NTNU
Innovation and Creativity

Dynamic Spectrum Management in DSL Systems

Pablo Villalba Villalba

Master of Science in Electronics
Submission date: March 2007
Supervisor: Nils Holte, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

The candidate shall study different methods for dynamic spectral management in DSL systems, with main emphasis on autonomeous methods like iterative waterfilling. The methods shall be implemented in Matlab.

Assignment given: 04. September 2006
Supervisor: Nils Holte, IET

NTNU
Norwegian University of
Science and Technology

Dynamic Spectrum Management in DSL Systems

Master Thesis

Pablo Villalba Villalba Supervisor: **Nils Holte**

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Electronics and Telecommunications
Norwegian University of Science and Technology (NTNU)
7491 Trondheim, Norway

**Thanks to my parents, to my brother,
to my grandparents, my girlfriend Tere,
my professor Nils Holte and all the
people who put their trust in me.**

INDEX OF CONTENTS

INDEX OF FIGURES

8 DSM using 10 pairs of cables

9 Maximization sum of bitrates

ACRONYMS

ADSL: Asymmetric Digital Subscriber Line

CAP: Carrierless Amplitude/Phase

CO: Central Office

CP: Customer Premises

DMT: Discrete Multi-Tone

DSL: Digital Subscriber Line

DSM: Dynamic Spectrum Management

EMC: Electro-magnetic Compatibility

FEXT: Far End Crosstalk

FM: Fixed Margin

ISB: Iterative Spectrum Balancing

ITU: International Telecommunication Union

MA: Margin Adaptive

Mbps: *Megabits* per second

NEXT: Near End Crosstalk

OSB: Optimal Spectrum Balancing

PSD: Power Spectral Density

QAM: Quadrature Amplitude Modulation

RA: Rate Adaptive

RUO: Reference Unbundling Offer

SNR: Signal-to-Noise Ratio

UPBO: Upstream Power Back Off

VDSL: Very high speed Digital Subscriber Line

1 INTRODUCTION AND OBJECTIVES

1.1 Introduction

The new generation of ADSL (Asymmetric Digital Subscriber Line) and VDSL (Very high speed Digital Subscriber Line) offers the capability to deliver digital television and video-on-demand services over the fixed access network in addition to telephony and high speed internet access. This is often referred to as “triple play”.

Nevertheless, VDSL has had a very limited deployment in Europe compared to countries such as South-Korea or USA so far. However, some countries in Europe have held field trials or started limited deployment of new high capacity DSL (Digital Subscriber Line) systems. Thus, in the period 2000 – 2004, Telenor operated one of the first VDSL field trials in Europe in the Stavanger area (on the west coast of Norway). Other countries are in the planning phase where regulators work very closely with the main network operators.

Regulators may be faced with a number of issues related to the introduction of these new systems, especially for VDSL/VDSL2. Such issues include frequency planning, EMC (Electro-magnetic Compatibility) and interference, restrictions on copper pair utilization, unbundling of sub-loops and co-location in outdoor cabinets.

In this Master Thesis we centre our programs in the simulation of VDSL systems. These systems can offer a transfer capacity exceeding 20 Mbit/s but only within a relatively short range (about 1 km). However we have used in our simulations a new method of operation called DSM or Dynamic Spectrum Management (and more specifically Water-filling method) in order to increase this value of bitrate. With this method, we can do an adaptive assignment of the spectrum in a multiuser environment.

The worst problem of DSL systems in a pair of cables is the interference of signals from other pairs that are in the same cable. This type of interference is called crosstalk. In real systems we can consider two types of crosstalk, NEXT (near end crosstalk), which is interference between opposite directions of transmission, and FEXT (far end

crosstalk), which is the interference between systems using the same direction of transmission. In this point we can say that we have centred our study considering only White Noise and FEXT in the transmission lines. Then, with DSM method we will be able to prevent the service faults caused by crosstalk.

Along our work in this Thesis we will consider different criterions and systems with a different number of pairs of cables to obtain the results, and we will try to improve these results for each pair of cables.

Nowadays, ITU-T (International Telecommunication Union) has approved the new VDSL2 standard, which is the latest in a series of standards for different DSL technologies for broadband communication in the fixed access network. Theoretically, VDSL2 can provide a transmission capacity up to 200 Mbit/s but we will not study this method here.

1.2 Objectives

The objectives that we will achieve with this project are to study different methods for Dynamic Spectrum Management in DSL systems, using Water-filling with diverse criterions but always based on autonomous methods. A scheme of autonomous system is described in figure 1.1. In this picture we can see the structure and the crosstalk NEXT and FEXT that we will explain in the next point:

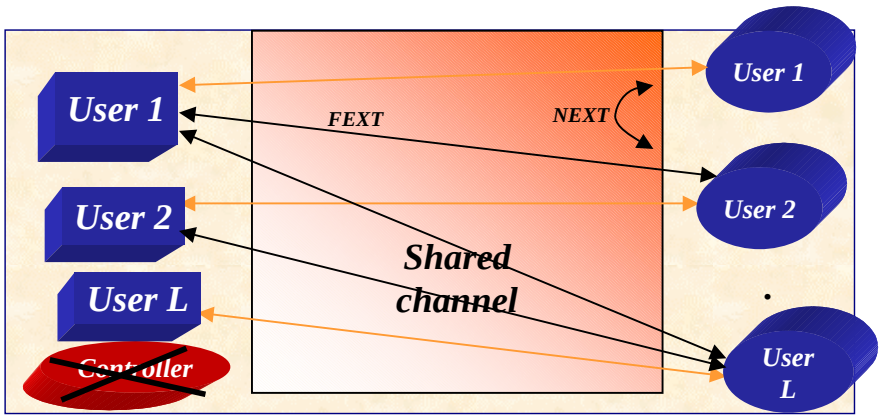


Figure 1.1: *System model in our study. Level 0 coordination or no coordination between different DSL lines. There is no coordination of signals or spectral use. There may however be autonomous control procedures that implement some form of DSM.*

Moreover we should use iterative Water-filling when we consider more than one pair of lines in order to achieve the target rates for each pair. We will use *Matlab*® to simulate the VDSL systems, DSM methods and the different kinds of interferences that we consider in our simulations.

On the other hand we will carry out the same programs without DSM methods. The existing spectrum management approach in DSL enforce PSD masks for all the modems, in which the PSD mask designs were based on the worst case crosstalk emission scenario. Then, the bitrate will be worse than with DSM. Thus, we will compare the results and we will see the improvements of DSM methods.

Finally, in the last points of the Thesis, we will use realistic values of interferences that were obtained in practice using many measurements. With these values we will obtain results closer to the reality.

At the end of this Thesis we will be able to compare the different results and decide which of the criterions used obtains better results. Thus, we will be able to decide whether to use or not to use PSD (Power Spectral Density) constraint, the criterion to maximize the sum of bitrates in all pairs of lines, the criterion to maximize the bitrate in the pair having the lowest bitrate etc.

2 CROSSTALK

A pair cable may contain up to 2000 different pairs. These pairs are organised in binder groups from 10 to up to 50 pairs. Moreover all the pairs are packed together within a cable sheath. The problem of this structure of the system is the electromagnetic interference that will be between all different pairs in the cable. Thus, this interference is called crosstalk, and it is distributed along the cable, and it varies stochastically both from pair to pair and along the cable length. We can differentiate two types of crosstalk, NEXT crosstalk and FEXT crosstalk. NEXT is simply defined as the unwanted signal coupling from a near-end transmitter into a pair measured at the same end. On the other hand, FEXT is defined as a measure of the unwanted signal coupling from a transmitter at the near-end into a neighbouring pair measured at the far-end.

In figure 2.1 and figure 2.2 we can see both effects between two pairs:

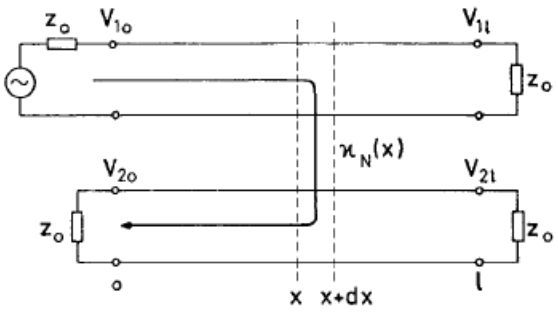


Figure 2.1: *Near end crosstalk model (NEXT) between two pairs.*

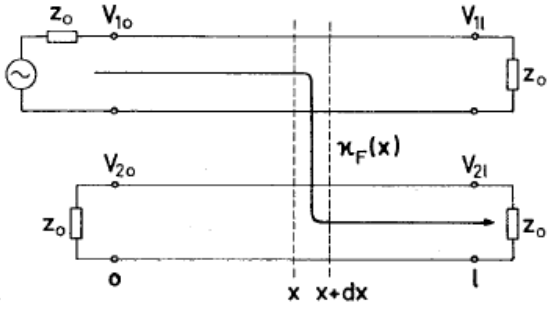


Figure 2.2: *Far end crosstalk model (FEXT) between two pairs.*

where Z_0 is the impedance, V is the voltage, $\kappa_N(x)$ is the normalised near end coupling coefficient and $\kappa_F(x)$ is the normalised far end coupling coefficient.

The near end crosstalk transfer function is given by:

$$H_{NE} = \frac{V_{20}}{V_{10}} = j\beta \int_0^l \kappa_N(x) e^{-2\alpha x - 2j\beta x} dx$$

and the far end crosstalk transfer function is given by:

$$H_{FE} = \frac{V_{2l}}{V_{1l}} = j\beta \int_0^l \kappa_F(x)dx$$

where "β" is the phase constant in rad/km and "α" is the attenuation constant in Neper/km.

In Figure 2.3 we can see a configuration in which two VDSL loops in the same binder emanate from the central office (CO) to the customer premises (CP). When both transmitters at the CP-side transmit with the same PSD, due to the difference in line attenuation, the FEXT caused by the short line can overpower the data signal in the long line.

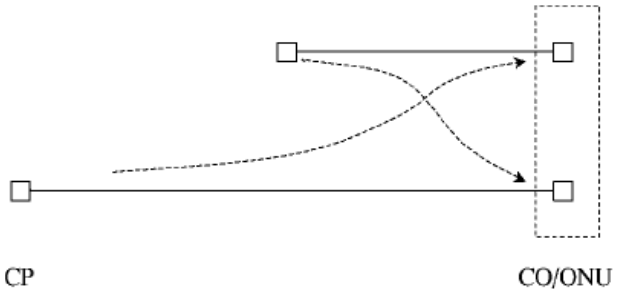


Figure 2.3: Two *VDSL loops in the same binder emanating from the central office (CO) to the customer premises (CP). The direction of the arrows indicate the interferences in upstream service.*

In the introduction, in figure 1.1 we saw a more complete system model with crosstalk FEXT and NEXT.

Nevertheless, in our study we have considered FEXT crosstalk only because we have simulated all the programs in the upstream band of VDSL (3.0 MHz to 5.1 MHz) and if we look at figure 2.3, upstream performance of the long line is severely affected by the upstream transmission of the short line. However, the downstream direction does not have the same problem because, although all transmitters at the CO-side also transmit at the same PSD, the FEXT they create into each other is identical at any fixed distance from CO.

To solve the problem in upstream, the short line must reduce its upstream PSD so that it does not cause excessive crosstalk into the long line. This reduction of the transmitted PSD in upstream is known as upstream power backoff (UPBO). On the other hand, the downstream FEXT level is always much weaker than the data signals, and for this reason it does not create a serious problem to downstream transmission.

3 VDSL STANDARD AND FREQUENCY PLAN

In the period 2000 – 2004, the first VDSL field trials in Europe used the frequency plan 998. Thus, for example, Telenor operated a VDSL pilot service in the Stavanger area on the west coast of Norway involving approximately 700 residential customers. Nevertheless, in 2004, Telenor introduced VDSL and ADSL2+ in their RUO (Reference Unbundling Offer) and the frequency plan was changed from 998 to 997. In Figure 3.1 we can see both plans of frequencies.

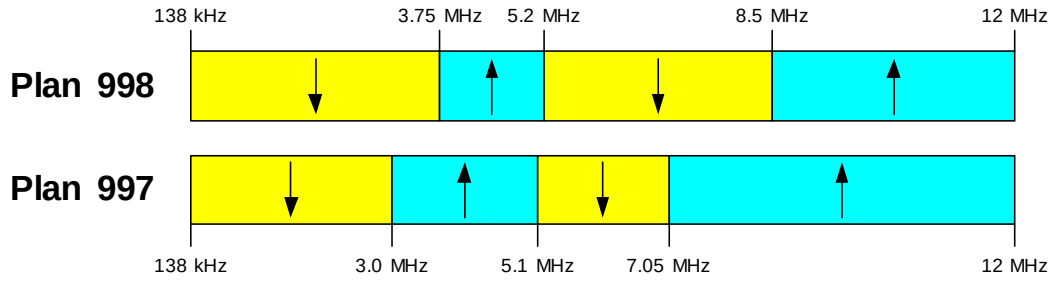


Figure 3.1: *Frequency plans for VDSL. The blue colour indicates the upstream band and the yellow colour the downstream band.*

In our simulation we will follow the plan of frequencies 997, and then, we will use the upstream band between 3.0MHz and 5.1MHz.

In addition, we will use DMT (discrete multi-tone) modulation because it achieves VDSL operational requirements more effectively than QAM or CAP modulation.

Moreover, VDSL standard with DMT modulation define the subchannel spacing like 4.3125 kHz. Thus, interoperability with ADSL is possible because both use the same subchannel spacing. Then, in our programs we have divided the upstream band ([3.0, 5.1] MHz) in 487 subchannels each of 4.3125 kHz.

All these specifications will be put in the Matlab program called *defxDSL.m* (see the program code in the Appendix or in the CD).

4 DSM USING 2 PAIRS OF CABLES

4.1 Introduction

As we mentioned in the introduction, we will study autonomous DSM systems. In this type of operation, the service provider sets data rates for the DSL services they will supply. Each modem is configured to run in one of three modes:

RA: rate adaptive, (maximize data rate).

MA: margin adaptive, (use all available power at given fixed data rate).

FM: fixed margin, (use only power needed to guarantee high-quality service).

The first two modes can improve the performance if the service provider sets the modes and data rates accordingly to DSM-standard guidelines, and require absolutely no coordination among DSL lines or service providers. Nevertheless, if we use all available power in a pair of cables, we will increase the interference in the other pairs of the same binder. For this reason the FM mode is the most favourable method to improve the overall use of the binder, and due to this, we will be able to increase the bitrates in all the pairs. In this point we will suppose that the binder is composed only by two pairs of cables. In the following points we will increase the number of pairs to three, five and ten in order to augment the realism of the simulations.

Moreover, in our programs of simulation, we will need to include information about the network (e.g. channel and crosstalk description, topology) and transmission parameters (e.g. power, data rate, bandwidth, energy/bit etc.). In the next point we explain these programs and moreover, we can see its code in the Appendix and in the CD.

4.2 Description of the programs

- ***defxDSL.m and initial.m***

The first program we should execute in our simulations is the ***defxDSL.m*** (see Appendix and file “DSM_2_pairs” in the CD). In this program we define the length of

both pair of lines, frequency of the Upstream band (in our case is [3.0, 5.1] MHz, following the 997 plan) and number of subchannels in order to divide the band in subchannels with 4.3125 KHz each. Moreover, we define the maximum and minimum bandwidth efficiency in bit/s/Hz. Notice that nowadays it is more difficult to find systems that work at more than 15 bit/s/Hz, and for this reason we have selected the maximum bandwidth efficiency of 14.5.

We select the transmitter output spectral density for each pair of cables at -50 dBm/Hz. This is:

$$-50dBm/Hz \rightarrow \quad 10^{\frac{-50}{10}} mW/Hz \rightarrow$$

$$10^{\frac{-50}{10}} mW/Hz \times 4.3125KHz = 0.043125mW$$

Then, we have 0.043125mW in each subchannel but we have 487 subchannels. Thus, the total power in mW will be:

$$0.043125mW \times 487 = 21mW$$

Moreover, we can select the type of attenuation depending on the thickness of the cable although we will always select attenuation proportional to sqrt(f) and 22.5 dB of attenuation at 1 MHz. This attenuation will be calculated in the program ***initial.m*** (see Appendix and file "DSM_2_pairs" in the CD).

The following parameters (AWGNon and FEon) are about the noise and the FEXT interference. We consider both effects like noise, and then, we will sum both effects with the receiver signal. Figure 4.1 explain the process:

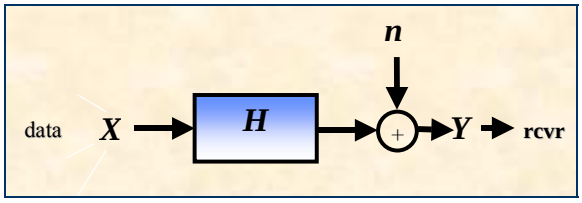


Figure 4.1: *Packet modelling of Matrix Channel.*

Thus, the channel model mathematically is:

$$Y = (X \times H) + n$$

where the dimensions of "*Y*"," *X*" and "*n*" in the case of two lines will be 2×1 and H would be 2×2. Then, "*n*" will be the sum of White Noise and FEXT.

Finally, we should put the ratio between length of disturbed pair and disturbing pairs in % for each system. This is:

$$L_{disturbing} = L_{disturbed} \times \frac{LFEdistRat}{100}$$

Thus, each system will have a different interference of FEXT from the other pair. We could see this idea in figure 2.3.

The last parameter that we use is the FEXT power sum in dB (FEXTps). It defines the FEXT sum at 1 MHz and 1 km for each pair of cables. We have used FEXTps=50.5dB as an example, but at the end of this thesis we will use realistic values that were obtained in practice.

- ***singcap.m and plotcap.m***

With the program ***singcap.m*** and ***plotcap.m*** (see Appendix and file "DSM_2_pairs" in the CD) we obtain the bitrate of each pair of lines. Thus, program singcap.m detects if we want to consider White Noise and FEXT, and sum both contributions like noise. Next, the program obtains the channel capacity per bandwidth unit (bandwidth efficiency) according to Shannon. The algorithm that we use is:

$$\eta(f) = \frac{\Delta C}{\Delta f} = \log_2\left(1 + \lambda \frac{S(f)}{N(f)}\right)$$

where $\lambda = 10^{-MARG/10}$ and "*MARG*" is the margin in dB relative to Shannon, "*S(f)*" is the signal spectral density at the receiver input and "$N(f)$" is the signal noise spectral density at the receiver input. Notice that the signal spectral density in linear is obtained by the small program ***sig.m*** and the white noise spectral density in linear is obtained by the small program ***whnoi.m***.

Finally, program *plotcap.m* obtains the capacity.

- ***fext. m***

In point two we talked about the interferences between pairs in the same binder. With program ***fext.m*** (see Appendix and file "DSM_2_pairs in the CD") we will be able to simulate FEXT phenomenon. Thus, we should define lengths of disturbing pairs. After it, we should define the output signal at the end of each cable following the next function:

$$P_{end} = 10^{PSD/10} \times \exp(-2 \cdot L \cdot Att_{line})$$

The crosstalk power transfer function is:

$$X_{Fi,j}(f) = \left|H_{Fi,j}(f)\right|^2 = P_{1Fi,j} \times F^2 \times L$$

where "L" is the cable length in Km, "F" is the frequency in MHz and "$P_{1Fi,j}$" is the normalized average FEXT given by:

$$P_{Fi,j}(f) = 10^{-FEXTps_{i,j}/10}$$

Thus, the interference that will be added with the White Noise in the program *singcap.m* will be:

$$FEXT = P_{end} \times X_{Fi,j}(f)$$

Finally, we have created the program that simulates the Water-filling method, but this program will be explained in detail in the next point.

4.3 Water-filling method

Nowadays, Water-filling is an algorithm used in most DMT-based modems to decide the energy and information distribution for the modem adaptively as a function of line conditions. The idea of Water-filling is to obtain the inverse of the SNR(f) (Signal-to-Noise Ratio) for the channel and fill that curve like a bowl with power until all available power has been used; hence the name water-filling. We can see this idea for a single user in figure 4.2:

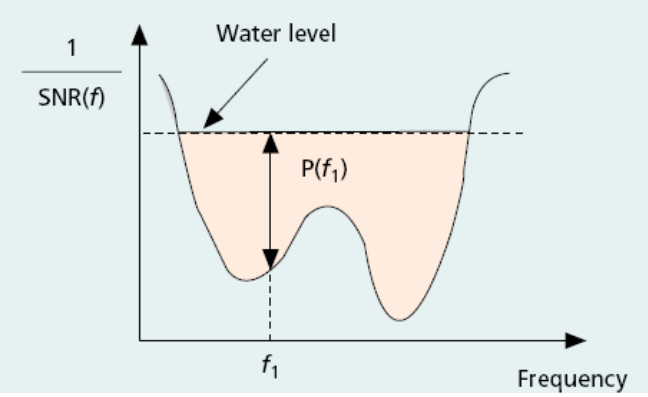


Figure 4.2: *Water-filling for a single user.*

The following function describes the Water level:

$$Water\_level = cons\tan t = \frac{1}{SNR(f)} + S_x(f)$$

where "$S_x(f)$" is the power spectral density and "f" can represent a discrete set of tones in a DMT system, $f = \frac{n}{T}, n = 1,..., N$, or a continuous variable of frequency.

On the other hand, we can obtain the number of bits transmitted in each tone n, with the next function:

$$b_n = \log_2\left(1 + \frac{SNR_n}{\Gamma}\right)$$

where "Γ" is the SNR gap and is equal to 1 in an ideal situation.

Thus, the data rate for a single user is:

$$R = \frac{1}{T}\sum_{n-1}^{N} b_n$$

where "T" is the length of the subchannel in seconds.

In our program ***waterfill.m*** (see Appendix and file "DSM_2_pairs" in the CD), we have followed the Hughes-Hartogs algorithm that generates a table of incremental energies required to transmit one additional bit on each of the subchannels. Then, at each step, one more bit is added to the subchannel that requires the least incremental energy. Thus, the energy to increment one bit in each subchannel is proportional to:

$$\Delta P_n = \frac{N(f)}{|H(f)|^2} \times \Delta f(mW)$$

where:

$$H(f) = 10^{-\frac{22.5 \cdot \sqrt{F} \cdot L}{20}}$$

$$\Delta f = 4.3125\text{e}3\text{Hz}$$

because, when we want to add the first bit:

$$1 = \log_2(1 + P_{n1} \cdot SNR_n) \rightarrow \Delta P_{n1} = P_{n1} - P_{n0} = \frac{1}{SNR_n}$$

and, when we want to add the second bit and the third bit:

$$2 = \log_2(1 + P_{n2} \cdot SNR_n) \rightarrow \Delta P_{n2} = P_{n2} - P_{n1} = \frac{3}{SNR_n} - \frac{1}{SNR_n} = \frac{2}{SNR_n}$$

$$3 = \log_2(1 + P_{n3} \cdot SNR_n) \rightarrow \Delta P_{n3} = \frac{4}{SNR_n}$$

.
.
.

Then, each time that we want to increase one bit in a subchannel, we will need twice as much energy as we consumed when we added the last bit.

4.4 Results Water-filling VS without Water-filling

The first step to obtain the results of the simulation is to describe its characteristics. Thus, in the program *defxDSL.m* we define a system with two lines, the first one with 1 Km and the second one with 0.7Km. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT, and then, the ratio between length of disturbed pair and disturbing pairs is 70 for system one and 142.85 for system two. The output spectral density is -50dBm/Hz and the other parameters can be seen in the program code.

Next, we obtain the results using Water-filling. We should run the next code lines in Matlab:

```
>> defxdsl
>> singcap
>> [y,ytot,b,btot,R] = waterfill(S1,Npsd1,21)
```

The results for pair one are:

ytot =20.9902 mW
btotal =4256 bits
R =18.3540 Mbps

The graphics are:

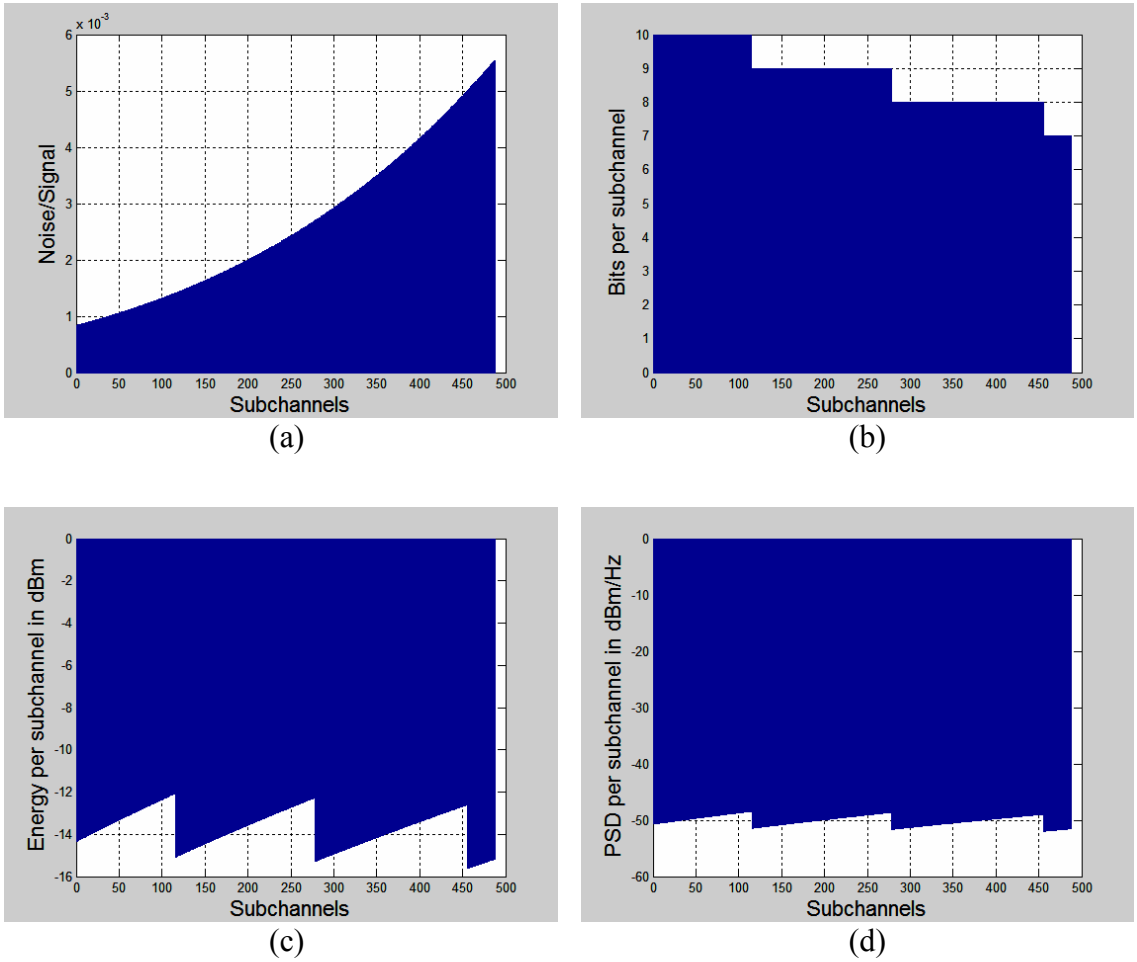


Figure 4.3: *Results for pair one with Water-Filling method. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in dBm and the PSD per subchannel in dBm/Hz respectively.*

For pair two we have:

```
>> [y,ytot,b,btot,R] = waterfill(S2,Npsd2,21)
```

And the results are:

ytot = 20.9703 mW
btotal = 8432 bits
R = 31.5028 Mbps

The graphics are:

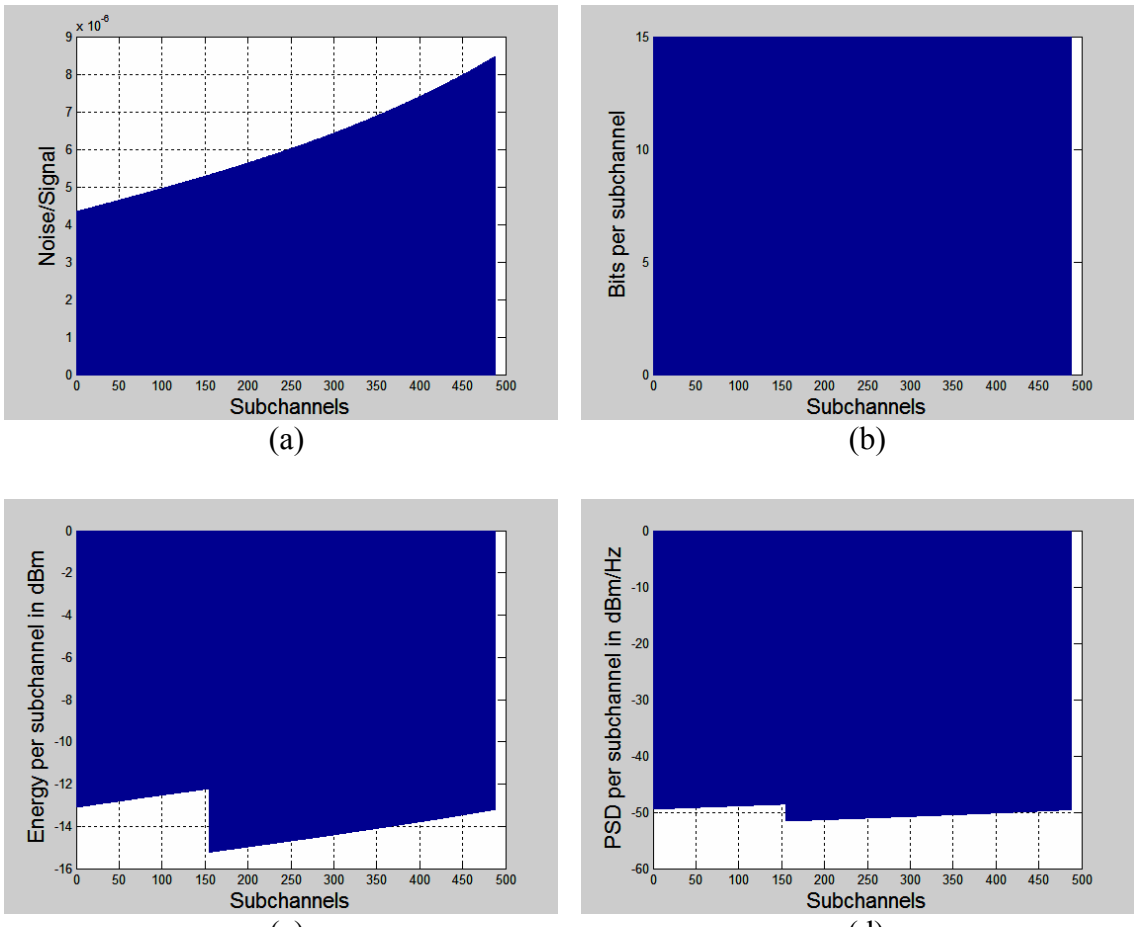


Figure 4.4: *Results for pair two with Water-Filling method. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in dBm and the PSD per subchannel in dBm/Hz respectively.*

Notice that the jumps of 3dB in Figure 4.3 (c) and Figure 4.4 (c) correspond to the increments of one more bit, because when we increment one bit we need to transmit 3dB more of power. In Figure 4.4 (c) this jump does not correspond to the increment of one more bit because we have a limitation of maximum bandwidth efficiency imposed

by the parameter "*BWEM*". As we explained, this is because nowadays it is more difficult to find systems that work at more than 15 bit/s/Hz.

On the other hand, we can see in the results that the longest line (pair one) has less capacity than the other line, because the attenuation of the longest line will be higher.

Now, if we obtain the values without Water-filling:

```
>> defxdsl
>> plotcap
```

R1 =16.2793 Mbps
R2 =30.4500 Mbps

As we can notice, the values of bitrate when we use Water-filling are higher than without Water-filling. The difference is around 2Mbps. Thus, we can offer better services using Water-filling.

4.5 Introduction to Rate Region and Iterative Water-filling

In this part of our simulation we will obtain the Rate Region of our systems and moreover we will create an iterative program of Water-filling in order to obtain the target rates that we need. These target rates should always be inside the bounds of the Rate Region.

Rate Region is the picture that depicts the combination of bitrates that we can achieve in each pair of cables in the same binder. Thus, the cables with less length will get better bitrates that the others because short lines have less losses. Moreover, if we increase the spectral density in one line, we will increase bitrate in this line, but probably we will obtain worse results in the other lines, because the FEXT in these lines will be higher. Figure 4.5 shows us a possible Rate Region for two DSL lines:

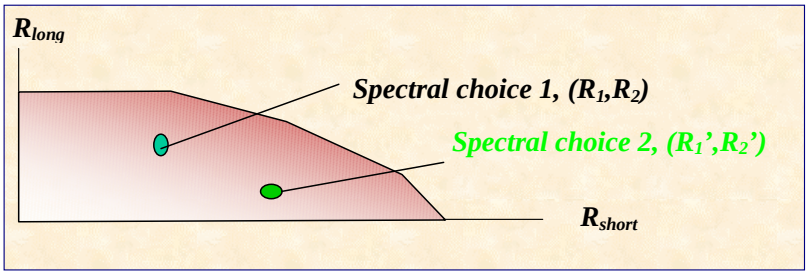


Figure 4.5: *Rate Regions for two DSL lines.*

Notice that when we consider more that two pairs of cables we should draw in more than two dimensions. Nevertheless, we will draw always in two dimensions putting the shortest line in axis "y" and the other lines in axis "x".

On the other hand, the Iterative Water-filling method executes Water-filling in each DSL system independently until all systems reach the target bitrate that we want. Thus, coordination is not necessary. In each step we will change the spectral density of the lines and we will recompute the signal and the interferences.

4.6 Programs Rate Region and Iterative Water-filling

- ***rate_region.m***

With the program called ***rate_region.m*** (see Appendix and file "Region_and_Iterwater" in the CD) we will obtain the Rate Region illustration. Thus, the inputs of the program are the total available power for system 1 and for system 2. Then, in the first step we consider that both pairs use the total available power. Next, we call the program *waterfill.m* and we compute the bitrates. In the next steps, we decrease the power of the shortest pair until the bitrate for this pair is zero and we recompute the interferences and the bitrates. Finally, if we draw the bitrates in each step we obtain the Rate Region.

Moreover we have created another program called ***rate_region1.m*** (see file "Region_and_Iterwater" in the CD) that obtains the Rate Region without Water-filling. The steps to achieve it are similar.

- ***iterwaterfill.m***

Program ***iterwaterfill.m*** (see Appendix and file "Region_and_Iterwater" in the CD) calls the program *waterfill.m* consecutively in order to reach the target rates for each pair of cables. Then, the inputs of the program should be these target rates. It is advisable to see the Rate Region first, because the target rates should be inside it. On the other hand, the outputs will be the power that was spent for each pair, and the bitrate that was reached. The process we follow in the program is described in figure 4.6:

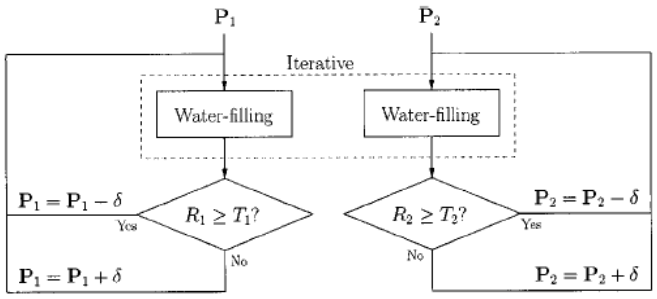


Figure 4.6: *Iterative Water-filling process.*
P_1 and P_2 are the power for pair 1 and pair 2. T_1 and T_2 are the Target Rates and R_1 and R_2 the bitrates.

If we look at the code of the program we can see that at the beginning we use δ=3dB but in the next steps we divide this value by two in order to obtain more exact results at the end. Moreover we add to each value of target rate a value equal to roughly 1% of the target rate to do better the equivalence $R \geq R\_Target$. Thus, we have $R > R\_Target + R\_Target/100$.

Finally, we will draw the attained bitrates step by step.

4.7 Rate Region and Iterative Water-filling results

The characteristics of the system that we used in this point of our simulation were the same than in the last point. Thus, in the program *defxDSL.m* we define two lines (1 Km and 0.7Km respectively). We consider White Noise (with -140dBm/Hz of spectral density) and FEXT, and, the ratio between the length of the disturbed pair and the disturbing pairs is 70 for system one and 142.85 for system two (the same values than in the last point). The output spectral density is -50dBm/Hz.

Next, we obtain the Rate Region figure using Water-filling. We should execute the next line of code in Matlab:

```
>> rate_region(21,21)
```

The graphic that we obtain is:

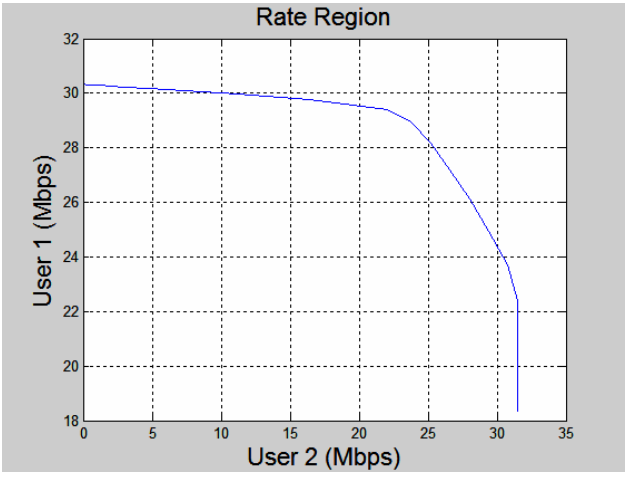


Figure 4.7: *Rate Region using Water-filling method. The lengths of the pairs are 1Km and 0.7Km respectively.*

As we can see in the last figure, User 2 achieves more bitrate than User 1 because the length of his line is shorter. Thus, the Rate Region characterizes all possible data rate combinations among all users subject to the power constraints. In the last figure we have a fall in the right part of the graphic. This fall occurs because we have imposed a restriction of bits in System 2 by the parameter BWEM=15, because nowadays it is more difficult to do systems that work at more than 15 bit/s/Hz.

In figure 4.8 we can see the distribution step by step of the PSD in dBm/Hz per subchannel that we followed to create the Rate Region of figure 4.7:

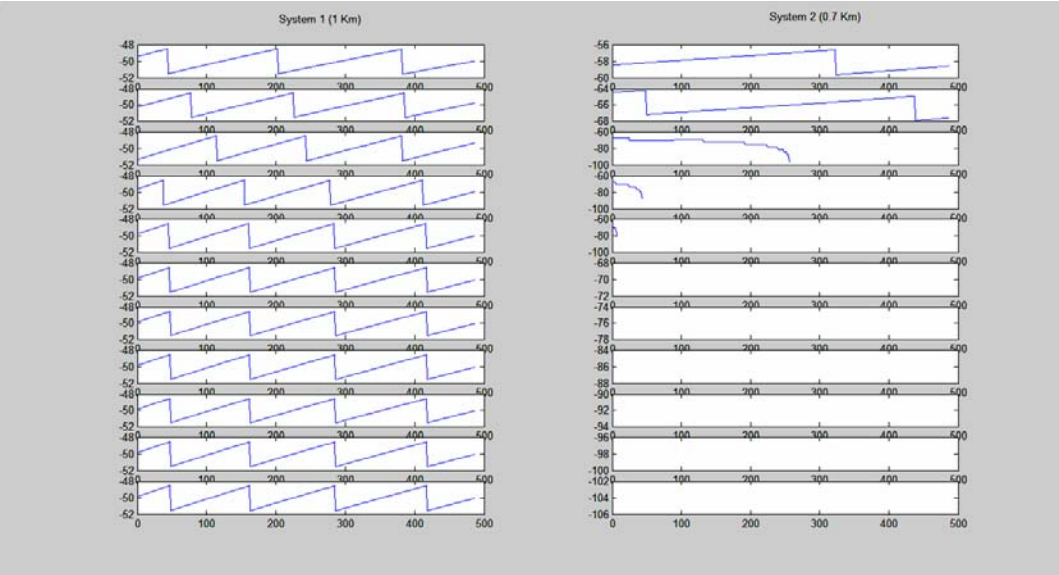


Figure 4.8: *Distribution of PSD to create the Rate Region. Axis "y" is the PSD in dBm/Hz and axis "x" the subchannels.*

As we can see, System 2 is decreasing its PSD until all the Rate Region is created. In System 1 the distribution of PSD along the subchannels is changing when we decrease the PSD of System 2. This happens because the FEXT in System 1 will be smaller when we have less power in System 2.

Now, if we compare the last Rate Region using PSD with the Rate Region without PSD we will be able to see the improvement when we use Water-filling.

We execute the next line of code in Matlab:

```
>> rate_region1(21,21)
```

The graphic that we obtain is:

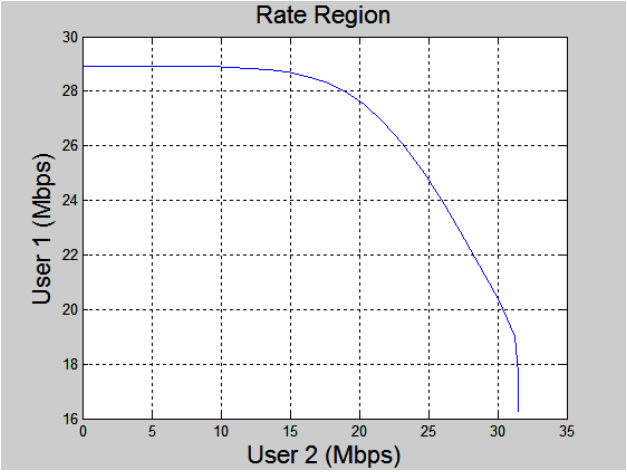


Figure 4.9: *Rate Region without Water-filling method. The lengths of the pairs are 1Km and 0.7Km respectively.*

As we see, the Rate Region is smaller now and the bitrates that we can achieve are smaller too.

In the next figures (figure 4.10 and figure 4.11) we will depict new Rate Regions using different lengths in the systems. Notice that when we change the length of the systems we should change the value of the variable *"LFEdistRat"* in the program *defxDSL.m* according to the expression: $L_{disturbing} = L_{disturbed} \times \frac{LFEdistRat}{100}$.

For the next graphics we have used:

User1: 1.5Km
User2: 1.2Km
LFEdistRat1=80 **LFEdistRat2=125**
FEXT1ps=50.5

User1: 1.3Km
User2: 1Km
LFEdistRat1=76.92 **LFEdistRat2=130**
FEXT1ps=50.5

User1: 1Km
User2: 0.7Km
LFEdistRat1=70 **LFEdistRat2=142.85**
FEXT1ps=50.5

Figure 4.10 represent different Rate Regions with Water-filling, and figure 4.11 represent the same, but without Water-filling:

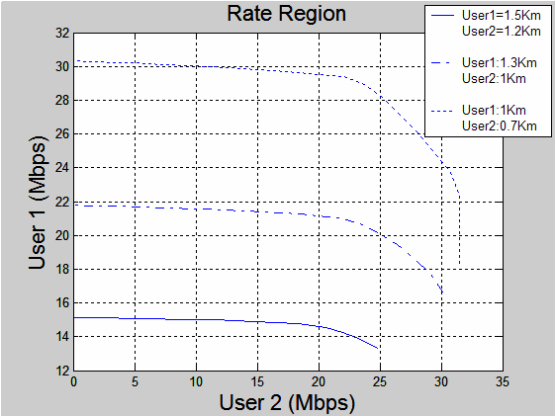


Figure 4.10: *Different Rate Regions using Water-filling.*

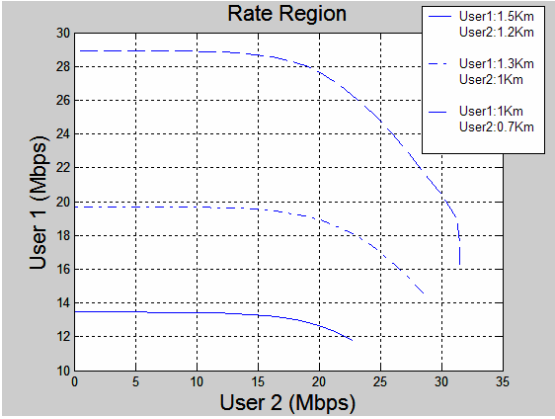


Figure 4.11: *Different Rate Regions without Water-filling.*

In the legend of the graphics we can see the length of the lines that we have used. Moreover, if we observe the results we can say that the shapes of the graphics with or without Water-filling are the same, but the results using Water-filling obtain higher values of bitrate, and therefore higher capacity for our VDSL applications. The difference between using and not using Water-filling is around 2Mbps of improvement with Water-filling.

Now, we can run the iterative program of Water-filling because we know the values of bitrates that are in the Rate Region, and then, the possible values that we can demand as target rates. Figure 4.12 correspond to the iterative Water-filling in a system with two pairs of cables. The first one with 1.5 Km long and the second one is 1.2 Km. The target rates that we have demanded are 14.5 Mbps for system 1 and 20 for system 2. As we can see in figure 4.10, these values are inside the Rate Region.

Then, we run the next code line in Matlab:

```
>> [R1_final,R2_final,ytot1_final,ytot2_final] = iterwaterfill(14.5,20)
```

The results and the graphic that we generate are the following:

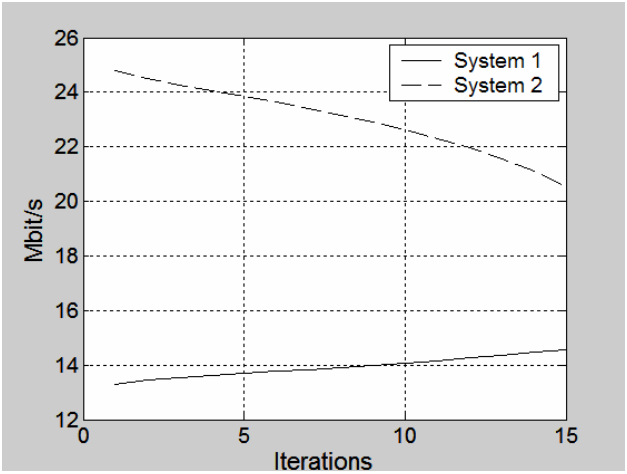


Figure 4.12: *Iterations in the iterative Water-filling. The program iterate until the target rates are achieved.*

R1_final = 14.5676 Mbps **ytot1_final = 21.0019 mW**
R2_final = 20.5491 Mbps **ytot2_final = 5.2279 mW**

The results show us that we have achieved the target rates. Moreover, in figure 4.13, we have drawn the PSD distribution in each iteration:

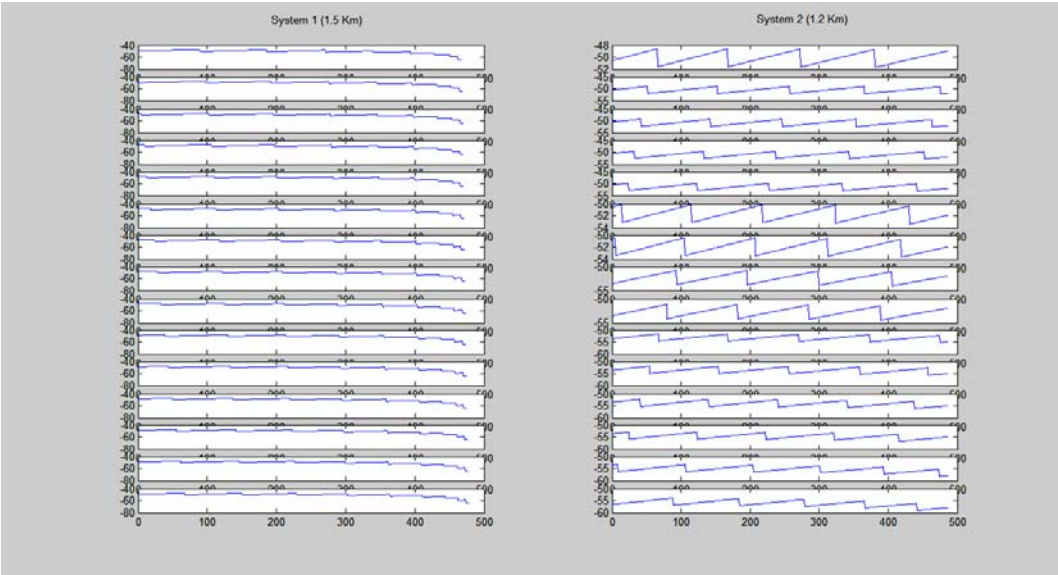


Figure 4.13: *Distribution of PSD in iterative Water-filing. Axis "y" is the PSD in dBm/Hz and axis "x" the subchannels.*

As we can see, the PSD in system 1 (the longest pair of cables) is similar in each iteration and in system 2 (the shortest pair of cables) is decreasing. This is because, when we reduce the PSD in system 2 we decrease the bitrate in system 2 and the FEXT in system 1, and then, we increase the bitrate in system 1. For this reason, in figure 4.12, the graphic for system 1 is increasing and the one for system 2 is decreasing.

5 DSM OPTIMIZATION WITH A PSD CONSTRAINT

5.1 Introduction

In this point, we have added to the program ***waterfill.m*** new code lines that create a PSD constraint in each subchannel. In the file "PSD_Constraint" of the CD, we can see the program *waterfill.m* with these new code lines.

Then, we will define a maximum PSD value. At the beginning we tried it with a constraint of 3dB higher that the PSD of total power. That is:

$$PSD\_\max = \frac{P_{total}}{N_{ch} \cdot \Delta f} \times 10^{\frac{3dB}{10}} = 2 \cdot 10^{-5}\, mW / Hz$$

where "P_{total}" is the total available power, equal to 21mW, "N_{ch}" is the number of subchannels, equal to 487, and "Δf" the length of the subchannels, equal to 4.3125e3 Hz.

Nevertheless, with this constraint the differences in the results were minimal. For this reason, at the end, we used a constraint of 1.5dB higher that the PSD of total power. That is:

$$PSD\_\max = \frac{P_{total}}{N_{ch} \cdot \Delta f} \times 10^{\frac{1.5dB}{10}} = 1.41 \cdot 10^{-5}\, mW / Hz$$

As we will be able to see in the results of this point, with this constraint, surprisingly, we have discovered that we generate better results. This is because with the constraint the power is distributed more uniformly. Thus, we can send more bits. Without the constraint of PSD all the power is consumed in low frequencies, and therefore we do not have any power left for the high frequencies. Then, we can send fewer bits in high frequencies. For this reason, the total number of bits is minor.

Moreover, we will draw the Rate Region and we will compare it with and without PSD constraint. We can see all these results in the next point.

5.2 Results

The system that we will use to generate the results will be two pairs of cables. The first one with 1.5 Km and the second one with 1.2 Km, and the constraint of PSD will be 1.5dB higher than the PSD of total power.

Then, if we obtain the values without constraint of PSD we have the next graphics:

```
>>defxDSL
>>singcap
>>[y,ytot,b,btotal,R] = waterfill(S1,Npsd1,L1,21)
```

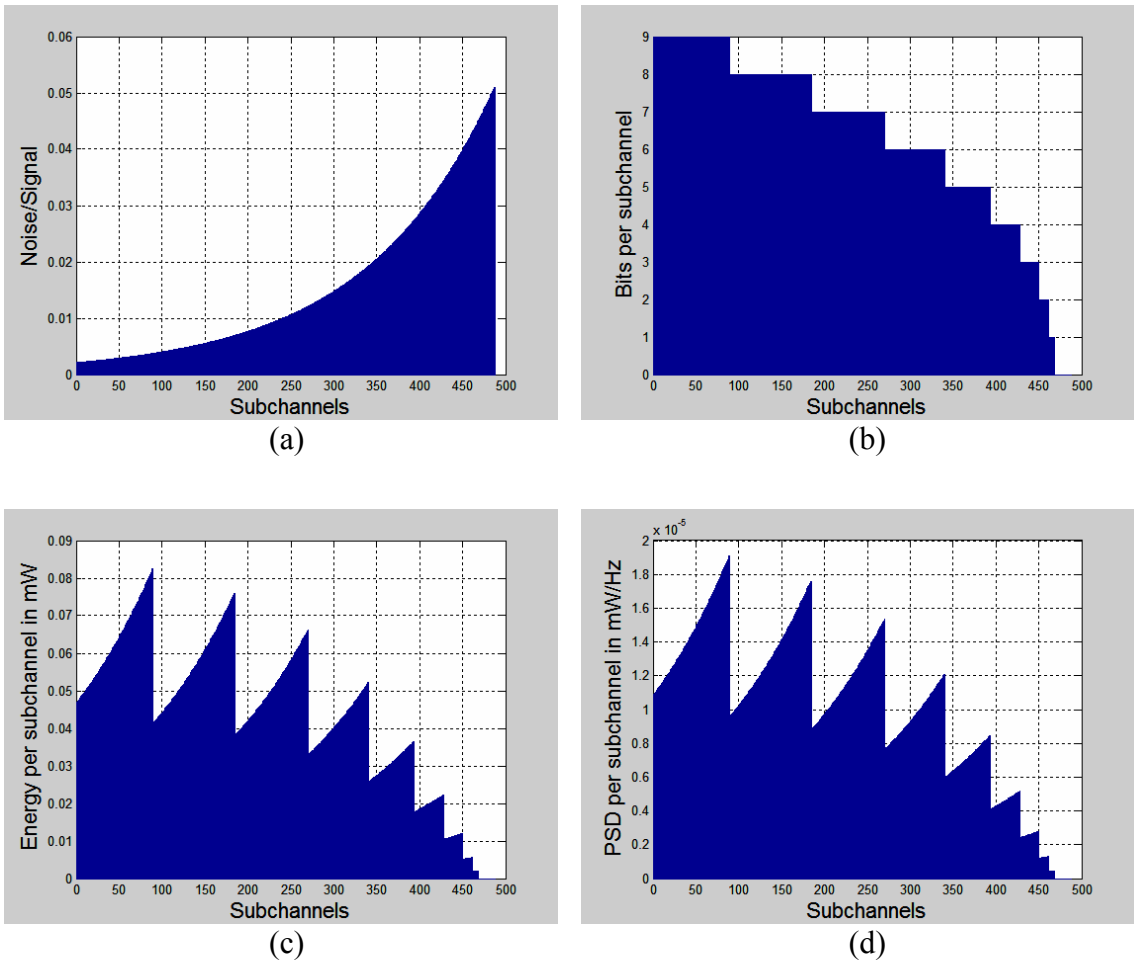


Figure 5.1: *Results for pair one with Water-Filling. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in mW and the PSD per subchannel in mW/Hz respectively.*

The numeric results are:

ytot = 20.9623 mW
btotal = 3082 bits
R = 13.2911 Mbps

As we can see in figure 5.1(d), in some subchannels, PSD is higher than maximum PSD (1.41e-5 mW/Hz). Then, in this case, constraint will affect. If we generate the results and graphics with PSD constraint we have:

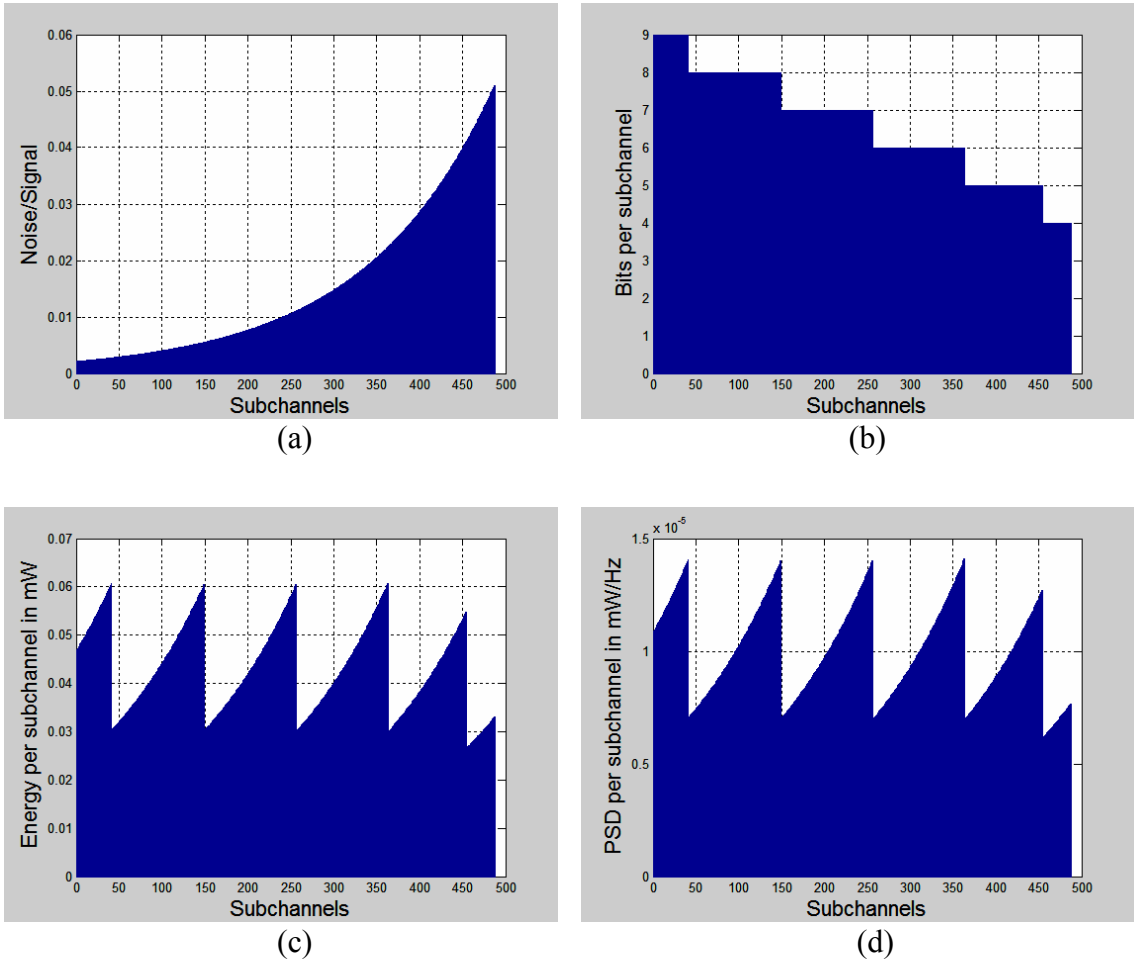


Figure 5.2: *Results for pair one with Water-Filling & PSD constraint. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in mW and the PSD per subchannel in mW/Hz respectively.*

The numeric results are:

ytot = 20.9907 mW
btotal = 3211 bits
R = 13.8474 Mbps

If we compare the last results with the results without PSD constraint we can see that PSD constraint generates higher capacity because the total number of transmitted bits is higher. We can see the reason of this in the graphics 5.1(d) and 5.2(d). With constraint (figure 5.2(d)) the PSD is distributed more uniformly. Nevertheless, without constraint (figure 5.1(d)) all the PSD is consumed in low frequencies, and then, we do not have any power for high frequencies. Then we can send fewer bits in high frequencies when we do not use PSD constraint. We can see it in figure 5.1(b) and 5.2(b).

We have not drawn the results for pair two because we obtain the same results with or without PSD constraint because, in all subchannels, PSD is smaller than the maximum value of PSD defined in the constraint.

In the next simulations, we will always use this PSD constraint in order to optimise the results.

Finally, we can compare the Rate Regions with and without constraint. In figure 5.3 and 5.4 we can see as the results with constraint generate bigger Rate Regions, that is, higher capacities.

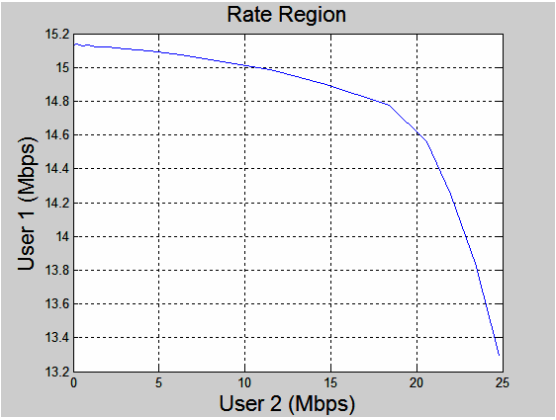


Figure 5.3: *Rate Region without PSD constraint.*

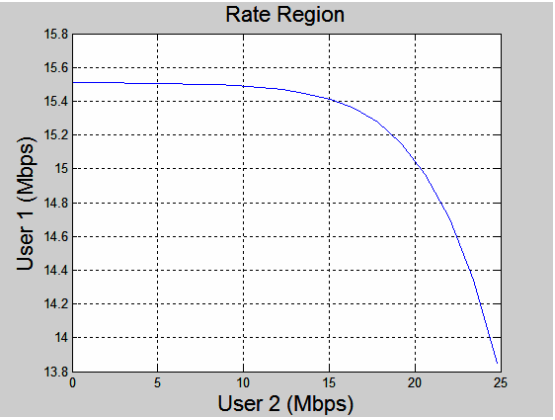


Figure 5.4: *Rate Region with PSD constraints.*

6 DSM USING 3 PAIRS OF CABLES

6.1 Results

Until now, we have studied the results with or without DSM using only two pairs of cables. In this point we will consider one pair more, and then, we will obtain the results using three pairs. Thus, we should define the characteristics of the new pair in the new program *defxDSL.m*, and moreover, we should consider the new contributions of crosstalk (in our case FEXT) from two pairs in the other one. Because of this, we have done changes in all the programs, but mainly in the program *fext.m* to adapt it for the case of three pairs. We can see these programs in the file "DSM_3_pairs" of the CD.

To obtain the next results we have defined three lines (1.5 Km, 1.2Km and 1.2Km respectively) in the program *defxDSL.m*. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT. In this point we have considered FEXT power sum at 1 MHz and 1 km equal to 50.5dB between system 1 and 2, and 2 and 3, but 45dB between systems 1 and 3. Now, we should define six ratios between lengths of disturbed and disturbing pairs. Thus, we have:

LFEdistRat12=80 → between pair 1 and pair 2.
LFEdistRat13=80 → between pair 1 and pair 3.
LFEdistRat21=125 → between pair 2 and pair 1.
LFEdistRat23=100 → between pair 2 and pair 3.
LFEdistRat31=125 → between pair 3 and pair 1.
LFEdistRat32=100 → between pair 3 and pair 2.

The output spectral density is defined as -50dBm/Hz.

On the other hand, in program *fext.m* we should consider the interference between pair 1 and pair 2, pair 1 and pair 3, and pair 2 and pair 3. For this reason the code program will be longer, but the main idea is the same that with only two pairs.

Finally, in the program *waterfill.m* we will use PSD constraint in all our results because as we saw in the last point we will obtain better results and higher capacity.

The results and graphics for pair 1 are:

```
>>defxDSL
>>singcap
>>[y,ytot,b,btotal,R] = waterfill(S1,Npsd1,L1,21)
```

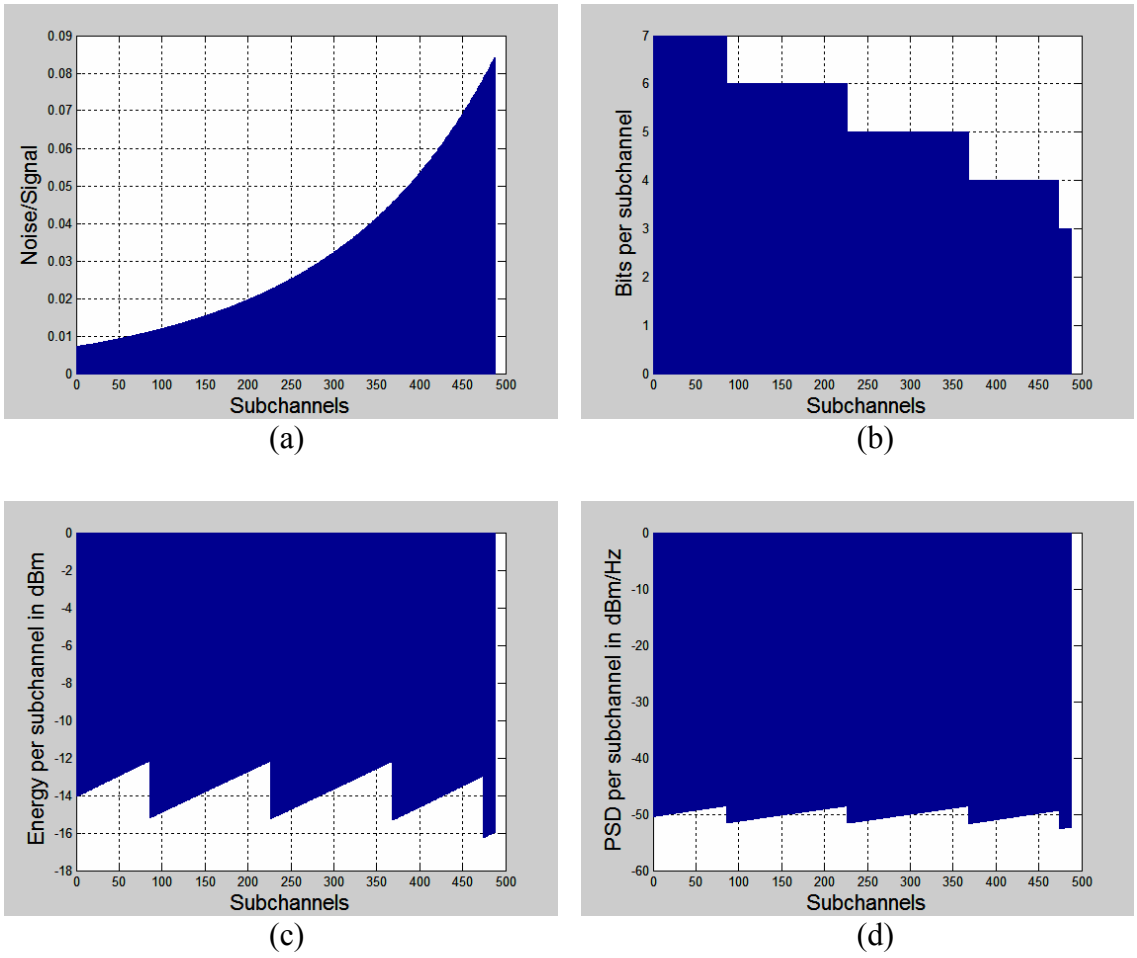


Figure 6.1: *Results for pair one with Water-Filling & PSD constraint. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in dBm and the PSD per subchannel in dBm/Hz respectively.*

ytot = 20.9739 mW
btotal = 2613 bits
R = 11.2686 Mbps

The results and graphics for pair 2 are:

```
>>defxDSL
>>singcap
>>[y,ytot,b,btotal,R] = waterfill(S2,Npsd2,L2,21)
```

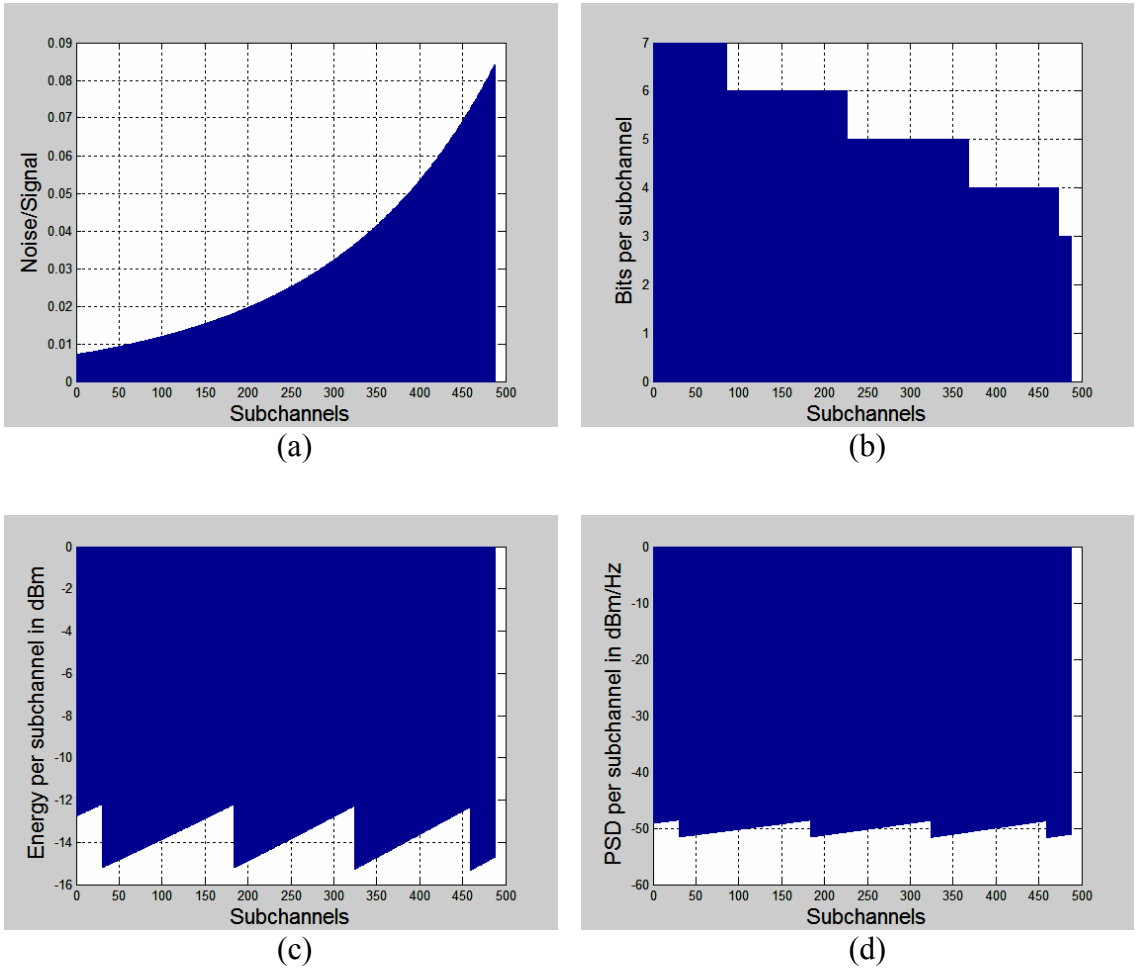


Figure 6.2: *Results for pair two with Water-Filling & PSD constraint. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in dBm and the PSD per subchannel in dBm/Hz respectively.*

ytot = 20.9934 mW
btotal = 5378 bits
R = 23.1926 Mbps

And finally, results and graphics for pair 3 are:

```
>>defxDSL
>>singcap
>>[y,ytot,b,btotal,R] = waterfill(S3,Npsd3,L3,21)
```

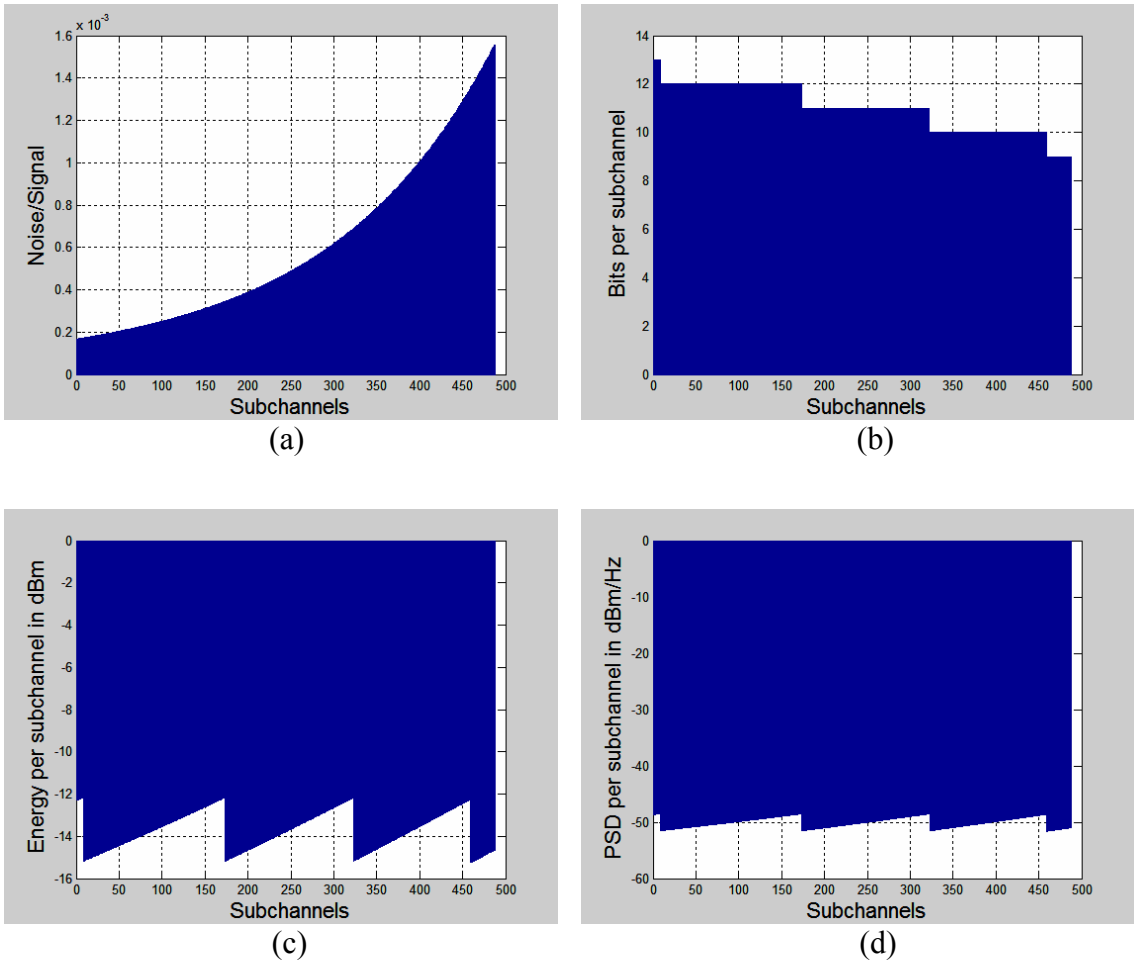


Figure 6.3: *Results for pair three with Water-Filling & PSD constraint. In figure (a) we can see the noise/signal along the subchannels. Figure (b) represents the bits per subchannel and figure (c) and (d) the energy per subchannel in dBm and the PSD per subchannel in dBm/Hz respectively.*

ytot = 20.9986 mW
btotal = 5345 bits
R =23.0503 Mbps

As we can see in the last results, the bitrates are higher for pair two and three, because their lengths are shorter. Moreover, we can see that results for pair two and three are similar but not equal. The reason is because all the parameters in both pairs are equal except the FEXT. In pair two, FEXT is 50.5 dB and in pair three 45 dB. So, the interference that pair two has from pair three is less than the interference that pair three has from pair two. We can prove it mathematically:

If we analyse the system at 4MHz (midpoint of our interval of frequencies in VDSL upstream) we have:

$$Attenuation = 22.5 \cdot \sqrt{4} = 45dB / Km$$

Then, each pair of cables with 1.2 Km will have -54dB of attenuation and the pair with 1.5Km will have -67.5dB. If PSD in the transmitter is -50dBm/Hz, in the receiver there will be:

$$PSD_{receiver\_1.5Km} = -50dBm/Hz + -67.5dB = -117.5dBm/Hz$$
$$PSD_{receiver\_1.2Km} = -50dBm/Hz + -54dB = -104dBm/Hz$$

On the other hand, the crosstalk power transfer function is:

$$X_{Fi,j}(f) = \left|H_{Fi,j}(f)\right|^2 = P_{1Fi,j} \times F^2 \times L$$

and then, the two types of FEXT that we defined were:

$$H_{FEXT\_1}(f) = 10^{\frac{-50.5}{10}} \times 4^2 \times 1.2 \longrightarrow -50.5 + 20\log 4 + 10\log 1.2 = -37.66dB$$

$$H_{FEXT\_3}(f) = 10^{\frac{-45}{10}} \times 4^2 \times 1.2 \longrightarrow -45 + 20\log 4 + 10\log 1.2 = -32.1dB$$

Thus, the crosstalk in each pair of cables with 1.2Km is:

$$N_{FEXT\_2} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 7.1282e-015$$

$$N_{FEXT\_3} = 10^{\frac{(-117.5-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 7.9199e-015$$

We have proved that system 3 has more crosstalk than system 2, and due to this can achieve less capacity.

Notice that we have only calculated the part of noise corresponding to FEXT, but at the end, we must add these values with the White Noise, which is equal for all pairs.

On the other hand, we can obtain the Rate Region, but in the case of three lines, the picture will have three dimensions. Nevertheless, we will not draw the picture in 3D because in it, it is difficult to observe the results. We will draw the Rate Region in two dimensions, firstly between pair one and pair two, and after it, between pair one and pair three. If we draw both graphics in the same picture we obtain figure 6.4:

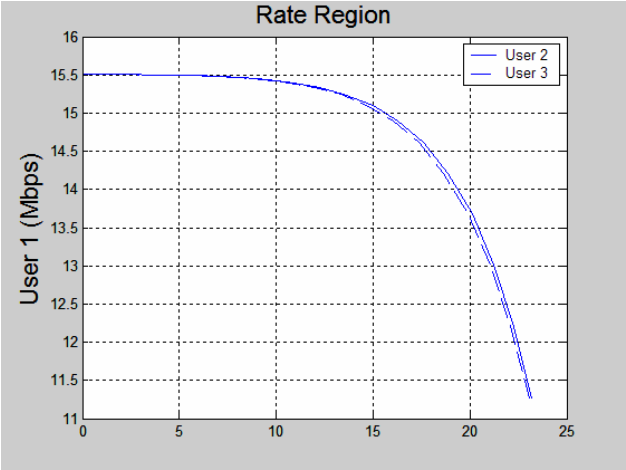


Figure 6.4: *Rate Region for three pairs. The lengths of the pairs are 1.5Km for User 1 and 1.2Km for User 2 and User 3.*

As we can see in the last figure the Rate Region for User 2 and for User 3 are very similar. As we already said, the reason of the small differences is the use of different FEXT value for system two and system three.

On the other hand, the highest value of bitrate for system one is 15.5 Mbps. We can achieve this value when the bitrate in system two and system three is zero, that is, when we decrease the PSD in these pairs, and then, decrease the crosstalk from these pairs in system one. In the same way, the highest value of bitrate for system two and three is around 23 Mbps when all the pairs use -50 dBm/Hz of PSD. Thus, the highest value of bitrate is for pairs two and three, because their lengths are shorter, and for this reason we have decreased the PSD of these pairs in order to obtain the Rate Region.

In the next figure we can see the distribution of the power spectral density in dBm/Hz per subchannel step by step:

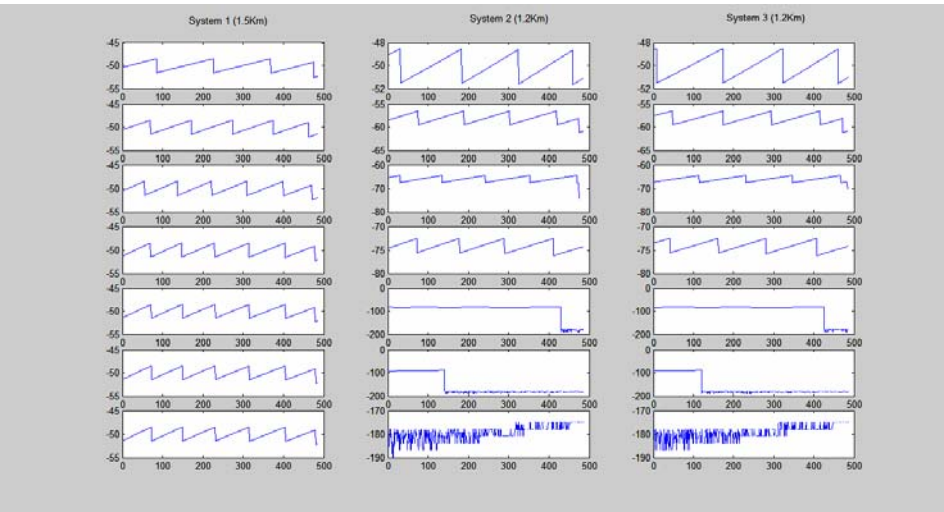


Figure 6.5: *Distribution of PSD to create the Rate Region. Axis "y" is the PSD in dBm/Hz and axis "x" the subchannels.*

As we said, in the first step all the pairs use -50dBm/Hz of PSD, and in the following steps we are decreasing the PSD in pair two and three in order to increase the bitrate in pair one.

On the other hand, we can compare the Rate Region and the results that we have obtained before, with the Rate Region and the results without Water-filling. In figure 6.6 we can see that the shapes of both Rate Regions are the same, but with Water-filling the Rate Region is bigger, and so we achieve higher capacities. If we compare the numeric results we can see that when we use Water-filling we increase the capacity around 2 Mbps.

RESULTS WITHOUT WATER-FILLING:

R1 =9.2749 Mbps

R2 =21.1164 Mbps

R3 =20.9702 Mbps

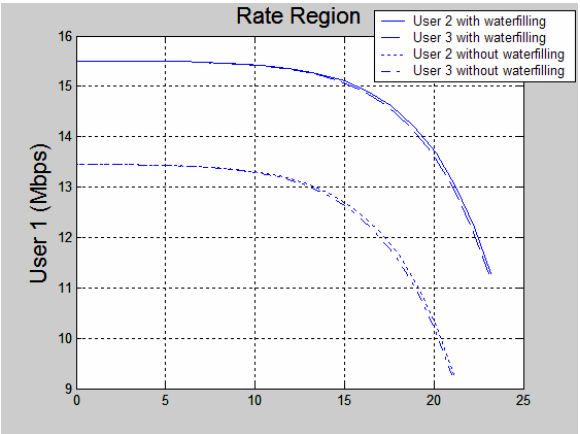


Figure 6.6: *Rate Regions with and without Water-filling.*

7 DSM USING 5 PAIRS OF CABLES

7.1 Results

A pair cable may contain up to 2000 different pairs, which are organised in binder groups from 10 to up to 50 pairs. In this point we will consider five pairs in the binder of cables, and then, we will obtain the results using five pairs. We will have to follow the same steps that we followed for three pairs, but now with more lines and therefore, with more crosstalk contributions.

Then, we should define the characteristics of the new pairs in the new program *defxDSL.m*, and moreover, in program *fext.m* we should consider the new contributions of crosstalk from the new pairs of cables. We can see these programs in the file "DSM_5_pairs" of the CD.

To obtain the results we have defined five lines (1.5 Km, 1.2Km, 1.2Km, 1.2Km and 1.2Km respectively) in the program *defxDSL.m*. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT. In this point we have considered FEXT power sum at 1 MHz and 1 km equal to 50.5dB between pairs 1 and 2, 1 and 4, 2 and 3, 2 and 4, 3 and 4 and finally between 4 and 5. The FEXT power sum between pairs 1 and 3, and between 1 and 5 is 48dB. Finally, the FEXT power sum between pairs 2 and 5, and between 3 and 5 is 45dB. We can see better these values in the next matrix:

$$FEXT_{ps} = \begin{bmatrix} 0 & 50.5 & 48.0 & 50.5 & 48.0 \\ 50.5 & 0 & 50.5 & 50.5 & 45.0 \\ 48.0 & 50.5 & 0 & 50.5 & 45.0 \\ 50.5 & 50.5 & 50.5 & 0 & 50.5 \\ 48.0 & 45.0 & 45.0 & 50.5 & 0 \end{bmatrix}$$

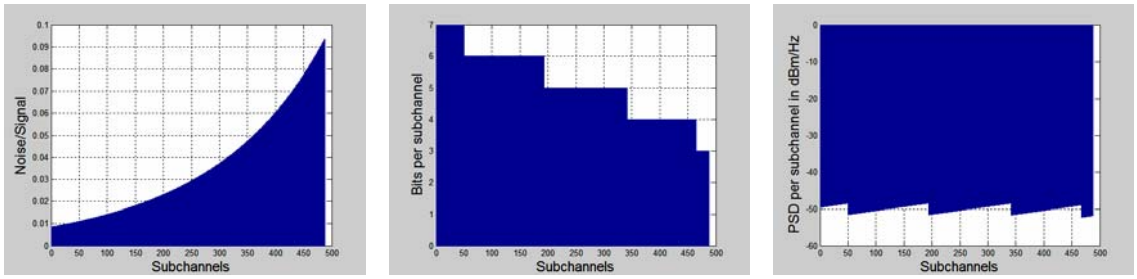
Notice that the matrix should always be symmetrical, because the crosstalk between pair *"i"* and pair *"j"* will be the same that between pair *"j"* and pair *"i"*.

Moreover, we should define twenty ratios between lengths of disturbed pair and disturbing pairs following the expression $L_{disturbing} = L_{disturbed} \times \frac{LFEdistRat}{100}$. We can see these values in the program *defxDSL.m* in the CD.

The output spectral density is defined as -50dBm/Hz.

Finally, we should add more code lines to the program *fext.m* in order to consider the new crosstalk contributions, and in program *waterfill.m* we will continue using PSD constraint.

The results and graphics for pair 1 are:

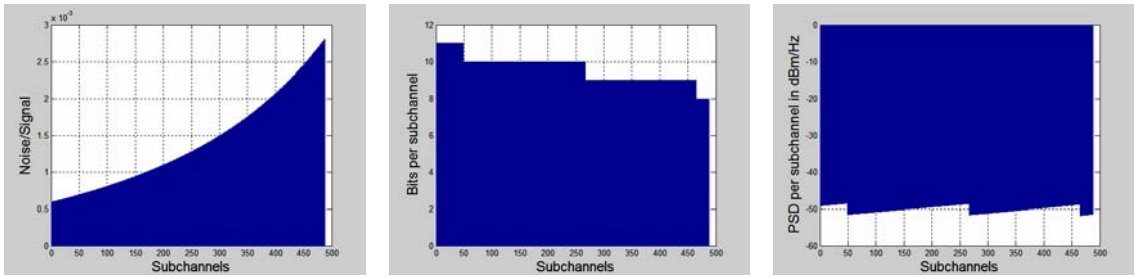


(a) (b) (c)

Figure 7.1: *Results for pair one. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9806 mW
btotal = 2510 bits
R = 10.8244 Mbps

The results and graphics for pair 2 are:

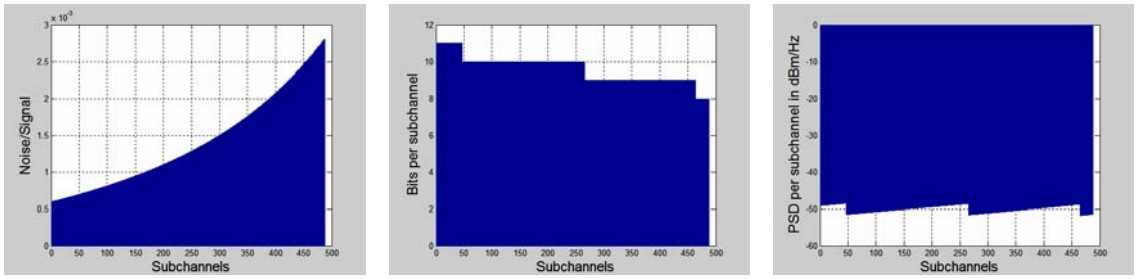


(a) (b) (c)

Figure 7.2: *Results for pair two. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9994 mW
btotal = 4674 bits
R = 20.1566 Mbps

The results and graphics for pair 3 are:

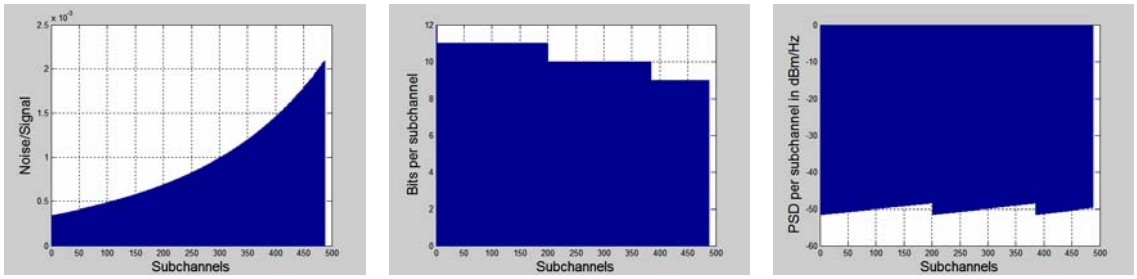


(a) (b) (c)

Figure 7.3: *Results for pair three. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9861 mW
btotal = 4670 bits
R = 20.1394 Mbps

The results and graphics for pair 4 are:

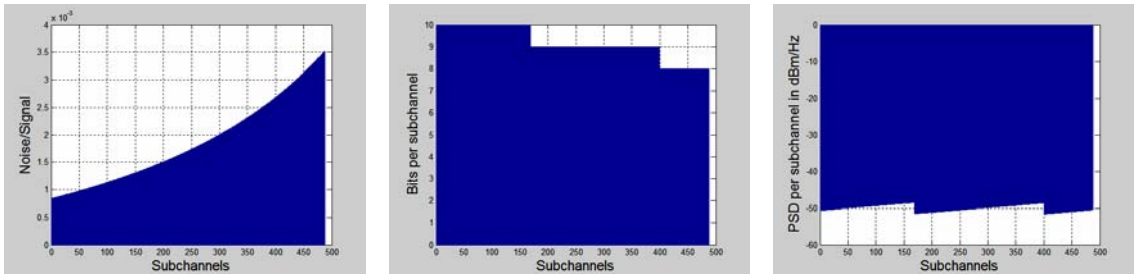


(a) (b) (c)

Figure 7.4: *Results for pair four. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.6449 mW
btotal = 4968 bits
R = 21.4245 Mbps

Finally, the results and graphics for pair 5 are:



(a) (b) (c)

Figure 7.5: *Results for pair five. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9757 mW
btotal = 4463 bits
R = 19.2467 Mbps

As we can see in the last figures, the pair of cables that achieve less capacity is the first one. Moreover, we can see that this pair has the highest Noise-Signal ratio. The reason is because this cable has 1.5Km and all the others have only 1.2Km. Then, the other cables have less attenuation loss.

On the other hand, pairs two, three, four and five have the same length but the results are different for each one. This is because each pair has different FEXT power sum, and for this reason each pair has different Noise-Signal ratio.

Thus, the pair with less capacity is the first because of its length. In the other pairs, the pair with less capacity is pair five, that is, the pair with more crosstalk from the other lines. We can prove it mathematically following the same method used for three pairs:

If we analyse the system at 4MHz we have:

$$Attenuation = 22.5 \cdot \sqrt{4} = 45 dB / Km$$

Then, each pair of cables with 1.2 Km will have -54dB of attenuation and the pair with 1.5Km will have -67.5dB of attenuation. If PSD in the transmitter is -50dBm/Hz, in the receiver there will be:

$$PSD_{receiver\_1.5Km} = -50dBm/Hz + -67.5dB = -117.5dBm/Hz$$

$$PSD_{receiver\_1.2Km} = -50dBm/Hz + -54dB = -104dBm/Hz$$

Following the crosstalk power transfer function we can obtain the three types of FEXT:

$$H_{FEXT\_1}(f) = 10^{\frac{-50.5}{10}} \times 4^2 \times 1.2 \longrightarrow -50.5 + 20\log 4 + 10\log 1.2 = -37.66dB$$

$$H_{FEXT\_2}(f) = 10^{\frac{-48}{10}} \times 4^2 \times 1.2 \longrightarrow -48 + 20\log 4 + 10\log 1.2 = -35.1dB$$

$$H_{FEXT\_3}(f) = 10^{\frac{-45}{10}} \times 4^2 \times 1.2 \longrightarrow -45 + 20\log 4 + 10\log 1.2 = -32.1dB$$

Thus, according to the matrix of FEXT power sum, the crosstalk in each pair of cables with 1.2Km is:

$$N_{FEXT\_2} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-32.1)}{10}} = 3.8499e\text{-}14$$

$$N_{FEXT\_3} = 10^{\frac{(-117.5-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-32.1)}{10}} = 3.8743e\text{-}14$$

$$N_{FEXT\_4} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 2.0775e\text{-}14$$

$$N_{FEXT\_5} = 10^{\frac{(-117.5-35.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 5.6467e\text{-}14$$

Thus, we have proved that system 5 has more crosstalk than system 2, 3 or 4, and for this reason can achieve less capacity. Moreover we can see that system 4 has the least crosstalk, and so this pair achieves the highest capacity.

Finally, we have represented the Rate Region with and without Water-filling. In figure 7.6 we can see the graphics and the difference between both methods. The shapes of both Rate Regions are the same, but with Water-filling the Rate Region is bigger, as we expected. Moreover, we have generated the results without Water-filling, and as we can see below the capacity is around 2Mbps lower. The results without Water-filling are:

R1 =8.8457 Mbps **R2 =18.0833 Mbps** **R3 =18.0675 Mbps**

R4 =19.3903 Mbps **R5 =17.1678 Mbps**

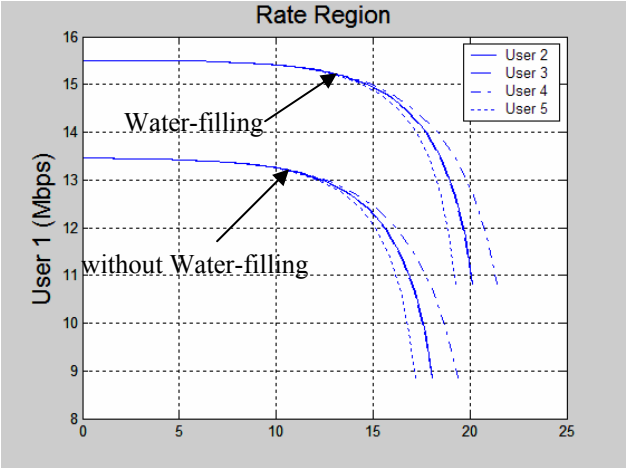


Figure 7.6: *Rate Regions with and without Water-filling.*

7.2 Realistic results using Statistical crosstalk model

In this point, we will generate the results for five pairs again, but now the results will be realistic, because all parameters will be based on measurements. Then, we will use a statistical model for crosstalk in twisted pair cables, where all individual pair combinations in the cable are modelled separately.

First, we should define the FEXT power sum matrix. In this case all the values of this matrix will have been obtained with measurements in a real system. Moreover, we will use a new crosstalk power transfer function where we will introduce a random variable. The new expression will be:

$$X_{Fi,j}(f) = \left|H_{Fi,j}(f)\right|^2 = x_F \times P_{1Fi,j} \times F^2 \times L$$

where "x_F" is a gamma distributed random variable with $v = 0.5$ and unity mean. This random variable is generated in Matlab by x_F = *randn*^2.

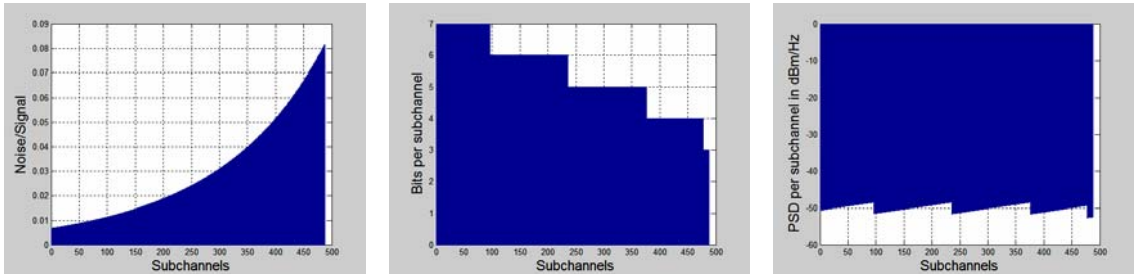
The other parameters are the same that we have used until now.

In the program called ***FEXT_rand.m*** (see Appendix or file "DSM_5_pairs_real" in the CD) we have defined all these parameters, that is, the matrix of FEXT power sum and the random variable. Attention must be paid to the fact that the matrix should be symmetrical. Moreover we have done small changes in program *fext.m* in order to adapt the program to the new expression of crosstalk power transfer.

The system that we have studied is the same that we studied in the last point, that is, five lines (1.5 Km, 1.2Km, 1.2Km, 1.2Km and 1.2Km respectively). The other parameters are the same, but the FEXT power sum matrix is the following:

$$FEXT_{ps} = \begin{bmatrix} 0 & 45.8 & 50.4 & 52.7 & 53.7 \\ 45.8 & 0 & 48.0 & 50.0 & 52.6 \\ 50.4 & 48.0 & 0 & 47.6 & 51.3 \\ 52.7 & 50.0 & 47.6 & 0 & 47.3 \\ 53.7 & 52.6 & 51.3 & 47.3 & 0 \end{bmatrix}$$

Then, if we generate the results and graphics for system 1 we have:

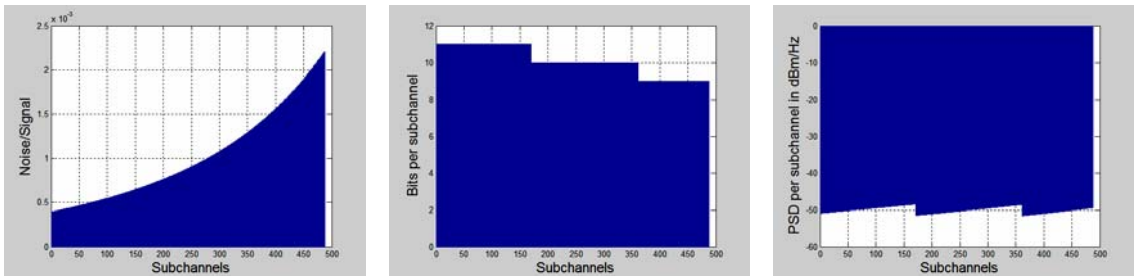


(a) (b) (c)

Figure 7.7: *Realistic results for pair one. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot =20.9842 mW
btotal =2645 bits
R =11.4066 Mbps

The results and graphics for pair 2 are:

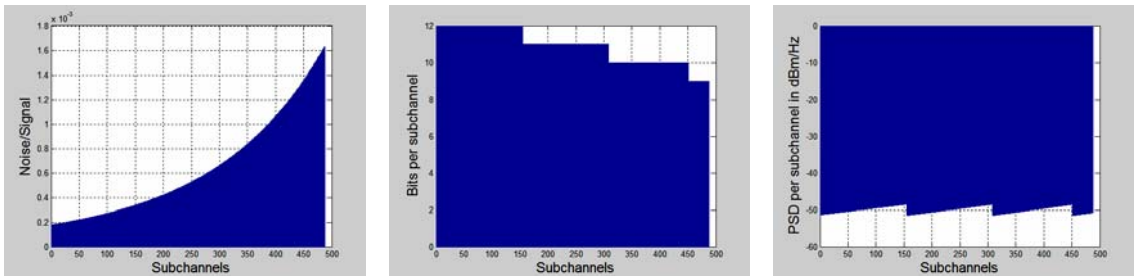


(a) (b) (c)

Figure 7.8: *Realistic results for pair two. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot =20.9775 mW
btotal =4914 bits
R =21.1916 Mbps

The results and graphics for pair 3 are:

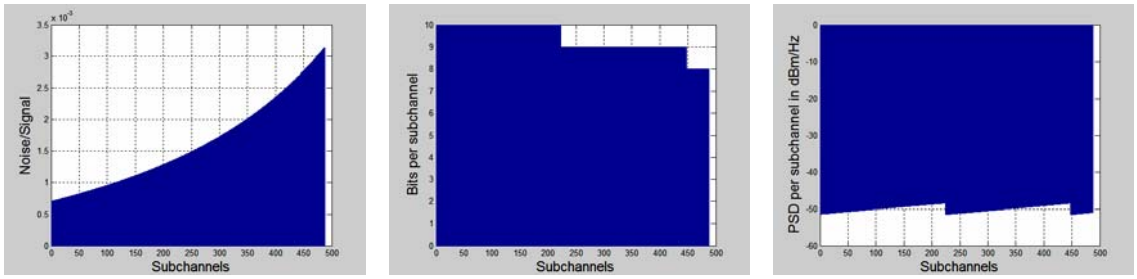


(a) (b) (c)

Figure 7.9: *Realistic results for pair three. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot =20.9910 mW
btotal =5296 bits
R =22.8390 Mbps

The results and graphics for pair 4 are:

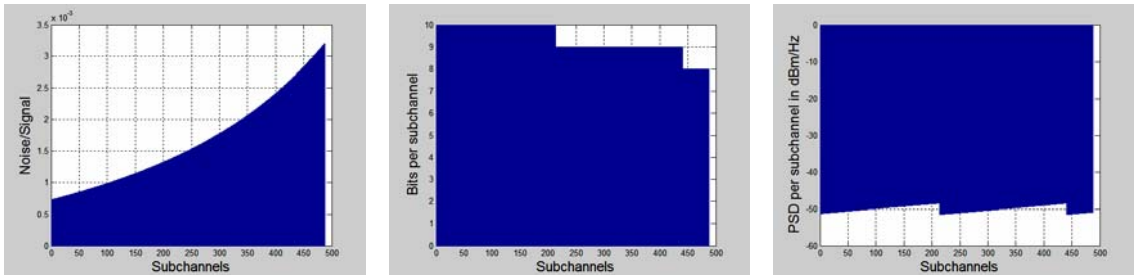


(a) (b) (c)

Figure 7.10: *Realistic results for pair four. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot =20.9182 mW
btotal =4566 bits
R =19.6909 Mbps

Finally, the results and graphics for pair 5 are:



(a) (b) (c)

Figure 7.11: *Realistic results for pair five. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot =20.9901 mW
btotal =4549 bits
R =19.6176 Mbps

As we can see, the results that we have obtained are similar to the results of the last point. The differences are caused by the uses of new FEXT power sum matrix and the random variable.

On the other hand we have generated the Rate Region. If we generate the Rate Region with Water-filling and without Water-filling with different programs, each program would use a different value of the random variable, and we could not compare both graphics. For this reason we have created a new program that obtains the Rate Region with and without Water-filling at the same time, in order to compare both. In figure 7.12 we can see both graphics:

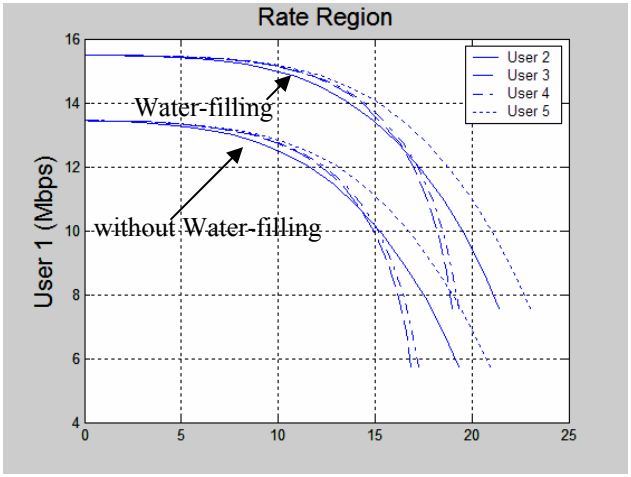


Figure 7.12: *Realistic Rate Regions with and without Water-filling.*

8 DSM USING 10 PAIRS OF CABLES

8.1 Results

Until now, we have seen the results considering two, three and five pair of cables. Nevertheless, we know that a pair cable may contain up to 2000 different pairs, which are organised in binder groups from 10 to up to 50 pairs. Thus, in this point, we will consider ten pairs in the binder of cables in order to obtain more realistic results. The steps we will follow will be the same, but now with more lines and crosstalk contributions. We can see the programs that we have used in the file "DSM_10_pairs" of the CD.

Thus, we have defined ten lines (1.5 Km for the first pair, and 1.2Km for the others nine pairs) in the program *defxDSL.m*. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT. The FEXT power sum matrix at 1 MHz and 1 km is the following:

$$FEXT_{ps} = \begin{bmatrix} 0 & 50.5 & 45.0 & 50.5 & 45.0 & 45.0 & 50.5 & 50.5 & 45.0 & 50.5 \\ 50.5 & 0 & 50.5 & 50.5 & 48.0 & 50.5 & 50.5 & 48.0 & 45.0 & 50.5 \\ 45.0 & 50.5 & 0 & 50.5 & 48.0 & 50.5 & 50.5 & 45.0 & 48.0 & 50.5 \\ 50.5 & 50.5 & 50.5 & 0 & 50.5 & 50.5 & 45.0 & 48.0 & 50.5 & 50.5 \\ 45.0 & 48.0 & 48.0 & 50.5 & 0 & 50.5 & 50.5 & 48.0 & 45.0 & 50.5 \\ 45.0 & 50.5 & 50.5 & 50.5 & 50.5 & 0 & 48.0 & 45.0 & 50.5 & 50.5 \\ 50.5 & 50.5 & 50.5 & 45.0 & 50.5 & 48.0 & 0 & 50.5 & 50.5 & 50.5 \\ 50.5 & 48.0 & 45.0 & 48.0 & 48.0 & 45.0 & 50.5 & 0 & 50.5 & 50.5 \\ 45.0 & 45.0 & 48.0 & 50.5 & 45.0 & 50.5 & 50.5 & 50.5 & 0 & 50.5 \\ 50.5 & 50.5 & 50.5 & 50.5 & 50.5 & 50.5 & 50.5 & 50.5 & 50.5 & 0 \end{bmatrix}$$

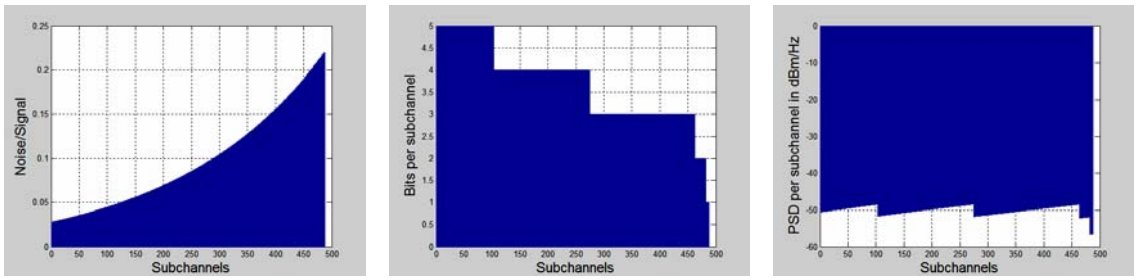
As we know, the matrix should be symmetric always, because, the crosstalk between pair *"i"* and pair *"j"* will be the same that between pair *"j"* and pair *"i"*. In the last matrix we have used three types of FEXT power sum: 50.5dB, 48dB and 45dB.

The next step is to define ninety ratios between lengths of disturbed pairs and disturbing pairs following the expression $L_{disturbing} = L_{disturbed} \times \frac{LFEdistRat}{100}$. We can see these values in the program *defxDSL.m* in the CD.

The output spectral density is defined as -50dBm/Hz.

Finally, we should change the program *fext.m* in order to consider the new crosstalk contributions, and in program *waterfill.m* we will continue using PSD constraint.

The results and graphics for pair 1 are:

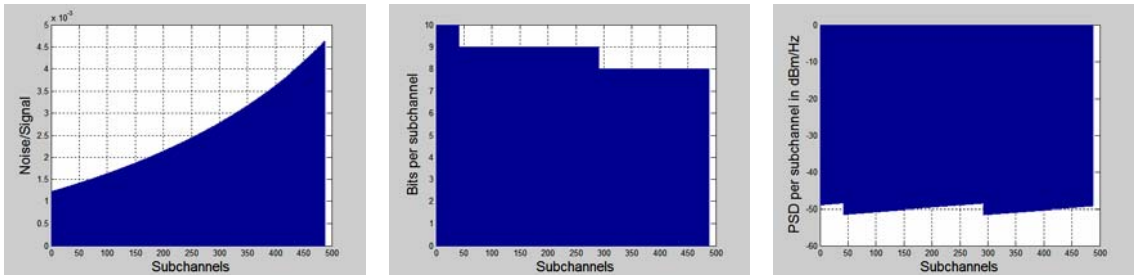


(a) (b) (c)

Figure 8.1: *Results for pair one. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9696 mW
btotal = 1808 bits
R = 7.7970 Mbps

The results and graphics for pair 2 are:

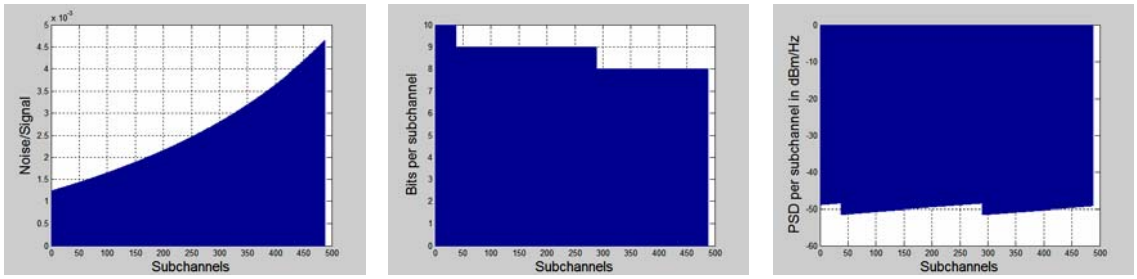


(a) (b) (c)

Figure 8.2: *Results for pair two. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9749 mW
btotal = 4227 bits
R = 18.2289 Mbps

The results and graphics for pair 3 are:

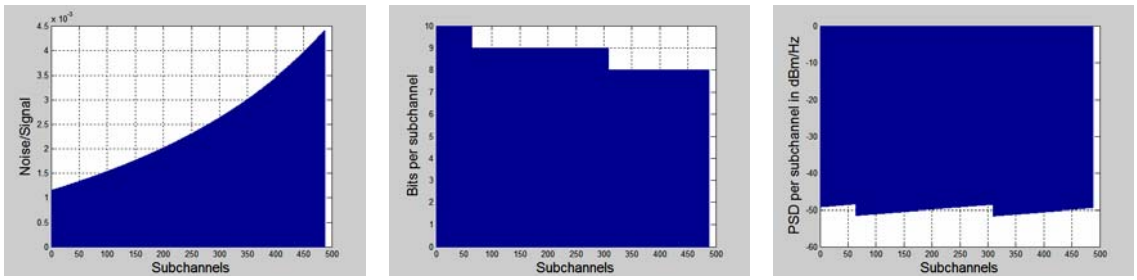


(a) (b) (c)

Figure 8.3: *Results for pair three. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9703 mW
btotal = 4221 bits
R = 18.2031 Mbps

The results and graphics for pair 4 are:

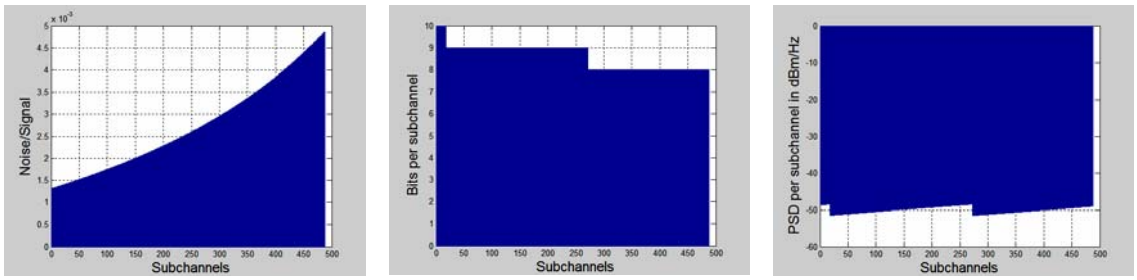


(a) (b) (c)

Figure 8.4: *Results for pair four. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9732 mW
btotal = 4267 bits
R = 18.4014 Mbps

The results and graphics for pair 5 are:

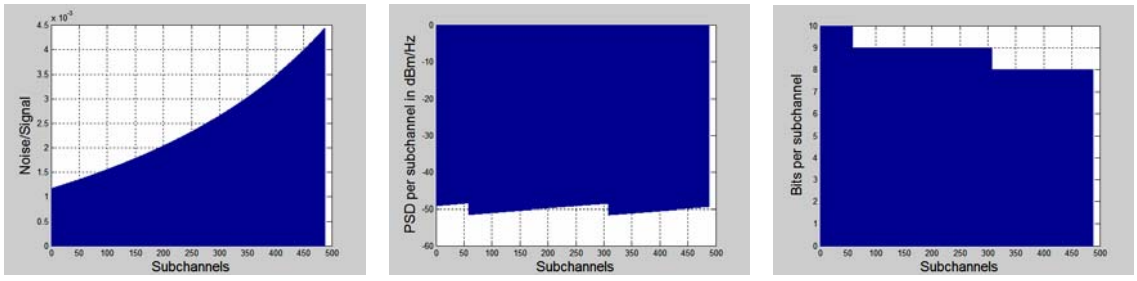


(a) (b) (c)

Figure 8.5: *Results for pair five. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9784 mW
btotal = 4184 bits
R = 18.0435 Mbps

The results and graphics for pair 6 are:

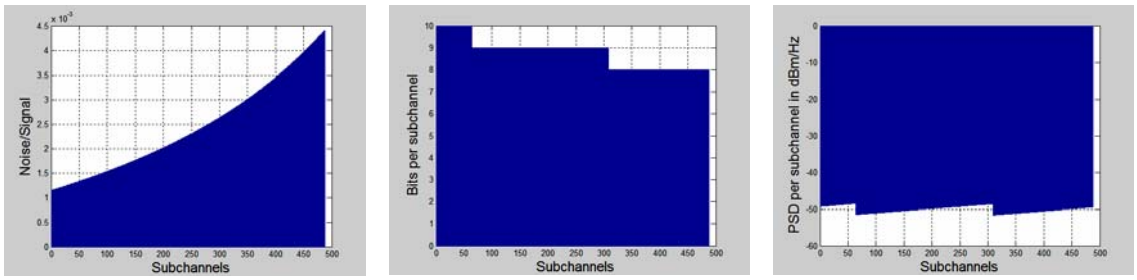


(a) (b) (c)

Figure 8.6: *Results for pair six. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9808 mW
btotal = 4261 bits
R = 18.3756 Mbps

The results and graphics for pair 7 are:

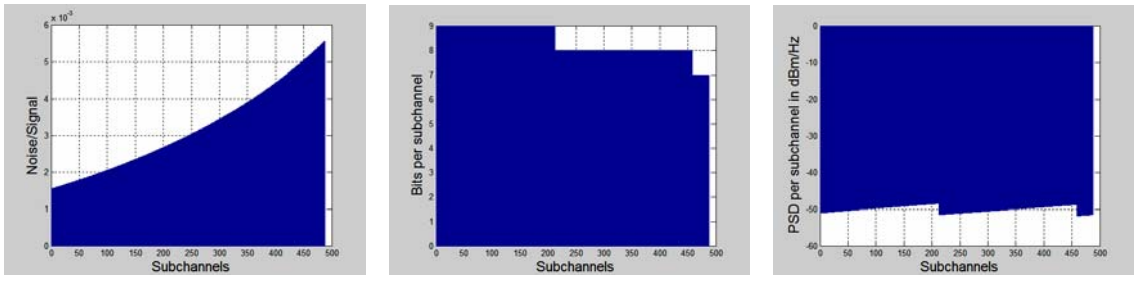


(a) (b) (c)

Figure 8.7: *Results for pair seven. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9732 mW
btotal = 4267 bits
R = 18.4014 Mbps

The results and graphics for pair 8 are:

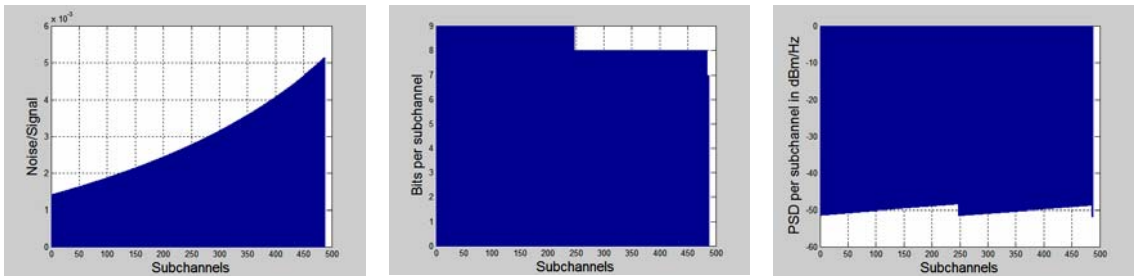


(a) (b) (c)

Figure 8.8: *Results for pair eight. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9984 mW
btotal = 4078 bits
R = 17.5864 Mbps

The results and graphics for pair 9 are:

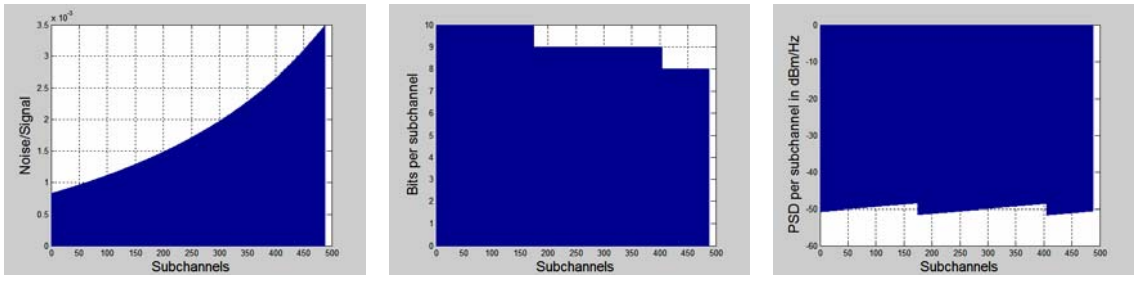


(a) (b) (c)

Figure 8.9: *Results for pair nine. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9991 mW
btotal = 4140 bits
R = 17.8538 Mbps

The results and graphics for pair 10 are:



(a) (b) (c)

Figure 8.10: *Results for pair ten. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9929 mW
btotal = 4473 bits
R = 19.2898 Mbps

As we know, the pair of cables that achieve less capacity is the first because this cable is 1.5Km and all the others are only 1.2Km, and therefore they have less attenuation. Thus, we can see how the first pair has the highest Noise-Signal ratio.

On the other hand, the other nine pairs have different because each pair has different FEXT power sum.

Thus, as we saw when we consider five pairs only, the pair with less capacity is the first because of its length. In the other pairs, the pair with less capacity is pair eight, that is, the pair with more crosstalk from the other lines. Next, we will prove it mathematically:

As we know, each pair of cables with 1.2 Km will have -54dB of attenuation and the pair with 1.5Km will have -67.5dB of attenuation. The PSD in the receiver will be:

$$PSD_{receiver\_1.5Km} = -50dBm/Hz + -67.5dB = -117.5dBm/Hz$$

$$PSD_{receiver\_1.2Km} = -50dBm/Hz + -54dB = -104dBm/Hz$$

The three types of FEXT are:

$$H_{FEXT\_1}(f) = 10^{\frac{-50.5}{10}} \times 4^2 \times 1.2 \longrightarrow -50.5 + 20\log 4 + 10\log 1.2 = -37.66dB$$

$$H_{FEXT\_2}(f) = 10^{\frac{-48}{10}} \times 4^2 \times 1.2 \longrightarrow -48 + 20\log 4 + 10\log 1.2 = -35.1dB$$

$$H_{FEXT\_3}(f) = 10^{\frac{-45}{10}} \times 4^2 \times 1.2 \longrightarrow -45 + 20\log 4 + 10\log 1.2 = -32.1dB$$

Thus, according to the matrix of FEXT power sum, the crosstalk in each pair of cables with 1.2Km is:

$$N_{FEXT\_2} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 8.3574e-014$$

$$N_{FEXT\_3} = 10^{\frac{(-117.5-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 8.4366e-014$$

$$N_{FEXT\_4} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 7.8095e-014$$

$$N_{FEXT\_5} = 10^{\frac{(-117.5-32.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 8.9845e-014$$

$$N_{FEXT\_6} = 10^{\frac{(-117.5-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 7.8887e-014$$

$$N_{FEXT\_7} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 7.8095e-014$$

$$N_{FEXT\_8} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 1.0678e-013$$

$$N_{FEXT\_9} = 10^{\frac{(-117.5-32.1)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-35.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-32.1)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 9.6610e-014$$

$$N_{FEXT\_10} = 10^{\frac{(-117.5-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} + 10^{\frac{(-104-37.66)}{10}} = 5.4892e-014$$

We have proved that system 8 has more crosstalk than the other pairs with 1.2Km. On the other hand, system 10 has the least crosstalk, and so this pair achieves the highest capacity.

Finally, in figure 8.11 we can see the graphics of the Rate Regions with and without Water-filling and the difference between both methods. As expected, the Rate

Region with Water-filling is bigger. Next, we present the numeric results without Water-filling. In this case the results are more reliable and more accurate to the reality because we have used ten pairs, and actually the pairs are organised in binder groups of 10 to 50. As we can see the conclusions are the same that we achieved with three or five pairs, that is, the capacity is around 2Mbps lower when we do not use Water-filling.

R1 =5.9699 Mbps R2 =16.1530 Mbps R3 =16.1280 Mbps R4 =16.3272 Mbps
R5 =15.9658 Mbps R6 =16.3007 Mbps R7 =16.3272 Mbps R8 =15.5087 Mbps
R9 =15.7709 Mbps R10 =17.2094 Mbps

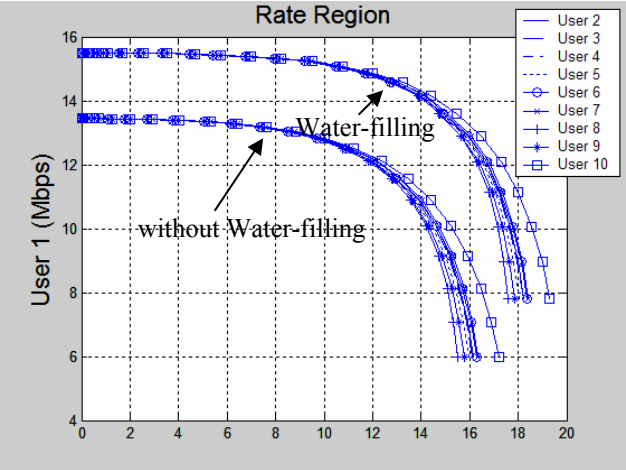


Figure 8.11: *Rate Regions with and without Water-filling.*

8.2 Realistic results using Statistical crosstalk model

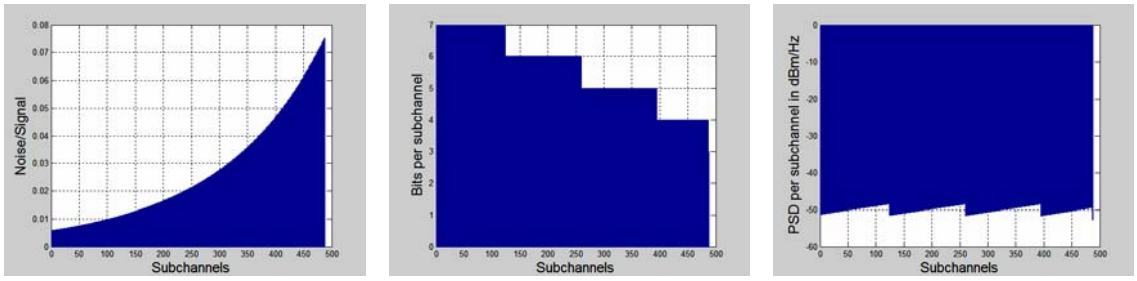
As we already did for five pairs of cables, in this point we will generate the results for ten pairs again using a statistical model for crosstalk in twisted pair cables, in order to obtain realistic results.

Then, we should define the FEXT power sum matrix based on measurements in a real system, and we will use the new crosstalk power transfer function where we introduce the random variable. This expression is the same that we saw already for five pairs. The other parameters are the same that we have used until now.

In the program called ***FEXT_rand.m*** (see Appendix or file "DSM_10_pairs_real" in the CD) we have defined the matrix of FEXT power sum and the random variable. The symmetrical matrix is the following:

$$FEXT_{ps} = \begin{bmatrix} 0 & 45.8 & 50.4 & 52.7 & 53.7 & 55.4 & 55.7 & 56.0 & 54.9 & 52.0 \\ 45.8 & 0 & 48.0 & 50.0 & 52.6 & 54.8 & 54.6 & 54.9 & 51.5 & 50.6 \\ 50.4 & 48.0 & 0 & 47.6 & 51.3 & 54.6 & 55.3 & 50.3 & 50.0 & 54.1 \\ 52.7 & 50.0 & 47.6 & 0 & 47.3 & 51.4 & 50.8 & 49.9 & 54.4 & 55.7 \\ 53.7 & 52.6 & 51.3 & 47.3 & 0 & 46.6 & 49.7 & 52.9 & 55.3 & 55.5 \\ 55.4 & 54.8 & 54.6 & 51.4 & 46.6 & 0 & 47.2 & 51.9 & 54.3 & 56.1 \\ 55.7 & 54.6 & 55.3 & 50.8 & 49.7 & 47.2 & 0 & 48.0 & 51.9 & 54.1 \\ 56.0 & 54.9 & 50.3 & 49.9 & 52.9 & 51.9 & 48.0 & 0 & 47.8 & 51.8 \\ 54.9 & 51.5 & 50.0 & 54.4 & 55.3 & 54.3 & 51.9 & 47.8 & 0 & 47.5 \\ 52.0 & 50.6 & 54.1 & 55.7 & 55.5 & 56.1 & 54.1 & 51.8 & 47.5 & 0 \end{bmatrix}$$

Then, if we generate the results and graphics for system 1 we have:

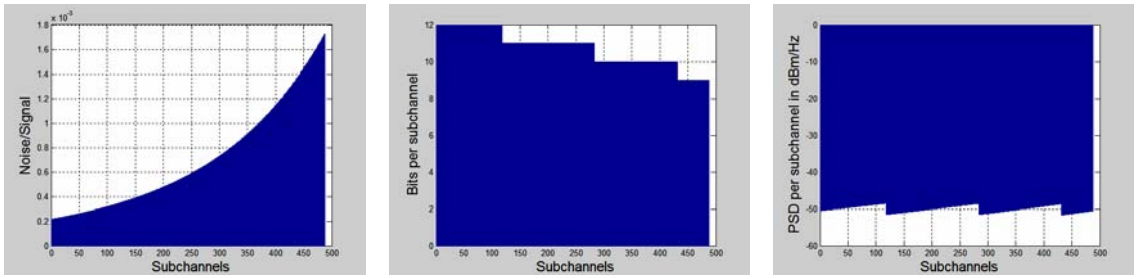


(a) (b) (c)

Figure 8.12: *Realistic results for pair one. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.8491 mW
btotal = 2723 bits
R = 11.7429 Mbps

The results and graphics for pair 2 are:

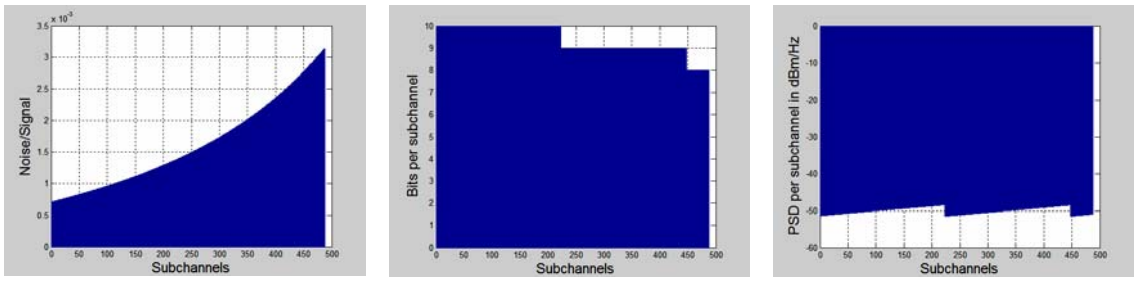


(a) (b) (c)

Figure 8.13: *Realistic results for pair two. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9835 mW
btotal = 5215 bits
R = 22.4897 Mbps

The results and graphics for pair 3 are:

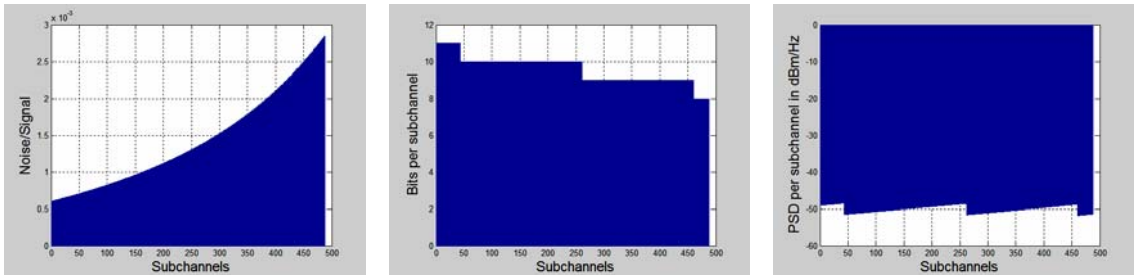


(a) (b) (c)

Figure 8.14: *Realistic results for pair three. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9590 mW
btotal = 4565 bits
R = 19.6866 Mbps

The results and graphics for pair 4 are:

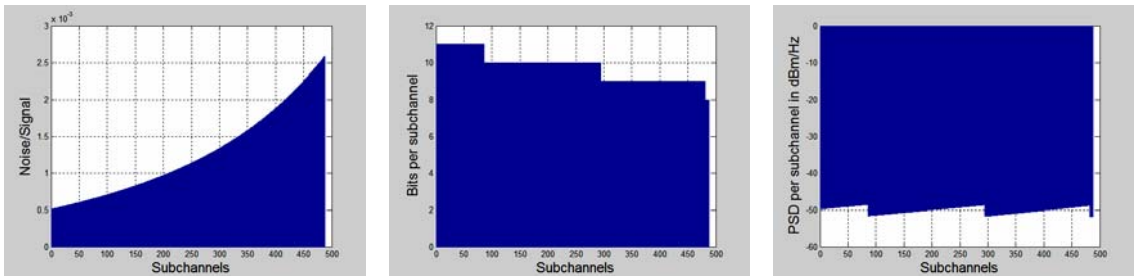


(a) (b) (c)

Figure 8.15: *Realistic results for pair four. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9985 mW
btotal = 4660 bits
R = 20.0963 Mbps

The results and graphics for pair 5 are:

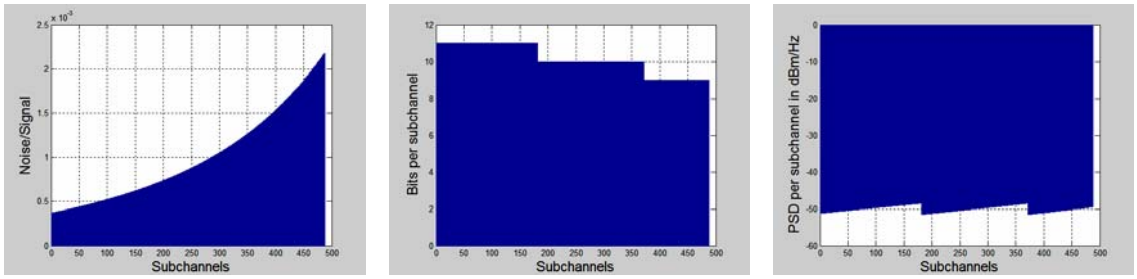


(a) (b) (c)

Figure 8.16: *Realistic results for pair five. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9716 mW
btotal = 4756 bits
R = 20.5103 Mbps

The results and graphics for pair 6 are:

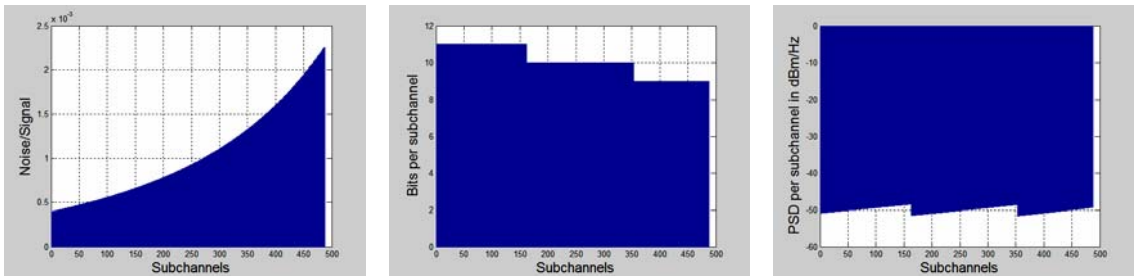


(a) (b) (c)

Figure 8.17: *Realistic results for pair six. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9234 mW
btotal = 4935 bits
R = 21.2822 Mbps

The results and graphics for pair 7 are:

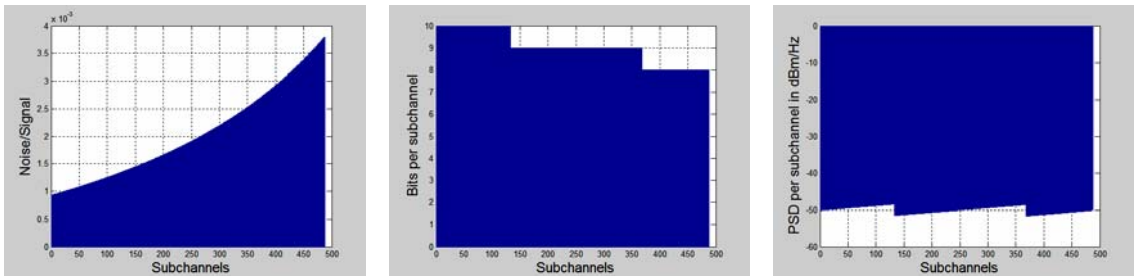


(a) (b) (c)

Figure 8.18: *Realistic results for pair seven. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9758 mW
btotal = 4898 bits
R = 21.1226 Mbps

The results and graphics for pair 8 are:

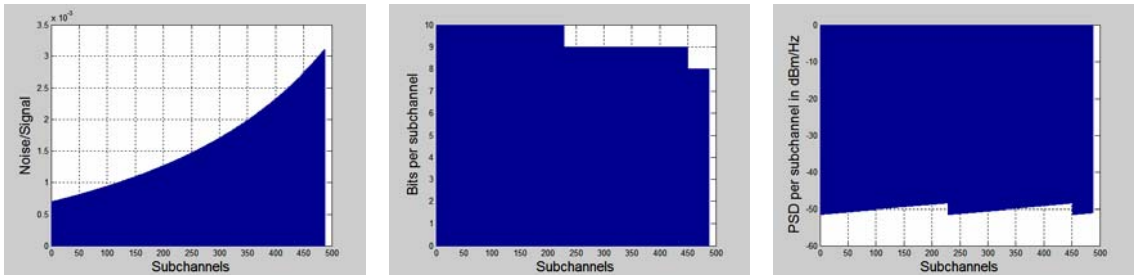


(a) (b) (c)

Figure 8.19: *Realistic results for pair eight. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9861 mW
btotal = 4397 bits
R = 18.9621 Mbps

The results and graphics for pair 9 are:

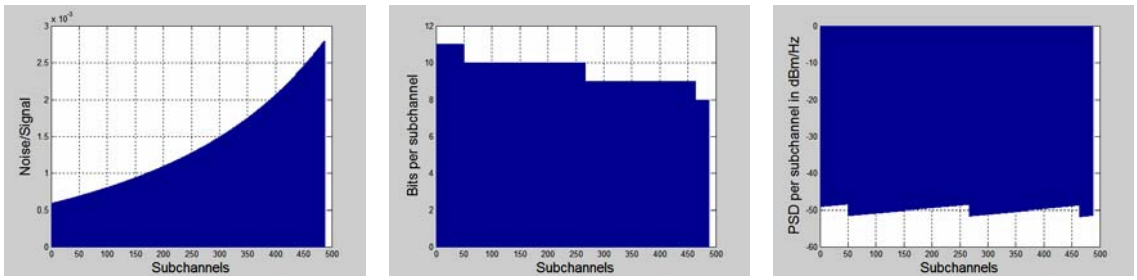


(a) (b) (c)

Figure 8.20: *Realistic results for pair nine. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.8988 mW
btotal = 4574 bits
R = 19.7254 Mbps

The results and graphics for pair 10 are:



(a) (b) (c)

Figure 8.21: *Realistic results for pair ten. Figure (a) is the noise/signal along the subchannels. Figure (b) is the bits per subchannel and figure (c) the PSD per subchannel.*

ytot = 20.9727 mW
btotal = 4675 bits
R = 20.1609 Mbps

As we can see, the results that we have obtained are a little higher than the results of the last point. The differences are because of the usages of the new FEXT power sum matrix and of the random variable.

On the other hand we have generated the Rate Region with and without Water-filling using the same value of the random variable for each one. In figure 8.22 we can see both graphics. As we expected, the Rate Region with Water-filling is around 2Mbps bigger than without Water-filling.

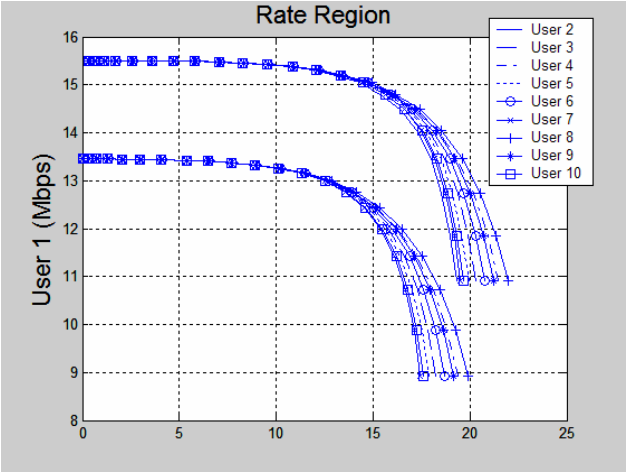


Figure 8.22: *Realistic Rate Regions with and without Water-filling*

9 MAXIMIZATION SUM OF BITRATES

9.1 Description of the program

Now, we will describe the program that we have created in order to maximize the sum of bitrates in all the pairs. The program is called ***iterwaterfill_maxsum.m*** (see Appendix and file "Max_Sum_5_pairs" or file "Max_Sum_10_pairs" in the CD).

The input of the program is the maximum power in mW that we have in each cable, and the outputs are the capacity and the transmitted energy that we have in each pair at the end. Then, at the beginning of the program we define the maximum power for each cable and we run the Water-filling. Thus, we will obtain the bitrates for each pair, as well as the sum of all.

The next step is to repeat the same process that we have done with maximum power, but now reducing the power in each pair every time, obtaining and storing the results with Water-filling. At the end, considering five pairs, we will have five different sums of bitrates (the sum when we reduce the power in system one, the sum when we reduce the power in system two, etc…). Then, we compare the five sums of bitrates and we take the results that maximize the sum of bitrates. Finally, we update the power in the pair that generated the maximum sum when we reduced its power.

The last process is done in the curl "*for,* but this curl is inside the curl *"while"* because we should repeat the process while we find values of the sum higher than in the last iteration. When we finish all the process we will have found the combination of power for each pair that generates the maximum sum of bitrates.

On the other hand, we have created another program called ***iter_maxsum.m*** (see file "Max_Sum_5_pairs" or file "Max_Sum_10_pairs" in the CD). It does the same as the previous program, but without Water-filling.

In the next point we will describe the system that we have used to obtain the results, and we will generate the results for five and ten pairs.

9.2 Results

9.2.1 Results for 5 pairs

To obtain the results we have defined five lines (1.5 Km, 1.4Km, 1.3Km, 1.2Km and 1.1Km respectively) in the program *defxDSL.m*. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT. The FEXT power sum matrix is the same that we used to obtain the realistic results. The problem is that if we used the gamma distributed random variable when we run *iterwaterfill_maxsum.m* and when we run *iter_maxsum.m*, the variable will have different value for each program, and then, we could not compare the results with and without Water-filling. For this reason we have created another program called ***comparative_maxsum.m*** (see file "Max_Sum_5_pairs" in the CD) that generates simultaneously the results with and without Water-filling. Thus, we will be able to compare both results.

Moreover, we should define the ratios between lengths of disturbed pairs and disturbing pairs, and the output spectral density as -50dBm/Hz. We can see these values in the program *defxDSL.m* in the CD. All the other programs are the same that we used when we generated the realistic results for five pairs.

Then, the graphic with Water-filling using the random variable and the realistic FEXT power matrix is:

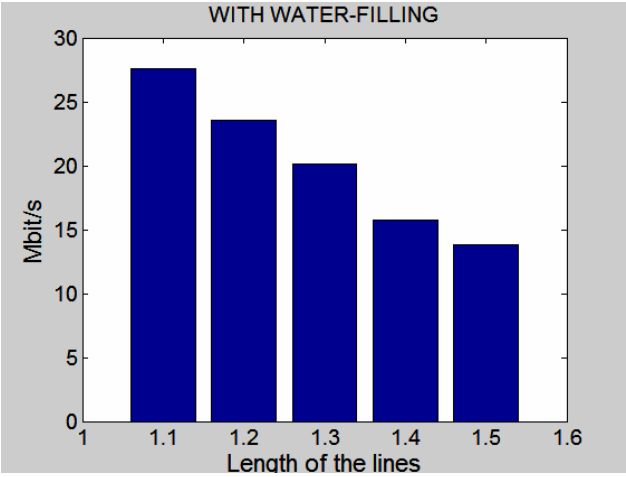


Figure 9.1: *Results that maximize sum of bitrates for five pairs with Water-filling.*

And the numeric results are:

R1_waterfill =13.8518 Mbps **ytot1_final =20.9788 mW**
R2_waterfill =15.7794 Mbps **ytot2_final =13.2497 mW**
R3_waterfill =20.1523 Mbps **ytot3_final =20.9716 mW**
R4_waterfill =23.5463 Mbps **ytot4_final =20.9775 mW**
R5_waterfill =27.6000 Mbps **ytot5_final =20.9764 mW**

Rsum_waterfill =102.1890 Mbps

On the other hand, the graphic without Water-filling is:

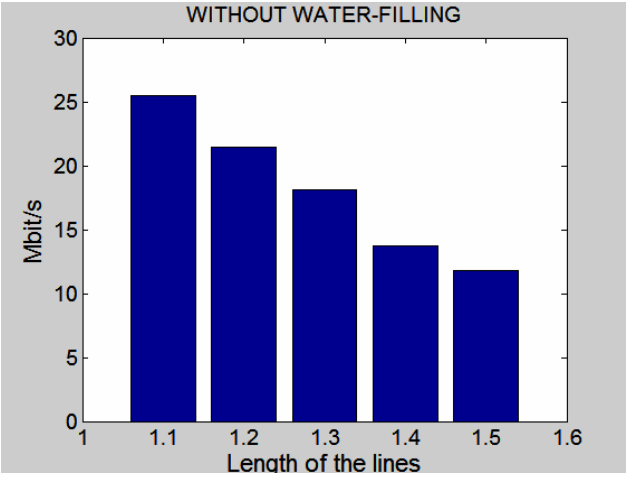


Figure 9.2: *Results that maximize sum of bitrates for five pairs without Water-filling.*

And the numeric results are:

R1 = 11.8174 Mbps
R2 = 13.7248 Mbps
R3 = 18.0727Mbps
R4 = 21.4676 Mbps
R5 = 25.5184 Mbps

Rsum = 91.8383 Mbps

As we can see in the last results, the difference between using and not using Water-filling is an increment around 2Mbps when we use Water-filling. This is the same conclusion that we have obtained in all the previous comparisons.

The highest sum that we can obtain with Water-filling is 102.1890 Mbps and without Water-filling is 91.8383 Mbps. Sometimes, when the line is longer the capacity in this line that maximizes the sum of all the bitrates, is less. For this reason the last graphics have a staircase shape, due to the fact that the lengths of the lines are ordered from shortest to longest. Nevertheless, other times we do not have this shape, because the capacity also depends on the crosstalk. Furthermore, we cannot obtain conclusions of the energy consumed for each pair because it not only depends on the length but also on crosstalk (in our case FEXT) in each line.

On the other hand, we can analyse and compare the last results with the results that we obtained in point 7.2 without the criterion to maximize sum of bitrates. The bitrates are different but mainly in the short pairs. Thus, for example, the bitrate for pair 5 with the criterion to maximize the sum was 27.6 Mbps and without the criterion was 19.6176 Mbps. In the long pairs the results are more similar. Thus, for example the bitrate for pair 1 with the criterion to maximize the sum was 13.8518 Mbps and without the criterion was 11.4066 Mbps. Finally we can say that with this criterion we obtain coherent results, and that it would be a good method to apply in practice.

9.2.2 Results for 10 pairs

In this case, we will consider ten lines that will be defined in the program *defxDSL.m* as 1.5 Km, 1.4 Km, 1.3 Km, 1.2 Km, 1.1 Km, 1.0 Km, and 0.9 Km, 0.8 Km, 0.7 Km and 0.6 Km. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT. The FEXT power sum matrix is the same that we used to obtain the realistic results. Here, if we use the random variable we have the same problem that we had with five pairs. For this reason we have created the program ***comparative_maxsum.m*** (see file "Max_Sum_10_pairs" in the CD) that generates simultaneously the results with and without Water-filling. The other variables are the same ones as always and we can see them in the program *defxDSL.m* in the CD. All the other programs are the same that we used when we generated the realistic results for ten pairs.

Thus, the graphic with Water-filling using the random variable and the realistic FEXT power matrix is:

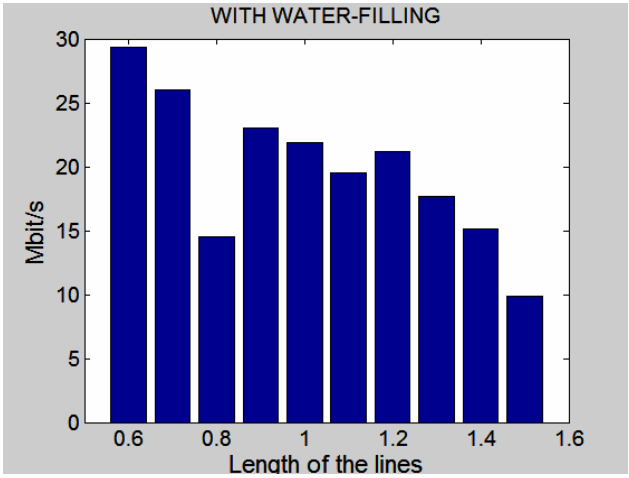


Figure 9.3: *Results that maximize sum of bitrates for ten pairs with Water-filling.*

And the numeric results are:

R1_waterfill = 9.8670 Mbps
R2_waterfill = 15.1498 Mbps
R3_waterfill = 17.6769 Mbps
R4_waterfill = 21.2046 Mbps
R5_waterfill = 19.5313 Mbps
R6_waterfill = 21.8514 Mbps
R7_waterfill = 23.0374 Mbps
R8_waterfill = 14.5331 Mbps
R9_waterfill = 26.0389 Mbps
R10_waterfill = 29.3638 Mbps

ytot1_final = 20.6928 mW
ytot2_final = 20.9789 mW
ytot3_final = 20.9974 mW
ytot4_final = 20.9761 mW
ytot5_final = 8.2853 mW
ytot6_final = 5.2887 mW
ytot7_final = 2.1093 mW
ytot8_final = 0.2122 mW
ytot9_final = 0.5321 mW
ytot10_final = 0.3359 mW

Rsum_waterfill = 198.2543 Mbps

On the other hand, the graphic without Water-filling is:

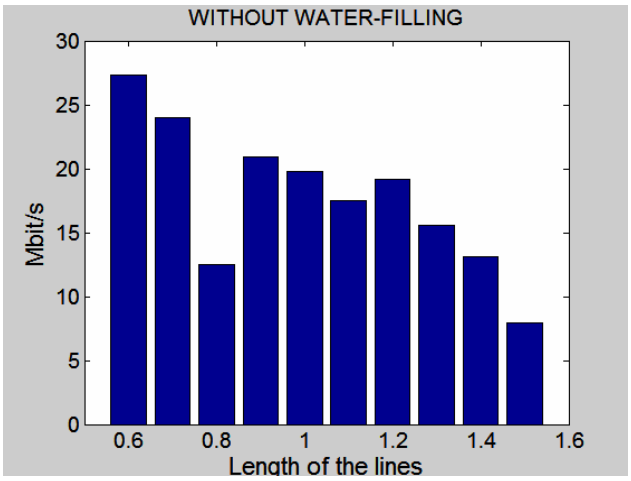


Figure 9.4: *Results that maximize sum of bitrates for ten pairs without Water-filling.*

And the numeric results are:

R1 = 7.9630 Mbps
R2 = 13.0978 Mbps
R3 = 15.5982 Mbps
R4 = 19.1294 Mbps
R5 = 17.4884 Mbps
R6 = 19.7719 Mbps
R7 = 20.9549 Mbps
R8 = 12.5345 Mbps
R9 = 23.9501 Mbps
R10 = 27.2959 Mbps

Rsum = 177.7842 Mbps

We know that the results considering ten pairs are more realistic than with five pairs because the cables are organized in binder groups of 10 to 50 pairs. Nevertheless the conclusions that we obtain with the last results are the same that we obtained with five pairs, that is, an increase of around 2 Mbps in the capacity when we use Water-filling. Moreover, this time we can see how the graphics do not have a staircase shape. Finally, the maximum sum of rates that we have achieved is 198.2543 Mbps when we use Water-filling and 177.7842 Mbps without Water-filling. If we run the program another time we will probably obtain other results because of the use of the random variable.

On the other hand, if we compare the last results with the results that we obtained in point 8.2 without the criterion to maximize sum of bitrates, we can say the same that we mentioned for five pairs. Thus, the bitrates are different but mainly in the short pairs. For example, the bitrate for pair 10 with the criterion to maximize the sum was 29.3638 Mbps and without the criterion was 20.1609 Mbps. Nevertheless in the long pairs, like the first one, the bitrate with the criterion to maximize the sum was 9.8670 Mbps and without the criterion was 11.7429 Mbps, which is more similar.

On balance, and as we said before, this criterion would be good in practice because the results never show discrimination, that is, we never obtain results that increase a lot the bitrates for a group of cables and decrease a lot the bitrates for the others. The results always maintain a balance.

10 MAXIMIZATION OF BITRATE IN THE PAIR HAVING THE LOWEST BITRATE

10.1 Description of the program

In this last point of our DSM study we will study the process in order to maximize the bitrate in the pair having the lowest bitrate. First, we will describe the program that we have done to maximize the bitrate in the pair having the lowest bitrate, that is, in the longest pair. The program is called ***iterwaterfill_maxlow.m*** (see Appendix and file "Max_Low_5_pairs" or file "Max_Low_10_pairs" in the CD). As we can see in the following lines, the program is similar to the program used in the last point to maximize the sum of bitrates.

The input of the program is the maximum power in *mW* that we have in each cable, and, the outputs are the capacity and the transmitted energy that we have in each pair at the end. Then, at the beginning of the program we define the maximum power for each cable and we run the Water-filling. Thus, we will obtain the bitrates for each pair, and with the Matlab function "[y,ind] = min()" we will be able to know what is the pair with the lowest bitrate. This pair will be the longest pair because it has more attenuation.

The next step is to repeat the same that we have done with maximum power, but now we are reducing the power in each pair every time, and we obtain and store the results with Water-filling. At the end, considering five pairs, we will have five different values of bitrates for the pair having the lowest bitrate (the bitrate when we reduce the power in system one, the bitrate when we reduce the power in system two, etc…). Then, we compare the five values of bitrates and we take the results that maximize the bitrate in the pair that had the lowest one. Finally, we update the power in the pair that generated the results that we took when we reduced its power.

The last process is done in the curl "*for,* but this curl is inside the curl *"while"* because we should repeat the process while we find values of the bitrate higher than in the last iteration. When we finish all the process we will have found the combination of

power for each pair that generates the maximum bitrate in the pair having the lowest bitrate.

On the other hand, we have done another program called ***iter_maxlow.m*** (see file "Max_Low_5_pairs" or file "Max_Low_10_pairs" in the CD). It does the same as the previous one but without Water-filling.

Next, we will describe the system that we have used to obtain the results, and we will generate the results for five and ten pairs.

10.2 Results

10.2.1 Results for 5 pairs

To obtain the results we will use the same system that we used in the last point to maximize the sum of bitrates. Thus, we have defined five lines (1.5 Km, 1.4Km, 1.3Km, 1.2Km and 1.1Km respectively) in the program *defxDSL.m*. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT. The FEXT power sum matrix is the same that we used to obtain the realistic results. In this case, to solve the problem of the use of the gamma distributed random variable, we have created the program called ***comparative_maxsum.m*** (see file "Max_Low_5_pairs" in the CD) that generates simultaneously the results with and without Water-filling. Nevertheless, we have realised that with the last program, each method (with and without Water-filling) achieves the maximization of the pair through different ways, that is, not always both methods achieve the maximization of the pair when we decrease the power in certain pair. For example, with Water-filling it is possible that we achieve the maximization of the pair when we reduce the power in pair two, and without Water-filling it is possible that we achieve the maximization of the pair when we reduce the power in pair three. For this reason we cannot use the program *comparative_maxsum.m.*

Then, we have decided not to use the random variable and this way we will be able to compare the results with and without Water-filling.

The other variables, like the spectral density or the ratios between disturber and disturbing pairs have the same values than in the last point and are defined in the program *defxDSL.m* in the CD. All the other programs are the same that we used when we generated the realistic results for five pairs.

Then, the graphic with Water-filling and the realistic FEXT power matrix without the use of the random variable is:

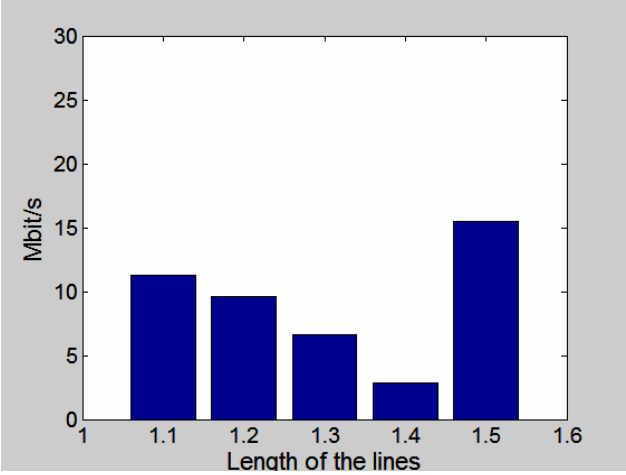


Figure 10.1: *Results that maximize the bitrate in the pair having the lowest bitrate when we consider five pairs with Water-filling.*

And the numeric results are:

R1_waterfill = 15.4776 Mbps
R2_waterfill = 2.8635 Mbps
R3_waterfill = 6.6542 Mbps
R4_waterfill = 9.6255 Mbps
R5_waterfill = 11.3031 Mbps

ytot1_final = 20.9999 mW
ytot2_final = 0.0588 mW
ytot3_final = 0.1270 mW
ytot4_final = 0.1341 mW
ytot5_final = 0.0846 mW

We can see in the last graph how the longest pair (the pair with 1.5 Km) has the highest capacity. If we do not use the last program with the criterion to maximize the bitrate in the pair having the lowest bitrate, this pair would have had the lowest bitrate because of its greater attenuation. As we can see in the final power of each pair, we have achieved our target reducing the power in the other pairs until we have got the

highest capacity in pair one. For this reason pair one uses all the power (21 mW approximately) while the other powers are around [0.05-0.14] mW.

On the other hand, the graphic without Water-filling is:

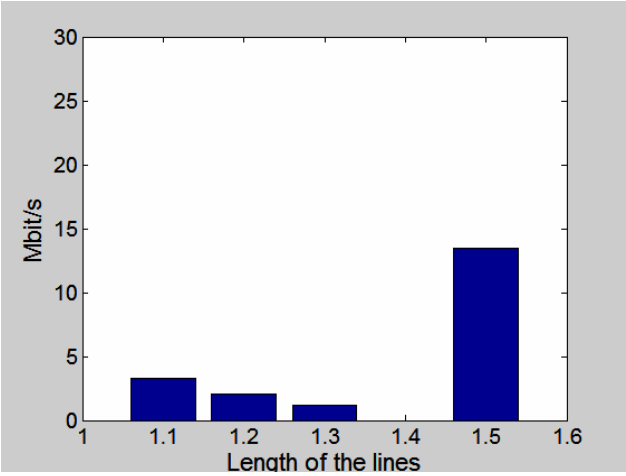


Figure 10.2: *Results that maximize the bitrate in the pair having the lowest bitrate when we consider five pairs without Water-filling.*

And the numeric results are:

R1 = 13.4534 Mbps
R2 = 0 Mbps
R3 = 1.2115 Mbps
R4 = 2.0778 Mbps
R5 = 3.2686 Mbps

In this case we can state the same conclusions that we obtained with Water-filling, but of course, the results show us lower capacity in the lines because we do not use Water-filling. The difference in the capacity of the longest pair is around 2 Mbps (the same difference than we usually obtain when we compare the results with and without Water-filling).

Finally, if we compare the last results with the results that we obtained in point 7.2 without the criterion to maximize the bitrate in the pair having the lowest bitrate, we can say that now the bitrate for pair one (the pair with the lowest bitrate at the beginning) is greater, as we hoped. Nevertheless, we have had to reduce the power in the other lines a great deal in order to achieve it. For this reason the bitrates in the other lines are too small, and then, this criterion is not good if we want to reach equilibrium between the

bitrates of the lines. For example in the last results without Water-filling, the bitrate in pair two is 0 Mbps, that is, we cannot send information in this line. Thus, this criterion will be valid only if we are looking for the maximization of the bitrate in the pair having the lowest bitrate, and the bitrates of the other lines do not matter.

10.2.2 Results for 10 pairs

Now, we will consider ten lines in order to obtain more realistic results. The lengths of the lines will be defined in the program *defxDSL.m* as 1.5 Km, 1.4 Km, 1.3 Km, 1.2 Km, 1.1 Km, 1.0 Km, and 0.9 Km, 0.8 Km, 0.7 Km and 0.6 Km. We consider White Noise (with -140dBm/Hz of spectral density) and FEXT using the FEXT power sum matrix that we used to obtain the realistic results. Here, if we use the random variable we have the same problem that we had with five pairs and, as already known, we cannot solve it with the program ***comparative_maxlow.m.*** Nevertheless we can find this program in the file "Max_Low_10_pairs" in the CD. The other variables and programs are the same that we used when we generated the realistic results for ten pairs.

The graphic with Water-filling and the realistic FEXT power matrix without the use of the random variable is:

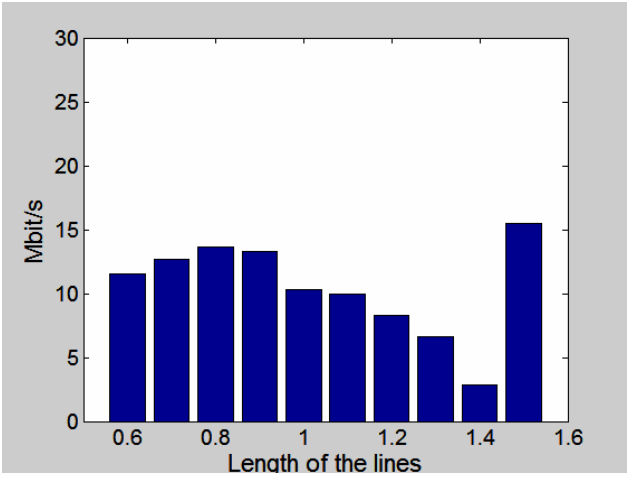


Figure 10.3: *Results that maximize the bitrate in the pair having the lowest bitrate when we consider ten pairs with Water-filling.*

And the numeric results are:

R1_waterfill = 15.4431 Mbps
R2_waterfill = 2.8376 Mbps
R3_waterfill = 6.5895 Mbps
R4_waterfill = 8.2584 Mbps
R5_waterfill = 9.9489 Mbps
R6_waterfill = 10.2681 Mbps
R7_waterfill = 13.3041 Mbps
R8_waterfill = 13.6448 Mbps
R9_waterfill = 12.7133 Mbps
R10_waterfill = 11.5704 Mbps

ytot1_final = 21.0000 mW
ytot2_final = 0.0586 mW
ytot3_final = 0.1264 mW
ytot4_final = 0.0830 mW
ytot5_final = 0.0534 mW
ytot6_final = 0.0213 mW
ytot7_final = 0.0213 mW
ytot8_final = 0.0085 mW
ytot9_final = 0.0021 mW
ytot10_final = 5.4089e-4 mW

The conclusions using ten pairs are the same that we said for five pairs, that is, the longest pair (the pair with 1.5 Km) has the highest capacity. For this reason pair one uses all the power (21 mW) and the other cables have reduced their powers in order to affect as little as possible the first one with their crosstalks. Thus, pair one will get the highest capacity.

Finally, the graphic with Water-filling and the realistic FEXT power matrix without the use of the random variable is:

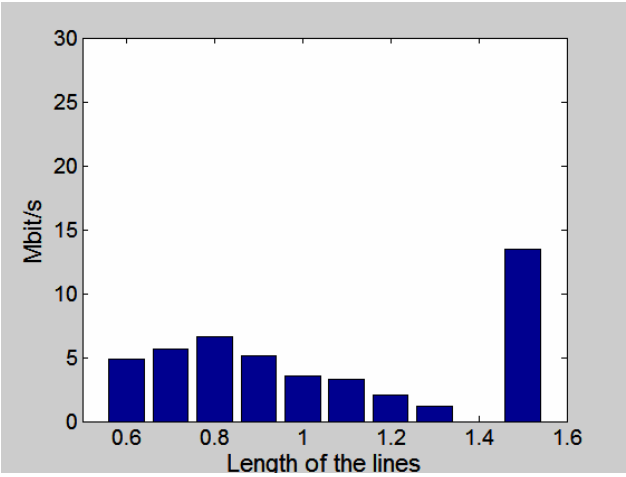


Figure 10.4: *Results that maximize the bitrate in the pair having the lowest bitrate when we consider ten pairs without Water-filling.*

And the numeric results are:

R1 = 13.4456 Mbps
R2 = 0 Mbps
R3 = 1.2033 Mbps
R4 = 2.0672 Mbps
R5 = 3.2552 Mbps
R6 = 3.5769 Mbps
R7 = 5.1396 Mbps
R8 = 6.5934 Mbps
R9 = 5.6993 Mbps
R10 = 4.8399 Mbps

As we hoped, the conclusions of the results without Water-filling are the same that we have already mentioned. Thus, if we compare these results with the ones obtained with the Water-filling, the difference in the capacity of the longest pair is around 2 Mbps, and the shapes of both pictures are similar.

On balance, and as we said for five pairs, this method will be valid only if we are looking for the maximization of the bitrates in the pair having the lowest bitrate, without caring the bitrates that we achieve in the other lines.

11 CONCLUSIONS

As we have seen, the main performance limitation of modern DSL networks, like VDSL, is the crosstalk among different pair of lines (users). In all the comparisons of results, deploying DSM techniques like Water-filling, a network operator can do an adaptive assignment of the spectrum in a multiuser environment, and then, can increase the data rates and the quality of the services in a broadband access.

The Water-filling method that we have used has achieved an increment of around 2 Mbps in the bitrates of the lines. Nevertheless we have realized that in the program Water-filling we achieve better results if we introduce a PSD constraint because, as we saw in point 5, the power will be distributed more uniformly. Also we can see this improvement if we compare the Rate Region with and without PSD constraint.

On the other hand, after considering two, three, five and ten pairs of cables we can conclude that the differences between the results with and without Water-filling are always the same. Nevertheless, we know that a pair cable may contain up to 2000 different pairs, which are organised in binder groups from 10 to up to 50 pairs, and that, is more realistic if we consider ten pairs instead of two, three or five. Moreover, we know that the final results depend greatly on the crosstalk (in our case FEXT), and for this reason we have used, in the last points, a FEXT power sum matrix with real values and the use of the gamma distributed random variable. The results that we have obtained are a little higher than the non realistic results but the reason is only because we use different values of FEXT power sum.

Finally, as we concluded in point 9 and 10, we can apply other criterions in our DSM method. Thus, in point 9, if we apply to criterion to maximize the sum of bitrates we will obtain good results that we could apply in practice if we want maximize the sum of bitrates and achieve results without discrimination between the lines. Nevertheless, if what we want to do is to maximize the bitrate in the pair having the lowest bitrate without caring about the bitrates that we achieve in the other lines, then, we can use the criterion seen in point 10. This criterion, as we saw, maximizes the bitrate in the pair having the lowest bitrate and discriminates the other lines. For this

reason, maybe some pairs do not have the capacity that their users need. Due to this, if we do not want discrimination this criterion will not be a good idea.

In future works, like in the continuation of this Master Thesis, new methods of DSM and the comparisons of the results between all methods could be studied. Recently two optimal but centralized DSM algorithms are proposed, the Optimal Spectrum Balancing (OSB) algorithm and the Iterative Spectrum Balancing (ISB) algorithm. The OSB algorithm addresses the spectrum management problem through the maximization of a weighted rate-sum across all users, which explicitly takes into account the damage done to the other lines when optimizing each line's spectra. Nevertheless, this method should be studied only for five lines, because it has an exponential complexity in the number of users, making it intractable for DSL network with more lines. As an improvement over the OSB algorithm in order to solve the last problem, ISB is proposed to implement the weighted-rate sum optimization in an iterative fashion over the users. This algorithm has quadratic complexity in the number of users, which makes the ISB viable for networks with more than five lines.

Moreover we could study the improvement when we use or not use the centralized algorithm. The last two methods are centralized and they require knowledge of the crosstalk channels between all lines. This information is difficult to obtain when we have more lines. On the other hand, there are semi-centralized algorithms like SCALE, which achieves good results with low complexity and does not require knowledge of the crosstalk channels. The problem is that this algorithm is not autonomous.

On balance, the bottlenecks of the DSM algorithms are the need for a low complexity to provide autonomous operation and nearoptimal rate region. Nevertheless, new methods called ASB algorithms can offer autonomous operation and nearoptimal rate region with similar complexity than Iterative Water-filling. The key concept of ASB is the "reference line", which allows each user to optimize it's transmit spectra to achieve its own target rate while minimizing the degradation caused to other users in the frequency-selective interference channel of DSL.

The study of the last algorithm and its results could be more interesting in the development of the generation of DSM methods and could be a topic for new studies in Master Thesis.

REFERENCES

[1] Electronic Communications Committee (ECC) Report 79, "High Capacity DSL Systems". March 2006.

[2] J. Cioffi, "Dynamic Spectrum Management," A detailed chapter on DSM by Prof. Cioffi. Chapter 11.

[3] "Dynamic Spectrum Management Project", January 2007 (url: http://www.stanford.edu/group/cioffi/dsm/).

[4] K. B. Song, S. T. Chung, G. Ginis and J. M. Cioffi, " Dynamic Spectrum Management for Next-Generation DSL Systems", IEEE Communications Magazine, October 2002.

[5] "Installation Effects Upon Alien Crosstalk and Equal Level Far End Crosstalk", January 2007 (url: http://www.belden.com/pdfs/Techpprs/ieacectp.htm).

[6] "VDSL Standards Report" January 2007 (url: http://www.dslprime.com/News_Articles/a/VDSL/vdsl.html).

[7] Nils Holte, "Broadband communication in existing copper cables by means of xdsl systems - a tutorial".

[8] Achankeng Leke and John M. Cioffi, "A Maximum Rate Loading Algorithm for Discrete Multitone Modulation Systems".

[9] John A. C., "Multicarrier Modulation for Data Transmission: An Idea Whose Time Has Come", May 1990.

[10] Wei Yu, George Ginis, and John M. Cioffi, Fellow, IEEE, "Distributed Multiuser Power Control for Digital Subscriber Lines" (2002).

[11] Wei Yu, Wonjong Rhee, Stephen Boyd, and John M. Cioffi, "Iterative Water-filling for Gaussian Vector Multiple Access Channels", (November 6, 2001).

[12] Nils Holte, "Simulation of Crosstalk in Twisted Pair Cables".

[13] R. Cendrillon, Wei Yu*, Member, IEEE*, M. Moonen*, Senior Member, IEEE*, J. Verlinden, and T. Bostoen, "Optimal Multiuser Spectrum Balancing for Digital Subscriber Lines" Vol. 54, No. 5, May 2006.

[14] Driton Statovci, Tomas Nordström, and Rickard Nilsson, "The Normalized-Rate Iterative Algorithm: A Practical Dynamic Spectrum Management Method for DSL", pp 1-16, 2005.

[15] Raphael Cendrillon, Jianwei Huang, Mung Chiang, and Marc Moonen,

"Autonomous Spectrum Balancing for Digital Subscriber Lines", September 1,2006.

APPENDIX - Matlab code of the developed programs

- Program **defxDSL.m**

```
%  Default input data for xDSL calculations
%
%
%
L1=1                %  Cable length of system 1 in km (initial value)
L2=0.7              %  Cable length of system 2 in km (initial value)
FL=3                %  lower frequency in MHz
FH=5.1              %  upper frequency in MHz
NF=487              %  number of frequencies. Usually the subchannel
                    %  spacing is 4.3125 KHz. Then NF=(FH-FL)/4.3125
MARG=3              %  margin in dB relative to Shannon (3dB for TCM +
                       impl.marg.)
BWEM=14.5           %  Maximum bandwidth efficiency in bit/s/Hz
BWEmin=1            %  Minimum bandwidth efficiency in bit/s/Hz
BWEstep=0.05        %  Step size in bandwidth efficiency in bit/s/Hz
Reff=1              %  Rate reduction due to overhead(CP, RS-code,…)
%
%  Signal
%
Sipsd1=-50          %  Transmitter output spectral density in dBm/Hz
Sipsd2=-50          %  Disturber output spectral density in dBm/Hz
AttMod=1
               % 1:  attenuation proportional to sqrt(f)
               % 2:  attenuation given as a table for .4 mm cable
               % 3:  attenuation given as a table for .6 mm cable
alfa1dB=22.5      %  Attenuation in dB at 1 MHz  (AttMod=1)
Falfa=[.1 .2 .35 .7 1 1.4 2  2.8 4]
               %  Frequency vector in MHz for attenuation (AttMod=2)
alfaM=zeros(1,length(Falfa));
alfaM(2,:)=[9.11 11.15 13.65 18.74 22.41 26.52 31.69 37.5 44.82]
              %  Attenuation vector for .4 mm cable in dB/km(AttMod=2)
alfaM(3,:)=[5.10 6.78 8.93 12.67 15.10 17.87 21.35 25.27 30.20]
           %  Attenuation vector for .6 mm cable in dB/km (AttMod=3)
beta1=10*pi    %  Phase constant in rad/km at 1 MHz
%
%  White noise
%
AWGNon=1    %  0: No white noise
            %  1: Additive white noise with specified spectral density
WNpsd=-140     %  White noise spectral density in dBm/Hz
%
%  self FEXT
%
FEon=2         %  0:  self FEXT off
               %  1:  50 pair empirical model for self FEXT
               %  2:  specified FEXT power sum
LFEdistRat1=70     %  Ratio between length of disturbed pair and
                      disturbing pairs in % in system 1
LFEdistRat2=142.85 %  Ratio between length of disturbed pair and
                      disturbing pairs in % in system 2
                  %  Always use LFEdistRat=100 for downstream
NPAF=1            %  Number of disturbing pairs for self FEXT (FEon=1)
FEXT1ps=50.5      %  FEXT power sum in dB at 1 MHz and 1 km for FEon=2
```

```
%  Calculate derived parameters
%
initial
```

- Program **initial.m**

```
%  Initialise data derived from input data for xDSL calculations
%
%  amplitude margin
%
AMARG=10^(-MARG*.1);
%
%  frequency vectors
%
dF=(FH-FL)/NF;
F=FL+(1:NF)*dF-dF/2;
FNR=(1:NF);
%
%  attenuation constant
%
if AttMod==1
   %
   %  Attenuation model 1,  sqrt(f) attenuation
   %
   alfa1N=alfa1dB*log(10)/20;
   alfa=alfa1N*sqrt(F);
else
   %
   %  Attenuation model 2 and 3, attenuation specified in a table
   %
   %  conversion to Neper
   %
   alfaNV=alfaM(AttMod,:)*log(10)/20;
   %
   %  Spline interpolation of attenuation
   %
   alfa=spline(Falfa,alfaNV,F)
end
%
%  phase constant
%
beta=beta1*F+alfa;
```

- Program **singcap.m**

```
%  Calculate total transmission capacity for onedirectional
   transmission
%  in two pairs
%  Signal power spectrum
%
sig
%
Npsd1=zeros(1,NF);
Npsd2=zeros(1,NF);
```

```
%
%  white noise
%
if AWGNon >= 1
   whnoi
   Npsd1=Npsd1+WN*ones(1,NF);
   Npsd2=Npsd2+WN*ones(1,NF);

end
%
%  Self FEXT
%
if FEon >= 1
   fext
   Npsd1=Npsd1+FE1;
   Npsd2=Npsd2+FE2;

end
%
%  Shannon bound with margin
%

y1=log2(1+AMARG*(S1)./Npsd1);
y2=log2(1+AMARG*(S2)./Npsd2);


%
%  Round down to nearest step (adaptive modulation)
%
y1=floor(y1/BWEstep)*BWEstep;
y2=floor(y2/BWEstep)*BWEstep;

%
%  Maximum bandwidth efficiency limitation
%
y1(y1>BWEM)=BWEM;
y2(y2>BWEM)=BWEM;

%
%  Minimum bandwidth efficiency limitation
%
y1(y1<BWEmin-1.e-6)=0;
y2(y2<BWEmin-1.e-6)=0;

cap1=sum(y1)*dF*Reff;
cap2=sum(y2)*dF*Reff;
```

- Program **plotcap.m**

```
%  Total transmissin capacity
%
%
R1=0;
R2=0;
   singcap
    R1=cap1
    R2=cap2
```

- Program **fext.m**

```
%  FEXT model
%
%  Lengths of disturbing pair, total length: LFES, overlap: LFE
%
LFES1=L1*LFEdistRat1/100;
LFES2=L2*LFEdistRat2/100;

LFE1=LFES1;
LFE2=LFES2;

if LFE1 > L1
   LFE1=L1;
end

if LFE2 > L2
   LFE2=L2;
end

%
%  Output signal + cable
%
FE1=10^(Sipsd2/10)*exp(-2*LFES1*alfa);
FE2=10^(Sipsd1/10)*exp(-2*LFES2*alfa);

%
%  FEXT model
%
if FEon==1
   %
   %  Empirical FEXT model, 99% power sum, 50 pair binder group
   %
   FE1=FE1.*(NPAF/49)^0.6*3.e-4.*F.^2*LFE1;
   FE2=FE2.*(NPAF/49)^0.6*3.e-4.*F.^2*LFE2;

else
   %
   %  High frequency FEXT model, specified FEXT power sum at 1 MHz
   %
   FE1=FE1.*10^(-FEXT1ps/10).*F.^2*LFE1;
   FE2=FE2.*10^(-FEXT1ps/10).*F.^2*LFE2;
end
```

- Program **waterfill.m**

```
function [y,ytot,b,btotal,R] = waterfill(S,Npsd,L,totalpow)


%   waterfill : Water Filling Algorithm.

% PARAMETERS:

%   y : Energies per channel Vector.
%   ytot : Total transmitted energy.
%   b : Bits per channel.
%   btot : Total transmitted bits.
%   R : Total capacity in Mbits/sec.
%   S : Signal vector in lineal.
%   Npsd : Noise per channel vector in lineal.
%   L : Length of the line.
%   totalpow : Total available power in lineal in (mW).

alfa1dB=22.5      %  Attenuation in dB at 1 MHz  (AttMod=1 in the
                     program defxDSL)

FL=3              %  lower frequency in MHz
FH=5.1            %  upper frequency in MHz
NF=487            %  number of frequencies. Usually the subchannel
                     spacing is 4.3125 KHz. Then NF=(FH-FL)/4.3125
dF=(FH-FL)/NF;
F=FL+(1:NF)*dF-dF/2;

BWEM=15        %  Maximum bandwidth efficiency in bit/s/Hz
BWEmin=1       %  Minimum bandwidth efficiency in bit/s/Hz
BWEstep=0.05   %  Step size in bandwidth efficiency in bit/s/Hz



noise_sig=Npsd./S;
level=Npsd./S;     %initial level of Water-filling

g=length(S);

% Energy in (mW) to increment one bit
enerbit=(Npsd./(abs(10.^(-alfa1dB.*L.*sqrt(F)./20))).^2).*4.3125e3;

enertot=0;    %total transmitted energy
b=zeros(1,g); %total transmitted bits

while (enertot <totalpow)
[value1,index]=min(level);
level(index)=level(index)+enerbit(index);
enertot=enertot+enerbit(index);
b(index)=b(index)+1;
enerbit(index)=2.*enerbit(index);
end

level(index)=level(index)-(enerbit(index)./2);
ytot=enertot-(enerbit(index)./2);
b(index)=b(index)-1;
btotal=sum(b);

energy=zeros(1,g);
```

```
for m=1:g
    energy(m)=level(m)-noise_sig(m);
    m=m+1;
end

y=energy;
ytot=sum(y);


%Capacity in Mbits/sec :

b=floor(b/BWEstep)*BWEstep;

%
%  Maximum bandwidth efficiency limitation
%
b(b>BWEM)=BWEM;

%
%  Minimum bandwidth efficiency limitation
%
b(b<BWEmin-1.e-6)=0;


%Capacity in Mbits/sec :

R=sum(b)*4.3125e-3;


% Drawing the figures...

bar([1:g],10*log10(y))
 xlabel('Subchannels','FontSize',16)
 ylabel('Energy per subchannel in dBm','FontSize',16)
 grid

 figure

bar([1:g],10*log10(y./4.3125e3))
 xlabel('Subchannels','FontSize',16)
 ylabel('PSD per subchannel in dBm/Hz','FontSize',16)
 grid

figure

bar([1:g],b)
 xlabel('Subchannels','FontSize',16)
 ylabel('Bits per subchannel','FontSize',16)
 grid

figure

bar([1:g],noise_sig)
 xlabel('Subchannels','FontSize',16)
 ylabel('Noise/Signal','FontSize',16)
 grid
```

- Program **Rate_region.m**

```
function rate_region(totalpower1,totalpower2)

% rate_region: Program that obtain the Rate Region.

% In this program we have considered that System 1 is longer than
  System 2.
% Then, System 2 needs less power than System 1. We are USING WATER-
  FILLING allways.


% PARAMETERS:
%   ytot1_final : Total transmitted energy in System 2.
%   ytot2_final : Total transmitted energy in System 1.
%   R1_final : Target Rate in Mbits/sec for System 1.
%   R2_final : Target Rate in Mbits/sec for System 2.
%   totalpower1 : Total available power for system 1.
%   totalpower2 : Total available power for system 2.


defxDSL;


 Sipsd1=10.*log10(totalpower1/( 4.3125e3.*NF)); % Final value
 Sipsd2=10.*log10(totalpower2/( 4.3125e3.*NF)); % Final value


singcap    % With this call we obtain S1,S2,Npsd1,Npsd2
totalpow1=((10^(Sipsd1/10)).*4.3125e3).*NF;
totalpow2dB=10.*log10(((10^(Sipsd2/10)).*4.3125e3).*NF); %Update the
                                                           total
                                                           available
                                                           power.
totalpow2=10.^(totalpow2dB./10); %Update the total available power.

%Call waterfilling for System 1
    [y1,ytot1,b1,btotal1,R1] = waterfill(S1,Npsd1,L1,totalpow1);
%Call waterfilling for System 2
    [y2,ytot2,b2,btotal2,R2] = waterfill(S2,Npsd2,L2,totalpow2);

i=1;
R11(i)= R1
R22(i)= R2


while R2>0


    totalpow2dB=totalpow2dB-2; % Update the total available power.
    totalpow2=10.^(totalpow2dB./10); % Update the total available
                                       power.

    Sipsd2=10.*log10(totalpow2/( 4.3125e3.*NF)); % Final value


    Singcap

 %Call waterfilling for System 1
 [y1,ytot1,b1,btotal1,R1] = waterfill(S1,Npsd1,L1,totalpow1);
```

```
 %Call waterfilling for System 2
[y2,ytot2,b2,btotal2,R2] = waterfill(S2,Npsd2,L2,totalpow2);


     i=i+1

    R11(i)= R1
    R22(i)= R2


end

%Drawing...
figure

plot(R22,R11)
 title('Rate Region','FontSize',16)
 xlabel('User 2 (Mbps)','FontSize',16)
 ylabel('User 1 (Mbps)','FontSize',16)
```

- Program **Iterwaterfill.m**

```
function [R1_final,R2_final,ytot1_final,ytot2_final] =
iterwaterfill(R1_Target,R2_Target)


% iterwaterfill : Iterative Water Filling Algorithm in order to obtain
%                 the target rates for two systems.

% In this program we have considered that System 1 is longer than
  System 2.
% Then, System 2 needs less power than System 1. We are using water-
  filling allways.


% PARAMETERS:
%   ytot1_final : Total transmitted energy in System 1.
%   ytot2_final : Total transmitted energy in System 2.
%   R1_final : Total capacity in Mbits/sec for System 1.
%   R2_final : Total capacity in Mbits/sec for System 2.
%   R1_Target : Target Rate for system 1.
%   R2_Target : Target Rate for system 2.



defxDSL;

     totalpower1=((10^(Sipsd1/10)).*4.3125e3).*NF;
     totalpower2=((10^(Sipsd2/10)).*4.3125e3).*NF;


singcap    % With this call we obtain S1,S2,Npsd1,Npsd2

ini_totalpower=[totalpower1,totalpower2];
totalpower=[totalpower1,totalpower2];
R_Target=[R1_Target,R2_Target];
```

```
%Call waterfilling for System 1
[y1,ytot1,b1,btotal1,R1] = waterfill(S1,Npsd1,L1,totalpower(1));
%Call waterfilling for System 2
[y2,ytot2,b2,btotal2,R2] = waterfill(S2,Npsd2,L2,totalpower(2));

 R=[R1,R2];
 K=1;   %Number of iterations

 C(1,K)=R(1);
 C(2,K)=R(2);

increasedB=3;
increase=10^(increasedB/10);


while (R_Target(1)>R(1) | R_Target(2)>R(2))

 for i=1:2    %number of Systems

if R(i)>(R_Target(i)+(R_Target(i))/100)
    totalpower(i)=totalpower(i)-increase;
end

if R(i)<R_Target(i)
     totalpower(i)=totalpower(i)+increase;
 end

  if totalpower(i)>ini_totalpower(i)
      totalpower(i)=ini_totalpower(i);
  end

end
  increasedB=increasedB/2;
  increase=10^(increasedB/10);

  K=K+1; %Number of iterations

  if K>30
      'Sorry,it is impossible to reach these Target Rates. Try with
others'
      return
  end

  Sipsd1=10.*log10((totalpower(1)/NF)/4.3125e3);
  Sipsd2=10.*log10((totalpower(2)/NF)/4.3125e3);

 singcap  % With this call we update S1,S2,Npsd1,Npsd2

%Call waterfilling for System 1
 [y1,ytot1,b1,btotal1,R1] = waterfill(S1,Npsd1,L1,totalpower(1));
%Call waterfilling for System 2
 [y2,ytot2,b2,btotal2,R2] = waterfill(S2,Npsd2,L2,totalpower(2));
 R=[R1,R2]
 C(1,K)=R(1);
 C(2,K)=R(2);


end

R1_final=R(1);
R2_final=R(2);
```

```
ytot1_final=totalpower(1);
ytot2_final=totalpower(2);

%Drawing...

figure
set(gcf,'DefaultLineLineWidth',1)   %  Default line size
set(gcf,'DefaultAxesFontSize',14)   %  Default axis font size
set(0,'DefaultAxesLineStyleOrder','-|--')
plot(1:K,C,'k')
xlabel('Iterations','FontSize',16)
ylabel('Mbit/s','FontSize',16)
legend('System 1','System 2')
grid
```

- Program **Iterwaterfill.m**

```
function y=FEXT_rand
%
%  Generation of one sample function of random FEXT matrix
%
%Values of FEXT power sum in dB:

FEXTps=[0 45.8 50.4 52.7 53.7
        45.8 0 48.0 50.0 52.6
        50.4 48.0 0 47.6 51.3
        52.7 50.0 47.6 0 47.3
        53.7 52.6 51.3 47.3 0];

%  Convert to crosstalk power
%
PFEXTps=10.^(-FEXTps/10);
for II=1:5
   PFEXTps(II,II)=0;
end


[N,M]=size(PFEXTps);
%
%  Generate gamma distribution with 0.5 degrees of freedom
%  for all elements
%
X=randn(N,M);
Z=X.^2;
%
%  Multiply by average crosstalk power
%
y=Z.*PFEXTps;
%
%  Extract upper triangular part for crosstalk represented as a matrix
%
if N == M
   y=triu(y);
   %
   %  Generate symmetrical lower triangular part
   %
   y=y+y';
end
```

- Program **Iterwaterfill_maxsum.m**

```
function
[R1_final,R2_final,R3_final,R4_final,R5_final,Rsumtotal,ytot1_final,yt
ot2_final,ytot3_final,ytot4_final,ytot5_final] =
iterwaterfill_maxsum(Pmax)


% iterwaterfill_maxsum : Iterative Water Filling Algorithm in order to
% maximize the sum of bitrates for five systems.

% We are USING WATER-FILLING allways.

% INPUTS:
%  Pmax : Maximum power in mW that we have in each cable


% PARAMETERS:
%   ytot1_final : Total transmitted energy in System 1.
%   ytot2_final : Total transmitted energy in System 2.
%   ytot3_final : Total transmitted energy in System 3.
%   ytot4_final : Total transmitted energy in System 4.
%   ytot5_final : Total transmitted energy in System 5.


%   R1_final : Total capacity in Mbits/sec for System 1.
%   R2_final : Total capacity in Mbits/sec for System 2.
%   R3_final : Total capacity in Mbits/sec for System 3.
%   R4_final : Total capacity in Mbits/sec for System 4.
%   R5_final : Total capacity in Mbits/sec for System 5.


defxDSL;

     maxpow1=Pmax; %Update the total available power.
     maxpow2=Pmax; %Update the total available power.
     maxpow3=Pmax; %Update the total available power.
     maxpow4=Pmax; %Update the total available power.
     maxpow5=Pmax; %Update the total available power.

     maxpow1dB=10.*log10(maxpow1); %Update the total available power.
     maxpow2dB=10.*log10(maxpow2); %Update the total available power.
     maxpow3dB=10.*log10(maxpow3); %Update the total available power.
     maxpow4dB=10.*log10(maxpow4); %Update the total available power.
     maxpow5dB=10.*log10(maxpow5); %Update the total available power.



singcap    % With this call we obtain
           % S1,S2,S3,S4,S5,Npsd1,Npsd2,Npsd3,Npsd4,Npsd5

maxpowdB=[maxpow1dB,maxpow2dB,maxpow3dB,maxpow4dB,maxpow5dB];
maxpow=[maxpow1,maxpow2,maxpow3,maxpow4,maxpow5];

%Call waterfilling for System 1
 [y1,ytot1,b1,btotal1,R1] = waterfill(S1,Npsd1,L1,maxpow(1));
%Call waterfilling for System 2
 [y2,ytot2,b2,btotal2,R2] = waterfill(S2,Npsd2,L2,maxpow(2));
```

```
%Call waterfilling for System 3
 [y3,ytot3,b3,btotal3,R3] = waterfill(S3,Npsd3,L3,maxpow(3));
%Call waterfilling for System 4
 [y4,ytot4,b4,btotal4,R4] = waterfill(S4,Npsd4,L4,maxpow(4));
%Call waterfilling for System 5
 [y5,ytot5,b5,btotal5,R5] = waterfill(S5,Npsd5,L5,maxpow(5));

 R=[R1,R2,R3,R4,R5];   % Bitrate with Water-filling
 K=2;
 Rsum(K)=sum(R);       % Sum of bitrate with Water-filling
 R11(K)=R1;
 R22(K)=R2;
 R33(K)=R3;
 R44(K)=R4;
 R55(K)=R5;
 ytot11(K)=ytot1;
 ytot22(K)=ytot2;
 ytot33(K)=ytot3;
 ytot44(K)=ytot4;
 ytot55(K)=ytot5;

increasedB=3;
increase=10^(increasedB/10);
Rsum(1)=Rsum(2)-1; % to come in the curl

while Rsum(K)>Rsum(K-1)
for i=1:5    %number of Systems
    maxpowdB(i)=maxpowdB(i)-increase;   % Update the total available
                                        % power.
    maxpow(i)=10.^(maxpowdB(i)./10);    % Update the total available
                                        % power.
    Sipsd(i)=10.*log10((maxpow(i)/NF)/4.3125e3);
    Sipsd1=Sipsd(1);
    Sipsd2=Sipsd(2);
    Sipsd3=Sipsd(3);
    Sipsd4=Sipsd(4);
    Sipsd5=Sipsd(5);

    singcap  % With this call we update
             % S1,S2,S3,S4,S5,Npsd1,Npsd2,Npsd3,Npsd4,Npsd5

%Call waterfilling for System 1
[y1,ytot1new,b1,btotal1,R1new] = waterfill(S1,Npsd1,L1,maxpow(1));
%Call waterfilling for System 2
[y2,ytot2new,b2,btotal2,R2new] = waterfill(S2,Npsd2,L2,maxpow(2));
%Call waterfilling for System 3
[y3,ytot3new,b3,btotal3,R3new] = waterfill(S3,Npsd3,L3,maxpow(3));
%Call waterfilling for System 4
[y4,ytot4new,b4,btotal4,R4new] = waterfill(S4,Npsd4,L4,maxpow(4));
%Call waterfilling for System 5
[y5,ytot5new,b5,btotal5,R5new] = waterfill(S5,Npsd5,L5,maxpow(5));


 Rsumnew(i)=R1new+R2new+R3new+R4new+R5new;
    R11(i)=R1new;
    R22(i)=R2new;
    R33(i)=R3new;
    R44(i)=R4new;
    R55(i)=R5new;
    ytot11(i)=ytot1new;
    ytot22(i)=ytot2new;
```

```
    ytot33(i)=ytot3new;
    ytot44(i)=ytot4new;
    ytot55(i)=ytot5new;


    maxpowdB(i)=maxpowdB(i)+increase;   % Update the total available
                                        % power.
    maxpow(i)=10.^(maxpowdB(i)./10);    % Update the total available
                                        % power.
    Sipsd(i)=10.*log10((maxpow(i)/NF)/4.3125e3);

end

    [y,m] = max(Rsumnew);
    maxpowdB(m)=maxpowdB(m)-increase;   % Update the total available
                                        % power.
    maxpow(m)=10.^(maxpowdB(m)./10);    % Update the total available
                                        % power.
    Sipsd(m)=10.*log10((maxpow(m)/NF)/4.3125e3);

    K=K+1;
    Rsum(K)=Rsumnew(m);
    R11(K)=R11(m);
    R22(K)=R22(m);
    R33(K)=R33(m);
    R44(K)=R44(m);
    R55(K)=R55(m);
    ytot11(K)=ytot11(m);
    ytot22(K)=ytot22(m);
    ytot33(K)=ytot33(m);
    ytot44(K)=ytot44(m);
    ytot55(K)=ytot55(m);


end

% Final Values

Rsumtotal=Rsum(K-1);
R1_final=R11(K-1);
R2_final=R22(K-1);
R3_final=R33(K-1);
R4_final=R44(K-1);
R5_final=R55(K-1);
R_final=[R1_final R2_final R3_final R4_final R5_final];

ytot1_final=ytot11(K-1);
ytot2_final=ytot22(K-1);
ytot3_final=ytot33(K-1);
ytot4_final=ytot44(K-1);
ytot5_final=ytot55(K-1);

%Drawing...

figure
set(gcf,'DefaultLineLineWidth',1)   %  Default line size
set(gcf,'DefaultAxesFontSize',14)   %  Default axis font size
bar(L,R_final)
axis([1 1.6 0 30])
xlabel('Length of the lines','FontSize',16)
ylabel('Mbit/s','FontSize',16)
```

- Program **Iterwaterfill_maxlow.m**

```
function
[R1_final,R2_final,R3_final,R4_final,R5_final,ytot1_final,ytot2_final,
ytot3_final,ytot4_final,ytot5_final] = iterwaterfill_maxlow(Pmax)


% iterwaterfill_maxlow : Iterative Water Filling Algorithm in order to
% obtain the best rate for the pair having the lowest bitrate.

% We are USING WATER-FILLING allways.

% INPUTS:
% Pmax : Maximun power in mW that we have in each cable


% PARAMETERS:
%   ytot1_final : Total transmitted energy in System 1.
%   ytot2_final : Total transmitted energy in System 2.
%   ytot3_final : Total transmitted energy in System 3.
%   ytot4_final : Total transmitted energy in System 4.
%   ytot5_final : Total transmitted energy in System 5.


%   R1_final : Total capacity in Mbits/sec for System 1.
%   R2_final : Total capacity in Mbits/sec for System 2.
%   R3_final : Total capacity in Mbits/sec for System 3.
%   R4_final : Total capacity in Mbits/sec for System 4.
%   R5_final : Total capacity in Mbits/sec for System 5.


defxDSL;

     maxpow1=Pmax; %Update the total available power.
     maxpow2=Pmax; %Update the total available power.
     maxpow3=Pmax; %Update the total available power.
     maxpow4=Pmax; %Update the total available power.
     maxpow5=Pmax; %Update the total available power.

     maxpow1dB=10.*log10(maxpow1); %Update the total available power.
     maxpow2dB=10.*log10(maxpow2); %Update the total available power.
     maxpow3dB=10.*log10(maxpow3); %Update the total available power.
     maxpow4dB=10.*log10(maxpow4); %Update the total available power.
     maxpow5dB=10.*log10(maxpow5); %Update the total available power.



singcap   % With this call we obtain
          % S1,S2,S3,S4,S5,Npsd1,Npsd2,Npsd3,Npsd4,Npsd5

maxpowdB=[maxpow1dB,maxpow2dB,maxpow3dB,maxpow4dB,maxpow5dB];
maxpow=[maxpow1,maxpow2,maxpow3,maxpow4,maxpow5];

%Call waterfilling for System 1
 [y1,ytot1,b1,btotal1,R1] = waterfill(S1,Npsd1,L1,maxpow(1));
%Call waterfilling for System 2
 [y2,ytot2,b2,btotal2,R2] = waterfill(S2,Npsd2,L2,maxpow(2));
%Call waterfilling for System 3
 [y3,ytot3,b3,btotal3,R3] = waterfill(S3,Npsd3,L3,maxpow(3));
```

```
 %Call waterfilling for System 4
[y4,ytot4,b4,btotal4,R4] = waterfill(S4,Npsd4,L4,maxpow(4));
 %Call waterfilling for System 5
 [y5,ytot5,b5,btotal5,R5] = waterfill(S5,Npsd5,L5,maxpow(5));

 R=[R1,R2,R3,R4,R5]   % Bitrate with Water-filling

 K=2;
 [y,ind] = min(R);
 Rlow(K)=R(ind);     % Pair having the lowest bitrate

 R11(K)=R1;
 R22(K)=R2;
 R33(K)=R3;
 R44(K)=R4;
 R55(K)=R5;
 ytot11(K)=ytot1;
 ytot22(K)=ytot2;
 ytot33(K)=ytot3;
 ytot44(K)=ytot4;
 ytot55(K)=ytot5;

increasedB=3;
increase=10^(increasedB/10);
Rlow(1)=Rlow(2)-1;  % to come in the curl


while Rlow(K) > Rlow(K-1)

for i=1:5    % number of Systems

    maxpowdB(i)=maxpowdB(i)-increase;   % Update the total available
                                        % power.
    maxpow(i)=10.^(maxpowdB(i)./10);    % Update the total available
                                        % power.
    Sipsd(i)=10.*log10((maxpow(i)/NF)/4.3125e3);
    Sipsd1=Sipsd(1);
    Sipsd2=Sipsd(2);
    Sipsd3=Sipsd(3);
    Sipsd4=Sipsd(4);
    Sipsd5=Sipsd(5);

    singcap  % With this call we update
             % S1,S2,S3,S4,S5,Npsd1,Npsd2,Npsd3,Npsd4,Npsd5

 %Call waterfilling for System 1
    [y1,ytot1new,b1,btotal1,R1new] = waterfill(S1,Npsd1,L1,maxpow(1));
 %Call waterfilling for System 2
    [y2,ytot2new,b2,btotal2,R2new] = waterfill(S2,Npsd2,L2,maxpow(2));
 %Call waterfilling for System 3
    [y3,ytot3new,b3,btotal3,R3new] = waterfill(S3,Npsd3,L3,maxpow(3));
 %Call waterfilling for System 4
    [y4,ytot4new,b4,btotal4,R4new] = waterfill(S4,Npsd4,L4,maxpow(4));
 %Call waterfilling for System 5
    [y5,ytot5new,b5,btotal5,R5new] = waterfill(S5,Npsd5,L5,maxpow(5));

    Rnew=[R1new,R2new,R3new,R4new,R5new];

    Rlownew(i)=Rnew(ind);

    R11(i)=R1new;
```

```
    R22(i)=R2new;
    R33(i)=R3new;
    R44(i)=R4new;
    R55(i)=R5new;

    ytot11(i)=ytot1new;
    ytot22(i)=ytot2new;
    ytot33(i)=ytot3new;
    ytot44(i)=ytot4new;
    ytot55(i)=ytot5new;


    maxpowdB(i)=maxpowdB(i)+increase;   %Update the total available
                                        %power.
    maxpow(i)=10.^(maxpowdB(i)./10);    %Update the total available
                                        %power.
    Sipsd(i)=10.*log10((maxpow(i)/NF)/4.3125e3);

end

    [y,n] = max(Rlownew);
    maxpowdB(n)=maxpowdB(n)-increase;   %Update the total available
                                        %power.
    maxpow(n)=10.^(maxpowdB(n)./10);    %Update the total available
                                        %power.
    Sipsd(n)=10.*log10((maxpow(n)/NF)/4.3125e3);

    K=K+1;
    Rlow(K)=Rlownew(n);
    R11(K)=R11(n);
    R22(K)=R22(n);
    R33(K)=R33(n);
    R44(K)=R44(n);
    R55(K)=R55(n);
    R=[R11(K),R22(K),R33(K),R44(K),R55(K)];

    ytot11(K)=ytot11(n);
    ytot22(K)=ytot22(n);
    ytot33(K)=ytot33(n);
    ytot44(K)=ytot44(n);
    ytot55(K)=ytot55(n);


end

% Final Values

R1_final=R11(K-1);
R2_final=R22(K-1);
R3_final=R33(K-1);
R4_final=R44(K-1);
R5_final=R55(K-1);
R_final=[R1_final R2_final R3_final R4_final R5_final];

ytot1_final=ytot11(K-1);
ytot2_final=ytot22(K-1);
ytot3_final=ytot33(K-1);
ytot4_final=ytot44(K-1);
ytot5_final=ytot55(K-1);
```

```
%Drawing...


figure
set(gcf,'DefaultLineLineWidth',1)   %  Default line size
set(gcf,'DefaultAxesFontSize',14)   %  Default axis font size
bar(L,R_final)
axis([1 1.6 0 30])
xlabel('Length of the lines','FontSize',16)
ylabel('Mbit/s','FontSize',16)
```