Guido Carlo Ferrante

Virtual Wiring of UWB Radio Links: Advanced Multi-User Multi-Network Impulsive Communications by Time-Reversal

Master Thesis

Advisor: Prof. M-G. Di Benedetto
Co-advisor: Prof. J. Fiorina

Sapienza University of Rome
School of Engineering
Department of Information Engineering,
Electronics and Telecommunications

October 3, 2011

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Prof. Maria-Gabriella Di Benedetto, who has always encouraged me to pursue this research. She provides me trust and enthusiastic support, sharing generously her ideas and perspectives. I am very grateful for the opportunity she has given me during this year: the Summer School and the visit to Supélec, Paris, last spring have been extremely interesting and fruitful. She is the kindest person I have ever met and I would be very glad to pursue my studies with her.

In Supélec I met my co-advisor, Prof. Jocelyn Fiorina, whose friendship, helpful advice and scientific vision have been, and still are, very precious to me.

Special thanks to a special person: Dr. Luca De Nardis, who made me feel at home from day one in ACTS Lab. I wish you the best!

I have benefited from the group of students, past and present, that have been part of ACTS Lab: Stefano, Lipa and Robbie have traveled with me on this journey.

I would like to thank all those in our Dept. have alleviated the most committed days, especially Nicola, Stefania, Marina and Daniela.

Many thanks to Eli, for her friendship, and Vale, who had to endure my habits of study for a long year. I can't not mention my partner in crime, Lele, with whom I shared a passion for radars: we were an outstanding team!

I am grateful to Prof. G. Rispoli for having brought renewed peace in my life.

And it is a pleasure to thank my parents and my brother for unconditional support and guidance.

Rome, october 3, 2011 G. C. F.

CONTENTS

INTRODUCTION

In this thesis we address the problem of finding, mainly in terms of BER, limitations and strenghts of Time Reversal, a technique found in acoustics [**DRF95**] [**Fin+09**] and applied and further developed in communications and in IR-UWB [**DBG04**] [**WS00**].

TR has spatial and temporal focusing properties (for an overview and other results on MUI and positioning, see for example [**JFB11**] and [**DN+**]). However, the energy collected by a RAKE receiver has an asymptotic value that depends on the peculiar UWB channel characteristics. This limitation is investigated by means of point process theory. The general theory of point process can be found in [**CI80**] [**DVJ08**] [**Str10**]. Its application to the Generalized Saleh-Valenzuela channel [**SV87**] is treated in [**GH06**]. An example of applicability of this approach is shown in [**HG07**]. The model of the channel is the IEEE 802.15.3a [**Foe02**] [**Mol+05**].

As a consequence, TR may be used for outperforming a system with an *all*-RAKE as well as reducing the fingers of the RAKE. We may aptly change the number of taps in TR and the number of fingers in RAKE, employing both partial TR and RAKE, without any loss in performance. The trade-off between complexity and performance has been already treated in [**PFDB09**]. We develop here a similar investigation under a power constraint and propose a simple trade-off for minimizing the average complexity.

Although TR provides several undoubted advantages, further investigations are needed on the robustness of this technique. We study the effects on BER of an error (or perturbation) on the precoder of TR, in analogy to existing studies on RAKE.

The following is the structure of the document:

In the first chapter we investigate the trade-off between the complexity of transmitter and receiver and performance and the intrinsic limitation in the energy that can be collected by the RAKE of IR-UWB systems.

In the second chapter we address the problem of studying how perturbations in TR may affect the performance of a single user as well as of a network.

Two appendices conclude this work, the first summarizing the results and the second showing the main codes.

1

— ∘ —

TRADE-OFF ON THE PERFORMANCE OF TR/RAKE SYSTEMS

Summary

In this chapter we address the problem of finding a trade-off on the complexity of the transmitter and the receiver of a UWB communication system over a multipath fading channel. We focus on a fixed transmit-receive processing scheme, namely a **T**ime-**R**eversal precoder at the transmitter and a (sub)optimal beamforming at the receiver employing a (**S**elective)**A**ll-**RAKE**.

Our contribution

In order to switch the complexity from the receiver to the transmitter, we analyse the performance of the system in terms of SNR and BER varying the number of taps K of the precoder and the number of fingers M of the equaliser. We find the trade-off between these numbers in a generalized Saleh-Valenzuela channel with L paths under a power constraint. We sketch an optimal solution under a specific design criteria, namely the minimization of the total number of taps and fingers.

§1.1 The basic model.

1.1.1 **Modulator.** The UWB communication system we consider (see Figure 1.1 on the following page) adopts an **I**mpulse-**R**adio signaling scheme, meaning that the ultrawide bandwidth characteristic is obtained radiating a (train of) basic pulse waveform $g(t)$ of very short duration, with a compact support in the *chip interval* $[0, T_{\mathrm{C}}]$. We focus on binary signaling schemes, both *orthogonal* and *antipodal*, in particular PPM and PAM respectively. In general, the wireless access in a network with many transmitters and receivers is provided by a time-hopping code (inherently periodic, of N_{P} say), uniformly distributed in $\mathcal{U}[0, N_{\mathrm{H}}] \cap \mathbf{Z}$, that delays $g(t)$ in one of the N_{H} chips composing a *frame* ($T_{\mathrm{F}} = N_{\mathrm{H}} T_{\mathrm{C}}$). Thus, the transmitter has a (fixed) vector $\boldsymbol{c} = [c_0, \ldots, c_{N_{\mathrm{P}}-1}]^T$ of discrete i.i.d. uniform random variables. For notational convenience, in the following we will use c_i instead of $c_{i \bmod N_{\mathrm{P}}}$.

Furthermore, in order to introduce redundancy, the modulator has the ability of coding a bit of information into N_{S} symbols, e.g. with a *repetition code* (in that case, $T_b = N_{\mathrm{S}} T_{\mathrm{F}}$).

The transmitted signal can be written as follows

$$s(t) = \sqrt{\mathcal{E}_b} \sum_{n \geq 0} g(t - nT_b; b_n)$$

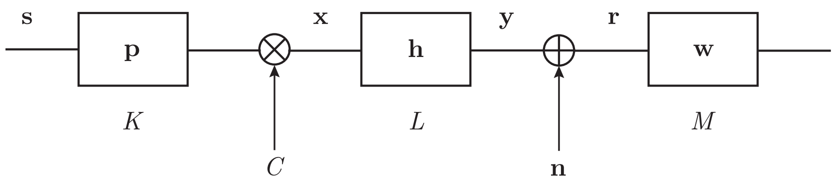


FIGURE 1.1: Basic model. (K, L, M) denotes that the prefilter has K taps, the channel has L paths and the RAKE has M fingers.

where

$$g(t; b_n) = \begin{cases} \sum_{i=0}^{N_S-1} (-1 + 2b_n) g(t - iT_F - c_{nN_s+i}T_C) & \text{for PAM} , \\ \sum_{i=0}^{N_S-1} g(t - iT_F - c_{nN_s+i}T_C - b_n\varepsilon) & \text{for PPM} . \end{cases}$$

Hereinafter in this chapter, we will consider $N_S = 1$. Furthermore, adopting a block transmission paradigm, w.l.o.g. we can rewrite the previous waveforms for the first information bit only:

$$g(t; b) = \begin{cases} (-1 + 2b) g(t - cT_C) & \text{for PAM} , \\ g(t - cT_C - b\varepsilon) & \text{for PPM} . \end{cases}$$

Thus, regardless of the time-hopping shift, in the PAM case, the basic pulse is simply $g(t)$. Because of the dimensionality of the signal space (that is 1), the two possible signals to be transmitted are:

$$s_m(t) = (-1 + 2m) g(t) \ , \quad m = 0, 1 \ ,$$

and a base for this space is (for instance) given by $\mathcal{B} = \{s_1(t)\}$.

In the PPM case, the signals are:

$$s_m(t) = g(t - m\varepsilon), \quad m = 0, 1 \ .$$

If $\varepsilon \geq T_M$, they are orthogonal, being T_M the duration of the pulse. Anyway, the signal space has dimension 2, and a base is (for instance) given by $\mathcal{B} = \{s_0(t), s_1(t)\}$.

1.1.2 **Channel.** The channel statistic for UWB communication is unique due to the ultra high resolution of receivers. Both IEEE 802.15.3a and IEEE 802.15.4a channel models are based on the seminal work of Saleh and Valenzuela. We discuss the channel thoroughly later, whereas here we aptly describe it in a by far simpler way, that is as much as we need now:

$$h(t) = \sum_{\ell=1}^{L} \alpha_\ell \delta(t - \tau_\ell) \ .$$

Note 1.1. *We stress that L is the number of paths of the channel.*

1.1.3 **Precoder.** We apply here the time-reversal concept introducing a filter that is nothing but the channel reversed (= inverted) in time:

$$p(t) = \sum_{k=1}^{L} \alpha_k \delta(t + \tau_k) \ .$$

We don't care about the causality of this filter, but it is evident that in a real experiment it would be necessary a delay (at least) equals to τ_L.

In general, we could use a lesser complex filter with $K \leq L$ taps, selecting only the K strongest paths of $h(t)$. In this case we have:

$$p(t) = \sum_{k \in \mathcal{K}} \alpha_k \delta(t + \tau_k) \ ,$$

where $\mathcal{K} \subseteq \{1, 2, \ldots, L\}$.

Note 1.2. *We stress that K is the number of taps of the prefilter.*

Note 1.3. To carry out a correct comparision of performance among various systems, we introduce a power constraint for the transmitter, namely, the power sent is constant. We taking into account this as follows. Let be $x(t)$ the signal sent, thus (see FIGURE 1.1 on the previous page)

$$x(t) = C(s * p)(t) = C \sum_{k \in \mathcal{K}} \alpha_k \, s(t + \tau_k) \ , \quad C \in \mathbf{R}^+ \ .$$

ASSUMPTION 1.1. We assume that $g(t)$ has a support $[0, T_\mathrm{M}]$ with

$$0 < T_\mathrm{M} \leq \min_{\substack{0 \leq i,j \leq L \\ i \neq j}} |\tau_i - \tau_j| \ .$$

We may paraphrase this condition stating that the smallest inter-arrival time is greater than the pulse width. It is clear that this is *not always* true, nonetheless it is a common hypotesis.

With this premise, we can straightforwardly compute the energy of $x(t)$

$$\mathcal{E}_x = C^2 \mathcal{E}_s \sum_{k \in \mathcal{K}} \alpha_k^2$$

The power constraint reads as $\mathcal{E}_x = \mathcal{E}_s$, so

$$C = \frac{1}{\sqrt{\sum_{k \in \mathcal{K}} \alpha_k^2}} \ .$$

Remark 1.1. This constrain implies that

$$\max_{t \geq 0} |h_\mathrm{e}(t)| = \sqrt{\sum_{k \in \mathcal{K}} \alpha_k^2} \ .$$

1.1.4 **Selective RAKE.** The optimum demodulator for processing a wideband signal is known as RAKE correlator. It was found by Price and Green in 1958 and it is the filter matched to the whole *useful* (= without noise and interference) signal at the receiver. In our case, let us call $y(t) = (x * h)(t)$ the useful signal and $r(t) = y(t) + n(t)$ the received signal corrupted by the WGN $n(t)$ with variance σ_n^2. Then the RAKE maximizes the SNR. We will return to this point later stressing that it is nothing but a Maximal-Ratio Combiner (MRC), that historically, however, was found later (in 1959 by Brennan).

In general, an S-RAKE with M fingers choose the M strongest path of the equivalent channel $h_\mathrm{e}(t) = C(h * p)(t)$, given by:

$$h_\mathrm{e}(t) = \frac{1}{\sqrt{\sum_{k\in\mathcal{K}} \alpha_k^2}} \sum_{k\in\mathcal{K}} \sum_{\ell=1}^{L} \alpha_k \alpha_\ell \delta(t - \tau_\ell + \tau_k).$$

1.1.5 **Signals.** We now rewrite this signal in order to emphasize some important properties. Let us start noting that we can write $h_\mathrm{e}(t)$ with a special partition of the set of indices $\{1, \dots, L\} \times \mathcal{K}$:

$$\begin{aligned} h_\mathrm{e}(t) = \frac{1}{\sqrt{\sum_{k\in\mathcal{K}} \alpha_k^2}} \Bigg\{ & \left[\sum_{k\in\mathcal{K}} \alpha_k^2\right] \delta(t) \\ & + \sum_{k\in\mathcal{K}} \alpha_k \sum_{\substack{\ell\in\mathcal{K} \\ \ell\neq k}} \alpha_\ell \delta(t - \tau_\ell + \tau_k) \\ & + \sum_{k\in\mathcal{K}} \alpha_k \sum_{\ell\notin\mathcal{K}} \alpha_\ell \delta(t - \tau_\ell + \tau_k) \Bigg\}. \end{aligned}$$

Let us explore the three terms in parentheses.

First term. With a *full*-TR ($K = L$), $h_\mathrm{e}(t)$ would be the (normalized) autocorrelation of the channel, the first term representing its energy. With a *partial*-TR, the first term is by far the greatest, but it decreases monotonically with the cardinality of $\mathcal{K}$ (= number of taps considered). This means that, with $M = 1$ (1-finger RAKE), the first term would represent the chosen path of the equivalent channel. We expect a performance increase with K.

Second term. This term is composed of $K(K-1)$ signals. It is an even function and represents a portion of the autocorrelation function that we would obtain if $K = L$. We may rewrite that in the following way:

$$\sum_{k\in\mathcal{K}} \alpha_k \sum_{\substack{\ell\in\mathcal{K} \\ \ell\neq k}} \alpha_\ell \delta(t - \tau_\ell + \tau_k) = \sum_{k\in\mathcal{K}} \alpha_k \sum_{\substack{\ell\in\mathcal{K} \\ \ell> k}} \alpha_\ell [\delta(t - \tau_\ell + \tau_k) + \delta(t + \tau_\ell - \tau_k)] \;.$$

Third term. This term is composed of $K(L-K)$ signals of minor entity. Note that the coefficients of this third term are always lower (in absolute value) that those in the second one. This implies that an M-RAKE, with $M \le K(K-1) + 1$, would choose the paths included in the firsts two terms, discarding those in the third one.

Note 1.4. Fixing L, we always have the following bounds: $K \le L$ and $M \le 1 + K(K-1) + K(L-K) = 1 + K(L-1)$.

§1.2 A SIMPLE CASE $(K, L, 1)$.

In this section[1] we prove that $(K, L, 1) \sim (1, L, K)$, $\forall K \leq L$, thus a *partial*-TR with a 1-RAKE has the same performance of a *partial*-RAKE without TR, provided that they have the same number of taps.

In order to do this, we fix some notation with a well known and simple example.

Example 1.1 (BINARY ANTIPODAL AND BINARY ORTHOGONAL SIGNALING SCHEMES). BPSK assigns to a bit m the following signal:

$$s_m(t) = (-1 + 2m)A\,g(t) \ , \quad m = 0, 1 \ .$$

The signal space, namely $\mathcal{S} = \text{Span}\{s_0(t), s_1(t)\} = \text{Span}\{s_1(t)\}$, has $\dim \mathcal{S} = 1$ and a basis is given by $\mathcal{B} = \{\phi_1(t)\}$, with $\phi_1(t) = s_1(t)/\sqrt{A}$. We have then to correlate with the basis functions and estimate the symbol (or bit, in this case) sent by means of a ML criterion. Thus, to demodulate such a signal, we may have only one correlator (with $s_1(t)$) and decide which bit it was likely sent looking at the sign. Nevertheless, we may also use two correlators (with $\phi_0(t)$ and $\phi_1(t)$) obtaining two samples (or *correlation metrics*), say CM_0 and CM_1, and decide looking at the sign of $D = CM_1 - CM_0$. To compute the BEP in AWGN, w.l.o.g. we can think to send the bit 1, evaluating the BEP as $\Pr\{D < 0\}$. Thus

$$D = CM_1 - CM_0 = 2A + \langle n(t), s_1(t)\rangle - \langle n(t), s_0(t)\rangle = 2A + 2\nu_1 \ ,$$

with $\nu_1 = \langle n(t), s_1(t)\rangle \sim \mathcal{N}(0, A^2\sigma_n^2)$, and hence the ratio between the powers of the useful part (σ_a^2) and the noise part (σ_ν^2) is

$$\frac{\sigma_a^2}{\sigma_\nu^2} = \frac{4A^4}{4\sigma_n^2 A^2} = \frac{A^2}{\sigma_n^2} \implies P_e = Q\left(\frac{\sigma_a}{\sigma_\nu}\right) = Q(\sqrt{2\gamma_b}) \ , \quad \gamma_b = \frac{A^2}{2\sigma_n^2}.$$

We can repeat this computation with a binary orthogonal signaling scheme, such as PPM. With similar notations, we have

$$D = CM_1 - CM_0 = A + \langle n(t), s_1(t)\rangle - \langle n(t), s_0(t)\rangle = A + \nu_1 - \nu_0 \ .$$

Now $(\nu_1 - \nu_0) \sim \mathcal{N}(0, 2A^2\sigma_n^2)$ and

$$\frac{\sigma_a^2}{\sigma_\nu^2} = \frac{A^4}{2\sigma_n^2 A^2} = \frac{A^2}{2\sigma_n^2} \implies P_e = Q\left(\frac{\sigma_a}{\sigma_\nu}\right) = Q(\sqrt{\gamma_b}) \ , \quad \gamma_b = \frac{A^2}{2\sigma_n^2}.$$

Thus, we may write the BEP of the generic binary signaling scheme as follows:

$$P_e = Q\left(\sqrt{(1-\rho)\gamma_b}\right) \ , \quad \gamma_b = \frac{A^2}{2\sigma_n^2} \text{ and } \rho = \frac{\langle s_0(t), s_1(t)\rangle}{\langle s_1(t), s_1(t)\rangle} \ .$$

This concludes the example. ◊

Let us find the BEP in the $(K, L, 1)$ case. The signal received (as illustrated in FIGURE 1.1 on page 4) is

$$r_m(t) = \left[\sum_{k\in\mathcal{K}} \alpha_k^2\right]^{\frac{1}{2}} s_m(t) + \frac{1}{\sqrt{\sum_{k\in\mathcal{K}} \alpha_k^2}} \sum_{k\in\mathcal{K}} \sum_{\substack{\ell=1\\ \ell\neq k}}^{L} \alpha_k \alpha_\ell s_m(t - \tau_\ell + \tau_k) + n(t) \ ,$$

[1]We write $(K, L, M) \sim (K', L', M')$ to denote two configurations that have the same performance in terms of a specified parameter (e.g. SNR or BER).

where $n(t)$ is a WGN with variance σ_n^2 and $s_m(t)$ is the signal that modulates a bit m. A 1-finger RAKE will correlate this signal with the highest path, i.e. likely[2] the correlation metric will be (we drop the explicit reference to the time-hopping code)

$$CM_1 = \left\langle r_1(t), \left[\sum_{k\in\mathcal{K}} \alpha_k^2\right]^{\frac{1}{2}} s_1(t) \right\rangle =: A + \nu$$

where

$$A := \mathcal{E}_s \sum_{k\in\mathcal{K}} \alpha_k^2 \quad \text{and} \quad \nu := \left\langle n(t), \left[\sum_{k\in\mathcal{K}} \alpha_k^2\right]^{\frac{1}{2}} s_1(t) \right\rangle \sim \mathcal{N}\left(0, \sigma_n^2 \mathcal{E}_s \sum_{k\in\mathcal{K}} \alpha_k^2\right) .$$

It yields

$$\gamma_b := \frac{A^2}{2\sigma_\nu^2} = \frac{\mathcal{E}_s \sum_{k\in\mathcal{K}} \alpha_k^2}{N_0} , \quad \sigma_n^2 := N_0/2 ,$$

that is the same well-known result of a selective K-RAKE.

Remark 1.2. This result shows the remarkable property that, having fixed L, in the plane (K, M), $K, M \geq 1$, an iso-BEP or iso-energy curve that starts in $(k, 1)$ will finish in $(1, k)$, irrespective of the modulation type (PPM or PAM).

§1.3 The general case (K, L, M).

A method to approach the general case is to think of $g(t)$ as a spike, thus of $x(t)$ as a spike train. From this perspective, the RAKE simply collects the energy of the M greatest paths (in absolute value) of the equivalent channel. The set of the amplitudes of all paths can be partitioned in the following way:

$$\begin{array}{ll} \left\{\sum_{k\in\mathcal{K}} \alpha_k^2\right\} , & \\ \{\alpha_k \alpha_\ell\} , & k \in \mathcal{K},\ \ell \in \mathcal{K},\ \ell \neq k, \\ \{\alpha_k \alpha_\ell\} , & k \in \mathcal{K},\ \ell \notin \mathcal{K} . \end{array}$$

It is not developed here a general framework to deal with this general problem. Nonetheless, we will present at the end of the chapter an insight into analytical approaches.

Remark 1.3. If $M = 1$, the first set is chosen. The energy collected by the RAKE is

$$\mathcal{E} = \mathcal{E}_s \sum_{\ell=1}^{K} \alpha_\ell^2 .$$

It is worth to note that, in this case, a TR precoder is *optimum* in the sense that it maximizes the SNR achievable at the receiver. Let us formulate the optimization problem. Suppose that $p(t)$ can be written as

$$p(t) = \sum_{k\in\mathcal{K}} p_k \delta(t + \tau_k) , \quad p_k \in \mathbf{R} .$$

[2]As far as the right term in the inner product is the highest. We can recognise that it would be pick of the normalized (auto)correlation of the channel if $\mathcal{K} \equiv$, that is of course the maximum of the signal in that case. In other cases, it can be shown that there is an evanescent probability for the alternative hypotesis.

The sent signal is

$$x(t) = C(s * p)(t) = C \sum_{k \in \mathcal{K}} p_k s(t + \tau_k) \ ,$$

and the energy normalization take the form

$$\mathcal{E}_x = C^2 \mathcal{E}_s \sum_{k \in \mathcal{K}} p_k^2 \equiv \mathcal{E}_s \implies C = \frac{1}{\sqrt{\sum_{k \in \mathcal{K}} p_k^2}} \ .$$

When this signal pass through the multipath channel we have

$$y(t) \coloneqq (x * h)(t) = \frac{1}{\sqrt{\sum_{k \in \mathcal{K}} p_k^2}} \left[\sum_{k \in \mathcal{K}} p_k \alpha_k \right] s(t) + \cdots \ ,$$

with the dots stating for minor terms that would be likely[3] discarded by a 1-finger RAKE. The energy collected is

$$\mathcal{E} = \mathcal{E}_s \frac{\left[\sum_{k \in \mathcal{K}} p_k \alpha_k \right]^2}{\sum_{k \in \mathcal{K}} p_k^2}$$

and we want to maximize it in the p_k's in order to maximize the SNR at the receiver. Let us set $\boldsymbol{\alpha} = [\alpha_k]_{k \in \mathcal{K}}^T$ and $\boldsymbol{p} = [p_k]_{k \in \mathcal{K}}^T$. These are vectors in $\mathbf{R}^K$. In our framework, we leave p_k unbounded because we set up later the normalization with C; however, we may regard the ratio as follows

$$\frac{\left[\sum_{k \in \mathcal{K}} p_k \alpha_k \right]^2}{\sum_{k \in \mathcal{K}} p_k^2} = \left[\sum_{k \in \mathcal{K}} \alpha_k \frac{p_k}{\sqrt{\sum_{j \in \mathcal{K}} p_j^2}} \right]^2 = \left[\sum_{k \in \mathcal{K}} \alpha_k \beta_k \right]^2 \ ,$$

constraining the β_k's to be finite in norm. These coefficients may be seen as the Dirac weights of $\beta(t) \coloneqq Cp(t)$. Now we can state the optimization problem as follows

$$(\mathscr{P}) \ \begin{cases} \max\limits_{\boldsymbol{\beta}} & |\boldsymbol{\beta}^T \boldsymbol{\alpha}|^2 \\ \text{s.t.} & \|\boldsymbol{\beta}\| = 1 \ . \end{cases}$$

The solution come by a straightforward application of the Cauchy-Schwarz inequality, $\boldsymbol{\beta}^\star = \boldsymbol{\alpha}/\|\boldsymbol{\alpha}\|$.

Remark 1.4. If $M = 1 + K(K-1)$, the union of the firsts two sets is chosen. The energy collected by the RAKE is

$$\mathcal{E} = \mathcal{E}_s \frac{1}{\sum_{k \in \mathcal{K}} \alpha_k^2} \left[\left(\sum_{k \in \mathcal{K}} \alpha_k^2 \right)^2 + \sum_{i \in \mathcal{K}} \sum_{\substack{j \in \mathcal{K} \\ j \neq i}} \alpha_i^2 \alpha_j^2 \right] = \mathcal{E}_s \left[\sum_{k \in \mathcal{K}} \alpha_k^2 + \frac{1}{\sum_{k \in \mathcal{K}} \alpha_k^2} \sum_{i \in \mathcal{K}} \sum_{\substack{j \in \mathcal{K} \\ j \neq i}} \alpha_i^2 \alpha_j^2 \right] .$$

[3] It remains valid the note 2 on the previous page.

Parameter	**Value**	**Units**
λ (ray arrival rate)	2	[GHz]
Λ (cluster arrival rate)	20	[MHz]
γ (ray decay factor)	2	[ns]
Γ (cluster decay factor)	5	[ns]
σ_1 (cluster fading std. dev.)	3.3941	[dB]
σ_2 (ray fading std. dev.)	3.3941	[dB]

TABLE 1.1: Channel model parameters.

Remark 1.5. If $K = L$ and $M = 1 + L(L-1)$, both maximum SNR and energy are achieved. The latter is

$$\mathcal{E} = \mathcal{E}_s \frac{1}{\sum_{\ell=1}^{L} \alpha_\ell^2} \left[\left(\sum_{\ell=1}^{L} \alpha_\ell^2 \right)^2 + \sum_{i=1}^{L} \sum_{\substack{j=1 \\ j \neq i}}^{L} \alpha_i^2 \alpha_j^2 \right] .$$

For future reference, we rewrite the terms in parentheses. The first term can be viewed as follows

$$\left(\sum_{\ell=1}^{L} \alpha_\ell^2 \right)^2 = \sum_{\ell=1}^{L} \sum_{j=1}^{L} \alpha_\ell^2 \alpha_j^2 ,$$

while the second term can take the form

$$\sum_{i=1}^{L} \sum_{\substack{j=1 \\ j \neq i}}^{L} \alpha_i^2 \alpha_j^2 = \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i^2 \alpha_j^2 - \sum_{\ell=1}^{L} \alpha_\ell^4 .$$

Now the whole parenthesis can be written as

$$\left(\sum_{\ell=1}^{L} \alpha_\ell^2 \right)^2 + \sum_{i=1}^{L} \sum_{\substack{j=1 \\ j \neq i}}^{L} \alpha_i^2 \alpha_j^2 = 2 \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i^2 \alpha_j^2 - \sum_{\ell=1}^{L} \alpha_\ell^4$$

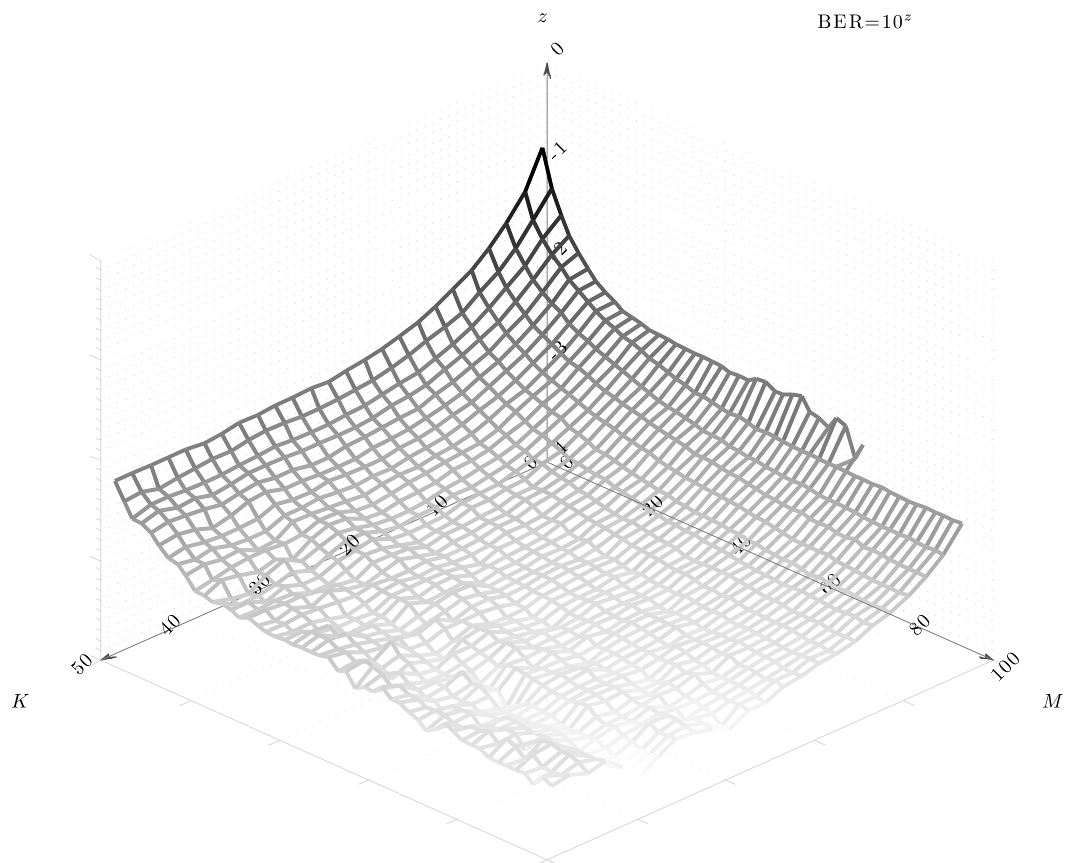
This form will be very useful.

§1.4 SIMULATION RESULTS.

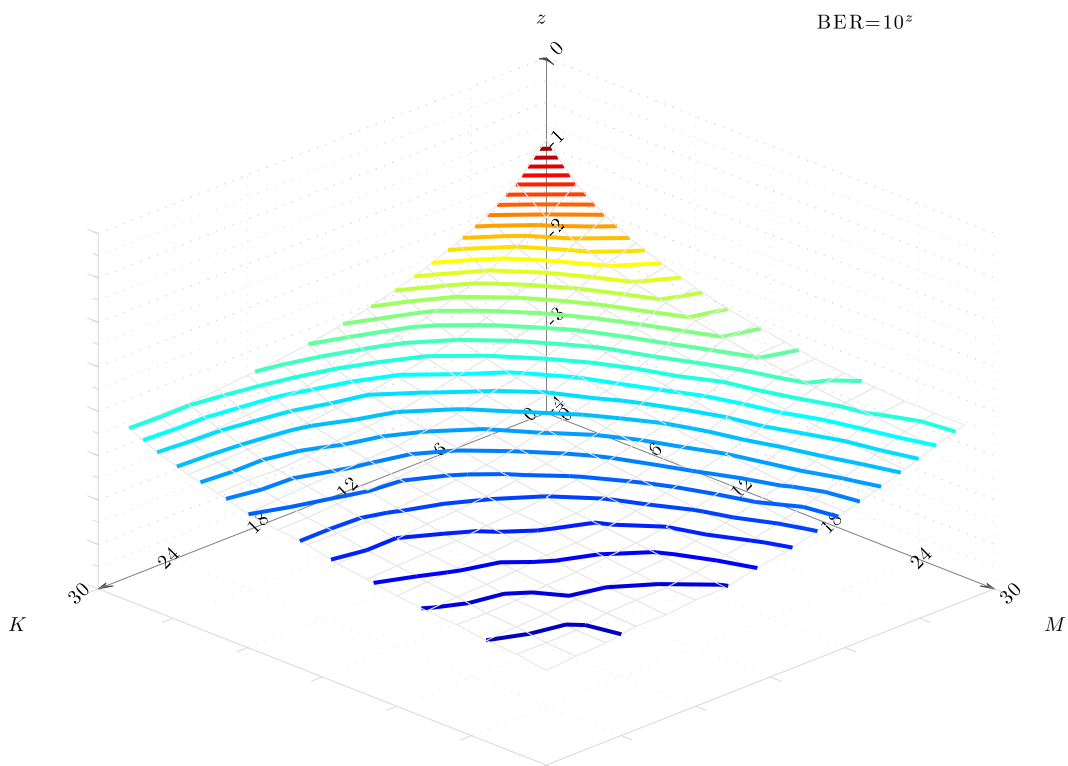
In this section, we turn to give an overview on results from simulations of a TH-UWB system in an IEEE 802.15.3a channel. The parameters adopted are listed in TABLE 1.1.

Here are presented results of TH-BPAM-UWB for BER; and TH-BPAM-UWB and TH-BPPM-UWB (that yields the same results) for the estimation of the energy collected by the RAKE receiver. BER with PPM follows the same trajectories with the usual 3 dB gap with respect to PAM .

In FIGURE 1.2 and 1.3, it is evident the monotonicity of both the BER and the energy surfaces with respect to each parameter. In the latter, we compare the iso-energy and iso-BER curves. As already stated via theoretical computations, we

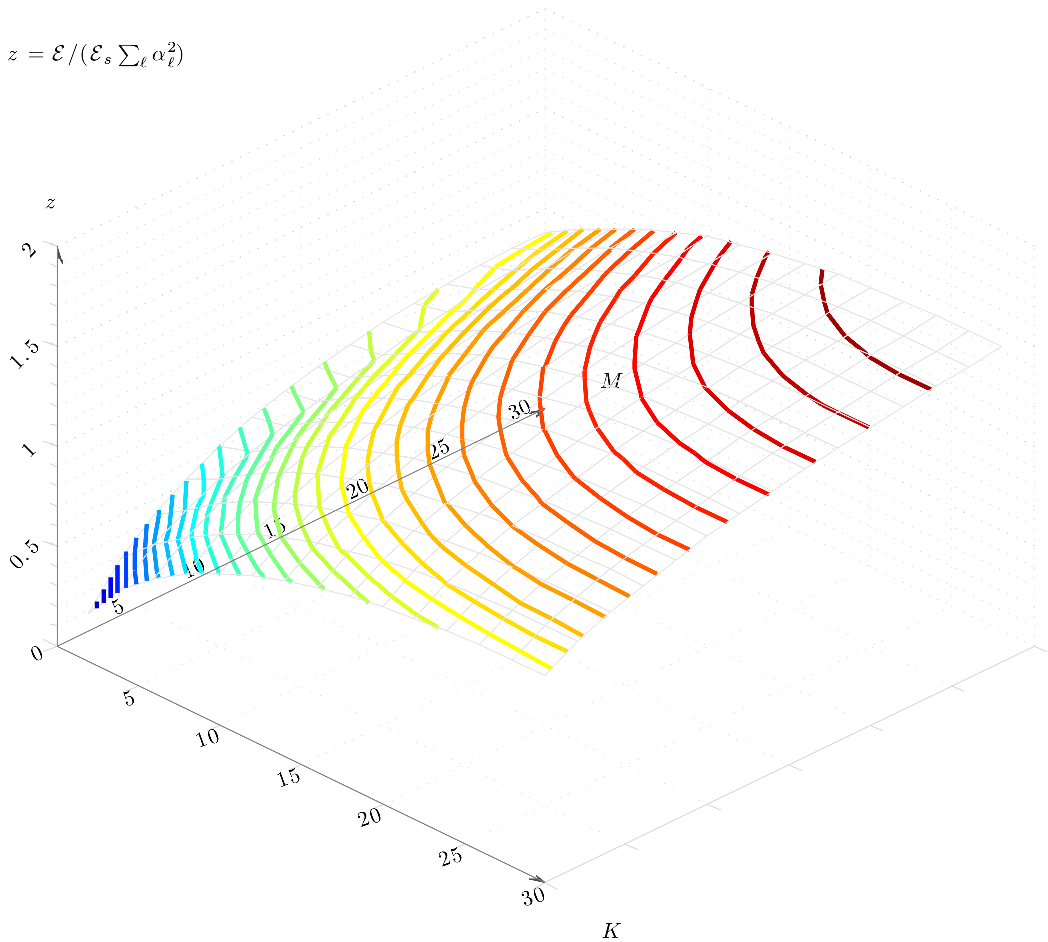


(A) *BER in function of the number of taps K and the number of fingers M.*

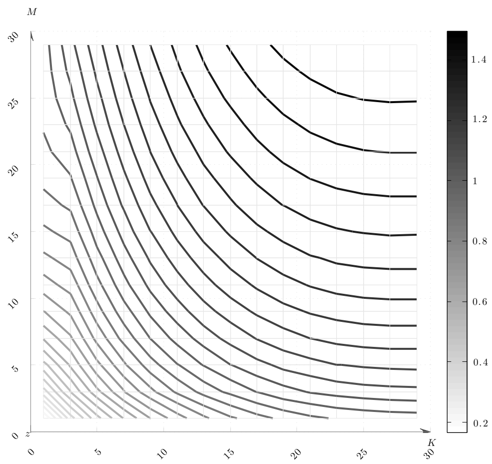


(B) *3D iso-BER curves.*

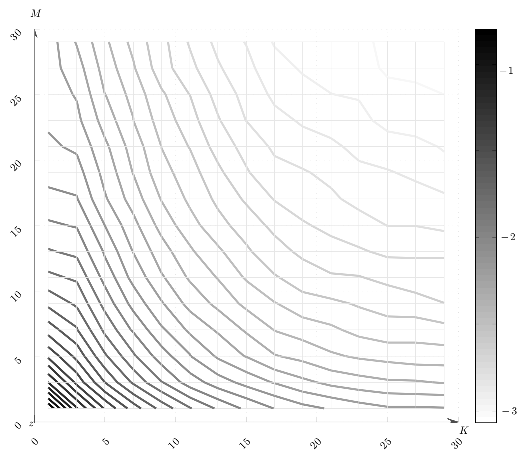
FIGURE 1.2: Average BER estimation ($\gamma_b = 5$ [dB]).



(A) *3D iso-energy curves. Average energy is normalized with respect to the* $(1, L, L)$ *case.*



(B) *Iso-energy curves.*



(C) *Iso-BER curves.*

FIGURE 1.3: Average energy estimation.

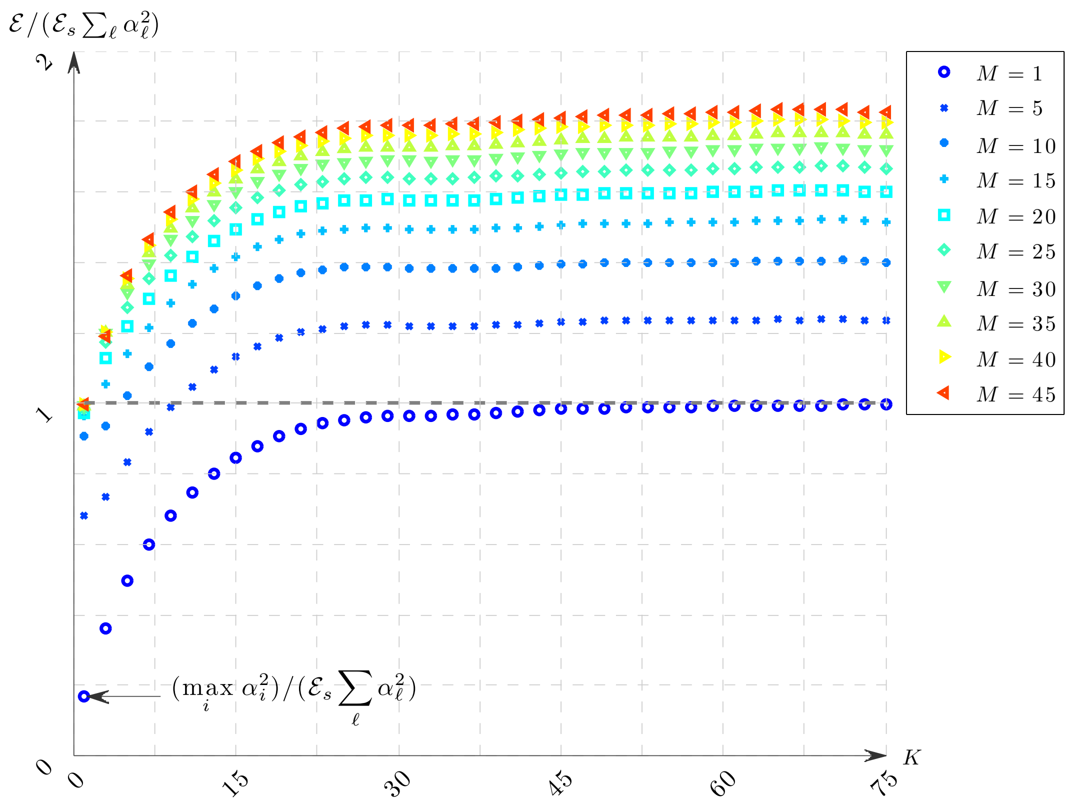


FIGURE 1.4: Average energy collected by an M-RAKE varying the number of taps K in the prefilter.

note that the generic curve in the plane (K, M) that starts in $(k, 1)$ will end in $(1, k)$. This suggests an obvious rule-of-thumb that provides a fairly good fitting considering hyperbolas as these curves.

Fixing a performance (BER or energy), we can start choosing a number c of fingers in a system with a RAKE receiver and without TR; then we move on the curve $(k, c/k)$, shifting the complexity from the receiver to the transmitter. We can switch all the complexity or just a part of it, or we can outperform the initial performance increasing the complexity of the transmitter.

Let us find, for example, the solution of the following problem: Minimize the total number of taps and fingers, fixing a performance. To be more specific, let be $(k, m) \in \mathbf{Z}_+^2$ the pair denoting the number of taps and fingers employed, respectively. Thus we want to solve the problem

$$\begin{cases} \min & k + m \qquad\qquad (k, m) \in \mathbf{Z}_+^2 \ , \\ \text{s.t.} & km = c \ , \end{cases}$$

where c is a feasible constant that depends on the performance to reach. We may generalise this problem assigning a cost to each choice. In this case the problem becomes

$$\begin{cases} \min & ak + bm \qquad\qquad (k, m) \in \mathbf{Z}_+^2, \quad a, b \in \mathbf{R}_+ \ , \\ \text{s.t.} & km = c \end{cases} \ .$$

We will proceed embedding the problem in $\mathbf{R}_+^2$ and then choosing the nearest integer pair in the lattice $\mathbf{Z}_+^2$, altough of course this couldn't be the true solution (anyway, it would be very close to it). By elementary calculus, we find that $k^\star = \sqrt{bc/a}$, $m^\star = \sqrt{ac/b}$ and the attained minimum is $2\sqrt{abc}$. If $a = b = 1$, the optimum number of taps as well as fingers is $\sqrt{c}$.

This result, which suffers of some inaccuracy due to the extremely simple model adopted accepting the hyperbola hypothesis, sheds some light on the problem of finding a trade-off on the complexity between transmitter and receiver. To summarize, a $(1, L, c)$ system, i.e. *no*-TR and *all*-RAKE, is approximately equivalent to a system $(\sqrt{c}, L, \sqrt{c})$.

In FIGURE 1.4 it is shown the average energy collected by the receiver in function of the number of taps of the prefilter. The plot is normalized with respect to the energy collected by an *all*-RAKE.

It is clearly visible that the energy collected by an $(L, L, 1)$ system is the same collected by an *all*-RAKE.

There exists an asymptotic energy that a system can collect. This is proven in the next sections, which are devoted to a thorough description of the channel model and very powerful analytic techniques based on theory of point processes.

§1.5 THE CHANNEL MODEL: A POINT PROCESS PERSPECTIVE.

We introduce the very basic concepts of point process theory in a quite informal way. We refer the more purists to [CI80] [DVJ08] whilst the casual reader surely would appreciate [Str10]. We pursue here a fairly intuitive line, giving a very concise, self-contained treatment of the (only) results we need.

1.5.1 **Why we are interested in point processes.** The theory of point processes is a vast and active area of probability. It finds its most powerful application in statistics for analyzing spatial data. Our goal is discover the properties of the channel by means of this theory.

In order to do this, we regard the channel response as follows

$$x(t) = \sum_{\ell=1}^{L} \gamma_\ell s(t - \tau_\ell) =: \sum_{\ell=1}^{L} \phi(\tau_\ell, \gamma_\ell)(t) \ ,$$

having defined $\phi(\tau_\ell, \gamma_\ell)(t) := \gamma_\ell s(t - \tau_\ell)$. Therefore, $x(t)$ is the sum of a function (in this case $\phi\colon \mathbf{R}^2 \to \mathbf{R}$) evaluated at random arguments (τ, γ). It is called a *shot-noise random variable*. The name derives from the *shot effect*, in which the point of a time process have an effect that continues for a time after the event represented by each random point. So point processes *on the line* (= in $\mathbf{R}$) aptly model random events in time, such as the arrivals of customers in a queue, of particles in a Geiger counter, of impulses in a neuron or, for us, in a receiver of electromagnetic field.

1.5.2 Point processes.

DEFINITION 1.1 (*Point process*). A point process is a *random countable set* $\Pi \subset \mathcal{S}$, $\mathcal{S} \subseteq \mathbf{R}^m$, such that for each *measurable* $A \subseteq \mathcal{S}$, the random variable

$$N(A) := \#\{\Pi \cap A\}$$

is (almost surely) finite. ◁

We call $\mathcal{S}$ the *state space*. Owing to the randomness of Π, N is a random variable. Thus, according to the basis of probability theory, we have to define a triple $(\Omega, \mathscr{F}, \mathbb{P})$

where Ω is a set (of *elementary outcomes*), $\mathscr{F}$ a σ-field of subset of Ω (*events*) and $\mathbb{P}$ a *probability measure* that assigns a number in $[0,1]$ to every event, $\mathbb{P}\colon \mathscr{F} \to [0,1]$.

A point process is a random variable whose outcome is a countable subset (= a set of *points*) of $\mathcal{S}$, thus it is a function

$$\Pi\colon \Omega \to 2^{\mathcal{S}}$$

denoting with $2^{\mathcal{S}}$ the class of all countable subsets of $\mathcal{S}$. A realisation of the point process, $\Pi(\bar{\omega})$ say, is a countable subset of $\mathcal{S}$.

Now, for fixed A, also $N(A)$ is a random variable, thus it is a function

$$N(A)\colon \Omega \to \overline{\mathbf{N}}\,, \qquad \overline{\mathbf{N}} = \{0, 1, 2, \ldots, \infty\}\ .$$

We require this function to be *measurable* for each A, which allow us to work with the measure $\mathbb{P}$. Therefore, we may take A as a (bounded) Borel subset of $\mathcal{S}$.

Example 1.2 (*Stars at night*). A pictorial example is given by the stars in the sky. The whole sky is $\mathbf{R}^2$. The underlying process fix the position of the stars, thus $\Pi(\bar{\omega})$ is the set of the visible stars (= a picture of the sky at a given time). $\mathcal{S}$ is the portion of the sky you can see. $2^{\mathcal{S}}$ is the set of all possible configurations of stars in the sky (= all possible pictures). A is a patch of the portion of the sky that you can see; then $N(A)$ is the number of stars in the patch. ♢

In the following, we abuse the notation writing Π instead of $\Pi(\omega)$ for the generic realisation of the point process.

To go further, we have to specify the properties of the underlying process. We restrict our discussion to Poisson point processes.

DEFINITION 1.2 (*Poisson point process*). A Poisson point process is a point process Π such that:

(*i*) if $\{A_i\}$ is a family of disjoint subsets of $\mathcal{S}$, then $N(A_i)$ are independent, and

(*ii*) $N(A) \sim \mathcal{P}(\mu(A))$,

where $\mathcal{P}(\mu)$ stands for Poisson distribution with parameter μ. ◁

Actually, the parameter is the mean. In fact, from direct computation, if $x \sim \mathcal{P}(\mu)$, then $\mathbb{E}\{x\} = \mu$. For this reason, $\mu(A)$ is called the *mean measure* of A. It is very useful to provide this measure with a non-negative function $\lambda\colon \mathcal{S} \to \mathbf{R}^+$ such that

$$\mu(A) = \int_A \lambda(x)\mathrm{d}x$$

In general $\mathcal{S} \subseteq \mathbf{R}^m$. The function λ is called *rate*, *intensity* or *density* of the process if $m = 1$, $m = 2$ or $m \geq 3$, respectively. Such a function is the tool that allow us to compute expected values of functions evaluated at process points, to the same extent that probability density functions are employed to compute expectations of functions of random variables. The existence of this function follows from Radon[4]-Nikodym[5] theorem and it is also named Radon-Nikodym derivative.

Note that, at least formally, we can write

$$\mu(A) = \int_A \mu(\mathrm{d}x)$$

[4] Johann Radon, austrian mathematician (1887–1956)

[5] Otto Nikodym, polish mathematician (1887–1974)

such that $\mu(\mathrm{d}x) = \lambda(x)\mathrm{d}x$.

From a genuinely elementary point of view, for $\Pi \cap A$ are random both the *number* and the *locations* of points. Thus we are able to write down the p.d.f. of the number n, that is Poissonian

$$p_{\mathrm{n}}(n) = \frac{\Lambda^n}{n!}\mathrm{e}^{-\Lambda} , \qquad \Lambda := \int_A \lambda(x)\mathrm{d}x$$

and the p.d.f. of the i.i.d. locations, that is uniform (this follows from the definition of Poisson process)

$$p_{\mathrm{x}_i}(x_i) = \lambda(x_i)/\Lambda , \qquad x_i \in A .$$

Thus we can find the joint p.d.f. for $\Pi \cap A$:

$$p_{\Pi\cap A}(n, x_1; \ldots, x_n) = p_{\mathrm{n}}(n) \prod_{i=1}^{n} p_{\mathrm{x}_i}(x_i) = \frac{1}{n!}\mathrm{e}^{-\Lambda} \prod_{i=1}^{n} \lambda(x_i) .$$

This is remarkable: now we are able to find the expectation of a generic function evaluated on the Poisson process, say $\Phi\colon 2^{\mathcal{S}} \to \mathbf{R}$, as

$$\mathbb{E}\{\Phi\} = \sum_{n\geq 0} p_{\mathrm{n}}(n) \int_{\mathcal{S}^n} \Phi(x_1; \ldots, x_n) \prod_{i=1}^{n} p_{\mathrm{x}_i}(x_i)\mathrm{d}x_1 \ldots \mathrm{d}x_n .$$

We are interested in sums like

$$\Phi = \sum_{x\in\Pi\cap A} \phi(x) .$$

We could straightforwardly obtain the expectation of these sums with the previous formula. We would find that

$$\mathbb{E}\{\Phi\} = \int_A \phi(x)\lambda(x)\mathrm{d}x = \int_A \phi(x)\mu(\mathrm{d}x) .$$

This is known as (a form of) the *Campbell's theorem*, but the most exciting form of this theorem allow us to find the moment generating function (m.g.f.) of Φ, thus its p.d.f.

$$M_\Phi(\theta) = \mathbb{E}\left\{\mathrm{e}^{\theta\Phi}\right\} = \exp \int_A \left[\mathrm{e}^{\theta\phi(x)} - 1\right] \mu(\mathrm{d}x) .$$

1.5.3 **Marked point processes.** Let Π be a Poisson process with mean measure μ. We associate a random variable $m_x \in \mathcal{M}$ (*mark* of x) to each point $x \in \Pi$. We assume that (*1*) the distribution of m_x may depend on x but not on other points of Π, and (*2*) the m_x for different x are independent.

The pair (x, m_x) can be regarded as a random point $x^\star \in \mathcal{S} \times \mathcal{M}$. The totality of points $x^\star$ forms a random countable subset $\Pi^\star = \{(x, m_x)\colon x \in \Pi\} \subset \mathcal{S} \times \mathcal{M}$. Now the sum on the product space takes the form

$$\Phi^\star = \sum_{x\in\Pi\cap A} \phi(x, m_x) ,$$

that is similar to the channel model we sketch at the beginning.

The fundamental result is that $\Pi^\star$ *is a Poisson process on the product space* $\mathcal{S} \times \mathcal{M}$. The *marking theorem* states that the mean measure on $\Pi^\star$ is

$$\mu^\star(A \times B) = \iint_{A\times B} \mu(\mathrm{d}x) p(x, \mathrm{d}m) ,$$

where $p(x, m)$ is the p.d.f. of m_x. In fact, at least formally, we have

$$\mu^\star(\mathrm{d}x \times \mathrm{d}m) = \lambda^\star(x, m)\mathrm{d}x\mathrm{d}m , \qquad \lambda^\star(x, m) = \lambda(x)p(x, m) .$$

The Campbell's theorem of a marked process is a rather plain generalisation of the basic version, that is

$$\mathbb{E}\left\{\mathrm{e}^{\theta\Phi}\right\} = \exp \iint_{A\times B} \left[\mathrm{e}^{\theta\phi(x,m)} - 1\right] \mu^\star(\mathrm{d}x \times \mathrm{d}m) .$$

Note 1.5. It is really important to note that a marked point process is nothing but a point process, on the product space of points and marks, with intensity function $\lambda^\star$. This will be nearly fundamental for the channel model.

1.5.4 **Cluster point processes.** A cluster process consists of the superposition of *clusters* centered at points of a parent point process, being each cluster another point process. The parent process is called *center* (or *centre*) *process*, whereas the cluster process is called *subsidiary* or *daughter process*. Each cluster is i.i.d. both from other clusters and parent process.

1.5.5 **Generalized Saleh-Valenzuela.** We think of the channel as a cluster process in the plane (τ, γ), where τ is the arrival time of paths and γ their amplitudes (or gains). We describe the channel in accordance to the standard:

Center process

- the center process *start times* τ follow a *homogeneous Poisson process of rate* C (in TABLE 1.1 we referred it to as Λ), and
- the center process *amplitudes* γ follow a p.d.f. that we call $f_{\tau\tau}(\gamma)$, which depends only on the value of the start time τ, being independent with each other amplitude; these amplitudes may be viewed as marks of the Poisson point process of the center start times τ, but we embed τ and γ into a two-dimensional (Poisson) point process.

The center point process is characterized by the intensity

$$\lambda_1^{\mathrm{c}}(\tau, \gamma) \coloneqq C f_{\tau\tau}(\gamma)\chi_{[0,+\infty)}(\tau) .$$

We refer to its measure by $N_1^{\mathrm{c}}(\mathrm{d}\tau \times \mathrm{d}\gamma)$ and mean measure by $\mu_1^{\mathrm{c}}(\mathrm{d}\tau \times \mathrm{d}\gamma)$. An exception has to be made for the first path in LOS scenarios because it always arrives at time $\tau = 0$. Its measure will be $N_0^{\mathrm{c}} = \chi_B(0, \gamma_{00})$, where $\chi_B(\cdot)$ is the characteristic (or indicator) function of set B (that is, $\chi_B(x) = 1$ if $x \in B$ and $\chi_B(x) = 0$ otherwise), and $\gamma_{00} \sim f_{00}$. In other words, it is as if we had defined an intensity $\lambda_0^{\mathrm{c}}(\tau, \gamma) \coloneqq \delta(\tau) f_{00}(\gamma)$. The measure of the center point process is thus

$$N^{\mathrm{c}}(B) = N_0^{\mathrm{c}}(B) + N_1^{\mathrm{c}}(B) .$$

Cluster process

- the cluster process *start times* s follow, conditional on the cluster start time τ, a *homogeneous Poisson process of rate* R (in TABLE 1.1 we referred it to as λ), and
- the cluster process *amplitudes* g follow a conditional p.d.f. that we call $f_{\tau s}(g)$, which depends only on the value of the time s (other than on τ, of course), being independent with each other amplitude; these amplitudes may be viewed as marks of the Poisson point process of the cluster start times s, but we embed s and g into a two-dimensional (Poisson) point process.

The cluster point process is characterized by the *conditional* intensity

$$\lambda^{\mathrm{r}}(s, g|\tau, \gamma) := R f_{\tau s}(g) \chi_{[\tau,+\infty)}(s).$$

We refer to its measure by $N^{\mathrm{r}}(\mathrm{d}s \times \mathrm{d}g|\tau, \gamma)$ and mean measure by $\mu^{\mathrm{r}}(\mathrm{d}s \times \mathrm{d}g|\tau, \gamma)$. All clusters are identical and independent Poisson point processes. The mean measure can be find as follows

$$\mu_i^{\mathrm{r}}(\mathrm{d}s \times \mathrm{d}g) = \iint_{\mathbf{R}^2} \mu_i^{\mathrm{c}}(\mathrm{d}\tau \times \mathrm{d}\gamma)\mu^{\mathrm{r}}(\mathrm{d}s \times \mathrm{d}g|\tau, \gamma) \ , \qquad i \in \{0, 1\} \ .$$

For $i = 0$, that is for the first cluster, we have

$$\mu_0^{\mathrm{r}}(\mathrm{d}s \times \mathrm{d}g) = R f_{0s}(g) \chi_{[0,+\infty)}(s)$$

whereas for $i = 1$, that is for successive clusters, we have

$$\mu_1^{\mathrm{r}}(\mathrm{d}s \times \mathrm{d}g) = \iint_{\mathbf{R}^2} R f_{\tau s}(g) \chi_{[\tau,+\infty)}(s) C f_{\tau\tau}(\gamma) \chi_{[0,+\infty)}(\tau) \mathrm{d}\tau \mathrm{d}\gamma$$

The measure of the cluster point process is thus

$$N^{\mathrm{r}}(B) = N_0^{\mathrm{r}}(B) + N_1^{\mathrm{r}}(B) \ .$$

The augmented point process measure is then

$$N(B) = N_0^{\mathrm{c}}(B) + N_1^{\mathrm{c}}(B) + N_0^{\mathrm{r}}(B) + N_1^{\mathrm{r}}(B) \ .$$

It is possible to show that the following three measures,

$$N_0^{\mathrm{c}}(B) \ , \quad N_0^{\mathrm{r}}(B) \quad \text{and} \quad N_1^{\mathrm{c}}(B) + N_1^{\mathrm{r}}(B) \ ,$$

are independent. We can compute expectations separately and then add up, as well as m.g.f. and then multiply them.

The reason why we wrote $f_{\tau s}(\cdot)$ as the p.d.f. of marks is due to the channel model structure that set the p.d.f. of the ray (= element of the cluster process) in s of the cluster that starts in τ to be a (1/2)-Bernoulli mixture of log-normals with second moment equals to $\Omega_0 \mathrm{e}^{-\tau/\tau_0} \mathrm{e}^{-(s-\tau)/s_0}$. In TABLE 1.1 on page 10 we wrote γ for s_0 and Γ for τ_0.

We write $\mathbb{E}_{\tau s}\{\cdot\}$ to denote an expectation with respect to $f_{\tau s}$. Note that odd moments are null and even moments are equal to those of one-sided log-normals.

To be precise, the channel model says that all paths share the same σ^2 log-normal parameter:

$$g \sim \ln\mathcal{N}(m_{\tau s}, \sigma^2) \ .$$

In general, we can compute the n^{th} moment as

$$\mathbb{E}_{\tau s}\{g^n\} = e^{nm_{\tau s}+n^2\sigma^2/2} \ .$$

The channel model provides the second moment, thus

$$\mathbb{E}_{\tau s}\{g^2\} = e^{2m_{\tau s}+2\sigma^2} \equiv \Omega_0 \mathrm{e}^{-\tau/\tau_0}\mathrm{e}^{-(s-\tau)/s_0} \ .$$

This relation introduces a constraint. Solving in $m_{\tau s}$, we have

$$m_{\tau s} = -\sigma^2 + \frac{1}{2}\ln\Omega_0 + \frac{1}{2}\left[-\frac{\tau}{\tau_0} - \frac{s-\tau}{s_0}\right] \ .$$

We can express each moment with respect to the second one:

$$\mathrm{g}_{\tau s}^{(n)} := \mathbb{E}_{\tau s}\{g^n\} = \mathbb{E}_{\tau s}\{g^2\}^{n/2}\mathrm{e}^{n(n/2-1)\sigma^2} \ .$$

Note that we are actually interested only in even moments.

1.5.6 **Channel expectations.** We have already mentioned that the following are three independent measure:

$$N_0^{\mathrm{c}}(B) \ , \quad N_0^{\mathrm{r}}(B) \quad \text{and} \quad N_1^{\mathrm{c}}(B) + N_1^{\mathrm{r}}(B) \ .$$

For brevity, we will refer to them as $N_1(B)$, $N_2(B)$ and $N_3(B)$, respectively. An expectation with respect the whole channel, say $\mathbb{E}\{\Phi\}$, can be computed as $\mathbb{E}\{\Phi\} = \mathbb{E}_1\{\Phi\} + \mathbb{E}_2\{\Phi\} + \mathbb{E}_3\{\Phi\}$, being $\mathbb{E}_i\{\Phi\}$ the expectation with respect to the measure N_i (= averaging with the corresponding mean measure). We have

1st component

$$\mathbb{E}_1\{\Phi\} = \int_{\mathbf{R}^2} \phi(s,g)\delta(s)f_{00}(g)\mathrm{d}s\mathrm{d}g = \int_{\mathbf{R}} \phi(0,g)f_{00}(g)\mathrm{d}g \ .$$

2nd component

$$\mathbb{E}_2\{\Phi\} = \int_{\mathbf{R}^2} \phi(s,g)Rf_{0s}(g)\chi_{[0,+\infty)}(s)\mathrm{d}s\mathrm{d}g \ .$$

3rd component

$$\begin{aligned}\mathbb{E}_3\{\Phi\} = &\int_{\mathbf{R}^2} \phi(\tau,\gamma)Cf_{\tau\tau}(\gamma)\chi_{[0,+\infty)}(\tau)\mathrm{d}\tau\mathrm{d}\gamma \\ &+ \int_{\mathbf{R}^2} Cf_{\tau\tau}(\gamma)\chi_{[0,+\infty)}(\tau)\left[\int_{\mathbf{R}^2} Rf_{\tau s}(g)\chi_{[\tau,+\infty)}(s)\phi(s,g)\mathrm{d}g\mathrm{d}s\right]\mathrm{d}\tau\mathrm{d}\gamma \ .\end{aligned}$$

For the m.g.f. we use the Campbell's theorem, obtaining

1st component

$$M_1^{\Phi}(\theta) = \mathbb{E}_1\{\mathrm{e}^{\theta\phi(0,g)}\} = \int_{\mathbf{R}} \mathrm{e}^{\theta\phi(0,g)}f_{00}(g)\mathrm{d}g \ .$$

2nd component

$$M_2^{\Phi}(\theta) = \exp\left(\int_{\mathbf{R}^2}\left[e^{\theta\phi(s,g)} - 1\right] R f_{0s}(g)\chi_{[0,+\infty)}(s)\mathrm{d}s\mathrm{d}g\right)$$

3rd component

$$M_3^{\Phi}(\theta) = \exp\left(\int_0^{\infty}\int_{\mathbf{R}}\left[e^{\theta\phi(\tau,\gamma)+\int_\tau^\infty\int_{\mathbf{R}}[e^{\theta\phi(s,g)}-1]Rf_{\tau s}(g)\mathrm{d}g\mathrm{d}s} - 1\right] C f_{\tau\tau}(\gamma)\mathrm{d}\gamma\mathrm{d}\tau\right)$$

The m.g.f. for Φ is $M(\theta) = M_1(\theta)M_2(\theta)M_3(\theta)$. Thus, to fix ideas, let be

$$\Phi = \sum_{(\tau,\gamma)\in B}\phi(\tau,\gamma) =: \int_B \phi(\tau,\gamma)N(\mathrm{d}\tau\times\mathrm{d}\gamma)\ .$$

The n^{th} moment of Φ will be

$$\mathbb{E}\{\Phi\} = \left.\frac{\mathrm{d}^n M}{\mathrm{d}\theta^n}\right|_{\theta=0}\ .$$

Now it's only a matter of straightforward computations, which are omitted for lack of intrinsic interest.

1.5.7 **Energy bound with TR.** We finally can compute the bound. In *Remark* 1.5 on page 10 we show that the energy collected by a *full*-TR/*all*-RAKE system is

$$\begin{aligned}\mathcal{E} &= \mathcal{E}_s\frac{1}{\sum_{\ell=1}^{L}\alpha_\ell^2}\left[2\sum_{i=1}^{L}\sum_{j=1}^{L}\alpha_i^2\alpha_j^2 - \sum_{\ell=1}^{L}\alpha_\ell^4\right] = \mathcal{E}_s\frac{1}{\sum_{\ell=1}^{L}\alpha_\ell^2}\left[2\left(\sum_{i=1}^{L}\alpha_i^2\right)^2 - \sum_{\ell=1}^{L}\alpha_\ell^4\right] \\ &= \mathcal{E}_s\sum_{\ell=1}^{L}\alpha_\ell^2\left[2 - \frac{\sum_{\ell=1}^{L}\alpha_\ell^4}{\left(\sum_{\ell=1}^{L}\alpha_\ell^2\right)^2}\right]\end{aligned},$$

whereas with a *full*-TR/1-RAKE we collect

$$\mathcal{E} = \mathcal{E}_s\sum_{\ell=1}^{L}\alpha_\ell^2\ .$$

We define now a mean ratio as follows

$$\bar{\rho} := 2 - \frac{\mathbb{E}\left\{\sum_{\ell=1}^{L}\alpha_\ell^4\right\}}{\mathbb{E}\left\{\left(\sum_{\ell=1}^{L}\alpha_\ell^2\right)^2\right\}}\ .$$

The numerator is the first moment of sum of amplitudes fourth-powers, whereas the denominator is the second moment of sum of squares. Thus, the functions $\phi(\tau,\gamma)$ to be used are $\phi(\tau,\gamma) := \gamma^4\chi_{[0,+\infty)}(\tau)$ and $\phi(\tau,\gamma) := \gamma^2\chi_{[0,+\infty)}(\tau)$, respectively. We then find the m.g.f. for both cases and consequently the moments. The result is

$$\bar{\rho} = 2 - \frac{1}{1 + e^{-4\sigma^2}\left[(4C\tau_0)/(2+Rs_0) + 2Rs_0(1+2C\tau_0)\right]}\ .$$

Adopting the channel model parameters of Table 1.1 on page 10, the numerical result is $\bar{\rho} \simeq 1.85353$ that is in accordance with the simulation of Figure 1.4.

2

— ∘ —

ROBUSTNESS ANALYSIS

Summary

In this chapter we address the problem of robustness of the TR approach. We have seen in the previous chapter that, in absence of MUI, we can introduce a *full*-TR in combination with a 1-RAKE to obtain the same performance of an *all*-RAKE. In the following we assume a model for the perturbation and study its effect on both systems. We also show simulation results in more complex scenarios.

Our contribution

The main contribution concerns the study of the effect on BER of an error introduced in the TR pre-filter.

§2.1 Introduction

We model the perturbation as an additive zero-mean gaussian process $\xi(t)$ with variance σ_ξ^2. Both *full*-TR pre-filter (without the normalisation constant that assures the power constraint at transmitter) and *all*-RAKE receiver have an impulse response with energy equals to the channel gain, thus the pertubation may be regarded as the error occured during the channel estimation process; otherwise, it may be just considered as an unavoidable amplitude error of the filters, or a combination of the two.

§2.2 Absence of interference.

2.2.1 **Robustness of TR.** We focus on a system with *full*-TR and 1-RAKE. In absence of any error, we have already seen that the signal sent (during a signaling period) carrying the bit 1 is

$$x(t) = \frac{1}{\left[\sum_{\ell=1}^{L} \alpha_\ell^2\right]^{\frac{1}{2}}} \sum_{\ell=1}^{L} \alpha_\ell s_1(t+\tau_\ell) \ ,$$

where $\{\alpha_\ell\}$ and $\{\tau_\ell\}$ are the sets of channel amplitudes and delays, respectively. The signal received is $r = x * h + n = y + n$ and a 1-RAKE will take the following correlation metric

$$CM_1 = \left\langle y(t) + n(t), \sqrt{\sum_{\ell=1}^{L} \alpha_\ell^2 s_1(t)} \right\rangle .$$

The decision will be based on the sign of this correlation metric. Let us rewrite this in vectorial form, setting $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_L]^T$ as the vector of channel amplitudes:

$$CM_1 = \mathcal{E}_s |\boldsymbol{\alpha}|^2 + \nu \ , \quad \nu \sim \mathcal{N}(0, \sigma_n^2 \mathcal{E}_s |\boldsymbol{\alpha}|^2) .$$

In presence of perturbation, the pre-filter amplitudes are not longer $\boldsymbol{\alpha}$ but $\boldsymbol{\alpha} + \boldsymbol{\xi}$ and the signal sent is

$$x(t) = \frac{1}{\sqrt{\sum_{\ell=1}^{L} [\alpha_\ell + \xi(\tau_\ell)]^2}} \sum_{\ell=1}^{L} (\alpha_\ell + \xi(\tau_\ell))\, s(t + \tau_\ell) .$$

Note the different normalisation required to assure the power constraint. The 1-RAKE expects to find $|\alpha| s(t)$ as previous and the correlation metric will be

$$CM_1 = \langle y(t), |\boldsymbol{\alpha}| s(t) \rangle = \mathcal{E}_s \frac{\boldsymbol{\alpha}^T (\boldsymbol{\alpha} + \boldsymbol{\xi})}{|\boldsymbol{\alpha} + \boldsymbol{\xi}|} |\boldsymbol{\alpha}| + \nu \ , \quad \nu \sim \mathcal{N}\left(0, \sigma_n^2 \mathcal{E}_s |\boldsymbol{\alpha}|^2\right) .$$

In other words, the decision is now based on the sign of

$$\left[\sqrt{\mathcal{E}_s} \frac{\boldsymbol{\alpha}^T (\boldsymbol{\alpha} + \boldsymbol{\xi})}{|\boldsymbol{\alpha} + \boldsymbol{\xi}|} + n \right] \sqrt{\mathcal{E}_s} |\boldsymbol{\alpha}| .$$

As might be expected, with an evanescent perturbation, $\xi \to 0$, we obtain the previous result. Our goal is find the PDF of the perturbed term, that will reveal us some insight into the effects that it produces.

PDF OF PERTURBED TERM. We want to find the PDF of

$$\Upsilon := \frac{\boldsymbol{\alpha}^T (\boldsymbol{\alpha} + \boldsymbol{\xi})}{|\boldsymbol{\alpha} + \boldsymbol{\xi}|} ,$$

given $\boldsymbol{\alpha}$ and assuming $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma_\xi^2 \mathbf{I})$, that is, a vector of L i.i.d. samples of the process $\xi(t)$.

We start considering

$$\zeta := \frac{\Upsilon}{|\boldsymbol{\alpha}|} = \frac{\boldsymbol{\alpha}^T (\boldsymbol{\alpha} + \boldsymbol{\xi})}{|\boldsymbol{\alpha}||\boldsymbol{\alpha} + \boldsymbol{\xi}|} ,$$

that has a scaled PDF with respect to Υ.

The key point is the application of an orthogonal transformation that drastically simplifies the ratio without changing its value. We can think of $\boldsymbol{\alpha}$ as the coordinates of a vector of $\mathbf{R}^L$ with respect to the canonical base $\mathcal{B}$. We can find another orthonormal base $\mathcal{B}'$ such that only the first coordinate of the vector is non-null. This is feasible via Gram-Schimdt orthogonalization, for example. We call the matrix of the basis changing $\mathbf{P}$. It is a well-known result that $\mathbf{P}$ is *orthogonal*, $\mathbf{P}^{-1} = \mathbf{P}^T$. As a consequence, $\mathbf{P}$ realize such a kind of *isometry*, that is, it does not change the norm of the transformed vector: if $\boldsymbol{\xi}' = \mathbf{P}\boldsymbol{\xi}$, then $|\boldsymbol{\xi}'| = |\boldsymbol{\xi}|$. Under that operator, $\boldsymbol{\alpha}$ become

$\boldsymbol{\alpha}' := \mathbf{P}\boldsymbol{\alpha} = [\alpha'_1, 0, \ldots, 0]^T$. The last ingredient is the following: if $\boldsymbol{\xi}$ is a gaussian r.v. with scalar covariance matrix $\mathbf{C} := \sigma_\xi \mathbf{I}$, then $\boldsymbol{\xi}'$ is still gaussian with the same covariance matrix. This is straightforward: in fact, a linear transformation of a gaussian r.v. yields still a gaussian r.v. and its covariance matrix is

$$\mathbb{E}\{\boldsymbol{\xi}'\boldsymbol{\xi}'^T\} = \mathbb{E}\{(\mathbf{P}\boldsymbol{\xi})(\mathbf{P}\boldsymbol{\xi})^T\} = \mathbf{P}\sigma_\xi^2 \mathbf{I}\mathbf{P}^T = \sigma_\xi^2 \mathbf{I}.$$

Hereinafter we do not longer write vectors and matrices in boldface. We can write the ratio as follows:

$$\zeta = \frac{\alpha^T(\alpha+\xi)}{|\alpha||\alpha+\xi|} = \frac{\alpha^T P^T P(\alpha+\xi)}{|P\alpha||P(\alpha+\xi)|} = \frac{\alpha'^T(\alpha'+\xi')}{|\alpha'||\alpha'+\xi'|} = a'^T \frac{\alpha'+\xi'}{|\alpha'+\xi'|} = \frac{\alpha'_1+\xi'_1}{|\alpha'+\xi'|} ,$$

having set $a' := \alpha'/|\alpha'| = [1, 0, \ldots, 0]^T$. Now

$$|\alpha'+\xi'| = \sqrt{(\alpha'_1+\xi'_1)^2 + \xi_2^2 + \cdots + \xi_L^2} = \sqrt{(\alpha'_1+\xi'_1)^2 + |\xi'_{-1}|^2} ,$$

where $\xi'_{-1} := [\xi'_2, \ldots, \xi'_L]^T$ is the vector ξ' without the first element. It turns out that ξ'_k are i.i.d. and that this property is inherited by $(\alpha'_1 + \xi'_1)$ and $|\xi'_{-1}|^2$. If we call

$$x := \frac{\alpha'_1+\xi'_1}{\sigma_\xi} \sim \mathcal{N}(\alpha'_1/\sigma_\xi, 1)$$

and

$$y := \frac{|\xi'_{-1}|}{\sigma_\xi} \sim \chi_{L-1} ,$$

we can write ζ as follows

$$\zeta = \frac{x}{\sqrt{x^2+y^2}} = \frac{x/y}{\sqrt{1+(x/y)^2}} ,$$

so the ratio depends only on $v := x/y$. It is actually useful consider the ratio

$$t := \frac{x}{y/\sqrt{\nu}} , \qquad \nu := L-1 ,$$

because it has a known distribution that is the non-central Student $\mathcal{T}$-distribution with ν degrees of freedom and non-central parameter $\delta := \alpha'_1/\sigma_\xi$. We call it $\mathcal{T}'_\nu(\delta)$. Explicitly, it has the following canonical form

$$p_{\mathcal{T}'_\nu(\delta)}(t) = \frac{2^\nu \mathrm{e}^{-\delta^2/2}\nu^{1+\nu/2}}{\pi(t^2+\nu)^{\frac{1+\nu}{2}}}\Gamma\left(\frac{1+\nu}{2}\right) H_{-1-\nu}\left(-\frac{\delta}{\sqrt{2}}\frac{t}{\sqrt{t^2+\nu}}\right) ,$$

where $H_n(x)$ is the Hermite polynomial[1].

We obtain the PDF of ζ directly:

$$\zeta = \frac{t}{\sqrt{\nu+t^2}} \implies p_\zeta(z) = p_{\mathcal{T}'_\nu(\delta)}\left(\sqrt{\nu}\frac{z}{\sqrt{1-z^2}}\right)\frac{\sqrt{\nu}}{\sqrt{(1-z^2)^3}} , \quad |z| \le 1 .$$

Note 2.1. We have implicitly assumed w.l.o.g. that $\alpha'_1 > 0$. In any case

$$\alpha_1'^2 = |\alpha|^2 = \sum_{\ell=1}^{L} \alpha_\ell^2 ,$$

and $\alpha_1'^2/\sigma_\xi^2$ may be viewed, at the same extent of $\mathcal{E}_b/N_0$, as a ratio between powers (of filter impulse response and perturbation, respectively). ◊

[1] Among the definitions of Hermite polynomial, we assume the one that see it satisfying the following ODE: $y'' - 2xy' + 2ny = 0$.

BEP WITH CHANNEL AND PERTURBATION. Once we have the PDF of ζ, we can find a closed formula for the BEP. The decision is based on the sign of

$$\sqrt{\mathcal{E}_s}|\alpha|\zeta + n \ .$$

Having set $u := \sqrt{\mathcal{E}_s}|\alpha|\zeta$, the BEP (given the channel, that is, given $|\alpha|$, and given the perturbation, that is, given ξ) is

$$P_{\mathrm{e}}|\alpha,\zeta = Q\left(\frac{u}{\sigma_n}\right) = Q\left(\frac{\sqrt{\mathcal{E}_s}|\alpha|}{\sqrt{\sigma_n^2}}\zeta\right) = Q\left(\sqrt{2\frac{\mathcal{E}_s|\alpha|^2}{N_0}\zeta}\right) = Q\left(\sqrt{(1-\rho)\gamma_b\zeta}\right) \ ,$$

that comprises the PAM and PPM cases ($\rho = -1$ and $\rho = 0$, respectively). We call $a := |\alpha|$ and $f_a(\cdot)$ its PDF. The whole bit error probability is

$$P_{\mathrm{e}} = \int_0^\infty \int_{-1}^1 Q\left(\sqrt{2\frac{\mathcal{E}_s a^2}{N_0}z}\right) f_a(a) p_\zeta(z)\ \mathrm{d}z\mathrm{d}a$$

Note 2.2. The integration order *can not* be changed because ζ depends actually on $|\alpha|$. $\Diamond$

This formula shows that there is a loss in performance with respect to a system based on a RAKE receiver without TR in trasmission, or –that is the same– an unperturbed TR system, that would have

$$P_{\mathrm{e}}|a = Q\left(\sqrt{(1-\rho)\gamma_b}\right) \ .$$

The effect of ζ in decreasing the argument of Q is definitely to rise (statistically) the BEP.

TR FLOOR. The most visible effect introduced by perturbed TRs is however the presence of a BEP floor. This is unavoidable if we last a coherent detector. Let us proceed in an approximate fashion. The detector is mistaken if

$$\sqrt{\mathcal{E}_s}\frac{\alpha^T(\alpha+\xi)}{|\alpha+\xi|} + n < 0 \ ,$$

given that it was sent the bit 1. There is a probability that $|n|$ is big enough to be responsible for the wrong decision, but however for $\mathcal{E}_s|\alpha|^2 \gg |n|$ we can imagine that the noise term is negligible. Thus the error occurs iff

$$\sqrt{\mathcal{E}_s}\frac{\alpha^T(\alpha+\xi)}{|\alpha+\xi|} < 0 \iff \alpha^T(\alpha+\xi) < 0$$

and we can apply as previous the matrix P to get a simpler form of this product, leading to

$$\alpha_1'^2 + \alpha_1'\xi_1' < 0 \ .$$

The error probabily is thus

$$P_{\mathrm{e}}^{\mathrm{floor}} \simeq Q\left(\sqrt{\frac{|\alpha|^2}{\sigma_\xi^2}}\right) = Q\left(\frac{|\alpha|}{\sigma_\xi}\right) .$$

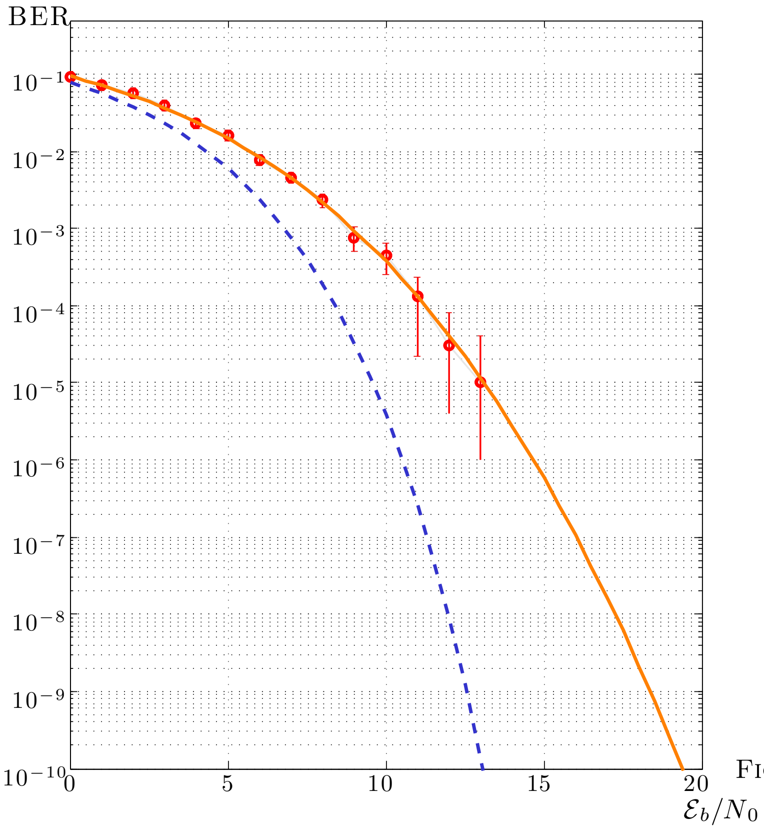


FIGURE 2.1: BER with AWGN channel (dashed, from theory) and multipath channel (solid, from theory, and circles, from simulations).

ENERGY LOSS. In Chapter 1 we show that TR is the optimum pre-coder in a system with a 1-finger RAKE, thus with respect to the criterion of maximization of peak of the received signal. This is no longer true in presence of a perturbation, hence we may guess that peak-energy is reduced. To prove this, let us find the PDF of ζ^2. In fact

$$\mathcal{E}_{\text{peak}} = \mathcal{E}_s|\alpha|^2\zeta^2$$

and ζ^2 may be viewed as the loss factor. We recall that

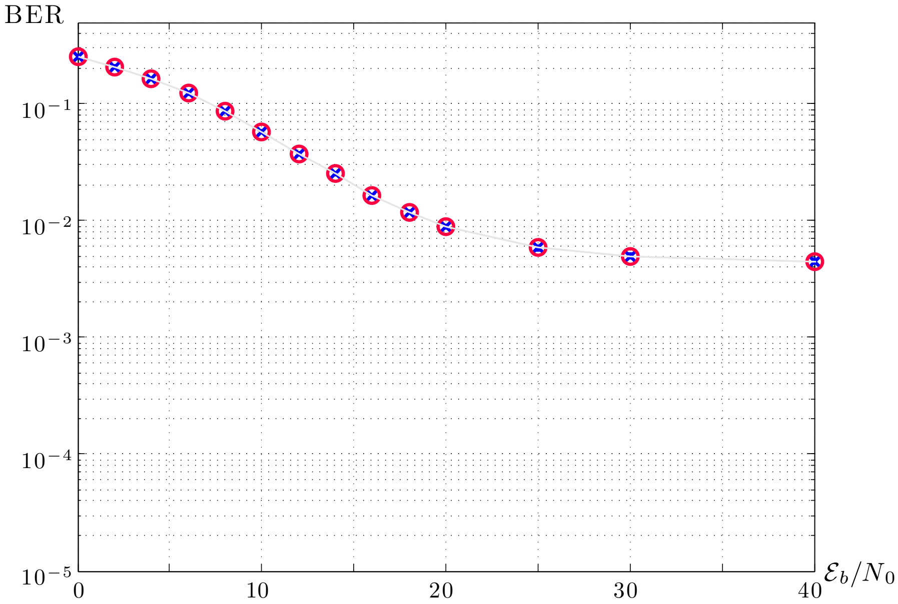
$$\zeta = \frac{\alpha_1' + \xi_1'}{|\alpha' + \xi'|} \ .$$

We can expand the expression as follows:

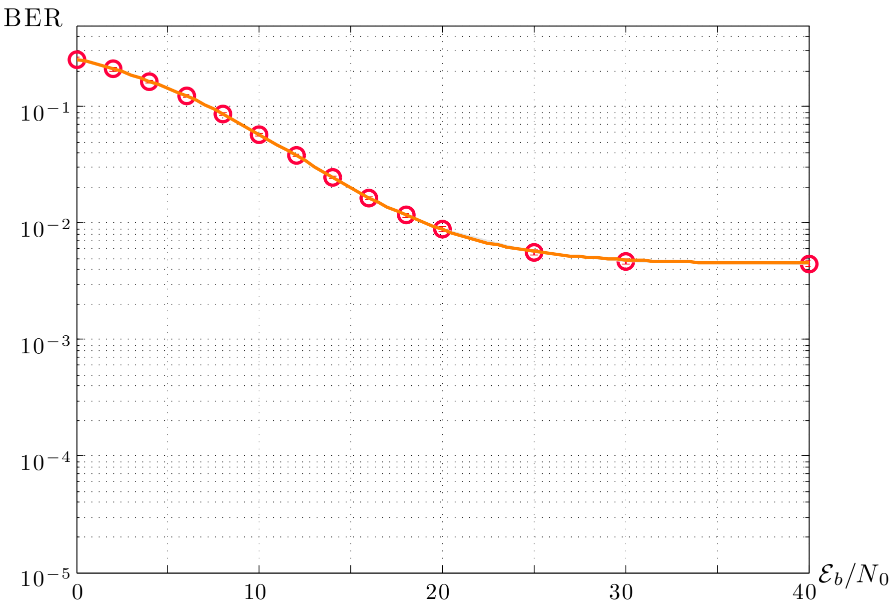
$$\zeta^2 = 1 - \frac{|\xi_{-1}'|^2}{(\alpha_1' + \xi_1')^2 + |\xi_1'|^2} = 1 - \frac{1}{1 + \dfrac{(\alpha_1' + \xi_{-1}')^2}{|\xi_{-1}'|^2}} \ .$$

As previous, it is useful to rewrite this as follows

$$\zeta^2 = 1 - \frac{1}{1 + \frac{1}{\nu}\dfrac{\left(\frac{\alpha_1'}{\sigma_\xi} + \frac{\xi_1'}{\sigma_\xi}\right)^2}{\frac{1}{\nu}\left|\frac{\xi_{-1}'}{\sigma_\xi}\right|^2}} \ .$$



(A) *Comparison between perturbed TR (circle) and RAKE (cross).*



(B) *Comparison between theory (solid) and simulations (circle).*

FIGURE 2.2: BER of TR and RAKE, comparisons.

In fact, we can now trace back the PDF to a known distribution. We have

$$\frac{\alpha_1'}{\sigma_\xi} + \frac{\xi_1'}{\sigma_\xi} \sim \mathcal{N}(\alpha_1'/\sigma_\xi, 1) \implies \left(\frac{\alpha_1'}{\sigma_\xi} + \frac{\xi_1'}{\sigma_\xi}\right)^2 \sim \chi_1'(\alpha_1'^2/\sigma_\xi^2)$$

and

$$\left|\frac{\xi_{-1}'}{\sigma_\xi}\right|^2 \sim \chi_\nu^2 \, .$$

It is known as (non-central) F (ratio) *distribution* the PDF that describes the quotient (or ratio) of two independent chi-square distribution. To be precise, if

$$X \sim \chi'_n(\lambda) \ , \quad Y \sim \chi'_m(\eta) \ ,$$

then

$$Z = \frac{X/n}{Y/m}$$

has a *doubly non-central F ratio distribution* of orders (n, m) and non-centrality parameters (λ, η),

$$Z \sim F'_{n,m}(\lambda, \eta).$$

In our case

$$\Psi := \frac{\left(\frac{\alpha'_1}{\sigma_\xi} + \frac{\xi'_1}{\sigma_\xi}\right)^2}{\frac{1}{\nu}\left|\frac{\xi'_{-1}}{\sigma_\xi}\right|^2} \sim F'_{1,\nu}(\alpha_1'^2/\sigma_\xi^2).$$

Thus the PDF of

$$\zeta^2 = 1 - \frac{1}{1 + \frac{1}{\nu}\Psi}$$

is the following:

$$p_{\zeta^2}(x) = \mathrm{e}^{-\lambda/2} \frac{(1-x)^{\frac{\nu}{2}-1}}{\sqrt{x} B\left(\frac{1}{2}, \frac{\nu}{2}\right)} \, {}_1F_1\left(\frac{\nu+1}{2}; \frac{1}{2}; \frac{\lambda}{2}x\right) \ , \quad x \in [0, 1] \ , \quad \lambda = \frac{\alpha_1^2}{\sigma_\xi^2} \ .$$

We call $\varrho := |\zeta| = \sqrt{\zeta^2}$ and show in FIGURE 2.3 on the following page histograms from simulations and the theoretical PDF. As it is visible, there is a loss in collectable energy at the receiver: as $\lambda \to \infty$, i.e. $\sigma_\xi \to 0$, the PDF tends to $\delta(x-1)$ and the loss vanishes, whereas the greater is σ_ξ^2, the greater is the mean loss.

2.2.2 **Robustness of RAKE.** We model a perturbation on RAKE fingers as follows

$$\hat{\alpha}_k := \alpha_k + \xi_k \ , \qquad \xi_k \sim \mathcal{N}(0, \sigma_\xi^2) \ , \quad k = 1, \ldots, L \ .$$

The k^{th} correlator in the receiver takes the projection of the received signal with the k^{th} path of the channel

$$\left\langle \sum_{\ell=1}^{L} \alpha_\ell s(t - \tau_\ell) + n(t), \hat{\alpha}_k s(t - \tau_k) \right\rangle = \langle \alpha_k s(t - \tau_k) + n(t), \hat{\alpha}_k s(t - \tau_k) \rangle =$$

that yields

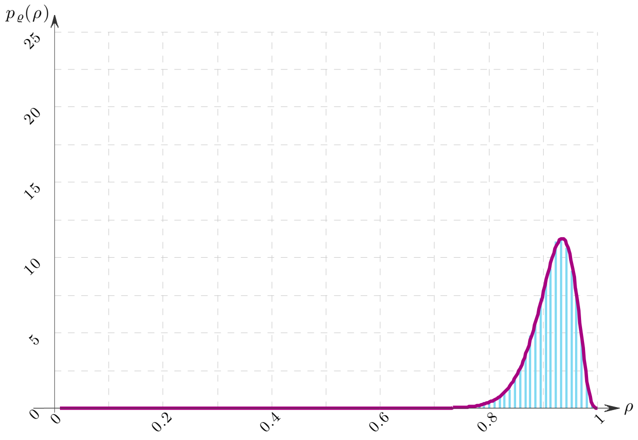
$$= \alpha_k^2 \mathcal{E}_s + \alpha_k \xi_k \mathcal{E}_s + n_k \alpha_k \sqrt{\mathcal{E}_s} + n_k \xi_k \sqrt{\mathcal{E}_s} \ .$$

An *all*-RAKE, that is, a Maximum-Ratio Combiner, integrates all pulses and gives

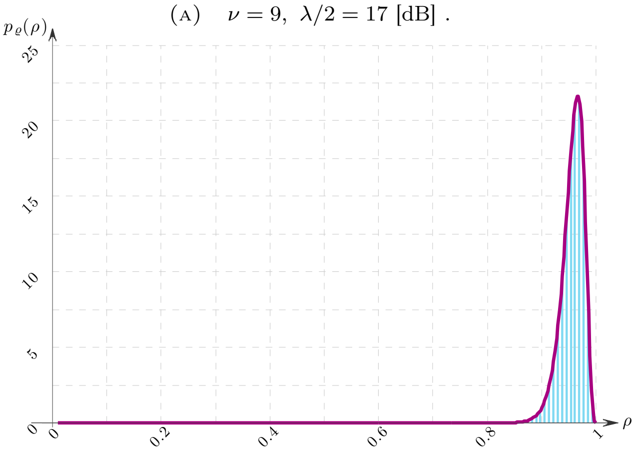
$$CM_1 = \sum_{k=1}^{L} (\alpha_k^2 \mathcal{E}_s + \alpha_k \xi_k \mathcal{E}_s + n_k \alpha_k \sqrt{\mathcal{E}_s} + n_k \xi_k \sqrt{\mathcal{E}_s})$$

as the decision variable.

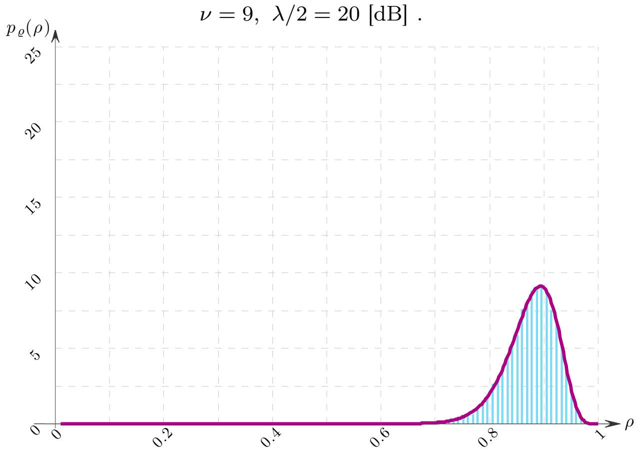
FIGURE 2.3: Energy carried by expected equivalent channel peak with respect to its maximum.



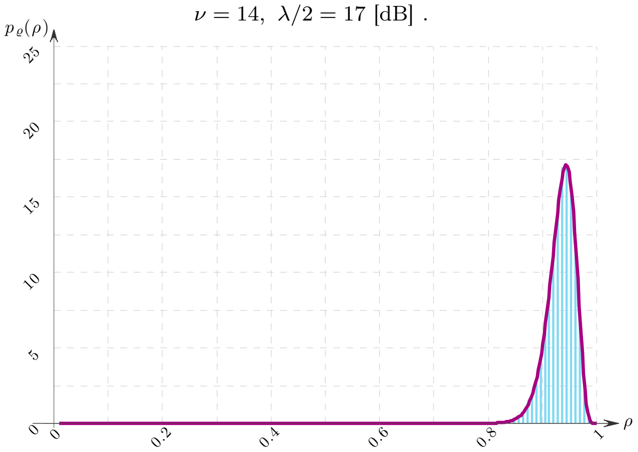
(A) $\nu = 9,\ \lambda/2 = 17$ [dB] .



$\nu = 9,\ \lambda/2 = 20$ [dB] .



$\nu = 14,\ \lambda/2 = 17$ [dB] .



$\nu = 14,\ \lambda/2 = 20$ [dB] .

Note 2.3. For an evanescent perturbation, $\xi_k \to 0$, the last expression reduces to the usual problem of a signal in noise:

$$\sum_{k=1}^{L}(\alpha_k^2\mathcal{E}_s + n_k\alpha_k\sqrt{\mathcal{E}_s}) = \mathcal{E}_s|\alpha|^2 + \alpha^T\sqrt{\mathcal{E}_s}n \ . \qquad \Diamond$$

Note 2.4. The three terms in $\alpha_k\xi_k\mathcal{E}_s + n_k\alpha_k\sqrt{\mathcal{E}_s} + n_k\xi_k\sqrt{\mathcal{E}_s}$ are *not* independent. $\Diamond$ We may drastically reduce the complexity of this problem neglecting the cross noise-perturbation term $n_k\xi_k$. The decision is based on the sign of

$$\sqrt{\mathcal{E}_s}|\alpha|^2 + \sqrt{\mathcal{E}_s}\alpha^T\xi + \alpha^T n + \xi^T n \simeq \sqrt{\mathcal{E}_s}|\alpha|^2 + \sqrt{\mathcal{E}_s}\alpha^T\xi + \alpha^T n \ .$$

The last two terms are independent, thus

$$\sqrt{\mathcal{E}_s}\alpha^T\xi + \alpha^T n \sim \mathcal{N}(0, \mathcal{E}_s|\alpha|^2\sigma_\xi^2 + |\alpha|^2\sigma_n^2) \ .$$

The BEP is then

$$Q\left(\sqrt{\frac{\mathcal{E}_s|\alpha|^2}{\mathcal{E}_s\sigma_\xi^2 + \sigma_n^2}}\right) ,$$

that, for high $\mathcal{E}_b/N_0$, reduces to

$$Q\left(\frac{|\alpha|}{\sigma_\xi}\right) \ .$$

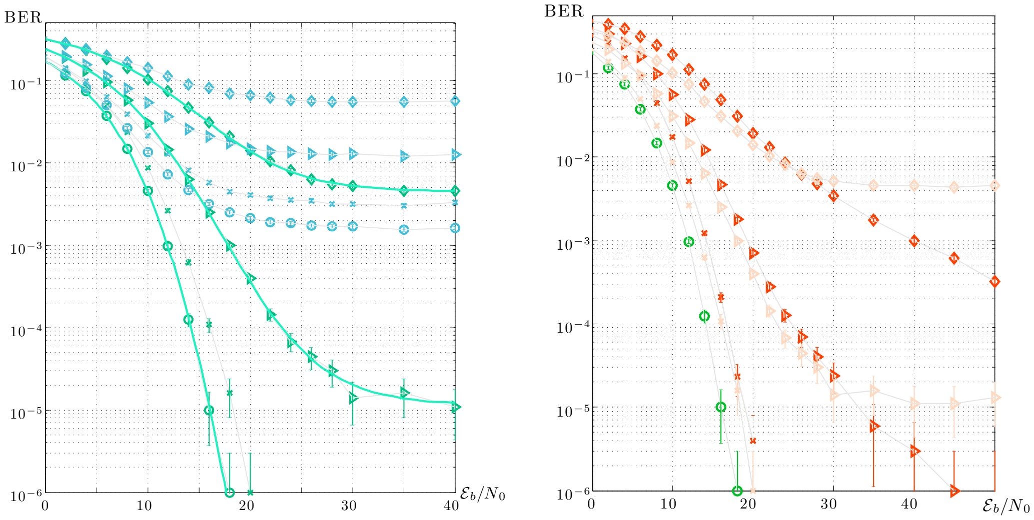
This result shows that the BEP floor is not peculiar of TR. Furthermore, its value, given a perturbation with equal variance, is the same of TR.

Note 2.5. We may show that neglecting the cross-noise term is actually conservative, that is, the true BEP is lower than the one predicted. However, the floor is the same. $\Diamond$

2.2.3 **Non-coherent detection.** An insight into the reason for the existence of BEP floor with TR has been already sketched (see § 2.2 on page 25). There are several ways to improve performance in terms of BEP: (1) adopt a coding technique (e.g. a repetition code) allow to reduce the BEP, at the expense of bit-rate, by a power equals to repetition order and open the opportunity of using an ML detector that partially exploits the MUI structure, leading to a further gain; (2) employing a non-coherent detector allow to break the perturbation floor at the expense of greater BEP for small $\mathcal{E}_b/N_0$. While in the first case the BEP floors due to MUI and perturbation are both reduced, in the latter the perturbation floor does not exist anymore, but the MUI floor still remains.

§2.3 Presence of interference.

2.3.1 **Frequency-selective channel as flat multi-channel and MUI.** It is well known that a slowly-fading frequency-selective channel can be viewed as a flat multi-channel. This perspective basically relies on a natural decomposition of the channel impulse response (= the *mask* within the correlator) into its constitutive pulses (aptly delayed and scaled). This point of view greatly simplifies the intuitive understanding of following statements and remarks.



(A) *PPM with perturbations, with (light blue) and without (cyan) MUI.*

(B) *Comparison between non-coherent (red) and coherent (pink) PPM .*

FIGURE 2.4: Non-coherent detector (circle: no perturbation, cross: $\lambda/2 = 20$ [dB], left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB], diamond: $\lambda/2 = 7$ [dB]).

We denote the channel by

$$h(t) := \sum_{\ell=1}^{L} \alpha_\ell \delta(t - \tau_\ell) \ .$$

The received signal in the signaling period $[0, T_{\mathrm{F}})$ is

$$r(t) = y(t) + n(t) + \sum_{q=1}^{Q} y^q(t) \ ,$$

where $y(t)$ is the useful signal, $n(t)$ is the WGN process and the last term is the MUI. In general we have, for the reference user

$$y(t) = \sqrt{\mathcal{E}_s} \sum_{\ell=1}^{L} \alpha_\ell s(t - \tau_\ell)$$

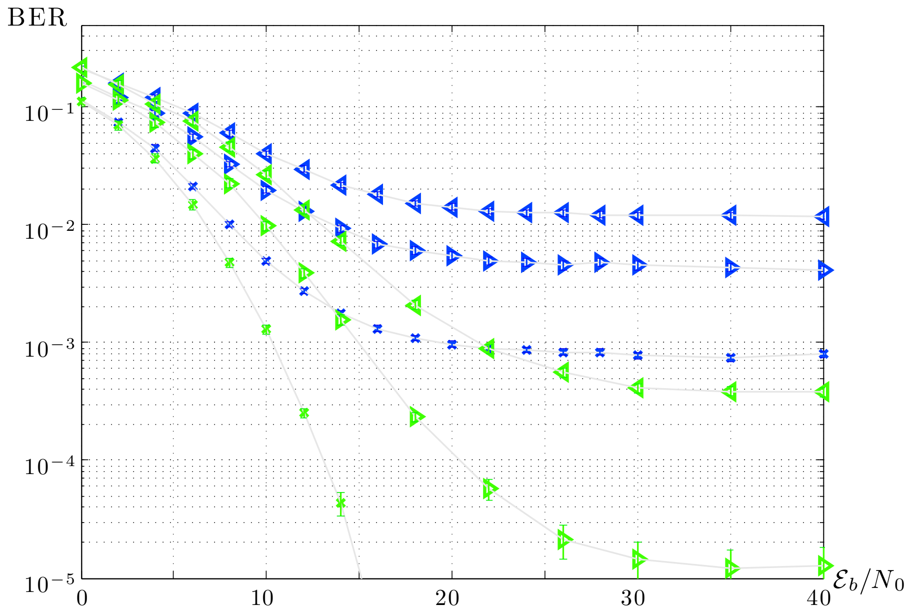
and for the q^{th} interfering user

$$y^q(t) = \sqrt{\mathcal{E}_s^q} \sum_{\ell^q=1}^{L^q} \alpha_\ell^q s^q(t - \tau_\ell^q - \theta^q) \ , \quad \theta^q \sim \mathcal{U}[0, T_{\mathrm{F}}) \ .$$

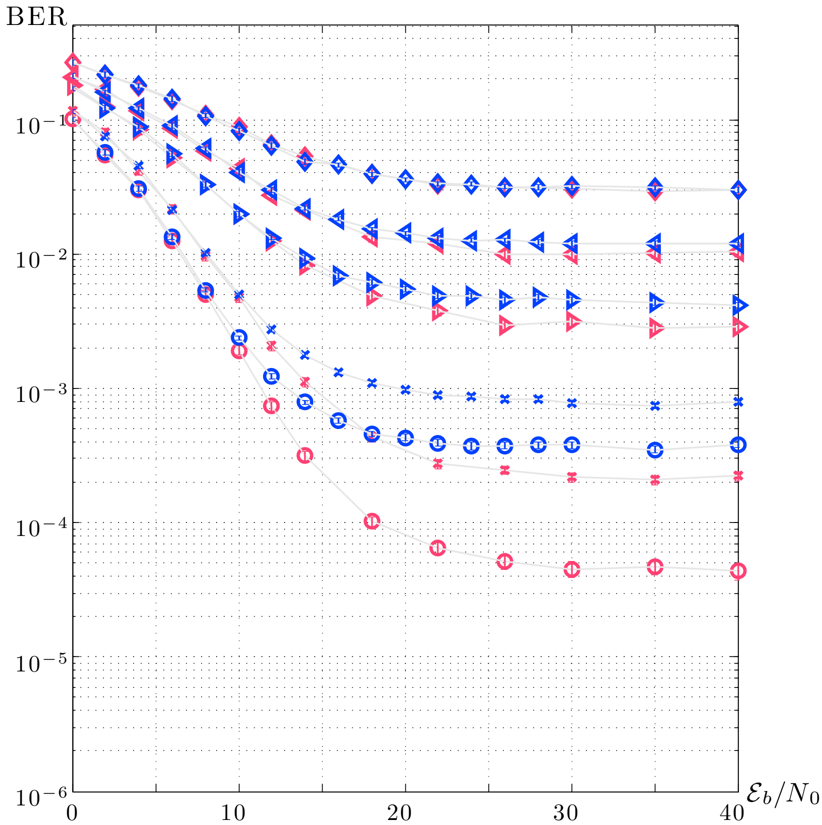
The correlator computes

$$\langle r(t), y(t) \rangle$$

for PAM, whereas for PPM have to consider also $\langle r(t), y(t - \delta) \rangle$, being δ the PPM-shift. For the sake of simplicity, we proceed with PAM, but with few changes we can obtain an analog PPM version.



(A) *TR with (blue) and without (green) MUI.*



(B) *TR and RAKE comparison.*

FIGURE 2.5: BER of TR and RAKE, comparisons (cross: $\lambda/2 = 20$ [dB], left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB]).

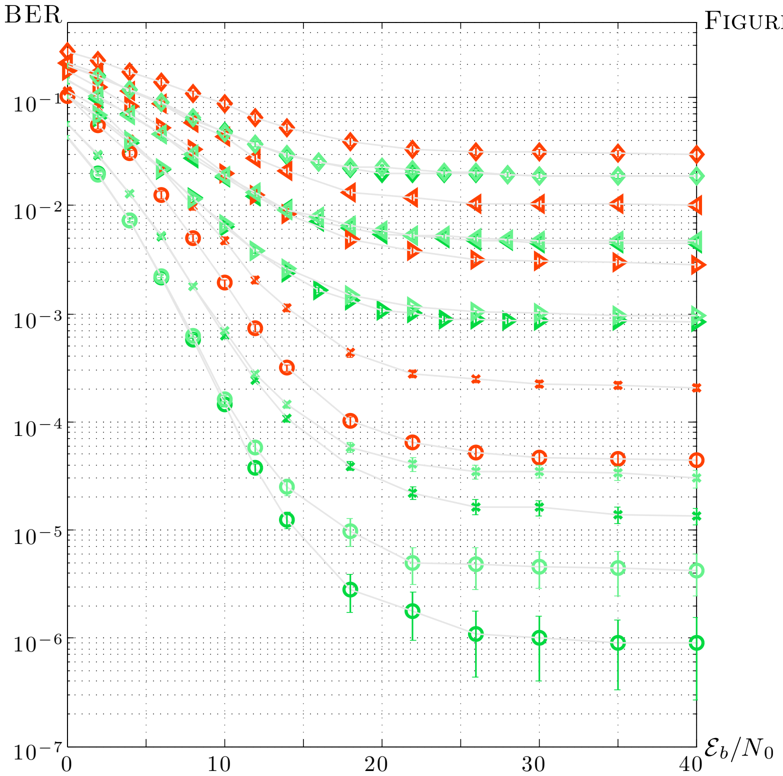


FIGURE 2.6: BER of RAKE with MUI (red) and TR/RAKE (with MUI: light green; with weak MUI: dark green) and perturbations (circle: no perturbation, cross: $\lambda/2 = 20$ [dB], left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB], diamond: $\lambda/2 = 7$ [dB]).

The correlated signal can be written in analogy to $r(t)$ as the sum of three terms:

$$CM = \psi + \nu + \zeta \ .$$

Moreover, each term can be viewed as the sum of L terms. For example

$$\psi := \left\langle y(t), \sum_{\ell=1}^{L} \alpha_\ell s(t-\tau_\ell) \right\rangle = \sum_{\ell=1}^{L} \left\langle y(t), \sqrt{\mathcal{E}_s}\alpha_\ell s(t-\tau_\ell) \right\rangle = \sum_{\ell=1}^{L} \psi_\ell \ .$$

We have to think of $s(t)$ as the basic pulse of IR modulations, e.g. a Scholtz-like pulse; hence, the ℓ^{th} correlator acts in a finite interval of τ_ℓ such as $[\tau_\ell, \tau_\ell + T_{\text{M}})$, being T_{M} the duration of the pulse:

$$\psi_\ell := \left\langle y(t), \sqrt{\mathcal{E}_s}\alpha_\ell s(t-\tau_\ell) \right\rangle = \left\langle \sqrt{\mathcal{E}_s}\alpha_\ell s(t-\tau_\ell), \sqrt{\mathcal{E}_s}\alpha_\ell s(t-\tau_\ell) \right\rangle = \mathcal{E}_s \alpha_\ell^2 \ .$$

The noise term is trivial, so we leave it out, whilst the MUI term is very attractive. The generic user q is viewed by the ℓ^{th} correlator as

$$\zeta_\ell^q := \left\langle y^q(t), \sqrt{\mathcal{E}_s}\alpha_\ell s(t-\tau_\ell) \right\rangle = \sqrt{\mathcal{E}_s}\alpha_\ell \left\langle y^q(t), s(t-\tau_\ell) \right\rangle$$

and considering equiprobable signs of channel amplitudes and asynchronous interference, we have

$$\zeta_\ell^q = \sqrt{\mathcal{E}_s}\alpha_\ell \sum_{\ell^q=1}^{L^q} \alpha_\ell^q \left\langle s^q(t-\tau_\ell^q-\theta^q), s(t-\tau_\ell) \right\rangle \doteq \sqrt{\mathcal{E}_s}\alpha_\ell \sum_{\ell^q=1}^{L^q} \alpha_\ell^q R_{ss}(\tau_\ell-\tau_\ell^q-\theta^q)$$

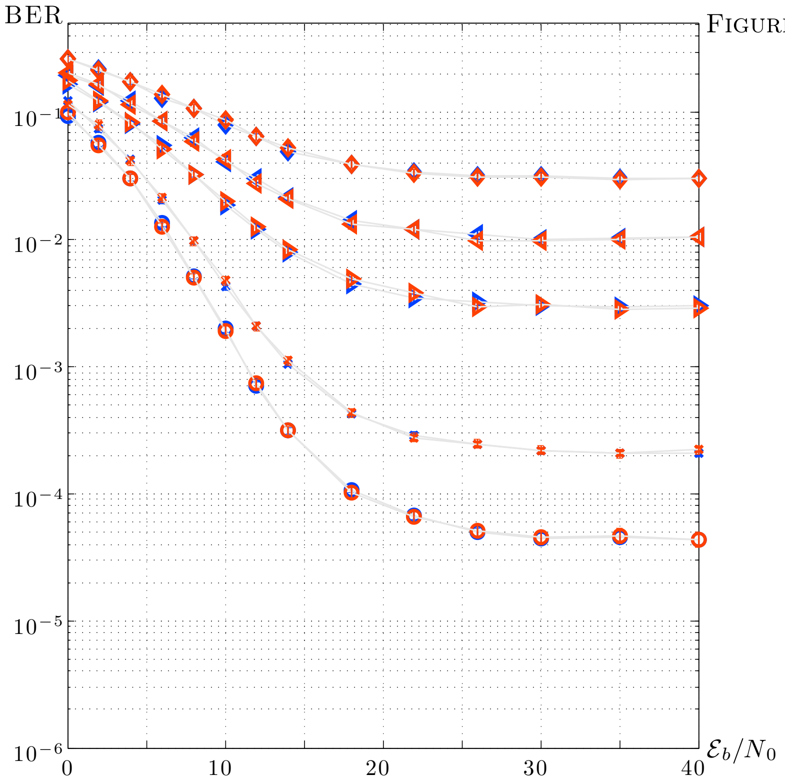


FIGURE 2.7: BER of RAKE (red) with MUI and TR (blue) with weak MUI (circle: no perturbation, cross: $\lambda/2 = 20$ [dB], left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB], diamond: $\lambda/2 = 7$ [dB]).

where the last equality is statistical (the second and third terms have the same PDF). Therefore, the whole receiver see the q^{th} interference as

$$\zeta^q := \sum_{\ell=1}^{L} \zeta_\ell^q \ .$$

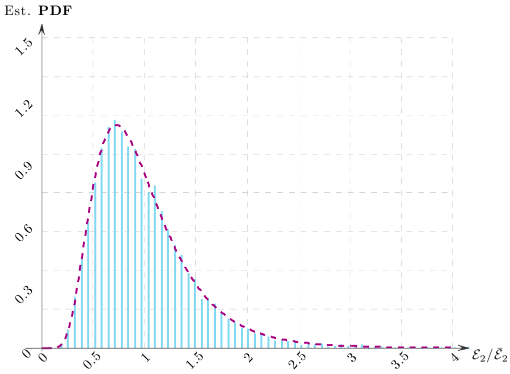
Finally, in presence of Q interferers, we have

$$\zeta = \sum_{q=1}^{Q} \zeta^q \ .$$

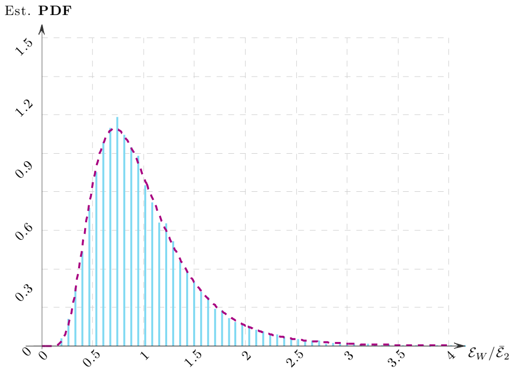
REFERENCE USER WITH TR. Introduction of TR results in replacing y with a scaled version of R_{yy}. To be precise, let us write $y(t) := \sqrt{\mathcal{E}_s}\eta(t)$. All expressions seen so far continue to be valid with

$$\eta(t) \mapsto \frac{1}{|\alpha|} R_{\eta\eta}(t) \ .$$

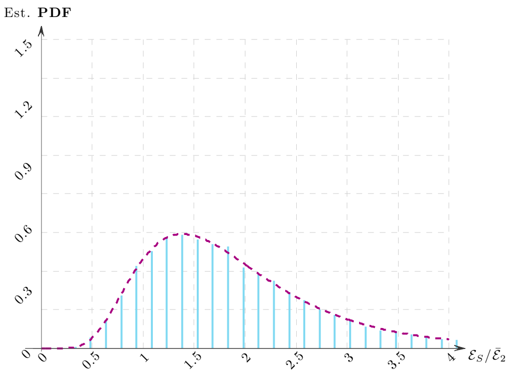
This expression shows that the received signal with TR is wider and with a peak. In Chapter 1 it is shown that this signal carries more energy than the previous one, the amount of which depending on channel parameters. An elementary upper-bound tells us that, for the peculiar form of UWB channels, we can not extract more than twice the energy. More than half of this energy is carried by the peak, that we call here *main pulse*. The rest of the energy is carried by other pulses, that we call *side pulses*. From this simplified description it is clear that we have reduced the average power of the received signal, mainly because of the presence of side pulses, that have at most half the power of the received signal without TR.



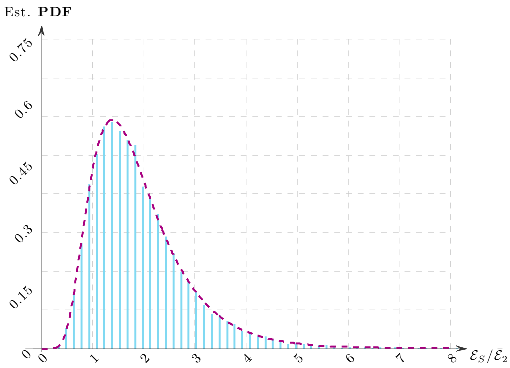
(A) *Without TR.*



(B) *With TR, independent channels.*



(C) *With TR, correlated channels.*



(D) *With TR, correlated channels (different scale).*

FIGURE 2.8: Energy of interfering signal during a signaling time (= time frame).

INTERFERING USER WITH TR. If TR is introduced by interfering users, we have two very different scenarios to face with. To fix ideas, let us consider that interfering user q is actually communicating with another user q'. Let be h_1 the channel between q and reference receiver, and h_2 between q and q'.

The received interference at reference receiver will be very different depending on the correlation between h_1 and h_2.

If they are independent, the received signal is the cross-correlation between two independent signals: it does not show any peak and, for the peculiar form of UWB channels, it turns out to have almost the same energy of the signal sent without TR (see (A) and (B) in FIGURE 2.8).

This is no longer true if they are *not* independent: the worst case occurs when they coincide and the interference is proportional to the autocorrelation of h_1. This is the worst case because (1) it shows a peak of interference that would require a specific design of the reference receiver, and (2) it has (= it interferes with) the maximum energy (see FIGURE 2.8). We call these two cases of interference respectively *weak* and *strong*.

Note 2.6. We emphasize that, in the first case, the average power of one interfering signal is decreased but its energy has remained the same: this implies that also the average interfering power in a frame has not changed.

PROS AND CONS. Let us summarize the effects of the introduction of TR on the reference user and on other users.

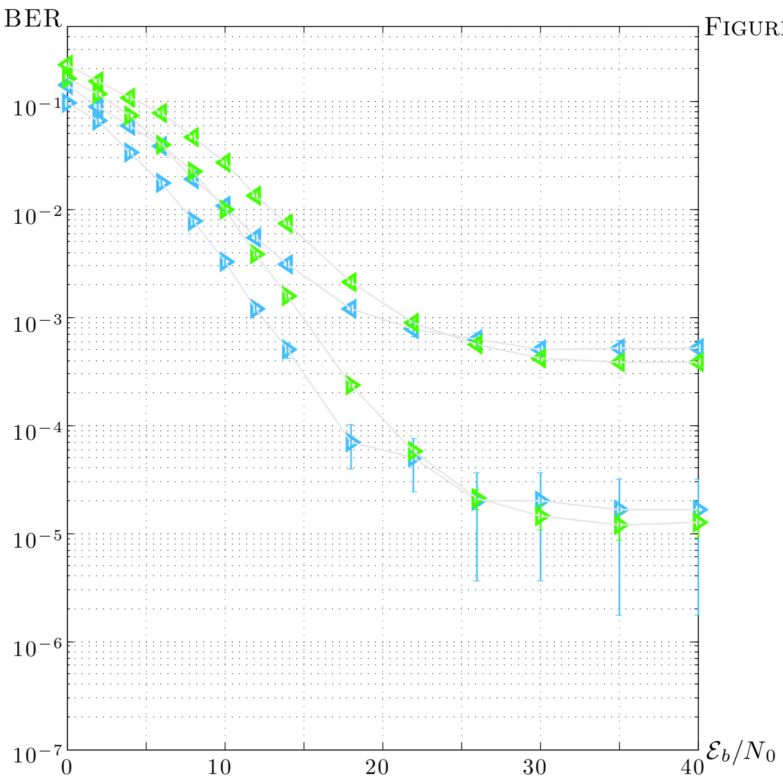


FIGURE 2.9: TR (green) and TR/RAKE (cyan) *without interference* (left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB]).

PROS of TR

- TR offers the possibility, in presence of noise but without interference, of using a RAKE with only 1 finger to achieve the same BER of a system using an *all*-RAKE without TR;
- TR offers the possibility of outperforming an *all*-RAKE system, because it increases the energy that is potentially detectable at the receiver;

CONS of TR

- TR does *not* offer the same performance of an *all*-RAKE in presence of interference, resulting in higher BER, due to the MUI internal structure;
- TR, if perturbed, produces a BER floor: this is avoidable with non-coherent receivers, but this option is viable in practice only if the MUI is absent, unless using a coding technique;

CONSEQUENCES. Let us draw a few consequences and show simulation results.

Let us start with a scenario without interference: put the TR pre-filter, remaining with a 1-finger RAKE at the receiver, does not change the BER with respect to a system with an *all*-RAKE without TR, even with perturbations (see FIGURE (A) 2.2 on page 26). As expected, to improve the BER we may use both TR *and* RAKE, but this is no longer true with interference: as a matter of fact, the BER floor with interference is higher if we use a large number of fingers in RAKE (see FIGURE 2.9).

This phenomenon is deeply mitigated in presence of interference. As a matter of fact, RAKE outperforms TR and so adoption of TR yields to a loss (see FIGURE 2.5

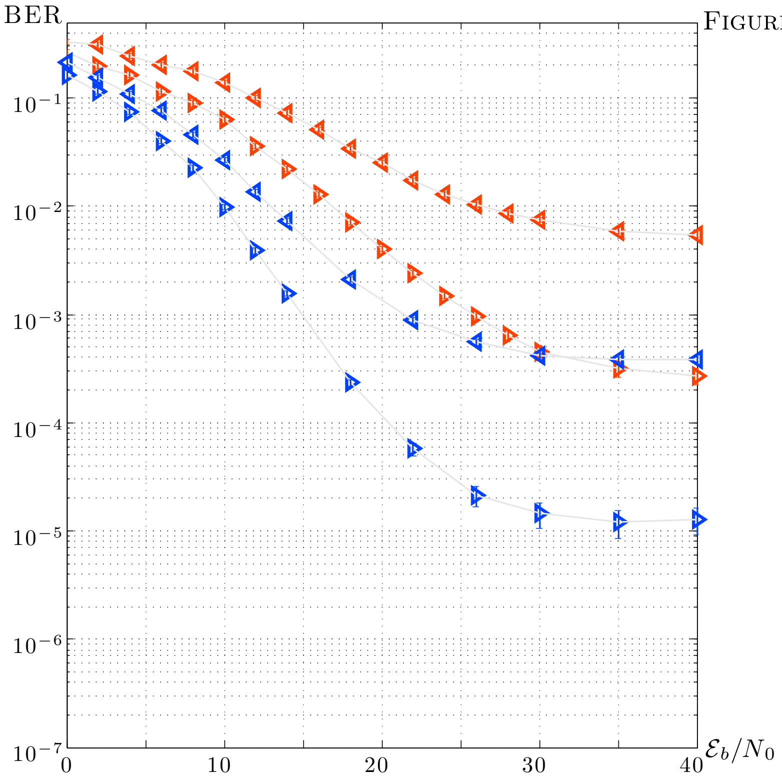


FIGURE 2.10: Comparison between TR (blue) and TR/RAKE (red) perturbed in both TR and RAKE (left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB]).

on page 31). In this case, we could exploit the full potential of TR increasing the number of fingers in RAKE receiver: it turns out that TR/RAKE actually outperforms an *all*-RAKE without TR (see FIGURE 2.6 on page 32), so there exists a minimum number of fingers such that the loss is zero.

If an interfering user uses TR, there are two cases to take into account depending on the focusing of interfering signal: (1) if it is not focused (weak interference), then TR performs as well as RAKE, hence there is no longer any loss (see FIGURE 2.7 on page 33), while (2) if it is focused (strong interference), then the loss remains and it is necessary to consider more fingers in RAKE receiver (see FIGURE 2.11 on the next page).

If TR of the interfering signal is perturbed, then the maximum interfering energy decreases (see FIGURE 2.3 on page 28), resulting in better BER (see FIGURE 2.12 on the next page).

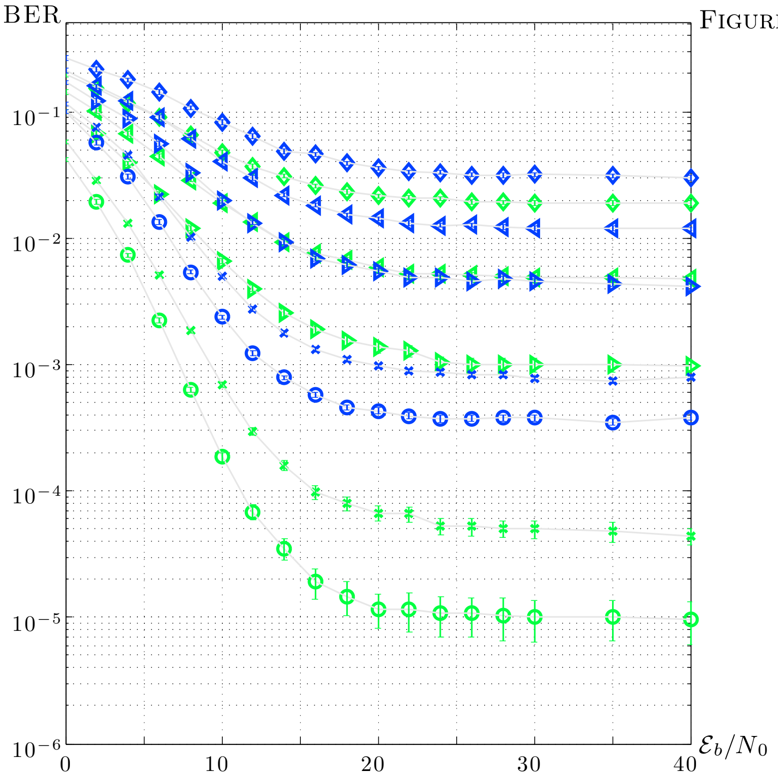


FIGURE 2.11: Comparison between TR (blue) and TR/RAKE (green) with perturbations (circle: no perturbation, cross: $\lambda/2 =$ 20 [dB], left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB], diamond: $\lambda/2 = 7$ [dB]).

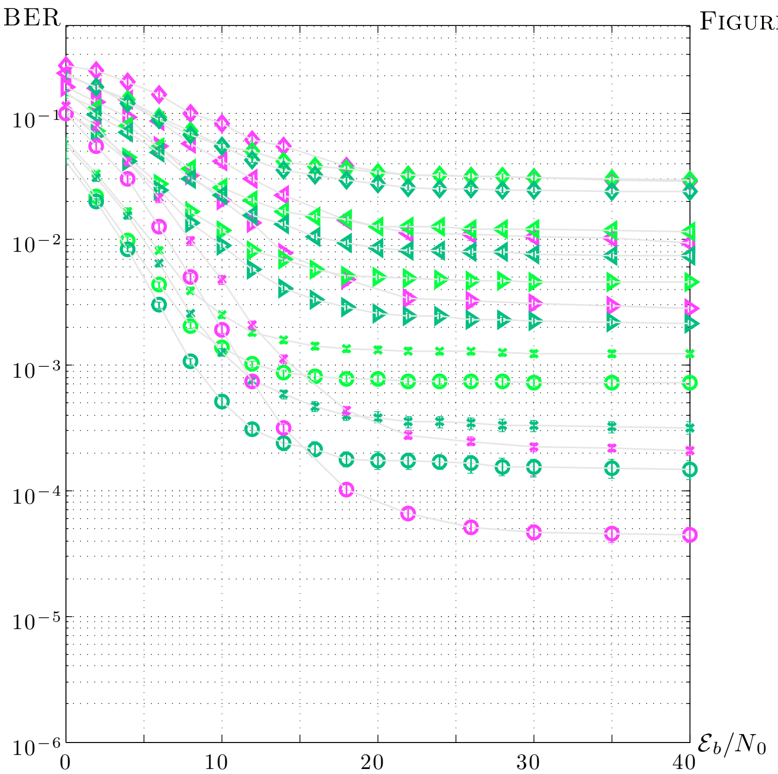


FIGURE 2.12: Comparison between RAKE (magenta) and TR/RAKE (light green, with MUI, and dark green, with interfering users employing perturbed pre-filters) with perturbations (circle: no perturbation, cross: $\lambda/2 = 20$ [dB], left-triangle: $\lambda/2 = 13$ [dB], right-triangle: $\lambda/2 = 10$ [dB], diamond: $\lambda/2 = 7$ [dB]).

CONCLUSION

In this thesis we have addressed the problem of finding limitations and strenghts of Time Reversal. The main results are three.

The first result regards the minimization of the average complexity, linked to the total number of taps and fingers, of IR-UWB systems by means of TR. To compare complexities, we start with a system that uses a c-RAKE (without TR) when the channel has L paths. If $c \ll L$, then all the systems with k taps and m fingers with $k + m = c$ have the same performance and thus are equivalent; otherwise, the system with the minimum complexity tends to have $\sqrt{c}$ taps and $\sqrt{c}$ fingers. This represents the big complexity gap that can be achieved with TR, passing from $c + 1$ to $2\sqrt{c}$ taps and fingers.

The second result concerns the maximum energy gain brought by TR in IR-UWB systems. By means of point process theory, we compute the energy limit that an IR-UWB system can collect. As a consequence, we know the best achievable BEP and, from a different point of view, the maximum allowable energy saving to preserve the performance.

The third one is actually a set of results on the robustness of TR in scenarios with and without MUI and on the effect of a perturbed TR on the MUI. A perturbation, as well as MUI, has the effect of introducing a BER floor, but TR can be exploited to reduce only the latter. Furthermore, the use of TR can lead to negative effects on other users. This happens when a strong cross-correlation between two channels (interfering transmitter and reference receiver or interfering receiver) occurs. However, with low SIR, TR offers usual advantages. Moreover, if MUI uses TR and it is perturbed, then the BEP of the reference user is better because of the deviation (due to the perturbation) from a condition of maximum interference.

CODE

In the following are presented few MATLAB codes useful to perform some of the simulation seen in past chapters.

Basic algorithms

Algorithm: energy estimation

```
%% loading channel database
load('ch_db.mat');
channel_database=h;
clear h;
nChannels = size(channel_database,1);
Ec = sum(channel_database.^2,2);
EcAve = mean(Ec);

% Each time is a multiple of 100 [ps]
fs = 10e9;

Tc = 10; % = 1 [ns]
Nh = 50;
Tf = Nh*Tc;
%Np = Nb*Ns;
Tm = 1;

nTaps = 50;
nFingers = 50;

tap=(1:10:nTaps);
finger=(1:10:nFingers);

EbAvgN0_dB_vec = 0:2:16;
Eb = 1;
EbAve = Eb*EcAve;
gamma_bAve = 10.^(EbAvgN0_dB_vec/10);
N0 = EbAve./gamma_bAve;

rel_err_des = 0.001;

err_count = zeros(length(EbAvgN0_dB_vec),length(tap),length(finger));
BERcount = zeros(length(EbAvgN0_dB_vec),length(tap),length(finger));
BER = zeros(length(EbAvgN0_dB_vec),length(tap),length(finger));

nTestFEC = 1e1; %For Each Channel

nAttempts = 1e5;
countpts=1;

%%
tic
for ebno=1:length(EbAvgN0_dB_vec),
  fprintf('\nEbN0:␣%d␣of␣%d␣',[ebno,length(EbAvgN0_dB_vec)]);
  countpts=1;
  for na=1:nAttempts,
    if na/nAttempts>countpts*0.1,
    fprintf('.');
    countpts = countpts+1;
    end

    s = zeros(1,Tf);
    b = round(rand);
    a = -1+2*b;
    c = floor(Nh*rand);
    s(c*Tc+1)=a;
    s = s/norm(s);

    h = channel_database(1+mod(na,nChannels),:);
    L = sum(h~=0);

    %
    for k=1:length(tap),%K=1:L,
```

```
      K=tap(k);

      if K<=L,
        pre_filter = fliplr(pathsel(h,K));
        x = sparseconv2(s,pre_filter);
        x = x/norm(x); % normalisation

        y = sparseconv2(x,h);
        ycirc = makecirc(y,Tf);

        %
        for n=1:length(finger),%N=1:1+L*(K-1),
        N=finger(n);

        if N<=1+K*(L-1),
          if 1/sqrt(BERcount(ebno,k,n)) > rel_err_des,
            rake = a*pathsel(ycirc,N);

            for nFEC=1:nTestFEC,
            noise = sqrt(N0(ebno)/2)*randn(1,Tf);

            r = ycirc+noise;
            a_est = sign(rake*r.');
            if a_est~=a,
              err_count(ebno,k,n) = err_count(ebno,k,n) + 1;
            end
            BERcount(ebno,k,n) = BERcount(ebno,k,n)+1;
            end
          end
        end
        end%of rakes for a given prefilter
      else
        break;
      end
    end%of prefilters for a given channel

  end%of error counting
  if BERcount(ebno,k,n)==0,
    break;
  end
end

for ebno=1:length(EbAvgN0_dB_vec),
  for k=1:length(tap),%K=1:nbig,
    for n=1:length(finger),%N=1:nbig,
    if BERcount(ebno,k,n)~=0,
      BER(ebno,k,n) = err_count(ebno,k,n) / BERcount(ebno,k,n);
    end
    end
  end
end
toc
```

Algorithm: BER estimation

```
%% Loading channel
% load('chdb.mat');
% channel_database=h;
% clear h;
% nChannels = size(channel_database,1);
% Eh = sum(channel_database.^2,2);
% EhAvg = mean(Eh);

% % % --- Trivial channel
% channel_database_old=channel_database;
% channel_database = zeros(size(channel_database_old));
% channel_database(:,1)=ones(size(channel_database,1),1);
% Eh = sum(channel_database.^2,2);
% EhAvg = mean(Eh);
% % % --- end of Trivial channel

%% Init algorithm

% Signal parameter
fs = 10e9;      % Each time is a multiple of 100 [ps]
dt = 100e-12;   % Simulator resolution = 100 [ps]
ndt = 1/(fs*dt);

% Each time is expressed in samples.
```

```
Tc = 20; % = 2 [ns]
Nh = 50;
Tf = Nh*Tc;
%Np = Nb*Ns;
Tm = 20;
Ns=1;

% System Parameters
EbAvgN0_dB_vec = [0:2:14 18:4:30 35 40];
N0 = EhAvg;

varPert_vec = linspace(0,0.5*N0/2,2);
PNR=10.^([-Inf -20 -13 -10 -7]/10);
varPert_vec = PNR * N0/2;

Q = 20;  % numero di interferenti
JNR_vec = -Inf;

% Parametri simulazione
BER = zeros(length(EbAvgN0_dB_vec),length(JNR_vec),length(varPert_vec));
BER_ML = zeros(length(EbAvgN0_dB_vec),length(JNR_vec),length(varPert_vec));

BERcount = zeros(length(EbAvgN0_dB_vec),length(JNR_vec),length(varPert_vec));
BERcount_ML = zeros(length(EbAvgN0_dB_vec),length(JNR_vec),length(varPert_vec));


th_count_max = 3e5;
Es = 1;

eps_std = 0.05;   % accuracy desired (std)

b_jammerTR = 0;   % Is jammer with or without TR?
b_jammerpTR = 0;  % Is jammer's pre-filter perturbed?
PNR_jammer = 0;
varPert_jammer = PNR_jammer * N0/2;
b_realcase = 1;   % not focused
b_worstcase = abs(1-b_realcase); % focused
b_ML = 0;
b_noise=1;

b_pTR = 0;      % Is perturbed the FULL-TR?
b_pRAKE = 0;    % Is perturbed the ALL-RAKE?
b_full_all=0;   % Max performance?

b_ppm = 0;
b_ppm_nc = 0;

b_long = 1;

tau=7e-10;
wf = zeros(1,Tm/dt * 1/fs);
t = [fliplr(-(0:dt:Tm/(2*fs))) (dt:dt:Tm/(2*fs))];

if tau==0, % rect-pulse
  sig = ones(1,length(wf));
  sig( end ) = 0;
  sig(1)=0;
else % Scholtz-pulse: 2nd order gaussian derivative
  sig = (1-4.*pi.*((t./tau).^2)).*...
    exp(-2.*pi.*((t./tau).^2));
end

sig = sig/norm(sig);

%% Start

for varPertToTest=1:length(varPert_vec),
  sigma_d = sqrt(varPert_vec(varPertToTest));
  fprintf('\nPerturbation␣no.␣%d␣of␣%d,\n',[varPertToTest length(varPert_vec)]);

  for jnrToTest=1:length(JNR_vec),
    fprintf('␣␣Jammer␣no.␣%d␣of␣%d,\n',[jnrToTest length(JNR_vec)]);

    JNR = JNR_vec(jnrToTest);
    jnr = 10^(JNR/10);        % without TR

    % Scenario
    if jnr==0,
    b_jamming=0;
    else
    b_jamming=1;
```

```
end

if b_jamming==1 && b_ML==1 && Ns>1,
thML=1; % ML decision with non-gaussian interference
else
thML=0; % ML on WGN
end

for ebToTest=1:length(EbAvgN0_dB_vec),
for demodML=0:thML,
  fprintf('␣␣EbN0␣no.␣%d␣of␣%d:',[ebToTest length(EbAvgN0_dB_vec)]);

  EbAvgN0_dB = EbAvgN0_dB_vec(ebToTest); % [dB]

  gamma_bAvg = 10^(EbAvgN0_dB/10); % desired
  Ex = gamma_bAvg;
  countpts=0;
  count = 0;  %iteration counter
  err_count=0;  %error counter
  if thML==1 && demodML==0,
    Z = [];
  end
  %
  tic

  p_current=0.5e-10;
  th=1/(p_current*eps_std^2);

  while (count<th) && count<th_count_max,
    if count/th>countpts*0.05,
    fprintf('.');
    countpts = countpts+1;
    end

    softDt=zeros(1,Ns);

    %% Useful signal
    s = zeros(1,Tf);
    b = round(rand);
    a = -1+2*b;
    c = floor(Nh*rand);
    if b_ppm==0,
    s(c*Tc+1)=a;
    else
    s(c*Tc+1+ (1-b))=1;
    end
    s = s/norm(s);
    if b_long,
    s = sparseconv2(s,sig);
    s = makecirc(s,Tf);
    end

    % channel
    h = channel_database(floor(1+(nChannels-1)*rand),:);
    L = sum(h~=0);

    if b_pTR,
      K=0;
    else
      K=1;
    end
    if b_full_all,
      K=0;
    end
    pre_filter = fliplr(pathsel(h,K));
    if b_pTR,
      pre_filter_nop = pre_filter;
      pre_filter(pre_filter~=0) = pre_filter(pre_filter~=0)+...
        sigma_d*randn(size(pre_filter(pre_filter~=0)));
    end

    if b_pRAKE,
      N=0;
    else
      N=1;
    end
    if b_full_all,
      N=0;
    end

    if b_pTR,
      x_nop = sparseconv2(s,pre_filter_nop);
      x_nop = x_nop/norm(x_nop) * sqrt(Ex); % normalisation
```

```
  y_nop = sparseconv2(x_nop,h);
  ycirc_nop = makecirc(y_nop,Tf);

  rake = a*pathsel(ycirc_nop,N);
else
  rake = a*pathsel(ycirc,N);
end

if b_ppm==1,
  rake=a*rake;
  if b==0,
    rake=circshift(rake.',-1).';
  end
end

if b_pRAKE,
  rake = rake/sqrt(Ex);
  rake(rake~=0) = rake(rake~=0)+sigma_d*randn(size(rake(rake~=0)));
end

%% Detection
E_ML = norm(rake)^2;
z=0;
for ns=1:Ns,
%% MUI

if b_jamming==1,
  if mod(count,100)==0,
    jammer = zeros(1,Tf);
    for q=1:Q,
    s_jammer = zeros(1,Tf);
    b_jammer = round(rand);
    a_jammer = -1+2*b_jammer;
    c_jammer = floor(Nh*rand);
    s_jammer(c_jammer*Tc+1)=a_jammer;
    s = s/norm(s); %
    if b_long,
      s = sparseconv2(s,sig);
      s = makecirc(s,Tf);
    end

    % channel
    h_jammer = channel_database(floor(1+(nChannels-1)*rand),:);

    if b_jammerTR,
      K_jammer=0;
    else
      K_jammer=1;
    end
    pre_filter_jammer = fliplr(pathsel(h_jammer,K_jammer));
    if b_jammerpTR,
      pre_filter_jammer(pre_filter_jammer~=0) = ...
        pre_filter_jammer(pre_filter_jammer~=0) + ...
        sqrt(varPert_jammer)*randn(size(pre_filter_jammer(pre_filter_jammer~=0)));
    end
    x_jammer = sparseconv2(s_jammer,pre_filter_jammer);
    x_jammer=x_jammer/norm(x_jammer)*sqrt(1/2*jnr*Tf/Q*Ex);
    if b_worstcase,
      y_jammer = sparseconv2(x_jammer,h_jammer);
    end
    if b_realcase,
      h_anotherjammer = channel_database(floor(1+(nChannels-1)*rand),:);
      y_jammer = sparseconv2(x_jammer,h_anotherjammer);
    end
    ycirc_jammer = makecirc(y_jammer,Tf);

    jammer=jammer+ycirc_jammer;

    end % of new jamming signal generation
  else
    jammer = circshift(jammer.',floor(rand*length(jammer))-1).';
  end
else
  jammer = zeros(1,Tf);
end

if thML==1 && demodML==0,
  z = z+rake*jammer.';
end
%% Pulse correlation
noise = sqrt(N0/2)*randn(1,Tf);
if b_noise==0, noise=zeros(size(noise)); end
```

```
        if b_long==1, noise=noise/sqrt(2); end
        r = ycirc+noise+jammer;

        if b_ppm==0,
          softDt(ns)=rake*r.';
        else
          if b_ppm_nc==0, % so it is coherent
            softDt(ns)=rake*(b*ycirc+noise+jammer).' - ...
              (circshift(rake.',1).')*((1-b)*ycirc+noise+jammer).';
          else %it is non-coherent
            rake_pos = zeros(size(rake));
            rake_pos(rake~=0)=1;
            softDt(ns) = norm(rake_pos.*(b*ycirc+noise+jammer))^2 - ...
            norm( (circshift(rake_pos.',1).') .*((1-b)*ycirc+noise+jammer))^2;
          end
        end

        end %of Ns
        % %%%%
        if thML==1 && demodML==0,
        Z = [Z z];
        end

        if demodML==0,
        decision_variable = sum(softDt);
        end

        if demodML==1,
        r_ML=0;
        for ns=1:Ns,
          r_ML = r_ML + abs(softDt(ns)+E_ML)^ML_pow - abs(softDt(ns)-E_ML)^ML_pow;
        end
        decision_variable = r_ML;
        end

        a_est = sign( decision_variable );

        if a_est~=a,
        err_count = err_count + 1;
        end
        count = count+1; %iteration counter

        p_current = count/(count+1) * p_current + (err_count/count)/(count+1);

        th=floor(1/(p_current*eps_std^2));
      end%of error counting

      if thML==1 && demodML==0,
        excess_kurt = kurtosis(Z)-3;
        ML_pow=fzero(inline(...
        ['(gamma(5/x)*gamma(1/x)/((gamma(3/x))^2))-3-',...
        num2str(excess_kurt)]),1);
      end

      if demodML==0,
        BER(ebToTest,jnrToTest,varPertToTest) = err_count/count;
        BERcount(ebToTest,jnrToTest,varPertToTest) = count;
      end

      if demodML==1,
        BER_ML(ebToTest,jnrToTest,varPertToTest) = err_count/count;
        BERcount_ML(ebToTest,jnrToTest,varPertToTest) = count;
      end


      if count<th, fprintf('␣Accuracy␣not␣reached.␣'); end
      toc
      if err_count==0, fprintf('␣␣␣␣␣␣␣␣␣Critical␣BER␣evaluation.␣Break.\n');
        break;
      end
    end
    if err_count==0, break; end
    end
  end
end
```

Classes

More complex and complete simulations have been performed by means of an OO approach. In order to do this, various classes have been written, some of which are outlined in the following. AX stands for ACTS, our lab.

AXAwgn

```
classdef AXAwgn < handle
    %AXAWGN AWGN class.
    %   Specify Eb/N0 [dB] white gaussian noise.

    properties

        % External
        fs = [];            % sampling frequency [Hz]
        L = [];             % length of noise vector [sample]
        Pn = [];            % noise power [V^2]
        % Not-mandatory
        Tb = [];            % bit period [s]
        SNRdB = [];         % snr [dB]

        % Internal
        noise = [];         % wgn object
        EbN0 = [];          % EbN0 parameter corresponding to SNRdB

    end %properties

    methods

        % constructor
        function this = AXAwgn(fs, L)
            this.fs = fs;
            this.L = L;
        end

        % init
        function init(obj)
            if ~isempty(obj.SNRdB) && ~isempty(obj.Tb),
                obj.EbN0 = 0.5 * 10^(obj.SNRdB/10) * (obj.Tb*obj.fs);
            end

        end

        % routines
        function gn(obj, fs, L, Pn)
            obj.noise = AXSignal( randn(1,L), fs );
            obj.noise.setPower(Pn);
        end

    end %methods

end
```

AXChannel

```
classdef AXChannel < handle
    %AXChannel Channel class.
    %   Generate several channels.

    properties

        % External
        fs = [];            % bit obj from the source
        mod = [];           % modulation type (0=802.15.3a, 1=802.15.4a)
        L = [];             % number of samples of the channel representation
        s;                  % signal from TX
        Ni;                 % num of finger of the equalizer
        No;                 % num of taps of the rake receiver

        % Internal
        pref;               % pre-filter
        h;                  % channel object
        x;                  % signal from TX passed through pre-filter
        y;                  % signal passed through the noise-free channel

        % Others
        hTR;                % TR equivalent channel: hTR = conv(pref,h)
        rake;               % rake taps

    end %properties

```

```
    methods

        % constructor
        function obj = AXChannel(mod, fs, L, Ni)
            obj.mod = mod;
            obj.fs = fs;
            obj.L = L;
            obj.Ni = Ni;
        end

        % exec
        function exec(obj)
            obj.genChannel;
            obj.genPrefilter;
            obj.y = AXSignal(sparseconv(obj.x.signal,obj.h.signal),obj.fs);
        end

        % genChannel
        function genChannel(obj)
            if obj.mod==0,
                obj.ieee802153a;
            end
        end

        % set
        function setSignal(obj, s)
            obj.s = s;
        end

        % genPrefilter
        function genPrefilter(obj)
            hi = pathsel(fliplr( obj.h.signal ), obj.Ni);
            obj.pref = AXSignal( hi, obj.fs );

            % --- 'x' has the same power of 's'
            obj.x = AXSignal(...
                   sparseconv(obj.pref.signal,obj.s.signal),obj.fs );
            currPow = obj.x.getPower;
            P_s = obj.s.getPower;
            obj.x.setPower( P_s );
            alpha = P_s / currPow;
            obj.pref.signal = sqrt(alpha) * obj.pref.signal;
            % ---
        end

        % genRake
        function genRake(obj, No)
            obj.No = No;
            obj.hTR = AXSignal( ...
                   sparseconv(obj.pref.signal,obj.h.signal),obj.fs );
            ho = pathsel(obj.hTR.signal, No);
            obj.rake = AXSignal( ho, obj.s.fs );
        end

        % aux routines
        function ieee802153a(obj)
            % at now only star topology available
            %gamm=1.7; A0=47; d=10; c0=10^(-A0/20);
            gamm=1.7; A0=47; d=10; c0=10^(-A0/20);
            ag = (c0/sqrt(d^gamm));

            [~,HF,~,~,~] = channelIEEE(obj.fs,ag^2,obj.L);
            obj.h = AXSignal(HF, obj.fs);
        end

    end %methods

end
```

AXDemodulator

```
classdef AXDemodulator < handle
    %AXDemodulator Modulator class.
    %   Demodulate electrical signals into decision variables.

    properties

        % External
        r = [];          % signal obj to demodulate
        No = [];         % no of fingers of the RAKE-receiver
```

```
        mod = [];       % modulation type (0=TH-PPM, 1=TH-PAM, 2=DS-PAM)

        s = [];
        h = [];

        % Internal
        Z = [];         % decision variable object

    end %properties

    methods

        % constructor
        function this = AXDemodulator(r, No, mod, s, h)
            this.r = r;
            this.No = No;
            this.mod = mod;

            this.s = s;
            this.h = h;
        end

        % routines
        function buildrake( )

        end

        function demodulate(obj)

            Nb = length(obj.s.b.signal);
            Na = length(obj.s.a.signal);
            Ns = Na / Nb;

            nc   = floor(obj.s.Tc * obj.s.fs);     % chip size [sample]
            ns   = nc * obj.s.Nh;                  % slot size [sample]
            neps = floor(obj.s.epsilon*obj.s.fs); % PPM shift size [sample]
            n    = ns .* Na;                       % ind. fun. size [sample]

            % Resync
            hShift = find(obj.hTR.signal~=0,1,'first');
            %rx = obj.sTR.signal;%+obj.noise.signal;%+obj.mui.signal;
            rx = rx(hShift:end);
            RAKE = obj.hRAKE.signal(hShift:end);

            obj.r = UFSignal( rx, obj.s.fs );
            z = zeros(1,Nb);
            zs = zeros(Nb,Ns);
            %bEst = zeros(1,Nb);

            % MASK is the basis mask signal
            mask = zeros(1,nc);
            wf = obj.s.w.signal; nw = length(wf);
            if nw>nc,
                warning('UFUser:demodulator',...
                        'Waveform␣size␣greater␣than␣chip␣size');
            end
            mask(1:nw) = wf;

            if nc-neps>=nw,
                mask(neps+1:neps+nw) = -wf;
            else
                mask(neps+1:nc) = -wf(1:nc-neps);
            end

            mR = conv( RAKE, mask );
            L = length(mR);
            idx = find( obj.s.indTH.signal ~= 0 );

            for n=1:Nb,
                for k=1:Ns,
                    i = idx((n-1)*Ns+k);
                    zs(n,k) = rx(i:i+L-1)*mR.';
                end
                z(n) = sum(zs(n,:));
            end
        end

    end %methods

end
```

AXEncoder

```
classdef AXEncoder < handle
    %AXEncoder Encoder class.
    %   Encode bits in symbols.

    properties

        Ns = [];         % bit rep number
        b;               % prev signal obj
        Ts = [];         % symbol period [s]
        a = [];          % symb obj (= encoded bit obj)

    end %properties

    methods
        %% Interface

        % Constructor
        function obj = AXEncoder(Ns)
            obj.Ns = Ns;
        end

        % exec
        function exec(obj)
            %symbols = rectpulse( -1+2*obj.b.signal, obj.Ns );
            symbols = reshape(ones(obj.Ns,1)*(-1+2*obj.b.signal),1,[]);
            obj.a = AXSignal( symbols, obj.Ns*obj.b.fs );
        end

        % get
        function s = getSignal(obj)
            s = obj.a;
        end

        % set
        function setSignal(obj, s)
            obj.b = s;
        end

        %% Aux routines




    end %methods

end
```

AXModulator

```
classdef AXModulator < handle
    %AXModulator Modulator class.
    %   Modulate symbols into electrical signals.

    properties

        % Input
        Tc=[];      % chip time [s]
        Ts=[];      % frame or slot time or average PRT [s]
        Np=[];      % code period
        mod=[];     % modulation type (0=TH-PPM, 1=TH-PAM, 2=DS-PAM)
        Tm=[];      % pulse duration
        PdBm=[];    % power (averaged in 1 slot)
        Nh=[];      % slot len in chips or max num of users mux in 1 slot

        a;          % symb obj [symb]

        epsilon=[]; % PPM shift [s]
        tau=[];     % pulse shaping factor (if 0, then rect-pulse)
        fs=[];

        % Processed
        c=[];       % code obj
        w=[];       % basis waveform (Scholtz-like or rect-like) obj
        ind=[];     % indicator function (where to put the waveforms)
        indTH=[];   % partial indicator function (only time-hopped)
        amp=[];     % amplitudes @ fs
        indamp=[];  % comb-like function with modulated Dirac amplitudes
        signalTH=[];% only-TH modulated signal (w/o PPM)
```

```
            signal = [];      % output signal

    end %properties

    methods

        % constructor
        function obj = AXModulator(mod,Tc,Ts,Nh,Np,Tm,PdBm,fs,epsilon,tau)
            obj.mod = mod;
            obj.Tc = Tc;
            obj.Ts = Ts;
            obj.Nh = Nh;
            obj.Np = Np;
            obj.Tm = Tm;
            obj.PdBm = PdBm;
            obj.fs = fs;
            obj.epsilon = epsilon;
            obj.tau = tau;
        end

        % set
        function setSignal(obj, s)
            obj.a = s;
        end

        % get
        function s = getSignal(obj)
            s = obj.signal;
        end

        % main routines
        function exec(obj)
            % Build the signal
            obj.gen;
        end

        % aux routines
        % routines
        function gencode(obj) % only 1 period
            if obj.mod == 0, % PPM case
                cod = round( (obj.Nh-1) * rand(1, obj.Np) );
                obj.c = AXSignal(cod,1/obj.Ts);
            else % PAM case
                cod = round( rand(1, obj.Np) );
                obj.c = AXSignal(-1+2*cod,1/obj.Ts);
            end
        end
        function genwf(obj)
            dt=1/obj.fs;
            t = [fliplr(-(0:dt:obj.Tm/2)) (dt:dt:obj.Tm/2)];

            if obj.tau==0, % rect-pulse
                L = length(t);
                s = ones(1,L);
                s( floor(L/2):end ) = 0;
                s(1)=0;
                %s(1)=0; s(end)=0;
            else % Scholtz-pulse: 2nd order gaussian derivative
                s = (1-4.*pi.*((t./obj.tau).^2)).*...
                        exp(-2.*pi.*((t./obj.tau).^2));
            end

            obj.w = AXSignal(s, obj.fs);
            obj.w.setEnergy(1);
        end

        function gensignal(obj)
            %% 1.
            Na = length(obj.a.signal);

            nc   = floor(obj.Tc * obj.fs);       % chip size [sample]
            ns   = nc * obj.Nh;                  % slot size [sample]
            neps = floor(obj.epsilon*obj.fs);    % PPM shift size [sample]
            n    = ns .* Na;                     % ind. fun. size [sample]

            comb      = zeros(1,n); % periodic Dirac function @ Ts
            combTH    = zeros(1,n); % positions of TH signal
            combTHPPM = zeros(1,n); % positions of TH+PPM signal

            for k = 1 : Na,
```

```
                % uniform pulse position
                index = 1 + (k-1)*ns;
                comb(index) = 1;

                % introduction of TH
                ck = obj.c.signal(1+mod(k-1,obj.Np));
                combTH(index + ck*nc) = 1;

                % introduction of PPM (after TH)
                ak = obj.a.signal(k);
                combTHPPM(index + ck*nc + (1+ak)/2*neps) = 1;

            end

            if obj.mod==0 || obj.mod==1, % TH (PPM or PAM)
                obj.ind    = AXSignal(combTHPPM, 1);
                obj.indTH = AXSignal(combTH, 1); %optimization tip: make
                % idx vectors *not* logical but integer
            else % DS (PAM)
                obj.ind    = AXSignal(comb, 1);
            end

            %% 2.
            Eslot = (10^(obj.PdBm/10))/1e3 * obj.Ts;

            ampl = ones(1, Na);

            if obj.mod==0, % TH-PPM
                ampl = sqrt(Eslot) * ampl;
            elseif obj.mod==1, % TH-PAM
                ampl = sqrt(Eslot) * obj.a.signal;
            else % DS-PAM
                q = floor( Na / obj.Np );
                r = Na - q*obj.Np;
                for k=1:q,
                    ampl( 1+(k-1)*obj.Np : k*obj.Np ) = obj.c.signal .* ...
                        obj.a.signal( 1+(k-1)*obj.Np : k*obj.Np );
                end
                ampl(Na-r+1:Na) = obj.c.signal(1:r) .* ...
                    obj.a.signal(1:r);
                ampl = sqrt(Eslot) * ampl;
            end

            %obj.amp = AXSignal( rectpulse(ampl,ns), obj.fs );
            obj.amp = AXSignal( reshape(ones(ns,1)*ampl,1,[]), obj.fs );

            %% 3.
            nw = length( obj.w.signal );

            obj.indamp = AXSignal(obj.ind.signal.*obj.amp.signal, obj.fs);
            %sig = zeros(1, n + nw - 1);
            sig = sparseconv( obj.indamp.signal, obj.w.signal );
            obj.signal = AXSignal( sig, obj.fs );

            if obj.mod==0 || obj.mod==1, % TH-case
                sTH = sparseconv( obj.indTH.signal.*obj.amp.signal,...
                    obj.w.signal );
                obj.signalTH = AXSignal( sTH, obj.fs );
            end

        end
        function gen(obj)
            obj.gencode;
            obj.genwf;
            obj.gensignal;
        end


    end %methods

end
```

AXReceiver

```
classdef AXReceiver < handle
    %AXTX Receiver class.
    %   RX class.

```

```
    properties

        % External

        % Internal
        TX;                % REF TX
        CH;                % REF CH

        SNR_dB;            % signal-to-noise-ratio [dB]
        EbN0_dB;           % Eb/N0 [dB]
        mui;               % MUI signal obj
        noise;             % noise signal obj
        sigtodem;          % signal for demodulation obj

        Z=[];              % decision variables from demodulation
        bit_est=[];        % estimated bits from detection

        BER_est;           % estimated BER
        % Others


    end %properties

    methods
        %% INTERFACE

        % constructor
        function obj = AXReceiver(tx,ch)
            obj.TX = tx;
            obj.CH = ch;
        end

        % gen noise
        function gennoiseEbN0(obj, Pref, EbN0_dB)
            Tb = obj.TX.source.Tb;
            fs = obj.TX.modulator.fs;
            ebn0 = 10^(EbN0_dB/10);          % linear
            snr = 2*ebn0/(Tb*fs);            % linear
            obj.SNR_dB = 10*log10(snr);
            Pn = Pref / snr;
            obj.gennoisePn(Pn);
        end

        function gennoiseSNR(obj, Pref, SNR_dB)
            Tb = obj.TX.source.Tb;
            fs = obj.TX.modulator.fs;
            snr = 10^(SNR_dB/10);            % linear
            EbN0 = 0.5 * snr * (Tb*fs);      % linear
            obj.EbN0_dB = 10*log10(EbN0);
            Pn = Pref / snr;
            obj.gennoisePn(Pn);
        end

        function gennoisePn(obj, Pn)
            L = length( obj.CH.y.signal );
            obj.noise = AXSignal( randn(1,L), obj.TX.modulator.fs );
            obj.noise.setPower(Pn);
        end

        % set multi user interference
        function setmui(obj, MUI)
            obj.mui = AXSignal( MUI, obj.TX.modulator.fs );
        end

        % define signal for demodulation
        function defsig(obj, flag)
            % flag=0: only useful signal (w/o any disturb)
            % flag=1: only interference
            % flag=2: useful signal + noise
            % flag=3: useful signal + interference
            % flag=4: useful signal + noise + interference
            switch flag,
                case 0
                    obj.sigtodem = obj.CH.y;
                case 1
                    obj.sigtodem = obj.mui;
                case 2
                    obj.sigtodem = AXSignal( ...
                        obj.CH.y.signal + obj.noise.signal, ...
                        obj.TX.modulator.fs );
```

```
                case 3
                    obj.sigtodem = AXSignal( ...
                        obj.CH.y.signal + obj.mui.signal, ...
                        obj.TX.modulator.fs );
                case 4
                    obj.sigtodem = AXSignal( ...
                        obj.CH.y.signal+obj.noise.signal+obj.mui.signal,...
                        obj.TX.modulator.fs );
            end
        end

        % execute
        function exec(obj)
           obj.demodulator;
           obj.detector;
           obj.BER_estimation;
        end

        function demodulator(obj)
            Nb = obj.TX.source.Nb;
            Ns = obj.TX.encoder.Ns;
            Na = Ns * Nb;

            nc   = floor(obj.TX.modulator.Tc *...
                         obj.TX.modulator.fs);    % chip size [sample]
            ns   = nc * obj.TX.modulator.Nh;      % slot size [sample]
            neps = floor(obj.TX.modulator.epsilon * ...
                        obj.TX.modulator.fs); % PPM shift size [sample]

            % Resync
            hShift = find(obj.CH.hTR.signal~=0,1,'first');
            rx = obj.sigtodem.signal(hShift:end);
            RAKE = obj.CH.rake.signal(hShift:end);

            z = zeros(1,Nb);
            zs = zeros(Nb,Ns);


            % MASK is the basis mask signal
            mask = zeros(1,nc);
            wf = obj.TX.modulator.w.signal;
            nw = length(wf);
            if nw>nc,
                warning('UFUser:demodulator',...
                        'Waveform size greater than chip size');
            end
            mask(1:nw) = wf;

            if nc-neps>=nw,
                mask(neps+1:neps+nw) = -wf;
            else
                mask(neps+1:nc) = -wf(1:nc-neps);
            end

            mR = conv( RAKE, mask );
            L = length(mR);
            idx = find( obj.TX.modulator.indTH.signal ~= 0 );

            for n=1:Nb,
                for k=1:Ns,
                    i = idx((n-1)*Ns+k);
                    zs(n,k) = rx(i:i+L-1)*mR.';
                end
                z(n) = sum(zs(n,:));
            end

            obj.Z = z;
        end

        function detector(obj)
            obj.bit_est = -sign(obj.Z);
        end

        function BER_estimation(obj)
           obj.BER_est=sum(abs(obj.bit_est-(obj.TX.encoder.a.signal)))/...
               length(obj.bit_est);
        end

        % getter

        % setter
```

```

        %% AUX




    end

end
```

AXSignal

```
classdef AXSignal < handle
    %AXSIGNAL Signal class.
    %   One-dimensional signals or sequences and their properties.

    properties

        fs=[];          % sampling frequency or rate [Hz]
        signal=[];      % signal samples [V^2] or sequence values
        t0=0;           % Optional - time reference to first sample

    end %properties

    methods

        % constructor
        function obj = AXSignal(signal, fs)
            obj.signal = signal;
            obj.fs = fs;
        end

        % routines
        function setEnergy(obj, E)
            % E: desired energy [V^2 s]
            Ts = 1/obj.fs;
            alpha = norm(obj.signal)*sqrt(Ts/E);
            obj.signal = obj.signal ./ alpha;
        end
        function en = getEnergy(obj)
            Ts = 1/obj.fs;
            en=Ts*norm( obj.signal )^2;
        end
        function setPower(obj, P)
            % P: desired average power [V^2]
            alpha = norm(obj.signal)/sqrt(length(obj.signal)*P);
            obj.signal = obj.signal ./ alpha;
        end
        function pow = getPower(obj)
            pow=norm( obj.signal )^2 / length(obj.signal);
        end

        function shiftsignal(obj, delta)
            %SHIFTSIGNAL One-dimensional signal shift of delta [s].
            %   It moves the signal of delta*fs samples creating a
            %   longer version which is a translation of the first.
            nshift = floor(abs( delta * obj.fs ));
            if delta > 0,
                obj.signal  = [zeros(1,nshift) obj.signal];
            else
                obj.signal  = [obj.signal zeros(1,nshift)];
                obj.t0 = obj.t0 - nshift / obj.fs; % = delta (slotted)
            end
        end


    end %methods

end
```

AXSignalUWB

```
classdef AXSignalUWB < AXSignal
    %AXSIGNALUWB UWB signal class.
    %   Subclass of signals.

    properties

        % - Special properties of UWB signals
```

```
        Tc=[];      % chip time [s]
        Ts=[];      % frame or slot time or average PRT [s]
        Np=[];      % code period
        mode=[];    % modulation type (0=TH-PPM, 1=TH-PAM, 2=DS-PAM)
        Tm=[];      % pulse duration
        PdBm=[];    % power (averaged in 1 slot)
        a=[];       % encoded sequence [symb]
        b=[];       % bit [bit]
        Nh=[];      % slot len in chips or max num of users mux in 1 slot

        % - Non-mandatory to set
        epsilon=[]; % PPM shift [s]
        tau=[];     % pulse shaping factor (if 0, then rect-pulse)

        % - Computed
        c=[];       % code (TH-both or DS-PAM)
        w=[];       % basis waveform, usually Scholtz-like (signal class)
        ind=[];     % indicator function (where to put the waveforms)
        indTH=[];   % partial indicator function (only time-hopped)
        amp=[];     % amplitudes @ fs
        indamp=[];  % comb-like function with modulated Dirac amplitudes
        signalTH=[];% only-TH modulated signal (w/o PPM)

    end

    methods

        % constructor
        function obj = AXSignalUWB(mode, b, a, Tc, Ts, Nh, Np, Tm, ...
                PdBm, fs, epsilon, tau)
            obj = obj@AXSignal([],fs);
            obj.mode = mode;
            obj.b = b;
            obj.a = a;
            obj.Tc = Tc;
            obj.Ts = Ts;
            obj.Nh = Nh;
            obj.Np = Np;
            obj.Tm = Tm;
            obj.PdBm=PdBm;
            obj.epsilon = epsilon;
            obj.tau = tau;
        end

        % routines
        function gencode(obj) % only 1 period
            if obj.mode == 0, % PPM case
                cod = round( (obj.Nh-1) * rand(1, obj.Np) );
                obj.c = AXSignal(cod,1/obj.Ts);
            else % PAM case
                cod = round( rand(1, obj.Np) );
                obj.c = AXSignal(-1+2*cod,1/obj.Ts);
            end
        end
        function genwf(obj)
            dt=1/obj.fs;
            t = [fliplr(-(0:dt:obj.Tm/2)) (dt:dt:obj.Tm/2)];

            if obj.tau==0, % rect-pulse
                L = length(t);
                s = ones(1,L);
                s( floor(L/2):end ) = 0;
                s(1)=0;
                %s(1)=0; s(end)=0;
            else % Scholtz-pulse: 2nd order gaussian derivative
                s = (1-4.*pi.*((t./obj.tau).^2)).*...
                        exp(-2.*pi.*((t./obj.tau).^2));
            end

            obj.w = AXSignal(s, obj.fs);
            obj.w.setEnergy(1);
        end
        function genind(obj)
            %GENIND Indicator function generator.
            %   COMB        represents a periodic Dirac function @ Ts
            %   COMBTH      represents positions of TH signal
            %   COMBTHPPM   represents positions of TH+PPM signal
            Nb = length(obj.b.signal);
            Na = length(obj.a.signal);
```

```
            Ns = Na / Nb;

            nc   = floor(obj.Tc * obj.fs);      % chip size [sample]
            ns   = nc * obj.Nh;                 % slot size [sample]
            neps = floor(obj.epsilon*obj.fs);   % PPM shift size [sample]
            n    = ns .* Na;                    % ind. fun. size [sample]

            comb       = zeros(1,n);
            combTH     = zeros(1,n);
            combTHPPM = zeros(1,n);

            for k = 1 : Na,

                % uniform pulse position
                index = 1 + (k-1)*ns;
                comb(index) = 1;

                % introduction of TH
                ck = obj.c.signal(1+mod(k-1,obj.Np));
                combTH(index + ck*nc) = 1;

                % introduction of PPM (after TH)
                ak = obj.a.signal(k);
                combTHPPM(index + ck*nc + (1+ak)/2*neps) = 1;

            end

            if obj.mode==0 || obj.mode==1, % TH (PPM or PAM)
                obj.ind   = AXSignal(combTHPPM, 1);
                obj.indTH = AXSignal(combTH, 1); %optimization tip: make
                % idx vectors *not* logical but integer w/idx numbers
            else % DS (PAM)
                obj.ind   = AXSignal(comb, 1);
            end
        end
        function genamp(obj)
            Nb = length(obj.b.signal);
            Na = length(obj.a.signal);
            Ns = Na / Nb;

            nc   = floor(obj.Tc * obj.fs);      % chip size [sample]
            ns   = nc * obj.Nh;                 % slot size [sample]
            neps = floor(obj.epsilon*obj.fs);   % PPM shift size [sample]
            n    = ns .* Na;                    % ind. fun. size [sample]

            Eslot = (10^(obj.PdBm/10))/1e3 * obj.Ts;

            ampl = ones(1, Na);

            if obj.mode==0, % TH-PPM
                ampl = sqrt(Eslot) * ampl;
            elseif obj.mode==1, % TH-PAM
                ampl = sqrt(Eslot) * obj.a.signal;
            else % DS-PAM
                q = floor( Na / obj.Np );
                r = Na - q*obj.Np;
                for k=1:q,
                    ampl( 1+(k-1)*obj.Np : k*obj.Np ) = obj.c.signal .* ...
                        obj.a.signal( 1+(k-1)*obj.Np : k*obj.Np );
                end
                ampl(Na-r+1:Na) = obj.c.signal(1:r) .* ...
                    obj.a.signal(1:r);
                ampl = sqrt(Eslot) * ampl;
            end

            obj.amp = AXSignal( rectpulse(ampl,ns), obj.fs );

        end
        function gensignal(obj)

            n = length( obj.ind.signal );
            nw = length( obj.w.signal );

            obj.indamp = AXSignal(obj.ind.signal.*obj.amp.signal, obj.fs);
            obj.signal = zeros(1, n + nw - 1);

            obj.signal = sparseconv( obj.indamp.signal, obj.w.signal );

            if obj.mode==0 || obj.mode==1, % TH-case
```

```
                sTH = sparseconv( obj.indTH.signal.*obj.amp.signal,...
                    obj.w.signal );
                obj.signalTH = AXSignal( sTH, obj.fs );
            end

        end
        function gen(obj)
            obj.gencode;
            obj.genwf;
            obj.genind;
            obj.genamp;
            obj.gensignal;
        end

    end

end
```

AXSource

```
classdef AXSource < handle
    %AXSOURCE Source class.
    %   Source of equally probable bits.

    properties

        Nb = [];        % number of bits
        Tb = [];        % bit period [s]
        b = [];         % bits obj

    end %properties

    methods
        %% Interface

        % constructor
        function obj = AXSource(Nb, Tb)
            obj.Nb = Nb;
            obj.Tb = Tb;
        end

        % execute
        function exec(obj)
            obj.genbit(obj.Nb,obj.Tb);
        end

        % get
        function s = getSignal(obj)
            s = obj.b;
        end

        % set: no signal to set for the source

        %% Aux routines

        % genbit
        function genbit(obj, Nb, Tb)
            %GENBIT Bit generator.
            %   Source of Nb equiprobable bernoulli IID bits at rate 1/Tb.
            bits = rand(1,Nb)>0.5 ;
            obj.b = AXSignal( bits, 1/Tb );
        end

    end %methods

end
```

AXTransmitter

```
classdef AXTransmitter < handle
    %AXTX Transmitter class.
    %   TX class.

    properties

        % External

        % Internal
        s;          % Signal
```

```

        source;     % Source obj
        encoder;    % Encoder obj
        modulator;  % Modulator obj

    end %properties

    methods
        %% INTERFACE

        % constructor
        function obj = AXTransmitter(Nb, Tb, Ns, mod, Tc, Ts, Nh, ...
                Np, Tm, PdBm, fs, epsilon, tau)

            obj.source = AXSource(Nb,Tb);
            obj.source.exec;
            is = obj.source.getSignal;
            b = is; %bkp

            obj.encoder = AXEncoder(Ns);
            obj.encoder.setSignal(is);
            obj.encoder.exec;
            is = obj.encoder.getSignal;
            a = is; %bkp

            obj.modulator = AXModulator(mod,Tc,Ts,Nh,Np,Tm,...
                PdBm,fs,epsilon,tau);
            obj.modulator.setSignal(is);
            obj.modulator.exec;
            is = obj.modulator.getSignal;
            suwb = is; %bkp

            obj.s = is;
        end
        % execute

        % getter

        % setter

        %% AUX

    end

end
```

BIBLIOGRAPHY

[DRF95] A Derode, P Roux, and M Fink. "Robust Acoustic Time Reversal with High-Order Multiple Scattering". In: *Physical Review Letters* 75.23 (1995), pp. 4206–4209. URL: http://www.ncbi.nlm.nih.gov/pubmed/10059846. (Cit. on p. 2).

[Fin+09] Mathias Fink et al. "Time-reversed waves and super-resolution". In: *Comptes Rendus Physique* 10.5 (2009), pp. 447–463. URL: http://linkinghub.elsevier.com/retrieve/pii/S1631070509001054. (Cit. on p. 2).

[DBG04] Maria-Gabriella Di Benedetto and Guerino Giancola. *Understanding Ultra-Wideband Radio Fundamentals*. Prentice-Hall, 2004. (Cit. on p. 2).

[WS00] M.Z. Win and R.A. Scholtz. "Ultra-wide bandwidth time-hopping spread-spectrum impulse radio for wireless multiple-access communications". In: *Communications, IEEE Transactions on* 48.4 (2000), pp. 679 –689. ISSN: 0090-6778. DOI: 10.1109/26.843135. (Cit. on p. 2).

[JFB11] G. Capodanno J. Fiorina and M.-G. Di Benedetto. "Impact of Time Reversal on Multi-User Interference in IR-UWB". In: *Ultra-Wideband, 2011. ICUWB 2011. IEEE International Conference on*. 2011. (Cit. on p. 2).

[DN+] Luca De Nardis et al. "Combining UWB with Time Reversal for improved communication and positioning". In: *Telecommunication Systems* (), pp. 1–14. ISSN: 1018-4864. (Cit. on p. 2).

[CI80] D. R. Cox and Valerie Isham. *Point Processes*. Chapman and Hall, 1980. (Cit. on pp. 2, 14).

[DVJ08] D. J. Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Elementary Theory and Methods*. Vol. 1. Springer, 2008. (Cit. on pp. 2, 14).

[Str10] Roy L. Streit. *Poisson Point Processes*. Springer, 2010. (Cit. on pp. 2, 14).

[SV87] A Saleh and R Valenzuela. "A Statistical Model for Indoor Multipath Propagation". In: *IEEE Journal on Selected Areas in Communications* 5.2 (1987), pp. 128–137. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1146527. (Cit. on p. 2).

[GH06] J. A. Gubner and K. Hao. "The IEEE 802.15.3a UWB Channel Model as a Two-Dimensional Augmented Cluster Process - transactions papers". In: *Information Theory, IEEE Transactions on* (2006). (Cit. on p. 2).

[Foe02] Jeff Foerster. "Channel Modeling Subcommittee Report (Final)". In: (2002). URL: http://www.ieee802.org/15/pub/2003/Mar03/02490r1P802-15_SG3a-Channel-Modeling-Subcommittee-Report-Final.zip. (Cit. on p. 2).

[**Mol+05**] A F Molisch et al. "IEEE 802.15. 4a channel model-final report". In: *Environments* 15 (2005), pp. 1–40. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.2038&rep=rep1&type=pdf. (Cit. on p. 2).

[**PFDB09**] D. Panaitopol, J. Fiorina, and M.-G. Di Benedetto. "Trade-off between the number of fingers in the prefilter and in the rake receiver in time reversal IR-UWB". In: *Ultra-Wideband, 2009. ICUWB 2009. IEEE International Conference on*. 2009, pp. 819–823. (Cit. on p. 2).

[**PS08**] John G. Proakis and Masoud Salehi. *Digital Communications (5th ed.)* McGraw-Hill, 2008.

[**HG07**] Kei Hao and J.A. Gubner. "The Distribution of Sums of Path Gains in the IEEE 802.15.3a UWB Channel Model". In: *Wireless Communications, IEEE Transactions on* (2007). (Cit. on p. 2).

[**FD09**] J. Fiorina and D. Domenicali. "The non validity of the gaussian approximation for multi-user interference in ultra wide band impulse radio: from an inconvenience to an advantage - transactions papers". In: *Wireless Communications, IEEE Transactions on* 8.11 (2009), pp. 5483–5489.