
S = np.fft.fft(inputNRZ)

for i in range (0,len(z)):
    #Computation of the FIR coefficients.
    F, I = math.modf(N[i]/2)
    k[i,:N[i]] = np.arange(-I, I+1)
    b[i,:N[i]] = cmath.sqrt(1j*c*T*T/(D*lamda*lamda*z[i]))*np.exp((-1j*math.pi*c*T*T*k[i,:N[i]]*k[i,:N[i]])/
(D*lamda*lamda*z[i]))
    #Computation of the chromatic dispersion
    H[i,:] = np.fft.ifftshift(np.exp(-1j*D*lamda*lamda*z[i]*((2*math.pi*f)**2)/(4*math.pi*c)))
    #Computation of the NRZ pulses with chromatic dispersion
    SS[i,:] = H[i,:]*S
    ss[i,:] = np.fft.ifft(SS[i,:])
    #Counterbalance the chromatic dispersion with FIR filter.
    filtered_ss[i,:] = scipy.signal.convolve(ss[i,:], b[i,:N[i]], mode='same')
```

```

#-----#
plt.figure(1)
plt.subplot(1, 2, 1)
plt.plot(t,inputNRZ)
plt.xlabel("Time [sec]",fontsize=7)
plt.ylabel("Amplitude",fontsize=7)
plt.xlim(0,noOfSamples*T)
plt.title('Input NRZ train pulses')
plt.grid()
ax = plt.subplot(1, 2, 2)
eyediagram(inputNRZ,16,noOfSamples,T*16,ax)
plt.tight_layout()
plt.show()

plt.figure(2)
for i in range(0,len(z)):
    plt.subplot(3,2,2*i+1)
    plt.plot(t,abs(ss[i,:,:]),'-b')
    plt.xlabel("Time [sec]",fontsize=7)
    plt.ylabel("Amplitude",fontsize=7)
    plt.xlim(0,noOfSamples*T)
    plt.title('NRZ pulses with chromatic dispersion, fiber_length = '+str(z[i])+'m',fontsize=7)
    plt.grid()
    ax = plt.subplot(3,2,2*i+2)
    eyediagram(abs(ss[i,:,:]),16,noOfSamples,T*16,ax)
plt.tight_layout()
plt.show()

plt.figure(3)
plt.subplot(2,3,1)
Sxx = S[:noOfSamples/2]*np.conj(S[:noOfSamples/2])/ (noOfSamples*noOfSamples)           #computing power with proper scaling
plt.plot(np.linspace(0, samplingRate/2, num = noOfSamples/2),10*np.log10(Sxx+le-12))
plt.xlabel("Frequency [Hz]",fontsize=7)
plt.ylabel("Magnitude(dB)",fontsize=7)
plt.xlim(0,samplingRate/2)
plt.title('Onesided Spectrum of the NRZ pulses',fontsize=7)
plt.grid()
for i in range(0,len(z)):
    plt.subplot(2,3,4+i)
    SSxx = SS[i,:noOfSamples/2]*np.conj(SS[i,:noOfSamples/2])/ (noOfSamples*noOfSamples) #computing power with proper
scaling
    plt.plot(np.linspace(0, samplingRate/2, num = noOfSamples/2),10*np.log10(SSxx+le-12))
    plt.xlabel("Frequency [Hz]",fontsize=7)
    plt.ylabel("Magnitude(dB)",fontsize=7)
    plt.xlim(0,samplingRate/2)
    plt.title('Onesided Spectrum of the NRZ pulses with chromatic dispersion \n fiber_length = '+str(z[i])+'m',fontsize=7)

    plt.grid()
plt.show()

plt.figure(4)
for i in range(0,len(z)):
    r = abs(b[i,:N[i]])
    phi = np.angle(b[i,:N[i]])

    plt.subplot(3,2,2*i+1)
    plt.plot(k[i,:N[i]],r)
    plt.ylim(0,0.04)
    plt.xlim(k[i,0],k[i,N[i]-1])
    plt.xlabel("Number of taps",fontsize=7)
    plt.ylabel("Amplitude",fontsize=7)
    plt.grid()
    plt.subplot(3,2,2*i+2)
    plt.plot(k[i,:N[i]],phi/math.pi,'.-')
    plt.xlim(k[i,0],k[i,N[i]-1])
    plt.xlabel("Number of taps",fontsize=7)
    plt.ylabel("Normalized phase [1/pi]",fontsize=7)
    plt.grid()
plt.suptitle("The FIR filter which counterbalances the chromatic dispersion.")
plt.tight_layout()
plt.show()

plt.figure(5)
for i in range(0,len(z)):
    amplMax = npamax(abs(filtered_ss[i,:])) #normalizes the filtered signal
    plt.subplot(3,2,2*i+1)
    plt.plot(t,abs(filtered_ss[i,:])/amplMax,'-b')
    plt.xlabel("Time [sec]",fontsize=7)
    plt.ylabel("Amplitude",fontsize=7)
    plt.xlim(0,noOfSamples*T)
    plt.title('After CD compensation, fiber_length = '+str(z[i])+'m',fontsize=7)
    plt.grid()
    ax = plt.subplot(3,2,2*i+2)
    eyediagram(abs(filtered_ss[i,:])/amplMax,16,noOfSamples,T*16,ax)
plt.tight_layout()
plt.show()

```