Τηλεπικοινωνιακά Συστήματα

Διάλεξη 6η

Θωμάς Καμαλάκης

Χαροκόπειο Πανεπιστήμιο Αθηνών

Οκτώβριος 2020

Περιεχόμενα

Ο θόρυβος στις επικοινωνίες

- Υπάρχουν πολλές πηγές θορύβου.
- Ο *ηλεκτρονικός θόρυβος* για παράδειγμα οφείλεται στην τυχαία κίνηση των ηλεκτρονίων η οποία καταγράφεται ως ένα τυχαίο ηλεκτρικό ρεύμα.
- Στις επικοινωνίες θεωρούμε (συνήθως) ότι ο θόρυβος *προστίθεται* στο σήμα.
- Αν $x(t)$ είναι το αρχικό σήμα στην είσοδο ενός συστήματος με θόρυβο, τότε το σήμα στην έξοδο $y(t)$ θα είναι:

$$y(t) = x(t) + n(t)$$

- όπου $n(t)$ ο θόρυβος.
- Αν το σύστημα έχει κρουστική απόκριση $h(t)$ θα έχουμε:

$$y(t) = \int_{-\infty}^{+\infty} h(t-\tau)x(\tau)\mathrm{d}\tau + n(t)$$

Χαρακτηριστικά θορύβου

- Συνήθως ο θόρυβος έχει μέση τιμή μηδέν, $\mathbb{E}\{n(t)\} = 0$.
- Η ισχύς του θορύβου καθορίζεται από το $\sigma^2 = \mathbb{E}\{n^2(t)\}$
- Η συνάρτηση αυτοσυσχέτισης δίνεται από την σχέση:

$$R_{nn}(t, t+\tau) = \mathbb{E}\{n(t)n(t+\tau)\}$$

- Συνήθως θεωρούμε ότι ο θόρυβος είναι *στατικός* οπότε:

$$R_{nn}(t, t+\tau) = R_{nn}(\tau)$$

- Η φασματική πυκνότητα του θορύβου δίνεται από την:

$$S_n(f) = \int_{-\infty}^{\infty} R_{nn}(\tau) \mathrm{e}^{-j2\pi f\tau} \mathrm{d}\tau$$

Μερικές παραδοχές

- Εμείς θα θεωρήσουμε ότι ο θόρυβος είναι: *λευκός*, *προσθετικός* και ακολουθεί την *κανονική* (Gaussian) κατανομή.
- Additive, White, Gaussian Noise - AWGN
- *Προσθετικός*, δηλαδή προστίθεται στο σήμα
- *Λευκός*, δηλαδή η φασματική του πυκνότητα είναι σταθερή, π.χ.

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

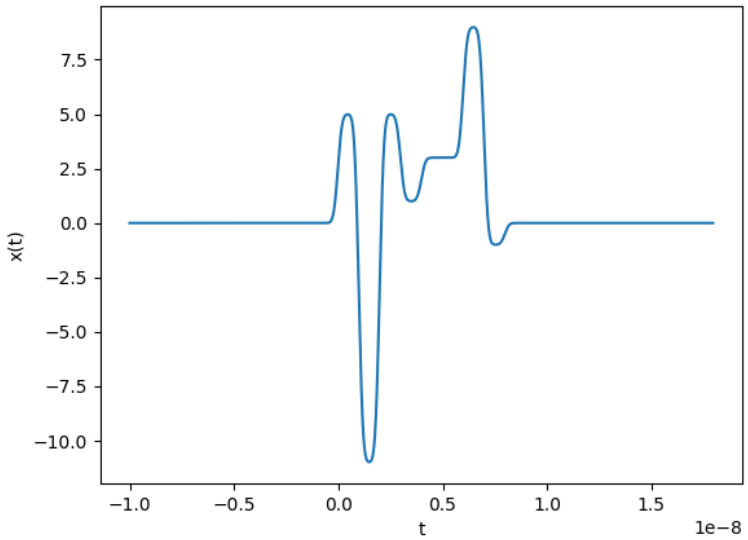
- Ανάλογα με το εύρος ζώνης του συστήματος μας B, ο θόρυβος θα έχει ισχύ:

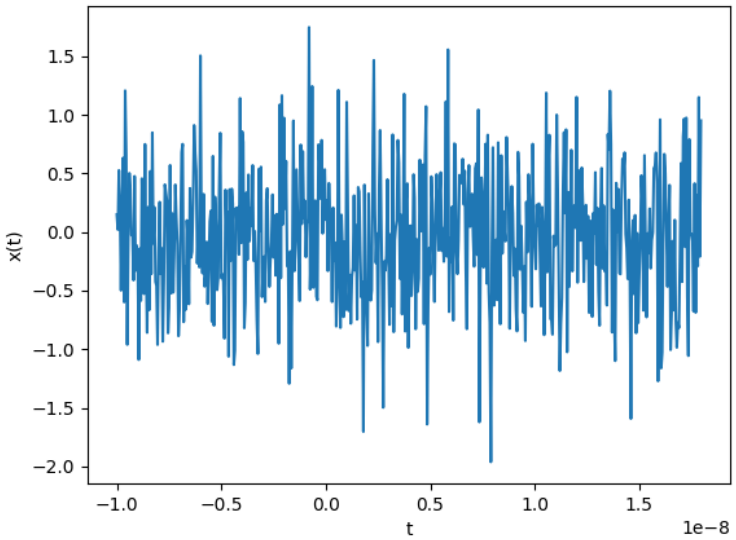
$$\sigma^2 = R_{nn}(0) = \mathbb{E}\left\{n^2(t)\right\} = \int_{-B}^{+B} \frac{N_0}{2} \mathrm{d}f = N_0 B \tag{1}$$

Παράδειγμα

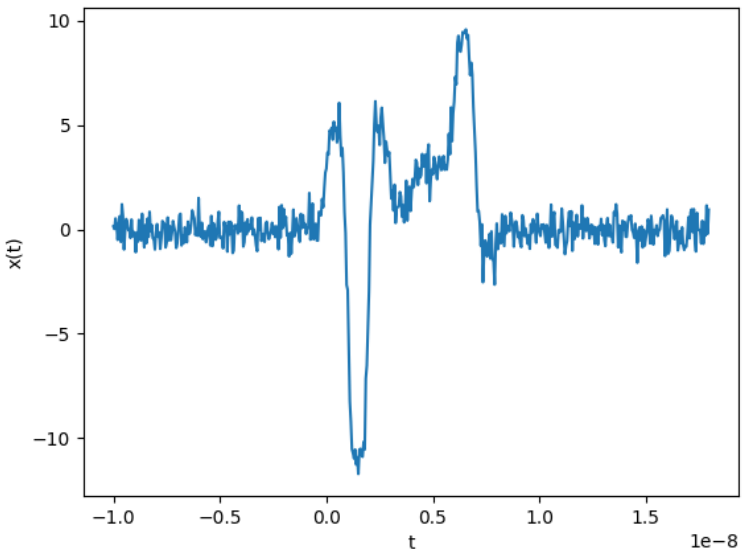
```
import matplotlib.pyplot as plt
import commlib as cl
import numpy as np

# Parameters
TS = 1e—9
samples_per_symbol = 20
tinitial = 0
tguard = 10 * TS
f0 = 1 / TS
N0 = 1e—10
B = 3e9

# system transfer function
H = lambda f : np.exp( —f**2.0 / f0 ** 2.0 / 2)

# Signal constellation
c = cl.pam_constellation(16, title = '16—PAM')

# Plot PAM constellation
plt.close('all')
c.plot()
c.plot_map()

# set bits to be transmitted
bits = cl.random_bits(32)

# build input waveform and plot
x = cl.digital_signal(TS = TS, samples_per_symbol
        = samples_per_symbol,
                        tinitial = tinitial, tguard
        = tguard, constellation = c)

x.modulate_from_bits( bits, constellation = c )
x.plot()

# create channel system and apply
s = cl.system(input_signal = x, transfer_function
        = H)
s.apply()

# get output signal and plot
y = s.get_output()
y.plot()

n = cl.white_noise(N0 = N0, B = B, t = y.t)
n.plot()

z = y + n
z.plot()
```

Παράδειγμα


7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
x(t)
−1.0
−0.5
0.0
0.5
1.0
1.5
t
1e−8


1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
x(t)
−1.0
−0.5
0.0
0.5
1.0
1.5
t
1e−8

Παράδειγμα



Το σήμα PAM στον πομπό

- Ας υποθέσουμε ότι η χρησιμοποιούμε τετραγωνικούς παλμούς $p(t)$ για να φτιάξουμε το σήμα PAM,

$$p(t) = \begin{cases} \sqrt{P}_{\max} & ,\text{εάν } t \in [0, T_S] \\ 0 & ,\text{διαφορετικά} \end{cases}$$

- Επομένως η κυματομορφή PAM στον πομπό μπορεί να γραφεί:

$$x(t) = \sum_q a_q p(t - qT_S)$$

- Στον πομπό η μέση ισχύς του σήματος θα είναι:

$$P_t = P_{\max}\, \mathbb{E}\left\{a_q^2\right\} = \beta^2 P_{\max} \frac{M^2 - 1}{3}$$

- Ας θεωρήσουμε ότι $P_{\max} = 1$. Ούτως ή άλλως για να ρυθμίσουμε την ισχύ P_t μπορούμε να πειράξουμε το β.

Το σήμα PAM στο δέκτη

- Στον δέκτη ας θεωρήσουμε ότι η κυματομορφή είναι παρόμοια, δηλαδή το κανάλι έχει αρκετά μεγάλο εύρος ζώνης ώστε να μην αλλοιώνει σημαντικά τους παλμούς $p(t)$.
- Αν L είναι οι απώλειες του καναλιού, τότε απουσία θορύβου η ισχύς στο δέκτη θα πρέπει να είναι:

$$P_{\mathrm{r}} = LP_{\mathrm{t}}$$

- Επομένως απουσία του θορύβου, στο δέκτη η κυματομορφή γράφεται:

$$y_0(t) = \sqrt{L}\sum_q a_q p(t - qT_{\mathrm{S}})$$

- Όταν έχουμε θόρυβο, το σήμα στο δέκτη θα είναι:

$$y(t) = \sqrt{L}\sum_q a_q p(t - qT_{\mathrm{S}}) + n(t)$$

Πηλίκο σήμα-προς-θόρυβο

- Η μέση ισχύς του σήματος $y_0(t)$ στο δέκτη είναι

$$P_{\mathrm{r}} = LP_{\mathrm{t}} = L\beta^2 \frac{M^2 - 1}{3}$$

- Η ενέργεια του σήματος θα είναι:

$$\mathcal{E}_{\mathrm{r}} = P_{\mathrm{r}} T_{\mathrm{S}} = L\beta^2 T_{\mathrm{S}} \frac{M^2 - 1}{3}$$

- Ορίζουμε το πηλίκο σήμα-προς-θόρυβο (signal-to-noise) ως το πηλίκο της ενέργειας του σήματος προς την φασματική πυκνότητα ισχύος N_0 του θορύβου:

$$\mathrm{SNR}_{\mathrm{S}} = \frac{\mathcal{E}_{\mathrm{r}}}{N_0} = L\beta^2 T_{\mathrm{S}} \frac{M^2 - 1}{3N_0}$$

Πηλίκο σήμα-προς-θόρυβο

- Το N_0 έχει μονάδες Watt/Hz = Watt $\times$ s = Joule.
- επομένως το πηλίκο SNR_{S} είναι *καθαρός αριθμός*.
- Αντιπροσωπεύει την ενέργεια που φτάνει στο δέκτη προς την φασματική πυκνότητα του θορύβου *ανά σύμβολο*.
- δεδομένου ότι κάθε σύμβολο αντιστοιχεί σε $\log_2 M$ bits μπορούμε να δούμε ότι η ενέργεια που αντιστοιχεί σε κάθε bit είναι:

$$\mathcal{E}_{\text{rb}} = \frac{\mathcal{E}_r}{\log_2 M}$$

- μπορούμε να ορίσουμε επομένως το πηλίκο σήματος-προς-θόρυβο *ανά bit* ως:

$$\text{SNR}_{\text{b}} = \frac{\mathcal{E}_{\text{rb}}}{N_0} = \frac{\text{SNR}_{\text{S}}}{\log_2 M} = L\beta^2 T_{\text{S}} \frac{M^2 - 1}{3N_0 \log_2 M}$$

Λογαριθμική κλίμακα

- Συνήθως τις απώλειες και το πηλίκο σήμα-προς-θόρυβο τα μετράμε και στην λογαριθμική κλίμακα (σε dB).
- ένας καθαρός αριθμός A μετατρέπεται σε dB ως εξής:

$$A[\text{dB}] = 10 \log_{10} A$$

- αν έχουμε την τιμή στην λογαριθμική κλίμακα τότε αντιστρέφουμε την παραπάνω σχέση για να την υπολογίσουμε στη γραμμική κλίμακα:

$$A = 10^{\frac{A[dB]}{10}}$$

Λογαριθμική κλίμακα

- Μπορούμε να μετρήσουμε και την ισχύ στην λογαριθμική κλίμακα.
- Αν έχουμε την ισχύ P σε mW ή W τότε η ισχύς σε dBm δίνεται από την σχέση:

$$P[\text{dBm}] = 10 \log_{10} \left(\frac{P}{1\text{mW}} \right)$$

- Αντιστρέφοντας έχουμε:

$$P = 1\text{mW} \times 10^{\frac{P[\text{dBm}]}{10}}$$

Πως λειτουργεί ο δέκτης

- Η πιθανότητα να κάνουμε σφάλμα κατά την αποκωδικοποίηση ενός bit ονομάζεται *πιθανότητα σφάλματος bit* ή πηλίκο εσφαλμένων bit (Bit error rate - BER).
- Στην περίπτωση του PAM μπορούμε να υπολογίσουμε προσεγγιστικά πόσο είναι το BER.
- Ας θεωρήσουμε ότι σε κάθε διάρκεια συμβόλου, ο δέκτης υπολογίζει την μέση τιμή του σήματος,

$$y_k = \frac{1}{T_S} \int_{(k-1)T_S}^{kT_S} y(t)\mathrm{d}t$$

- και χρησιμοποιεί το y_k για να καταλάβει το σύμβολο που έχει λάβει.

$$y_k = y_{0k} + n_k = \frac{1}{T_S} \int_{(k-1)T_S}^{kT_S} y_0(t)\mathrm{d}t + \frac{1}{T_S} \int_{(k-1)T_S}^{kT_S} n(t)\mathrm{d}t$$

Συνιστώσες σήματος και θορύβου

- Η συνιστώσα του σήματος γράφεται:

$$y_{0k} = \frac{1}{T_S}\int_{(k-1)T_S}^{kT_S} y_0(t)\mathrm{d}t = \sum_q \frac{a_q\sqrt{L}}{T_S}\int_{(k-1)T_S}^{kT_S} p(t-qT_S)p(t-kT_S)\mathrm{d}t$$

- Δεδόμενου ότι οι παλμοί $p(t)$ δεν επικαλύπτονται, θα έχουμε όταν $q \neq k$:

$$\frac{1}{T_S}\int_{(k-1)T_S}^{kT_S} p(t-qT_S)p(t-kT_S) = 0$$

- Όταν $q = p$ θα έχουμε:

$$\frac{1}{T_S}\int_{(k-1)T_S}^{kT_S} p(t-qT_S)p(t-kT_S)\mathrm{d}t = \frac{1}{T_S}\int_{(k-1)T_S}^{kT_S} p^2(t-kT_S)\mathrm{d}t = P_{\max} = 1$$

Συνιστώσες σήματος και θορύβου

- Επομένως η συνιστώσα του σήματος θα είναι $y_{0k} = a_k\sqrt{L}$.
- Για τον θόρυβο θα έχουμε:

$$\mathbb{E}\{n_k\} = \frac{1}{T_S}\int_{(k-1)TS}^{kTS} \mathbb{E}\{n(t)\}\,\mathrm{d}t = 0$$

$$\sigma^2 = \mathbb{E}\{n_k^2\} = \frac{1}{T_S^2}\,\mathbb{E}\left\{\left(\int_{(k-1)T_S}^{kT_S} n(t)\mathrm{d}t\right)^2\right\} =$$

$$\frac{1}{T_S^2}\int_{(k-1)T_S}^{kT_S}\int_{(k-1)T_S}^{kT_S} \mathbb{E}\{n(t)n(t')\}\,\mathrm{d}t\mathrm{d}t' = \frac{1}{T_S^2}\int_{(k-1)T_S}^{kT_S}\int_{(k-1)T_S}^{kT_S} R_{nn}(t,t')\mathrm{d}t\mathrm{d}t'$$

- Για τον λευκό θόρυβο έχουμε $S_n(f) = N_0/2$ οπότε $R_{nn}(t,t') = \frac{N_0}{2}\delta(t-t')$

Συνιστώσες σήματος και θορύβου

- Ας θυμηθούμε μία βασική ιδιότητα της συνάρτησης δ:

$$\int_{-\infty}^{+\infty} \delta(t)\mathrm{d}t = 1$$

- με μία αλλαγή μεταβλητής εύκολα δείχνουμε ότι:

$$\int_{-\infty}^{+\infty} \delta(t-t')\mathrm{d}t = 1$$

- Αν πάρουμε οποιαδήποτε a και b τέτοια ώστε $a < t' < b$ θα έχουμε:

$$1 = \int_{-\infty}^{+\infty} \delta(t-t')\mathrm{d}t = \int_{-\infty}^{a} \delta(t-t')\mathrm{d}t + \int_{a}^{b} \delta(t-t')\mathrm{d}t + \int_{b}^{+\infty} \delta(t-t')\mathrm{d}t = \int_{a}^{b} \delta(t-t')\mathrm{d}t$$

Συνιστώσες σήματος και θορύβου

- Στην προηγούμενη σχέση αγνοήσαμε τα $\int_{-\infty}^{a}$ και $\int_{b}^{+\infty}$ επειδή η συνάρτηση $\delta(t-t')$ εκεί ισούται με μηδέν.
- Επομένως για οποιαδήποτε $a < t' < b$ θα πρέπει να έχουμε:

$$\int_{a}^{b} \delta(t-t')\mathrm{d}t = 1$$

- Το εσωτερικό ολοκλήρωμα για στον υπολογισμό του $\mathbb{E}\left\{n_k^2\right\}$ γράφεται:

$$\int_{(k-1)T_\mathrm{S}}^{kT_\mathrm{S}} R_{nn}(t,t')\mathrm{d}t = \frac{N_0}{2}\int_{(k-1)T_\mathrm{S}}^{kT_\mathrm{S}} \delta(t-t')\mathrm{d}t = \frac{N_0}{2}$$

Συνιστώσες σήματος και θορύβου

- Οπότε:

$$\sigma^2 = \mathbb{E}\left\{n_k^2\right\} = \frac{1}{T_\mathrm{S}^2}\int_{(k-1)T_\mathrm{S}}^{kT_\mathrm{S}}\int_{(k-1)T_\mathrm{S}}^{kT_\mathrm{S}} R_{nn}(t,t')\mathrm{d}t\mathrm{d}t' = \frac{1}{T_\mathrm{S}^2}\int_{(k-1)T_\mathrm{S}}^{kT_\mathrm{S}} \frac{N_0}{2}\mathrm{d}t' = \frac{N_0}{2T_\mathrm{S}}$$

- Τελικά ο δέκτης λαμβάνει το αρχικό σύμβολο a_k εξασθενημένο κατά $\sqrt{L}$ μαζί με μία τυχαία συνιστώσα n_k
- Αν χωρίσουμε το διάστημα $[(k-1)T_\mathrm{S}, kT_\mathrm{S}]$ σε πολύ μικρά διαστήματα θα έχουμε:

$$n_k = \frac{1}{T_\mathrm{S}}\int_{(k-1)T_\mathrm{S}}^{kT_\mathrm{S}} n(t)\mathrm{d}t \cong \frac{1}{T_\mathrm{S}}\sum_i n(t_i)\Delta t$$

- όπου t_i είναι σημεία μέσα στο διάστημα και $\Delta t = t_{i+1} - t_i$.
- Η παραπάνω σχέση μας λέει ότι το n_k είναι άρθοισμα από Gaussian τυχαίες μεταβλητές $n(t_i)$ οπότε είναι και αυτή Gaussian.

Αποκωδικοποίηση συμβόλων

- Αφού ο δέκτης υπολογίσει το y_k πρέπει να κάνει μία εκτίμηση του συμβόλου a_k από το οποίο προήρθε.

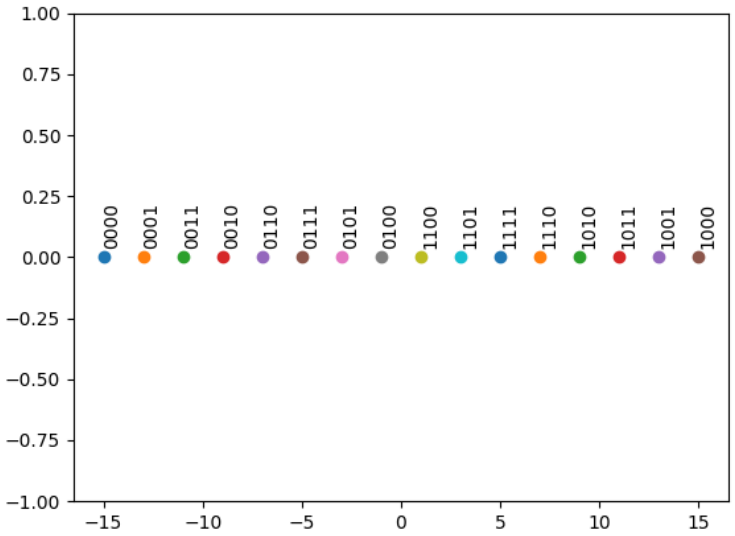
$$y_k = y_{0k} + n_k = a_k\sqrt{L} + n_k$$

- Μία λογική είναι να δούμε σε πιο αρχικό σύμβολο A_m είναι πιο κοντά το $y_k/\sqrt{L}$.
- Υπολογίζουμε δηλαδή όλες τις αποστάσεις d_m,

$$d_m^2 = \left\| y_k - A_m\sqrt{L} \right\|^2$$

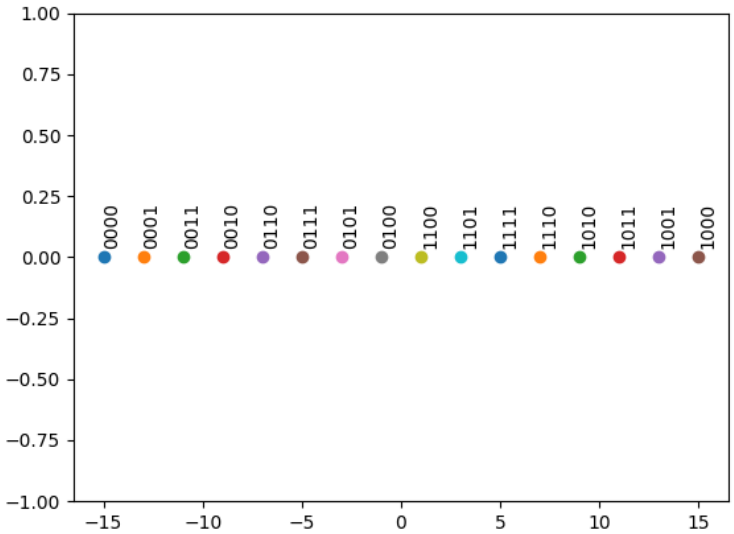
- και θεωρούμε ότι ο πομπός μετέδωσε το σύμβολο A_q με την ελάχιστη απόσταση $d_q = \min\{d_m\}$.

Πότε συμβαίνουν σφάλματα;



- Ας υποθέσουμε ότι έχουμε στείλει το σύμβολο $_p$ στον πομπό
- Για να συμβεί σφάλμα θα πρέπει ο θόρυβος n_k να μας πετάξει πιο κοντά σε ένα άλλο σύμβολο A_q από ότι στο A_p.
- Στο πομπό τα σύμβολα απέχουν μεταξύ τους απόσταση $A_{p+1} - {}_p = 2\beta$.
- Στο δέκτη τα σύμβολα απέχουν μεταξύ τους απόσταση $\sqrt{L}A_{p+1} - \sqrt{L}A_p = 2\beta\sqrt{L}$.

Πότε συμβαίνουν σφάλματα;



- Αν δεν έχουμε εκπέμψει ακριανό σύμβολο ($m \neq 1$ και $m \neq M$) τότε για να μας «πετάξει» ο θόρυβος σε άλλο σύμβολο, θα πρέπει $n_k > \beta\sqrt{L}$ ή $n_k < -\beta\sqrt{L}$.
- Για $m = 1$, θα πρέπει να έχουμε $n_k > \beta\sqrt{L}$ ενώ για $m = M$, θα πρέπει να έχουμε $n_k < -\beta\sqrt{L}$.

Πιθανότητα σφάλματος

- Ας υποθέσουμε ότι τα σύμβολα είναι ισοπίθανα, δηλαδή $\Pr\{a_k = A_m\} = \frac{1}{M}$
- Η πιθανότητα να κάνουμε σφάλμα P_e στην αποκωδικοποίηση *συμβόλου* γράφεται ως εξής:

$$P_\mathrm{e} = \sum_m \Pr\{\mathrm{e}|a_k = A_m\} \Pr\{a_k = A_m\} = \frac{1}{M} \sum_m \Pr\{\mathrm{e}|a_k = A_m\}$$

- Για ακριανά σύμβολα:

$$\Pr\{\mathrm{e}|a_k = A_1\} = \Pr\left\{n_k > \beta\sqrt{L}\right\}$$

$$\Pr\{\mathrm{e}|a_k = A_M\} = \Pr\left\{n_k < -\beta\sqrt{L}\right\}$$

- Για εσωτερικά σύμβολα $2 \leq m \leq M-1$,

$$\Pr\{\mathrm{e}|a_k = A_m\} = \Pr\left\{n_k < -\beta\sqrt{L}\right\} + \Pr\left\{n_k > \beta\sqrt{L}\right\}$$

Πιθανότητα σφάλματος

- Υπολογισμός των πιθανοτήτων που σχετίζονται με τα n_k.

$$\Pr\{n_k > \alpha\} = \int_{\alpha}^{+\infty} f_n(x)\mathrm{d}x = \frac{1}{\sigma\sqrt{2\pi}} \int_{a}^{+\infty} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x = \frac{1}{\sigma\sqrt{2\pi}} \int_{\alpha/\sigma}^{+\infty} \mathrm{e}^{-\frac{y^2}{2}}\mathrm{d}y = Q\left(\frac{a}{\sigma}\right)$$

$$\Pr\{n_k < -\alpha\} = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{-\alpha} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x = \frac{1}{\sigma\sqrt{2\pi}} \int_{\alpha}^{+\infty} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\,\mathrm{d}x = Q\left(\frac{a}{\sigma}\right)$$

Πιθανότητα σφάλματος

- Για εξωτερικά σύμβολα:

$$\Pr\{\mathrm{e}|a_k = A_1\} = \Pr\{\mathrm{e}|a_k = A_M\} = Q\left(\frac{\beta\sqrt{L}}{\sigma}\right)$$

- Για εσωτερικά σύμβολα:

$$\Pr\{\mathrm{e}|a_k = A_m\} = 2Q\left(\frac{\beta\sqrt{L}}{\sigma}\right)$$

- Η μέση πιθανότητα σφάλματος *συμβόλου* είναι:

$$P_\mathrm{e} = \frac{1}{M}\sum_m \Pr\{\mathrm{e}|a_k = A_m\} = 2\frac{M-1}{M}Q\left(\frac{\beta\sqrt{L}}{\sigma}\right) = 2\frac{M-1}{M}Q\left(\beta\sqrt{\frac{2LT_\mathrm{S}}{N_0}}\right)$$

Πιθανότητα σφάλματος

- Δεδομένου ότι:

$$\mathrm{SNR_S} = L\beta^2 T_\mathrm{S} \frac{M^2-1}{3N_0} \Rightarrow \frac{2L\beta^2 T_\mathrm{S}}{N_0} = \frac{6}{M^2-1}\mathrm{SNR_S}$$

- Επομένως μπορούμε να εκφράσουμε την πιθανότητα σφάλματος απευθείας βάσει του πηλίκου σήματος-προς-θόρυβο:

$$P_\mathrm{e} = 2\frac{M-1}{M}Q\left(\sqrt{\frac{6\mathrm{SNR_S}}{M^2-1}}\right) = 2\frac{M-1}{M}Q\left(\sqrt{\frac{6\mathrm{SNR_b}\log_2 M}{M^2-1}}\right)$$

Πιθανότητα σφάλματος bit

- Περιμένουμε ότι τα πιο πιθανά σφάλματα μας «πετάνε» σε διπλανό σύμβολο.
- Δεδομένου ότι τα γειτονικά σύμβολα διαφέρουν κατά ένα bit έπεται ότι σχεδόν κάθε φορά που συμβαίνει ένα σφάλμα συμβόλου συνεπάγεται και ένα σφάλμα bit.
- Αν μεταδίδουμε N σύμβολα τότε μεταδίδουμε και $N \times \log_2 M$ bits.
- Ο αριθμός των εσφαλμένων bit N_b ισούται περίπου με τον αριθμό των εσφαλμένων συμβόλων N_s.
- για μεγάλο N, η πιθανότητα εσφαλμένου bit θα δίνεται από την σχέση:

$$P_\mathrm{b} = \frac{N_\mathrm{b}}{N \log_2 M} \cong \frac{N_\mathrm{s}}{N} \frac{1}{\log_2 M} = \frac{P_\mathrm{e}}{\log_2 M} = 2\frac{M-1}{M \log_2 M} Q\left(\sqrt{\frac{6\mathrm{SNR}_\mathrm{b} \log_2 M}{M^2-1}}\right)$$

Προσομοίωση Monte-Carlo

- Μπορούμε να επαληθεύσουμε τα παραπάνω αποτελέσματα χρησιμοποιώντας προσομοίωση Monte-Carlo.
- Δημιουργούμε τυχαία bits, τα μετατρέπουμε σε σύμβολα, προσθέτουμε θόρυβο, αποκωδικοποιούμε τα σύμβολα και συγκρίνουμε τα τελικά και τα αρχικά σύμβολα καθώς και τα αρχικά και τελικά bits.
- Η λογική της προσομοίωσης Monte-Carlo είναι να κάνει έναν μεγάλο αριθμό τυχαίων πειραμάτων και να υπολογίζει τα διάφορα μεγέθη στατιστικά.
- Γλιτώνουμε τα μαθηματικά αλλά πληρώνουμε σε υπολογιστική ισχύ.

Η κλάση monte_carlo

```
404 class monte_carlo:
405     def __init__(self, max_iterations = 1000, generate = None,
406                        apply = None, measure = None, report_step = 10,
407                        report = False):
408 
409         self.max_iterations = max_iterations
410         self.report_step = report_step
411         self.report = report
412 
413     def execute(self):
414         for i in range(self.max_iterations):
415             if self.report and ( np.mod(i, self.report_step) == 0 ):
416                 print('iteration %d / %d' %(i, self.iterations) )
417             self.generate()
418             self.apply()
419             self.measure()
420 
421             if self.terminate():
422                 self.termination_condition = True
423                 self.iterations_performed = i
424                 return
425 
426         self.termination_condition = False
427         self.iterations_performed = i
```

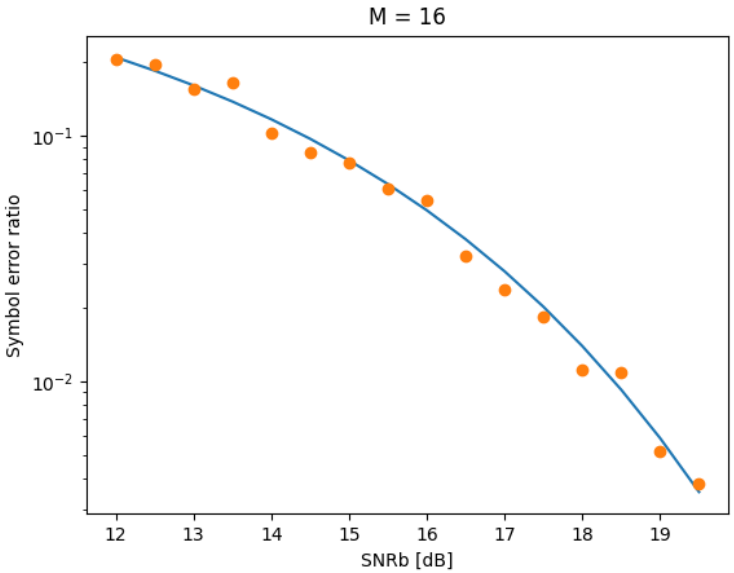
Η κλάση pam_simulation

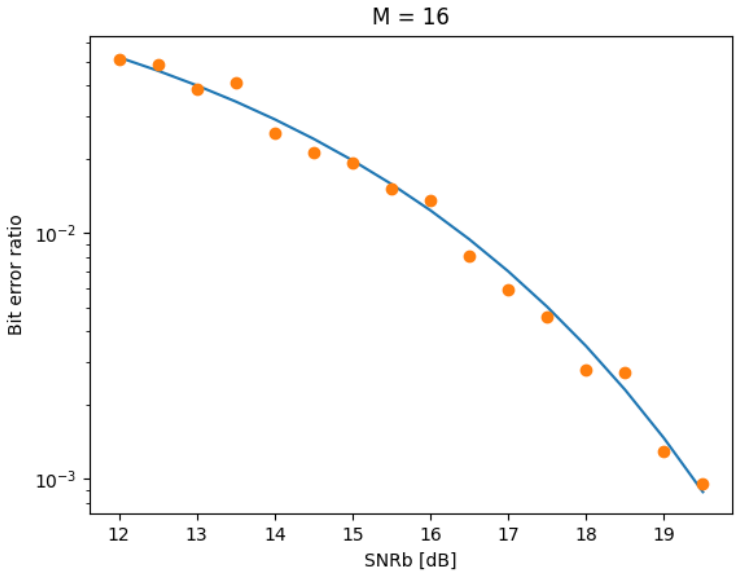
```
class pam_simulation(monte_carlo):

    def __init__(self, max_iterations = 1000, M =
        16, SNRbdB = 10, report_step = 10,
       max_symbol_errors = 100):

        super().__init__(max_iterations =
       max_iterations, report_step = report_step
       )
        self.M = M
        self.m = np.log2(M).astype(int)
        self.SNRbdB = SNRbdB
        self.SNRb = 10 ** (SNRbdB / 10)
        self.constellation = pam_constellation(M)
        self.sigma = np.sqrt( (M ** 2.0 - 1) / (
       6 * self.SNRb *np.log2(M) ) )
        self.symbol_errors = 0
        self.bit_errors = 0
        self.max_symbol_errors =
       max_symbol_errors

    def generate(self):
        bits = random_bits( self.m )
        symbol = self.constellation.
       bits_to_symbols( bits )
        noise = self.sigma * np.random.randn( 1 )

        self.symbol = symbol
        self.input_bits = bits
        self.noise = noise

    def apply(self):
        output = self.symbol + self.noise
        [self.decoded_symbol, self.decoded_bits,
       _ ] = self.constellation.decode( output )

    def measure(self):
        self.bit_errors += np.sum( np.abs(self.
       decoded_bits - self.input_bits) ).astype(
       int)
        self.symbol_errors += int(self.symbol !=
       self.decoded_symbol)

    def terminate(self):
        return self.symbol_errors >= self.
       max_symbol_errors
```

pamber.py

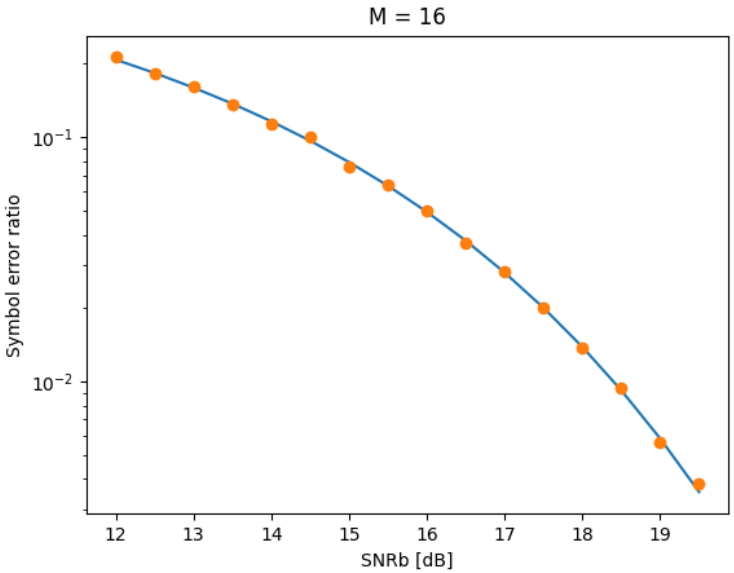
```
from commlib import pam_simulation, Qfunction
import matplotlib.pyplot as plt
import numpy as np

SNRbdBs = np.arange(12, 20, 0.5)
max_iterations = 100000
report_step = 100
max_symbol_errors = 1000
M = 16
Pes = np.zeros( SNRbdBs.size )
Pes2 = np.zeros( SNRbdBs.size )
Peb = np.zeros( SNRbdBs.size )
Peb2= np.zeros( SNRbdBs.size )

for i, SNRbdB in enumerate(SNRbdBs):
    SNRb = 10 ** (SNRbdB / 10)
    q = 6 * SNRb * np.log2(M) / (M ** 2.0 — 1)
    Pes[i] = 2 * (M—1) / M * Qfunction ( np.sqrt(q) )
    Peb[i] = 2 * (M—1) / M * Qfunction ( np.sqrt(q) ) / np.log2(M)

    s = pam_simulation(M = M, SNRbdB = SNRbdB,
                       max_iterations = max_iterations,
                       report_step = report_step,
                       max_symbol_errors = max_symbol_errors)
    s.execute()

    Pes2[i] = s.symbol_errors / s.iterations_performed
    Peb2[i] = s.bit_errors / s.iterations_performed / s.m
```

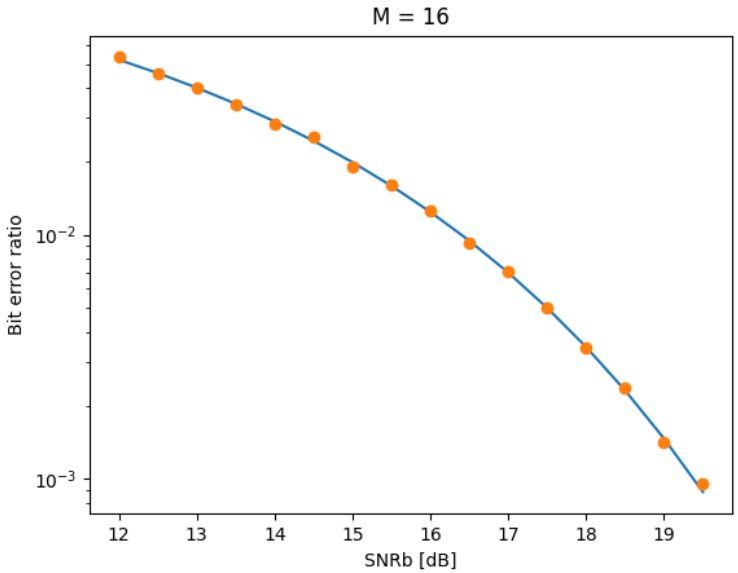
Σύγκριση (`max_symbol_errors=100`)





Σύγκριση (`max_symbol_errors=1000`)





Σύγκριση σχημάτων PAM - pambervsM.py

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

SNRbdBs = np.arange(5, 40, 0.1)
n = np.arange(1,7,1)
Ms = 2 ** n
Ms = Ms.astype(int)

Pes = np.zeros( [SNRbdBs.size, Ms.size] )
Peb = np.zeros( [SNRbdBs.size, Ms.size] )

for i, SNRbdB in enumerate(SNRbdBs):
    for j, M in enumerate(Ms):

        c = pam_constellation( M = M, SNRbdB =
       SNRbdB )
        Pes[i, j] = c.ser()
        Peb[i, j] = c.ber()


plt.close('all')
```

```
for j, M in enumerate(Ms):
    plt.figure(1)
    plt.semilogy( SNRbdBs, Pes[:,j], label = 'M =
        %d' % M)
    plt.figure(2)
    plt.semilogy( SNRbdBs, Peb[:,j], label = 'M =
        %d' % M)

plt.figure(1)
plt.xlabel('SNRb [dB]')
plt.ylabel('SER')
plt.legend()
plt.ylim([1e-12, 1])

plt.figure(2)
plt.xlabel('SNRb [dB]')
plt.ylabel('BER')
plt.legend()
plt.ylim([1e-12, 1])
```

Σύγκριση σχημάτων PAM - pambervsM.py