ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
(Τ.Ε.Ι.) ΛΑΜΙΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΔΙΔΑΚΤΙΚΕΣ ΣΗΜΕΙΩΣΕΙΣ

ΕΡΓΑΣΤΗΡΙΟ: «ΘΕΩΡΙΑ ΠΛΗΡΟΦΟΡΙΑΣ-ΚΩΔΙΚΕΣ»

Δρ. ΒΑΡΖΑΚΑΣ ΠΑΝΑΓΙΩΤΗΣ

ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΛΑΜΙΑ

ΜΑΙΟΣ 2006

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Οι διδακτικές σημειώσεις που ακολουθούν έχουν γραφτεί για να καλύψουν τις ανάγκες του Εργαστηρίου "**Θεωρία Πληροφορίας-Κώδικες**" του Ε΄ εξαμήνου του προγράμματος σπουδών του Τμήματος Ηλεκτρονικής του Τ.Ε.Ι. Λαμίας. Η συγγραφή αυτών των διδακτικών σημειώσεων κρίθηκε απαραίτητη διότι ένας ηλεκτρονικός θα πρέπει να έχει τη δυνατότητα να εφαρμόζει τη θεωρία πληροφορίας και τους κώδικες σε πρακτικά συστήματα επικοινωνιών και να διαχειρίζεται τις διαφορετικές πηγές πληροφορίας που εμφανίζονται στα διάφορα συστήματα μετάδοσης. Με δεδομένη τη δυσκολία ύπαρξης αντίστοιχου υλικού (hardware), στα θέματα που παρουσιάζονται στις σημειώσεις που ακολουθούν, χρησιμοποιείται κατάλληλο λογισμικό (πρόγραμμα MATLAB).

Η θεωρία πληροφορίας (information theory) βρίσκει εφαρμογές όχι μόνο στην ηλεκτρονική, τις τηλεπικοινωνίες, τη θερμοδυναμική και τη βιολογία (δηλαδή τις θετικές επιστήμες) αλλά και στις οικονομικές και κοινωνικές επιστήμες και στη φιλοσοφία. Πατέρας της θεωρίας πληροφορίας, θεωρείται παγκοσμίως ο Claude Shannon ο οποίος ως ερευνητής σε διάστημα μόλις 13 χρόνων συνέταξε περισσότερα από 1000 άρθρα. Από τα πρώτα άρθρα του Claude Shannon το 1948 και μετέπειτα, η θεωρία πληροφορίας αποτελεί ένα ανεξάρτητο πεδίο έρευνας του οποίου τα συμπεράσματά του βρίσκουν συνεχώς πολλαπλές εφαρμογές.

Τέλος, θα πρέπει να ειπωθεί από το συγγραφέα, ότι η επέκταση και τυχόν διορθώσεις και υποδείξεις στις παρούσες σημειώσεις είναι πάντα ευπρόσδεκτες από τους συναδέλφους και τους φοιτητές του τμήματος. Επίσης, η διεθνής βιβλιογραφία στα θέματα που παρουσιάζονται στις σημειώσεις είναι τεράστια αλλά στο τέλος των σημειώσεων δώθηκαν οι πιο σημαντικές πηγές που χησιμοποιήθηκαν από το συγγραφέα.

Δρ. Βαρζάκας Παναγιώτης
Επίκουρος Καθηγητής
Τμήμα Ηλεκτρονικής
Τ.Ε.Ι Λαμίας

ΕΙΣΑΓΩΓΗ: ΕΙΣΑΓΩΓΗ ΣΤΟ MATLAB

Το λογισμικό που θα μας βοηθήσει σε αυτό το Εργαστήριο είναι το λογισμικό πακέτο αριθμητικών υπολογισμών **MATLAB** (**http://www.mathworks.com/products/matlab**). Το MATLAB (**MAT**rix **LAB**oratory) είναι ένα περιβάλλον πουπροσφέρει δυνατότητες για μαθηματικούς υπολογισμούς, ανάλυση δεδομένων, γραφικές απεικονίσεις,ανάπτυξη αλγορίθμων, εξομοίωση και μοντελοποίηση συστημάτων σε συνδυασμό με μία υψηλού επιπέδου γλώσσα προγραμματισμού.

Βασικές εντολές-Οδηγίες

Στη συνέχεια δίνονται κάπις βασικές εντολές του MATLAB,οι οποίες θα είναι χρήσιμες στις ασκήσεις του εργαστηρίου που ακολουθούν.

- Μπορούμε να παρακαλουθήσουμε την εισαγωγή στο MATLAB, πληκτρολογώντας intro στο prompt του MATLAB
- Υπάρχει η δυνατότητα στο MATLAB να παρέχεται βοήθεια μέσω των επόμενων εντολών:

  Help
  Help plot

- Οι γραφικές παραστάσεις πραγματοποιούνται εύκολα μέσω της εντολής plot η οποία αναπαρασταίνει ένα διάνυσμα y ως προς ένα διάνυσμα x. Π.χ. γράψτε και εκτελέστε το παρακάτω πρόγραμμα:

```
x=[-3 –1 0 1 3 ];
y=x.*x-3*x ;
plot(x,y)
z=x+y*sqrt(-1) ;
plot(z)
```

- Χρησιμοποιήστε τον ενσωματωμένο editor του MATLAB (File→New→mfile, γράψιμο του κώδικα και μετά αποθήκευση στον κατάλογο bin του MATLAB με το όνομα που θέλετε π.χ. program.m) για να δημιουργήσετε ένα αρχείο script με το όνομα program.m

- Εντολή echo off: όχι εμφάνιση των εντολών που έχουν εκτελεστεί στην έξοδο (οθόνη)

- plot(x, y), grid on : εμφάνιση πλέγματοςστη γραφική παράσταση που θα παραχθεί.

- Εισαγωγή ονομασιών (label) (“λεζάντες”) στους δύο άξονες x και y με τις επόμενες εντολές:

```
xlabel('megethos toy aksona x');
ylabel("megethos toy aksona y');
```

- Η συνέχιση της εκτέλεσης του προγράμματος μετά από το πάτημα οποιουδήποτε κουμπιού μέσω της εντολής:

pause % Press a key to see….

- Απαίτηση ο άξονας x (ή ο y) να είναι λογαριθμικός μέσω της εντολής:

semilogx(x, y)

- μία μεταβλητή μπορεί να λάβει πεπερασμένες τιμές σε διαστήματα τα οποία ορίζονται από εμάς, και με δοσμένο βήμα αύξησης της όπως παρουσιάζεται στην επόμενη εντολή για παράδειγμα γα τη μεταβλητή w:

w=[1:10,12:2:100,105:5:500, 510:10:5000,5025:25:20000,20050:50:100000];

- ορισμός και εισαγωγή στοιχείων ενός πίνακα (Pinakas) (κατά γραμμές):

Pinakas=[1 1 0 1 0 0 0; 0 1 1 0 1 0 0; 1 1 1 0 0 1 0; 1 0 1 0 0 0 1]

- Εισαγωγή τίτλου στα γραφληματα που παράγονται μέσω της επόμενης εντολής π.χ.:

title('Titlos ths grafikhs parastashs')

- η εισαγωγή σχολίων (Comment) μέσα στο πρόγραμμα γίνεται αφού πρώτα γραφτεί ο χαρακτήρας % και μετά ακολουθήσει το σχόλιο (δηλαδή: %σχολιο…..). Το τμήμα αυτό του προγράμματος δεν θα εκτελεστεί. Η εισαγωγή σχολίου σε μία γραμμή μπορεί να γίνει και άμεσα με το δεξί πλήκτρο του ποντικιού επιλέγοντας Comment
- Η εκτέλεση του προγράμματος γίνεται μέσω της εντολής: Run (ή με το πλήκτρο F5)
- η έξοδος των αποτελεσμάτων παρουσιάζεται στο παράθυρο Command Window του MATLAB
- οι γραφικές παραστάσεις που παράγονται από τα προγράμματα μπορούν να αποθηκευθούν ως αρχεία με προέκταση fig π.χ. ονομα.fig

Τέλος, θα πρέπει να επιπωθεί ότι στο Εργαστήριο απαιτείται η χρησιμοποίηση έκδοσης του MATLAB 6.0 ή νεώτερης.

ΑΣΚΗΣΗ 1η

Υπολογισμός εντροπίας πηγής πληροφορίας χωρίς μνήμη

1.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι να υπολογιστεί η *εντροπία* (entropy) μιας πηγής πληροφορίας χωρίς μνήμη δύο δυαδικών συμβόλων και να πραγματοποιηθεί η αναπαράστασή της σε συνάρτηση με την πιθανότητα εμφάνισης των συμβόλων.

1.2 Θεωρητικό μέρος

Ας θεωρήσουμε μία πηγή πληροφορίας στην οποία οι πιθανότητες εμφάνισης των συμβόλων της είναι ανεξάρτητες του χρόνου δηλαδή θεωρούμε μία *στατική πηγή* πληροφορίας (πηγή χωρίς μνήμη). Οι πηγές πληροφορίας χωρίς μνήμη, χαρακτηρίζονται από την εντροπία τους $H_{\text{πηγή χωρίς μνήμη}}$ ή μέση κατά σύμβολο πληροφορία που δίνεται από τη παρακάτω σχέση, [1-3]:

$$H_{\text{πηγή χωρίς μνήμη}} = \sum_{i=1}^{n} p_i \cdot \log_2\left(\frac{1}{p_i}\right) \tag{1.1}$$

όπου στη σχέση (1.1), *n* είναι το πλήθος των συμβόλων της πηγής και p_i η πιθανότητα εμφάνισης του αντιστοίχου συμβόλου *i*.

Ας θεωρήσουμε μία πηγή χωρίς μνήμη δύο μόνο συμβόλων. Αν η πιθανότητα εμφάνισης του ενός συμβόλου είναι p, του άλλου συμβόλου θα είναι (1-p). Συνεπώς σύμφωνα με τη σχέση (1.1), η εντροπία της συγκεκριμένης πηγής πληροφορίας θα δίνεται από τη σχέση:

$$H_{\text{πηγή δύο συμβόλων}} = p \cdot \log_2\left(\frac{1}{p}\right) + (1-p) \cdot \log_2\left(\frac{1}{1-p}\right) \tag{1.2}$$

Η εντροπία της προηγούμενης πηγής γίνεται *μέγιστη* όταν οι πιθανότητες εμφάνισης των δύο συμβόλων είναι ίσες δηλαδή όταν τα δύο σύμβολα της πηγής είναι ισοπίθανα. Η προηγούμενη παρατήρηση ισχύει γενικά για οποιαδήποτε πηγή πληροφορίας δηλαδή η σχέση (1.1) παρουσιάζει μέγιστο (μέγιστο της εντροπίας) όταν όλα τα *n* σύμβολα της πηγής είναι ισοπίθανα, [2].

1.3 Εργαστηριακό μέρος

Να γράψετε στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης της εντροπίας πηγής πληροφορίας χωρίς μνήμη δύο συμβόλων. Να παρατηρήσετε αν η εντροπία έχει μέγιστη τιμή και ποια είναι αυτή η τιμή. Να εξηγήσετε το αποτέλεσμα. Η αναπαράσταση να πραγματοποιηθεί σε συνάρτηση με την πιθανότητα εμφάνισης p του ενός από τα δύο σύμβολα της πηγής πληροφορίας.

Πρόγραμμα

```
function H=Entropy(p)
echo off
p=[0:0.1:1];
for i=1:11
H(i)=-p(i)*log2(p(i))-(1-p(i))*log2(1-p(i));
```

```
End

pause % Press a key to see a plot of entropy versus probability of binary alphabet
clf
plot(p, H), grid on;
xlabel('Probability p of the binary symbol');
ylabel('Binary Entropy H ');
```

Εξήγηση Προγράμματος

Στην αρχή του προγράμματος απαιτούμε να μην εμφανίζονται πάλι στην έξοδο οι εντολές που εκτελούνται χρησιμοποιώντας την εντολή echo off. Είναι γνωστό, από τη θεωρία, ότι η πιθανότητα εμφάνισης *p* ενός συμβόλου μιας πηγής πληροφορίας, λαμβάνει τιμές μεταξύ του 0 και του 1. Συνεπώς στο πρόγραμμα υπολογισμού της εντροπίας μιας πηγής πληροφορίας χωρίς μνήμη δύο συμβόλων, θα πρέπει να ορίσουμε μία μεταβλητή p η οποία θα λαμβάνει τις αντίστοιχες τιμές. Αυτό πραγματοποιείται μέσω της εντολής:

```
p=[0:0.1:1];                                                    (1.3)
```

Είναι προφανές ότι δίνουμε διακριτές τιμές στη μεταβλητή p διότι στη συνέχεια θα προχωρήσουμε σε αντίστοιχη γραφική αναπαράσταση. Το βήμα αύξησης της μεταβλητής p θεωρήσαμε ότι είναι ίσο με 0.1. Θα μπορούσαμε να θεωρήσουμε και μεγαλύτερη τιμή για το βήμα αύξησης της μεταβλητής p, αλλά θα πρέπει να προχωρήσουμε στη συνέχεια σε γραφική παράσταση η οποία θα πρέπει να έχει ικανοποιητικές τιμές για τις δύο μεταβλητές (δηλαδή να είναι ικανοποιητικό το πλήθος των τιμών των δύο αξόνων x και y).

Με δεδομένο ότι η μεταβλητή p λαμβάνει διακριτές τιμές, ορίσαμε μία συνάρτηση H η οποία θα αναπαρασταίνει την εντροπία της πηγής και η οποία θα λαμβάνει αντίστοιχα διακριτές τιμές. Αυτό πραγματοποιείται μέσω της εντολής:

```
H(i)=-p(i)*log2(p(i))-(1-p(i))*log2(1-p(i));                    (1.4)
```

Με δεδομένο το βήμα αύξησης της πιθανότητας εμφάνισης του συμβόλου p, να είναι ίσο με 0.1 και τις τιμές της από 0 έως και 1, η εντροπία της πηγής H θα λαμβάνει διακριτές τιμές οι οποίες θα πρέπει να υπολογίζονται κάθε φορά για την αντίστοιχη τιμή της πιθανότητας p. Η διαδικασία αυτή απαιτεί τη δημιουργία ενός *βρόχου επανάληψης* (loop). Ο βρόχος επανάληψης θα εκτελείται, με δεδομένο το βήμα αύξησης της πιθανότητας p να είναι ίσο με 0.1, έντεκα φορές. Σε κάθε εκτέλεση θα υπολογίζεται μία τιμή για την εντροπία H αντίστοιχη της τιμής της πιθανότητας εμφάνισης του συμβόλου p. Η προηγούμενη διαδικασία πραγματοποιείται συνολικά με τον παρακάτω βρόχο επανάληψης:

```
for i=1:11
H(i)=-p(i)*log2(p(i))-(1-p(i))*log2(1-p(i));                    (1.5)
End
```

Η εντολή:

```
pause % Press a key to see a plot of entropy versus probability of binary alphabet(1.6)
```

προχωρεί στη φάση της γραφικής παράστασης της εντροπίας H σε συνάρτηση με την πιθανότητα p μετά από το πάτημα ενός κουμπιού.

Η γραφική παράσταση της εντροπίας Η σε συνάρτηση με την πιθανότητα εμφάνισης του ενός συμβόλου p, πραγματοποιείται με τη βοήθεια της εντολής *plot(x,y):*

plot(p, H), grid on; (1.6)

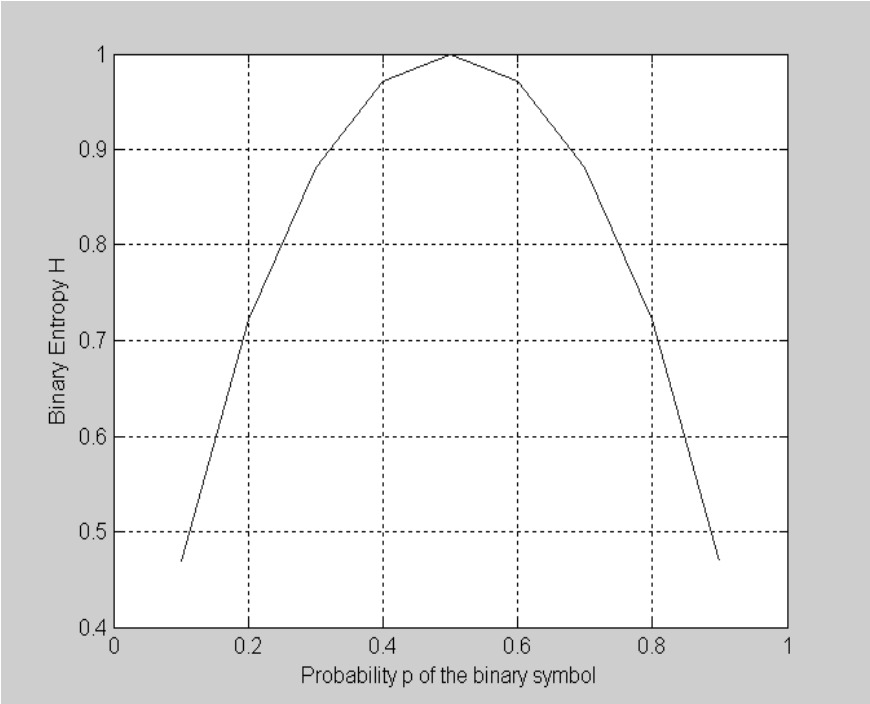
Επίσης, έχουμε ζητήσει από το MATLAB να εμφανίσει "πλέγμα" (*grid on*) στη γραφική παράσταση που θα παράγει.

Τέλος, στην γραφική παράσταση της εξόδου δίνουμε ονομασίες (label) ("λεζάντες") στους δύο άξονες x και y με τις εντολές:

xlabel('Probability p of the binary symbol');
ylabel('Binary Entropy H '); (1.7)

Έξοδος προγράμματος

Η έξοδος του προγράμματος που αναλύθηκε προηγουμένως, στο MATLAB, [4], παρουσιάζεται στην επόμενη εικόνα 1.1.



Εικόνα 1.1 *Εντροπία πηγής χωρίς μνήμη* (Binary entropy, H) δύο συμβόλων σε συνάρτηση με την *πιθανότητα εμφάνισης του ενός δυαδικού συμβόλου* p (Probablity of the binary symbol, p)(*Έξοδος από εκτέλεση του προγράμματος στο MATLAB).*

.

1.4 Πρόσθετες εργασίες

1. Από την έξοδο του προγράμματος να βρείτε για ποια τιμή της πιθανότητας εμφάνισης του συμβόλου p η εντροπία της πηγής γίνεται μέγιστη και να βρείτε την αντίστοιχη μέγιστη τιμή της.
2. Να γράψετε πρόγραμμα υπολογισμού της εντροπίας πηγής χωρίς μνήμη στην οποία εκπέμπονται τρία σύμβολα με αντίστοιχες πιθανότητες εμφάνισης 0.2, 0.3, 0.5.

ΑΣΚΗΣΗ 2η

Υπολογισμός χωρητικότητας καναλιού για συμμετρικό δυαδικό κανάλι σε συνάρτηση με το λόγο ενέργειας ανά bit προς τη φασματική πυκνότητα θορύβου του καναλιού για διαμόρφωση BPSK

2.1 Σκοπός της άσκησης

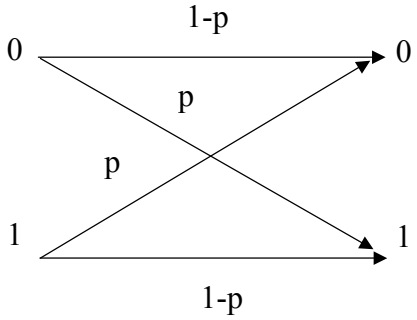
Σκοπός της άσκησης είναι να υπολογιστεί και να αναπαρασταθεί η χωρητικότητα συμμετρικού δυαδικού καναλιού για την περίπτωση διαμόρφωσης BPSK σε συνάρτηση με το λαμβανόμενο λόγο της ενέργειας ανά bit προς τη φασματική πυκνότητα του θορύβου του καναλιού επικοινωνίας.

2.2 Θεωρητικό μέρος

Η *χωρητικότητα καναλιού* (channel capacity) εξ ορισμού είναι ο μέγιστος ρυθμός μετάδοσης δεδομένων με αξιοπιστία σε ένα κανάλι επικοινωνίας, [5]. Αξιόπιστη μετάδοση δεδομένων σε ένα κανάλι επικοινωνίας, είναι δυνατή όταν μπορεί να βρεθεί ένας κώδικας με ικανοποιητικά μεγάλο μήκος, έτσι ώστε η *πιθανότητα λάθους* (probability of error) να τείνει σε μηδενική τιμή όταν το μήκος του κώδικα αυξάνεται συνεχώς, [6-9]. Η χωρητικότητα καναλιού συμβολίζεται συνήθως με C (με μονάδες bits/sec) και ρυθμοί μετάδοσης δεδομένων R (bits/sec) με αξιοπιστία στο κανάλι είναι δυνατοί, μόνο όταν ισχύει:

$$R < C \tag{2.1}$$

Γενικά, με τη χρησιμοποίηση ενός καναλιπύ επικοινωνίας εκπέμπουμε το σήμα πληροφορίας προς έναν ή περισσότερους προορισμούς. Το σήμα πληροφορίας κατά την εκπομπή του στο κανάλι, υπόκειται σε ντετερμινιστικές ή όχι αλλαγές. Για παράδειγμα ντετερμινιστικές αλλαγές είναι η *εξασθένιση* (attenuation), η *γραμμική και μη γραμμική παραμόρφωση* (linear and non-linear distortion) κ.α. Αλλαγές στο σήμα πληροφορίας κατά η μετάδοσή του μπορεί να περιγράφονται πιθανοκρατικά, όπως για παράδειγμα η πρόσθεση *θορύβου* (noise), οι *διαλείψεις πολυδιόδευσης* (multipath fading) κ.α., [5]. Δεδομένου ότι οι ντετερμινιστικές αλλαγές μπορούν να θεωρηθούν ως ειδικές περιπτώσεις τυχαίων αλλαγών, το κανάλι επικοινωνίας στη γενική περίπτωση περιγράφεται από τη σχέση μεταξύ των σημάτων εισόδου και των σημάτων εξόδου. Στην απλούστερη περίπτωση, το κανάλι επικοινωνίας μοντελοποιείται ως μία υπό συνθήκη πιθανότητα μεταξύ της εισόδου του καναλιού και της εξόδου του καναλιού. Ένα τέτοιο μοντέλο καναλιού ονομάζεται *διακριτό κανάλι* χωρίς *μνήμη* (Discrete Memoryless Channel, DMC) και περιγράφεται από το αλφάβητο (σύνολο συμβόλων) της εισόδου X, το αλφάβητο (σύνολο συμβόλων) της εξόδου Y και τον *πίνακα πιθανοτήτων μετάβασης* (channel transition probability matrix) $p(x|y)$, δεδομένων όλων των $x \in X$ και $y \in Y$. Ειδική περίπτωση του διακριτού καναλιού χωρίς μνήμη, είναι το *δυαδικό συμμετρικό κανάλι* (Binary Symmetric Channel, BSC). Το συμμετρικό δυαδικό κανάλι αντιστοιχεί στην περίπτωση στην οποία εκπέμπονται μόνο δύο σύμβολα εισόδου, τα 0 και 1, δηλαδή $X = Y = \{0,1\}$ και $p(y=0|x=1) = p(y=1|x=0) = \mathrm{p}$ (*πιθανότητα λάθους στο εκπεμπόμενο σύμβολο*) (crossover probability). Σχηματικά, το δυαδικό συμμετρικό κανάλι παρουσιάζεται στην επόμενη εικόνα 2.1.



Εικόνα 2.1 Αναπαράσταση δυαδικού συμμετρικού καναλιού (BSC).

Για την περίπτωση ενός δυαδικού συμμετρικού καναλιού, η χωρητικότητα καναλιού δίνεται από τη σχέση:

$$C = 1 - H_{\text{πηγή χωρίς μνήμη}}(p) \tag{2.2}$$

όπου p είναι η πιθανότητα λάθους (crossover probability) στο εκπεμπόμενο σύμβολο (δυαδικό) και $H_{\text{πηγή χωρίς μνήμη}}(p)$ είναι η εντροπία της πηγής πληροφορίας. Για την περίπτωση της δυαδικής διαμόρφωσης BPSK, η πιθανότητα λάθους p συνδέεται με το λόγο γ ενέργειας ανά bit E_b προς τη *φασματική πυκνότητα θορύβου* (noise spectral density) του *προσθετικού λευκού* και *κατανομής Gauss θορύβου* (Additive White Gaussian Noise, AWGN) του καναλιού N_0 (energy per bit per N_0), [5,10], με τη σχέση:

$$p = Q\left(\sqrt{2\gamma}\right) = \frac{1}{2} \cdot erfc\left(\sqrt{\gamma}\right) \tag{2.3}$$

όπου στη σχέση (2.3), *erfc*(x) είναι η *συνάρτηση λάθους* (Error Function), [5,10,11], και *Q(x)* είναι η παρακάτω συνάρτηση (ολοκλήρωμα):

$$Q(x) = \int_{x}^{\infty} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{z^2}{2}} dz \tag{2.4}$$

Η σχέση (2.3) προκύπτει χρησιμοποιώντας την ιδιότητα μεταξύ των *Q(x)* και *erfc(x)*:

$$Q\left(\sqrt{2}x\right) = \frac{erfc(x)}{2} \tag{2.5}$$

Η φασματική πυκνότητα θορύβου του καναλιού N_0, περιγράφει το μέγεθος της ισχύος του καναλιοού επικοινωνίας AWGN και δίνεται σε (Watt/Hz) για ένα συγκεκριμένο κανάλι. Θα πρέπει να επιπωθεί ότι προσθετικός λευκός θόρυβος Gauss είναι ο θόρυβος ο οποίος προστίθεται στο στιγμιαίο πλάτος του εκπεμπομένου σήματος, παρουσιάζει ισοκατανομή τησισχύος σε όλες τις συχνότητες (λευκός θόρυβος) και οι στιγμιαίες τιμές του πλάτους του ακολουθούν την κατανομή Gauss (κανονική κατανομή), [10].

2.3 Εργαστηριακό μέρος

Να γράψετε στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης της χωρητικότητας συμμετρικού δυαδικού καναλιού σε συνάρτηση με το λόγο $\gamma=\frac{E_b}{N_0}$ της ενέργειας ανά bit E_b προς τη φασματική πυκνότητα θορύβου του καναλιού N_0.

Πρόγραμμα

```
echo off
gamma_db=[-20:1:20];
gamma=10.^(gamma_db/10);
for i=1:41
e(i)=0.5*erfc(sqrt(gamma(i)));
H(i)=-log2(e(i))*(e(i))-log2(1-e(i))*(1-e(i));
c(i)=1-H(i);
end
pause % Press a key to see a plot of channel capacity versus SNR/bit
clf
semilogx(gamma, c), grid on;
xlabel ('Energy per bit per noise spectral efficiency for BPSK');
ylabel('Channel Capacity of BSC for BPSK modulation');
```

Εξήγηση Προγράμματος

Κατ αρχήν είναι αναγκαίο να ορίσουμε τη μεταβλητή $\gamma=\frac{E_b}{N_0}$. Στη μεταβλητή γ δίνουμε τιμές σε dB, από –20dB έως 20dB με βήμα αύξησης ίσο με 1 (πρόκειται για ενδεικτικές τιμές). Αυτό πραγματοποιείται μέσω της εντολής:

```
gamma_db=[-20:1:20];
```
(2.6)

Στη διαδικασία υπολογισμού της χωρητικότητας του δυαδικού συμμετρικού καναλιού, η μεταβλητή γ θα πρέπει να είναι καθαρός αριθμός. Έτσι μετατρέπουμε τη μεταβλητή γ από την έκφραση της σε dB σε καθαρό αριθμό με την παρακάτω εντολή:

```
gamma=10.^(gamma_db/10);
```
(2.7)

διότι ισχύει:

$$\gamma\left(\text{καθαρός αριθμός}\right)=10^{\frac{\gamma(\text{dB})}{10}} \tag{2.8}$$

Σύμφωνα με τη σχέση (2.3), θα πρέπει να υπολογιστεί η πιθανότητα p, στο εκπεμπόμενο σύμβολο. Αυτό γίνεται στο πρόγραμμα με τη βοήθεια της εντολής:

```
e(i)=0.5*erfc(sqrt(gamma(i)));
```
(2.9)

στην οποία η πιθανότητα p δίνεται από τη μεταβλητή e. Η συνάρτηση λάθους στο MATLAB δίνεται από την έκφραση *erfc(x)*.

Στη συνέχεια η εντροπία της πηγής πληροφορίας χωρίς μνήμη υπολογίζεται μέσω της εντολής:

```
H(i)=-log2(e(i))*(e(i))-log2(1-e(i))*(1-e(i));          (2.10)
```

δεδομένης της σχέσης (1.2). Ο υπολογισμός της χωρητικότητας C του δυαδικού συμμετρικού καναλιού πραγματοποιείται με τη βοήθεια της εντολής:

```
c(i)=1-H(i);                                            (2.11)
```

Για κάθε τιμή του λόγου γ θα πρέπει να υπολογίζεται μία αντίστοιχη τιμή της χωρητικότητας του καναλιού C. Για να επιτευχθεί αυτό είναι απαραίτητος ένας βρόχος επανάληψης στον οποίο θα υπολογίζεται ένα συγκεκριμένο πλήθος τιμών (στο πρόγραμμα που παρουσιάστηκε προηγούμενα το πλήθος αυτό είναι 41 (*i=1:41*).

Η γραφική παράσταση παρουσιάζεται μετά από το πάτημα (press) οποιοδήποτε κουμπιού μέσω της εντολής:

```
pause % Press a key to see a plot of channel capacity versus SNR/bit   (2.12)
```

Η γραφική παράσταση παροσυιάζεται έτσι στη συνέχεια δίνοντας με τις δύο επόμενες εντολές, ονομασίες στους δύο άξονες:

```
xlabel('Energy per bit per noise spectral efficiency for BPSK');
ylabel('Channel Capacity of BSC for BPSK modulation');          (2.13)
```

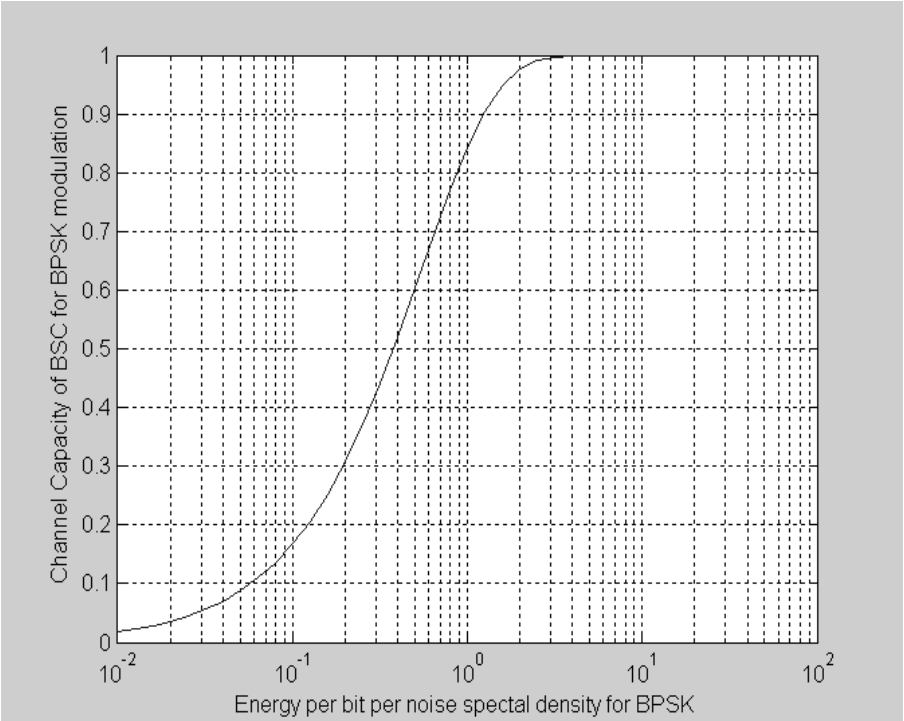
Προηγουμένως έχουμε απαιτήσει ο άξονας x να είναι λογαριθμικός μέσω της εντολής:

```
semilogx(gamma, c), grid on;                            (2.14)
```

Με την προηγούμενη εντολή, επίσης, έχουμε ζητήσει από το MATLAB να εμφανίσει "πλέγμα" (*grid on*) στην γραφική παράσταση που θα παραχθεί.

Έξοδος προγράμματος

Η έξοδος του προγράμματος που αναλύθηκε προηγουμένως, στο MATLAB, παρουσιάζεται στην επόμενη εικόνα 2.1.



Εικόνα 2.1 Χωρητικότητα δυαδικού συμμετρικού καναλιού σε συνάρτηση με το λόγο γ (ενέργεια ανά bit προς φασματική πυκνότητα θορύβου του καναλιού) (*Έξοδος από εκτέλεση του προγράμματος στο MATLAB)* για την περίπτωση BPSK διαμόρφωσης.

2.4 Πρόσθετες εργασίες

1. Δικαιολογείστε μετά την εκτέλεσητης άσκησης, τη μέγιστη τιμή την οποία λαμβάνει η χωρητικότητα καναλιού του δυαδικού συμμετρικού καναλιού.
2. Να γράψετε πρόγραμμα στο MATLAB το οποίο θα υπολογίζει και θα αναπαρασταίνει τη χωρητικότητα καναλιού ενός δυαδικού συμμετρικού καναλιού σε συνάρτηση με την *πιθανότητα λάθους* p στο εκπεμπόμενο σύμβολο (bit) (crossover probability) με δεδομένο ότι $0 \le p \le 1$. Για ποια τιμή του η χωρητικότητα καναλιού είναι ελάχιστη και ποια η ελάχιστη τιμή της χωρητικότητας του καναλιού; Δικαιολογείστε αναλυτικά την απάντησή σας.

ΑΣΚΗΣΗ 3η

Υπολογισμός του ρυθμού λαθών σε συμμετρικό δυαδικό κανάλι σε συνάρτηση με το λόγο ενέργειας ανά bit προς τη φασματική πυκνότητα θορύβου του καναλιού για διαμόρφωση BPSK

3.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι να υπολογιστεί και να αναπαρασταθεί για ένα δυαδικό συμμετρικό κανάλι επικοινωνίας, ο *ρυθμός λαθών* (Bit Error Rate, BER) σε συνάρτηση με το με το λόγο $\gamma=\frac{E_b}{N_0}$, της ενέργειας ανά bit E_b προς τη φασματική πυκνότητα θορύβου του καναλιού N_0. Ο θόρυβος του καναλιού επικοινωνίας θεωρείται *λευκός, προσθετικός θόρυβος κατανομής Gauss* (Additive White Gaussian Noise, AWGN).

3.2 Θεωρητικό μέρος

Για δυαδικό συμμετρικό κανάλι, η πιθανότητα λάθους p (crossover probability) στο εκπεμπόμενο σύμβολο (bit) εξαρτάται από τη χρησιμοποιούμενη διαμόρφωση και από τη τιμή του *λόγου σήματος προς θόρυβο* (Signal-to Noise Ratio, SNR) στη λήψη (έξοδος του καναλιού). Όταν η διαμόρφωση που χρησιμοποιείται είναι BPSK τότε η πιθανότητα λάθους (ή διαφορετικά ο ρυθμός λαθών) δίνεται από τη σχέση, [5, 11]:

$$\text{BER}_{\text{BPSK}} = \text{p} = Q\left(\sqrt{2\gamma}\right) = \frac{1}{2}\cdot erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \qquad (3.1)$$

όπου $\gamma=\frac{E_b}{N_0}$ είναι ο λόγος της ενέργειας E_b ανά λαμβανόμενο σύμβολο (bit) προς τη φασματική πυκνότητα N_0 του θορύβου του καναλιού, μέγεθος αντίστοιχο με το λόγο σήμα προς θόρυβο αλλά στο επίπεδο του εκπεμπόμενου συμβόλου (bit για δυαδικό σύστημα σηματοδοσίας).

3.3 Εργαστηριακό μέρος

Να γράψετε στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης του ρυθμού λαθών για δυαδικό συμμετρικό κανάλι σε συνάρτηση με το λόγο $\gamma=\frac{E_b}{N_0}$ της ενέργειας ανά bit E_b προς τη φασματική πυκνότητα του θορύβου του καναλιού N_0.

Πρόγραμμα

```
gamma_db=[-50:0.1:50];
gamma=10.^(gamma_db./10);
err=0.5*erfc(sqrt(gamma));
clf
semilogx(gamma_db,err), grid on;
xlabel ('Energy per bit per noise spectral density for BSC-BPSK');
ylabel ('BER of BPSK');
```

Εξήγηση Προγράμματος

Κατ αρχήν ορίζουμε τη μεταβλητή $\gamma=\frac{E_b}{N_0}$. Στη μεταβλητή γ δίνουμε τιμές σε dB, από –50dB έως 50dB με βήμα αύξησης ίσο με 0.1 (πρόκειται για ενδεικτικές τιμές). Αυτό πραγματοποιείται μέσω της εντολής:

```
gamma_db=[-50:0.1:50];
```
(3.2)

Με δεδομένη την έκφραση του γ σε dB, μετατρέπουμε το λόγο γ σε καθαρό αριθμό με την εντολή:

```
gamma=10.^(gamma_db./10);
```
(3.3)

Η πιθανότητα λάθους (ρυθμός λαθών) (BER) (μεταβλητή err) υπολογίζεται μέσω της εντολής:

```
err=0.5*erfc(sqrt(gamma));
```
(3.4)

Η γραφική παρουσιάζεται με ονομασίες στους δύο άξονες τις επόμενες:

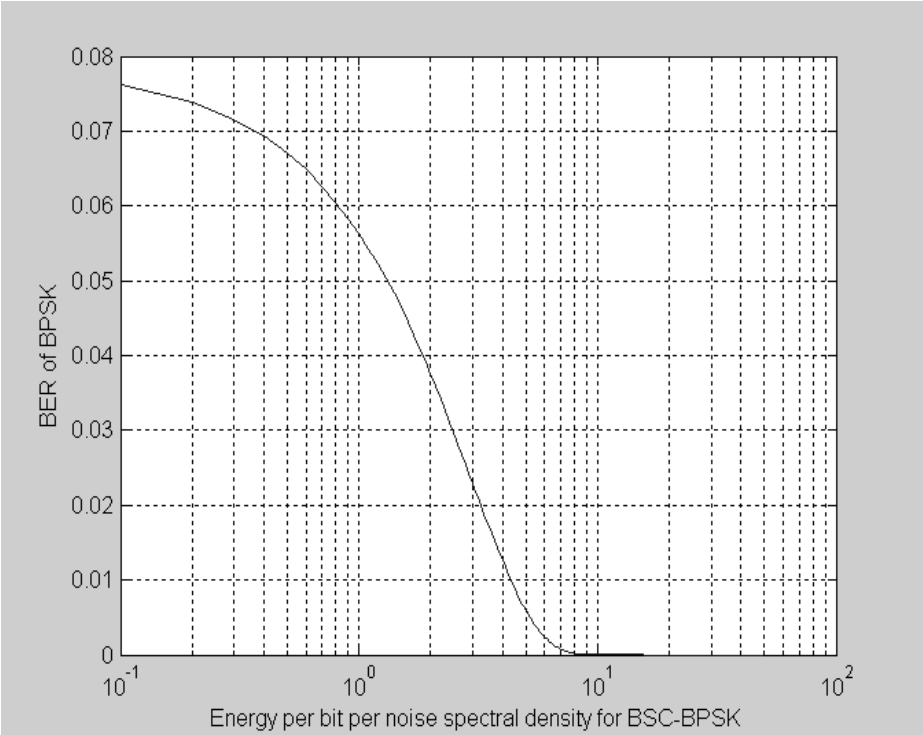
```
xlabel('Energy per bit per noise spectral density for BSC-BPSK');
ylabel('BER of BPSK');
```
(3.5)

Επίσης, έχουμε απαιτήσει ο άξονας x να είναι λογαριθμικός μέσω της εντολής:

```
semilogx(gamma, c), grid on;
```
(3.6)

Με την προηγούμενη εντολή, επίσης, έχουμε ζητήσει από το MATLAB να εμφανίσει "πλέγμα" (*grid on*) στην γραφική παράσταση που θα παραχθεί.

Έξοδος προγράμματος



Εικόνα 3.1 Ρυθμός λαθών (BER) για δυαδικό συμμετρικό κανάλι (BSC) σε συνάρτηση με το λόγο ενέργειας ανά bit προς τη φασματική πυκνότητα θορύβου του καναλιού για διαμόρφωση BPSK (*Έξοδος από εκτέλεση του προγράμματος στο MATLAB).*

3.4 Πρόσθετες εργασίες

1. Ποιος ρυθμός λαθών κατά προσέγγιση, για την περίπτωση καναλιού επικοινωνίας με λόγο ενέργειας ανά bit προς τη φασματική πυκνότητα θορύβου του καναλιού ίσο με 0dB;
2. Να γράψετε πρόγραμμα στο MATLAB το οποίο θα υπολογίζει και θα αναπαρασταίνει το ρυθμό λαθών για δυαδικό συμμετρικό κανάλι με διαμόρφωση DPSK. Σας δίνεται η έκφραση του ρυθμού λαθών BER_{DPSK}, p, για τη διαμόρφωση DPSK, [5,11]:

$$\text{BER}_{\text{DPSK}} = \text{p} = \frac{1}{2} \cdot e^{-\frac{E_b}{N_0}} \tag{3.7}$$

3. Εξηγείστε γιατί η αύξηση του λόγου $\frac{E_b}{N_0}$ οδηγεί σε μείωση του ρυθμού λαθών σε ένα σύστημα διαμόρφωσης BPSK.

ΑΣΚΗΣΗ 4η

Υπολογισμός χωρητικότητας καναλιού για κανάλι περιορισμένου εύρους ζώνης με προσθετικό, λευκό θόρυβο κατανομής Gauss

4.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι να υπολογιστεί και να αναπαρασταθεί η χωρητικότητα ενός καναλιού επικοινωνίας *περιορισμένου εύρους ζώνης συχνοτήτων* (bandlimited channel) σε συνάρτηση με το εύρος ζώνης του καναλιού και με το λόγο σήμα προς θόρυβο (Signal-to-Noise Ratio) στην έξοδο του καναλιού. Ο θόρυβος του καναλιού επικοινωνίας θεωρείται λευκός, προσθετικός, θόρυβος Gauss δηλαδή θεωρούμε κανάλι επικοινωνίας AWGN.

4.2 Θεωρητικό μέρος

Για την επρίπτωση ενός καναλιού επικοινωνίας, εκτός από το μοντέλο του δυαδικού συμμετρικού καναλιού, ένα ενδιαφέρον μοντέλο καναλιού που συναντάται στην πράξη συνεχώς είναι το μοντέλο του καναλιού περιορισμένου εύρους ζώνης συχνοτήτων στο οποίο η εκπεμπόμενη ισχύς του σήματος παρουσιάζει ένα πάνω όριο (περιορισμός της ισχύος εκπομπής σε μία μέγιστη τιμή) και ο θόρυβος είναι AWGN θόρυβος, [10]. Αν υποτεθεί ότι ο θόρυβος έχει φασματική πυκνότητα N_0, η εκπεμπόμενη ισχύς είναι ίση με P και το εύρος ζώνης συχνοτήτων είναι W τότε ο C.Shannon, θεωρούμενος πατέρας της θεωρίας πληροφορίας και παγκοσμίως γνωστός μαθηματικός και ερευνητής [6,7], απέδειξε τη σημαντικότατη επόμενη έκφραση για τη χωρητικότητα C (σε bits/sec) του συγκεκριμένου καναλιού επικοινωνίας:

$$C = W \cdot \log_2\left(1 + \frac{P}{N_0 \cdot W}\right) \tag{4.1}$$

Αποδεικνύεται, ότι όταν ο λόγος $\frac{P}{N_0}$ τείνει στο άπειρο, τότε και η χωρητικότητα του καναλιού C τείνει στο άπειρο δηλαδή η αύξηση της ισχύος εκπομπής μπορεί να οδηγήσει σε αύξηση του ρυθμού εκπομπής των δεδομένων (σε bits/sec). Αντίστοιχα, όταν το εύρος ζώνης του καναλιού W, τείνει στο άπειρο η χωρητικότητα του καναλιού C τείνει σε συγκεκριμένη τιμή η οποία προσδιορίζεται από το λόγο $\frac{P}{N_0}$, όπως φαίνεται από την επόμενη σχέση, [5,11]:

$$\lim_{W\to\infty} C = \lim_{W\to\infty} W \cdot \log_2\left(1 + \frac{P}{N_0 \cdot W}\right) = \frac{P}{N_0 \cdot \ln 2} = 1.4427 \cdot \frac{P}{N_0} \tag{4.2}$$

δηλαδή η αύξηση του εύρους συχνοτήτων του καναλιού απεριόριστα δενοδηγεί σε αντίσοιχη αύξηση του ρυθμού εκπομπής δεδομένων.

4.3 Εργαστηριακό μέρος

1. Να γράψετε στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης της χωρητικότητας καναλιού C με εύρος ζώνης συχνοτήτων W=3000Hz σε συνάρτηση με το λόγο $\frac{P}{N_0}$ (να δώσετε τιμές στο λόγο $\frac{P}{N_0}$ από –20 έως 30dB)(θεωρείστε τη σχέση (4.1).

2. Να γράψετε στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης της χωρητικότητας καναλιού C με λόγο $\frac{P}{N_0}$ ίσο με 25dB σε συνάρτηση με το εύρος ζώνης συχνοτήτων του καναλιού W.

Πρόγραμμα

```
echo on

pn0_db=[-20:0.1:30];

pn0=10.^(pn0_db./10);

capacity=3000.*log2(1+pn0/3000);

pause % Press a key to see a plot of channel capacity versus P/N0

clf

semilogx(pn0, capacity)

title('Capacity vs P/N0 in an AWGN channel')
xlabel('P/No');
ylabel('Channel Capacity (bits/second)');
clear
w=[1:10,12:2:100,105:5:500, 510:10:5000,5025:25:20000,20050:50:100000];
pn0_db=25;
pn0=10.^(pn0_db./10);
capacity=w.*log2(1+pn0./w);
pause % Press a key to see a plot of channel capacity versus bandwith

clf

semilogx(w, capacity), grid on;
title('Capacity vs bandwidth in an AWGN channel')

xlabel('Bandwidth (Hz)');
ylabel('Channel Capacity (bits/second)');
```

Εξήγηση Προγράμματος

Στην αρχή του προγράμματος ορίζουμε το λόγο $\frac{P}{N_0}$ με τη βοήθεια της εντολής:

```
pn0_db=[-20:0.1:30];
```
(4.3)

Στη μεταβλητή *pn0_db* δίνουμε τιμές από –20 έως 30 dB (σύμφωνα με την εκφώνηση της άσκησης). Στη συνέχεια προχωράμε με την επόμενη εντολή, στη μετατροπή των τιμών της μεταβλητής *pn0_db* σε καθαρό αριθμό διότι στη σχέση (4.1) ο λόγος $\frac{P}{N_0}$ πρέπει να είναι εκφρασμένος σε καθαρό αριθμό:

pn0=10.^(pn0_db./10); (4.4)

Η χωρητικότητα (capacity) του καναλιού εύρους ζώνης W=3000Hz, υπολογίζεται με την επόμενη εντολή:

capacity=3000.*log2(1+pn0/3000); (4.5)

εφαρμόζοντας άμεσα τη σχέση (4.1).

Στη συνέχεια με το πάτημα ενός κουμπιού (*pause % Press a key to see a plot of channel capacity versus P/N0*) εμφανίζεται η γραφική παράσταση της χωρητικότητας C του AWGN καναλιού (εκφρασμένη σε bits/sec) σε συνάρτηση με το λόγο $\frac{P}{N_0}$ για την περίπτωση που το εύρος ζώνης του καναλιού είναι ίσο με W=3000Hz (εικόνα 4.1). Η γραφική παράσταση έχει τον άξονα x λογαριθμικό μέσω της εντολής:

semilogx(pn0, capacity) (4.6)

ενώ η όλη γραφική παράσταση (εικόνα 4.1) έχει τίτλο (*title*) με τη βοήθεια της εντολής:

title('Capacity vs P/N0 in an AWGN channel') (4.7)

Στους δύο άξονες x και y, δίνουμε αντίστοιχες ονομασίες μέσω των εντολών:

xlabel('P/No');
ylabel('Channel Capacity (bits/second)'); (4.8)

Μετά από τη συγκεκριμένη έξοδο του προγράμματος, είναι απαραίτητο όλες οι μεταβλητές που έχουν ήδη χρησιμοποιηθεί να πάρουν την τιμή 0. Αυτό πετυχαίνεται με την εντολή:

Clear (4.9)

Έτσι στη συνέχεια προχωρούμε στη πραγματοποίηση της γραφικής παράστασης της χωρητικότητας C του AWGN καναλιού σε συνάρτηση με το εύρος ζώνης συχνοτήτων W του καναλιού. Έτσι δίνουμε αρχικά τιμές στο W με την εντολή:

w=[1:10,12:2:100,105:5:500, 510:10:5000,5025:25:20000,20050:50:100000]; (4.10)

Από την εκφώνηση της άσκησης πρέπει να είναι: $\frac{P}{N_0}$=25dB. Έτσι με τις επόμενες δύο εντολές δίνουμε αυτή τη τιμή στο λόγο $\frac{P}{N_0}$ και τον μετατρέπουμε σε καθαρό αριθμό όπως απαιτεί η σχέση (4.1):

pn0_db=25; (4.11)

```
pn0=10.^(pn0_db./10);
```

Ο υπολογισμός της χωρητικότητας C του καναλιού πραγματοποιείται μέσω της εντολής:

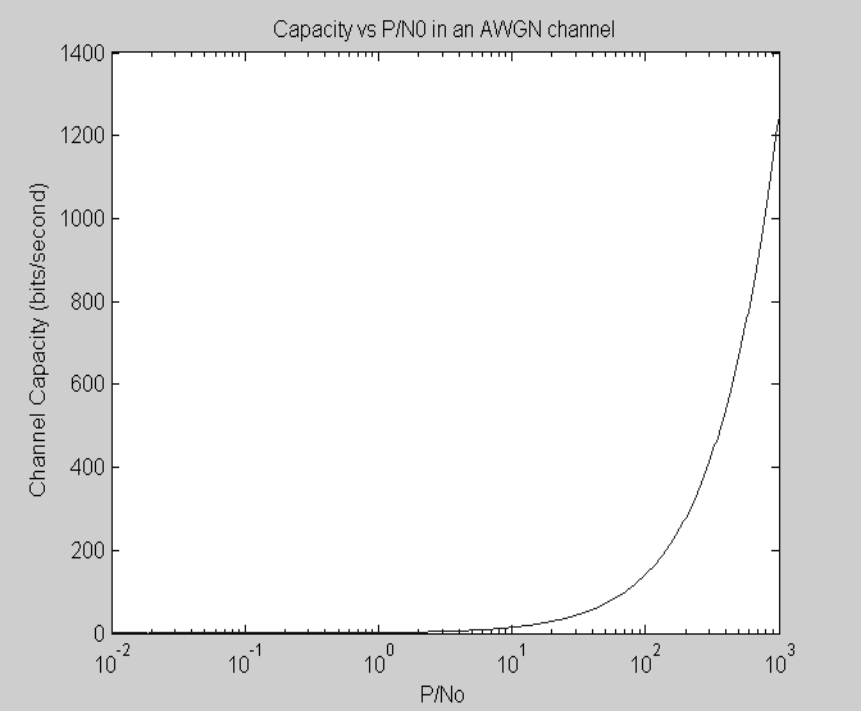
```
capacity=w.*log2(1+pn0./w);                    (4.12)
```

ενώ με το πάτημα ενός κουμπιού (*pause % Press a key to see a plot of channel capacity versus bandwith*) λαμβάνουμε την αντίστοιχη έξοδο (εικόνα 4.2). Ο άξονας x ζητάμε να είναι λογαριθμικός και θέτουμε *ονομασίες* (*label*) στους άξονες και *τίτλο* (*title*) στη γραφική παράσταση μέσω των επόμενων εντολών:

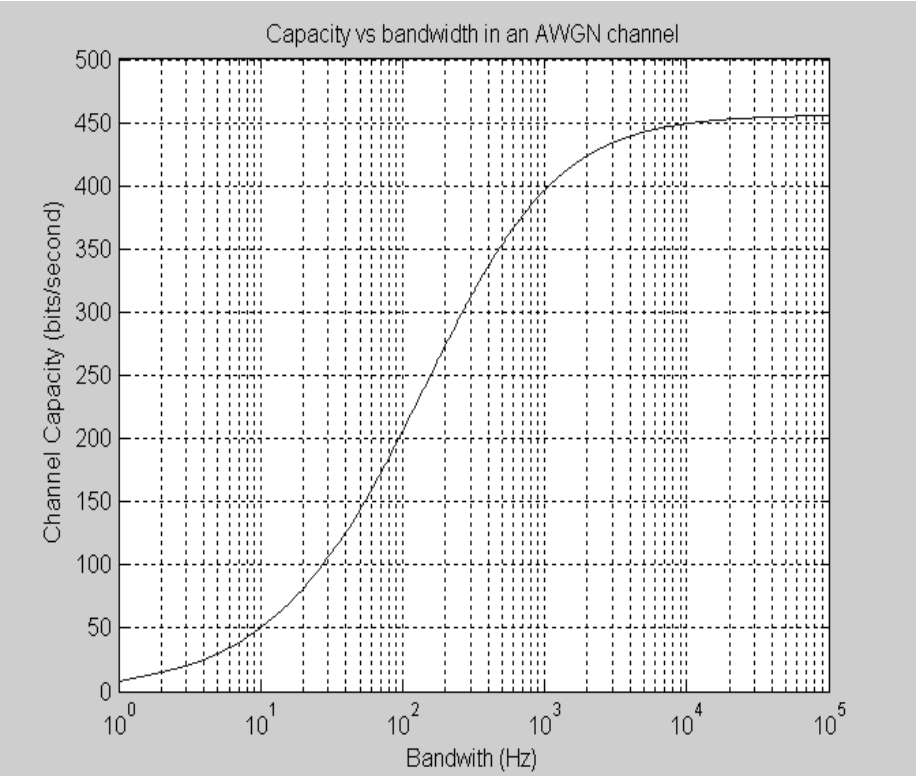
```
semilogx(w, capacity), grid on;
title('Capacity vs bandwidth in an AWGN channel')    (4.13)

xlabel('Bandwidth (Hz)');
ylabel('Channel Capacity (bits/second)');
```

Έξοδος προγράμματος



Εικόνα 4.1 Χωρητικότητα (channel capacity) (σε bits/sec) AWGN καναλιού περιορισμένου εύρους ζώνης σε συνάρτηση με το λόγο $\frac{P}{N_0}$ (εκπεμπόμενη ισχύς προς φασματική πυκνότητα θορύβου του καναλιού επικοινωνίας) για εύρος ζώνης του καναλιού ίσο με W=3000Hz (*Έξοδος από εκτέλεση του προγράμματος στο MATLAB).*



Εικόνα 4.2 Χωρητικότητα (channel capacity) AWGN καναλιού (σε bits/sec) περιορισμένου εύρους ζώνης σε συνάρτηση με το εύρος ζώνης του καναλιού (Bandwidth) (σε Hz) για $\frac{P}{N_0}$=25dB (*Έξοδος από εκτέλεση του προγράμματος στο MATLAB).*

4.4 Πρόσθετες εργασίες

1. Ποια η χωρητικότητα καναλιού ενός AWGN καναλιού επικοινωνάς εύρους ζώνης συχνοτήτων 5000 Hz με λόγο $\frac{P}{N_0}$=25dB;
2. Ποια η χωρητικότητα AWGN καναλιού με λόγο $\frac{P}{N_0}$=30 dB και εύρος ζώνης ίσο με W=3000 Hz;
3. Εξηγείστε αναλυτικά το παρακάτω πρόγραμμα, μετά την εκτέλεσή του, όπως και την αντίστοιχη έξοδό του στο MATLAB:

Πρόγραμμα

```
echo off
w=[1:5:20,25:20:100,130:50:300,400:100:1000,1250:250:5000,5500:500:10000];
pn0_db=[-20:1:30];
pn0=10.^(pn0_db./10);
for i=1:45
   for j=1:51
c(i,j)=w(i)*log2(1+pn0(j)/w(i));
end
end
echo on
k=[0.9,0.8,0.5,0.6];
s=[-70,35];
```

```
surfl(w,pn0_db,c′,s, k)
title('capacity vs.bandwidth and SNR')
```

4. Να αποδείξετε με μαθηματικό τρόπο τη σχέση (4.2) (σας δίνεται ότι $\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^{x} = e$).

ΑΣΚΗΣΗ 5η

Υπολογισμός πιθανότητας λάθους αποκωδικοποίησης σε απλούς κώδικες επανάληψης

5.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι να γραφτεί στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης της *πιθανότητας λάθους* (error probability, P_e) στο αποκωδικοποιημένο bit για *απλούς κώδικες επανάληψης (*simple repetition codes).

5.2 Θεωρητικό μέρος

Η μετάδοση δεδομένων μέσω ενός καναλιού επικοινωνίας το οποίο παρουσιάζει θόρυβο έχει ως χαρακτηριστικό την εμφάνιση *λαθών* (errors) κατά τη λήψη των εκπεμπόμενων συμβόλων (bit). Θα πρέπι να ειπωθεί ότι στην πράξη δεν υάρχει κανάλι επικοινωνίας χωρίς να παρουσιάζει θόρυβο (αθόρυβο) (noiseless). Το προηγούμενο πρόβλημα είναι δυνατό να λυθεί μερικά με την πρόσθεση επιπλέον bits αλλά με την συνεπακόλουθη μείωση του *ρυθμού εκπομπής δεδομένων* (transmission rate). Η πρόσθεση επιπλέον bits μεσκοπό τη μείωση του ρυθμού λαθών στην αποκωδικοποίηση ονομάζεται *κωδικοποίηση* (coding). Η κωδικοποίηση χωρίζεται γενικά σε δύο κατηγορίες: τη *γραμμική*, *μπλοκ* (block) κωδικοποίηση και τη *συγκεραστική* (convolutional) κωδικοποίηση, [1,2].

Στη μπλοκ κωδικοποίηση, οι ακολουθίες της εξόδου της πηγής πληροφορίας (μήνυμα πληροφορίας), με μήκος *k* απεικονίζονται σε δυαδικές ακολουθίες (*κωδικές λέξεις*) (codewords) με μήκος *n* έτσι ώστε ο *ρυθμός* του παραγομένου κώδικα να είναι ίσος με $\frac{k}{n}$ ανά εκπομπή, [1]. Ένας τέτοιος κώδικας ονομάζεται *κώδικας (n,k)* και αποτελείται από 2^k (πλήθος κωδικών λέξεων) κωδικές λέξεις με μήκος *n*, οι οποίες πολλές φορές συμβολίζονται με $c_1, c_2, .. c_{2^k}$.

Στους κώδικες μπλοκ, η απεικόνιση των διαφόρων ακολουθιών εξόδου της πηγής πληροφορίας στις κωδικές λέξεις πραγματοποιείται ανεξάρτητα και η έξοδος του *κωδικοποιητή* (encoder) εξαρτάται μόνο από την τρέχουσα ακολουθία εξόδου της πηγής (ακολουθία εισόδου στον κωδικοποιητή) μήκους *k* και σε καμία περίπτωση από τις προηγούμενες ακολουθίες εισόδου στον αποκωδικοποιητή, [1,2,5].

Αντίθετα, στους συγκεραστικούς κώδικες, αντίθετα, οι ακολουθίες εξόδου της πηγής μήκους k_0 απεικονίζονται σε ακολουθίες μήκους n_0, αλλά σε αυτή τη περίπτωση οι ακολουθίες εξόδου εξαρτώνται όχι μόνο από τις πιο πρόσφατες k_0 ακολουθίες εισαλλά και από τις *(L-1)* k_0 ακολουθίες εισόδου στον κωδικοποιητή, [1,2,5].

Μία πολύ απλή περίπτωση μπλοκ κωδικοποίησης είναι ο *απλός κώδικας επανάληψης* (simple repetition code). Σε έναν απλό κώδικα επανάληψης με τον οποίο επιθυμούμε να εκπέμψουμε τα δυαδικά σύμβολα “0” και “1” σε ένα δυαδικό συμμετρικό κανάλι (BSC), αντί να εκπέμψουμε τα “0” και “1”, εκπέμπουμε αντίστοιχα μία ακολουθία από “0” και “1” στην θέση αντίστοιχα του “0” και “1”. Το μήκος των δύο αυτών ακολουθιών επιλέγεται να είναι ένας *περιττός* (odd) αριθμός *n*. Έτσι, ένας απλός κώδικας επανάληψης μπορεί να παρασταθεί με την παρακάτω αντιστοιχία:

$$\begin{aligned} 0 &\rightarrow \overbrace{00 \ \underbrace{\ldots}_{n \text{ περιττός}} \ 00} \\ 1 &\rightarrow \overbrace{11 \ \underbrace{\ldots}_{n \text{ περιττός}} \ 11} \end{aligned} \tag{5.1}$$

Η διαδικασία της *αποκωδικοποίησης* (decoding) βασίζεται σε μία πλειοψηφική απόφαση: αν η πλειοψηφία (σε πλήθος) των λαμβανομένων συμβόλων, είναι τα "0" τότε αποφασίζεται ότι το εκπεμπόμενο bit είναι το "0". Αν αντίθετα, η πλειοψηφία των λαμβανομένων συμβόλων είναι οι "1" τότε αποφασίζεται ότι το εκπεμπόμενο bit είναι το "1".

Στην διαδικασία της αποκωδικοποίησης, λάθος παρατηρείται όταν τουλάχιστον *(n+1)/2* από τα εκπεμπόμενα bit έχουν ληφθεί λάθος. Αν το κανάλι επικοινωνίας είναι ένα BSC κανάλι, το οποίο εμφανίζει *πιθανότητα λάθους* (crossover probability) στο εκπεμπόμενο bit ίση με ε, τότε η πιθανότητα λάθους αποκωδικοποίησης για έναν απλό κώδικα επανάληψης (n,k) αποδεικνύεται ότι δίνεται από την παρακάτω σχέση, [4,5,11]:

$$p_e = \sum_{k=(n+1)/2}^{n} \binom{n}{k} \cdot \varepsilon^k \cdot (1-\varepsilon)^{n-k} \tag{5.2}$$

Για παράδειγμα, σε έναν απλό κώδικα επανάληψης με n=5 και ε=0.001=10^{-3}, η πιθανότητα λάθους είναι ίση με:

$$p_e = \sum_{k=3}^{5} \binom{5}{k} \cdot 0.001^k \cdot (0.999)^{5-k} = 9.99 \times 10^{-10} \cong 10^{-9} \tag{5.3}$$

Το αποτέλεσμα της σχέσης (5.3), δείχνει ότι αν εκπέμψουμε 5 bits στη θέση του ενός, τότε πετυχαίνουμε μείωση της πιθανότητας λάθους από 0.001(=10^{-3}) σε 10^{-9}. Οπωσδήποτε μείωση της πιθανότητας λάθους έχει σαν αντίτιμο την μείωση του ρυθμού εκπομπής στο κανάλι και την αύξηση της πολυπλοκότητας στο συγκεκριμένο σύστημα. Δηλαδή, στο συγκεκριμένο παράδειγμα, ο ρυθμός εκπομπής μειώνεται από 1 bit κάθε μία φορά που χρησιμοποιούμε το κανάλι επικοινωνίας σε 1 bit κάθε 5 φορές που χρησιμοποιούμε το κανάλι επικοινωνίας. Η πολυπλοκότητα του συστήματος αυξήθηκε διότι η αποκωδικοποίηση σε έναν απλό κώδικα επανάληψης απαιτείται, στην περίπτωση αυτή, να γίνει μέσω ενός αποκωδικοποιητή πλειοψηφικής λογικής. Τέλος θα πρέπει να ειπωθεί ότι η πιθανότητα λάθους p_e μειώνεται με την αύξηση του n. Για παράδειγμα, για n=9, για τη προηγούμενη περίπτωση, έχουμε εφαρμόζοντας τη σχέση (5.2):

$$p_e = \sum_{k=5}^{9} \binom{9}{k} \cdot 0.001^k \cdot (0.999)^{9-k} = 9.97 \times 10^{-16} \cong 10^{-15} \tag{5.4}$$

Από το προηγούμενο παράδειγμα, μπορούμε να συμπεράνουμε ότι αν θέλουμε να μειώσουμε την πιθανότητα λάθους σε έναν απλό κώδικα επανάληψης στο μηδέν, θα πρέπει να αυξήσουμε το μήκος του κώδικα επανάληψης (n) στο άπειρο και αντίστοιχα να μειώσουμε έτσι το ρυθμό εκπομπής δεδομένων στο κανάλι στο μηδέν. Όμως ο C. Shannon απέδειξε, [6,7], ότι μπορούμε να διατηρήσουμε το ρυθμό εκπομπής σε τιμή μικρότερη από τη χωρητικότητα του καναλιού επικοινωνίας, με πιθανότητα λάθους η οποία να τείνει σε μηδενική τιμή $(p_e \to 0)$ αλλά χρησιμοποιώντας κατά την εκπομπή στο κανάλι επικοινωνίας, πιο σύνθετης δομής κώδικες, από ότι ένας απλός κώδικας επανάληψης.

5.3 Εργαστηριακό μέρος

Να γράψετε στο MATLAB πρόγραμμα υπολογισμού και αναπαράστασης της *πιθανότητας λάθους* στο αποκωδικοποιημένο bit για απλό κώδικα επανάληψης, στον οποίο σας δίνεται ότι η πιθανότητα λάθους σε ένα εκπεμπόμενο bit στο κανάλι επικοινωνίας είναι ίση με ε=0.3 και να γίνει γραφική παράσταση της πιθανότητας λάθους αποκωδικοποίησης p_e σε συνάρτηση με το *μήκος του κώδικα* (μήκος του μπλοκ) (block-length)(n). Να δώσετε τιμές στο n όλες τις περιττές τιμές από 1 έως και 61.

Πρόγραμμα

```
echo off
ep=0.3;
for i=1:2:61
   p(i)=0;
   For j=(i+1)/2:i
   p(i)=p(i)+prod(1:i)/(prod(1:j)*prod(1☹i-j)))*ep^j*(1-ep)^(i-j);
end
end
stem((1:2:61),p(1:2:61))
xlabel(‚n’)
ylabel(‚pe’)
title(‘Error probability as a function of n in simple repetition code’)
```

Εξήγηση Προγράμματος

Από την εκφώνηση της άσκησης μπορούμε να βρούμε την έκφραση για την πιθανότητα p_e εφαρμόζοντας τη σχέση (5.2) όπως παρουσιάζεται παρακάτω:

$$p_e = \sum_{k=(n+1)/2}^{n} \binom{n}{k} \cdot 0.3^k \cdot (1-0.3)^{n-k} = \sum_{k=(n+1)/2}^{n} \binom{n}{k} \cdot 0.3^k \cdot 0.7^{n-k} \qquad (5.5)$$

Αρχικά δίνουμε ως σταθερά τη συγκεκριμένη τιμή της πιθανότητας ε μέσω της μεταβλητής *ep*:

```
ep=0.3;
```
(5.6)

Το μήκος του κώδικα n, δίνεται μέσω της μεταβλητής *i*. Η μεταβλητή *i* λαμβάνει όλες τις περιττές τιμές από 1 έως 61, για αυτό το λόγο και γράφουμε στο βρόχο επανάληψης που θα χρησιμοποιήσουμε την επόμενη εντολή:

```
for i=1:2:61
```
(5.7)

δηλαδή χρησιμοποιούμε ήμα αύξησης ίσο με 2.

Για να υπολογίσουμε το άθροισμα που απαιτείται στη σχέση (5.5), ορίζουμε έναν μονοδιάστατο πίνακα *p(i)* ο οποίος θα έχει ως περιεχόμενο μετά το τέλος κάθε εκτέλεσης του βρόχου επανάληψης την τιμή του p_e για την αντίστοιχη τιμή του n. Απαιτείται όμως μετά το τέλος κάθε εκτέλεσης του βρόχου επανάληψης, το περιεχόμενο του *p(i)* να μηδενίζεται για αυτό και δίνεται η παρακάτω εντολή:

```
p(i)=0;
```
(5.8)

στην αρχή του βρόχου επανάληψης. Η μαθηματική έκφραση (συνδυασμοί των *n* ανά *k*) $\binom{n}{k} \overset{\text{ορ}}{=} \frac{n!}{k!\,(n-k)!}$ δίνεται στο MATLAB μέσω της έκφρασης:

prod(1:i)/(prod(1:j)*prod(1:(i-j))) (5.9)

(το *i* αντιστοιχεί στο *n* και το *j* αντιστοιχεί στο *k*) όπου ισχύει για την έκφραση του *n!* στο MATLAB:

n!=i!=prod(1:i) (5.10)

Παρατηρούμε ότι στο πρόγραμμα υπάρχουν δύο βρόχοι επανάληψης (ένας εξωτερικός και ένας εσωτερικός). Στον εξωτερικό βρόχο βρίσκεται η πιθανότητα p_e για συγκεκριμένο *n* ενώ στον εσωτερικό βρόχο κάθε φορά υπολογίζεται για το συγκεκριμένο *n* το άθροισμα που απαιτείται από τη σχέση (5.5). Η γραφική παράσταση της πιθανότητας λάθους p_e, για τις διάφορες τιμές του *n* από 1 έως και 61, πραγματοποιείται μέσω της εντολής:

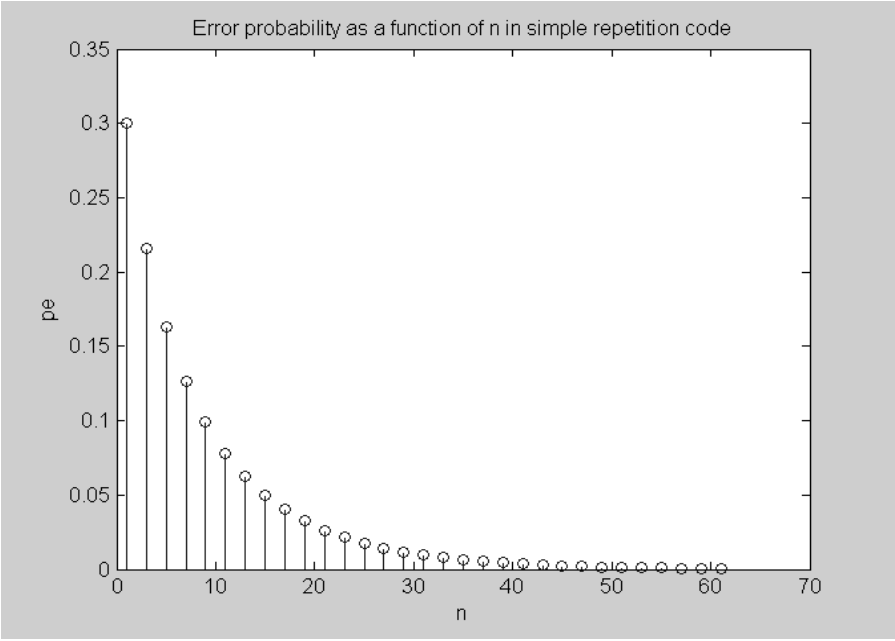
stem((1:2:61),p(1:2:61)) (5.10)

(γραφική παράσταση κατά βήματα)(διακριτές τιμές).

Τέλος, θέτουμε *ονομασίες* (*label*) στους άξονες x και y όπως και *τίτλο* (*title*) στη γραφική παράσταση μέσω των επόμενων εντολών:

```
xlabel('n')
ylabel('pe')
title('Error probability as a function of n in simple repetition code')
```
(5.11)

Έξοδος προγράμματος



Εικόνα 5.1 *Πιθανότητα λάθους p_e αποκωδικοποίησης* (Error probability) για έναν απλό κώδικα επανάληψης σε συνάρτηση με το μήκος του κώδικα *n* (*Έξοδος από εκτέλεση του προγράμματος στο MATLAB*).

5.4 Πρόσθετες εργασίες

1. Εξηγείστε γιατί η αύξηση του n οδηγεί σε μείωση της πιθανότητας p_e;
2. Δώστε πρόγραμμα στο MATLAB το οποίο θα υπολογίζει την τιμή της πιθανότητας λάθους αποκωδικοποίησης για απλό κώδικα επανάληψης με μήκος n=7.
3. Αν για έναν απλό κώδικα επανάληψης απαιτούμε να έχουμε πιθανότητα λάθους αποκωδικοποίησης ίση με 0.05, ποιο θα πρέπει να είναι κατά προσέγγιση το μήκος του κώδικα για την περίπτωση ενός καναλιού με ε=0.3; Για το κανάλι αυτό, ποια είναι η τιμή της χωρητικότητας του καναλιού;
4. Αν για έναν απλό κώδικα επανάληψης απαιτούμε να έχουμε πιθανότητα λάθους αποκωδικοποίησης ίση με 0.05, ποιο θα πρέπει να είναι κατά προσέγγιση το μήκος του κώδικα για την περίπτωση ενός καναλιού με ε=0.1; Απαντήστε γράφοντας ένα αντίστοιχο πρόγραμμα στο MATLAB. Για το κανάλι αυτό, ποια είναι η τιμή της χωρητικότητας του καναλιού;
5. Να αποδείξετε με μαθηματικό τρόπο τη σχέση (5.2).

ΑΣΚΗΣΗ 6η

Εύρεση των κωδικών λέξεων κώδικα Hamming (n,k) με δεδομένο τον πίνακα γεννήτορα του κώδικα

6.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι να παραχθούν μέσω ενός προγράμματος στο MATLAB οι κωδικές λέξεις για ένα κώδικα Hamming (*n*,*k*) με δεδομένο τον πίνακα γεννήτορα του κώδικα.

6.2 Θεωρητικό μέρος

6.2.1 Γραμμικοί κώδικες Μπλοκ

Οι γραμμικοί μπλοκ κώδικες είναι γενικά οι πιο ευρέως χρησιμοποιούμενοι κώδικες. Ένας κώδικας μπλοκ είναι *γραμμικός* αν ένας οποιαδήποτε γραμμικός συνδυασμός δύο κωδικών λέξεων του κώδικα αποτελεί, πάλι κωδική λέξη του συγκεκριμένου κώδικα, [1,5,9]. Γενικά, οι γραμμικοί κώδικες μπλοκ περιγράφονται από τον *πίνακα γεννήτορά* τους (generator matrix) *G*, ο οποίος είναι ένας ($k \times n$) δυαδικός πίνακας έτσι ώστε κάθε κωδική λέξη *c* του κώδικα να προκύπτει από τη σχέση:

$$c = uG \tag{6.1}$$

όπου *u* είναι η ακολουθία εισόδου μήκους *k* (μήνυμα)(η είσοδος στον κωδικοποιητή). Η πράξη μεταξ'υτων πινάκων είναι πράξη modulo-2 (δηλαδή πράξη XOR). Μία σημαντική παράμετρος σε ένα γραμμικό κώδικα μπλοκ, η οποία καθορίζει και τη δυνατότητα διόρθωσης λαθών του κώδικα, είναι η *ελάχιστη απόσταση* (απόσταση Hamming) του κώδικα η οποία ορίζεται ως η ελάχιστη απόσταση Hamming μεταξύ δύο οποιονδήποτε κωδικών λέξεων του κώδικα, [11]. Η ελάχιστη απόσταση του κώδικα συμβολίζεται με $d_{\min}$ και δίνεται από τη σχέση, [11]:

$$d_{\min} = \min_{i \neq j} d_H\left(c_i, c_j\right) \tag{6.2}$$

Για τους γραμμικούς κώδικες, η ελάχιστη απόσταση Hamming είναι ίση με το *ελάχιστο βάρος* w_{min} (*minimum weigth,* w_H) του κώδικα το οποίο ορίζεται από τη σχέση:

$$w_{\min} = \min_{C_i \neq 0} w_H\left(c_i\right) \tag{6.3}$$

δηλαδή είναι το ελάχιστο *βάρος* (*weigth,* w_H) του κώδικα, αναφερόμενοι σε όλες τις μη μηδενικές κωδικές λέξεις του κώδικα (το βάρος w_H μιας οποιασδήποτε κωδικής λέξης ορίζεται ως ο αριθμός των "1" που παρουσιάζονται σε αυτή την κωδική λέξη).

Ένας γραμμικός κώδικας μπλοκ είναι σε *συστηματική μορφή* (systematic form) αν ο πίνακας γεννήτορας του *G* έχει την παρακάτω μορφή:

$$\mathrm{G}=\begin{bmatrix} 1\ 0 & \ldots. & 0 & p_{1,1} & p_{1,2} & . & . & p_{1,n-k} \\ 0\ 1 & \ldots. & 0 & p_{2,1} & p_{2,2} & . & . & p_{2,n-k} \\ & & & & & & & \\ 0\ 0 & \ldots.. & 1 & p_{k,1} & p_{k,2} & . & . & p_{k,n-k} \end{bmatrix} \tag{6.4}$$

ή

$$G=[I_k \mid P] \tag{6.5}$$

όπου στη σχέση (6.5) το αριστερό τμήμα του πίνακα δηλαδή ο I_k, είναι ο ($k \times k$) *μοναδιαίος πίνακας* (identity matrix) και *P* είναι ένας $k \times (n-k)$ πίνακας. Σε ένα συστηματικό κώδικα, τα

πρώτα k bits της κωδικής λέξης είναι τα bits της πληροφορίας (μηνύματος) και τα υπόλοιπα ($n-k$) bits είναι τα *bits ελέγχου της ισοτιμίας* (parity-check bits).

Ο πίνακας ελέγχου της ισοτιμίας (parity check matrix) H σε έναν κώδικα είναι ένας $(n-k)\times n$ δυαδικός πίνακας, τέτοιος ώστε για όλες τις κωδικές λέξεις c του κώδικα να ισχύει η σχέση:

$$cH^t = 0 \tag{6.6}$$

όπου H^t είναι ο *πίνακας ανάστροφος* (transpose matrix) του πίνακα H (αυτός ο πίνακας που προκύπτει αν τις γραμμές του πίνακα H τις κάνουμε στήλες και αντίστροφα).

Προφανώς ισχύει:

$$GH^t = 0 \tag{6.7}$$

Αν ο πίνακας γεννήτορας G είναι σε συστηματική μορφή τότε ισχύει:

$$H=[-P^t|I_{n-k}] \tag{6.8}$$

(το πρόσημο (–) στη προηγούμενη σχέση δεν έχει κάποιο νόημα γιατί στο δυαδικό σύστημα αρίθμησης ισχύει: "1"="-1").

6.2.2 Κώδικες Hamming

Οι κώδικες Hamming είναι γραμμικοί κώδικες μπλοκ διαστάσεων $(2^m-1,\ 2^m-m-1)$ που παρουσιάζουν ελάχιστη απόσταση ίση με 3 και έχουν έναν πολύ απλό πίνακα ελέγχου ισοτιμίας. Ο πίνακας ελέγχου ισοτιμίας ο οποίος είναι ένας πίνακας $m\times(2^m-1)$ πίνακας, έχει σαν στήλες όλες τις δυαδικές ακολουθίες με μήκος m, εκτός από την μηδενική ακολουθία. Για παράδειγμα, για m=3 έχουμε έναν (7,4) κώδικα του οποίου ο πίνακας ελέγχου της ισοτιμίας, σε συστηματική μορφή, είναι ο παρακάτω:

$$H=\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{6.9}$$

Εφαρμόζοντας τη σχέση (6.5) (και με τη βοήθεια της (6.8)) βρίσκουμε τον πίνακα γεννήτορα G του συγκεκριμένου κώδικα:

$$G=\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{6.10}$$

σε συστηματική μορφή (δηλαδή $G=[I_4 \mid P]$.

6.3 Εργαστηριακό μέρος

Να ορίσετε στο MATLAB έναν κώδικα Hamming με m=3 δηλαδή έναν (n,k)=(7,4) κώδικα Hamming. Στη συνέχεια να δώσετε σαν είσοδο τον πίνακα γεννήτορα G του πίνακα, τον επόμενο πίνακα:

$$G=\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

Να βρείτε μέσω του MATLAB τον πίνακα ελέγχου της ισοτιμίας του συγκεκριμένου κώδικα και να βρείτε τη κωδική λέξη που αντιστοιχεί στο παρακάτω μήνυμα μήκους k=4 bits:[1 0 1 1].

Πρόγραμμα

```
n=7; k=4;
genmat=[1 1 0 1 0 0 0; 0 1 1 0 1 0 0; 1 1 1 0 0 1 0; 1 0 1 0 0 0 1]
parmat=hammgen(3)
msg= [1 0 1 1];
code=encode(msg, n, k, 'hamming')
msg'
code'
```

Εξήγηση Προγράμματος

Στο πρόγραμμα που θα αναπτυχθεί, θα χρησιμοποιήσουμε τις έτοιμες ρουτίνες που διαθέτει το MATLAB σχετικές με τους Hamming κώδικες. Αρχικά ορίζουμε τις διαστάσεις του Hamming κώδικα με την επόμενη εντολή:

$$\text{n=7; k=4;} \tag{6.12}$$

Στη συνέχεια, θα χρησιμοποιήσουμε τη συνάρτηση *hammgen* η οποία παράγει ως έξοδο *τον πίνακα ελέγχου ισοτιμίας* για το συγκεκριμένο κώδικα. Από την εκφώνηση της άσκησης παρατηρούμε ότι ο πίνακας γεννήτορας G είναι στη μορφή:

$$G=[P \mid I_{k=4}] \tag{6.13}$$

Εισάγουμε στο MATLAB τον πίνακα γεννήτορα *G* που μας δίνεται από την εκφώνηση της άσκησης μέσω της εντολής:

$$\text{genmat=[1 1 0 1 0 0 0; 0 1 1 0 1 0 0; 1 1 1 0 0 1 0; 1 0 1 0 0 0 1]} \tag{6.14}$$

Στη συνέχεια μέσω της συνάρτησης *hammgen* παράγουμε τον πίνακα ελέγχου της ισοτιμίας *H (parmat)* ο οποίος έχει τη γενική μορφή:

$$H=[\, I_{n-k} \mid -P^t \,]= =[\, I_3 \mid -P^t] \tag{6.15}$$

όταν ο πίνακας G έχει τη μορφή της σχέσης (6.13). Δηλαδή γράφουμε την εντολή:

$$\text{parmat=hammgen(3)} \tag{6.16}$$

Με τη προηγούμενη εντολή παράγεται ως έξοδος από το πρόγραμμα ο πίνακας ελέγχου της ισοτιμίας *H*:

$$parmat=\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{6.17}$$

Ο ορισμός του μηνύματος, msg (message), μήκους k=4 το οποίο θα κωδικοποιηθεί, θα πραγματοποιηθεί με την εντολή:

```
msg= [1 0 1 1];                                    (6.18)
```

Η κωδικοποίηση του μηνύματος, πραγματοποιείται μέσω της εντολής:

```
code=encode(msg, n, k, 'hamming')                  (6.19)
```

η οποία καλεί την ρουτίνα του MATLAB που αναφέρεται στους κώδικες Hamming.

Η έξοδος της προηγούμενης εντολής είναι αντίστοιχα:

```
Code'=[ 1 0 0 1 0 1 1]                             (6.20)
```

Παρατηρούμε ότι τα τελευταία 4 bits στην κωδική λέξη που παράγεται, είναι τα bits του συγκεκριμένου μηνύματος πληροφορίας που δώσαμε και τα 3 πρώτα bits της κωδικής λέξης είναι τα *bits ελέγχου της ισοτιμίας* (parity-check bits).

Έξοδος προγράμματος

Παρακάτω παρατηρούμε την έξοδο του προγράμματος στο MATLAB που αναλύθηκε στην προηγούμενη παράγραφο:

```
n=7; k=4;
        genmat=[1 1 0 1 0 0 0; 0 1 1 0 1 0 0; 1 1 1 0 0 1 0; 1 0 1 0 0 0 1]

genmat =

    1    1    0    1    0    0    0
    0    1    1    0    1    0    0
    1    1    1    0    0    1    0
    1    0    1    0    0    0    1

        parmat=hammgen(3)

parmat =

    1    0    0    1    0    1    1
    0    1    0    1    1    1    0
    0    0    1    0    1    1    1

        msg= [1 0 1 1];
        code=encode(msg, n, k, 'hamming')

code =

    1
    0
    0
    1
    0
    1
    1
```

```
        msg'
ans =

    1
    0
    1
    1

        code'

ans =

    1    0    0    1    0    1    1
```

6.4 Πρόσθετες εργασίες

1. Να τροποποιήσετε το προηγούμενο πρόγραμμα ώστε να κωδικοποιήσετε το μήνυμα [0 0 1 0].
2. Σας δίνεται κώδικας Hamming (15,11). Πόσα bits περιλαμβάνει η κωδική λέξη του συγκεκριμένου κώδικα; Πόσα bits έχει κάθε μήνυμα πληροφορίας για το συγκεκριμένο κώδικα; Να γράψετε πρόγραμμα στο MATLAB το οποίο να θεωρεί ως πίνακα γεννήτορα *G* του κώδικα τον παρακάτω και να βρείτε την κωδική λέξη για ένα οποιοδήποτε μήνυμα μηνύματος ορίσετε εσείς με κατάλληλο μήκος:

$$G=[P \mid I_{11}] \tag{6.21}$$

όπου ο P είναι ο παρακάτω πίνακας:

$$P=\begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{6.22}$$

ΑΣΚΗΣΗ 7η

Εύρεση πίνακα συνδρόμου, ανίχνευση και διόρθωση λάθους σε κώδικα Hamming (n,k)

7.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι η παραγωγή του πίνακα συνδρόμου σε έναν κώδικα Hamming (*n*,*k*) με δεδομένο τον πίνακα ελέγχου ισοτιμίας του κώδικα. Επίσης, θα πραγματοποιηθούν *ανίχνευση* (detection) και *διόρθωση* (correction) *απλού λάθους* (single error) στη λαμβανόμενη κωδική λέξη.

7.2 Θεωρητικό μέρος

Έστω ένας κώδικας Hamming (*n,k*). Αν για αυτόν τον κώδικα γνωρίζουμε τον πίνακα ελέγχου της ισοτιμίας του *H* είναι δυνατόν να βρούμε τον πίνακα γεννήτορα του κώδικα *G*. Αν ο πίνακας ελέγχου της ισοτιμίας του κώδικα *H* είναι στην παρακάτω μορφή:

$$H=[-P^t| I_k] \tag{7.1}$$

τότε ο πίνακας γεννήτορας *G* του κώδικα έχει αντίστοιχα την παρακάτω μορφή:

$$G=[I_k \mid P] \tag{7.2}$$

Η κωδική λέξη που παράγεται εκπέμπεται στο κανάλι επικοινωνίας. Εξαιτίας του θορύβου του καναλιού, η λαμβανόμενη κωδική λέξη μπορεί να παρουσιάζει ένα απλό ή περισσότερα λάθη δηλαδή άλλο bit ή bits να έχει εκπεμφθεί στο κανάλι και διαφορετικό ή διαφορετικά bits να έχουν αποκωδικοποιηθεί στη λήψη.

Είναι λογικό να μοντελοποιήσουμε τα λάθη που παρουσιάζονται στη λήψη με ένα *διάνυσμα* (vector ή table) το οποίο περιέχει 0 στη θέση εκείνη που δεν παρουσιάστηκε λάθος και 1 σε οποιαδήποτε θέση έγινε λάθος κατά την αποκωδικοποίηση. Το διάνυσμα αυτό ονομάζεται *διάνυσμα συνδρόμου* (syndrome table) *S* ή απλά *σύνδρομο*. Η χρησιμότητά του είναι ότι με δεδομένο το διάνυσμα συνδρόμου μπορούμε να ανιχνεύσουμε αν υπάρχει απλό λάθος στη λαμβανομένη κωδική λέξη και να τα διορθώσουμε. Συγκεκριμένα, αν πολλαπλασιάσουμε την λαμβανόμενη κωδική λέξη *R* με τον πίνακα ελέγχου ισοτιμίας *H* θα βρούμε τον πίνακα (διάνυσμα) του συνδρόμου *S*. Το σύνδρομο που υπολογίζεται θα πρέπει να αποτελεί μία στήλη του πίνακα ελέγχου της ισοτιμίας *H*. Η συγκεκριμένη στήλη δείχνει και τη θέση στη λαμβανόμενη λέξη που έχει συμβεί το λάθος κατά τη μετάδοση στο κανάλι επικοινωνίας π.χ. αν προέκυψε η δεύτερη στήλη ($2^η$) τότε έχει γίνει λάθος στο $2^ο$ λαμβανόμενο bit στη λαμβανόμενη λέξη *R*.

$$R \cdot H^t = S \tag{7.3}$$

Έτσι η σωστή κωδική λέξη (*CorrectedCodeword)* θα προκύψει αν στη λαμβανόμενη κωδική λέξη *R* προσθέσουμε με πράξη *modulo-2* (αποκλειστικό ή) το διάνυσμα του συνδρόμου *S*. Δηλαδή ισχύει:

$$CorrectedCodeword = R \oplus S \tag{7.4}$$

Στην αποκωδικοποίηση, υπάρχει ένας πίνακας ο οποίος ονομάζεται *πίνακας αποκωδικοποίησης* (decoding table) που δείχνει στον αποκωδικοποιητή πως θα

διορθωθούν τα λάθη που παρατηρούνται στη λαμβανόμενη λέξη. Στους κώδικες Hamming υπάρχει η δυνατότητα διόρθωσης ενός απλού λάθους σε κάθε λαμβανόμενη κωδική λέξη.

7.3 Εργαστηριακό μέρος

Πρόγραμμα

```
n=7; k=4;      % the parameters of the Hamming code
parmat=[1 1 1 0 1 0 0; 1 1 0 1 0 1 0; 1 0 1 1 0 0 1]   % parity-check matrix
genmat=gen2par(parmat)     % find the Generator matrix from the parity-check matrix
msg=[1 0 0 0];%  message
code=encode(msg, n, k, 'hamming')  %  coding with Hamming code (n,k)
trt=syndtable(parmat)  %  Produce decoding table
recd=[1 1 0 1 1 1 0]     %   the received codeword
syndrome=rem(recd*parmat', 2)  %  syndrome in binary
syndrome_de=bi2de(syndrome, 'left-msb');        % Convert to decimal
corrvect=trt(1+syndrome_de,:)
correctdcode=rem(corrvect+recd,2)    % find the corrected codeword
msg';
code';
```

Εξήγηση Προγράμματος

Στην επόμενη πρώτη εντολή του προγράμματος, ορίζουμε τις παραμέτρους του συγκεκριμένου κώδικα Hamming:

```
n=7; k=4; % the parameters of the Hamming code
```
(7.5)

Στη συνέχεια εισάγουμε στο πρόγραμμα τον πίνακα ελέγχου της ισοτιμίας *H,* (*parmat*) ο οποίος θεωρούμε ότι είναι ο παρακάτω πίνακας:

$$H=\begin{bmatrix}1 & 1 & 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 0 & 1 & 0 & 1 & 0\\ 1 & 0 & 1 & 1 & 0 & 0 & 1\end{bmatrix} \quad (7.6)$$

μέσω της εντολής:

```
parmat=[1 1 1 0 1 0 0; 1 1 0 1 0 1 0; 1 0 1 1 0 0 1]   % parity-check matrix
```
(7.7)

Με δεδομένο τον πίνακα ελέγχου ισοτιμίας *H* βρίσκουμε τον πίνακα γεννήτορα *G* με τη βοήθεια της εντολής:

```
genmat=gen2par(parmat) % find the Generator matrix from the parity-check matrix
```
(7.8)

ο οποίος στο συγκεκριμένο κώδικα είναι ο παρακάτω πίνακας:

$$genmat=\begin{bmatrix}1 & 0 & 0 & 0 & 1 & 1 & 1\\ 0 & 1 & 0 & 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 0 & 1 & 1\end{bmatrix} \quad (7.9)$$

Στη συνέχεια θεωρούμε ότι το μήνυμα που θα κωδικοποιήσουμε είναι το [1 0 0 0] με μήκος 4 bits:

```
msg=[1 0 0 0];        % message                    (7.10)
```

και καλούμε τη ρουτίνα (συνάρτηση) κωδικοποίησης *encode* του MATLAB που αναφέρεται στους Hamming κώδικες:

```
code=encode(msg, n, k, 'hamming') %  coding with Hamming code (n,k)    (7.11)
```

Η κωδική λέξη η οποία αντιστοιχεί στο μήνυμα αυτό είναι η:

```
code=1    1    0    1    0    0    0                    (7.12)
```

Η εύρεση του πίνακα αποκωδικοποίησης (*trt*) παράγεται μέσω της ρουτίνας *syntable* του MATLAB:

```
trt=syndtable(parmat)  %  Produce decoding table        (7.13)
```

με δεδομένο τον πίνακα ελέγχου ισοτιμίας *H* (*parmat*) και η οποία για το συγκεκριμένο κώδικα παράγει τον παρακάτω πίνακα αποκωδικοποίησης:

$$\text{trt}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (7.14)$$

Στο πρόγραμμα θεωρούμε ότι η *λαμβανόμενη κωδική λέξη* (*recd*) είναι η *recd*=[1 1 0 1 1 1 0]. Ο υπολογισμός του συνδρόμου (*syndrome)* (σε δυαδική μορφή) για τη συγκεκριμένη κωδική λέξη γίνεται εφαρμόζοντας τη σχέση (7.3) και με τη βοήθεια της επόμενης εντολής:

```
syndrome=rem(recd*parmat', 2)  %    syndrome in binary    (7.15)
```

στην οποία η λαμβανόμενη κωδική λέξη *recd* και ο πίνακας H^t αθροίζονται κατά *modulo 2* (υπόλοιπο της διαίρεσης με το 2). Έτσι λαμβάνουμε το σύνδρομο (*syndrome)* σε δυαδική μορφή:

```
syndrome =

     1     0     0                                    (7.16)
```

Είναι δυνατό να μετατρέψουμε το σύνδρομο σε *δεκαδική μορφή* (decimal). Αυτό πετυχαίνεται με τη συνάρτηση του MATAB *bi2de*, όπως παρουσιάζεται στη παρακάτω εντολή του προγράμματος:

syndrome_de=bi2de(syndrome, 'left-msb'); % Convert to decimal (7.17)

Έξοδος προγράμματος

Παρακάτω παρατηρούμε την έξοδο του προγράμματος στο MATLAB που αναλύθηκε στην προηγούμενη παράγραφο:

```
n=7; k=4;        %  the parameters of the Hamming code
parmat=[1 1 1 0 1 0 0; 1 1 0 1 0 1 0; 1 0 1 1 0 0 1]  % parity-check matrix

parmat =

    1    1    1    0    1    0    0
    1    1    0    1    0    1    0
    1    0    1    1    0    0    1

genmat=gen2par(parmat)     % find the Generator matrix from the parity-check matrix

genmat =

    1    0    0    0    1    1    1
    0    1    0    0    1    1    0
    0    0    1    0    1    0    1
    0    0    0    1    0    1    1

msg=[1 0 0 0];   % message
code=encode(msg, n, k, 'hamming')       %  coding with Hamming code (n,k)

code =

    1
    1
    0
    1
    0
    0
    0

trt=syndtable(parmat)         %  Produce decoding table
trt =
    0    0    0    0    0    0    0
    0    0    0    0    0    0    1
    0    0    0    0    0    1    0
    0    0    0    1    0    0    0
    0    0    0    0    1    0    0
    0    0    1    0    0    0    0
    0    1    0    0    0    0    0
    1    0    0    0    0    0    0

recd=[1 1 0 1 1 1 0]       %       the received codeword
```

```
recd =

    1     1     0     1     1     1     0

syndrome=rem(recd*parmat', 2) ;      %   syndrome in binary

syndrome =

    1     0     0

syndrome_de=bi2de(syndrome, 'left-msb'); % Convert to decimal
corrvect=trt(1+syndrome_de,:)

corrvect =

    0     0     0     0     1     0     0

correctdcode=rem(corrvect+recd,2) ;      %   find the corrected codeword

correctdcode =

    1     1     0     1     0     1     0

msg';
code';
```

7.4 Πρόσθετες εργασίες

1. Σας δίνεται ο πίνακας ελέγχου ισοτιμίας της συγκεκριμένης άσκησης. Να βρείτε αν οι επόμενες λαμβανόμενες λέξεις παρουσιάζουν απλό λάθος και να βρείτε τη σωστή κωδική λέξη:

   - recd=[1 1 1 1 1 1 1]
   - recd=[0 0 0 0 0 0 0]
   - recd=[0 1 0 1 0 1 0]

2. Πιστεύετε ότι η διαδικασία αποκωδικοποίησης με τη βοήθεια του συνδρόμου, όπως περιγράφτηκε στη συγκεκριμένη άσκηση, οδηγεί πάντα σε επιτυχή αποκωδικοποίηση;

ΑΣΚΗΣΗ 8η

Κωδικοποίηση με χρησιμοποίηση κυκλικού κώδικα (n,k)

8.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι η παραγωγή των κωδικών λέξεων (πολυωνύμων) για την περίπτωση ενός *κυκλικού κώδικα* (cyclic code) (*n*,*k*) με δεδομένο το πολυώνυμο γεννήτορα του κώδικα. Επίσης σκοπός της άσκησης είναι να περιγραφτεί η διαδικασία παραγωγής του πίνακα γεννήτορα και του πίνακα ελέγχου ισοτιμίας για έναν κυκλικό κώδικα (*n*,*k*) όπως και η διαδικασία της αποκωδικοποίησης των λαμβανομένων κωδικών λέξεων.

8.2 Θεωρητικό μέρος

Ένας γραμμικός κώδικας ονομάζεται κυκλικός αν επιπρόσθετα από τις ιδιότητες των γραμμικών κωδίκων έχει την ιδιότητα μία οποιαδήποτε *κυκλική ολίσθηση* (cyclic shift) μιας κωδικής του λέξης να αποτελεί και αυτή κωδική λέξη του συγκεκριμένου κώδικα,[11]. Οι κυκλικοί κώδικες μπορούν να περιγραφτούν ισοδύναμα είτε με τον πίνακα γεννήτορα *G* και τον πίνακα ελέγχου ισοτιμίας τους *H* είτε με τη βοήθεια του *πολυωνύμου γεννήτορα* (generator polynomial). Συνεπώς, αν $C = \left[c_{n-1}\ c_{n-2}\ ...c_1 c_0\right]$ είναι μία κωδική λέξη ενός κυκλικού κώδικα τότε και η κυκλική του ολίσθηση $\left[c_{n-2}\ c_{n-3}\ ...c_0 c_{n-1}\right]$ θα είναι επίσης κωδική λέξη του κώδικα.

Για να περιγράψουμε τους κυκλικούς κώδικες, αντιστοιχούμε σε κάθε κωδική λέξη $C = \left[c_{n-1}\ c_{n-2}\ ...c_1 c_0\right]$ ένα πολυώνυμο $C(p)$ βαθμού $\le n-1$, το οποίο ορίζεται ως εξής:

$$C(p) = c_{n-1}p^{n-1} + c_{n-2}p^{n-2} + ... + c_1 p + c_0 \tag{8.1}$$

Είναι δυνατό να γράψουμε τη προηγούμενη σχέση (8.1) ως εξής:

$$pC(p) = c_{n-1}p^{n} + c_{n-2}p^{n-1} + ... + c_1 p^2 + c_0 p \tag{8.2}$$

Αν διαιρέσουμε το αριστερό μέλος της σχέσης (8.2) με $p^n + 1$, τότε λαμβάνουμε:

$$\frac{pC(p)}{p^n + 1} = c_{n-1} + \frac{C_1(p)}{p^n + 1} \tag{8.3}$$

όπου

$$C_1(p) = c_{n-2}p^{n-1} + c_{n-3}p^{n-2} + ... + c_1 p^2 + c_0 p + c_{n-1} \tag{8.4}$$

Δεδομένου ότι το $C_1(p)$ είναι το υπόλοιπο της διαίρεσης του $pC(p)$ με το $p^n + 1$ μπορούμε να γράψουμε:

$$C_1(p) = pC(p) \bmod \left(p^n + 1\right) \tag{8.5}$$

Μπορούμε να παράγουμε έναν κυκλικό κώδικα (*n*,*k*), χρησιμοποιώντας το πολυώνυμα γεννήτορα $g(p)$ βαθμού (*n*-*k*). Το πολυώνυμο γεννήτορας ενός (*n*,*k*) κυκλικού κώδικα, έχει την παρακάτω γενική μορφή:

$$g(p) = p^{n-k} + g_{n-k-1}p^{n-k-1} + ... + g_1 p + 1 \tag{8.6}$$

Παράλληλα μπορούμε να ορίσουμε το *πολυώνυμο του μηνύματος* (message polynomial) ως εξής:

$$X(p) = x_{k-1}p^{k-1} + x_{k-2}p^{k-2} \ldots + x_1 p + x_0 \tag{8.7}$$

όπου το $[x_{k-1}\ x_{k-2} \ldots x_1 x_0]$ αναπαρασταίνει τα k bits του μηνύματος (πληροφορία). Το γινόμενο των πολυωνύμων $X(p)\cdot g(p)$ είναι ένα πολυώνυμο βαθμού μικρότερου ή ίσου με το (n-1) και το οποίο αναπαρασταίνει μία κωδική λέξη του κυκλικού κώδικα.

Παράδειγμα

Ας θεωρήσουμε ένα κώδικα με μήκος n=7 (σύνολο των bits μετά την κωδικοποίηση). Το πολυώνυμο p^7+1 μπορεί να γραφτεί ως γινόμενο παραγόντων:

$$p^7 + 1 = (p+1)\cdot(p^3 + p^2 + 1)\cdot(p^3 + p + 1) \tag{8.8}$$

Για να παράγουμε ένα κυκλικό κώδικα (n,k), θα πρέπει να θεωρήσουμε ένα από τα επόμενα πολυώνυμα ως πολυώνυμο γεννήτορα:

$$g_1(p) = (p^3 + p^2 + 1) \tag{8.9}$$

$$g_2(p) = (p^3 + p + 1) \tag{8.10}$$

Οι κώδικες οι οποίοι μπορούν να παραχθούν από τα δύο προηγούμενα πολυώνυμα, είναι ισοδύναμοι. Για παράδειγμα, τα μηνύματα [0001] και [1110] κωδικοποιούνται αντίστοιχα μέσω του πολυωνύμου $g_1(p) = (p^3 + p^2 + 1)$ ως [0001101] και [1000110].

Η πρόσθεση '+' μεταξύ των διαφόρων παραγόντων των πολυωνύμων είναι πράξη modulo 2 (λογική πράξη αποκλειστικού Η, EXOR, Exclusive OR). Συνεπώς, όταν μετά το συνήθη πολλαπλασιασμό εμφανίζονται δύο ίδιοι παράγοντες τότε αυτοί απαλείφονται μεταξύ τους ανά δύο π.χ. $p^3+p^3=0$.

Έστω ένας κυκλικός κώδικας (n,k). Τότε ο πίνακας γεννήτορας G για τον προηγούμενο κώδικα, δίνεται σε συστηματική μορφή από την επόμενη έκφραση:

$$G=[I_k|\ P] \tag{8.11}$$

Αντίστοιχα, ο πίνακας ελέγχου ισοτιμίας H του κυκλικού κώδικα (n,k), δίνεται σε συστηματική μορφή από την παρακάτω έκφραση:

$$H=[I_{n-k}\ |P^t] \tag{8.12}$$

8.3 Εργαστηριακό μέρος

Πρόγραμμα

```
n=15; k=5;
genpoly = cyclpoly(15,5)
[parmat,genmat] = cyclgen(15, cyclpoly(15,5))
msg=[0 0 0 1 1];
code=encode(msg, n, k, 'cyclic', genpoly)
newmsg=decode(code,n,k, 'cyclic', genpoly)
```

```
code1=[1 1 1 1 0 0 1]
newmsg=decode(code1,n,k, 'cyclic', genpoly)
```

Εξήγηση Προγράμματος

Στο πρόγραμμα, το οποίο φαίνεται προηγούμενα, ορίζουμε κατ' αρχήν τις παραμέτρους του κυκλικού κώδικα δηλαδή δίνουμε τις τιμές στις μεταβλητές *n* και *k*. Συγκεκριμένα αυτό γίνεται μέσω της εντολής:

$$\text{n=15; k=5;} \tag{8.13}$$

Υπενθυμίζουμε ότι *k* είναι το μήκος του μηνύματος το οποίο θα κωδικοποιηθεί και *n* είναι ο αριθμός των συνολικών bits μετά την κωδικοποίηση με τον κυκλικό κώδικα.

Η εύρεση του πολυωνύμου γεννήτορα του κυκλικού κώδικα, το οποίο στο πρόγραμμα ονομάζεται *genpoly*, πραγματοποιείται μέσω της επόμενης εντολής:

$$\text{genpoly = cyclpoly(15,5)} \tag{8.14}$$

η οποία έχει την παρακάτω γενική μορφή:

$$\text{genpoly = cyclpoly(n,k)} \tag{8.15}$$

Για το συγκεκριμένο κυκλικό κώδικα (15,5) το πολυώνυμο γεννήτορας, βρίσκεται να είναι:

genpoly =

$$1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \tag{8.16}$$

Η έκφραση που λαμβάνεται στο MATLAB δίνει τους συντελεστές του πολυωνύμου γεννήτορα του συγκεκριμένου κώδικα κατά αύξουσα σειρά. Έτσι η προηγούμενη έξοδος αναπαρασταίνει το πολυώνυμο γεννήτορα:

$$g(p) = 1 + x^5 + x^{10} \tag{8.17}$$

Είναι δυνατό να ευρεθεί ο πίνακας γεννήτορας *G* και ο πίνακας ελέγχου ισοτιμίας *H* του κυκλικού κώδικα. Για το σκοπό αυτό καλούμε στο MATLAB τη ρουτίνα (συνάρτηση) *cyclgen*. Έτσι, για το συγκεκριμένο κυκλικό κώδικα η εντολή:

$$\text{[parmat,genmat] = cyclgen(15, cyclpoly(15,5))} \tag{8.18}$$

η οποία γενικά συντάσσεται ως εξής:

$$\text{cyclgenn(n, cyclpoly(n,k))} \tag{8.19}$$

δίνει αντίστοιχα για τον πίνακα ελέγχου ισοτιμίας *parmat* και πίνακα γεννήτορα *genmat* τις παρακάτω εξόδους:

parmat =

Columns 1 through 13

1	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0

Columns 14 through 15

0	0
0	0
0	0
1	0
0	1
0	0
0	0
0	0
1	0
0	1

και

genmat =

Columns 1 through 13

1	0	0	0	0	1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0

Columns 14 through 15

0	0
0	0
0	0
1	0
0	1

Παρατηρούμε ότι ο πίνακας γεννήτορας έχει διαστάσεις (5×15) δηλαδή (k×n) ενώ ο πίνακας ελέγχου ισοτιμίας (10×15) δηλαδή ((n-k)×n).

Στη συνέχεια στο πρόγραμμα εισάγουμε ένα *μήνυμα* (msg) μήκους k=5 και συγκεκριμένα το [0 0 0 1 1] μέσω της εντολής:

msg=[0 0 0 1 1]; (8.20)

Το προηγούμενο μήνυμα κωδικοποιείται με τη βοήθεια της εντολής:

code=encode(msg, n, k, 'cyclic', genpoly) (8.21)

Η κωδική λέξη που προκύπτει είναι η επόμενη:

```
code =
    0
    0
    0
    1
    1
    0
    0
    0
    1
    1
    0
    0
    0
    1
    1
```

η οποία όπως παρατηρούμε έχει το σωστό και αναμενόμενο μήκος 15 bits (n=15).

Είναι δυνατό να προχωρήσουμε στην αποκωδικοποίηση (decoding) ενός λαμβανομένου μηνύματος για την περίπτωση ενός κυκλικού κώδικα με τη χρησιμοποίηση της επόμενης εντολής:

newmsg=decode(code,n,k, 'cyclic', genpoly) (8.22)

Εφαρμόζοντας έτσι την εντολή αυτή για τη *κωδική λέξη* (code) που προέκυψε προηγουμένως, βρίσκουμε άμεσα ότι αυτή αντιστοιχεί σωστά στο αρχικό μήνυμα (newmsg):

```
newmsg =

    0
    0
    0
    1
    1
```

Αν στη συνέχεια θεωρήσουμε ότι λαμβάνουμε την επόμενη *κωδική λέξη* (code1), μήκους 15 bits:

code1=[1 1 1 1 0 0 1 0 1 1 1 1 1 1 0]

τότε με την εντολή:

newmsg=decode(code1,n,k, 'cyclic', genpoly) (8.23)

την αποκωδικοποιούμε λαμβάνοντας ως νέο μήνυμα (newmsg) το επόμενο:

newmsg =

1
1
1
1
0

το οποίο όπως σωστά αναμένεται έχει μήκος 5 bits.

Έξοδος προγράμματος

Στη συνέχεια παρατηρούμε την έξοδο του προγράμματος στο MATLAB που αναλύθηκε στην προηγούμενη παράγραφο:

```
n=15; k=5;
genpoly = cyclpoly(15,5)

genpoly =

     1     0     0     0     0     1     0     0     0     0     1

[parmat,genmat] = cyclgen(15, cyclpoly(15,5))

parmat =

  Columns 1 through 13

     1     0     0     0     0     0     0     0     0     0     1     0     0
     0     1     0     0     0     0     0     0     0     0     0     1     0
     0     0     1     0     0     0     0     0     0     0     0     0     1
     0     0     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     1     0     0
     0     0     0     0     0     0     1     0     0     0     0     1     0
     0     0     0     0     0     0     0     1     0     0     0     0     1
     0     0     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     1     0     0     0

  Columns 14 through 15

     0     0
     0     0
     0     0
     1     0
     0     1
     0     0
     0     0
     0     0
     1     0
     0     1
genmat =

  Columns 1 through 13

     1     0     0     0     0     1     0     0     0     0     1     0     0
     0     1     0     0     0     0     1     0     0     0     0     1     0
     0     0     1     0     0     0     0     1     0     0     0     0     1
     0     0     0     1     0     0     0     0     1     0     0     0     0
     0     0     0     0     1     0     0     0     0     1     0     0     0

  Columns 14 through 15

     0     0
     0     0
     0     0
     1     0
```

```
    0     1

msg=[0 0 0 1 1];
code=encode(msg, n, k, 'cyclic', genpoly)

code =

    0
    0
    0
    1
    1
    0
    0
    0
    1
    1
    0
    0
    0
    1
    1

newmsg=decode(code,n,k, 'cyclic', genpoly)
Single-error patterns loaded in decoding table.  1008 rows remaining.
2-error patterns loaded.  918 rows remaining.
3-error patterns loaded.  648 rows remaining.
4-error patterns loaded.  243 rows remaining.
5-error patterns loaded.  0 rows remaining.

newmsg =

    0
    0
    0
    1
    1

code1=[1 1 1 1 0 0 1 0 1 1 1 1 1 1 0]

code1 =

  Columns 1 through 13

    1     1     1     1     0     0     1     0     1     1     1     1     1

  Columns 14 through 15

    1     0

newmsg=decode(code1,n,k, 'cyclic', genpoly)
Single-error patterns loaded in decoding table.  1008 rows remaining.
```

```
2-error patterns loaded.  918 rows remaining.
3-error patterns loaded.  648 rows remaining.
4-error patterns loaded.  243 rows remaining.
5-error patterns loaded.  0 rows remaining.

newmsg =

     1
     1
     1
     1
     0
```

8.4 Πρόσθετες εργασίες

1. Βρείτε τον πίνακα ελέγχου της ισοτιμίας του κυκλικού κώδικα (21,7). Να γραφτεί αντίστοιχο πρόγραμμα εύρεσης του πίνακα στο MATLAB. Οι διαστάσεις του πίνακα είναι αυτές που αναμένετε θεωρητικά;
2. Να κωδικοποιήσετε με τη βοήθεια ενός προγράμματος στο MATLAB το μήνυμα πληροφορίας [1 0 1 0] με έναν κυκλικό κώδικα (15,4).
3. Αν η λαμβανόμενη κωδική λέξη σε έναν κυκλικό κώδικα (7,4) είναι η [0 1 0 1 0 1 0], να βρείτε με τη βοήθεια ενός προγράμματος στο MATLAB σε ποιο αρχικό μήνυμα αντιστοιχεί αυτή η κωδική λέξη. Τι παρατηρείτε;
4. Βρείτε τον πίνακα γεννήτορα του κυκλικού κώδικα (7,3). Να γραφτεί αντίστοιχο πρόγραμμα εύρεσης του πίνακα γεννήτορα στο MATLAB. Οι διαστάσεις του πίνακα είναι αυτές που αναμένονται θεωρητικά; Ποιο είναι το πολυώνυμο γεννήτορας του συγκεκριμένου κώδικα και ποιος είναι ο βαθμός του πολυωνύμου γεννήτορα;

ΑΣΚΗΣΗ 9η

BCH Κώδικες-Κωδικοποίηση-Αποκωδικοποίηση

9.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι η παραγωγή των κωδικών λέξεων ενός κώδικα BCH δηλαδή η περιγραφή της κωδικοποίησης στο συγκεκριμένο κώδικα. Επίσης στην άσκηση περιγράφεται η αποκωδικοποίηση των κωδικών λέξεων ενός BCH κώδικα και θεωρείται η περίπτωση πρόσθεσης τυχαίου θορύβου (noise) στη λαμβανόμενη κωδική λέξη δηλαδή θεωρείται η περίπτωση λάθους στην αποκωδικοποίηση του αρχικού μηνύματος πληροφορίας.

9.2 Θεωρητικό μέρος

Οι BCH (Bose-Chaudhuri-Hocquenghem) κώδικες αποτελούν μία ευρεία κατηγορία *κυκλικών κωδίκων* και περιλαμβάνουν δυαδικά και μη δυαδικά αλφάβητα. Οι κώδικες BCH ανακαλύφθηκαν το 1959-1960 από τρεις ερευνητές από τα ονόματα των οποίων και δόθηκε το συγκεκριμένο όνομα στην κατηγορία αυτή των γραμμικών κυκλικών κωδίκων, [15,16]. Αποτελούν μία πολύ σημαντική κατηγορία γραμμικών κωδίκων διότι προσφέρουν σημαντική απόδοση (λόγος k/n), παρουσιάζουν ευρεία περιοχή τιμών για τα n και k και τέλος η πολυπλοκότητα των *κωδικοποιητών* (encoders) τους είναι σχετικά μικρή.

Οι BCH κώδικες (n,k) (n είναι το μήκος της κωδικής λέξης και k είναι το μήκος του μηνύματος το οποίο θα κωδικοποιήσουμε) σχεδιάζονται δηλαδή βρίσκονται τα n και k με τη βοήθεια των παρακάτω γενικών θεωρητικών σχέσεων, [11,12]:

$$n = 2^m - 1 \tag{9.1}$$

$$n - k \le mt \tag{9.2}$$

$$d_{\min} = 2t + 1 \tag{9.3}$$

όπου m ($m \ge 3$) και t είναι αυθαίρετοι θετικοί αριθμοί. Το μέγεθος d_{min} ονομάζεται *ελάχιστη απόσταση Hamming* (minimum Hamming distance) και μας δηλώνει αριθμητικά μέσω της παραμέτρου t, ότι ο συγκεκριμένος κώδικας είναι δυνατό να διορθώσει μέχρι και t σφάλματα ανά κωδική λέξη. Το μήκος της κωδικής λέξης n σε έναν κώδικα BCH δίνεται πάντα από τη σχέση (9.1) ενώ ο αριθμός των λαθών που μπορεί να διορθώσει ο κώδικας (ικανότητα διόρθωσης λαθών του κώδικα) περιορίζεται από την επόμενη σχέση:

$$t < (2^m - 1)/2 \tag{9.4}$$

Συνεπώς, ο σχεδιαστής ενός τηλεπικοινωνιακού συστήματος έχει τη δυνατότητα να επιλέξει από ένα μεγάλο σύνολο από μήκη κωδίκων και ρυθμών κωδίκων. Τα πολυώνυμα γεννήτορες των BCH κωδίκων προκύπτουν από τους *παράγοντες* (factors) των πολυωνύμων $p^{2^m-1}+1$. Οι μη δυαδικοί BCH κώδικες περιλαμβάνουν τους κώδικες *Reed-Solomon* οι οποίοι θα περιγραφτούν σε επόμενη άσκηση.

Στη κωδικοποίηση ενός (n,k) BCH κώδικα, ένα μήνυμα είναι ένας k στηλών δυαδικός *πίνακας Galois*, [14-16]. Ένα μαθηματικό *πεδίο Galois* (Galois field) είναι ένα αλγεβρικό σύνολο (πεδίο) το οποίο αποτελείται από ένα πεπερασμένο αριθμό στοιχείων. Ένα πεδίο Galois, είναι δυνατό να έχει 2^m στοιχεία όπου m είναι ένας ακέραιος αριθμός μεταξύ του 1 και του 16. Ένα τέτοιο πεδίο Galois συμβολίζεται ως GF(2^m). Τα πεδία

Galois βρίσκουν μεγάλη εφαρμογή στη θεωρία *κωδικοποίησης* και *ανίχνευσης-διόρθωσης λαθών* (error-control coding). Συγκεκριμένα, για ένα BCH κώδικα, κάθε *γραμμή* (row) ενός αντίστοιχου πίνακα Galois, αναπαρασταίνει μία κωδική λέξη του κώδικα BCH.

Στους κώδικες BCH, το μήκος *k* του κάθε μηνύματος είναι ένας θετικός ακέραιος πάντα μικρότερος του *n*. Πρέπει να σημειωθεί ότι μόνο κάποιες τιμές από τους θετικούς ακεραίους τους μικρότερους από το *n* είναι επιτρεπτές ως τιμές του *k*. Για το λόγο αυτό δίνονται στη βιβλιογραφία αντίστοιχοι πίνακες με τις αποδεκτές τιμές των παραμέτρων *n* και *k*, [11].

Τέλος, οι κώδικες BCH έχουν πολλές εφαρμογές στα τηλεπικοινωνιακά συστήματα. Μία από τις πιο γνωστές είναι η εφαρμογή τους στα *κυψελωτά συστήματα κινητής τηλεφωνίας* (cellular mobile telephony systems) όπου ένας *περιορισμένου μήκους κώδικας BCH* (shortened BCH code) χρησιμοποιείται για τα *σήματα σηματοδοσίας* (signaling messages) που δηλώνουν στον *κινητό σταθμό* (mobile unit), εκτός των άλλων στοιχείων, την ισχύ που πρέπει να εκπέμψει ο κινητός σταθμός και σε ποια συγκεκριμένη συχνότητα του συστήματος.

9.3 Εργαστηριακό μέρος

Πρόγραμμα 1^{ο}

```
n=15; k=5; % the parameters of the BCH code
msg = [1 0 0 1 0]  %  one message with k=5 bits length
gp= bchpoly (15, 5) %  generation of the polynomial generator for the BCH code
c = encode(msg, n, k, 'bch',gp) %  encoding the message
```

Εξήγηση 1^{ου} Προγράμματος

Αρχικά ορίζουμε τις παραμέτρους του BCH κώδικα, *n* και *k* μέσω της εντολής:

```
n=15; k=5; % the parameters of the BCH code
```
(9.5)

Οι τιμές που έχουν δοθεί στο συγκεκριμένο πρόγραμμα είναι δεκτές από τη σχετική βιβλιογραφία. Με δεδομένο ότι *k*=5, ορίζουμε ένα *μήνυμα* (msg) πληροφορίας με αριθμό bits ίσο με 5, το οποίο και στη συνέχεια θα κωδικοποιήσουμε με το συγκεκριμένο κώδικα BCH.

Ένας κώδικας BCH,όπως έχει ήδη αναφερθεί στο θεωρητικό μέρος της άσκησης, είναι ένας κυκλικός κώδικας. Έτσι η παραγωγή των κωδικών του λέξεων θα γίνεται μέσω ενός πολυωνύμου γεννήτορα. Στο προηγούμενο πρόγραμμα, η παραγωγή του πολυωνύμου γεννήτορα (μεταβλητή gp) γίνεται με την εντολή:

```
gp= bchpoly (15, 5) %  generation of the polynomial generator for the BCH code
```
(9.6)

η οποία έχει την παρακάτω γενική σύνταξη:

```
"πολυώνυμο γεννήτορας"= = bchpoly (15, 5)
```
(9.7)

Τέλος, στο πρόγραμμα προχωρούμε στη κωδικοποίησης του μηνύματος (msg) με την επόμενη εντολή:

```
c = encode(msg, n, k, 'bch',gp) %  encoding the message
```
(9.8)

παράγοντας την αντίστοιχη κωδική λέξη την οποία ονομάζουμε ως c.

Έξοδος 1ου προγράμματος

```
>> n=15; k=5;
msg = [1 0 0 1 0]

msg =

    1     0     0     1     0

gp= bchpoly (15, 5)

gp =

    1     1     1     0     1     1     0     0     1     0     1

c = encode(msg, n, k, 'bch',gp)

c =

    1
    0
    0
    0
    0
    1
    1
    1
    0
    1
    1
    0
    0
    1
    0
>>
```

Πρόγραμμα 2^{ο}

```
n=15; k=5; % the parameters of the BCH code
dat=randint(1,k)   %  one random message with length k bits
genpoly=bchpoly(n,k); % generation of the generator polynomial for BCH code
msg = gf(dat);   % Galois field
c1=encode(dat, n,k, 'bch', genpoly)   %  encoding the ‘dat’ with the (n,k) BCH code
t=2  % assuming  t errors in each codeword
noise=randerr(1,n,t) %  production of random  noise
codeword_noisy=rem(c1+noise',2)% received codeword with noise(XOR beetween arrays)
dec_message=decode(codeword_noisy,n,k,'bch',genpoly)   %  decoding the codeword with
noise
```

Εξήγηση 2^{ου} Προγράμματος

Όπως και στο πρόγραμμα που προηγήθηκε αρχικά, ορίζουμε τις δύο παραμέτρους του κώδικα BCH δηλαδή δίνουμε τιμές στα *n* και *k*. Αυτό γίνεται μέσω της εντολής:

```
n=15; k=5; % the parameters of the BCH code
```
(9.9)

Είναι δυνατό στο MATLAB, οι κωδικές λέξεις που θα κωδικοποιήσουμε με τη χρησιμοποίηση ενός κώδικα να επιλεγούν τυχαία. Έτσι στην εντολή:

```
dat=randint(1,k)   %  one random message with length k bits
```
(9.10)

παράγουμε *τυχαία* (random) μία κωδική λέξη, με μήκος *k* bits, δηλαδή κωδική λέξη στην οποία οι τιμές των bits αυτής έχον επιλεγεί με τυχαίο τρόπο με τη συγκεκριμένη εντολή. Αν θέλαμε να παράγουμε τυχαία *l* κωδικές λέξεις μήκους *k* bits, η εντολή θα ήταν στην παρακάτω γενική μορφή:

```
dat=randint(l,k)
```
(9.11)

όπως έχουμε ήδη αναφέρει, ένας κώδικας BCH είναι ένας κυκλικός κώδικας. Η εντολή:

```
genpoly=bchpoly(n,k); % generation of the generator polynomial for BCH code
```
(9.12)

μας δίνει ως έξοδο το πολυώνυμο γεννήτορα του κώδικα BCH. Συγκεκριμένα, η έκφραση που λαμβάνεται στο MATLAB δίνει τους συντελεστές του πολυωνύμου γεννήτορα του συγκεκριμένου κώδικα BCH κατά αύξουσα σειρά.

Με την εντολή:

```
msg = gf(dat);   % Galois field
```
(9.13)

δηλώνουμε ότι το μήνυμα (msg) είναι ένα στοιχείο πεδίου Galois. Η κωδικοποίηση του στοιχείου (dat) πραγματοποιείται με την εντολή:

```
c1=encode(dat, n,k, 'bch', genpoly)   %  encoding the ‘dat’ with the (n,k) BCH code
```
(9.14)

Στη συνέχεια του προγράμματος θεωρούμε ότι κάθε κωδική λέξη παρουσιάζει t λάθη (πλήθος λαθών), κάτι το οποίο δίνεται όπως ακολουθεί:

```
t=2  % assuming  t errors in each codeword
```
(9.15)

Επίσης, με την επόμενη εντολή, προχωρούμε στην παραγωγή τυχαίου θορύβου (noise) δηλαδή παράγεται ένα διάνυσμα μήκους n (=15) στο οποίο κάποιες θέσεις του έχουν την τιμή 1:

```
noise=randerr(1,n,t) %  production of random  noise
```
(9.16)

Η κωδική λέξη στην οποία θα υπάρχει και τυχαίος θόρυβος, (λαμβανόμενη κωδική λέξη παρουσία θορύβου) θα προκύπτει ως το λογικό άθροισμα ("λογικό Η") της αρχικής κωδικής λέξης και του διανύσματος του τυχαίου θορύβου δηλαδή θα πρέπει να γραφτεί:

```
codeword_noisy=rem(c1+noise',2)% received codeword with noise  (XOR beetween arrays)
```
(9.17)

Τέλος, η αποκωδικοποίηση της λαμβανόμενης κωδικής λέξης (codeword_noisy) πετυχαίνεται με την εκτέλεση της συνάρτησης decode, όπως παρουσιάζεται στην εντολή που ακολουθεί:

```
dec_message=decode(codeword_noisy,n,k,'bch',genpoly)  %decoding the codeword with

noise
```
(9.18)

η έξοδος της οποίας είναι το *αποκωδικοποιημένο μήνυμα* (dec_message).

Έξοδος προγράμματος

```
>> n=15; k=5;
dat=randint(1,k) % one message with length k bits

dat =

    0     1     0     1     0

genpoly=bchpoly(n,k);    % generation of the generator polynomial for BCH code
msg = gf(dat);   %  Galois field
c1=encode(dat, n,k, 'bch', genpoly)   % encoding with BCH code

c1 =

    0
    0
    0
    1
    1
    1
    0
    1
    1
    0
    0
    1
    0
    1
```

```
    0

t=2     % t errors in each codeword

t =

    2

noise=randerr(1,n,t)  % addittion of noise

noise =

  Columns 1 through 13

    0     1     0     0     1     0     0     0     0     0     0     0     0

  Columns 14 through 15

    0     0

codeword_noisy=rem(c1+noise',2)   %   XOR between Arrays

codeword_noisy =

    0
    1
    0
    1
    0
    1
    0
    1
    1
    0
    0
    1
    0
    1
    0

dec_message=decode(codeword_noisy,n,k,'bch',genpoly)   % decoding process

dec_message =

    0
    1
    0
    1
    0

>>
```

9.4 Πρόσθετες εργασίες

1.Να βρείτε και να εκφράσετε σε μαθηματική μορφή το πολυώνυμο γεννήτορα ενός κώδικα BCH (31,26).

2. Μέχρι πόσα λάθη ανά κωδική λέξη μπορεί να διορθώσει ένας BCH κώδικας (15,7);

3. Βρείτε μέσω προγράμματος την κωδική λέξη που αντιστοιχεί στο μήνυμα [100110] για ένα BCH κώδικα (31,6).

ΑΣΚΗΣΗ 10η

Reed-Solomon Κώδικες-Κωδικοποίηση-Αποκωδικοποίηση

10.1 Σκοπός της άσκησης

Σκοπός της συγκεκριμένης άσκησης είναι η παραγωγή των κωδικών λέξεων ενός κώδικα Reed-Solomon, [15]. Επίσης, με τη χρησιμοποίηση του λογισμικού MATLAB κωδικοποιούνται και αποκωδικοποιούνται μηνύματα με ένα κώδικα Reed-Solomon συγκεκριμένων παραμέτρων. Τέλος, πραγματοποιείται αποκωδικοποίηση των λαμβανομένων κωδικών λέξεων του κώδικα Reed-Solomon, παρουσία *τυχαίου θορύβου* (random noise) του καναλιού επικοινωνίας στο οποίο θεωρούμε ότι μεταδίδεται η κωδική λέξη του κώδικα.

10.2 Θεωρητικό μέρος

Οι κώδικες Reed-Solomon σχεδιαστήκανε το1960 από τους Reed και Solomon, [15]. Οι κώδικες αυτοί είναι *μη δυαδικοί κώδικες* (non binary codes)και έχουν μεγάλη σημασία για τα τηλεπικοινωνιακά συστήματα στα οποία τα σφάλματα λόγω θορύβου του καναλιού επικοινωνίας εμφανίζονται κατά *ριπές* (bursts errors) όπως επίσης και για τα συστήματα ακουστικών CD (Compact Disk) (compact disk audio technology).

Οι κώδικες Reed-Solomon, είναι μπλοκ κώδικες οι οποίοι χρησιμοποιούν αλφάβητα εισόδου και εξόδου με πλήθος συμβόλων 2^m δηλαδή $\{0,\ 1,\ 2,\ ...,\ 2^m - 1\}$.

Το μήκος της κωδικής λέξης n (αριθμός συμβόλων ανά κωδική λέξη) (μήκος μη-δυαδικής κωδικής λέξης) είναι ίσο με 2^m-1 (ακέραιες τιμές μεταξύ 3 και 2^m-1). Το προηγούμενο μήκος του κώδικα μπορεί να αυξηθεί σε 2^m ή 2^m+1 αν κάτι τέτοιο είναι επιθυμητό. Οι κώδικες σχεδιάζονται για να διορθώνουν e_0 λάθη σε ένα μπλοκ από n σύμβολα. Το μήκος του μηνύματος που κωδικοποιείται είναι k (αριθμός συμβόλων ανά μήνυμα) (θετικός ακέραιος μικρότερος από n έτσι ώστε το $(n-k)$ να είναι άρτιος) και ο αριθμός των απαιτουμένων *bits ελέγχου ισοτιμίας* (parity symbols) για να είναι δυνατή η διόρθωση e_0 λαθών είναι $(n-k)=n-2e_0= 2^m-1$. Ο αριθμός m των bits/σύμβολο, είναι ακέραιος αριθμός μεταξύ του 3 και του 16. Με δεδομένες τις προηγούμενες παραμέτρους, ο κώδικας Reed-Solomon είναι δυνατό να διορθώσει μέχρι $t = (n-k)/2$ λάθη (error-correction capability of the code). Οι κώδικες Reed-Solomon παρουσιάζουν ελάχιστη απόσταση $d_{min}=(n-k+1)$ γεγονός που τους καθιστά ιδιαίτερα ελκυστικούς.

10.3 Εργαστηριακό μέρος

Πρόγραμμα 1ο

```
echo off
n=7;
k=3;
m=3;
msg=gf([1 6 4; 0 4 3], m);
c=rsenc(msg,n,k)
```

Εξήγηση 1ου Προγράμματος

Στην αρχή του προγράμματος απαιτούμε να μην εμφανίζονται στην έξοδο οι εντολές που εκτελούνται χρησιμοποιώντας την εντολή echo off. Στη συνέχεια ορίζουμε τις παραμέτρους του κώδικα Reed-Solomon δηλαδή δίνουμε συγκεκριμένες τιμές στον αριθμό των συμβόλων ανά κωδική λέξη n και θεωρούμε και τον αριθμό των συμβόλων k

στο κάθε μήνυμα που θα κωδικοποιήσουμε στη συνέχεια. Κάθε μήνυμα το οποίο θα κωδικοποιηθεί αποτελείται από *m*=3 bits και στο συγκεκριμένο πρόγραμμα με την εντολή:

```
msg=gf([1 6 4; 0 4 3],m);                    (10.1)
```

ζητούμε να κωδικοποιηθούν τα δύο μηνύματα μήκους *k*=3 τα οποία και δίνονται μέσω ενός πίνακα Galois (gf). Με την τελευταία εντολή του προγράμματος προχωρούμε στην κωδικοποίηση (ρουτίνα "rsenc") του μηνύματος msg. Η έξοδος του προγράμματος παρουσιάζεται παρακάτω. Παρατηρούμε ότι κάθε μία από τις γραμμές του πίνακα Galois (μήνυμα) κωδικοποιείται ώστε τελικά να παραχθεί μία κωδική λέξη με *n*=7 σύμβολα επιλεγμένα από το αλφάβητο με 2^(3)=8 στοιχεία. Στην έξοδο του προγράμματος παρουσιάζεται και το πολυώνυμο γεννήτορας του κώδικα (primitive polynomial).

Έξοδος 1ου προγράμματος

```
c = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

   1   6   4   4   3   6   3
   0   4   3   3   7   4   7
```

Πρόγραμμα 2ο

```
echo off
m=3;    %  number of bits per symbol
n=2^m-1;   %  codeword length
k=3;   % message length
t=(n-k)/2;  %  error correcting capability of the code
nw=5;     % number of word to process
msgw=gf(randint(nw,k,2^m),m);    % random k-symbol messages
c=rsenc(msgw,n,k)    %  coding the message "msgw"
d=rsdec(c,n,k)       %  decoding the message "msgw"
noise =(1+randint(nw,n,2^m-1)).*randerr(nw,n,t); % t errors/row
cnoisy = c +noise    % codeword with noise
[dc,nerrs,corrcode] = rsdec(cnoisy,n,k); % decoding process
isequal(dc,msgw) & isequal(corrcode, c)     %   the  decoder can correct t errors/row
noise1=(1+randint(nw,n,2^m-1)).*randerr(nw,n,t+1); % add more than t errors/row   ((t+1)
eroors/row)
cnoisy1 = c +noise1 % new codeword with "noise1"
[dc,nerrs,corrcode] = rsdec(cnoisy1,n,k); % decoding the "cnoisy1" codeword
isequal(dc,msgw) & isequal(corrcode, c) % the  decoder can not correct t+1 errors
```

Εξήγηση 2ου Προγράμματος

Όπως και στο πρόγραμμα που προηγήθηκε, στην αρχή του προγράμματος εισάγουμε τις παραμέτρους του συγκεκριμένου κώδικα Reed-Solomon με τις επόμενες εντολές:

```
m=3;    %  number of bits per symbol
n=2^m-1;   %  codeword length                    (10.2)
k=3;   % message length
```

t=(n-k)/2;% error correcting capability of the code

Θα πρέπει να παρατηρήσουμε ότι δίνουμε ως είσοδο και τον αριθμό των λαθών *t* που ο κώδικας Reed-Solomon μπορεί να διορθώσει. Στη συνέχεια θέτουμε τον αριθμό των λέξεων (number of words) (*nw*) οι οποίες στη συνέχεια θα κωδικοποιηθούν μέσω της εντολής:

nw=5; % number of word to process (10.3)

Έτσι στην επόμενη εντολή του προγράμματος:

msgw=gf(randint(nw,k,2^m),m); % random k-symbol messages (10.4)

επιλέγουμε με τυχαίο τρόπο τα σύμβολα της κάθε μιας από αυτές τις κωδικές λέξεις με *k*=3 σύμβολα (και με 3 bits κάθε σύμβολο) και η κάθε μία από αυτές επιλέγεται από ένα σύνολο με 8 στοιχεία.Η κωδικοποίηση με τον συγκεκριμένο κώδικα Reed-Solomon πραγματοποιείται στη συνέχεια με την εντολή:

c=rsenc(msgw,n,k) % coding the message "msgw" (10.5)

παράγοντας τις αντίστοιχες κωδικές λέξεις (*c*).

Αμέσως μετά δείχνουμε, με την επόμενη εντολή, πως θα γίνει η αποκωδικοποίηση των παραγομένων κωδικών λέξεων *c* δίνοντας ως έξοδο τα αρχικά μηνύματα *d*:

d=rsdec(c,n,k) % decoding the message "msgw" (10.6)

Μπορούμε να παράγουμε τυχαίο θόρυβο (θεωρείται ότι προέρχεται από το κανάλι επικοινωνίας) και να δημιουργήσουμε λάθη σε κάθε γραμμή του πίνακα που μας δείχνει το σύνολο των κωδικοποιημένων λέξεων που έχουμε παράγει προηγούμενα. Έτσι με τις εντολές:

noise =(1+randint(nw,n,2^m-1)).*randerr(nw,n,t); % t errors/row (10.7)

cnoisy = c +noise % codeword with noise (10.8)

παράγουμε αρχικά τον *θόρυβο* (noise) και τον προσθέτουμε στην κωδική λέξη *c* (πίνακα κωδικών λέξεων) παράγοντας την ενθόρυβη κωδική λέξη (ενθόρυβο πίνακα κωδικών λέξεων) *cnoisy* κάθε γραμμή του οποίου παρουσιάζει *t*=2 τυχαία λάθη λόγω θορύβου. Η ενθόρυβη κωδική λέξη αποκωδικοποιείται:

[dc,nerrs,corrcode] = rsdec(cnoisy,n,k); % decoding process noise (10.9)

Με την εντολή:

isequal(dc,msgw) & isequal(corrcode, c) % the decoder can correct t errors/row (10.10)

εξετάζεται αν η αποκωδικοποιημένη κωδική λέξη (dc) συμπίπτει με τη σωστή αρχικά εκπεμπόμενη κωδική λέξη (corrcode). Επειδή έχουμε εισάγει *t*=2 λάθη ανά γραμμή στο πίνακα των κωδικών λέξεων και ο κώδικας μπορεί να διορθώσει μέχρι 2 λάθη ανά γραμμή, οι κωδικές λέξεις συμπίπτουν και η εντολή (10.10) δίνει ως έξοδο λογικό “1”. Στη συνέχεια εισάγουμε (*t*+1) λάθη σε κάθε μία από τις κωδικές λέξεις του πίνακα, παράγοντας μία νέα *ενθόρυβη κωδική λέξη* (*noise1*):

noise1=(1+randint(nw,n,2^m-1)).*randerr(nw,n,t+1);%add more than t errors/row ((t+1)

eroors/row (10.11)

και η οποία στη συνέχεια αποκωδικοποιείται με την εντολή:

[dc,nerrs,corrcode]=rsdec(cnoisy1,n,k); % decoding the "cnoisy1" codeword (10.12)

Στη περίπτωση αυτή η σύγκριση της αποκωδικοποιημένης κωδικής λέξης (δηλαδή του πίνακα κωδικών λέξεων) και της αρχικής κωδικής λέξης δεν οδηγεί σε σύμπτωση διότι η δυνατότητα του συγκεκριμένου κώδικα Reed-Solomon είναι να διορθώνει μέχρι *t* λάθη οφειλόμενα στο θόρύβο του καναλιού επικοινωνίας. Έτσι η έξοδος της τελευταίας εντολής του προγράμματος:

isequal(dc,msgw) & isequal(corrcode, c)% the decoder can not correct t+1 errors (10.13)

δίνει λογικό "0".

Έξοδος 2^{ου} προγράμματος

>>
c = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

1	4	3	5	6	2	0
5	7	1	1	3	7	3
2	7	1	5	4	3	0
7	2	4	0	1	6	5
1	3	5	5	7	3	7

d = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

1	4	3
5	7	1
2	7	1
7	2	4
1	3	5

cnoisy = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

2	4	3	5	6	2	3
5	7	1	7	3	5	3
2	7	0	5	7	3	0
0	2	4	0	1	6	1

```
    1     3     4     3     7     3     7


ans =
    1
cnoisy1 = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)

Array elements =

    1     4     3     0     1     2     4
    5     0     5     1     3     1     3
    0     1     1     5     7     3     0
    3     1     4     0     1     7     5
    1     3     5     0     7     7     3


ans =

    0

>>
```

10.4 Πρόσθετες εργασίες

1. Να γράψετε πρόγραμμα στο MATLAB το οποίο θα παράγει τις κωδικές λέξεις ενός κώδικα Reed-Solomon με τις εξής παραμέτρους:

   - αριθμός bits/σύμβολο του κώδικα: 5
   - ελάχιστη απόσταση του κώδικα: 9

   Να κωδικοποιήσετε μέσω του προγράμματος δύο μηνύματα πληροφορίας κατάλληλου μήκους *k*. Μέχρι πόσα λάθη μπορεί να διορθώσει ο κώδικας αυτός;

2. Σας δίνεται το παρακάτω πρόγραμμα στο MATLAB:

```
echo on
m=4;
n=2^m-1;
k=11;
t=(n-k)/2;
nw=7;
msgw=gf(randint(nw,k,2^m),m);
c=rsenc(msgw,n,k
noise =(1+randint(nw,n,2^m-1)).*randerr(nw,n,t+3);
cnoisy = c +noise
d=rsdec(c,n,k)
[dc,nerrs,corrcode] = rsdec(cnoisy,n,k);
isequal(dc,msgw) & isequal(corrcode, c)
```

   Να εκτελέσετε το προηγούμενο πρόγραμμα, να εξηγήσετε το σκοπό κάθε εντολής του και να ερμηνεύσετε την έξοδο κάθε εντολής του προγράμματος

ΑΣΚΗΣΗ 11η

Επιδόσεις γραμμικών Hamming κωδίκων σε ορθογώνιο και σε μη ορθογώνιο σύστημα σηματοδοσίας

11.1 Σκοπός της άσκησης

Σκοπός της άσκησης είναι να υπολογιστεί και να αναπαρασταθεί μέσω προγραμμάτων στο MATLAB η απόδοση των γραμμικών Hamming κωδίκων μπλοκ, για τα δύο διαφορετικά συστήματα σηματοδοσίας δηλαδή για τη *ορθογώνια σηματοδοσία* (orthogonal signaling) και για τη *μη ορθογώνια σηματοδοσία* (antipodal signaling). Η άσκηση αναφέρεται και στους δύο τρόπους αποκωδικοποίησης δηλαδή *απλής απόφασης* (soft-decision decoding) και *αυστηρής απόφασης* (hard-decision decoding).

11.2 Θεωρητικό μέρος

Στην μη ορθογώνια σηματοδοσία το λογικό "0" μεταδίδεται στο κανάλι επικοινωνίας μέσω ενός παλμού με πλάτος +A και το λογικό "1" μεταδίδεται στο κανάλι επικοινωνίας μέσω ενός παλμού –A. Αντίστοιχα, στη περίπτωση της ορθογώνιας σηματοδοσίας, το λογικό "0" μεταδίδεται στο κανάλι μέσω ενός παλμού με πλάτος 0 και το λογικό "1" μεταδίδεται στο κανάλι μέσω ενός παλμού +A.

Στους γραμμικούς κώδικες μπλοκ η αποκωδικοποίηση είναι δυνατό να επιτευχθεί με δύο τρόπους, όπως έχει ήδη αναφερθεί.

Στην αποκωδικοποίηση απλής απόφασης, η λαμβανόμενη κωδική λέξη αντιστοιχείται σε εκείνο το αρχικό μήνυμα του οποίο παρουσιάζει την *ελάχιστη Ευκλείδεια απόσταση* d^E (minimum Euclidean distance) από το λαμβανόμενο μήνυμα. Αν η ελάχιστη απόσταση ενός γραμμικού κώδικα μπλοκ είναι d_{min}, τότε η ελάχιστη Ευκλείδεια απόσταση δίνεται, για τα δύο σχήματα σηματοδοσίας, από τη σχέση, [11,12]:

$$d^E = \begin{cases} \sqrt{d_{\min} E}, & \text{για ορθογώνια σηματοδοσία} \\ \sqrt{2 d_{\min} E}, & \text{για μη ορθογώνια σηματοδοσία} \end{cases} \qquad (11.1)$$

όπου στη σχέση (11.1), E είναι η ενέργεια ανά στοιχείο (bit) της λαμβανόμενης κωδικής λέξης.

Όπως ήδη γνωρίζουμε κάθε κωδική λέξη ενός (n,k) γραμμικού κώδικα μπλοκ, έχει n στοιχεία (bits) συνεπώς η ενέργεια ανά κάθε κωδική λέξη του κώδικα είναι nE και δεδομένου ότι κάθε κωδική λέξη περιέχει k bits πληροφορίας, η ενέργεια ανά bit E_b βρίσκεται όπως ακολουθεί:

$$E_b = \frac{nE}{k} = \frac{E}{R_c} \qquad (11.2)$$

όπου στη σχέση(11.2), R_c είναι η *απόδοση* (ή *ρυθμός*) (rate) του κώδικα ίση με k/n (το μέγεθος αυτό δηλώνει το ποσοστό των bits στην παραγόμενη κωδική λέξη, τα οποία είναι "ωφέλιμη" πληροφορία δηλαδή είναι τα bits του μηνύματος το οποίο έχει κωδικοποιηθεί).

Η πιθανότητα λάθους αποκωδικοποίησης p_e στη λαμβανόμενη κωδική λέξη (μήνυμα) (πιθανότητα λάθους αποκωδικοποίησης του μηνύματος) (message errror) για ένα γραμμικό κώδικα μπλοκ και για τη περίπτωση αποκωδικοποίησης ***απλής απόφασης***, περιορίζεται από τη παρακάτω σχέση:

$$p_e \le (M-1)\cdot Q\left(\frac{d^E}{\sqrt{2N_0}}\right) \tag{11.3}$$

όπου στη σχέση (11.3), $M=2^k$ είναι ο αριθμός των κωδικών λέξεων του γραμμικού κώδικα μπλοκ (μία κωδική λέξη παράγεται για κάθε έναν από τους διαφορετικούς συνδυασμούς των k bits πληροφορίας του μηνύματος) και N_0 είναι η φασματική πυκνότητα του θορύβου του καναλιού επικοινωνίας. Η σχέση (11.3) με τη βοήθεια του συνδυασμού των (11.1) και (11.2) γίνεται:

$$p_e \le \begin{cases} (M-1)\cdot Q\left(\dfrac{d_{\min}R_cE_b}{\sqrt{2N_0}}\right), \text{για ορθογώνια σηματοδοσία} \\ (M-1)\cdot Q\left(\dfrac{d_{\min}R_cE_b}{\sqrt{N_0}}\right), \text{για μη ορθογώνια σηματοδοσία} \end{cases} \tag{11.4}$$

Τα όρια που δίνει η σχέση (11.4) συνήθως δίνονται μόνο για μεγάλες τιμές του λόγου $\gamma_b = \dfrac{E_b}{N_0}$.

Αντίστοιχα, στην περίπτωση αποκωδικοποίησης ***αυστηρής απόφασης***, η πιθανότητα λάθους αποκωδικοποίησης p_e στη λαμβανόμενη κωδική λέξη (λάθος αποκωδικοποίηση μηνύματος) περιορίζεται από την επόμενη σχέση:

$$p_e \le (M-1)\cdot\left[4p(1-p)\right]^{\frac{d_{\min}}{2}} \tag{11.5}$$

όπου, στη σχέση (11.5), p είναι η πιθανότητα λάθους στο εκπεμπόμενο bit (δηλαδή στο "0" ή "1") (πιθανότητα λάθους που παρουσιάζει το BSC καναλιού)(crossover probalility). Η πιθανότητα λάθους στο εκπεμπόμενο bit δίνεται από τη παρακάτω σχέση, για τις δύο περιπτώσεις σηματοδοσίας:

$$p = \begin{cases} Q\left(\sqrt{\dfrac{E}{N_0}}\right) = Q\left(\sqrt{\dfrac{R_cE_b}{N_0}}\right), \text{για ορθογώνια σηματοδοσία} \\ Q\left(\sqrt{\dfrac{2E}{N_0}}\right) = Q\left(\sqrt{\dfrac{2R_cE_b}{N_0}}\right), \text{για μη ορθογώνια σηματοδοσία} \end{cases} \tag{11.6}$$

11.3 Εργαστηριακό μέρος

1. Να γραφεί πρόγραμμα στο MATLAB το οποίο θα υπολογίζει και θα αναπαρασταίνει την πιθανότητα λάθους αποκωδικοποίησης μηνύματος σε έναν γραμμικό κώδικα μπλοκ Hamming (15,11) για την περίπτωση αποκωδικοποίησης ***αυστηρής απόφασης*** και για ***ορθογώνια*** σηματοδοσία (orthogonal signaling) συναρτήσει του λόγου $\gamma_b = \dfrac{E_b}{N_0}$ (περιοχή τιμών του γ_b από 0 έως 15dB). Να χρησιμοποιηθεί η παρακάτω μαθηματική σχέση:

$$Q\left(\sqrt{2}\,y\right)=\frac{erfc\,(y)}{2} \tag{11.7}$$

Πρόγραμμα

```
echo off
 gamma_b_db=[0:0.1:15]; % Energy per bit/No in dB (γb in dB)
 gamma_b=10.^(gamma_b_db/10);
 q=(1/2).*erfc(0.6*sqrt(gamma_b)); % orthogonal signaling
 q1=(4*q).*(1-q)
 p_err=2047*(q1.^(3/2));
 loglog(gamma_b,p_err)
 xlabel ('Energy per bit per No')
ylabel ('Symbol error for a Hamming code (15,11) (HD-orthogonal signaling)')
```

Εξήγηση Προγράμματος

Στην πρώτη εντολή του προγράμματος, απαιτούμε να μην εμφανίζονται στην οθόνη οι εντολές που εκτελούνται μέσω της εντολής echo off. Στη συνέχεια ορίζουμε την περιοχή τιμών (όπως ορίζεται από την εκφώνηση της άσκησης) του λόγου $\gamma_b=\frac{E_b}{N_0}$ ενέργειας ανά bit προς τη φασματική πυκνότητα του θορύβου του καναλιού με την επόμενη εντολή:

```
gamma_b_db=[0:0.1:15]; % Energy per bit/No in dB (γb in dB)
```
(11.8)

Ο λόγος $\gamma_b=\frac{E_b}{N_0}$ μετατρέπεται σε καθαρό αριθμό με την εντολή:

```
gamma_b=10.^(gamma_b_db/10);
```
(11.9)

Για τον υπολογισμό της πιθανότητας λάθους αποκωδικοποίησης συμβόλου για αυστηρή αποκωδικοποίηση και για ορθογώνια σηματοδοσία, θα χρησιμοποιήσουμε τη σχέση (11.5) και τον κλάδο της σχέσης (11.6) που αναφέρεται στη ορθογώνια σηματοδοσία. Έτσι προκύπτουν οι τρεις επόμενες εντολές του προγράμματος:

```
q=(1/2).*erfc(0.6*sqrt(gamma_b)); % orthogonal signaling
```
(11.10)

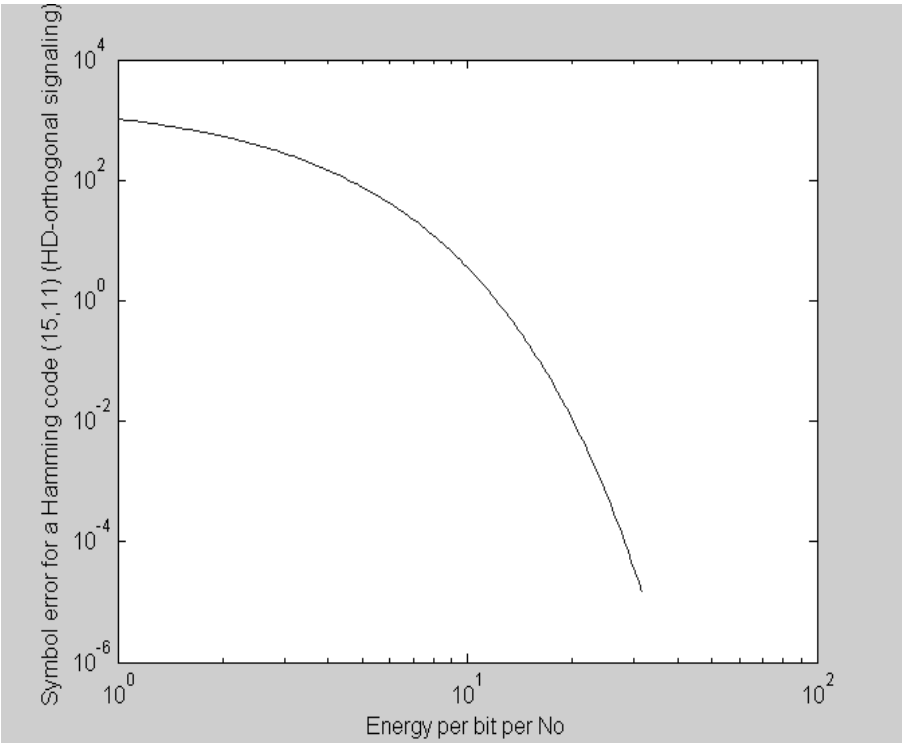
```
q1=(4*q).*(1-q)
```
(11.11)

```
p_err=2047*(q1.^(3/2));
```
(11.12)

θεωρώντας όμως και τη σχέση (11.7) που συνδέει τις συναρτήσεις $Q(y)$ και $erfc(y)$(εξηγείστε πως προκύπτει ο συντελεστής 0.6 στην εντολή (11.11)). Επίσης θα πρέπει να θεωρήσουμε και την απόδοση του κώδικα, η οποία στη περίπτωση μας είναι ίση με $R_c=k/n=11/15=0.733$.

Στη συνέχεια του προγράμματος, προχωρούμε σε αναπαράσταση σε λογαριθμικούς άξονες και για τα δύο μεγέθη, ενώ παράλληλα δίνουμε αντίστοιχες ονομασίες στους δύο άξονες.

Έξοδος προγράμματος



Εικόνα 11.1 Πιθανότητα *λάθους αποκωδικοποίησης συμβόλου* (symbol error) για κώδικα Hamming (15,11) για τη περίπτωση απλής αποκωδικοποίησης με ορθογώνια σηματοδοσία, σε συνάρτηση με το λόγο $\frac{E_b}{N_0}$ ενέργειας ανά bit προς τη φασματική πυκνότητα θορύβου του καναλιού επικοινωνίας (*Έξοδος από εκτέλεση του προγράμματος στο MATLAB*).

11.4 Πρόσθετες εργασίες

1. Να γραφεί πρόγραμμα στο MATLAB το οποίο θα υπολογίζει και θα αναπαρασταίνει την πιθανότητα λάθους αποκωδικοποίησης μηνύματος σε έναν γραμμικό κώδικα μπλοκ Hamming (15,11) για την περίπτωση αποκωδικοποίησης ***αυστηρής απόφασης*** και για ***μη ορθογώνια*** σηματοδοσία (antipodal signaling) συναρτήσει του λόγου $\gamma_b = \frac{E_b}{N_0}$ (περιοχή τιμών του γ_b από 0 έως 15dB). Να χρησιμοποιηθεί η σχέση (11.7).

2. Να γραφεί πρόγραμμα στο MATLAB το οποίο θα υπολογίζει και θα αναπαρασταίνει την πιθανότητα λάθους αποκωδικοποίησης σε ένα γραμμικό κώδικα μπλοκ Hamming (15,11) για την περίπτωση αποκωδικοποίησης ***απλής απόφασης*** και ***για μη ορθογώνια*** σηματοδοσία συναρτήσει του λόγου $\gamma_b = \frac{E_b}{N_0}$ (περιοχή τιμών του γ_b από 0 έως 15dB). Να χρησιμοποιηθεί η σχέση (11.7).
3. Να γραφεί πρόγραμμα στο MATLAB το οποίο θα υπολογίζει και θα αναπαρασταίνει την πιθανότητα λάθους αποκωδικοποίησης σε ένα γραμμικό κώδικα μπλοκ Hamming (15,11) για την περίπτωση αποκωδικοποίησης ***απλής απόφασης*** και για ***ορθογώνια***

σηματοδοσία συναρτήσει του λόγου $\gamma_b = \frac{E_b}{N_0}$ (περιοχή τιμών του γ_b από 0 έως 15dB). Να χρησιμοποιηθεί η σχέση (11.7).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] T.Cover and J.Thomas, *Elements of Information Theory*, New York: *Wiley*, 1991.

[2] Ν.Σ.Τζάννες, *Θεωρία Μετάδοσης Πληροφοριών*, Τόμος ΙΙ, *Εισαγωγή στις Θεωρίες Shannon και Κωδίκων*, Πάτρα, 1981.

[3] Δ.Χ.Βούκαλης, *Θεωρία Πληροφοριών και Κωδίκων,* Εκδόσεις ΙΩΝ, Περιστέρι, 1994.

[4] J.G.Proakis and M.Salehi, *Contemporary Communication Systems Using MATLAB*, PWS, *Publishing Company*, 1998.

[5] J.G. Proakis, *Digital Communications*, 3rd Edit., *McGraw-Hill*, 1995.

[6] C.E.Shannon, "*A mathematical theory of communication*", *Bell System Tech. Journal*, pp. 17-28, July 1948.

[7] C.E.Shannon, "*Communication in the presence of noise*", *Proc. of the IRE*, vol. 37, pp. 10-21, Jan. 1949.

[8] C.E.Shannon and W.Weaver, *A Mathematical Theory of Communication*, Urbana, IL: Univ. Illinois Press, 1949.

[9] R.G.Gallager, "*Information Theory and Reliable Communication*", *John Wiley & Sons*, 1968.

[10] Κ.Καρούμπαλος, *Εισαγωγή στη Θεωρία Θορύβου*, Αθήνα 1986.

[11] J.G.Proakis, M.Salehi, *Μετάφραση*: Κ.Καρούμπαλος, Ζέρβας Ε., Καραμπογιάς Σ., Σαγκριώτης Ε., *Συστήματα Τηλεπικοινωνιών*, Ε.Κ.Π.Α., Αθήνα 2002.

[12] R.Ziemer, R.Peterson, *Introduction to Digital Communication, Mcmillan*, 1992.

[13] H.P.Hsu, *Αναλογικές και Ψηφιακές Επικοινωνίες*, Σειρά Schaum, *Μετάφραση*: Ι.Βαρδιάμπασης, Εκδόσεις Τζιόλας, 2002.

[14] S.Lin and D.J.Costelo, Jr., *Error Control Coding:Fundamentals and Applications*, Prentice Hall, 1983.

[15] I.S.Reed and G.Solomon, "*Polynomial Codes over Certain Finite Fields*", *Journal of the Society for Industrial and Applied Mathematics*, June 1960.