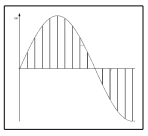
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ – ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΨΗΦΙΑΚΕΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΕΣ

Εργαστηριακή Άσκηση #2	2013-2014	Καφφέζας Γιώργος	**ΑΜ** 4465



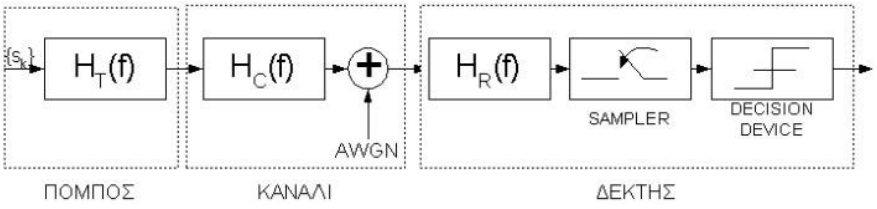
Χαρακτηριστικά συστήματος

Η παρούσα άσκηση υλοποιήθηκε σε λειτουργικό σύστημα Windows 7 Professional SP1 (x64). Όσον αφορά το υλικό του συστήματος, ο επεξεργαστής του είναι Intel Core 2 Quad Q6600 στα 2.4GHz, με διαθέσιμη RAM DDR3 4GB. Τέλος, η έκδοση MATLAB που χρησιμοποιήθηκε είναι η R2012b (8.0.0.783), πάλι για 64-bit, και η αναφορά συντάχθηκε με LibreOffice Writer 3.5.7.2.

• • • • • •

Α' Μέρος

Στο πρώτο μέρος μας ζητείται να υλοποιήσουμε το τηλεπικοινωνιακό σύστημα που περιγράφεται, και έχει σχηματικά την ακόλουθη μορφή:



Το ζητούμενο τηλεπικοινωνιακό σύστημα βασικής ζώνης

Το παραπάνω σύστημα υλοποιείται στον κώδικα που βρίσκεται στο παράρτημα της παρούσας αναφοράς, με όνομα *script_2_a.m*. Για τη σχεδίασή του ακολουθήθηκαν οι υποδείξεις της εκφώνησης, και θα αναλύσουμε στο σημείο αυτό εν συντομία τα μέρη από τα οποία αποτελείται, όπως αυτά φαίνονται και στο σχήμα αλλά και στην περιγραφή του. Λεπτομερέστερη ανάλυσή τους υπάρχει και με σχόλια πάνω στον ίδιο τον κώδικα.

Αρχικά, λοιπόν, έχουμε το σήμα εισόδου μας, το οποίο πρέπει να υπερδειγματοληπτήσουμε κατά 4, δηλαδή πρέπει να διαβάζουμε 4 σύμβολα αντί για 1 ανά περίοδο. Αυτό το επιτυγχάνουμε εισάγοντας 3 μηδενικά ανάμεσα στα σύμβολα της ακολουθίας εισόδου, με τη βοήθεια της ενδογενούς συνάρτησης της MATLAB upsample[1] (αφαιρώντας τα 3 τελευταία μηδενικά που προσθέτει, μιας και θέλουμε μηδενικά μόνο ανά δύο σύμβολα).

Έπειτα, υλοποιούμε το φίλτρο που απαιτείται να υπάρχει στον πομπό, και το κάνουμε ως φίλτρο τετραγωνικής ρίζας ανυψωμένου συνημιτόνου, όπως μας υποδεικνύεται στην άσκηση. Γι' αυτό χρησιμοποιούμε τη συνάρτηση rcosfir με ορίσματα τον roll-off παράγοντα που μας δίνεται (δηλαδή 0.3), με ρυθμό 4 (λόγω των τεσσάρων δειγμάτων ανά περίοδο) και με τα υπόλοιπα στις default[2] τιμές (δηλαδή T_s=1 sec και συνολικό αριθμό περιόδων δειγματοληψίας 6 που εκτείνονται από το -3 έως το 3). Έτσι, λοιπόν, το σήμα που θα φύγει από τον πομπό θα είναι το αποτέλεσμα της συνέλιξης του φίλτρου με το υπερδειγματοληπτημένο σήμα μας. Επιλέχτηκε η συνάρτηση conv[3] της MATLAB για την υλοποίηση αυτή, καθώς η filter αγνοούσε[4] την καθυστέρηση (delay) και επέστρεφε διάνυσμα ίδιου μήκους με της εισόδου, γεγονός που δεν ήταν εύκολα διαχειρίσιμο, τουλάχιστον όχι τόσο, όσο των αποτελεσμάτων της conv.

Στη συνέχεια περνάμε στην υλοποίηση του καναλιού, και καθορίζουμε τη συμπεριφορά του για την περίπτωση που είναι ιδανικό και μη-ιδανικό. Στην περίπτωση που είναι ιδανικό, τότε στο φίλτρο του δέκτη έχουμε το άθροισμα του σήματος μετά από το φίλτρο του πομπού συν τον AWGN θόρυβο, ενώ στην αντίθετη περίπτωση, που το κανάλι δεν είναι ιδανικό, έχουμε τη συνέλιξη του σήματος από τον πομπό με

1 http://www.mathworks.com/help/signal/ref/upsample.html
2 http://radio.feld.cvut.cz/matlab/toolbox/comm/rcosfir.html
3 http://www.mathworks.com/help/matlab/ref/conv.html
4 http://www.edaboard.com/thread39126.html

την κρουστική απόκριση του καναλιού (όπως αυτή καθορίζεται στην εκφώνηση). Την κρουστική απόκριση του καναλιού πρέπει να τη δειγματοληπτήσουμε εξίσου, ώστε να έχουμε τα επιθυμητά αποτελέσματα.

Στο σημείο αυτό πρέπει να προστεθεί ο θόρυβος του συστήματος, ο οποίος θα είναι λευκός Gaussian θόρυβος. Ο θόρυβος αυτός θέλουμε να έχει ισχύ σύμφωνα με το Signal-to-Noise Ration (SNR) που θέλουμε να έχουμε, και αυτό καθορίζεται από τον εξής τύπο $SNR_{db}=10\log_{10}(P_s/\sigma^2)$, όπου σ είναι η διασπορά του θορύβου. Έτσι, υπολογίζουμε[5] πρώτα την ισχύ του σήματος που φεύγει από τον πομπό, και δεδομένου του SNR, υπολογίζουμε τη διασπορά του θορύβου ως τη ρίζα της ισχύος του θορύβου. Εν συνεχεία, διαχωρίζουμε τις περιπτώσεις όπου έχουμε πραγματικούς και μιγαδικούς αστερισμούς, σύμφωνα με την σημείωση της εκφώνησης, και παράγουμε τυχαίες τιμές με τη συνάρτηση randn, ενώ μετά υπολογίζουμε τον θόρυβο και και τον προσθέτουμε στο σήμα του πομπού.

Έχοντας καλύψει την υλοποίηση του πομπού και του καναλιού, φτάνουμε στην υλοποίηση του δέκτη, και αρχικά έχουμε το φίλτρο του. Το φίλτρο του δέκτη είναι ίδιο με το φίλτρο του πομπού, σύμφωνα και με την εκφώνηση αλλά και με τη θεωρία που γνωρίζουμε για άγνωστο κανάλι.

Επίσης, καθώς έχουμε ήδη πραγματοποιήσει υπερδειγματοληψία, οφείλουμε[6] να υλοποιήσουμε υποδειγματοληψία ώστε να μπορέσουμε να έχουμε στην έξοδο του συστήματός μας σήμα όσο πιο παρόμοιο με το αρχικό. Πριν το κάνουμε όμως αυτό λαμβάνουμε υπ' όψιν μας την καθυστέρηση που έχει επέλθει στη μετάδοση του σήματός μας, όπως διευκρινίζεται και στην εκφώνηση της άσκησης, ώστε να «πετάξουμε» όσα σύμβολα από την αρχή και το τέλος του σήματός μας. Έτσι, υπολογίζουμε τον αριθμό αυτών των συμβόλων για την περίπτωση του ιδανικού και του μη-ιδανικού καναλιού, λαμβάνοντας υπ' όψιν τις συνελίξεις που έχουν γίνει και τους μη-αιτιατούς συντελεστές που έχουν προκύψει. Για το ιδανικό κανάλι είναι τα σύμβολα που πετάμε από την αρχή και από το τέλος είναι 24, ενώ για το μη-ιδανικό 48. Οπότε τώρα πραγματοποιούμε και την υποδειγματοληψία, χρησιμοποιώντας τη συνάρτηση downsample[7] της MATLAB.

Τέλος, υλοποιείται η διάταξη απόφασης για την ακολουθία που προκύπτει μετά την υποδειγματοληψία. Χρησιμοποιούμε το κριτήριο Maximum-Likelihood (ML) και τις μετρικές που ορίζονται στη σχέση (7.5.41) στο βιβλίο[8] και υπολογίζουμε τις Ευκλείδιες αποστάσεις για κάθε σύμβολο του σήματος με το σύμβολο του αλφαβήτου, ώστε να επιλέξουμε τη μικρότερη από αυτές (κάνουμε ταξινόμηση με αύξουσα σειρά στο διάνυσμα όπου αποθηκεύουμε τα αποτελέσματα και επιλέγουμε το πρώτο), και να περάσουμε στην έξοδο το αντίστοιχο σύμβολο του αλφαβήτου με το οποίο μοιάζει περισσότερο.

Συνοψίζοντας, λοιπόν, έχουμε πλέον υλοποιημένο το ζητούμενο τηλεπικοινωνιακό σύστημα βασικής ζώνης και μπορούμε να προχωρήσουμε στην εξέταση της επίδοσής του στα διάφορα είδη διαμόρφωσης που μας ζητούνται.

Β' Μέρος

Το παρόν μέρος υλοποιείται στον κώδικα με ονομασία *script_2_b.m* που βρίσκεται στο παράρτημα. Στο μέρος αυτό πρέπει να δώσουμε μια δυαδική ακολουθία διαμορφωμένη κατά πλάτος (2-PAM), με ισοπίθανα σύμβολα αλφαβήτου {-1,+1}. Για να πραγματοποιήσουμε αξιόπιστες μετρήσεις Bit Error Rate (BER) θα ορίσουμε ένα μεγάλο αριθμό δεδομένων, και συγκεκριμένα 10^5 δυαδικά ψηφία δεδομένων (για να έχουμε και την ανάλογη ακρίβεια). Την ακολουθία που μας ζητείται την δημιουργούμε με τη βοήθεια της συνάρτηση randsrc[9] της MATLAB, η οποία επιστρέφει μια ακολουθία από σύμβολα δοσμένου αλφαβήτου με ίσες πιθανότητες εμφάνισης.

Στη συνέχεια καλούμε τη συνάρτηση που υλοποιεί το σύστημα του Α' μέρους για ιδανικό και μη-ιδανικό κανάλι και για τις διάφορες τιμές SNR που έχουμε, και σώζουμε τα σήματα που μας επιστρέφουν. Για να μπορέσουμε να κάνουμε σωστά τις μετρήσεις BER και τις ακόλουθες συγκρίσεις, πρέπει να

5 Υπάρχουν διάφοροι τρόποι υπολογισμού της ισχύος, εμείς επιλέξαμε τον υπολογισμό στο πεδίο του χρόνου, σύμφωνα και με το παράδειγμα εδώ: http://www.mathworks.com/help/signal/examples/measuring-the-power-of-deterministic-periodic-signals.html

6 http://www.eetimes.com/document.asp?doc_id=1275556

7 http://www.mathworks.com/help/signal/ref/downsample.html

8 J. G. Proakis & M. Salehi, «Συστήματα Τηλεπικοινωνιών», σ.428.

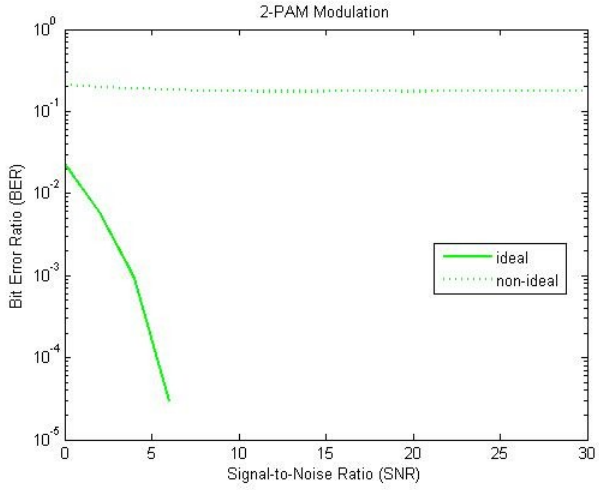
9 http://www.mathworks.com/help/comm/ref/randsrc.html

μετατρέψουμε τα σύμβολα σε δυαδικά ψηφία. Έτσι, από τη στιγμή που έχουμε δύο σύμβολα στο αλφάβητό μας, μας χρειάζεται μόνο ένα bit για την αναπαράστασή τους. Επιλέγουμε, λοιπόν, να αντιστοιχίσουμε το -1 στο 0 και το 1 στο 1, για ευκολία στη μετατροπή.

Έτσι, αφού μετατρέψουμε το σήμα εισόδου και τα διάφορα σήματα εξόδου σε δυαδική αναπαράσταση, προχωράμε στις μετρήσεις BER με τη χρήση της ενδογενούς συνάρτησης της MATLAB biterr,[10] η οποία επιστρέφει τον αριθμό των σφαλμάτων και το ζητούμενο ποσοστό που μας ενδιαφέρει. Τα αποτελέσματα για τις ποικίλες τιμές του SNR είναι τα ακόλουθα:

2-PAM Modulation		
SNR	**BER ιδανικού καναλιού**	**BER μη-ιδανικού καναλιού**
0	0.023010	0.210360
2	0.005830	0.196440
4	0.000930	0.189310
6	0.000030	0.183080
8	0.000000	0.178380
10	0.000000	0.177620
12	0.000000	0.175940
14	0.000000	0.176630
16	0.000000	0.177660
18	0.000000	0.176640
20	0.000000	0.176620
22	0.000000	0.177100
24	0.000000	0.177320
26	0.000000	0.176820
28	0.000000	0.177110
30	0.000000	0.177200

Τα αποτελέσματα αυτά τα περνάμε στο τέλος του κώδικα στο ίδιο γράφημα, το οποίο είναι το ακόλουθο:



10 http://www.mathworks.com/help/comm/ref/biterr.html

Γ' Μέρος

Το παρόν μέρος υλοποιείται στον κώδικα με ονομασία *script_2_c.m* που βρίσκεται στο παράρτημα. Στο μέρος αυτό πρέπει να δώσουμε μια δυαδική ακολουθία διαμορφωμένη κατά πλάτος (4-PAM), με ισοπίθανα σύμβολα αλφαβήτου $\{-3/\sqrt{5}, -1/\sqrt{5}, 1/\sqrt{5}, 3/\sqrt{5}\}$. Τα σύμβολα αυτά προέκυψαν επειδή θέλουμε να μπορούμε να συγκρίνουμε τα αποτελέσματα μεταξύ των διαφορετικών διαμορφώσεων, και για να το κάνουμε αυτό πρέπει η ισχύς του σήματος να είναι ίδια και για τις δύο κωδικοποιήσεις. Έτσι, σύμφωνα και με τη θεωρία στη σελίδα 391 του βιβλίου[11] έχουμε:

$$E_{2\text{-PAM}}=(1/2)*(-1)^2+(1/2)*(+1)^2=1$$

Οπότε, για την διαμόρφωση 4-PAM θα ισχύει ο εξής τύπος, μιας και τα σύμβολα είναι ισοπίθανα:[12]

$$E_{4\text{-PAM}}=(1/4)*(-3x)^2+(1/4)*(-x)^2+(1/4)*(x)^2+(1/4)*(3x)^2=1 \leftrightarrow x=1\sqrt{5}$$

Ακολουθούμε την ίδια λογική όπως και με την διαμόρφωση 2-PAM, δημιουργώντας ένα τυχαίο σήμα εισόδου, δίνοντάς το ως είσοδο στο σύστημά μας για τα διάφορα SNR και τα δύο κανάλια, κτλ. Μόνο που τώρα έχουμε τέσσερα σύμβολα στο αλφάβητο εισόδου. Αυτό μας αναγκάζει να χρησιμοποιήσουμε δύο bit για την αναπαράστασή τους, ώστε να μπορέσουμε να προβούμε στις μετρήσεις και τις ανάλογες συγκρίσεις. Έτσι, αντιστοιχούμε με την σειρά τα σύμβολα ως εξής: το $-3/\sqrt{5}$ στο 00, το $-1/\sqrt{5}$ στο 01, το $1/\sqrt{5}$ στο 10 και το $3/\sqrt{5}$ στο 11. Πραγματοποιώντας τις μετρήσεις BER για τη συγκεκριμένη διαμόρφωση, έχουμε τα εξής αποτελέσματα:

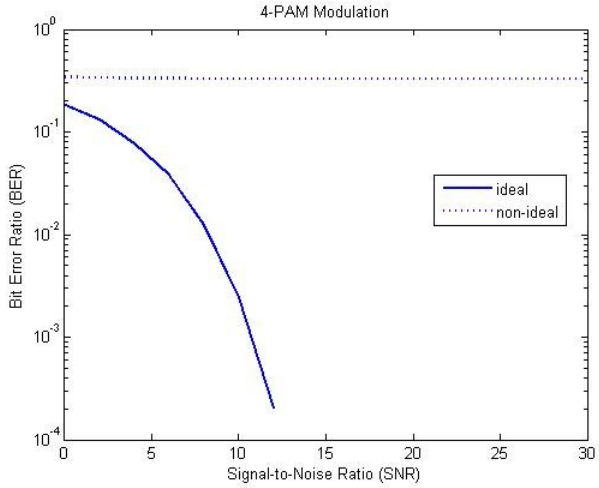
4-PAM Modulation		
SNR	**BER ιδανικού καναλιού**	**BER μη-ιδανικού καναλιού**
0	0.183295	0.343100
2	0.131210	0.337730
4	0.077140	0.335910
6	0.037765	0.332900
8	0.012395	0.330395
10	0.002435	0.329375
12	0.000205	0.329070
14	0.000000	0.328765
16	0.000000	0.328510
18	0.000000	0.328440
20	0.000000	0.327845
22	0.000000	0.327890
24	0.000000	0.327700
26	0.000000	0.328295
28	0.000000	0.328150
30	0.000000	0.327970

11 J. G. Proakis & M. Salehi, «Συστήματα Τηλεπικοινωνιών», σ.391.

12 Αντίστοιχες περιπτώσεις επιβεβαίωσης της επιλογής μας βρίσκονται στις δύο ακόλουθες τοποθεσίες:

- http://www.dsplog.com/2007/10/07/symbol-error-rate-for-pam/
- http://web.mit.edu/6.02/www/f2012/handouts/tutprobs/noise.html

Το γράφημα όπου εμφανίζονται τα παραπάνω αποτελέσματα είναι το ακόλουθο:



Ένα σημείο που αξίζει να παρατηρηθεί, πριν προχωρήσουμε στο επόμενο μέρος, είναι ότι για την αποθήκευση των σημάτων εισόδου και εξόδου του συστήματος με δυαδική αναπαράσταση χρησιμοποιήθηκαν τρισδιάστατα μητρώα[13] με μήκος ίσο με το μήκος του σήματος και πλάτος ίσο με δύο, καθώς πλέον κάθε σύμβολο αντιστοιχίζεται σε δύο bit. Αυτό δεν επηρεάζει την υλοποίηση και τη σύγκριση για τον υπολογισμό του BER σε κάθε περίπτωση, καθώς η συνάρτηση biterr που χρησιμοποιούμε συγκρίνει στοιχείο προς στοιχείο στην αντίστοιχη θέση του μητρώου.

Δ' Μέρος

Το παρόν μέρος υλοποιείται στο αρχείο κώδικα με την ονομασία script_2_d.m που βρίσκεται στο παράρτημα της αναφοράς. Στο μέρος αυτό πρέπει να δώσουμε μια δυαδική ακολουθία διαμορφωμένη κατά πλάτος (4-QAM), με ισοπίθανα σύμβολα αλφαβήτου {(1+j)/√2, (-1+j)/√2, (1-j)/√2, (-1-j)/√2}. Τα σύμβολα αυτά προκύπτουν με τον ίδιο τρόπο που προέκυψαν και στην περίπτωση της 4-PAM, καθώς θέλουμε να μπορούμε να συγκρίνουμε τα αποτελέσματα των διαφορετικών διαμορφώσεων, και γι' αυτό πρέπει να έχουν όλες την ίδια ισχύ.

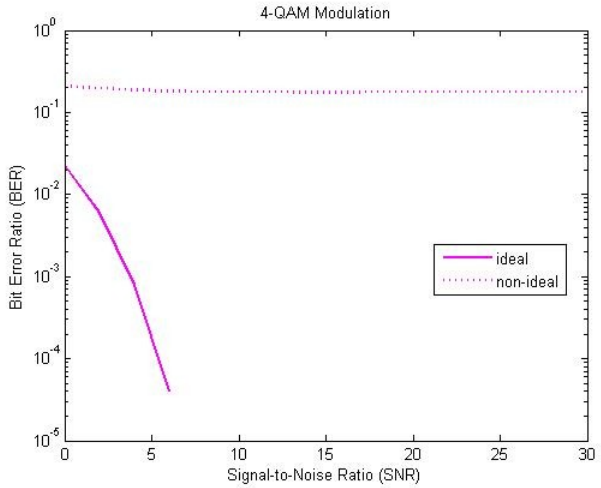
Ακολουθούμε την ίδια λογική με τις προηγούμενες διαμορφώσεις 2-PAM και 4-PAM, και μετατρέπουμε τα σύμβολά μας σε δυαδική αναπαράσταση. Θα ακολουθήσουμε γι' αυτό τις οδηγίες της εκφώνησης, και θα αντιστοιχίσουμε τα δυαδικά ψηφία που βρίσκονται στις ζυγές θέσεις στην μία ορθογώνια φέρουσα και αυτά που βρίσκονται στις μονές στην άλλη. Έτσι, θα τα αντιστοιχίσουμε ως εξής: το (1+j)/√2 στο 11, το (-1+j)/√2 στο 01, το (1-j)/√2 στο 10 και το (-1-j)/√2 στο 00. Πάλι θα χρησιμοποιήσουμε τρισδιάστατα μητρώα για την αποθήκευση των δυαδικών αναπαραστάσεων, χωρίς να έχουμε πρόβλημα στην υπόλοιπη υλοποίηση. Πραγματοποιώντας, λοιπόν, τις μετρήσεις BER για τη συγκεκριμένη διαμόρφωση, έχουμε τα εξής αποτελέσματα:

4-QAM Modulation		
SNR	**BER ιδανικού καναλιού**	**BER μη-ιδανικού καναλιού**
0	0.022545	0.208655
2	0.005930	0.196160
4	0.000790	0.187930
6	0.000040	0.182565

13 http://www.mathworks.com/help/matlab/math/multidimensional-arrays.html

8	0.000000	0.178630
10	0.000000	0.177325
12	0.000000	0.176780
14	0.000000	0.175890
16	0.000000	0.176605
18	0.000000	0.177305
20	0.000000	0.176745
22	0.000000	0.177040
24	0.000000	0.177475
26	0.000000	0.177300
28	0.000000	0.177590
30	0.000000	0.177535

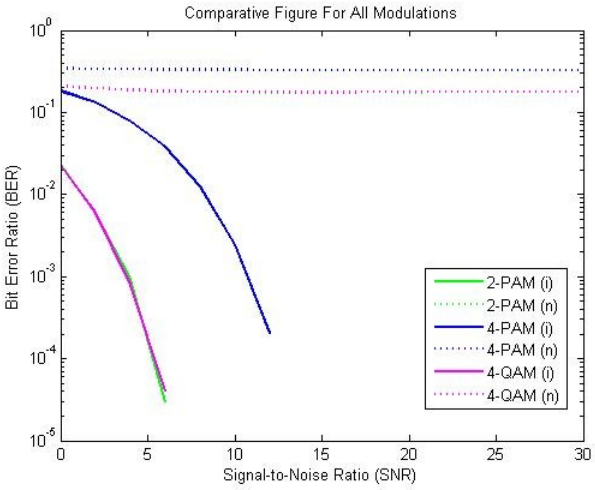
Το γράφημα όπου εμφανίζονται τα παραπάνω αποτελέσματα είναι το ακόλουθο:



Ε' Μέρος

Στο τελευταίο μέρος της παρούσας άσκησης μας ζητείται να παρουσιάσουμε τις μετρήσεις BER και για τα δύο κανάλια, για όλες τις δοθείσες τιμές SNR, και για τα τρία είδη διαμορφώσεων 2-PAM, 4-PAM και 4-QAM. Το μέρος αυτό υλοποιείται στο αρχείο κώδικα με όνομα *script_2_e.m* που βρίσκεται στο παράρτημα της αναφοράς. Η λειτουργία του είναι απλή: αρχικά εκτελεί τα script των προηγούμενων μερών με τη σειρά, αποθηκεύει τα αποτελέσματά τους προσωρινά, σχεδιάζει το κοινό γράφημα που απαιτείται και τέλος γράφει τα αποτελέσματα στο αρχείο *results_2_e.txt* για να είναι συγκεντρωμένα και προσπελάσιμα πιο άμεσα (περιλαμβάνονται ήδη στην αναφορά, μιας και είναι τα στοιχεία των πινάκων που έχουμε παραθέσει παραπάνω). Έπειτα από την συνολική αυτή εκτέλεση, προκύπτει το ακόλουθο γράφημα:

(το γράφημα βρίσκεται στην αμέσως επόμενη σελίδα)



Πριν ξεκινήσουμε τον όποιο σχολιασμό, θα ήταν δόκιμο να διευκρινιστεί ότι θα γίνει σχολιασμός συνολικά για τις γραφικές παραστάσεις που έχουν προκύψει, δηλαδή θα γίνει και σχολιασμός που θα έπρεπε να είναι στο Β', Γ' και Δ' μέρος κάθε φορά. Αυτό επιλέχθηκε κυρίως γιατί μπορούν να ειπωθούν κάποια κοινά πράγματα για όλα τα γραφήματα, και έτσι να εξοικονομηθεί χώρος στην αναφορά.

Παρατηρώντας, λοιπόν, σε μια πρώτη φάση τα αποτελέσματα όπως παρουσιάζονται στο κοινό γράφημα, θα μπορούσαμε να πούμε ότι για την περίπτωση του ιδανικού καναλιού το BER μειώνεται όσο αυξάνεται το SNR, ώσπου παίρνει σχεδόν μηδενικές ή και μηδενικές τιμές. Αυτό συμβαίνει καθώς ο λόγος SNR εκφράζει το πόσο πιο ισχυρό είναι το σήμα μας από τον θόρυβο, και προφανώς, όταν έχουμε ένα ισχυρό σήμα και έναν ασθενή θόρυβο, τότε το ποσοστό λαθών είναι ελάχιστο.

Μια δεύτερη σχετική παρατήρηση είναι ότι σε κάθε περίπτωση διαμόρφωσης και για κάθε διαφορετική τιμή SNR, το BER για το μη-ιδανικό κανάλι είναι σχεδόν σταθερό (με μια ελάχιστη μείωση, που οφείλεται σε αυτό που περιγράψαμε και μόλις πριν). Ο λόγος που δεν παρατηρούμε ιδιαίτερη μείωση, όπως συμβαίνει στο ιδανικό κανάλι, είναι διότι ο θόρυβος του συστήματος είναι πολύ μεγαλύτερος εξαιτίας της συνέλιξης της κρουστικής απόκρισης του καναλιού με το μεταδιδόμενο σήμα μας.

Μια τρίτη παρατήρηση έχει να κάνει με το γεγονός ότι η γραφική παράσταση της διαμόρφωσης 2-PAM συμπίπτει σχεδόν με αυτήν της 4-QAM. Δηλαδή, η πιθανότητα σφαλμάτων και στις δύο διαμορφώσεις είναι περίπου ίδια. Αυτό το βλέπουμε και εξετάζοντας τις τιμές στους πίνακες πιο πάνω, αλλά επιβεβαιώνεται και θεωρητικά,[14] καθώς σύμφωνα με τα όσα ισχύουν για τις πιθανότητες σφάλματος των συμβόλων PAM και QAM, έχουμε τον ακόλουθο τύπο (όπου b είναι ο αριθμός των bit αναπαράστασης):

$$BER_{QAM}(b) \approx BER_{PAM}(b/2)$$

Στην περίπτωσή μας, το b ισούται με 2, καθώς έχουμε 2-PAM και 4-QAM.

Αν συγκρίνουμε, τώρα, τις επιδόσεις του συστήματος για τις διαμορφώσεις 2-PAM και 4-PAM, θα δούμε ότι η δεύτερη έχει αρκετά μεγαλύτερες τιμές BER, και για τα δύο είδη καναλιών. Αυτό είναι αναμενόμενο,[15] καθώς (διαισθητικά) έχουμε πλέον περισσότερα bit να στείλουμε για κάθε σύμβολο, άρα είναι πιθανότερο να αυξηθούν τα λάθη κατά την αποστολή και την απόφαση, μιας και υπάρχουν περισσότερα πιθανά σύμβολα που μπορεί να είναι κοντά, όσον αφορά τις Ευκλείδιες αποστάσεις.

Συνοψίζοντας τις δύο τελευταίες παρατηρήσεις μας, θα μπορούσαμε να πούμε ότι, αφού η 4-QAM έχει την ίδια πιθανότητα λαθών με την 2-PAM, και αφού όμως έχει και την ίδια ικανότητα μετάδοσης πληροφορίας με την 4-PAM, είναι η διαμόρφωση που θα έπρεπε να προτιμήσουμε προς χρήση από αυτές τις τρεις, ως η καλύτερη μεταξύ τους.

---

14 Μια σχετικά λεπτομερής ανάλυση παρατίθεται στο βιβλίο “Signal Processing and Optimization for Transceiver Systems” των P. P. Vaidyanathan, See-May Phoong και Yuan-Pei Lin, και συγκεκριμένα στη σελίδα 27. Το βιβλίο είναι προσβάσιμο εν μέρει μέσω του Google Books: http://books.google.gr/books?id=Icq1bNpr3xUC&pg

15 Βλ. και γραφικές παραστάσεις: http://www.ant.uni-bremen.de/whomes/rinas/dfsim/linearmodreference.html#pam

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΩΝ

Στο σημείο αυτό παρατίθενται όλοι οι κώδικες MATLAB που υλοποιήθηκαν και χρησιμοποιήθηκαν για την εκπόνηση της άσκησης. Παρατίθενται περίπου με τη σειρά που προσδιορίζεται από τα ερωτήματα. Παρουσιάζεται αρχικά το όνομα του αρχείου (με κατάληξη .m) και στη συνέχεια ο ίδιος ο κώδικας, με αριθμημένες γραμμές και μορφοποίηση παρόμοια με τον επεξεργαστή της MATLAB.

• • • • • •

Όνομα αρχείου: *script_2_a.m*

```
1.  function [output] = script_2_a(input,s,channel,SNR) % ---------------------
2.  % INPUT ------------------------------------------------------------------
3.  %   input: input signal
4.  %   s: source's alphabet
5.  %   channel: type of the channel ('i' for ideal - 'n' for non-ideal)
6.  %   SNR: signal to noise ratio (in dB)
7.  % OUTPUT -----------------------------------------------------------------
8.  %   output: output signal
9.  % ------------------------------------------------------------------------
10.
11. % ----------------------------------------------------------[Upsampling input]
12. % Upsampling rate.
13. up_rate = 4;
14. % Upsampling the input signal.
15. upsampled_signal = upsample(input,up_rate);
16. % Calculating the length of the new signal.
17. u_s_length = length(upsampled_signal);
18. % Removing the last three zeros.
19. upsampled_signal = upsampled_signal(1:(u_s_length-(up_rate-1)));
20. % Printing out upsampled signal's length.
21. fprintf('Upsampled signal length: %d\n',length(upsampled_signal));
22.
23. % ----------------------------------------------------[Filter in transmitter]
24. % Specifying the input signal sampling period T_s (in seconds).
25. Ts = 1;
26. % Specifying number of sampling periods.
27. M = 6;
28. % Calculating total period.
29. T = M*Ts;
30. % Specifying the roll-off factor.
31. roll_off = 0.3;
32. % Creating transmitter's raised cosine FIR filter.
33. transmitter_filter = rcosfir(roll_off,[-T/2,T/2],up_rate,Ts,'sqrt');
34. % Calculating the signal with respect to the filter.
35. transmitted_signal = conv(transmitter_filter,upsampled_signal);
36. % Printing out transmitted signal's length.
37. fprintf('Transmitted signal length: %d\n',length(transmitted_signal));
38.
39. % -----------------------------------------------------------[Filter in channel]
40. % Impulse response of the non-ideal channel.
41. h = [0.01 0.04 -0.05 0.06 -0.22 -0.5 0.72 0.36 0 0.21 0.04 0.08 0.02];
42. % Upsampling the non-ideal channel's response.
43. upsampled_h = upsample(h,up_rate);
44. % Calculating the length of the upsampled response.
45. u_h_length = length(upsampled_h);
46. % Removing the last three zeros.
47. upsampled_h = upsampled_h(1:(u_h_length-(up_rate-1)));
48. % Specifying the type of the channel.
49. % If the channel is ideal, the transmitted signal passes.
```

```
50.  if (channel == 'i'), channeled_signal = transmitted_signal;
51.  % If it's non-ideal, the convolution of impulse and signal passes.
52.  elseif (channel == 'n')
53.      channeled_signal = conv(upsampled_h,transmitted_signal);
54.  end
55.  % Printing out channeled signal's length.
56.  fprintf('Channeled signal length: %d\n',length(channeled_signal));
57.
58.  % ------------------------------------------[Calculating noise in channel]
59.  % Calculating the length of the channeled signal.
60.  c_s_length = length(channeled_signal);
61.  % Calculating the average power of the signal.
62.  Ps = sum(abs(channeled_signal).^2)/c_s_length;
63.  % Calculating the average power of the noise.
64.  Pn = Ps/(10^(SNR/10));
65.  % Noise's average power is equal to squared noise standard deviation.
66.  sigma = sqrt(Pn);
67.  % Creating noise randomly.
68.  if(isreal(channeled_signal) == 1) % For real numbers.
69.      random_noise = randn(c_s_length,1);
70.  else % For complex numbers.
71.      random_noise = (randn(c_s_length,1) + 1j*randn(c_s_length,1))/sqrt(2);
72.  end
73.  % Calculating the final channel noise.
74.  channel_noise = sigma * random_noise;
75.  % Adding noise to the signal.
76.  signal_with_noise = channeled_signal + channel_noise;
77.  % Printing out signal's length with noise.
78.  fprintf('Signal with noise length: %d\n',length(signal_with_noise));
79.
80.  % ----------------------------------------------------[Filter in receiver]
81.  % Filter in receiver is the same as in transmitter.
82.  receiver_filter = transmitter_filter;
83.  % Calculating the incoming signal with respect to the filter.
84.  received_signal = conv(receiver_filter,signal_with_noise);
85.  % Printing out received signal's length.
86.  fprintf('Received signal length: %d\n',length(received_signal));
87.
88.  % ----------------------------------------------------[Downsampling signal]
89.  % Calculating the filter's taps.
90.  N = length(transmitter_filter);
91.  % Calculating non-ideal channel's taps.
92.  N_n = length(upsampled_h);
93.  % Calculating the delay of the signal.
94.  if (channel == 'i') % For ideal channel.
95.      delay = 2*(((N-1)*Ts)/2);
96.  elseif (channel == 'n') % For non-ideal channel.
97.      delay = 2*(((N-1)*Ts)/2) + (((N_n-1)*Ts)/2);
98.  end
99.  fprintf('Delay: %d\n',delay);
100. % Removing delay from our signal.
101. temp_signal = received_signal(delay+1:(end-delay));
102. % Downsampling received signal with respect to delay.
103. downsampled_signal = downsample(temp_signal,up_rate);
104. % Printing out downsampled signal's length.
105. fprintf('Downsampled signal length: %d\n',length(downsampled_signal));
106.
107. % ------------------------------------------------------[Decision process]
108. % Calculating the length of the downsampled signal.
109. d_s_length = length(downsampled_signal);
110. % Initializing the output signal vector.
111. output = zeros(d_s_length,1);
112. % Calculating the length of the alphabet.
113. s_length = length(s);
```

```
% Initializing th D vector of distance metrics.
D = zeros(s_length,1);
% Finding the minimum distance of each signal's value from the symbols.
for i=1:d_s_length
    % Calculating the squared distance.
    for j=1:s_length
        D(j) = (downsampled_signal(i) - s(j))^2;
    end
    % Sorting D vector with ascending order, minimum distance is on top.
    [D,I] = sort(D,'ascend');
    % Returning the symbol with the smallest Euclidean distance.
    output(i) = s(I(1));
end

end
```

Όνομα αρχείου: *script_2_b.m*

```
function [BER_i,BER_n] = script_2_b(SNR) % --------------------------------
% INPUT ----------------------------------------------------------------------
%   SNR: desired signal-to-noise ratios
% OUTPUT ---------------------------------------------------------------------
%   BER_i: bit error rate for the ideal channel
%   BER_n: bit error rate for the non-ideal channel
% ----------------------------------------------------------------------------

% Printing intro.
fprintf('|||||||||||| 2-PAM Modulation ||||||||||||\n');
fprintf('----------------------------------------\n');

% Calculating number of different SNR values.
SNR_length = length(SNR);
% Specifying 2-PAM's alphabet.
s = [-1 1];
% Number of data to be used.
data_number = 10^5;
% Generating a random input signal of symbols with equal probability.
input_signal = randsrc(data_number,1,s);
% Calculating the length of the generated signal.
i_s_length = length(input_signal);

% Printing out input signal's length for testing.
fprintf('Input signal length: %d\n',i_s_length);
fprintf('----------------------------------------\n');

% Initializing vectors to save the results.
output_signal_i = zeros(i_s_length,SNR_length);
output_signal_n = zeros(i_s_length,SNR_length);
% Running the system for 2-PAM modulation.
for j=1:SNR_length
    fprintf('|||> SNR|db = %d\n',SNR(j));
    fprintf('---- [i]\n');
    output_signal_i(:,j) = script_2_a(input_signal,s,'i',SNR(j));
    fprintf('---- [n]\n');
    output_signal_n(:,j) = script_2_a(input_signal,s,'n',SNR(j));
    fprintf('\n');
end
fprintf('----------------------------------------\n');

% Converting signals to binary represantation, in order to calculate BER.
% Initializing a vector to save the results.
```

```
44.   binary_input = zeros(i_s_length,1);
45.   % We need 1 bit per symbol, and so we assign '0' to -1 and '1' to 1.
46.   for i=1:i_s_length
47.       % Converting only the negative values of input signal to zero.
48.       if(input_signal(i) == -1)
49.           binary_input(i) = 0;
50.       elseif(input_signal(i) == 1)
51.           binary_input(i) = 1;
52.       end
53.   end
54.
55.   % Initializing vectors to save the results.
56.   binary_output_i = zeros(i_s_length,SNR_length);
57.   binary_output_n = zeros(i_s_length,SNR_length);
58.   % Converting the 2 output signals for each case respectively.
59.   for j=1:SNR_length
60.       for i=1:i_s_length
61.           % Converting output signal for ideal channel.
62.           if(output_signal_i(i,j) == -1)
63.               binary_output_i(i,j) = 0;
64.           elseif(output_signal_i(i,j) == 1)
65.               binary_output_i(i,j) = 1;
66.           end
67.           % Converting output signal for non-ideal channel.
68.           if(output_signal_n(i,j) == -1)
69.               binary_output_n(i,j) = 0;
70.           elseif(output_signal_n(i,j) == 1)
71.               binary_output_n(i,j) = 1;
72.           end
73.       end
74.   end
75.
76.   % Calculating the bit error ratios (BER) for each case of SNR value.
77.   n_i = zeros(SNR_length,1); BER_i = zeros(SNR_length,1);
78.   n_n = zeros(SNR_length,1); BER_n = zeros(SNR_length,1);
79.   for j=1:SNR_length
80.       [n_i(j),BER_i(j)] = biterr(binary_output_i(:,j),binary_input);
81.       [n_n(j),BER_n(j)] = biterr(binary_output_n(:,j),binary_input);
82.   end
83.
84.   % Plotting the results as BER/SNR on one figure.
85.   figure;
86.   semilogy(SNR,BER_i,'-g','LineWidth',2); hold on;
87.   semilogy(SNR,BER_n,':g','LineWidth',2); hold off;
88.   legend('ideal','non-ideal','Location','Best');
89.   xlabel('Signal-to-Noise Ratio (SNR)');
90.   ylabel('Bit Error Ratio (BER)');
91.   title('2-PAM Modulation');
92.
93.   end
```

Όνομα αρχείου: *script_2_c.m*

```
1.    function [BER_i,BER_n] = script_2_c(SNR) % --------------------------------
2.    % INPUT ----------------------------------------------------------------
3.    %   SNR: desired signal-to-noise ratios
4.    % OUTPUT ---------------------------------------------------------------
5.    %   BER_i: bit error rate for the ideal channel
6.    %   BER_n: bit error rate for the non-ideal channel
7.    % ----------------------------------------------------------------------
8.
```

```
9.    % Printing intro.
10.   fprintf('|||||||||||| 4-PAM Modulation ||||||||||||\n');
11.   fprintf('----------------------------------------\n');
12.
13.   % Calculating number of different SNR values.
14.   SNR_length = length(SNR);
15.   % Specifying 4-PAM's alphabet.
16.   s = [-3/sqrt(5) -1/sqrt(5) 1/sqrt(5) 3/sqrt(5)];
17.   % Number of data to be used.
18.   data_number = 10^5;
19.   % Generating a random input signal of symbols with equal probability.
20.   input_signal = randsrc(data_number,1,s);
21.   % Calculating the length of the generated signal.
22.   i_s_length = length(input_signal);
23.
24.   % Printing out input signal's length for testing.
25.   fprintf('Input signal length: %d\n',i_s_length);
26.   fprintf('----------------------------------------\n');
27.
28.   % Initializing vectors to save the results.
29.   output_signal_i = zeros(i_s_length,SNR_length);
30.   output_signal_n = zeros(i_s_length,SNR_length);
31.   % Running the system for 4-PAM modulation.
32.   for j=1:SNR_length
33.       fprintf('|||> SNR|db = %d\n',SNR(j));
34.       fprintf('---- [i]\n');
35.       output_signal_i(:,j) = script_2_a(input_signal,s,'i',SNR(j));
36.       fprintf('---- [n]\n');
37.       output_signal_n(:,j) = script_2_a(input_signal,s,'n',SNR(j));
38.       fprintf('\n');
39.   end
40.   fprintf('----------------------------------------\n');
41.
42.   % Converting signals to binary represantation, in order to calculate BER.
43.   % Initializing a matrix and a vector to save the results.
44.   binary_input = zeros(i_s_length,2);
45.   % We need 2 bits per symbol, and so we assign the symbols as followed:
46.   % '00' to -3/sqrt(5), '01' to -1/sqrt(5)
47.   % '10' to 1/sqrt(5), '11' to 3/sqrt(5)
48.   for i=1:i_s_length
49.       if(input_signal(i) == -3/sqrt(5))
50.           binary_input(i,1) = 0;
51.           binary_input(i,2) = 0;
52.       elseif(input_signal(i) == -1/sqrt(5))
53.           binary_input(i,1) = 0;
54.           binary_input(i,2) = 1;
55.       elseif(input_signal(i) == 1/sqrt(5))
56.           binary_input(i,1) = 1;
57.           binary_input(i,2) = 0;
58.       elseif(input_signal(i) == 3/sqrt(5))
59.           binary_input(i,1) = 1;
60.           binary_input(i,2) = 1;
61.       end
62.   end
63.
64.   % Initializing vectors to save the results.
65.   binary_output_i = zeros(i_s_length,2,SNR_length);
66.   binary_output_n = zeros(i_s_length,2,SNR_length);
67.   % Converting the 2 output signals for each case respectively.
68.   for j=1:SNR_length
69.       for i=1:i_s_length
70.           % Converting output signal for ideal channel.
71.           if(output_signal_i(i,j) == -3/sqrt(5))
72.               binary_output_i(i,1,j) = 0;
```

```
            binary_output_i(i,2,j) = 0;
        elseif(output_signal_i(i,j) == -1/sqrt(5))
            binary_output_i(i,1,j) = 0;
            binary_output_i(i,2,j) = 1;
        elseif(output_signal_i(i,j) == 1/sqrt(5))
            binary_output_i(i,1,j) = 1;
            binary_output_i(i,2,j) = 0;
        elseif(output_signal_i(i,j) == 3/sqrt(5))
            binary_output_i(i,1,j) = 1;
            binary_output_i(i,2,j) = 1;
        end
        % Converting output signal for non-ideal channel.
        if(output_signal_n(i,j) == -3/sqrt(5))
            binary_output_n(i,1,j) = 0;
            binary_output_n(i,2,j) = 0;
        elseif(output_signal_n(i,j) == -1/sqrt(5))
            binary_output_n(i,1,j) = 0;
            binary_output_n(i,2,j) = 1;
        elseif(output_signal_n(i,j) == 1/sqrt(5))
            binary_output_n(i,1,j) = 1;
            binary_output_n(i,2,j) = 0;
        elseif(output_signal_n(i,j) == 3/sqrt(5))
            binary_output_n(i,1,j) = 1;
            binary_output_n(i,2,j) = 1;
        end
    end
end

% Calculating the bit error ratios (BER) for each case of SNR value.
n_i = zeros(SNR_length,1); BER_i = zeros(SNR_length,1);
n_n = zeros(SNR_length,1); BER_n = zeros(SNR_length,1);
for j=1:SNR_length
    [n_i(j),BER_i(j)] = biterr(binary_output_i(:,:,j),binary_input);
    [n_n(j),BER_n(j)] = biterr(binary_output_n(:,:,j),binary_input);
end

% Plotting the results as BER/SNR on one figure.
figure;
semilogy(SNR,BER_i,'-b','LineWidth',2); hold on;
semilogy(SNR,BER_n,':b','LineWidth',2); hold off;
legend('ideal','non-ideal','Location','Best');
xlabel('Signal-to-Noise Ratio (SNR)');
ylabel('Bit Error Ratio (BER)');
title('4-PAM Modulation');

end
```

Όνομα αρχείου: *script_2_d.m*

```
function [BER_i,BER_n] = script_2_d(SNR) % --------------------------------
% INPUT ----------------------------------------------------------------------
%   SNR: desired signal-to-noise ratios
% OUTPUT ---------------------------------------------------------------------
%   BER_i: bit error rate for the ideal channel
%   BER_n: bit error rate for the non-ideal channel
% ----------------------------------------------------------------------------

% Printing intro.
fprintf('||||||||||| 4-QAM Modulation |||||||||||\n');
fprintf('---------------------------------------\n');

```

```
% Calculating number of different SNR values.
SNR_length = length(SNR);
% Specifying 4-QAM's alphabet.
s = [(1+1j)/sqrt(2) (-1+1j)/sqrt(2) (-1-1j)/sqrt(2) (1-1j)/sqrt(2)];
% Number of data to be used.
data_number = 10^5;
% Generating a random input signal of symbols with equal probability.
input_signal = randsrc(data_number,1,s);
% Calculating the length of the generated signal.
i_s_length = length(input_signal);

% Printing out input signal's length for testing.
fprintf('Input signal length: %d\n',i_s_length);
fprintf('----------------------------------------\n');

% Initializing vectors to save the results.
output_signal_i = zeros(i_s_length,SNR_length);
output_signal_n = zeros(i_s_length,SNR_length);
% Running the system for 4-QAM modulation.
for j=1:SNR_length
    fprintf('|||> SNR|db = %d\n',SNR(j));
    fprintf('---- [i]\n');
    output_signal_i(:,j) = script_2_a(input_signal,s,'i',SNR(j));
    fprintf('---- [n]\n');
    output_signal_n(:,j) = script_2_a(input_signal,s,'n',SNR(j));
    fprintf('\n');
end
fprintf('----------------------------------------\n');

% Converting signals to binary represantation, in order to calculate BER.
% Initializing a matrix and a vector to save the results.
binary_input = zeros(i_s_length,2);
% We need 2 bits per symbol, and so we assign the symbols as followed:
% '11' to (1+1j)/sqrt(2), '01' to (-1+1j)/sqrt(2)
% '10' to (1-1j)/sqrt(2), '00' to (-1-1j)/sqrt(2)
for i=1:i_s_length
    if(input_signal(i) == (1+1j)/sqrt(2))
        binary_input(i,1) = 1;
        binary_input(i,2) = 1;
    elseif(input_signal(i) == (-1+1j)/sqrt(2))
        binary_input(i,1) = 0;
        binary_input(i,2) = 1;
    elseif(input_signal(i) == (1-1j)/sqrt(2))
        binary_input(i,1) = 1;
        binary_input(i,2) = 0;
    elseif(input_signal(i) == (-1-1j)/sqrt(2))
        binary_input(i,1) = 0;
        binary_input(i,2) = 0;
    end
end

% Initializing vectors to save the results.
binary_output_i = zeros(i_s_length,2,SNR_length);
binary_output_n = zeros(i_s_length,2,SNR_length);
% Converting the 2 output signals for each case respectively.
for j=1:SNR_length
    for i=1:i_s_length
        % Converting output signal for ideal channel.
        if(output_signal_i(i,j) == (1+1j)/sqrt(2))
            binary_output_i(i,1,j) = 1;
            binary_output_i(i,2,j) = 1;
        elseif(output_signal_i(i,j) == (-1+1j)/sqrt(2))
            binary_output_i(i,1,j) = 0;
            binary_output_i(i,2,j) = 1;
```

```
            elseif(output_signal_i(i,j) == (1-1j)/sqrt(2))
                binary_output_i(i,1,j) = 1;
                binary_output_i(i,2,j) = 0;
            elseif(output_signal_i(i,j) == (-1-1j)/sqrt(2))
                binary_output_i(i,1,j) = 0;
                binary_output_i(i,2,j) = 0;
            end
            % Converting output signal for non-ideal channel.
            if(output_signal_n(i,j) == (1+1j)/sqrt(2))
                binary_output_n(i,1,j) = 1;
                binary_output_n(i,2,j) = 1;
            elseif(output_signal_n(i,j) == (-1+1j)/sqrt(2))
                binary_output_n(i,1,j) = 0;
                binary_output_n(i,2,j) = 1;
            elseif(output_signal_n(i,j) == (1-1j)/sqrt(2))
                binary_output_n(i,1,j) = 1;
                binary_output_n(i,2,j) = 0;
            elseif(output_signal_n(i,j) == (-1-1j)/sqrt(2))
                binary_output_n(i,1,j) = 0;
                binary_output_n(i,2,j) = 0;
            end
        end
    end

    % Calculating the bit error ratios (BER) for each case of SNR value.
    n_i = zeros(SNR_length,1); BER_i = zeros(SNR_length,1);
    n_n = zeros(SNR_length,1); BER_n = zeros(SNR_length,1);
    for j=1:SNR_length
        [n_i(j),BER_i(j)] = biterr(binary_output_i(:,:,j),binary_input);
        [n_n(j),BER_n(j)] = biterr(binary_output_n(:,:,j),binary_input);
    end

    % Plotting the results as BER/SNR on one figure.
    figure;
    semilogy(SNR,BER_i,'-m','LineWidth',2); hold on;
    semilogy(SNR,BER_n,':m','LineWidth',2); hold off;
    legend('ideal','non-ideal','Location','Best');
    xlabel('Signal-to-Noise Ratio (SNR)');
    ylabel('Bit Error Ratio (BER)');
    title('4-QAM Modulation');

    end
```

Όνομα αρχείου: *script_2_e.m*

```
% Specifying SNR values.
SNR = [0:2:30]; %#ok<NBRAK>

% Running our telecommunication system for each modulation.
[BER_i_2PAM,BER_n_2PAM] = script_2_b(SNR); % For 2-PAM.
[BER_i_4PAM,BER_n_4PAM] = script_2_c(SNR); % For 4-PAM.
[BER_i_4QAM,BER_n_4QAM] = script_2_d(SNR); % For 4-QAM.

% Plotting the results as BER/SNR on one figure.
figure;
semilogy(SNR,BER_i_2PAM,'-g','LineWidth',2); hold on;
semilogy(SNR,BER_n_2PAM,':g','LineWidth',2); hold on;
semilogy(SNR,BER_i_4PAM,'-b','LineWidth',2); hold on;
semilogy(SNR,BER_n_4PAM,':b','LineWidth',2); hold on;
semilogy(SNR,BER_i_4QAM,'-m','LineWidth',2); hold on;
semilogy(SNR,BER_n_4QAM,':m','LineWidth',2); hold off;
```

```
17.  legend('2-PAM (i)','2-PAM (n)',...
18.         '4-PAM (i)','4-PAM (n)',...
19.         '4-QAM (i)','4-QAM (n)',...
20.         'Location','Best');
21.  xlabel('Signal-to-Noise Ratio (SNR)');
22.  ylabel('Bit Error Ratio (BER)');
23.  title('Comparative Figure For All Modulations');
24.
25.  % Creating a file to save the results.
26.  file_id = fopen('results_2_e.txt','w+');
27.  % Saving results for 2-PAM modulation.
28.  fprintf(file_id,'-----------------------------------------------\n');
29.  fprintf(file_id,'||||||||||||||||||||||||||||||||||| 2-PAM |||\n');
30.  fprintf(file_id,'BER (ideal channel) --------------------------\n');
31.  for i=1:length(BER_i_2PAM)
32.      fprintf(file_id,'   [%d] %f\n',SNR(i),BER_i_2PAM(i));
33.  end
34.  fprintf(file_id,'BER (non-ideal channel) ----------------------\n');
35.  for i=1:length(BER_n_2PAM)
36.      fprintf(file_id,'   [%d] %f\n',SNR(i),BER_n_2PAM(i));
37.  end
38.  % Saving results for 4-PAM modulation.
39.  fprintf(file_id,'-----------------------------------------------\n');
40.  fprintf(file_id,'||||||||||||||||||||||||||||||||||| 4-PAM |||\n');
41.  fprintf(file_id,'BER (ideal channel) --------------------------\n');
42.  for i=1:length(BER_i_4PAM)
43.      fprintf(file_id,'   [%d] %f\n',SNR(i),BER_i_4PAM(i));
44.  end
45.  fprintf(file_id,'BER (non-ideal channel) ----------------------\n');
46.  for i=1:length(BER_n_4PAM)
47.      fprintf(file_id,'   [%d] %f\n',SNR(i),BER_n_4PAM(i));
48.  end
49.  % Saving results for 4-QAM modulation.
50.  fprintf(file_id,'-----------------------------------------------\n');
51.  fprintf(file_id,'||||||||||||||||||||||||||||||||||| 4-QAM |||\n');
52.  fprintf(file_id,'BER (ideal channel) --------------------------\n');
53.  for i=1:length(BER_i_4QAM)
54.      fprintf(file_id,'   [%d] %f\n',SNR(i),BER_i_4QAM(i));
55.  end
56.  fprintf(file_id,'BER (non-ideal channel) ----------------------\n');
57.  for i=1:length(BER_n_4QAM)
58.      fprintf(file_id,'   [%d] %f\n',SNR(i),BER_n_4QAM(i));
59.  end
60.
61.  % Closing text file.
62.  fclose(file_id);
```