Μάθημα 10: Διαμόρφωση κατά πλάτος

Θωμάς Καμαλάκης

27 Σεπτεμβρίου 2023

1 Εισαγωγή

Στο μάθημα, θα αρχίσουμε να συζητούμε πως γίνεται η μετάδοση του σήματος χρησιμοποιώντας *διαμόρφωση πλάτους*. Η τεχνική αυτή χρησιμοποιείται αρκετά συχνά στην πράξη και συνίσταται στην αποτύπωση του ψηφιακού σήματος στο πλάτος ενός φυσικού μεγέθους, όπως η ηλεκτρική τάση ή το ηλεκτρική και μαγνητική ένταση του ηλεκτρομαγνητικού πεδίου. Ανάλογα με την περιοχή του φάσματος που χρησιμοποιείται, μπορούμε να διαχωρίσουμε τα συστήματα σε *βασικής ζώνης* (baseband), όταν το φάσμα του σήματος μας είναι συγκεντρωμένο κοντά στη συχνότητα $f = 0$ ή σε *ζωνοπεράτα* (passband) συστήματα όπου το φάσμα είναι "μαζεμένο" γύρω από μία κεντρική συχνότητα $f = f_c$ και το εύρος ζώνης του σήματος B είναι πολύ μικρότερο από το f_c, $B << f_c$. Στην εικόνα 1, δείχνουμε τυπικά φάσματα για ένα σήμα βασικής ζώνης και ένα ζωνοπερατό σήμα.

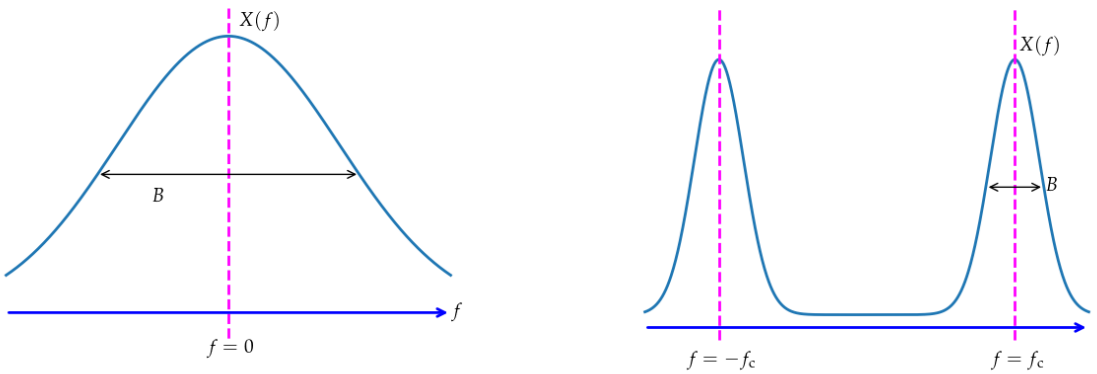
Έχει μία ιδιαίτερη σημασία να σχολιάσουμε την συμμετρία που παρατηρούμε στα φάσματα αυτά. Αν θυμηθούμε τον ορισμό του μετασχηματισμού Fourier

$$X(f) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{+\infty} x(t)\mathrm{e}^{-j2\pi ft}\mathrm{d}t \tag{1}$$

Αν θεωρήσουμε ένα πραγματικό σήμα $x(t)$, είχαμε δείξει ότι $X^*(-f) = X(-f)$, δηλαδή το $X(-f)$ είναι το μιγαδικό συζυγές του $X(f)$, οπότε θα έχουμε και

$$|X(f)| = |X(-f)| \tag{2}$$

Σύμφωνα με την παραπάνω σχέση, το μέτρο του φάσματος $|X(f)|$ θα είναι συμμετρικό γύρω από το $f = 0$ και για το λόγο αυτό έχουμε ζωγραφίσει τα φάσματα να είναι συμμετρικά. Παρατηρούμε στην εικόνα 1 την διαφορετική μορφή των φασμάτων. Τα σήματα βασικής ζώνης (εικόνα 1a) χρησιμοποιούνται συνήθως στην ενσύρματη μετάδοση. Για παράδειγμα, όταν μεταδίδουμε ένα σήμα πάνω από μία δισύρματη γραμμή μεταφοράς (όπως π.χ. το παλιό τηλεφωνικό καλώδιο) τότε συνήθως τα δεδομένα αποτυπώνονται σε ένα σήμα βασικής ζώνης. Όταν μεταδίδουμε το σήμα ασύρματα, θα πρέπει να χρησιμοποιήσουμε ζωνοπερατά σήματα. Π.χ. στο WiFi χρησιμοποιούμε συχνότητες κοντά στο $f_c = 2.4$ GHz και τα σήματα έχουν εύρος $B \cong 20$MHz για τα παλιότερα συστήματα και $B \cong 40$MHz για τα πιο σύγχρονα.



(a) (b)

Εικόνα 1: Παραδείγματα φασμάτων σημάτων: (a) βασικής ζώνης και (b) ζωνοπερατό.

Στην περίπτωση της διαμόρφωσης πλάτους βασικής ζώνης, εφαρμόζουμε το ψηφιακό σήμα $x(t)$ απευθείας στο κανάλι, π.χ. ως τάση στην δισύρματη γραμμή μεταφοράς. Όπως είδαμε και στα προηγούμενα το σήμα αυτό θα έχει τη μορφή:

$$x(t) = \sum_{k=0}^{N} x_k p(t - kT_S) \tag{3}$$

Στην (3), το x_k είναι το πλάτος του σήματος που αντιστοιχεί στο $k^{οστό}$ σύμβολο προς μετάδοση, T_S είναι η διάρκεια του κάθε συμβόλου. Στις επόμενες παραγράφους θα εξετάσουμε λίγο πιο συστηματικά την μετάδοση της πληροφορίας με αυτό τον τρόπο που ονομάζεται *Διαμόρφωση Παλμού κατά Πλάτος* (pulse amplitude modulation - PAM) στη βασική ζώνη αλλά και με την μετάδοση σε υψηλότερες συχνότητες (ζωνοπερατά σήματα).

2 Κωδικοποίηση

Η πληροφορία που θέλουμε να μεταδώσουμε συνήθως έχει την μορφή δυαδικών bit b_m τα οποία μπορούν να πάρουν τιμές μηδέν ή ένα. Οπότε τα συστήματα μας θα πρέπει να μεταδώσουν μηνύματα της μορφής:

$$\mathbf{b} = \{b_0, b_1, \cdots, b_p, \cdots\} \tag{4}$$

Θα πρέπει με κάποιον τρόπο να αντιστοιχήσουμε στα $\mathbf{b}$ ένα σύνολο από σύμβολα

$$\mathbf{x} = \{x_0, x_1, \cdots, x_k, \cdots\} \tag{5}$$

Ένας προφανής τρόπος να γίνει αυτή η αντιστοίχιση είναι κάθε bit b_p να αντιστοιχηθεί σε ένα ακριβώς σύμβολο x_p, για παράδειγμα εάν το $b_p = 1$ να θέσουμε $x_p = 1$ και εάν $b_p = 0$ να θέσουμε $x_p = -1$,

$$\begin{aligned} \{0\} &\rightarrow -1 \\ \{1\} &\rightarrow +1 \end{aligned} \tag{6}$$

Έτσι αντιστοιχούμε *ένα bit ανά σύμβολο*. Ωστόσο υπάρχουν και άλλες επιλογές που συνήθως οδηγούν σε καλύτερα αποτελέσματα. Θα μπορούσαμε να ομαδοποιήσουμε τα bit σε ομάδες των m διαδοχικών bit. Έτσι π.χ. αν $m = 3$, η πρώτη ομάδα είναι η $\{b_0, b_1, b_2\}$, η δεύτερη ομάδα είναι η $\{b_3, b_4, b_5\}$ κ.ο.κ. Αν θεωρήσουμε m bits υπάρχουν

$$M = 2^m \tag{7}$$

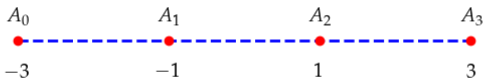
διαφορετικοί συνδυασμοί που μπορούν να προκύψουν. Για παράδειγμα αν $m = 2$ τότε οι πιθανοί συνδυασμοί μίας ομάδας δύο διαδοχικών bit είναι $\{00, 01, 10, 11\}$ οπότε έχουμε $M = 4 = 2^2$ διαφορετικούς συνδυασμούς. Θα μπορούσαμε να αναθέσουμε σε κάθε συνδυασμό ένα διαφορετικό σύμβολο. Για παράδειγμα,

$$\begin{aligned} \{00\} &\rightarrow -3 \\ \{01\} &\rightarrow -1 \\ \{10\} &\rightarrow +1 \\ \{11\} &\rightarrow +3 \end{aligned} \tag{8}$$

Η αντιστοίχιση συνδυασμών bit σε πλάτη συμβόλων όπως στην (6) και στην (8), ονομάζεται *κωδικοποίηση*. Η κωδικοποιήσεις αυτές έχουν το χαρακτηριστικό ότι το σύνολο των πιθανών τιμών A_n που μπορεί να λάβει ένα σύμβολο είναι ισαπέχουσες. Έτσι για παράδειγμα στην (8) έχουμε $M = 4$ πιθανές τιμές $A_0 = -3$, $A_1 = -1$, $A_2 = +1$ και $A_3 = +3$ για όλα τα x_k. Στην εικόνα 2 δείχνουμε τα 4 πιθανά σύμβολα από όπου εύκολα προκύπτει ότι τα διαδοχικά σύμβολα είναι ισαπέχοντα. Επίσης τα πλάτη A_k είναι συμμετρικά γύρω από το μηδέν.

Υπάρχει ωστόσο ένα πρόβλημα με την κωδικοποίηση στην (8): τα σφάλματα στις επικοινωνίες που μας οδηγήσουν σε εσφαλμένη αποκωδικοποίηση εξαιτίας του θορύβου, όπως θα δούμε και παρακάτω. Έτσι εμείς μπορεί να θέλουμε να μεταδώσουμε το $\{00\}$ οπότε στέλνουμε το πλάτος -3 αλλά ο θόρυβος που προστίθεται μπορεί να μας φέρει πιο κοντά στο -1 και ο δέκτης να θεωρήσει ότι έλαβε αυτό το σύμβολο και όχι το -3. Οπότε θα αποκωδικοποιήσει τα bit $\{01\}$ και όχι το $\{00\}$ και θα λάβει χώρα ένα σφάλμα καθώς το δεύτερο bit θα αποκωδικοποιηθεί ως 1 και όχι ως 0. Τα πράγματα είναι χειρότερα όταν θέλουμε να εκπέμψουμε το $\{01\}$ οπότε στέλνουμε το -1, ο θόρυβος μας στέλνει πιο κοντά στο $+1$ και ο δέκτης εσφαλμένα καταλαβαίνει ότι στείλαμε το $\{10\}$. Σε αυτή την περίπτωση έχουμε δύο σφάλματα στα bit του δέκτη: το πρώτο bit αντί για 0 αποκωδικοποιείται ως 1 ενώ για το δεύτερο bit αποκωδικοποιείται ως 0 ενώ στέλνουμε 1.

Για να έχουμε μικρό αριθμό σφαλμάτων, θα πρέπει να ελαττώσουμε τον αριθμό των εσφαλμένων bit όταν ο θόρυβος μας "πετάει" σε γειτονικά σύμβολα. Για να το κάνουμε αυτό θα πρέπει να επιλέξουμε μία εναλλακτική κωδικοποίηση όπως φαίνεται παρακάτω



Εικόνα 2: Τα 4 πιθανά πλάτη των συμβόλων του PAM οταν $M = 4$.

$$\begin{array}{l} \{00\} \rightarrow -3 \\ \{01\} \rightarrow -1 \\ \{11\} \rightarrow +1 \\ \{10\} \rightarrow +3 \end{array} \tag{9}$$

Στην κωδικοποίηση (9) έχουμε αλλάξει την σειρά των δύο τελευταίων συνδυασμών bit στην (8) από $\{10\}, \{11\}$ σε $\{11\}, \{10\}$. Με τον τρόπο αυτό είναι φανερό ότι πλέον δεν θα έχουμε δύο σφάλματα bit όταν πηγαίνουμε από το -1 στο $+1$ αλλά ένα. Επίσης αν κοιτάξουμε προσεκτικά, σε καμία περίπτωση δεν έχουμε σφάλματα όταν πηγαίνουμε σε γειτονικό σύμβολο. Π.χ. αν πάμε από το -3 στο -1 η ακολουθία bit που αποκωδικοποιούμε θα είναι $\{01\}$ ενώ θα έπρεπε να είναι $\{00\}$, οπότε πάλι κάνουμε σφάλμα σε ένα μόνο bit.

Υπάρχει ένας συστηματικός τρόπος να επιλέγουμε την κατάλληλη κωδικοποίηση που οδηγεί σε ένα μόνο σφάλμα bit όταν πηγαίνουμε σε γειτονικό σύμβολο, ο κώδικας *Gray*. Για να φτιάξουμε τον κώδικα Gray ακολουθούμε έναν αναδρομικό τρόπο κατασκευής σύμφωνα με τον οποίο ο κώδικας Gray $\mathbf{G}_m$ με m bits προκύπτει από τον κώδικα Gray $\mathbf{G}_{m-1}$ με $m-1$ bits. Ας θεωρήσουμε για παράδειγμα τον κώδικα Gray $\mathbf{G}_2$ στην (9)

$$\mathbf{G}_2 = [00, 01, 11, 10] \tag{10}$$

Στην ουσία το $\mathbf{G}_2$ περιέχει τους διαδοχικούς συνδυασμούς των συνδυασμών της κωδικοποίησης στην (10). Στην γενική περίπτωση το $\mathbf{G}_m$ είναι ένα διάνυσμα που περιέχει όλους τους συνδυασμούς με την σειρά που ανατίθενται σε διαδοχικά σύμβολα. Αν ξέρουμε το $\mathbf{G}_m$ επομένως εύκολα αναθέτουμε συνδυασμούς σε διαδοχικά σύμβολα όπως στην (10).

Για να φτιάξουμε το $\mathbf{G}_3$ από το $\mathbf{G}_2$ προσθέτουμε καταρχήν ένα 0 μπροστά από τους συνδυασμούς του $\mathbf{G}_2$. Έχουμε επομένως τους συνδυασμούς:

$$[000, 001, 011, 010] \tag{11}$$

Προς το παρόν έχουμε 4 συνδυασμούς ενώ χρειαζόμαστε $2^3 = 8$ συνδυασμούς. Για να τους φτιάξουμε παίρνουμε τους συνδυασμούς του $\mathbf{G}_2$, τους αντιστρέφουμε, δηλαδή θεωρούμε με την σειρά $[10, 11, 01, 00]$ και προσθέτουμε το 1 μπροστά οπότε φτιάχνουμε την ακολουθία:

$$[110, 111, 101, 100] \tag{12}$$

Ενώνοντας τις δύο ακολουθίες έχουμε:

$$[000, 001, 011, 010, 110, 111, 101, 100] \tag{13}$$

που αν προσέξουμε έχει την επιθυμητή ιδιότητα, δηλαδή οι διαδοχικοί συνδυασμοί απέχουν κατά ένα ακριβώς bit. Θέτουμε επομένως

$$\mathbf{G}_3 = [000, 001, 011, 010, 110, 111, 101, 100] \tag{14}$$

```
['0000', '0001', '0011', '0010', '0110', '0111', '0101', '0100', '1100',
'1101', '1111', '1110', '1010', '1011', '1001', '1000']
```

Εικόνα 3: Οι συνδυασμοί του κώδικα Gray για $m = 4$.

```python
def gray_code(m):

    if m==1:
        g = ['0','1']

    elif m>1:
        gs = gray_code(m-1)
        gsr = gs[::-1]
        gs0 = ['0' + x for x in gs]
        gs1 = ['1' + x for x in gsr]
        g= gs0 + gs1
    return g

g = gray_code(4)
print(g)
```

Listing 1: Υπολογισμός κώδικα Gray

Στο listing 1 δείχνουμε τον τρόπο υπολογισμού του κώδικα Gray χρησιμοποιώντας την αναδρομική σχέση που συζητήσαμε. Στην τετριμμένη περίπτωση όπου $m = 1$, ο κώδικας Gray είναι απλά το $\mathbf{G}_1 = [0, 1]$. Διαφορετικά εάν $m > 1$ τότε υπολογίζουμε τους συνδυασμούς του $\mathbf{G}_m$ από τους συνδυασμούς του $\mathbf{G}_{m-1}$ στους οποίους προσθέτουμε μπροστά το 0 ενώ προσθέτουμε το 1 στους συνδυασμούς του $\mathbf{G}_{m-1}$ σε ανεστραμμένη όμως σειρά.

Στην εικόνα 3 δείχνουμε το αποτέλεσμα για $m = 4$. Οι διαδοχικοί συνδυασμοί που παράγονται είναι φανερό ότι διαφέρουν κατά ένα ακριβώς bit.

3 Αστερισμοί Συμβόλων

Η κωδικοποίηση Gray αφορά την αντιστοίχιση των συνδυασμών bit σε πλάτη συμβόλων A_k. Στην ενότητα αυτή θα ασχοληθούμε λίγο περισσότερο με τα A_k. Το σύνολο των δυνατών A_k συχνά ονομάζεται *αστερισμός συμβόλων*. Στην περίπτωση του PAM με $m = 2$ bit ανά σύμβολο έχουμε όπως είδαμε, $M = 2^m = 4$ δυνατές τιμές των A_k και η εικόνα 2 δείχνει μία επιλογή για τις τιμές των A_k η οποία είναι συμμετρική γύρω από το 0 και τα διαδοχικά πλάτη ισαπέχουν, δηλαδή θα έχουμε πάντα:

$$A_{k+1} - A_k = 2\beta \tag{15}$$

Το 2β είναι η απόσταση μεταξύ δύο διαδοχικών συμβόλων Ένα σύστημα PAM με πλήθος συμβόλων M ονομάζεται M-PAM και για το 4-PAM στην εικόνα 2 έχουμε $\beta = 1$. Στην γενικότερη περίπτωση όπου το πλήθος των συμβόλων είναι $M = 2^m$ μπορούμε να καταλήξουμε σε μία σχέση για τα A_k στην περίπτωση του M-PAM. Από την (15) προκύπτει ότι:

$$A_k = A_0 + 2k\beta \tag{16}$$

Από την παραπάνω σχέση και αν υποθέσουμε ότι μετρούμε τα A_k από $k = 0$ μέχρι το $k = M - 1$ προκύπτει ότι $A_{M-1} = A_0 + 2\beta(M - 1)$. Εφόσον τα A_k είναι συμμετρικά ως προς το 0 θα πρέπει $A_{M-1} = -A_0$ και επομένως θα έχουμε

$$-A_0 = A_0 + 2\beta(M - 1) \Rightarrow A_0 = -\beta(M - 1) \tag{17}$$

και χρησιμοποιώντας την (16) προκύπτει ότι

$$A_k = -\beta(M - 1) + 2k\beta = (2k - M + 1)\beta \tag{18}$$

Η (18) μας παρέχει μία σχέση για να υπολογίζουμε τα A_k για το M-PAM στην περίπτωση όπου τα πλάτη είναι συμμετρικά και διαδοχικά ισαπέχοντα.

3.1 Η κλάση `constellation`

Θα χρησιμοποιήσουμε μία γενική κλάση, την `constellation` για να περιγράψουμε τους αστερισμούς των συμβόλων σε ένα πιο αφαιρετικό επίπεδο και στη συνέχεια θα υλοποιήσουμε την `pam_constellation` η οποία κληρονομεί την `constellation` και περιγράφει τον αστερισμό του M-PAM. Στο listing 2 δείχνουμε την κλάση `constellation`.

```
class constellation:
    """
    Generic constellation class
    """
    def __init__(self, title = None):
       self.map = {}
       self.bits = []
       self.bits_str = []
       self.symbols = []
       self.title = title
       self.m = None

    def avg_power(self):
        """
        Average power of constellation symbols
        """
        return np.mean( np.abs(self.symbols) ** 2.0)

    def plot(self, figure_no = None, plot_type = 'o'):
        """
        Plot constellation symbols
        """
        if figure_no is None:
            plt.figure()
        else:
            plt.figure(figure_no)

        cr = np.real(self.symbols)
        ci = np.imag(self.symbols)
        plt.plot(cr, ci, plot_type)
        plt.xlabel('Real')
        plt.ylabel('Imag')

        if self.title is not None:
            plt.title(self.title)

    def plot_map( self, figure_no = None, disp_x = 0.0, disp_y = 0.0,
                  rotation = 90, plot_type = 'bo', axis_equal = True):
        """
        Plot constellation symbols and corresponding bit sequences
        """
        self.plot( figure_no = figure_no, plot_type = plot_type)

        for i, bits in enumerate(self.bits_str):
            symbol = self.symbols[i]
            bits_str = self.bits_str[i]
            plt.text( np.real(symbol) + disp_x, np.imag(symbol) + disp_y, bits_str,
                      rotation = rotation)

        if self.title is not None:
            plt.title(self.title)

        if axis_equal:
            plt.axis('equal')

    def set_symbols( self, symbols ):
        """
        Set the constellation symbols
        """
        self.symbols = symbols

    def set_gray_bits( self, m ):
        """
        Map symbols to bits using Gray coding
        """
        g = gray_code( m )
        self.m = m

        for i, cw in enumerate(g):
            self.bits_str.append(cw)
            self.bits.append( str_to_bitsarray( cw ) )
            self.map[ cw ] = self.symbols[ i ]

    def bits_to_symbols( self, bits, return_groups = False ):
        """
        Convert a stream of bits to symbols
```

```
        """

        symbols = []
        bitgroups = []
        i = 0
        j = 0

        if not isinstance(bits, str):
            bits = array_to_str(bits)

        while i < len(bits):

            key = bits[ i : i + self.m ]
            bitgroups.append( key )
            symbols.append( self.map[ key ] )
            i += self.m
            j += 1

        if not return_groups:
            return np.array(symbols)
        else:
            return np.array(symbols), bitgroups

    def find_closest( self, sample ):
        """
        find symbol closest to value
        """
        return np.abs( self.symbols - sample ).argmin()

    def decode( self, sample ):
        """
        decode symbol based on value
        """

        i = self.find_closest( sample )
        return [self.symbols[i], self.bits[i], self.bits_str[i] ]

    def symbols_to_bits(self, samples):
        bits = ''
        for sample in samples:
            _, _, bits_str = self.decode( sample )
            bits += bits_str
        return bits
```

Listing 2: Η κλάση `constellation`

Στην `__init__` αρχικοποιούμε την κλάση, θέτοντας κάποιες αρχικές τιμές σε γνωρίσματα που θα χρειαστούμε παρακάτω. Το `symbols` περιέχει τα πλάτη των συμβόλων, το `bits` τις ακολουθίες bits που αντιστοιχούν στα σύμβολα, το `bits_str` περιέχει τις ίδιες ακολουθίες αλλά σε μορφή `string`, το `title` περιέχει το όνομα του αστερισμού, ενώ το `m` αποθηκεύει τον αριθμό των bit ανά σύμβολο m. Το `map` θα χρησιμοποιηθεί για να αποθηκεύει την αντιστοίχιση των συνδυασμών bit σε σύμβολα όπως π.χ. στην εξίσωση 8.

Η κλάση `constellation` όπως θα καταλάβατε είναι μία γενική κλάση καθώς δεν γίνεται κάποια ανάθεση αρχικών πλατών σε κάποιο γνώρισμα όπως αυτή που περιγράφεται στην (18). Τις ιδιαιτερότητες του M-PAM θα τις υλοποιήσουμε σε μία δεύτερη κλάση `pam_constellation` που θα κληρονομεί μεθόδους από την `constellation`. Όταν θα υλοποιήσουμε κλάσεις που θα περιγράφουν άλλα συστήματα διαμόρφωσης τότε και αυτές θα κληρονομούν μεθόδους της `constellation` που είναι κοινές και επομένως δεν χρειάζεται να τις υλοποιήσουμε για κάθε κλάση. Ας αναφερθούμε συνοπτικά στα γνωρίσματα και τις μεθόδους της κλάσης `constellation`. Σε ότι αφορά τα γνωρίσματα:

- το `symbols` είναι θα περιέχει τα πλάτη των συμβόλων A_k. Για την περίπτωση του $4-PAM$, σύμφωνα και με την εικόνα 2, τα στοιχεία του πίνακα θα είναι -3, -1, 1 και 3.
- το `bits` είναι μία *λίστα* που περιέχει ως στοιχεία arrays από bits που αντιστοιχούν σε κάθε ένα από τα πλάτη των συμβόλων. Έτσι π.χ. για $M = 4$, το `bits` περιέχει τα στοιχεία $(0, 0)$, $(0, 1)$, $(1, 1)$ και $(1, 0)$.
- το `bits_str` είναι μία *λίστα* που περιέχει ως στοιχεία `stings` που περιέχουν τα bits που αντιστοιχούν σε κάθε ένα από τα πλάτη των συμβόλων. Έτσι π.χ. για $M = 4$, το `bits_str` περιέχει τα strings "00", "01", "11" και "10".
- το `map` είναι ένα λεξικό που θα μας βοηθήσει να αντιστοιχίσουμε συνδυασμούς bit σε σύμβολα. Έτσι για παράδειγμα στην περίπτωση όπου έχουμε 4-PAM με τα πλάτη που περιγράφονται στην εικόνα 2, το `map` θα

είναι:

$$00 \to -3$$
$$01 \to -1$$
$$11 \to +1$$
$$10 \to +3$$

Σε ότι αφορά τις μεθόδους εδώ κάνουμε μία σύντομη αναφορά και στις παρακάτω ενότητες θα δούμε περισσότερες λεπτομέρειες για τη φιλοσοφία πίσω από την υλοποίηση τους:

- `avg_power`: Υπολογίζει την μέση ισχύ των συμβόλων.
- `plot`: Κάνει την γραφική παράσταση των πλατών των συμβόλων που υπάρχουν στο γνώρισμα `symbols`.
- `plot_map`: Κάνει την γραφική παράσταση των παραπάνω συμβόλων αναγράφοντας όμως και την ομάδα bit που αντιστοιχεί στο κάθε ένα.
- `set_symbols`: στην ουσία καταχωρεί κάποιες τιμές στο γνώρισμα `symbols`
- `set_gray_bits`: Εδώ υπολογίζουμε τα bits ενός κώδικα Gray του οποίου το μήκος των λέξεων m το περνάμε ως παράμετρο στην μέθοδο. Αφού κατασκευάσουμε τον κώδικα Gray $\mathbf{G}_m$ τότε τα αποθηκεύουμε διαδοχικά στα γνωρίσματα `bits` ως arrays και `bits_str` ως stings. Επίσης κατασκευάζουμε ανάλογα το λεξικό του γνωρίσματος `map`.
- `bits_to_symbols`: Η μέθοδος αυτή δέχεται ως είσοδο μία σειρά από bit και επιστρέφει τα σύμβολα στα οποία αντιστοιχεί σύμφωνα με την αντιστοίχιση που έχει καθοριστεί στο γνώρισμα `map`.
- `find_closest`: Μας επιστρέφει το δείκτη του συμβόλου του οποίου η τιμή είναι πιο κοντά στην τιμή `sample`. Χρησιμοποιείται για την αποκωδικοποίηση του μηνύματος.
- `decode`: Αποκωδικοποιεί το δείγμα `sample` επιστρέφοντας τόσο το πιο κοντινό σύμβολο όσο και τα αντίστοιχα bits.

3.2 Η κλάση `pam_constellation`

```
438 class pam_constellation(constellation):
439     """
440     PAM constellation class
441     """
442
443     def __init__(self, M, beta = 1, title = None, SNRbdB = None):
444         super().__init__(title = title)
445
446         self.M = M
447         self.m = np.log2(M).astype(int)
448         self.SNRbdB = SNRbdB
449
450         symbols = np.zeros( M )
451         for i in range( M ):
452             symbols [ i ] = 2 * i - M + 1
453
454         self.set_symbols( symbols )
455         self.set_gray_bits( self.m )
456
457     def ser(self):
458         SNRb = 10 ** ( self.SNRbdB / 10)
459         q = 6 * SNRb * self.m / (self.M ** 2.0 - 1)
460         return 2 * (self.M-1) / self.M * Q ( np.sqrt(q) )
461
462     def ber(self):
463         return self.ser() / self.m
```

Listing 3: Η κλάση `pam_constellation`

Θα υλοποιήσουμε παράγωγη κλάση, την `pam_constellation` για να περιγράψουμε πιο συγκεκριμένα την διαμόρφωση PAM. Στο listing 3 δείχνουμε πως έχει υλοποιηθεί αυτή η κλάση. Βλέπουμε πως η `pam_constellation` κληρονομεί όλα τα γνωρίσματα και τις μεθόδους της `constellation` και διαφοροποιείται την `__init__` όπου αφού αρχικά καλείται η `__init__` του γονέα (δηλαδή της κλάσης `constellation` και αρχικοποιούνται τα γνωρίσματα που είδαμε στην ενότητα 3.1. Στη συνέχεια αρχικοποιούνται διάφορα γνωρίσματα: το `M` και το `m` είναι το πλήθος

των συμβόλων και ο αριθμός των bit/σύμβολο αντίστοιχα και διέπονται από την (7). Επίσης αρχικοποιούνται και τα σύμβολα στο γνώρισμα `symbols` καθότι γνωρίζουμε ότι δίνονται από την (18) και φτιάχνεται η αντιστοίχιση συμβόλων και ομάδων bits με την `set_gray_bits`.

Υπάρχουν και άλλες δύο μέθοδοι της `pam_constellation`, οι `ser` και `ber` που θα συζητήσουμε παρακάτω.

4 Μέση ισχύς του σήματος

Ένα πράγμα που σίγουρα μας ενδιαφέρει είναι η αναμενόμενη τιμή $\mathbb{E}\left\{x_k^2\right\}$ των πλατών x_k των συμβόλων στην (3). Θυμηθείτε ότι η PSD $S_X(f)$ ενός ψηφιακού σήματος δίνεται από την σχέση:

$$S_X(f) = \frac{\sigma_x^2}{T_\mathrm{S}}|P(f)|^{2}\text{..} \tag{19}$$

όπου $\sigma_x^2 = \mathbb{E}\left\{x_k^2\right\}$. Αν θεωρήσουμε ότι τα x_k λαμβάνουν τιμές από τα A_p με την ίδια πιθανότητα τότε θα έχουμε:

$$\Pr\left\{x_k = A_p\right\} = \frac{1}{M} \tag{20}$$

Για παράδειγμα στην περίπτωση του 4-PAM της εικόνας 2 θα έχουμε:

$$\Pr\left\{x_k = -3\right\} = \Pr\left\{x_k = -1\right\} = \Pr\left\{x_k = +1\right\} = \Pr\left\{x_k = +3\right\} = \frac{1}{4} \tag{21}$$

Η μέση ισχύς των συμβόλων επομένως γράφεται ως εξής:

$$\mathbb{E}\left\{x_k^2\right\} = \sum_{p=0}^{M-1} A_p^2 \Pr\left\{x_k = A_p\right\} = \frac{1}{M}\sum_{p=0}^{M-1} A_p^2 \tag{22}$$

Παρατηρείστε ότι η (22) είναι ακριβώς ο τρόπος που χρησιμοποιεί η μέθοδος `avg_power` της κλάσης `constellation` στο listing 2. Στην περίπτωση των πλατών (18), θα έχουμε επομένως:

$$\mathbb{E}\left\{x_k^2\right\} = \frac{1}{M}\sum_{p=0}^{M-1} A_p^2 = \frac{\beta^2}{M}\sum_{p=0}^{M-1}(2k - M + 1)^2 = \frac{\beta^2}{M}\sum_{p=0}^{M-1}(4k^2 - 4Mk + 1) \tag{23}$$

Το άθροισμα στο δεξιό μέρος της (23) γράφεται ως εξής:

$$\sum_{p=0}^{M-1}(4k^2 - 4Mk + 1) = 4\sum_{p=0}^{M-1} k^2 + 4M\sum_{p=0}^{M-1} k + M \tag{24}$$

Τα παραπάνω αθροίσματα υπολογίζονται από γνωστούς τύπους:

$$\sum_{q=0}^{Q} q = \frac{Q(Q+1)}{2} \tag{25}$$

$$\sum_{q=0}^{Q} q^2 = \frac{Q(Q+1)(2Q+1)}{6} \tag{26}$$

Συνδυάζοντας τις παραπάνω εξισώσεις, μπορούμε να δείξουμε ότι:

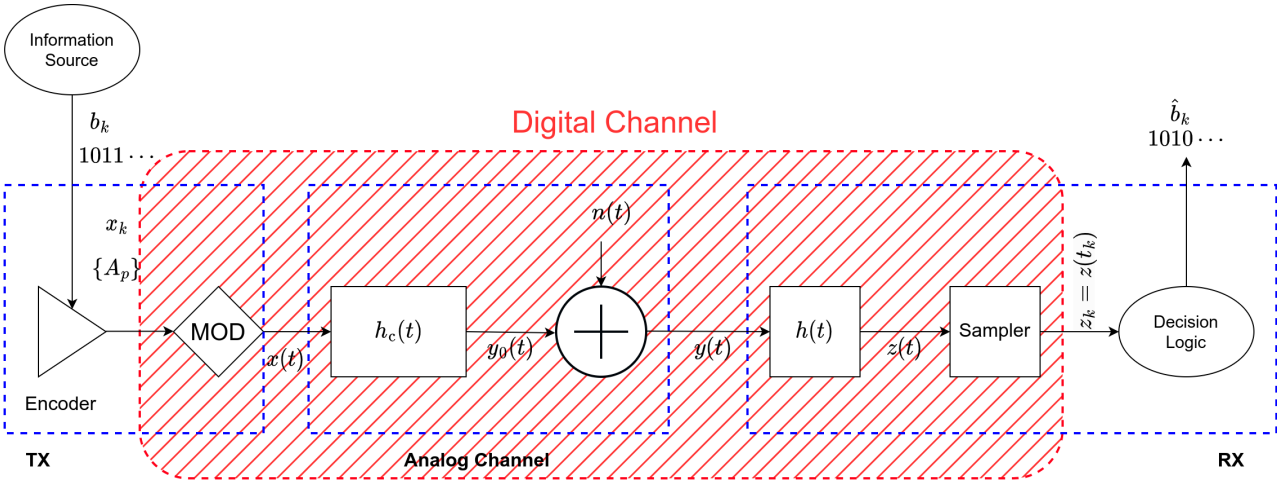
$$\sigma_x^2 = \mathbb{E}\left\{x_k^2\right\} = \beta^2\frac{M^2 - 1}{3} \tag{27}$$

```python
from commlib import pam_constellation
beta = 1.0
M = 16

c = pam_constellation(beta = beta, M = M)
p1 = c.avg_power()
print('Power computed from avg_power:', p1)

p2 = beta ** 2.0 * (M**2 - 1) / 3
print('Power computed from PAM formula:', p2)
```

Listing 4: pampower.py

```
Power computed from avg_power: 85.0
Power computed from PAM formula: 85.0
```

Εικόνα 4: Σύγκριση της (27) με την (22)



Εικόνα 5: Μοντέλο συστήματος PAM

Μπορούμε να συγκρίνουμε την (27) με την (22). Για τον σκοπό αυτό χρησιμοποιούμε το listing 4 και στην εικόνα 4 βλέπουμε τη σύγκριση των αποτελεσμάτων για $M = 16$.

Η μέση ισχύς του σήματος δίνεται από την:

$$\mathcal{P}_X = \int_{-\infty}^{+\infty} S_X(f)\mathrm{d}f \tag{28}$$

Αντικαθιστώντας την (19), έχουμε:

$$\mathcal{P}_X = \frac{\sigma_x^2}{T_\mathrm{S}} \int_{-\infty}^{+\infty} |P(f)|^2 \mathrm{d}f = \frac{\sigma_x^2}{T_\mathrm{S}} \int_{-\infty}^{+\infty} |p(t)|^2 \mathrm{d}t \tag{29}$$

Στην τελευταία σχέση χρησιμοποιήσαμε την ταυτότητα του Parseval. Η ενέργεια του παλμού γράφεται ως εξής:

$$\mathcal{E}_p = \int_{-\infty}^{+\infty} |p(t)|^2 \mathrm{d}t \tag{30}$$

Από τις παραπάνω σχέσεις προκύπτει ότι η μέση ισχύς $\mathcal{P}_X$ υπολογίζεται ως γινόμενο της μέσης ισχύς των συμβόλων σ_x^2 επί την ενέργεια του παλμού $\mathcal{E}_p$ διαιρεμένη δια την διάρκεια του συμβόλου T_S.

$$\mathcal{P}_X = \sigma_x^2 \frac{\mathcal{E}_p}{T_\mathrm{S}} \tag{31}$$

Ειδικά στην περίπτωση όπου τα πλάτη δίνονται από την (18), χρησιμοποιούμε την (27) για να δείξουμε ότι:

$$\mathcal{P}_X = \beta^2 \frac{M^2 - 1}{3T_\mathrm{S}} \mathcal{E}_p \tag{32}$$

5 Το μοντέλο του συστήματος

Στην εικόνα 5 δείχνουμε ένα μοντέλο ενός συστήματος PAM. Όπως φαίνεται και στην εικόνα ξεκινάμε από την πηγή της πληροφορίας η οποία γεννά bit b_k προς μετάδοση. Τα bit αυτά μετατρέπονται σε σύμβολα x_k στον κωδικοποιητή (encoder). Τα σύμβολα x_k στην περίπτωση του PAM είναι ένα από τα πλάτη A_p που είδαμε προηγουμένως στην (18). Στη συνέχεια ο διαμορφωτής (modulator - MOD) αναλαμβάνει να συνθέσει την κυματομορφή του PAM $x(t)$ σύμφωνα με την (3). Ο κωδικοποιητής και ο διαμορφωτής είναι μέρη του πομπού του συστήματος (transmitter - TX).

Στη συνέχεια το σήμα μεταδίδεται στο κανάλι. Σε μία πρώτη προσέγγιση το κανάλι περιγράφεται από την κρουστική του απόκριση $h_c(t)$ ή ισοδύναμα από την συνάρτηση μεταφορά $H_c(f)$ και τον θόρυβο $n(t)$. Οπότε το σήμα $y(t)$ στην έξοδο του καναλιού θα είναι

$$y(t) = x(t) \circledast h_c(t) + n(t) \tag{33}$$

Η $h_c(t)$ περιγράφει την ντετερμινιστική (μη τυχαία) επίδραση του καναλιού στο σήμα. Απουσία θορύβου ($n(t) = 0$) η έξοδος του συστήματος γράφεται ως εξής:

$$y_s(t) = x(t) \circledast h_c(t) \tag{34}$$

Το σήμα y_s αποτελεί την έξοδο του καναλιού εάν θεωρήσουμε ότι αυτό είναι επομένως χωρίς θόρυβο. Συνήθως το κανάλι μπορεί να εξασθενεί τις υψηλές συχνότητες οπότε γενικά η $H_c(f)$ μειώνεται όσο αυξάνει το f και αυτό πρέπει να αποτυπώνεται στην $h_c(t)$. Η στατιστική συμπεριφορά του θορύβου εξαρτάται από το κανάλι και γενικά υπάρχουν αρκετά διαφορετικές πηγές θορύβου που πρέπει κανείς να λάβει υπόψη του. Βλέπουμε ότι με αυτή την περιγραφή το κανάλι επιδρά στο αναλογικό σύστημα $x(t)$ και στην έξοδο του παράγεται το αναλογικό σήμα $y(t)$, αφορά δηλαδή αναλογικά σήματα και για αυτό τον λόγο αναφέρεται ως το *αναλογικό κανάλι*.

Το σήμα $y(t)$ στην έξοδο του καναλιού λαμβάνεται από τον δέκτη (receiver - RX). Ο δέκτης περιγράφεται από την δική του κρουστική απόκριση $h(t)$. Η $h(t)$ συμπεριλαμβάνει αφενός μεν τις μη ιδανικότητες του δέκτη και κυρίως την συχνοτική συμπεριφορά του αλλά και από τις τεχνικές που έχουν υλοποιηθεί στο εσωτερικό του σε μία προσπάθεια να αναδειχθεί καλύτερα η συνιστώσα του σήματος και να υποβαθμιστεί ο θόρυβος. Στη συνέχεια ο δέκτης δειγματοληπτεί το $z(t)$ και προσπαθεί να αποκωδικοποιήσει τα σύμβολα και τελικά να εκτιμήσει τα bits $\hat{b}_k$. Παρουσία του θορύβου ενδέχεται να συμβαίνουν σφάλματα, δηλαδή να έχουμε $\hat{b}_k \neq b_k$. Σκοπός μας είναι να προσπαθήσουμε να ποσοτικοποιήσουμε τις επιδόσεις του συστήματος βάσει της πιθανότητας σφάλματος bit P_b η οποία ορίζεται ως:

$$P_b = \Pr\left\{\hat{b}_k \neq b_k\right\} \tag{35}$$

Για να καταλάβουμε καλύτερα την επίδραση του θορύβου και πως θα υπολογίσουμε την P_b, θα κάνουμε μία σειρά από παραδοχές.

1. Απουσία θορύβου, το σήμα στο δέκτη προκύπτει ως έξοδος $y_s(t)$ ενός LTI συστήματος το οποίο σε πρώτη προσέγγιση θεωρούμε ότι έχει συνάρτηση μεταφοράς:

   $$H_c(f) = \sqrt{L} \tag{36}$$

   Είδαμε στο μάθημα 5 ότι ένα τέτοιο σύστημα ονομάζεται *επίπεδο* ενώ ο συντελεστής L περιγράφει τις απώλειες του καναλιού. Η κρουστική απόκριση του προκύπτει από τον αντίστροφο μετασχηματισμό Fourier του $H_c(f)$ και θα είναι:

   $$h_c(t) = \sqrt{L}\delta(t) \tag{37}$$

   Απουσία θορύβου επομένως θα έχουμε:

   $$y_s(t) = h_c(t) \circledast x(t) = \sqrt{L}\delta(t) \circledast x(t) = \sqrt{L}x(t) = \sqrt{L}\sum_{k=0}^{N} x_k p(t - kT_S) \tag{38}$$

   Ως εκ τούτου οι παλμοί $p(t)$ είναι ίδιοι τόσο για τον πομπό (σήμα $x(t)$) όσο και για τον δέκτη (σήμα $y_0(t)$).

2. Οι παλμοί $p(t)$ είναι περιορισμένοι μέσα στο διάστημα ενός συμβόλου $[-T_S/2, T_S/2]$. Αυτό σημαίνει ότι οι παλμοί $p(t - kT_S)$ δεν επικαλύπτονται και επομένως για $(k - \frac{1}{2})T_S \leq t < (k + \frac{1}{2})T_S$ θα έχουμε από την (3):

   $$x(t) = x_k p(t - kT_S) \tag{39}$$

   Η ενέργεια του παλμού $p(t)$ συμβολίζεται με $\mathcal{E}_p$ και ισούται με

   $$\mathcal{E}_p = \int_{-\frac{T_S}{2}}^{\frac{T_S}{2}} |p(t)|^2 \mathrm{d}t \tag{40}$$

   ενώ στο δέκτη, απουσία θορύβου το σήμα θα γράφεται στο ίδιο χρονικό διάστημα:

   $$y_s(t) = x_k\sqrt{L}p(t - kT_S) \tag{41}$$

   Είναι εύκολο να δείξουμε ότι σε αναλογία με την (29), η ισχύς του σήματος θα είναι

   $$\mathcal{P}_R = L\mathcal{P}_X = L\frac{\sigma_x^2}{T_S}\mathcal{E}_p \tag{42}$$

3. Ο θόρυβος $n(t)$ είναι *λευκός προσθετικός Gaussian θόρυβος* (AWG), οπότε όπως έχουμε δει στα προηγούμενα:

   - η PSD είναι σταθερή στο εύρος ζώνης του σήματος $y(t)$ οπότε έχουμε:

   $$S_n(f) = \frac{N_0}{2} \tag{43}$$

   ενώ η συνάρτηση αυτοσυσχέτισης $R_{nn}(\tau)$ θα δίνεται από την

   $$R_{nn}(\tau) = \frac{N_0}{2}\delta(\tau) \tag{44}$$

   - Ο θόρυβος είναι *προσθετικός* δηλαδή προστίθεται απευθείας στο σήμα που θα είχαμε απουσία του θορύβου. Οπότε το σήμα στο δέκτη θα είναι:

   $$y(t) = y_s(t) + n(t) = x_k\sqrt{L}p(t - kT_S) + n(t) \tag{45}$$

   - Ο θόρυβος ακολουθεί *Gaussian* κατανομή οπότε κάθε χρονική στιγμή το $n(t)$ είναι μια κανονική τυχαία μεταβλητή.

4. Το $n(t)$ έχει μέση τιμή μηδέν,

   $$\mathbb{E}\{n(t)\} = 0 \tag{46}$$

   Όταν είχαμε συζητήσει τις ιδιότητες του θορύβου AWG, είχαμε αποδείξει ότι οι παραπάνω συνθήκες έχουν ως αποτέλεσμα τα δείγματα του θορύβου $n(t_1)$ και $n(t_2)$ δύο διαφορετικές χρονικές στιγμές t_1 και t_2 $(t_1 \neq t_2)$ να είναι ανεξάρτητες τυχαίες μεταβλητές.

5. Δεδομένου του σήματος $y(t)$ που λαμβάνει, ο δέκτης μπορεί να κάνει μία επιπλέον επεξεργασία περνώντας το $y(t)$ από ένα σύστημα LTI με κρουστική απόκριση $h(t)$ οπότε υπολογίζει ένα σήμα:

   $$z(t) = y(t) \circledast h(t) \int_{-\infty}^{+\infty} h(t-\tau)y(\tau)\mathrm{d}\tau \tag{47}$$

   Ο σκοπός του συστήματος αυτού είναι να μειώσει την επίδραση του θορύβου $n(t)$ ώστε ο δέκτης να μπορέσει πιο εύκολα το σήμα. Ο δέκτης δειγματοληπτεί το $z(t)$ στις στιγμές $t = t_k = (k + \frac{1}{2})T_S$ και αποφασίζει για τα σύμβολα βάσει των τιμών των δειγμάτων $z_k = z(kT_S)$. Μπορούμε να δούμε ότι

   $$z_k = z(t_k) = y(t_k) \circledast h(t_k) = x_k\sqrt{L}\int_{-\infty}^{+\infty} p(\tau - kT_S)h(t_k - \tau)\mathrm{d}\tau + \int_{-\infty}^{+\infty} n(\tau)h(t_k - \tau)\mathrm{d}\tau \tag{48}$$

   Μπορούμε να αναγνωρίσουμε δύο συνιστώσες στο δεύτερο μέρος: την συνεισφορά του σήματος η οποία γράφεται:

   $$s_k = x_k\sqrt{L}\int_{-\infty}^{+\infty} p(\tau - kT_S)h(t_k - \tau)\mathrm{d}\tau \tag{49}$$

   και την συνεισφορά του θορύβου:

   $$n_k = \int_{-\infty}^{+\infty} n(\tau)h(t_k - \tau)\mathrm{d}\tau \tag{50}$$

   Δεδομένου ότι το n_k είναι η έξοδος ενός LTI συστήματος που στην είσοδο του έχει θόρυβο AWG, προκύπτει όπως δείξαμε στο μάθημα 8, θα έχει κανονική κατανομή με μηδενική μέση τιμή και διακύμανση:

   $$\sigma_n^2 = \frac{N_0}{2}\int_{-\infty}^{\infty} |h(t)|^2\mathrm{d}t = \frac{N_0}{2}\int_{-\infty}^{\infty} |H(f)|^2\mathrm{d}f \tag{51}$$

6. Λαμβάνοντας υπόψη την τιμή ζ που υπολογίζεται για το z_k στο δέκτη, θα πρέπει να έχουμε μία στρατηγική να εκτιμάει το σύμβολο x_k που μεταδόθηκε. Το x_k θα είναι ένα από τα M πιθανά σύμβολα του PAM A_p όπου $0 \leq p < M$. Ας υποθέσουμε ότι ο δέκτης επιλέγει ότι το $z_k = \zeta$ αντιστοιχεί σε ένα από τα πιθανά σύμβολα PAM έστω το r_k. Απουσία θορύβου, θα πρέπει $r_k = x_k$. Παρουσία θορύβου μπορούμε να επιλέξουμε το αποκωδικοποιημένο σύμβολο έτσι ώστε να είναι το πιο πιθανό να έχει μεταδοθεί από τον πομπό. Δηλαδή θέτουμε $r_k = A_q$ όπου το A_q είναι το σύμβολο που μεγιστοποιεί την πιθανότητα ο πομπός να έχει μεταδώσει το A_q δεδομένου ότι η τιμή για το z_k είναι ίση με ζ,

   $$P_k = \Pr\{x_k = A_q | z_k = \zeta\} \tag{52}$$

   Η P_k αναφέρεται ως *εκ των υστέρων πιθανότητα* με την έννοια ότι ερχόμαστε να την υπολογίσουμε μετά τον υπολογισμό της τιμής ζ του z_k. Οπότε θέλουμε να έχουμε μέγιστη την εκ των υστέρων πιθανότητα κάτι που διεθνώς αναφέρεται ως maximum a posteriori probability (MAP).

Θα χρησιμοποιήσουμε τις παραπάνω παραδοχές για να υπολογίσουμε τις επιδόσεις των συστημάτων επικοινωνιών που χρησιμοποιούν PAM αλλά και σε άλλου είδους διαμορφώσεις. Θα ξεκινήσουμε δείχνοντας ποιο είναι το βέλτιστο σύστημα LTI για τον δέκτη, επιλέγοντας την καλύτερη δυνατή $h(t)$ για να ελαχιστοποιήσουμε την επίδραση του θορύβου. Πρώτα όμως πρέπει να ορίσουμε το πηλίκο σήμα-προς-θόρυβο που θα χρησιμοποιήσουμε στην βελτιστοποίηση.

Μία χρήσιμη παρατήρηση είναι ότι μπορούμε να κάνουμε την οποιαδήποτε ανάλυση στο σύστημα για το σύμβολο $k = 0$ καθώς η συμπεριφορά του συστήματος δεν εξαρτάται από τη θέση του συμβόλου k: ο θόρυβος $n(t)$ είναι στατικός και το σύστημα από τον πομπό έως τον δέκτη είναι χρονικά αναλλοίωτο. Επομένως θεωρούμε ότι $k = 0$ στην περαιτέρω ανάλυση και έχουμε:

$$z_0 = z(t_0) = x_0\sqrt{L}\int_{-\infty}^{+\infty} p(\tau)h(t_0 - \tau)\mathrm{d}\tau + n_0 \tag{53}$$

Αν ορίσουμε το Γ_c ως εξής:

$$\Gamma_c = L\left(\int_{-\infty}^{+\infty} p(\tau)h(t_0 - \tau)\mathrm{d}\tau\right)^2 \tag{54}$$

τότε θα έχουμε:

$$z_0 = x_0\sqrt{\Gamma_c} + n_0 \tag{55}$$

Απουσία του θορύβου ($n_0 = 0$) η ισχύς του σήματος στην έξοδο του δέκτη θα είναι ανάλογη του $\{z_0^2\} = \Gamma_c\{x_0^2\}$. Η ισχύς των συμβόλων είναι το $\{x_0^2\}$ οπότε ο συντελεστής Γ_c προκύπτει ότι είναι το κέρδος του ψηφιακού καναλιού.

6 Ο Βέλτιστος Δέκτης

Θα προσπαθήσουμε τώρα να ελαχιστοποιήσουμε την επίδραση του θορύβου επιλέγοντας το $h(t)$ έτσι ώστε το σ_n^2 να είναι το ελάχιστο δυνατό ενώ το κέρδος του καναλιού Γ_c να είναι μέγιστο. Επομένως θέλουμε να μεγιστοποιήσουμε την ποσότητα:

$$\frac{\Gamma_c}{\sigma_n^2} = \frac{L}{2N_0}\frac{\left(\int_{-\infty}^{+\infty} h(t_0 - \tau)p(\tau)\mathrm{d}\tau\right)^2}{\int_{-\infty}^{\infty}|h(\tau)|^2\mathrm{d}\tau} \tag{56}$$

Για να μεγιστοποιήσουμε την (56) θα πρέπει να επιλέξουμε το βέλτιστο $h(t)$. Όταν έχουμε ένα ολοκλήρωμα της μορφής

$$I = \left|\int_{-\infty}^{+\infty} f(\tau)g(\tau)\mathrm{d}\tau\right| \tag{57}$$

τότε μπορούμε να δείξουμε ότι για αυτό πάντα ισχύει:

$$I^2 = \left(\int_{-\infty}^{+\infty} f(\tau)g(\tau)\mathrm{d}\tau\right)^2 \leq \int_{-\infty}^{+\infty}|f(\tau)|^2\mathrm{d}\tau\int_{-\infty}^{+\infty}|g(\tau)|^2\mathrm{d}\tau \tag{58}$$

και πως η ισότητα ισχύει όταν το $f(\tau)$ είναι ανάλογο του $g(t)$, δηλαδή υπάρχει ένα λ, τέτοιο ώστε

$$f(\tau) = \lambda g^*(\tau) \tag{59}$$

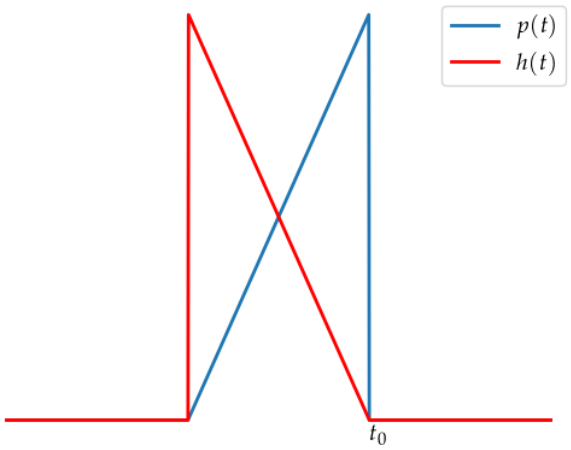
όπου το $g^*(\tau)$ είναι το μιγαδικό συζυγές του $g(\tau)$. Στην περίπτωση αυτή το μεγιστοποιείται και επομένως η βέλτιστη επιλογή για την συνάρτηση μεταφοράς του LTI συστήματος είναι:

$$h(t_0 - \tau) = \lambda p^*(\tau) \tag{60}$$

ή ισοδύναμα:

$$h(\tau) = \lambda p^*(t_0 - \tau) \tag{61}$$

Από την παραπάνω σχέση προκύπτει ότι ο βέλτιστος δέκτης είναι αυτός του οποίου η απόκριση $h(\tau)$ προκύπτει από την μετατόπιση της βασικής κυματομορφής του παλμού $p(t)$ κατά t_0 (που είναι η χρονική στιγμή της δειγματοληψίας) και την αντιστροφή του ως προς τον άξονα του χρόνου τ. Στην εικόνα 6 δείχνουμε ένα παράδειγμα για το πως υπολογίζεται η βέλτιστη απόδοση ενός δέκτη, όταν η βασική κυματομορφή παλμού $p(t)$ είναι τριγωνική και η χρονική στιγμή δειγματοληψίας t_0 είναι στην κορυφή του τριγώνου. Σύμφωνα με την (61), θα πρέπει να πάρουμε την συμμετρική συνάρτηση $p(-t)$ του $p(t)$ και να την μετατοπίσουμε στο $t = t_0$ οπότε προκύπτει η κόκκινη συνάρτηση στην εικόνα.



Εικόνα 6: Ένα παράδειγμα απόκρισης βέλτιστου δέκτη.

Για αυτή την επιλογή έχουμε

$$\Gamma_{\mathrm{c}} = L\left(\int_{-\infty}^{+\infty} p(\tau)h(t_0-\tau)\mathrm{d}\tau\right)^2 L\lambda^2\left(\int_{-\infty}^{+\infty}|p(\tau)|^2\mathrm{d}\tau\right)^2 = L\lambda^2\mathcal{E}_p^2 \tag{62}$$

όπου $\mathcal{E}_p$ είναι η ενέργεια του παλμού $p(t)$,

$$\mathcal{E}_p = \int_{-\infty}^{+\infty}|p(\tau)|^2\mathrm{d}\tau \tag{63}$$

Επίσης αν υπολογίσουμε το σ_n^2 θα βρούμε

$$\sigma_n^2 = \frac{N_0}{2}\int_{-\infty}^{\infty}|h(t)|^2\mathrm{d}t = \frac{\lambda^2 N_0}{2}\int_{-\infty}^{\infty}|p(t_0-t)|^2\mathrm{d}t = \frac{\lambda^2 N_0}{2}\int_{-\infty}^{\infty}|p(t')|^2\mathrm{d}t' = \frac{\lambda^2 N_0}{2}\mathcal{E}_p \tag{64}$$

όπου κάναμε μία αλλαγή μεταβλητής ολοκλήρωσης $\tau' = t_0 - \tau$. Αν υπολογίσουμε το βέλτιστο πηλίκο $\Gamma_{\mathrm{c}}/\sigma_n^2$, προκύπτει ότι:

$$\frac{\Gamma_{\mathrm{c}}}{\sigma_n^2} = \frac{2L}{N_0}\int_{-\infty}^{+\infty}p^2(\tau)\mathrm{d}\tau = \frac{2L\mathcal{E}_p}{N_0} \tag{65}$$

Παρατηρούμε ότι το πηλίκο αυτό είναι ανεξάρτητο από το λ και επομένως μπορούμε να θεωρήσουμε ότι $\lambda = 1$ οπότε το βέλτιστο $h(t)$ δίνεται από την σχέση:

$$h(\tau) = p^*(t_0-\tau) \tag{66}$$

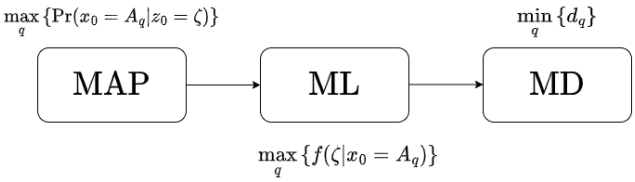
7 Κριτήριο αποκωδικοποίησης

Έχει ενδιαφέρον να αποσαφηνίσουμε τι ακριβώς σημαίνει το MAP στην (52) για την περίπτωση του PAM. Χρησιμοποιώντας τον κανόνα του Bayes μπορούμε να γράψουμε την πιθανότητα αυτή ως:

$$P_0 = \Pr\{x_0 = A_q | z_0 = \zeta\} = \frac{f(\zeta|x_0=A_q)P(x_0=A_q)}{f(\zeta)} \tag{67}$$

όπου η $f(\zeta)$ είναι η PDF του z_0 ενώ η $f(\zeta|x_0=A_q)$ είναι η PDF του z_0 δεδομένου ότι η τιμή του x_0 είναι ίσο με A_q. Αν θεωρήσουμε ότι τα σύμβολα είναι ισοπίθανα

$$P\{x_0 = A_q\} = \frac{1}{M} \tag{68}$$



Εικόνα 7: Ισοδύναμες στρατηγικές αποκωδικοποιήσης.

Επομένως θα έχουμε:

$$P_0 = \Pr\{x_0 = A_q | z_0 = \zeta\} = f(\zeta | x_0 = A_q) \frac{1}{M f(\zeta)} \tag{69}$$

Στην παραπάνω έκφραση μόνο το $f(\zeta|x_0 = A_q)$ εξαρτάται από το A_q και επομένως για να έχουμε MAP θα πρέπει να μεγιστοποιήσουμε αυτή την ποσότητα. Η πιθανότητα $f(\zeta|x_0 = A_q)$ ονομάζεται στην βιβλιογραφία η *πιθανοφάνεια* του z_0 όταν το x_0 έχει την τιμή A_q. Προκύπτει επομένως ότι στην περίπτωση των ισοπίθανων συμβόλων το MAP είναι ισοδύναμο με την μέγιστη πιθανοφάνεια (maximum likelyhood - ML).

Το z_0 γράφεται

$$z_0 = x_0 \sqrt{\Gamma_c} + n_0 \tag{70}$$

Το n_0 είναι κανονική τυχαία μεταβλητή με μέση τιμή μηδέν και διακύμανση σ_n^2 όπως δίνεται από την (51). Αν ορίσουμε:

$$Z = \frac{z_0}{\sqrt{\Gamma_c}} = x_0 + \frac{n_0}{\sqrt{\Gamma_c}} \tag{71}$$

Το Z δεδομένης της τιμής του x_0 θα είναι μία κανονική τυχαία μεταβλητή με μέση τιμή x_0 και διακύμανση ίση με

$$\sigma^2 = \sigma_n^2 / \Gamma_c = \frac{N_0}{2\Gamma_c} \mathcal{E}_p = \frac{N_0}{2L\mathcal{E}_p} \tag{72}$$

Η πυκνότητα πιθανότητας του Z δεδομένης της τιμής του x_0, $f(\zeta|x_0 = A_q)$ θα δίνεται επομένως από την

$$f(\zeta | x_0 = A_q) = \frac{1}{\sigma_n \sqrt{2\pi}} \exp\left(-\frac{(\zeta - A_q)^2}{2\sigma^2} \right) \tag{73}$$

Σύμφωνα με την παραπάνω σχέση και εφόσον το e^{-x^2} είναι φθίνουσα συνάρτηση του x, το $f(\zeta|x_0 = A_q)$ μεγιστοποιείται για το σύμβολο A_q που έχει το ελάχιστο $(\zeta - A_q)^2$. Όταν ο δέκτης υπολογίσει το Z, η καλύτερη επιλογή είναι επομένως να ελαχιστοποιήσουμε την ποσότητα

$$d_q = |Z - A_q| \tag{74}$$

που είναι η απόσταση του Z από τα σύμβολα A_q του συστήματος. Το βέλτιστο σύμβολο είναι αυτό που έχει την μικρότερη απόσταση d_q. Στην εικόνα 7 δείχνουμε τις ισοδύναμες στρατηγικές όπου με MD έχουμε συμβολίσει την ελάχιστη απόσταση (minimum distance - MD)

8 Πηλίκο σήμα-προς-θόρυβο

Πριν προχωρήσουμε στον υπολογισμό της πιθανότητας σφάλματος (35) έχει νόημα να ορίσουμε ένα χρήσιμο μέτρο που χαρακτηρίζει πόσο ισχυρός είναι ο θόρυβος σε σχέση με το σήμα. Είδαμε ότι η ισχύς του σήματος στην είσοδο του δέκτη δίνεται από την (42) οπότε η ενέργεια του σήματος στην διάρκεια T_S θα είναι:

$$\mathcal{E}_R = \mathcal{P}_R T_S = L\sigma_x^2 \mathcal{E}_p \tag{75}$$

Μπορούμε επίσης να υπολογίσουμε την ενέργεια του σήματος στην διάρκεια ενός bit T_b. Θυμηθείτε ότι ο αριθμός των bit ανά σύμβολο m συνδέεται με το πλήθος M των συμβόλων $M = 2^m$ ή ισοδύναμα:

$$m = \log_2 M \tag{76}$$

Επομένως η διάρκεια του bit θα είναι:

$$T_b = \frac{T_S}{m} = \frac{T_S}{\log_2 M} \tag{77}$$

Η ενέργεια επομένως *ανά bit* θα δίνεται από την:

$$\mathcal{E}_{\mathrm{b}} = \mathcal{P}_R T_{\mathrm{b}} = \frac{L\sigma_x^2 \mathcal{E}_p}{\log_2 M} = L\beta^2 \frac{M^2-1}{3\log_2 M}\mathcal{E}_p \tag{78}$$

Ορίζουμε το *πηλίκο σήμα-προς-θόρυβο ανά bit* (signal to noise ration per bit - $\mathrm{SNR_b}$) ως τον λόγο της ενέργειας του σήματος στη διάρκεια ενός bit προς την φασματική πυκνότητα ισχύος του θορύβου N_0 ως εξής:

$$\mathrm{SNR_b} = \frac{\mathcal{E}_b}{N_0} \tag{79}$$

Ορισμένες φορές χρησιμοποιούμε και το πηλίκο σήμα-προς-θόρυβο ανά σύμβολο $\mathrm{SNR_S}$ το οποίο ορίζεται ως η ενέργεια του σήματος στην διάρκεια του συμβόλου προς την φασματική πυκνότητα του θορύβου:

$$\mathrm{SNR_S} = \frac{\mathcal{E}_R}{N_0} \tag{80}$$

Τα πηλίκα αυτά είναι καθαροί αριθμοί και δεν έχουν μονάδες. Όπως είχαμε δει και στα προηγούμενα η ισχύς το θορύβου στο εύρος συχνοτήτων $f \in [-B, B]$ είναι ίση με $N_0 B$. Αν η ισχύς μετριέται σε Watt τότε έπεται ότι το N_0 πρέπει να έχει μονάδες Watt/Hz και εφόσον το Hz είναι 1/sec έπεται ότι οι μονάδες του N_0 θα είναι Watt $\times$ sec δηλαδή Joule. Οπότε το N_0 έχει μονάδες ενέργειας και τα πηλίκα $\mathrm{SNR_S}$ και $\mathrm{SNR_b}$ θα πρέπει να είναι καθαροί αριθμοί.

Στην περίπτωση του PAM θα έχουμε:

$$\mathrm{SNR_b} = \frac{L\beta^2}{N_0}\frac{M^2-1}{3\log_2 M}\mathcal{E}_p \tag{81}$$

$$\mathrm{SNR_S} = \frac{L\beta^2}{N_0}\frac{M^2-1}{3}\mathcal{E}_p \tag{82}$$

Ποια είναι η πρακτική αξία του $\mathrm{SNR_S}$ και του $\mathrm{SNR_b}$; Αν ξεκινήσουμε με το $\mathrm{SNR_S}$ στην (80) βλέπουμε ότι στον αριθμητή έχουμε το $\mathcal{E}_R$ το οποίο μετράει την ενέργεια που λαμβάνουμε για κάθε σύμβολο στο δέκτη ενώ στον παρανομαστή έχουμε την φασματική πυκνότητα N_0 που καθορίζει την ισχύ του θορύβου. Επομένως εάν σε ένα σύστημα έχουμε υψηλή τιμή του $\mathrm{SNR_S}$ περιμένουμε η επίδραση το θορύβου να είναι ασθενέστερη από όταν το $\mathrm{SNR_S}$ είναι μικρότερο. Το ίδιο ισχύει και για το $\mathrm{SNR_b}$. Η διαφορά μεταξύ του $\mathrm{SNR_S}$ και του $\mathrm{SNR_b}$ έγκειται στο γεγονός ότι το πρώτο αναφέρεται στην ενέργεια του συμβόλου και το δεύτερο στην ενέργεια του bit. Οπότε φανταστείτε ένα σενάριο που θέλουμε να γνωρίζουμε πιο είναι το κατάλληλο πηλίκο σήμα-προς-θόρυβο που χρειάζεται ένα σύστημα για να λειτουργήσει. Αν για το σκοπό αυτό υπολογίσουμε το $\mathrm{SNR_S}$ τότε αναφερόμαστε στα σύμβολα και όχι στα bit. Οπότε έχουμε μία πληροφορία για την απαιτούμενη ενέργεια $\mathcal{E}_R$ που χρειάζεται το σύστημα να λάβει στο δέκτη για να λειτουργήσει αλλά αυτή η ενέργεια είναι υπολογισμένη ανά σύμβολο και όχι ανά bit το οποίο για εμάς είναι η βασική μονάδα της πληροφορίας.

Ας δώσουμε ένα παράδειγμα. Έστω συγκρίνουμε δύο διαφορετικά συστήματα A που μεταδίδει 4bits ανά σύμβολο ($m = 4$ και $M = 16$) και το B που που μεταδίδει 1bit ανά σύμβολο ($m = 1$, $M = 2$). Έστω επίσης ότι βρίσκουμε ότι το απαιτούμενο $\mathrm{SNR}_S(A)$ και το A είναι διπλάσιο από αυτό του $\mathrm{SNR}_S(B)$, $\mathrm{SNR}_S(A) = 2\mathrm{SNR}_S(B)$. Ποιο σύστημα χρειάζεται την μικρότερη ενέργεια για να λειτουργήσει; Ίσως αρχικά να θεωρήσουμε ότι αυτό είναι το B αφού απαιτεί το μισό πηλίκο-σήμα-προς θόρυβο και επομένως εάν το N_0 είναι το ίδιο και για τα δύο συστήματα, χρειάζεται την μισή $\mathcal{E}_R$, $\mathcal{E}_R(A) = 2\mathcal{E}_R(B)$. Αυτό είναι σωστό αλλά μην ξεχνάμε ότι το $\mathcal{E}_R$ μετράει την ενέργεια ανά σύμβολο και όχι ανά bit που για μας είναι η βασική μονάδα της πληροφορίας. Για το έχουμε $\mathrm{SNR}_R(A) = 4\mathrm{SNR}_b(A)$ εφοσον μεταδίδουμε 4bits ανά σύμβολο ενώ για το B έχουμε $\mathrm{SNR}_R(B) = \mathrm{SNR}_b(B)$ και επομένως $2\mathcal{E}_b(A) = \mathcal{E}_b(B)$. Άρα η απαιτούμενη ενέργεια *ανά bit* είναι τελικά μικρότερη για το A και επομένως αυτό το σύστημα απαιτεί την μικρότερη ενέργεια για να μεταδώσει την ίδια πληροφορία.

9 Η κλάση `pam_link`

```
467 class pam_link:
468
469     def __init__(self, M = 4, L = 1, SNRbdB = 16, Rb = 2,
470                        samples_per_symbol = 10, guard_syms = 10,
471                        Nbits = 10000):
472         self.M = M
473         self.m = np.log2(M)
474         self.L = L
475         self.samples_per_symbol = samples_per_symbol
476         self.c = pam_constellation( M, SNRbdB = SNRbdB )
```

```
        self.SNRbdB = SNRbdB
        self.SNRb = to_linear(SNRbdB)
        self.SNRs = self.SNRb * self.m
        self.SNRbsdB = to_dB( self.SNRs )
        self.Rb = Rb
        self.Tb = 1/Rb
        self.TS = self.Tb * self.m
        self.RS = 1 / self.TS
        self.Ep = self.TS
        self.Gc = L * self.Ep ** 2
        self.sx2 = self.c.avg_power()
        self.ES = L * self.sx2 * self.Ep
        self.Eb = self.ES / self.m
        self.N0 = self.Eb / self.SNRb
        self.guard_syms = guard_syms
        self.Nbits = Nbits

    def modulate(self, bits):
        """
        Create analog representation of PAM signal
        """
        self.bk = bits
        self.bk_array = str_to_array( self.bk )
        self.xk = self.c.bits_to_symbols( self.bk )
        self.xt = digital_signal(TS = self.TS,
                                 samples_per_symbol = self.samples_per_symbol,
                                 guard_syms = self.guard_syms,
                                 symbols = self.xk)

    def transmit(self):
        """
        Transmit signal through channel and add AWGN
        """
        self.yt0 = self.L * self.xt
        self.nt = awgn(self.xt.t, self.N0)
        self.yt = self.yt0 + self.nt

    def p(self, t = None):
        """
        PAM pulse
        """
        samples = np.zeros( self.xt.N )
        samples[0 : self.samples_per_symbol] = 1
        return signal(t = self.xt.t, samples = samples)

    def h(self, t = None):
        """
        Optimal receiver impulse response
        """
        return self.p()

    def z(self):
        """
        Signal at the output of the receiver
        """
        h = self.h()
        return self.yt.conv( h )

    def calc_estimates(self):
        """
        Calculate symbol estimates
        """
        self.zt = self.z()
        self.tk = self.xt.tguard + np.arange(0, self.xt.Nsymbols) * self.TS + self.TS - self.xt.Dt
        self.zk = self.zt.interp(self.tk)
        self.zkn = self.zk / np.sqrt(self.Gc)

    def comp_estimates(self):
        """
        Plot symbol estimates and original symbols
        """
        plt.plot(self.zkn, 'o', label = 'estimates')
        plt.plot(self.xk, 'o', label = 'original')
        plt.xlabel('$k$')
        plt.legend()

```

```
553     def decode(self):
554         """
555         Decode the symbols
556         """
557         self.calc_estimates()
558         self.rbk = self.c.symbols_to_bits( self. zkn )
559         self.rbk_array = str_to_array(self.rbk)
560         self.errors = np.sum (np.abs( self.rbk_array - self.bk_array) )
561         self.Pb = self.errors / self.bk_array.size
562 
563     def simulate(self):
564         """
565         Simulate the end-to-end system
566         """
567         self.bk_array = np.random.randint(0, 2, size = self.Nbits)
568         self.bk = array_to_str( self.bk_array )
569         self.modulate( self.bk )
570         self.transmit()
571         self.decode()
```

Listing 5: Η κλάση `pam_link`

Για να καταλάβουμε καλύτερα τις θεωρητικές έννοιες που είδαμε στο μάθημα αυτό, υλοποιούμε την κλάση `pam_link`. Θεωρούμε ότι οι βασικοί παλμοί $p(t)$ του συστήματος είναι τετραγωνικοί. Αναφέρουμε μερικά γνωρίσματα της κλάσης:

- `M`: Ο αριθμός των διακριτών συμβόλων του PAM.
- `m`: Ο αριθμός των bit ανά σύμβολο.
- `L`: Ο συντελεστής απωλειών του καναλιού.
- `SNRbdB`: Το πηλίκο σήμα-προς-θόρυβο ανά bit σε dB.
- `Rb`: Ο ρυθμός μετάδοσης bit του συστήματος.
- `samples_per_symbol`: Το πλήθος των δειγμάτων του σήματος $x(t)$ που αντιστοιχούν σε κάθε δείγμα.
- `guard_syms`: Στο σήμα $x(t)$ υπάρχει δεξιά και αριστερά στον άξονα του χρόνου μία περίοδος φύλαξης (guard period) όπου το σήμα θα είναι ίσο με μηδέν. Το `guard_syms` είναι το εύρος της περιόδου αυτής μετρούμενη σε διάρκειες συμβόλων.
- `TS`: Η διάρκεια του κάθε συμβόλου.
- `Tb`: Η διάρκεια του κάθε bit.
- `Nbits`: Το πλήθος των bit προς μετάδοση.
- `Ep`: Η ενέργεια του βασικού παλμού $p(t)$.
- `ES`: Η ενέργεια του
- `N0`: Η φασματική πυκνότητα ισχύος του θορύβου.

Επίσης συνοψίζουμε μερικές από τις μεθόδους της κλάσης:

- `modulate`: Χρησιμοποιείται για να δημιουργήσει την κυματομορφή $x(t)$ που αντιστοιχεί στα bits που καθορίζονται από το όρισμα `bits`.
- `transmit`: Προσθέτει τον θόρυβο και εξασθενεί το σήμα σύμφωνα με το συντελεστή απωλειών του καναλιού.
- `p`: Ο βασικός παλμός PAM
- `h`: Η ιδανική κρουστική απόκριση του δέκτη του PAM (που στην περίπτωση τετραγωνικού παλμού $p(t)$ ταυτίζεται με το $p(t)$.
- `z`: Υπολογισμός της εξόδου του δέκτη $z(t)$.
- `calc_estimates`: Εκτίμηση των συμβόλων εξόδου z_k και $z_k/\sqrt{\Gamma_c}$.
- `comp_estimates`: Κάνει γραφική παράσταση της εκτίμησης των συμβόλων και των αρχικών συμβόλων στο ίδιο γράφημα.

- `decode`: Υπολογισμός των bit στην έξοδο του δέκτη από τις εκτιμήσεις των συμβόλων z_k.
- `simulate`: Προσομοίωση του συστήματος από άκρη-σε-άκρη. Στην ουσία η μέθοδος δημιουργεί τυχαία bit, τα διαμορφώνει στο σήμα το οποίο το μεταδίδει από το κανάλι, προσθέτοντας θόρυβο και κάνοντας την αποκωδικοποίηση των συμβόλων και τον υπολογισμό των bit εξόδου στον δέκτη.

```
import numpy as np
import matplotlib.pyplot as plt
from commlib import pam_link

M = 4                           # PAM order
SNRbdB = 16                     # Specified SNR per bit
Nbits = 10000                   # Number of bits
samples_per_symbol = 10         # samples per bit
plt.close('all')

link = pam_link(M = M, SNRbdB = SNRbdB,
                samples_per_symbol=samples_per_symbol,
                Rb = 2e9, Nbits = Nbits)
link.simulate()
plt.figure()
link.yt.plot( norm_by_t = link.TS, line_type = 'b-', xlabel='$t$', label='$x(t)$' )
link.xt.plot( norm_by_t = link.TS, line_type = 'r-', xlabel='$t$', label='$y(t)$' )
plt.legend()
plt.xlim([0, 100])
plt.grid()

plt.figure()
link.comp_estimates()
plt.xlim([0, 2000])

print('Error probability: %e' %link.Pb)
```

Listing 6: decode.py

Στο listing 6 δείχνουμε ένα παράδειγμα προσομοίωσης του συστήματος με την βοήθεια της κλάσης `pam_link`. Στην εικόνα 8 έχουμε παραστήσει τις κυματομορφές του 4-PAM και τις εκτιμήσεις των συμβόλων για δύο διαφορετικές τιμές του $\mathrm{SNR_b}$. Παρατηρούμε ότι, όπως άλλωστε είναι αναμενόμενο, οι κυματομορφή του PAM παρουσιάζει έντονες αποκλίσεις στο δέκτη από την αρχική στον πομπό για χαμηλό πηλίκο σήμα προς θόρυβο. Επίσης για υψηλό σήμα προς θόρυβο οι εκτιμήσεις των συμβόλων σχηματίζουν ζώνες γύρω από τα αρχικά σύμβολα ενώ για χαμηλό $\mathrm{SNR_b}$ οι ζώνες αυτές αρχίζουν να επικαλύπτονται. Δεδομένου ότι η αποκωδικοποίηση γίνεται με την λογική της πλησιέστερης απόστασης, στη δεύτερη περίπτωση θα έχουμε και σαφώς περισσότερα σφάλματα.

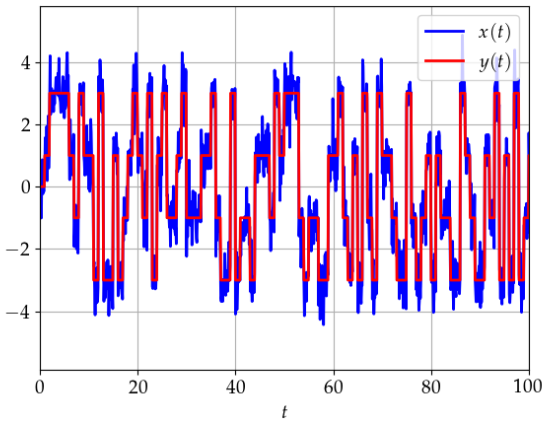
10 Υπολογισμός της πιθανότητας σφάλματος

Στην ενότητα αυτή θα υπολογίσουμε την πιθανότητα (35) βάσει των όσων μάθαμε μέχρι τώρα. Στην εικόνα 7 είδαμε ότι στο δέκτη, δεδομένης της τιμής του Z, θεωρούμε ότι λάβαμε το σύμβολο A_p εάν αυτό έχει την ελάχιστη απόσταση d_p από το ζ.

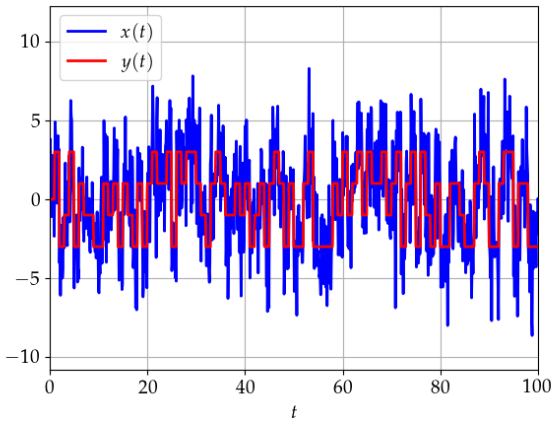
Στην εικόνα 9, δείχνουμε μερικά παραδείγματα αποκωδικοποίησης για την περίπτωση του 4-PAM. Στις περιπτώσεις (a) και (b) μεταδίδουμε στον πομπό το σύμβολο A_1 ενώ στις (c) και (d) μεταδίδουμε το σύμβολο A_0. Στην περίπτωση (a) όλα πάνε καλά: ο δέκτης λαμβάνει το ζ το όποιο είναι πλησιέστερα στο αρχικό σύμβολο A_1 από οποιοδήποτε άλλο A_p. Οπότε δεν θα συμβεί σφάλμα στην αποκωδικοποίηση. Ο δέκτης θα αποφανθεί ότι το σύμβολο που ήθελε να στείλει ο πομπός είναι το A_1. Στην περίπτωση (b) όμως έχουμε πρόβλημα: ο θόρυβος "πέταξε" την εκτίμηση ζ πιο κοντά στο A_0 από ότι στο αρχικό σύμβολο A_1 και επομένως ο δέκτης θα κάνει λάθος στην αποκωδικοποίηση. Στην εικόνα (c) και (d) μεταδίδουμε ένα σύμβολο που είναι στην άκρη (δεν υπάρχει δηλαδή άλλο σύμβολο αριστερότερα του A_0). Και στις δύο περιπτώσεις δεν κάνουμε λάθος επειδή το ζ βρίσκεται πιο κοντά στο A_0. Παρατηρείστε ότι στην περίπτωση το (d), όσο αριστερότερα και να βρεθεί το ζ από το A_0, δεν θα γίνει σφάλμα καθώς το πλησιέστερο σύμβολο παραμένει το A_0.

Βάσει του παραπάνω παραδείγματος μπορούμε να σκεφτούμε την γενικότερη περίπτωση του M-PAM όπου έχουμε M διακριτά σύμβολα A_k όπου $0 \leq k \leq M-1$. Αν συμβολίσουμε με E το ενδεχόμενο εσφαλμένης αποκωδικοποίησης *συμβόλου* τότε μπορούμε να διαπιστώσουμε τα εξής:

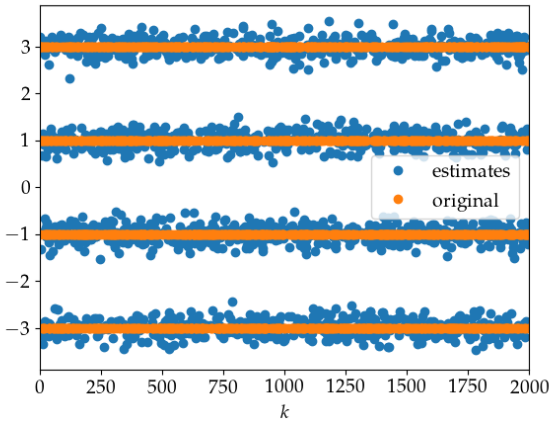
- Αν ο πομπός εκπέμψει ένα σύμβολο A_k που δεν είναι στην άκρη, δηλαδή $k \neq 0, M-1$ τότε για να συμβεί σφάλμα το ζ θα πρέπει να βρεθεί πιο κοντά σε κάποιο άλλο σύμβολο είτε προς τα αριστερά είτε προς τα δεξιά.



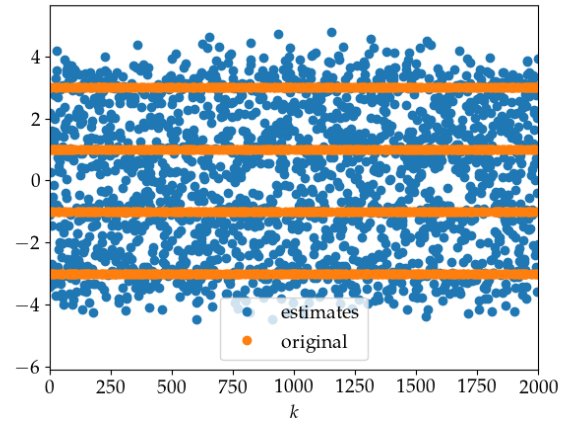
(a) $\mathrm{SNR_b} = 16\mathrm{dB}$



(b) $\mathrm{SNR_b} = 5\mathrm{dB}$

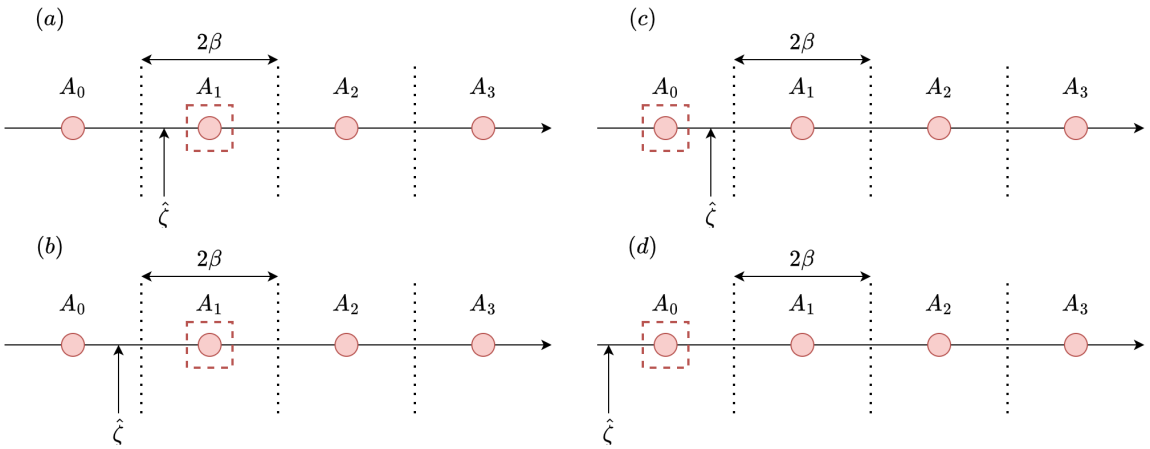


(c) $\mathrm{SNR_b} = 16\mathrm{dB}$



(d) $\mathrm{SNR_b} = 5\mathrm{dB}$

Εικόνα 8: Κυματομορφές και εκτιμήσεις συμβόλων για το 4−PAM



Εικόνα 9: Αποκωδικοποίηση στην περίπτωση του 4-PAM.

Για να συμβεί αυτό θα πρέπει να έχουμε $Z > A_k + \beta$ ή $Z < A_k - \beta$. Οι δύο αυτές συνθήκες συνδυάζονται στο $|Z - A_k| > \beta$. Οπότε:

$$\Pr\{E|x_0 = A_k\} = \Pr\{|Z - A_k| > \beta | x_0 = A_k\} \text{ εάν } k \neq 0, M-1 \tag{83}$$

- Αν ο πομπός εκπέμψει το αριστερότερο σύμβολο A_0, τότε σφάλμα συμβαίνει μόνο όταν $Z - A_0 > \beta$ και επομένως:

$$\Pr\{E|x_0 = A_0\} = \Pr\{Z - A_0 > \beta | x_0 = A_0\} \tag{84}$$

- Αν ο πομπός εκπέμψει το δεξιότερο σύμβολο A_{-1}, τότε σφάλμα συμβαίνει μόνο όταν $\zeta - A_{M-1} < -\beta$, οπότε:

$$\Pr\{E|x_0 = A_{M-1}\} = \Pr\{Z - A_{M-1} < -\beta | x_0 = A_{M-1}\} \tag{85}$$

Σε κάθε περίπτωση εάν το $x_0 = A_k$ τότε θα έχουμε $Z - A_k = n_0/\sqrt{\Gamma_c} = \hat{n}_0$. Η τυχαία μεταβλητή $w = Z - A_q = n_0/\sqrt{\Gamma_c}$ ακολουθεί κανονική κατανομή, έχει μέση τιμή μηδέν και τυπική απόκλιση σ^2 που δίνεται από την (72). Αν συμβολίσουμε με E το ενδεχόμενο εσφαλμένης αποκωδικοποίησης *συμβόλου* τότε μπορούμε να γράψουμε:

$$\Pr\{E\} = \sum_k \Pr\{E|x_0 = A_q\} \Pr\{x_0 = A_q\} = \frac{1}{M} \sum_k \Pr\{E|x_0 = A_q\} = \\ \frac{1}{M}\left((M-2)\Pr\{|w| > \beta\} + \Pr\{w > \beta\} + \Pr\{w < -\beta\}\right) \tag{86}$$

Μπορούμε να υπολογίσουμε τις πιθανότητες χρησιμοποιώντας την συνάρτηση Q. Όπως είδαμε στο 6ο μάθημα θα έχουμε:

$$\Pr\{w > \beta\} = Q\left(\frac{\beta}{\sigma}\right) \tag{87}$$

Επίσης είχαμε δει ότι:

$$\Pr\{w < -\beta\} = 1 - Q\left(-\frac{\beta}{\sigma}\right) = Q\left(\frac{\beta}{\sigma}\right) \tag{88}$$

Για την πιθανότητα $\Pr\{|w| > \beta\}$ θα έχουμε:

$$\Pr\{|w| > \beta\} = \Pr\{w > \beta\} + \Pr\{w < -\beta\} = 2Q\left(\frac{\beta}{\sigma}\right) \tag{89}$$

Η πιθανότητα σφάλματος $\Pr\{E\}$ θα γραφτεί:

$$\Pr\{E\} = 2\frac{M-1}{M} Q\left(\frac{\beta}{\sigma}\right) \tag{90}$$

Ας ορίσουμε το γ ως εξής:

$$\gamma = \frac{\beta^2}{\sigma^2} \tag{91}$$

Από την (78) μπορούμε να βρούμε ότι:

$$\beta^2 = \frac{\mathcal{E}_b}{L\mathcal{E}_p} \frac{3\log_2 M}{M^2 - 1} \tag{92}$$

και λαμβάνοντας υπόψη την (72),

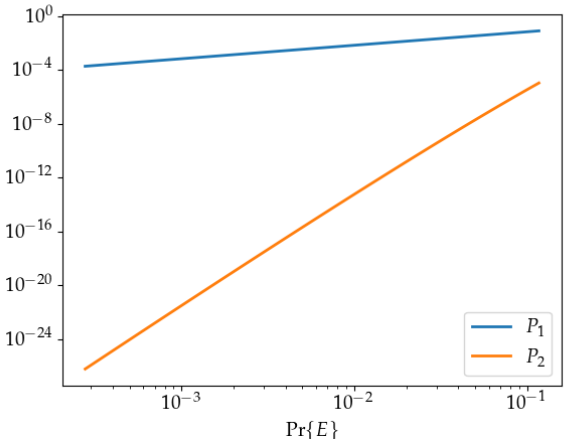
$$\gamma = \frac{\mathcal{E}_b}{N_0} \frac{6\log_2 M}{M^2 - 1} = \frac{6\log_2 M}{M^2 - 1} \mathrm{SNR_b} \tag{93}$$

Η έκφραση για την πιθανότητα σφάλματος συμβόλου γράφεται και ως εξής:

$$\Pr\{E\} = 2\frac{M-1}{M} Q\left(\sqrt{\gamma}\right) = 2\frac{M-1}{M} Q\left(\sqrt{\frac{6\log_2 M}{M^2 - 1} \mathrm{SNR_b}}\right) \tag{94}$$

Αν και η πιθανότητα αυτή είναι χρήσιμη, όπως είπαμε και στα προηγούμενα μας ενδιαφέρει τελικά η πιθανότητα σφάλματος *bit* στην (35) και όχι συμβόλου. Ας θεωρήσουμε την περίπτωση του 4-PAM όπου η αντιστοίχιση των συμβόλων σε bit δίνεται από την (9). Αν υποθέσουμε ότι μεταδίδουμε το σύμβολο $A_0 = -3$ (δηλαδή την ακολουθία bit 00) και ας υπολογίσουμε την πιθανότητα ο θόρυβος να μας στείλει στο γειτονικό σύμβολο A_1, δηλαδή να έχουμε $|Z - A_1| \leq \beta$. Η πιθανότητα αυτή γράφεται:

$$P_1 = \Pr\{A_1|x_0 = A_0\} = \Pr\{|Z - A_1| < \beta | x_0 = A_0\} = \Pr\{|w + A_0 - A_1| < \beta | x_0 = A_0\} \tag{95}$$



Εικόνα 10: Οι πιθανότητες P_1 και P_2 στην περίπτωση του 4-PAM.

Δεδομένου ότι $A_0 - A_1 = -2\beta$ η πιθανότητα αυτή γράφεται ως εξής:

$$P_1 = \Pr\{\beta < w < 3\beta\} = Q\left(\frac{\beta}{\sigma}\right) - Q\left(\frac{3\beta}{\sigma}\right) \tag{96}$$

Με τον ίδιο τρόπο μπορούμε να υπολογίσουμε την πιθανότητα να βρεθούμε από το A_0 στο επόμενο μη γειτονικό σύμβολο A_2,

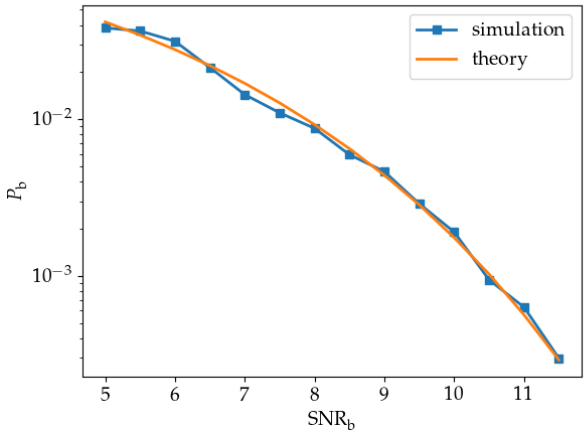
$$P_2 = \Pr\{3\beta < w < 5\beta\} = Q\left(\frac{3\beta}{\sigma}\right) - Q\left(\frac{5\beta}{\sigma}\right) \tag{97}$$

```
import numpy as np
import matplotlib.pyplot as plt
from utils import Q

M = 4                               # PAM order
m = np.log2(M)                      # bits per symbol
SNRbdBs = np.linspace(4, 12, 100) # Specified SNR per bit
SNRb = 10 ** ( SNRbdBs / 10)
q = 6 * SNRb * m / (M ** 2.0 - 1)
PE = 2 * (M-1) / M * Q ( np.sqrt(q) )

P1 = Q ( np.sqrt(q) ) - Q ( 3 * np.sqrt(q) )
P2 = Q ( 3 * np.sqrt(q) ) - Q ( 5 * np.sqrt(q) )

plt.close('all')
plt.figure()
plt.loglog(PE, P1, label = '$P_1$')
plt.loglog(PE, P2, label = '$P_2$')
plt.legend()
plt.xlabel(u'$\Pr\{E\}$')
```

Listing 7: compP.py

Έχει ενδιαφέρον να υπολογίσουμε τις πιθανότητες P_1 και P_2 χρησιμοποιώντας το listing 7. Στην εικόνα 10 συγκρίνουμε τις πιθανότητες αυτές και παρατηρούμε ότι για $\Pr\{E\} \leq 0.1$, η πιθανότητα P_1 είναι πολύ μεγαλύτερη από την P_2 και επομένως είναι πολύ πιο πιθανό ο θόρυβος να μας "πετάξει" σε ένα γειτονικό σύμβολο από ότι σε ένα μη γειτονικό όπως το A_2. Αυτό είναι ένα γενικότερο συμπέρασμα που ισχύει ανεξαρτήτως της τάξης M του PAM ή του αρχικού συμβόλου x_0. Με πολύ καλή προσέγγιση μπορούμε να θεωρήσουμε ότι όταν συμβαίνει σφάλμα συμβόλου τότε μεταβαίνουμε σε γειτονικό σύμβολο το οποίο αν χρησιμοποιήσουμε κωδικοποίηση Gray θα διαφέρει κατά ένα bit από το αρχικό. Αν θεωρήσουμε ότι μεταδίδουμε Q σύμβολα τότε αν το Q είναι πολύ μεγάλο το πλήθος των εσφαλμένων συμβόλων θα είναι $\cong Q\Pr\{E\}$ και εφόσον για κάθε εσφαλμένο σύμβολο έχουμε ένα εσφαλμένο bit ο αριθμός των εσφαλμένων bit θα είναι και αυτός $\cong Q\Pr\{E\}$. Ο συνολικός αριθμός bit που μεταδίδεται είναι $Q\log_2 M$. Η πιθανότητα σφάλματος bit υπολογίζεται ως ο λόγος του αριθμού των εσφαλμένων bit προς το συνολικό αριθμό bit, δηλαδή:

$$P_{\mathrm{b}} \cong \frac{Q\Pr\{E\}}{Q\log_2 M} = 2\frac{M-1}{M\log_2 M}Q\left(\sqrt{\frac{6\log_2 M}{M^2-1}\mathrm{SNR}_{\mathrm{b}}}\right) \tag{98}$$



Εικόνα 11: Σύγκριση της αναλυτικής έκφρασης και του αριθμητικού αποτελέσματος για το 4-PAM.

```
import numpy as np
import matplotlib.pyplot as plt
from commlib import pam_link

M = 4                           # PAM order
SNRbdBs = np.arange(5, 12, 0.5) # Specified SNR per bit
Nbits = 1000                    # Number of bits
samples_per_symbol = 10         # samples per bit
Pbs = np.zeros( SNRbdBs.size )
Pbt = np.zeros( SNRbdBs.size )

error_th = 100

for i, SNRbdB in enumerate(SNRbdBs):
    errors = 0
    bit_counter = 0
    while (errors <= error_th):
        link = pam_link(M = M,
                        SNRbdB = SNRbdB,
                        samples_per_symbol = samples_per_symbol,
                        Rb = 2e9,
                        Nbits = Nbits)

        link.simulate()
        errors += link.errors
        bit_counter += Nbits
        Pb = errors / bit_counter
        print(' SNRb[dB]=%6.2f dB, errors=%d, transmitted bits=%d, Pb = %e' %(SNRbdB, errors,
    bit_counter, Pb) )
    Pbs[i] = Pb
    Pbt[i] = link.c.ber()

plt.close('all')
plt.figure()
plt.semilogy( SNRbdBs, Pbs, '-s', label = 'simulation' )
plt.semilogy( SNRbdBs, Pbt, label = 'theory' )
plt.xlabel('$\mathrm{SNR}_\mathrm{b}$')
plt.ylabel('$P_\mathrm{b}$')
plt.legend()
```

Listing 8: decode.py

Χρησιμοποιούμε το listing 8 για να επιβεβαιώσουμε την πιθανότητα σφάλματος (98). Στην ουσία πραγματοποιούμε προσομοίωση Monte Carlo χρησιμοποιώντας την κλάση `pam_link` μεταδίδοντας ομάδες από 1000 bit μέχρις ούτω να συμβούν περισσότερα από 100 σφάλματα. Αν N_{be} είναι ο αριθμός των εσφαλμένων bit και N ο συνολικός αριθμός των bit που μεταδόθηκαν τότε η πιθανότητα σφάλματος bit προσεγγίζεται από την σχέση:

$$P_{\mathrm{b}} \cong \frac{N_{\mathrm{be}}}{N} \tag{99}$$

Στην εικόνα 11 δείχνουμε το αποτέλεσμα της σύγκρισης των δύο μεθόδων από όπου προκύπτει η πολύ καλή

συμφωνία της (98) με την προσομοίωση Monte Carlo.