Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Telecommunication Engineering

MESSAGE PASSING ALGORITHMS FOR CLOUD RADIO ACCESS NETWORK OF CELLULAR SYSTEMS

Supervisor
Prof. Stefano Tomasin

Master Candidate
Alessandro Brighente

12 December 2016

Academic Year 2015/2016

Contents

Abstract **iii**

1 Introduction **1**

2 Cloud Radio Access Network technology **3**
2.1 Mobile Network Architecture 3
2.1.1 RRH and BBU Implementation 5
2.2 Advantages and Challenges of the C-RAN Architecture 6
2.3 C-RAN Architecture Deployment Scenarios 7
2.4 System Model:Multi-Cell MIMO Networks 8
2.5 Mathematical Model for MIMO Systems 8
2.6 Multiple Antenna Channels 9
2.7 Cellular Network Modelling 10

3 Message Passing Decoding **13**
3.1 Marginal Functions 13
3.2 Factor Graphs 13
3.3 The Sum-Product Algorithm 14
3.4 Gaussian Message Passing and Channel Sparsification 16
3.4.1 Gaussian Message Passing Algorithm 18
3.4.2 Randomized Gaussian Message Passing Algorithm 21
3.4.3 Computational Complexity Analysis 23
3.4.4 Convergence of the Algorithms 24

4 Channel Capacity **29**
4.1 Full Channel Matrix 30
4.2 Diagonal Channel Matrix 31
4.3 Orthonormal Channel Matrix 31
4.4 Sparse Channel Matrix 32
4.5 Capacity with GMP and RGMP Receivers 33
4.6 Achievable Sum-Rate Results 33

5 Channel Sparsification Modelling and Analysis **37**
5.1 Sparsification Methods 37
5.1.1 Power-Based Sparsification 38
5.1.2 Orthogonal Users-Based Sparsification 39
5.1.3 Antennas-Selection-Based Sparsification 39
5.2 Simulation Results 42
5.2.1 Relative Error and Sparsification Level: Power-Based . . 42

5.2.2 Achievable Sum-Rate: Power-Based 45
5.2.3 Relative Error and Sparsification Level: Orthogonal Users 45
5.2.4 Achievable Sum-Rate: Orthogonal Users 47
5.2.5 Relative Error and Sparsification Level: CBM 49
5.2.6 Achievable Sum-Rate: CBM 51
5.2.7 Relative Error and Sparsification Level: MIBM 52
5.2.8 Achievable Sum-Rate: MIBM 54
5.3 Pre-Coding Sparsification 55
5.3.1 Known Users Pre-Coding 56
5.3.2 Power Based Pre-Coding 57
5.3.3 Known Users plus Powerful Ones 57
5.4 Simulation Results for Pre-Coding Sparsification Methods 58
5.4.1 Known users pre-coding: relative error and channel capacity 58
5.4.2 Power Based Pre-Coding: Relative Error and Achievable Sum-Rate 61
5.4.3 Known Users plus Powerful Ones Pre-Coding: Relative Error and Achievable Sum-Rate 71
5.5 Decoding Computational Complexity Analysis 77
5.5.1 Comparison of Presented Sparsification Methods 82

6 Conclusions **89**

Abstract

Cloud Radio Access Network is a promising architecture for 5G networks. However the computational complexity of the decoding process over this architecture results prohibitively high. In particular, with proposed Gaussian message passing algorithms, the computational complexity grows quadratically with the number of users and base stations. We will introduce two different approaches to overcome this issue: a centralised one, where reduction of the computational complexity is achieved via channel sparsification, and a distributed one, where the reduction of computational complexity is achieved via pre-coding performed at each base station. Notice that the centralised approach still requires that all signals received at each one of the base stations must be sent to the central BBU Pool, whereas the distributed approach reduces the burden of information that has to be sent to the BBU Pool. Hence the latter approach not only reduces the computational complexity but also the amount of information that flows through the network connecting the base stations with the central Pool. Both approaches will have different implementations and will be tested in terms of relative error, computational complexity and achievable sum-rate.

Chapter 1

Introduction

As mobile data transmission volume continuously rises, novel mobile network architectures have been proposed to address the challenges imposed by modern communication scenarios. Cloud Radio Access Network (C-RAN) is one of the above mentioned architectures, which is expected to meet the need of data-hungry users, while maintaining a low price for data usage. The main idea behind C-RANs is to centralize baseband processing in a pool, which is shared among sites, in such a way that adaptation to non-uniform traffic is possible and resources (i.e. base stations) are used more efficiently. However, the signal processing needed to handle communications in such an architecture has a high computational complexity, due to the above mentioned fact that all antennas Remote Radio Heads (RRH) in a certain area refer to a central parallel elaboration unit. This means that operations as full-scale RRH coordination become impractical due to prohibitively high computational complexity. However the centralised BBU pool enables the exploitation of the diversity and interference present in a communication channel for decoding purpose.
Algorithms of message passing decoding have been proposed in [2], where the high computational complexity issue has been addressed via channel sparsification, i.e. by setting to zero some channel matrix coefficients according to a certain rule.
In this work we will first introduce more details about C-RANs, especially about the architecture of such a system and the main advantages it brings respect to classical architectures. Then we will introduce the system model for the communication scenario over which decoding algorithms will be applied, the theory behind message passing decoding and then introduce the message passing algorithms proposed in [2]. Then, conscious of the fact that in large network message passing decoding presents an high computational

complexity, we search for metrics that allow a sparsification of the channel matrix or methods to reduce the amount of data that has to be processed for the decoding of the transmitted signals. All methods will then be analysed in terms of relative error, amount of introduced sparsification and channel capacity of the system exploiting such methods, as well as their computational complexity. This metrics will then be used for a comparison of the proposed algorithms.

Chapter 2

Cloud Radio Access Network technology

2.1 Mobile Network Architecture

C-RAN is mobile network architecture implemented over a cellular network, where baseband resources are pooled in a central unit, so they can be shared among several base stations. The networks we are here introducing are called cellular network as the area covered by a mobile network is divided into cells, each one with its own antennas and responsible for the users located in it.
C-RAN architecture is targeted by mobile network operators and is seen as a typical mobile network supporting supporting soft and green fifth generation technologies (5G). The traditional communication model for cellular networks expects users referring to a Base Station (BS), which handles communications in its coverage area. We can divide the function of a BS in two main groups:

1. Baseband Processing: in this group we find functions for modulation and demodulation, sampling, channel coding and decoding, Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) among others;

2. Radio Functionalities: among which we find digital processing modules, frequency filtering, power amplification and resource allocation.

Traditionally, radio and baseband processing functionalities are integrated into each single base station. A first evolution from this model is provided by BS with RRHs architecture, in which BSs are separated into RRH and Base Band processing Unit (BBU). RRHs are statically assigned to BBUs, so that each cell has its own BS

where the radio module refers to is own BBU. BSs with RRH architecture can be seen in Fig. 2.1 (a).
In order to optimize BBU utilization between heavily and lightly loaded BSs, C-RAN architecture has been introduced. BBUs are centralized in the BBU Pool, a virtualized cluster which is capable of base band processing operations and is shared among cells. An example of this type of architecture can be seen in figure 2.1 (b).
A few words must be spent on the virtualization concept. In fact we stated that the base band processing in C-RANs is carried out by a central virtualized pool. With the word *virtualization* we mean a logical structure which enables the creation of logically isolated networks over possibly shared abstracted physical networks. In the BBU pooling contest this concept can be used to separate data storage, applications, operating systems and management control. Hence the functions of a base station are realized by a Virtual BS, implemented as a software instance.

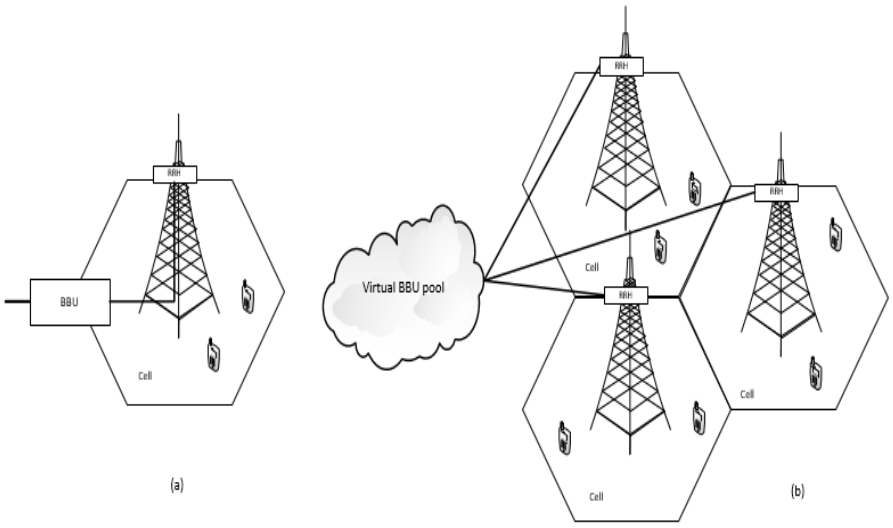

Figure 2.1: Cellular network architecture: BS with RRH (a) and C-RAN (b)

In C-RANs, in order to reduce transport network overhead, the split between BBU and RRH can be implemented in two different ways: fully centralized solution and partially centralized solution.
In fully centralized solutions functionality of levels 1, 2 and 3 of International Organization for Standardization Opens System Interconnection (ISO/OSI) stack are implemented in the central BBU pool, a solution that intrinsically generates high bandwidth data

transmission between the two modules. The second solution, partially centralized, co-locates level 1 functionality in both RRH and BBU in order to reduce the burden in terms of bandwidth on transport links. However this latter solution considerably reduces resource sharing, making impossible to efficiently support advanced feature. Thus, this second approach is considered sub-optimal respect to the first one. The choice of the solution also influences the physical medium to be used. In fact, if the latter solution is implemented, microwave connections between RRHs and BBU can be considered, whereas, if the first solution is implemented, full C-RAN deployment is possible only with fibre connections. Copper-based physical medium solutions are not considered for C-RAN architecture, as Digital Subscriber Lines (DSL)-based access can offer only up to 10-100Mb/s connections.

2.1.1 RRH and BBU Implementation

Even if we used the same name for RRH passing from traditional architectures to C-RAN, requirements and solutions have to be revisited in order to make them compatible with C-RANs. The first main difference is that, in the analysed architecture, signals have to propagate over many kilometres, therefore an additional delay is incurred. The second difference is that C-RAN architecture is expected to have a bigger amount of traffic flowing at higher speed with respect to traditional architectures. Therefore RRHs must support higher data rates, up to some dozens of Gb/s, and hence both radio modules and communication protocols must be revised. Existing RRHs are expected to work in a plug-and-play manner when connected to a C-RAN.
As before stated, a BBU pool is composed by a set of interconnected BBUs. Such connections between modules are expected to work with low latency,high speed and high reliability besides the support of dynamic carrier scheduling and high scalability. A BBU pool intended to manage a medium-sized urban network needs to support 100 base stations distributed in a 5×5 km and it is beneficial when it supports additional services as Content Distribution Network (CDN), Deep Packet Inspection (DPI) and Distributed Service Network (DSN). Hence we need a method for hiding physical characteristics of the BBU pool and enable dynamic resource allocation. This can be obtained with virtualization methods, which are intended to create logically isolated networks over abstracted physical network, enabling a flexible and dynamic sharing of network resources. Hence Virtual Base Stations (VBS) are created as soft-

ware instances realizing the functions of a BS, enabling furthermore a separation of data storage, operating systems and management control. The key features and hence requirements of a virtualized network are isolation, customization and efficient resource allocation. In particular, for what concerns our work, we are interested in:

- computational resources virtualization, including clock synchronization among BSs, to ensure massive parallelism for real time applications, minimization of computational latency between different operating systems and reducing communication latency;
- network resources virtualization, including different authentication and security methods, switching between virtual network operators and different usage of bandwidth resources.

2.2 Advantages and Challenges of the C-RAN Architecture

The C-RAN architecture results advantageous for macro cells as well as for small cells. In fat, a centralized BBU Pool enables an efficient utilization of BBUs and reduction the cost of BSs deployment and operation, while reducing power consumption and increasing flexibility. Furthermore, even mobile network operators can benefit from C-RAN architecture; since response time of application servers is noticeably shorter if data is cached in BBU Pool, they can offer users more attractive Service Level Agreements (SLAs).
One of the key features of the C-RAN architecture is its adaptability to non-uniform traffic. In fact, if we consider the daily traffic distribution in each cell, we can notice that peaks of traffic occur at different hours, especially if we consider cell located in residential areas and in office areas. Hence, since the required base band processing is performed in the central Pool, the overall utilization can be improved. Moreover during the night, since resource demand is lower, some BBUs in the pool can be switched off without affecting the overall network coverage.
Another advantage of C-RAN architecture is the easiness of upgrades. In fact, if we need to expand network coverage area, we can simply connect a new RRH to the already existing BBU Pool. If network coverage must be enhanced, we can split existing cells or add additional RRHs to the BBU Pool. Furthermore, if the overall network capacity shall be increased, we can easily upgrade the BBU

Pool either by adding hardware or by substitution of the existing BBUs with more powerful ones.
However C-RAN architecture comes with requirements as the need for high bandwidth, strict latency and jitter as well as low cost transport network. In fact the links between RRHs and BBU Pool are affected by a huge overhead, and the centralized Pool is expected to support 10-1000 BSs sites, meaning a vast amount of data that moves towards it.
Furthermore we need to manage BBUs cooperation interconnection and clustering. In fact the number and deployment of active BBU/RRHs units within the Pool must be optimized in order to achieve energy savings, whereas cells must be optimally clustered to the Pools in order to prevent BBU Pool and transport network overload. Another challenge imposed by C-RAN architecture is its large Inter Cell Interference (ICI), thus calling for new transmission/reception schemes in order to deal with such problem on the cell edges and achieve optimal throughput at this points.

2.3 C-RAN Architecture Deployment Scenarios

C-RAN architecture is intended as a deployment applicable to most typical scenarios, as macro-cell, micro-cell, pico-cell as well as indoor applications.
In case of green field deployment, we need to place RRHs and BBU Pool according to the network planning. Furthermore physical medium and transport solutions can be designed according to C-RAN specific requirements. In [4] are evaluated the most beneficial C-RAN deployments, concluding that the best strategy is to serve 20-30% of office BSs and 70-80% of residential BSs in one BBU Poo.
C-RAN can be also implemented for capacity boosting. In fact, as stated in [5], the most promising way to increase network capacity is to add new cells. In mobile network within an underlying macro cell we can employ a certain number of small cells to boost network capacity. Small cells can take advantage from C-RAN architecture as required signalling resources are reduced and they are supported by a single BBU Pool. In order to exploit C-RAN for capacity improvement existing BBUs can hence be moved to the central pool, whereas existing RRHs can be left in their positions or new ones can be added.

2.4 System Model:Multi-Cell MIMO Networks

In section 2.2 we have seen that one of the key limiting factors in cellular systems is the frequency reuse factor, which implies inter-cell interference problems. In order to overcome such an issue we introduce the concept of multi-cell cooperation, which has been proved to be one of the best solutions to dramatically increase system's performance in terms of capacity. The key idea behind this type of systems is to literally exploit inter-cell interference by processing data coming from different interfering cells in a jointly manner. Therefore the system mimics a large virtual Multiple-Input Multiple-Output (MIMO) array, inheriting the benefits of such a technology. Current designs for Code Division Multiple Access (CDMA) or frequency hopping spread spectrum systems do allow full frequency re-use in each cell. In particular in CDMA networks a mobile can communicate with several base stations and, thanks to selection diversity, select the best of this connections at any given time. The combination of soft hand-off techniques and power control allow a full frequency reuse in each cell. All comes however with a severe interference condition at the cell edge. Interference is treated as noise at the receiver, and hence all signals come with useful information, which will be exploited in the decoding process.

2.5 Mathematical Model for MIMO Systems

Consider a transmitting device equipped with K transmitting antennas which sends information to a receiving device equipped with N receiving antennas. Each transmitting antenna sends signals to all receiving antennas and each one of such signals experiences path loss and fading, elements that characterise the transmission channel. We will denote as $h_{n,k} \in \mathbb{C}$ the channel coefficient expressing effects of path loss and fading for the signal transmitted from transmitting antenna k to receive antenna n. The overall channel can be hence characterised by the channel matrix which entries are elements of the type $h_{n,k}$, hence resulting in a $\mathbb{C}^{N\times K}$ matrix.
If we denote as $\mathbf{x} = [x_1, x_2, ..., x_K]$ the transmitted signal vector, with entry k being the signal transmitted from antenna k, and with $\mathbf{H}$ the channel matrix, the received signal is

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w} \tag{2.1}$$

where $\mathbf{w} = [w_1, w_2, ..., w_N]^T$, with $[]^T$ denoting the transposed version of the vector, is the additive white Gaussian noise vector, with

w_n being the noise component of the n^{th} channel. We will henceforth assume that each one of the entries of vector $\mathbf{w}$ is a Gaussian random variable with zero mean and variance $\frac{N_0}{2}$.
Fig. 2.2 shows the block schema of a MIMO channel.

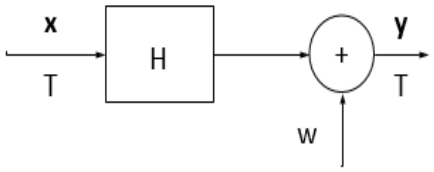

Figure 2.2: Block scheme of a MIMO channel

2.6 Multiple Antenna Channels

Given the previously depicted transmission scenario and the received signal (2.1), we obtain the block scheme of Fig. 2.2 by applying the Singular Value Decomposition (SVD) to the channel matrix $\mathbf{H}$, obtaining

$$\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^H \tag{2.2}$$

where $\mathbf{U}$ is a $N \times N$ unitary matrix, $\mathbf{V}$ is a $K \times K$ unitary matrix and $\mathbf{D}$ is an $N \times K$ diagonal matrix, with all zero elements except the main diagonal, that contains the *singular values*, and they correspond to the square roots of eigenvalues of either $\mathbf{H}\mathbf{H}^H$ or $\mathbf{H}^H\mathbf{H}$. If we denote

$$\mathrm{x}_p = \mathbf{V}\mathbf{x} \tag{2.3}$$

and

$$\mathrm{y}_p = \mathbf{U}^*\mathbf{y} \tag{2.4}$$

and apply the inverse mapping $\mathbf{V}$ and $\mathbf{U}$ to the block scheme of Figure 2.2 we obtained the block representation of Figure 2.3.

Since $\mathbf{D}$ is a diagonal matrix, by applying a pre-coding and post processing operations respectively at transmitter and receiver, we

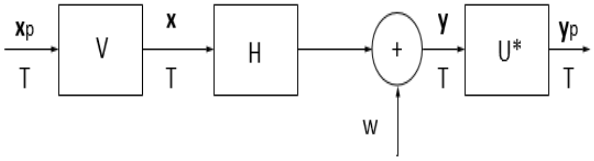

Figure 2.3: Block scheme for MIMO channels with SVD for H

can obtain the representation of the MIMO channel as a parallel of channels of the type

$$\mathbf{y}_p = \mathbf{D}\mathbf{x}_p + \mathbf{w}_p \tag{2.5}$$

where

$$\mathbf{w}_p = \mathbf{U}^H \mathbf{w} \tag{2.6}$$

We can notice that the input-output relations in (2.5) and (2.6) assume slightly different forms depending on the values of K and N. In particular, if $N \leq K$ we have

$$y_{p,n} = \mathbf{D}_n x_{p,n} + w_{p,n} \tag{2.7}$$

for $n = 1, ..., N$, so that contributions $x_{p,N+1}, ..., x_{p,K}$ do not contribute to the received signal. If we instead have $N > K$ the received signal is

$$y_{p,n} = \mathbf{D}_n x_{p,n} + w_{p,n} \tag{2.8}$$

for $n = 1, ..., K$, whereas

$$y_{p,n} = w_{p,n} \tag{2.9}$$

for $n = K + 1, ..., N$, i.e. outputs are associated to noise only for this index set. Given this consideration we can compute the number of *active channels* as $\min(N, K)$.

2.7 Cellular Network Modelling

We stated that, in order to exploit ICI, we can see the C-RAN-based cellular system as a virtual MIMO. Fig. 2.4 presents one of the possible communication scenarios. We consider N_c cells with BS equipped with N_a omnidirectional receive antennas, whereas users are N_u per cell and equipped with single antenna devices. The total number of receiving antennas is hence $N = N_c\, N_a$, whereas the total

number of users is $K = N_c\, N_u$.
We suppose that the channel matrix $\mathbf{H}$ is composed by elements of the type

$$\mathbf{h}_{n,k} = \gamma_{n,k} d_{n,k}^{-\frac{\alpha}{2}}, \tag{2.10}$$

where $\gamma_{n,,k}$ is the independent identically distributed (i.i.d) Rayleigh fading coefficient, with mean $m_\gamma = 0$ and variance $\sigma_\gamma^2 = 1$, $d_{n,k}$ is the distance between the k^{th} user and the n^{th} antenna of cell c and α being the path loss coefficient. If we denote as $\mathcal{A}(c) = [n_1, n_2, ..., n_{N_a}]$ the set of coefficients of antennas of cell c, then $n \in \mathcal{A}(c)$. The overall channel matrix is a full $\mathbb{C}^{N \times K}$ matrix.
Furthermore, we will assume that the noise component is a complex normal random variable with zero mean and $N_0\mathbf{I}$ covariance matrix. We will denote the noise component as $\mathbf{w} = [w_1, w_2, ..., w_{N_a}]^T$, with i.i.d. entries.
We also assume unitary variance of the transmitted signal, hence $E[\mathbf{x}_k \mathbf{x}_k^H] = \mathbf{I}$.
Notice that power control techniques can be applied to the system in order to obtain a better performance. This implies a pre-multiplication of the transmitted signal by $P_k^{\frac{1}{2}}$, where P_k is the transmit power allocated for the k^{th} user. We assume that all users are allocated the same transmission power, i.e. $P_k = P\ \forall k \in \{1, 2..., K\}$. Equation (2.1) is hence updated for power control by a simple multiplication, obtaining

$$\mathbf{y} = P^{\frac{1}{2}}\mathbf{H}\mathbf{x} + \mathbf{w} \tag{2.11}$$

Now we assume that, on average, noise power is -10dB. Since users signal powers decrease with the inverse of the distance from the BS, we notice that, if users transmit with power $P = 0$dB and is situated on the cell border, at the receiver it will suffer an SNR of -16dB. We hence multiply all channel matrix coefficients by a factor 50 in order to bring border cell's users SNR to 0dB.

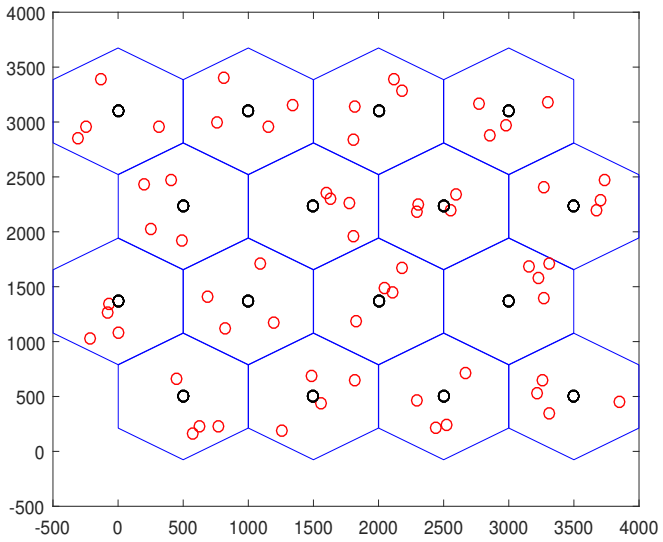

Figure 2.4: Cellular network scenario for message passing algorithms extension: users are denoted with red circles, while antennas with back circles

Chapter 3

Message Passing Decoding

3.1 Marginal Functions

Let us consider a collection of K variables $\mathbf{x} = [x_1, x_2, ..., x_K]$, each one assuming values in a finite domain or alphabet $\mathcal{X}$. Consider a function $\mathbf{f}(\mathbf{x})$, i.e. a function of the entries of vector $\mathbf{x}$ with domain $\mathcal{S} = \mathcal{X} \times \mathcal{X} \times ... \times \mathcal{X}$ of dimension K and co-domain $\mathcal{R}$. If we assume that sum operations in $\mathcal{R}$ are well defined, then associated to each function $\mathbf{f}(\mathbf{x})$ are K marginal functions $f_k(x_k)$. In order to obtain the value $f_k(x)$ for each $x_k = x \in \mathcal{X}$ we must sum $\mathbf{f}(\mathbf{x})$ over all the configurations that have $x_k = x$ and we denote this sum as

$$f_k(x) = \sum_{\sim x_k} \mathbf{f}(x_1, x_2, ..., x_K) = \sum_{x_1} \sum_{x_2} ... \sum_{x_{k-1}} \sum_{x_{k+1}} ..., \sum_{x_K} \mathbf{f}(x_1, x_2, ..., x_). \tag{3.1}$$

With message passing algorithm we aim at finding efficient ways to compute marginal functions by exploiting the way in which global function factors. In the next section we will introduce factor graphs, a graphical representation of procedures needed to compute marginals of functions.

3.2 Factor Graphs

A Factor Graph (FG) is a bipartite graph which expresses the factorization (3.1). The representation of a factor graph is hence based on the function to expressed and is composed by:

- a variable node for each one of the variables x_k, denoted with a circle;
- a factor node for each one of the local functions f_k, denoted with a square;

- an edge connecting variable node x_k to factor node f_k if and only if x_k is argument of f_k.

An example of a FG is showed in Fig. 3.1. In order to compute marginal values we use Message Passing (MP) algorithms applied over FGs. Notice that MP algorithms can be applied on loopy-free FGs as well as on loopy FGs. However convergence cannot be assured on the latter. We will hence first introduce the sum-product algorithm, which computes marginal values for local functions and hence discuss convergence of the presented algorithm.

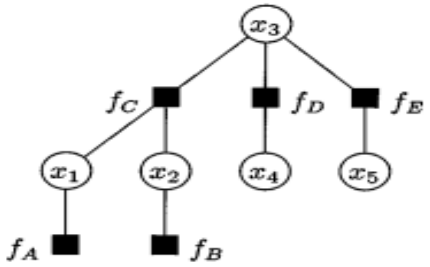

Figure 3.1: Factor graph example. Fig taken from [7].

3.3 The Sum-Product Algorithm

The message passing algorithm starts at the leaves of the FG, where each leaf variable node sends an identity function message to its parent and each leaf factor node f_k sends a description of f_k to its parent. Once a vertex receives a message from all of its children it computes the message to send to its parent. If such a vertex is a variable node it simply sends the product of the messages received from its children, whereas a factor node f_k with parent x_k forms the product of f_k with all messages received from its children and then sums this result as in (3.1), i.e. $\sum_{\sim x_k}$. Notice that with product of the messages we mean an appropriate description of the point-wise product of the corresponding functions. The computation ends at the root node, where marginal value is computed as product of all incoming messages. The operations executed by sum-product algorithm can be summarized in the following 4 rules:

1. half edges for variables x_i shall be interpreted as edges with leaf factor $f_j(x_i) = 1$;

2. the message exiting a *leaf* node is $\mu_{f_j \to x_i}(x_i) = f_j(x_i)$;
3. the message exiting a *factor* node shall be updated via the *sum-product* rule

$$\mu_{f_j \to x_i}(x_i) = \sum_{\sim x_i} f_j(x_i, x_a, x_b ...) \mu_{x_a \to f_j}(x_a) \mu_{x_b \to f_j}(x_b) ... \quad (3.2)$$

4. the *marginal* on variable x_i is obtained as

$$h(x_i) = \mu_{f_k \to x_i}(x_i) \mu_{f_j \to x_i}(x_i), \quad (3.3)$$

where f_k and f_j are two adjacent nodes.

Among the others, a class of functions which can be represented via FGs is the one of probability distributions. Given the transmitted signal vector $\mathbf{x}$ and the corresponding received signal vector $\mathbf{y}$, for each fixed observation $\mathbf{y}$ the corresponding a-posteriori probability (APP) distribution $p(\mathbf{x}|\mathbf{y})$ for the components of $\mathbf{x}$ is proportional to the function

$$\mathbf{f}(\mathbf{x}) = p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}), \quad (3.4)$$

where $p(\mathbf{y}|\mathbf{x})$ is the conditional probability of received signal vector $\mathbf{y}$ when $\mathbf{x}$ is transmitted and $p(\mathbf{x}$ is the a-priori probability distribution of the transmitted signal vector $\mathbf{x}$. Since we fixed the value of $\mathbf{y}$, $\mathbf{f}$ in (3.4) is a function of $\mathbf{x}$ only with $\mathbf{y}$ as parameter. Hence we can write $p((\mathbf{y}|\mathbf{x})$ as $f_{\mathbf{y}}(\mathbf{x})$, meaning that the parametric form is the same when different decoding instances are analysed, whereas parameter $\mathbf{y}$ changes. Furthermore if the channel is memoryless we have the factorization

$$p(y_1, y_2, ...y_N | x_1, x_2, ..., x_N) = \prod_{n=1}^{N} p(y_i|x_i) \quad (3.5)$$

and hence

$$\mathbf{f}(\mathbf{x}) = \prod_{n=1}^{N} p(y_i|x_i) p(x_i). \quad (3.6)$$

Hence a FG can be built with factor nodes being the different $p(y_i|x_i)p(x_i)$.
This graph is characterized by loops, meaning that the MP algorithm is implemented in an iterative form with no natural termination. Hence messages will be passed multiple times over the same edge. In practice this means that, if MP algorithm converges, it is not clear whether the results are good approximations of the marginals. We demand to [8] for sufficient conditions for convergence of the sum-product algorithm.

3.4 Gaussian Message Passing and Channel Sparsification

We now apply the theory introduced for message passing algorithm to C-RAN architecture. We assume that transmitted signal entries $\mathbf{x} = \{x_1, x_2, ..., x_K\}$ are distributed as i.i.d. complex Gaussian variables, so that, together with the received signal (2.1), they are jointly Gaussian. We can hence apply the optimal linear detector minimum mean square error (MMSE), which returns the decision statistics of the transmitted signal as

$$\hat{\mathbf{x}} = P^{\frac{1}{2}}\mathbf{H}^H(P\mathbf{H}\mathbf{H}^H + N_0\mathbf{I})^{-1}\mathbf{y}, \tag{3.7}$$

where $\mathbf{y} = [y_1, y_2, ..., y_N]^T$. If our aim was to solve this problem with the matrix inversion (3.7) the computational complexity would be $\mathcal{O}(N^3)$. This results to be prohibitively high in large networks with hundreds or thousands of users and RRHs, i.e. very large $\mathbf{H}$. So we wish to derive an algorithm with linear computational complexity in the number of users and/or RRHs.
Thanks to the hypothesis, the MMSE results to be the maximum a posteriori probability (MAP) detector [3], where the a-posteriori probabilities are of the type $p(\mathbf{x}|\mathbf{y})$.
Notice that

$$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}), \tag{3.8}$$

which, since we assume that all entries in $\mathbf{x}$ are independent, can be rewritten as

$$p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}) \prod_{j=1}^{K} p(x_j), \tag{3.9}$$

where

$$p(y_n|\mathbf{x}) = \frac{1}{\sqrt{2\pi N_0}} e^{-\frac{(y_n - \mathbf{h}_n^H\mathbf{x})(y_n - \mathbf{h}_n^H\mathbf{x})^H}{2N_0}}, \tag{3.10}$$

with $\mathbf{h}_n$ being the n^{th} row of the channel matrix, and

$$p(x_k) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x_k x_k^*}{2}}, \tag{3.11}$$

with $[]^*$ denoting the complex conjugate.
The idea is to exploit the factorization in (3.8) and (3.9) to resolve Maximum a-Posteriori Probability (MAP) problem by iteratively update the values of $p(y_n|\mathbf{x})$ and $\hat{x}_k$ by means of the max-product algorithm.
An example of a factor graph representation of a C-RAN is given

in figure 3.2, where input symbols x_i are treated as variable nodes, whereas function nodes are given by input symbol probabilities $p(x_k)$ at the user side and by probabilities $p(y_n|\mathbf{x})$ at the RRH side.

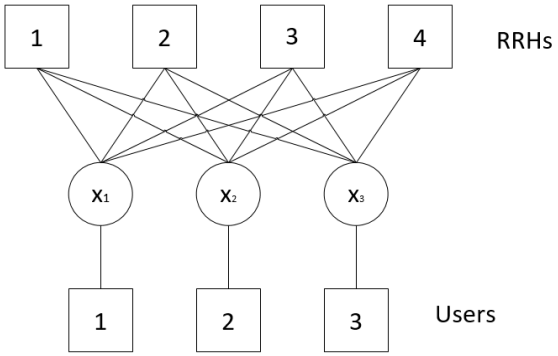

Figure 3.2: Factor graph representation of a C-RAN with 3 users and 4 RRHs

However convergence cannot be guaranteed, because of the fact that the graph used to represent the scenario includes loops, ad it has been demonstrated that message passing over loopy graphs does not always converge. We hence introduce the channel sparsification. With this approach we want to overcome the scalability problem. Since users and BSs can be distributed over large areas, RRHs may receive only few strong signals, the ones coming from the nearest users. Therefore we can sparsify the channel matrix by neglecting entries associated to distances above a certain threshold d_0. In particular, we can say that entry $h_{n,k}$ of the channel matrix is different from zero if and only if the distance between user k and RRH n is lower than a certain threshold distance.
Denoting the sparsifyied channel matrix with $\hat{\mathbf{H}}$ and defining $\dot{\mathbf{H}} = \mathbf{H} - \hat{\mathbf{H}}$, the received signal can be expressed as

$$\mathbf{y} = P^{\frac{1}{2}}\hat{\mathbf{H}}\mathbf{x} + P^{\frac{1}{2}}\dot{\mathbf{H}}\mathbf{x} + \mathbf{n}, \tag{3.12}$$

and the MMSE estimation of $\mathbf{x}$ becomes

$$\hat{\mathbf{x}} = P^{\frac{1}{2}}\hat{\mathbf{H}}^H(P\hat{\mathbf{H}}\hat{\mathbf{H}}^H + \hat{N}_0\mathbf{I})^{-1}\mathbf{y}, \tag{3.13}$$

with

$$\hat{N}_0 = PE\left[\sum_{j\neq k}|\dot{\mathbf{H}}_{n,j}|^2\right] + N_0. \tag{3.14}$$

After the sparsification of the channel matrix we can rewrite the

probability density function (3.8) as

$$p(\mathbf{x}|\mathbf{y}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_{\mathcal{I}_i}) \prod_{j=1}^{K} p(x_j), \tag{3.15}$$

where $\mathcal{I}_n$ is the set of indices of users whose distance with RRH n is less than the specified threshold.
We can state that the number of messages that will be exchanged will be lower respect to full channel matrix case, since only users located in a circle centred in a BS and with constant radius equal to the distance threshold will be served by a certain RRH. The threshold value is assumed to be predetermined and constant regardless of the network size. This assumption is valid for a scenario where the number of involved RRHs N scales as the number of users K, where it has been demonstrated that the distance threshold does not increase with the network size to achieve a certain signal to interference plus noise ratio [3].

3.4.1 Gaussian Message Passing Algorithm

The Gaussian Message Passing (GMP) algorithm is the MP algorithm where all variables are Gaussian distributed. This is the case of our scenario, since both $\{x_k\}$ and $\{y_n\}$ are Gaussian distributed, even messages will follow the same distribution and will be hence characterized by their mean and variance. This fact also characterizes the name of the algorithm, i.e., the adjective Gaussian refers to the fact that entries follow this distribution.
Given the iteration number t, the mean and variance information messages sent by check node $p(y_n|\mathbf{x})$ to variable node x_k are denoted as $m^{(t)}_{y_n \to x_k}$ and $v^{(t)}_{y_n \to x_k}$. At the same manner, we denote messages sent by variable node x_k to factor node $p(y_n|\mathbf{x})$ as $m^{(t)}_{x_k \to y_n}$ and $v^{(t)}_{x_k \to y_n}$.
Given the channel sparsification method, we can state that messages will be exchanged only among users located in a radius d_0 from a certain RRH. The algorithm steps are shown in Algorithm 1.
The algorithm receives as input the channel matrix, whether sparsified or not, and the received signal vector and returns the estimates of the components of the transmitted signal vector. We see that algorithm steps are repeated estimates of mean and variance of the components of both transmitted and received signals. Initialization is based on the assumptions for the statistics of the transmitted signal, i.e. We notice that estimates of mean and variance values are firstly computed and then updated in an iterative way, until a

stopping criterion is satisfied.
The convergence of the algorithm is poorly improved with this approach. In fact it is known that tree-type factor graph guarantee message passing convergence, but this is not our case. Precisely, the convergence of the algorithm depends on the schedule of messages

to be exchanged.

Data: $\hat{\mathbf{H}}$,$\mathbf{y}$
Result: $\hat{x}_k \forall k$
initialization: $t = 0$, $m^{(0)}_{x_k \to y_n} = 0$,$v^{(0)}_{x_k \to y_n} = 1 \forall k, n$;
while *stopping condition unsatisfied* **do**
  $t \leftarrow t + 1$;
  for *n=1 to N* **do**
    for *k=1 to K* **do**
      if $\hat{H}_{n,k} \neq 0$ compute

$$v^{(t)}_{y_n \to x_k} = \frac{1}{P_k |\hat{H}_{n,k}|^2} \left(\hat{N}_0 + P_k \sum_{j \neq k} |\hat{H}_{n,j}|^2 v^{(t-1)}_{x_j \to y_n} \right) \tag{3.16}$$

$$m^{(t)}_{y_n \to x_k} = \frac{1}{P_k^{\frac{1}{2}} \hat{H}_{n,k}} \left(y_n - P_k^{\frac{1}{2}} \sum_{j \neq k} \hat{H}_{n,j} m^{(t-1)}_{x_j \to y_n} \right) \tag{3.17}$$

    end
    for *k=1 to K* **do**
      if $\hat{H}_{n,k} \neq 0$ compute

$$v^{(t)}_{x_k \to y_n} = \left(\sum_{\hat{H}_{j,k} \neq 0, j \neq n} \frac{1}{v^{(t)}_{y_j \to x_k}} + 1 \right)^{-1} \tag{3.18}$$

$$m^{(t)}_{x_k \to y_n} = v^{(t)}_{x_k \to y_n} \left(\sum_{\hat{H}_{j,k} \neq 0, j \neq n} \frac{m^{(t)}_{y_j \to x_k}}{v^{(t)}_{y_j \to x_k}} \right) \tag{3.19}$$

    end
  end
end
 for *k=1 to K* **do**
  compute:

$$v_k = \left(\sum_{\hat{H}_{n,k} \neq 0} \frac{1}{v^{(t)}_{y_n \to x_k}} + 1 \right)^{-1} \tag{3.20}$$

$$\hat{x}_k = v_k \left(\sum_{\hat{H}_{n,k} \neq 0} \frac{m^{(t)}_{y_n \to x_k}}{v^{(t)}_{y_n \to x_k}} \right) \tag{3.21}$$

end

Algorithm 1: Gaussian Message Passing (GMP)

3.4.2 Randomized Gaussian Message Passing Algorithm

The main issue arising in Algorithm 1 is convergence. In fact, as before stated, message passing over loopy graphs is not guaranteed to converge. In order to address this problem we consider a randomized scheduling update of messages obtaining the Randomized Gaussian Message Passing (RGMP) algorithm showed in Algorithm 2. While in Algorithm 1 messages are updated in parallel in a synchronous message passing, here the update is executed in a sequential randomly permuted order. In fact, as we can see in Algorithm 2, before the computation of the messages, a permutation σ of the set $\{1, ..., K\}$ is picked at random, following a uniform distribution, i.e. every permutation has the same probability of being chosen. Then messages are sequentially updated following the indexes of σ. In order to clarify this aspect let us consider the permutation of the set $\{1, 2, 3\}$ obtained at the n^{th} iteration being $\sigma = \{3, 1, 2\}$. The message passing update first considers user's variable node x_3, and first updates all messages on the edges connecting such a node. Then the update of messages on edges connecting x_1 and then x_2. We can see that the computational complexity is unchanged respect to Algorithm 1, the only difference being the scheduling approach, which tries to mitigate the loopy effect on algorithm's convergence.

Data: $\hat{\mathbf{H}}$,$\mathbf{y}$
Result: $\hat{x}_k \forall k$
initialization: $t = 0$, $m^{(0)}_{x_k \to y_n} = 0$,$v^{(0)}_{x_k \to y_n} = 1 \forall k, n$;
while *stopping condition unsatisfied* **do**
  $t \leftarrow t + 1$;
  chose randomly a permutation σ of $\{1, ..., K\}$;
  for *n=1 to N* **do**
    for *i=1 to K* **do**
      if $\hat{H}_{n,\sigma(i)} \neq 0$ compute

$$v^{(t)}_{y_n \to x_{\sigma(i)}} = \frac{1}{P_k |\hat{H}_{n,\sigma(i)}|^2} \Bigg(\hat{N}_0 + P_{\sigma(i)} \sum_{j<i} |\hat{H}_{n,\sigma(j)}|^2 \; v^{(t)}_{x_{\sigma(j)} \to y_n} + P_{\sigma(i)} \sum_{j>i} |\hat{H}_{n,\sigma(j)}|^2 v^{(t-1)}_{x_{\sigma(j)} \to y_n} \Bigg) \tag{3.22}$$

$$m^{(t)}_{y_n \to x_{\sigma(i)}} = \frac{1}{P^{\frac{1}{2}}_{\sigma(i)} \hat{H}_{n,\sigma(i)}} \Bigg(y_n - P^{\frac{1}{2}}_{\sigma(i)} \sum_{j<i} \hat{H}_{n,\sigma(j)} \; m^{(t)}_{x_{\sigma(j)} \to y_n} - P^{\frac{1}{2}}_{\sigma(i)} \sum_{j>i} \hat{H}_{n,\sigma(j)} m^{(t-1)}_{x_{\sigma(j)} \to y_n} \Bigg) \tag{3.23}$$

    end
    for *i=1 to K* **do**

$$v^{(t)}_{x_{\sigma(i)} \to y_n} = \Bigg(\sum_{\hat{H}_{j,\sigma(i)} \neq 0, j \neq n} \frac{1}{v^{(t)}_{y_j \to x_{\sigma(i)}}} + 1 \Bigg)^{-1} \tag{3.24}$$

$$m^{(t)}_{x_{\sigma(i)} \to y_n} = v^{(t)}_{x_{\sigma(i)} \to y_n} \Bigg(\sum_{\hat{H}_{j,\sigma(i)} \neq 0, j \neq n} \frac{m^{(t)}_{y_j \to x_{\sigma(i)}}}{v^{(t)}_{y_j \to x_{\sigma(i)}}} \Bigg) \tag{3.25}$$

    end
  end
end
for *k=1 to K* **do**
  compute:

$$v_k = \Bigg(\sum_{\hat{H}_{n,k} \neq 0} \frac{1}{v^{(t)}_{y_n \to x_k}} + 1 \Bigg)^{-1} \tag{3.26}$$

$$\hat{x}_k = v_k \Bigg(\sum_{\hat{H}_{n,k} \neq 0} \frac{m^{(t)}_{y_n \to x_k}}{v^{(t)}_{y_n \to x_k}} \Bigg) \tag{3.27}$$

end

Algorithm 2: Randomized Gaussain Message Passing (RGMP).

3.4.3 Computational Complexity Analysis

Let us first assume that no channel sparsification method is applied and, hence, the channel matrix results to be a full $\mathbb{C}^{NxK}$, with N and K being respectively the number of RRHs and users. Analysing Algorithms 1 and 2, we can see that the main iterations consist on two cycles, one into the other. We can start as an example to iterate on the index k of the users and then, inside this main cycle, iterate on the number n of users. Due to the message updating schedule required by the algorithms, i.e. the use of values for the mean and variance of signal computed at iteration t or $t-1$, we have to include two cycles on the index value of the user n, where, for each one of the values of n, a sum over K elements is performed as a cycle. Given the full channel matrix $\mathbf{H}$, and given the check on the value of $H_{n,k}$ made by the algorithms before starting an iteration, we can see that if all channel matrix entries are non-zero the complexity of the algorithm is $\mathcal{O}(NK^2)$. If the number of users and BSs grows in the same way, i.e. $K \propto N$ the complexity is $\mathcal{O}(N^3)$, equal to the one obtained for MMSE resolution.
If we consider instead a sparsified channel matrix the number of iterations will be reduced by the fact that some of the users are located at a distance beyond the threshold, and hence the corresponding entry in the channel matrix will be set to zero. Therefore, considering the condition $H_{n,k} \neq 0$ before every computation of the algorithm, each one of the N iterations on the BSs will not necessarily include an iteration for each users. However, the computational complexity results to be $\mathcal{O}(N^3)$, given the same considerations made for the previous case.
Same considerations can be made for Random Gaussian Message Passing, where the only difference is the message update schedule.
We can hence state that the computational complexity of the algorithms depend on the number of users and on the number of BSs. However no improvement has been made on the computational complexity, since we started from an $\mathcal{O}(N^3)$ for the pure MMSE to a $\mathcal{O}(NK^2)$ with message passing decoding.

3.4.4 Convergence of the Algorithms

Message passing algorithms are characterized by the number of iterations over which they are run. In particular, if we take a closer look to Algorithms 1 and 2, we can notice that, before the estimation of the variance, messages are exchanged among FG nodes for a certain number of times. Different number of iterations lead to different results and hence different errors.
As a first observation we present the results obtained with the set-up used in [3]. We consider a network composed by 4 cells and hence 4 BSs, with a single user per BS, ie.e $N = 4$,$K = 4$. The channel is characterized by an SNR, defined as P/N_0, of 100dB and by transitions determined by matrix

$$H = 10^{-5} \begin{pmatrix} -0.1458+0.2401i & -2.0998-0.7353i & -2.1459-2.0284i & 21.5306+6.5308i \\ 17.7199+18.8315i & 1.8431-2.4183i & 5.7441+2.0536i & 0.4837-3.0383i \\ 5.1714-14.5292i & 0.1184-1.5314i & -10.3012+0.1049i & 2.4388-0.8546i \\ -25.2041-16.2758i & 1.1697-0.3792i & 2.2858-0.2858i & 6.0425-2.6317i \end{pmatrix} \tag{3.28}$$

We consider the received signal

$$\mathbf{y} = [1.6847-7.1280i, -20.9794+3.6052i, -3.0214+3.8041i, 21.5306+6.5308i]^T \tag{3.29}$$

In order to evaluate the performance of the decoding algorithm we show the relative error versus the number of iterations, where we define the relative error as

$$\epsilon = \frac{||PH^H H x^{(t)} - P^{\frac{1}{2}} H^H \mathbf{y}||}{||P^{\frac{1}{2}} H^H \mathbf{y}||} \tag{3.30}$$

Figure 3.3 reports the results obtained with GMP and RGMP. We can see that random scheduling for message update performs better than sequential update with order $\{1, 2, 3, 4\}$. In particular we can notice that a larger number of iterations does not necessarily mean better estimates. In fact, with GMP we reach a minimum relative error at about 22 iterations. Different considerations should be made regarding the number of iterations of RGMP, where after about 20 iterations the relative error becomes negligible and hence further efforts in decoding process do not lead to significantly better results. We can hence state that RGMP is preferable over GMP because the same relative error is reached with a smaller number of iterations. Such a parameter does not significantly influences system's performance in terms of speed of the decoding process in the presented scenario where only 4 users and 4 BSs are present, but will be significantly impact decoding time in larger networks.

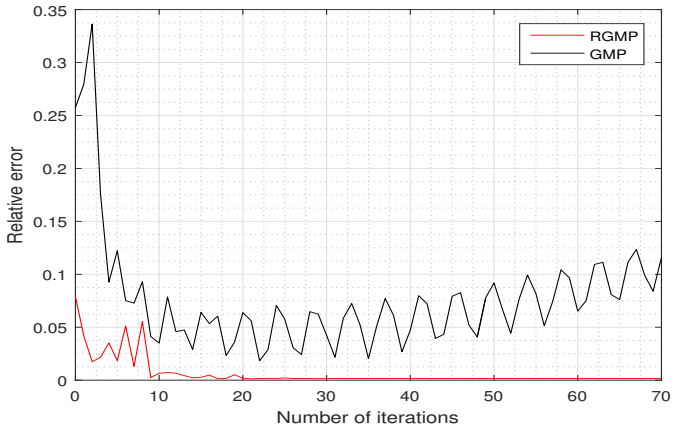

Figure 3.3: Relative error vs. number of iterations with Fan et al. set up.

We now investigate the effects of noise on system's performance in terms of relative error. We compared a set-up similar to the one presented in [3], where the number of BSs is equal to the number of users, i.e. $N = K = 4$, but where we generate random transmitted signal and channel matrix. The transmitted signal vector is composed by 4 entries, generated as complex Gaussian random variables with zero mean and unitary variance, while the channel matrix is generated as zero mean complex Gaussian random variables as well, but with variance $1/N$. The received signal is affected by complex Gaussian noise with zero mean and variance N_0. We tested the algorithms with two SNR values, 5dB and 100 dB, where the transmission power has been maintained constant. The obtained results strongly depend on the realization of the channel matrix for high SNR values, but we mainly observed two behaviours presented in Fig. 3.4 and 3.5, where in both figures upper case shows 5dB SNR results whereas the lower one shows 100dB SNR results. We can see that in both cases a relative error equal to 0 can be obtained, the main difference is the way this result is reached with GMP. In fact we can see that in figure 3.4 a zero relative error can be obtained with a small number of iterations and, while increasing such parameter the error diverges. Figure 3.5 presents the opposite case, i.e. we start with an high relative error and by increasing the number of iterations it decreases until reaching almost zero. We can hence state that about 20 iterations is a good choice for GMP, since it can guarantee a small relative error for both cases. We can also state that RGMP does not diverge. In particular we can see that in figure 3.4 a zero relative error is reached for every iteration number,

whereas in figure 3.5 we need approximatively the same number of iterations required by GMP in order to obtain a small relative error. The 5 dB SNR case performs approximatively the same in both cases, the main difference is the number of iterations after which GMP reaches the minimum relative error, whereas RGMP requires approximatively the same number of iterations to reach such a minimum.

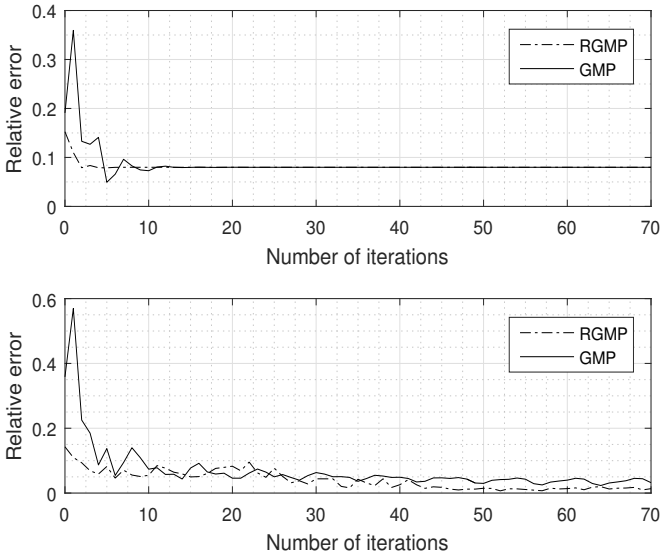

Figure 3.4: Relative error vs. number of iterations with random signal generation case 1: upper figure 5dB SNR, lower 100dB SNR.

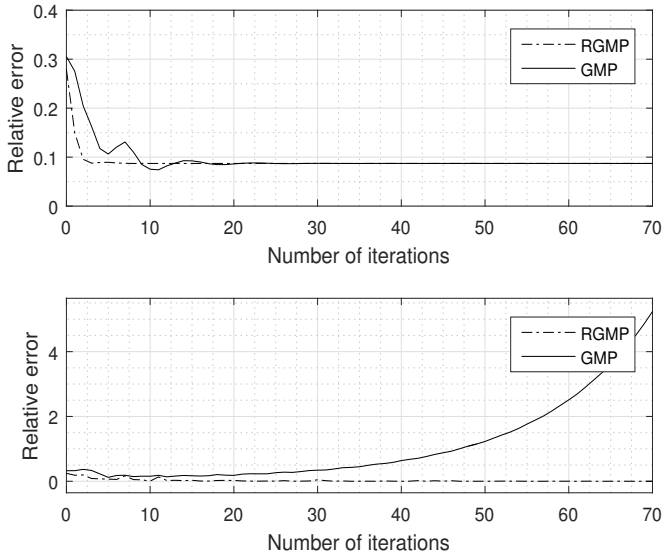

Figure 3.5: Relative error vs. number of iterations with random signal generation case 2: upper figure 5dB SNR, lower 100dB SNR.

Chapter 4

Channel Capacity

In this chapter we aim to compute the capacity of the transmission links of the previously depicted scenario. We define the capacity of a transmission link as the upper limit rate at which information can be reliably transmitted and we can compute it as

$$C = \max_{p_x(a)} I(x;y) \,[\text{bit/ch.use}] \tag{4.1}$$

with $I(x;y)$ representing the mutual information.
Since the communication channel is characterized by matrix $\mathbf{H}$, which is assumed to be square given the fact that we assumed that users and BSs grown in the same way, we compute the capacity for different constant matrices. In particular, we consider:

1. a full $\mathbb{C}^{N\times K}$ channel matrix,
2. an orthogonal $\mathbb{C}^{N\times K}$ channel matrix,
3. a diagonal $\mathbb{C}^{N\times K}$ channel matrix,
4. a sparse $\mathbb{C}^{N\times K}$ channel matrix.

We henceforth consider the same communication scenario described in previous chapters, where all signals (transmitted, noise and received) are Gaussian distributed.
Before starting the discussion about capacities, we recall a fundamental result, named after Shannon-Hartley theorem, which states what follows:

Theorem 4.0.1. *The capacity C of a channel with bandwidth B and with additive white Gaussian noise is given by*

$$C = B\log_2(1+\Gamma)\,[\text{bit/s}], \tag{4.2}$$

with Γ being the signal-to-noise ratio.

This result will be used in the following sections.

4.1 Full Channel Matrix

Let us consider a channel matrix of the type

$$H = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,K} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N,1} & h_{N,2} & \cdots & h_{N,K} \end{pmatrix} \tag{4.3}$$

Given the definition of the received signal expressed by (2.1), we can expand it at the n^{th} receiver as

$$y_n = P_1^{1/2} h_{n,1} x_1 + P_2^{1/2} h_{n,2} x_2 + ... + P_K^{1/2} h_{n,K} x_K + w_n. \tag{4.4}$$

Such a representation enables the view of the component of the received signal. We can identify the useful signal as $P_1 H_{n,n} x_n$, and hence all the other elements depending on $x_{k \neq n}$ can be identified as interference. Hence we can rewrite the received signal as

$$y_n = P_n^{1/2} h_{n,n} x_n + s_n. \tag{4.5}$$

where s_n represent the contributions of both noise and interference at the receiver's BS n.
If now recall the fact that both $\mathbf{x}$ and $\mathbf{w}$ are Gaussian distributed, in particular with zero mean and variance respectively one and σ_w^2, we can define the statistic of the received signal s_n. In fact, as such a signal results in being a sum of Gaussian variables, it is Gaussian distributed too, with zero mean and variance $\sigma_{s_n}^2 = \sum_{j=1, j \neq n}^{K} P_j |h_{n,j}|^2 + \sigma_w^2$.
If we implement a MMSE receiver, in order to find estimates of the transmitted signal, the received signal $\mathbf{y}$ will be multiplied by the matrix $P^{\frac{1}{2}} \mathbf{H}^H (P \mathbf{H} \mathbf{H}^H + N_0 \mathbf{I})^{-1}$, which will be referred to as MMSE matrix. Therefore, both the useful signal and interference/noise components s_n, will be multiplied by this matrix, operation that further modifies the statistics of the signal components. If we denote with M the MMSE matrix and with D the matrix resulting from the multiplication of the channel matrix H with the MMSE matrix, we can state that the useful signal will be Gaussian distributed with zero mean and variance given by

$$\sigma_{u_n} = P |D_{n,n}|^2, \tag{4.6}$$

whereas s_n will be Gaussian distributed with zero mean and variance

$$\sigma_{s_n}^2 = \sum_{j=1, j \neq n}^{K} (P |D_{n,j}|^2 + M_{n,j} \sigma_w^2) \tag{4.7}$$

The total sum-rate capacity of the channel hence is

$$C = \sum_{n=1}^{N} \log_2(1 + \frac{\sigma_{u_n}^2}{\sigma_{s_n}^2}) \, [\text{bit/s/Hz}] \tag{4.8}$$

4.2 Diagonal Channel Matrix

Let us consider a channel matrix of the type

$$H = \begin{pmatrix} h_{1,1} & 0 & \cdots & 0 \\ 0 & h_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_{N,K} \end{pmatrix} \tag{4.9}$$

Given the definition of the received signal as expressed by (2.1), we can see that with this matrix the received signal at the n^{th} receiver is

$$y_n = P_n^{1/2} h_{n,n} x_n + w_n. \tag{4.10}$$

We can see that (4.10) differs from the one presented in previous section from the fact that, thanks to the diagonal form of the channel matrix, the received signal is only composed by useful signal and additive Gaussian noise, without interference.
If we adopt an MMSE receiver and the same notation introduced in previous section we can state that the signal-to-noise ratio at the receiver for the n^{th} user will be

$$\Gamma_n = \frac{P|h_{n,n}|^2}{\sum_{j \neq n}^{K} |M_{n,j}|^2 \sigma_w^2} \tag{4.11}$$

Hence the capacity for a single user is

$$C_n = \log_2(1 + \Gamma_n) \, [\text{bit/s/Hz}] \tag{4.12}$$

as stated from Shannon-Hartley theorem, and the total sum-rate capacity of the channel hence results

$$C = \sum_{n=1}^{N} \log_2(1 + \Gamma_n) \, [\text{bit/s/Hz}] \tag{4.13}$$

4.3 Orthonormal Channel Matrix

An orthonormal matrix is defined as a square matrix whose rows and columns are orthogonal unit vectors. Let us consider a particular

orthogonal channel matrix of the type

$$H = \begin{pmatrix} 0 & 0 & \cdots & h_{1,K} \\ h_{2,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & h_{N,2} & \cdots & 0 \end{pmatrix} \tag{4.14}$$

with rows/columns composed by all zero elements except for one. With this configuration the received signal at BS n is

$$y_n = P_k^{1/2} h_{n,k} x_k + w_n. \tag{4.15}$$

We can notice that (4.15) has the same form of (4.10), where the only difference is the channel coefficient associated to the input signal.
By adopting an MMSE receiver, the matrix resulting from the product of channel matrix **H** and MMSE matrix M assumes a diagonal form. The sum-rate capacity hence is the one obtained for the diagonal matrix case, i.e. (4.13).

4.4 Sparse Channel Matrix

Let us consider a channel matrix of the type

$$H = \begin{pmatrix} h_{1,1} & 0 & \cdots & h_{1,K} \\ 0 & h_{2,2} & \cdots & h_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ h_{N,1} & h_{N,2} & \cdots & 0 \end{pmatrix} \tag{4.16}$$

where some row entries assume zero value. The received signal at the n^{th} BS is

$$y_n = \sum_{k=1}^{K} P_k^{1/2} h_{n,k} x_k + w. \tag{4.17}$$

We notice that (4.17) is similar to (4.4), except for the fact that some entries $h_{n,k}$ of the channel matrix are zero. Hence, if we adopt an MMSE receiver and consider as useful signal $P_n^{1/2} d_{n,n} x_n$ we can follow the same considerations made for the full channel matrix and obtain that the total sum-rate capacity for the system is given by (4.8) Notice that the value of σ_s^2 also suffers the sparsification of the channel matrix, i.e. the sum of the variances will be composed by the only elements with $d_{n,k} \neq 0$.

4.5 Capacity with GMP and RGMP Receivers

We will here analyse the capacity obtained with message passing receivers. This type of computation is independent from the type of channel matrix, since the message passing task is to retrieve the transmitted signal by means of iterative operations, and hence it returns a value which is assumed to be the MMSE of trasnmitted signal.
Since along with the estimate of the transmitted signal components we obtain measures of their variances, we can use this values to compute the capacity of a channel employing a GMP/RGMP receiver. In fact, we can take as a measure of SNR for the n^{th} sub-channel the ratio between the estimated signal power and the variance associated to that estimate. Notice that the estimated signal power will be given by the power allocated to that user, since GMP and RGMP return the exact value of the transmitted signal.
The capacity of the channel hence is

$$C = \sum_{n=1}^{N} \log_2(1 + \frac{P_n}{\hat{\sigma}_{x_n}}) \, [\text{bit/s/Hz}]. \tag{4.18}$$

Statistical measures of SNR can also be taken . In fact, if we consider a certain number of decoding processes for a transmitted signal which will be affected by noise that changes with iterations, the SNR at the receiver can be written as

$$\Gamma_{stat} = \frac{E[|\hat{\mathbf{x}}|^2]}{E[|\hat{\mathbf{x}} - \mathbf{x}|^2]}, \tag{4.19}$$

which results in a sum-rate capacity

$$C = \log_2(1 + \Gamma_{stat}) \, [\text{bis/s/Hz}]. \tag{4.20}$$

This result holds for both GMP/RGMP and MMSE receivers.

4.6 Achievable Sum-Rate Results

In this section we present the achievable sum-rate results obtained by simulations for channels characterized by the four different matrices and receivers employing MMSE, GMP and RGMP. The achievable sum-rate has been computed by mean of the statistical SNR presented in previous section as a function of SNR. The scenario over which the simulations have been run is characterized by 4 users and 4 BS and each user is associated to a BS. Furthermore we assume

that all users have been allocated the same constant transmission power, i.e. $P_k = P \ \forall k \in \{1, .., K\}$. Diagonal and orthonormal channel matrix realizations have i.i.d. entries, distributed as zero mean and unit variance Gaussian random variables, whereas full and sparse channel matrices entries are i.i.d. Gaussian random variables with zero mean and sum of the variances per row equal to N_0. Message passing algorithms have been run over different number of iterations, depending on whether the receiver employs GMP or RGMP. In particular we choose to run GMP over 40 iterations, whereas RGMP has been run over 15 iterations. Results are presented in figures 4.1, 4.2 and 4.3.
Before discussing the results we recall here that both GMP and RGMP aim at reproducing the same results obtained with MMSE, i.e. the decoding process is different but the result shall be the same. Therefore we expect to obtain the same achievable sum-rate with the different receivers. We can notice that our hypothesis is verified. In fact, looking at the evolution of the capacity vs. SNR over figures, we can see that they both have the same performance.
A second consideration shall be made about the comparison between capacity with diagonal and orthogonal matrices. Recalling what presented and discussed in section 5.3 for the orthogonal matrix, the MMSE receiver aims at diagonalizing the overall matrix, i.e. the product of channel matrix and MMSE matrix. Therefore what we see at the receiver is a channel characterized by a diagonal matrix and hence we expect that the achievable sum-rate of such a channel matches that of a diagonal channel matrix. We can see that this is verified in all figures, recalling again the fact that the presented message passing algorithms emulate MMSE detection.
The last consideration is about the results obtained with full and sparse channel matrices. Achievable sum-rate in this cases follows a different behaviour respect to diagonal and orthogonal channel matrices, in particular it assumes lower values. This is due to the fact that each user suffers from interference from other users.

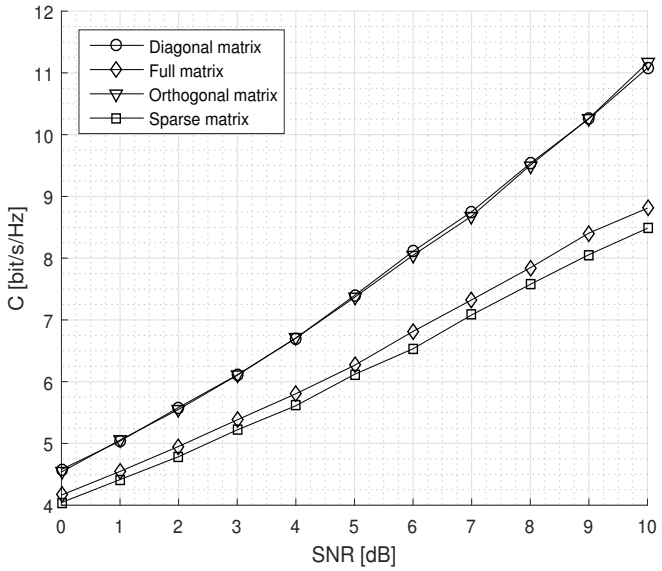

Figure 4.1: Achievable sum-rate results with MMSE receiver over the 4 types of channel matrices.

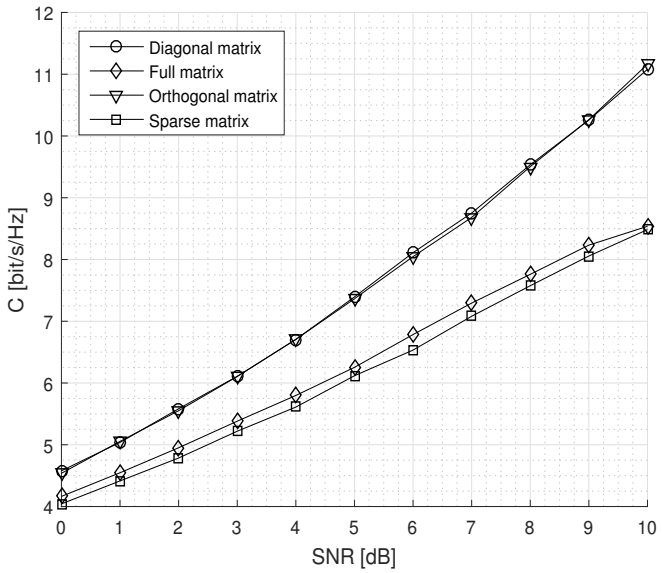

Figure 4.2: Achievable sum-rate results with GMP receiver over the 4 types of channel matrices.

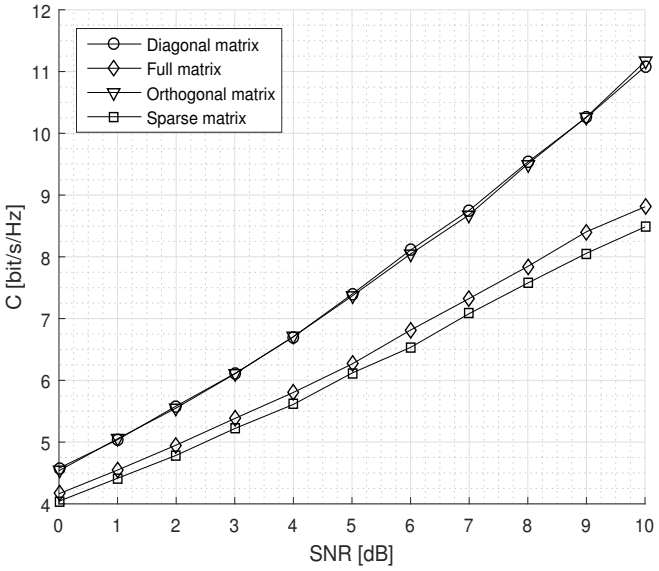

Figure 4.3: Achievable sum-rate results with RGMP receiver over the 4 types of channel matrices.

Chapter 5

Channel Sparsification Modelling and Analysis

We stated in section 3.4.3 that both MMSE and message passing decoding complexity depend on the number of users and BSs of the network: in the previously analysed case, where only 4 users and BSs were present, decoding time was relatively low, but now that we are introducing decoding of the signals coming from $K = 64$ users received from $N = 128$ BSs a complexity of $\mathbf{O}(N^3)$ or $\mathbf{O}(NK^2)$ strongly impact on the advantages that C-RANs should have.
In next sections we hence introduce methods to sparsify the channel matrix, i.e., reduce the number of coefficients $\neq 0$, in order to reduce the complexity of the message passing decoders. The number of iterations of the message passing algorithm to be used obviously impacts system's performance in terms of execution time. Hence, recalling the results obtained in section 4.3.4, we prefer RGMP over GMP.
Our aim is to reduce the computation complexity of message passing algorithms in order to make them more advantageous over MMSE signals estimation obtainde by channel matrix inversion. We hence extend the number of users and BSs of the analysed network and investigate techniques which can lead to simplifications of the model and hence to better exploitation of message passing decoding.

5.1 Sparsification Methods

As stated before the computational complexity of message passing algorithms depend on the number of elements present in the network we are considering. In particular, since RGMP receives as input the channel matrix and the received signal and operates only on couples

(n, k) with $H(n, k) \neq 0$, our aim is to introduce zeros in the channel matrix making reasonable assumptions and modifying the algorithm in order to support the correct decoding of received signals with the lowest possible amount of time. Next sections will present the selected methods and results obtained for channel sparsification.

5.1.1 Power-Based Sparsification

Let us consider the signals received from a single BS of the cellular network. The main idea behind this approach is that users located away from the cell reach BS's antennas with lower average power than users located nearby the BS. Therefore we pre-process the estimated channel matrix and put a threshold on the power of its coefficients. In particular, denoting with P_{min} the minimum square module of the coefficients of the channel matrix that can seriously interfere with a user in the selected cell, we set all channel matrix coefficients below P_{min} to zero, whereas all coefficients above P_{min} remain unaffected. Obviously we can not discard the effects of the users received with low power, as they however affect the signal received by the considered BS. Denoting with $\mathbf{H}$ the full channel matrix and with $\hat{\mathbf{H}}$ the sparsified channel matrix we can subtract the second from the first, and obtain the matrix $\tilde{\mathbf{H}} = \mathbf{H} - \hat{\mathbf{H}}$, which is composed only by channel coefficients below the selected threshold. We can hence use (3.14) (substituting $\dot{\mathbf{H}}$ with $\tilde{\mathbf{H}}$) to modify noise power in order to take into account the interference played by neglected entries in $\hat{\mathbf{H}}$. In formulas, consider channel matrix entry $H_{n,k}$. Then

$$\hat{H}_{n,k} = \begin{cases} H_{n,k} & \text{if } |H_{n,k}|^2 \geq P_{\min} \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

This approach is similar to the one presented in [3], but with the main difference that thresholding is not applied on distance but on channel coefficients squared module.
We can see that a fundamental role is played by P_{min}. In fact, if we select a too small value for this parameter channel sparsification do not affect sufficiently system's performance in terms of execution time of decoding algorithms, whereas if we set a too high threshold value the result is that we loose useful informations and hence decoding algorithms will introduce too much errors.

5.1.2 Orthogonal Users-Based Sparsification

Let us consider the scenario of Fig. 2.4, where N_c cells and a total number of N_u users are present. We assume that signals coming from the same BS are on adjacent entries of the received signal vector $\mathbf{y}$, thus adjacent rows of $\mathbf{H}$ belong to the same BS. We can hence divide the overall channel matrix $\mathbf{H}$ into N_c sub-channel matrices $\mathbf{H}_c \in \mathbb{C}^{N_c \times N_u}$ with $c \in \{1, 2, .., N_c\}$ representative of a single cell. We can here highlight the effects of every user to a single antenna of a BS and consider the fact that two or more users can perform orthogonal transmission. We can hence introduce the second approach for channel sparsification. Recalling the fact that each BS is responsible for messages sent from users located into its cell, if a user located outside such a cell is performing an orthogonal transmission with respect to the BS' users we set to zero the sub-channel matrix coefficients of the outside-located user. After performing this operation for every sub-channel matrix we can put them together and retrieve the overall channel matrix.
In formulas, let us consider the channel

$$\mathbf{h}_{k_1} = [H(n_1, k_1), H(n_2, k_1), ..., H(n_{N_a}, k_1)] \tag{5.2}$$

from user k_1 to all RRHs belonging to a certain BS with indices in the set $\mathcal{A} = \{n_1, n_2, ..., n_{N_a}\}$. Considering a second user k_2, the channel to the same RRH set is

$$\mathbf{h}_{k_2} = [H(n_1, k_2), H(n_2, k_2), ..., H(n_{N_a}, k_2)]. \tag{5.3}$$

The orthogonality among channels to the same BS is established by the internal product of the channels and we consider that two channel are semi-orthogonal if the product is below a threshold T_{prod}, i.e.,

$$\mathbf{h}_{k_1}\mathbf{h}_{k_2}^T < T_{\text{prod}}. \tag{5.4}$$

As done for power-based channel sparsification, we select a threshold which leads to approximatively the same relative error of RGMP with full channel matrix while reducing the cost in terms of time of the decoding process. Notice that, in this case, we do not take into account the fact that users are however present and interfere with users for which the BS is responsible as done for the previous case, since we are analysing orthogonality, condition over which users do not interact in any way.

5.1.3 Antennas-Selection-Based Sparsification

Let us examine again the presented scenario: the entire area has been divided into N_c cells, each one containing N_u users and a BS

with N_a omnidirectional antennas located in the same place. We consider another type of channel sparsification assuming that the N_a signals coming from the N_a antennas are over-representative of signals transmitted by N_u users, i.e. the ones located in the considered cell. We can therefore sparsify the channel matrix by reducing the number of its row selecting the 4 antennas of a BS which are more representative of the users located inside the cell. We can hence consider the sub-channel matrices introduced in previous section choose the most representative antennas. After completing this task over each one of the sub-channel matrices we can reconstruct the overall channel matrix and feed it to RGMP algorithm.
Antennas selection hence results in an optimization problem, where we must choose the parameter we want to optimize. We aim at optimizing the channel capacity in order to maintain high the number of bit/s/Hz we can transmit over the resulting channel.
Let us assume that all antennas in our cellular network scenario are receiving signals from all single antenna users. If we denote N_t as the number of transmitting antennas (equal to the number of users) and with N_r the total number of receive antennas, we can write the overall system's capacity as

$$C = \log_2 \left[\det \left(I_{N_r} + \frac{\bar{\Gamma}}{N_t} H H^H \right) \right], \tag{5.5}$$

with I_{N_r} being the $N_r \times N_r$ identity matrix and with $\bar{\Gamma}$ the mean SNR per receiving branch. In order to select the L_r receiving antennas that maximize capacity we should compute

$$C_{sel} = \max_{\mathcal{S}} \left\{ \log_2 \left[\det \left(I_{L_r} + \frac{\bar{\Gamma}}{N_t} \tilde{H} \tilde{H}^H \right) \right] \right\} \tag{5.6}$$

with $\mathcal{S}$ being the set of all possible channel matrices obtained by deleting $N_r - L_r$ rows from H and whose cardinality is $\binom{N_r}{L_r}$. The selection of the optimal set of antennas requires the computation of a number of determinants which is equal to the cardinality of $\mathcal{S}$.
Based on results obtained in [6], we decided to implement two different receive antenna selection algorithms, namely correlation based method (CBM) and mutual information based method (MIBM).
Notice that normalization of the channel matrix is here performed over rows, since we are interested in antennas characteristics and not users ones.

Correlation Based Method

Let us consider the previously depicted scenario, where we choose $N_r - L_r$ representative antennas out of 8 from each one of the cells composing our cellular network, where N_r is the total number of antennas per BS, while L_r is the number of antennas we are not going to consider. With correlation based method (CBM) we consider couples of antennas, and hence channel matrix rows $h_{n\in\mathcal{A}(c)} = [H(n,1), H(n,2), ..., H(n,K)]$, where $\mathcal{A}(c)$ denotes the set of indexes of antennas belonging to cell c, and measure their correlation as

$$c_{n_1,n_2} = \mathbb{E}[|h_{n_1} h_{n_2}^T|^2] \tag{5.7}$$

for each couple $\{n_1, n_2\} \in \mathcal{A}(c)$, where $E[]$ denotes the expected value. CBM steps are shown in Algorithm 2.

Data: H, N_{ac}
Result: H
for *c=1 to* N_c **do**
  for *n=1 to* $N_a - N_{ac}$ **do**
    1. compute correlation for each couple $\{n_1, n_2\} \in \mathcal{A}(c)$ with (5.7),
    2. choose the couple with highest correlation,
    3. set to zero the row of H corresponding to the antenna of the considered couple with lower value of $\sum_{k=1}^{K} |H(n,k)|^2$
  end
end

Algorithm 3: Correlation Based Method (CBM)

Since we are considering a distributed scenario, where antennas are located in groups of eight in different locations, we will not consider the correlation of a row with all the others, but only with the rows corresponfing to antennas located in the same cell.
We therefore proceed as follows. Recalling the definition of sub-channel matrices given in section 5.1.2, we divide the overall channel matrix into 16 sub-channel matrices and, for each row, we compute the correlation with all the other rows. When we find the maximum over this values we delete one of the two rows resulting with highest correlation, choosing the one with smaller power. We iterate this operation until we are left with 4 rows out of 8. The resulting sub-channel matrices are then grouped together to form the channel matrix that will be feed to RGMP decoder.

Mutual Information Based Method

Let us consider the received signals y_m and y_l from antennas respectively m and l belonging to a certain cell, and consider their mutual information value $I(y_l; y_m)$. If mutual information value is small we can state that the two received signals carry almost completely different information, whereas if mutual information value is high we can state that the two signal are carrying similar information. In mutual information based method (MIBM) we choose one from this two signals and set to zero the sub-matrix row of the other one without loss of information. This approach is very similar to the one discussed before, where the only difference is that we are interested in the highest mutual information value instead of correlation.
In formulas, as shown in [6], mutual information is upper bounded by

$$I(y_l; y_m) \leq \min \left\{ \log_2 \left(1 + ||\mathbf{h}_l||^2 \frac{\bar{\Gamma}}{N_t} \right), \log_2 \left(1 + ||\mathbf{h}_m||^2 \frac{\bar{\Gamma}}{N_t} \right) \right\}, \tag{5.8}$$

with $\mathbf{h}_l$ and $\mathbf{h}_m$ being the n^{th} and l^{th} rows of the channel matrix and $\bar{\Gamma}$ being the average SNR at the receiver.
Mutual information between signal components can be written as

$$I(x_l; x_m) = \log_2 \frac{\|\mathbf{h}_l\|^2 \|\mathbf{h}_m\|^2}{\|\mathbf{h}_l\|^2 \|\mathbf{h}_m\|^2 + |\langle h_l, h_m \rangle|^2}, \tag{5.9}$$

which can be normalized as

$$I_0(x_l; x_m) = \frac{I(x_l; x_m)}{\min\{|\log_2(\|\mathbf{h}_l\|^2)|, |\log_2(\|\mathbf{h}_m\|^2)|\}}. \tag{5.10}$$

MIBM steps are showed in Algorithm 2, with the only difference that correlation (5.7) must be substituted with normalized mutual information (5.10) in step 1.

5.2 Simulation Results

5.2.1 Relative Error and Sparsification Level: Power-Based

Results regarding relative error are presented in figure 5.1, where each user transmits a signal generated as complex Gaussian random variable with zero mean and unitary variance. We can see that the relative error has been plotted vs. the number of RGMP iterations for pure RGMP (i.e. without sparsification) and RGMP with $P_{min} = 0.001$, $P_{min} = 0.005$, $P_{min} = 0.01$ and $P_{min} = 0.1$, for both 5

and 100 dB SNR.
We can notice that, as we expected, a higher value of P_{min} negatively impacts the system relative error, which becomes larger as P_{min} increases.
We can see that with $P_{min} = 0.001$ we reach approximatively the same of relative error obtained with the full channel matrix, in particular when $SNR = 100$ dB.
Even if we consider $P_{min} = 0.005$ we can see from figure 5.1 that the relative error differs from the full channel matrix one of $\approx 5 \cdot 10^{-2}$ in 100 dB SNR case, and a smaller difference is present when $SNR = 5$ dB, value that could be acceptable if we really want to reduce algorithm's execution time. In fact we can see that the minimum relative error is obtained with a smaller number of iterations in 100 dB case and that channel matrix coefficients different from zero are reduced to 1345, approximatively 8 times less.
However our aim was to simplify the algorithm by sparsification of the channel matrix, and different thresholds necessarily mean different amounts of sparsification. Table 5.1 reports the number of channel matrix coefficients different from zero after the sparsification process with different threshold values.
From the table we can notice that, with $P_{min} = 0.001$, the number of channel matrix coefficients different from zero has been reduced from 8192 to 4121, meaning that each RGMP iteration will take a smaller amount of time to be concluded.

	Full H	$P_{min} = 0.001$	$P_{min} = 0.005$	$P_{min} = 0.01$	$P_{min} = 0.1$
number of $H(n,k) \neq 0$	8192	4121	1345	755	154

Table 5.1: Sparsification levels for power-based method.

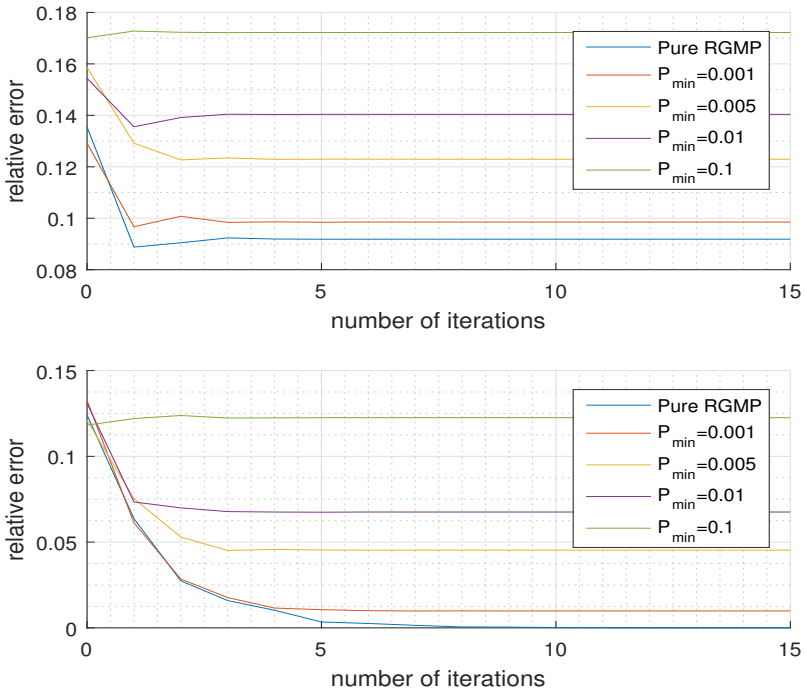

Figure 5.1: Relative error vs. number of RGMP iterations for different P_{min}: upper Fig. with 5 dB SNR, lower with 100 dB SNR.

5.2.2 Achievable Sum-Rate: Power-Based

In order to show thresholding effects on system's performance we computed the achievable sum-rate of the channel when the above mentioned values of P_{min} are applied to sparsify its matrix. The obtained results can be seen in figure 5.2. Achievable sum-rate is plotted versus SNR, where users transmission power is maintained constant. We can see that, as we expected, the achievable sum-rate of the channel is quite the same for RGMP with full channel matrix and with $P_{min} = 0.001$ (at least for small SNR values), whereas all the other cases differ from pure RGMP of dozens of bit/s/Hz, expecially for high SNR values. We can hence state that, even if relative error is not so distant from pure RGMP's one, the transmission capacity is strongly affected by the selected threshold value.

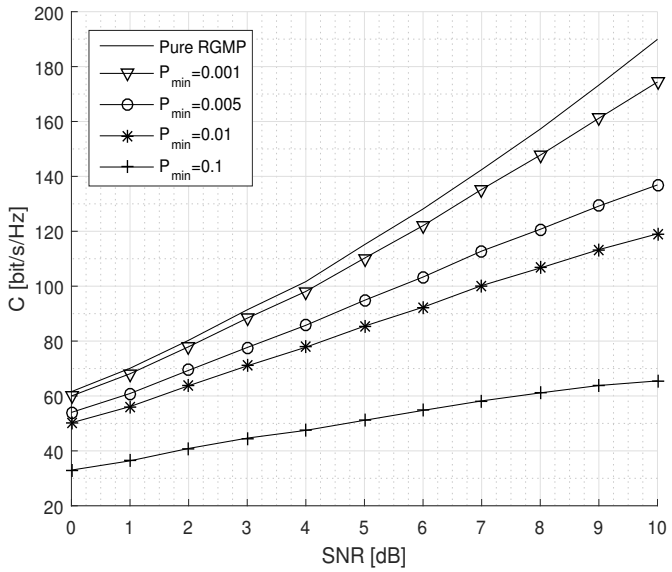

Figure 5.2: Achievable sum-rate vs.SNR for different P_{min} values

5.2.3 Relative Error and Sparsification Level: Orthogonal Users

Relative error is a good measure for testing system's performance in order to choose a suitable parameter value. Figure 5.3 reports relative error vs. SNR for pure RGMP (i.e. without orthogonal users analysis) and for RGMP with $T_{prod} = 0.0025$, $T_{prod} = 0.005$, $T_{prod} = 0.0075$ and $T_{prod} = 0.01$. We to report results obtained with

both 5 and 100dB SNR and with users transmission power kept constant in order to highlight effects of noise on relative error with this sparsification method. We notice that the noisy case (i.e. 5 dB SNR)is more performing in terms of distance of relative errors for sparsified matrices from pure RGMP's one, whereas the 100 dB SNR case presents a relative error that grows with T_{prod}. Since typical scenarios are affected by noise levels which more resembles 5dB SNR case, we can state that this sparsification method is quite effective for maintaining an error level similar to the full channel matrix case. Furthermore, by taking a closer look to the line plots for the 5dB SNR case (figure 5.4), we can see that the distance from sparsification methods relative error from pure RGMP's one is lower than 10^{-5} . In particular, by setting $T_{prod} = 0.0075$, we get the lowest relative error. Notice also that all sparsification methods, except for $T_{prod} = 0.005$, exhibit a relative error lower than pure RGMP after 3 MP iterations.

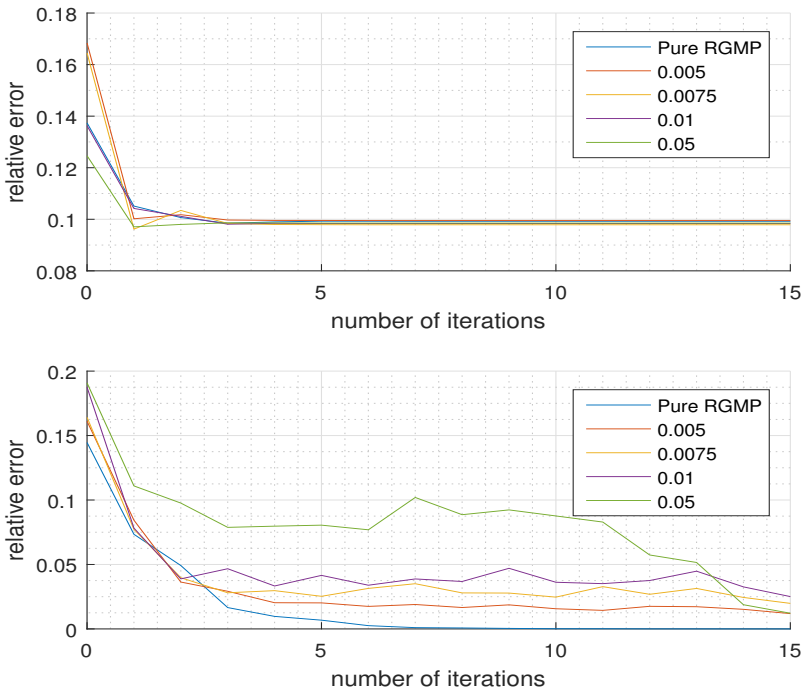

Figure 5.3: Relative error vs. number of RGMP iterations for different T_{prod}: upper Fig. 5 dB SNR, lower 100 dB SNR case.

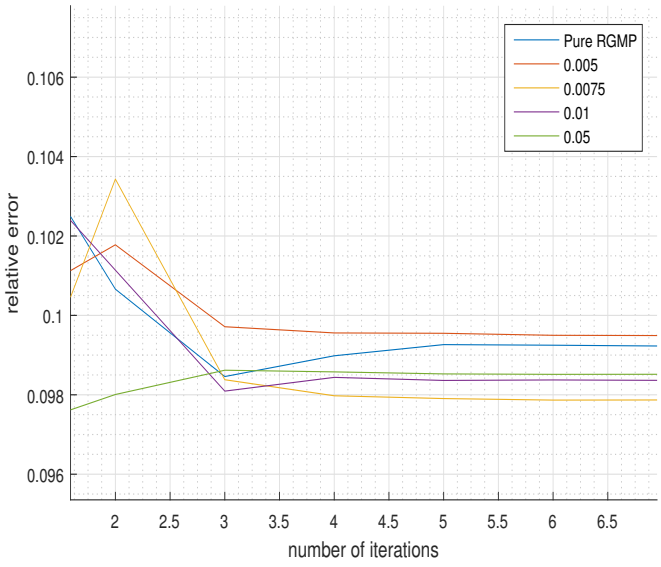

Figure 5.4: Relative error vs. number of RGMP iterations for different T_{prod}: $5dB$ SNR.

We focus now on channel matrix sparsification. Table 5.2 reports the number of coefficients which are different from 0 after the sparsification with different T_{prod}.
We notice that, with respect to results obtained with power-based sparsification method, the sparsification level is here lower, i.e. the number of channel coefficients that can be set to zero without too much affecting system's error is lower. Hence decoding time for algorithms is higher in this case. However we must consider the fact that with orthogonal-users based sparsification method we can use a lower number of message passing iterations to obtain the same relative error given by RGMP. Hence the computational complexity for the entire decoding process results lower respect to pure RGMP's one. Furthermore we must say that orthogonality is a more stringent condition to be verified,i.e. threshold values for orthogonality, in order to be significative, shall be set to low values. We can augment sparsification level with higher T_{prod} values, but at the cost of higher relative errors for high SNR values and with losses in capacity as will be discussed in following section.

5.2.4 Achievable Sum-Rate: Orthogonal Users

We want to show the effects of the different thresholding values in terms of achievable sum-rate, comparing the plots of achievable

	Full H	$T_{prod} =$ 0.005	$T_{prod} =$ 0.0075	$T_{prod} =$ 0.01	$T_{prod} =$ 0.05
number of $H(n,k) \neq 0$	8192	6728	5560	4288	648

Table 5.2: Sparsification levels for orthogonal users-based method

sum-rate vs. SNR for pure RGMP and RGMP with T_{prod} values before discussed. Notice that transmission power of users has been kept constant and hence the varying part of SNR is noise level. The number of RGMP has been set to 4, given the considerations made on relative error and relative obtained results.
achievable sum-rate plots are shown in figure 5.5. As we could expect, achievable sum-rate at low SNR is similar for all RGMP realizations, whereas they start to diverge while SNR increases. This is in accordance to relative error results discussed above. We can hence state that the best threshold value can be decided if we can estimate SNR at the receiver, and hence decide to use the highest T_{prod}, which leads to a greater sparsification, or a lower one, which leads to a smaller sparsification but maintains the characteristics of full-matrix RGMP. Notice that, at low SNRs, achievable sum-rate loss with high T_{prod} is negligible, encouraging us to consider as threshold value a high T_{prod}, which maintains system's achievable sum-rate and considerably sparsifies channel matrix.

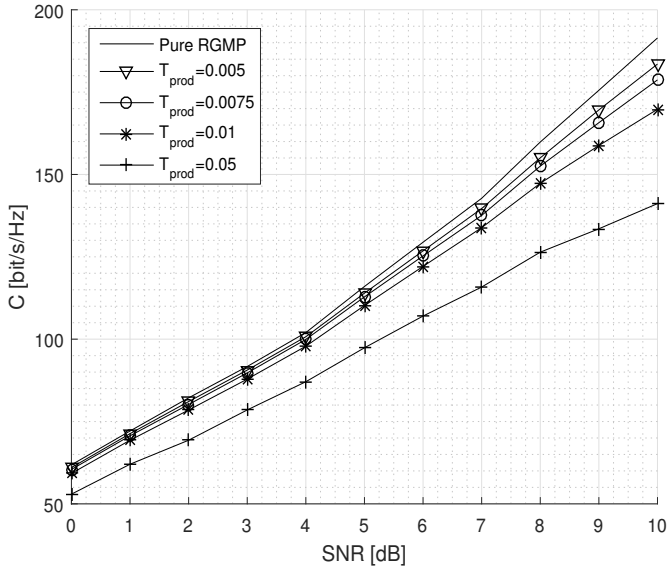

Figure 5.5: Achievable sum-rate vs. SNR for different T_{prod} values.

5.2.5 Relative Error and Sparsification Level: CBM

We are now interested in selecting the number of receive antennas to be used in order to decode the transmitted signal with the lowest possible relative error. In order to chose such a parameter we run simulations for pure RGMP (i.e., without antenna selection), and for RGMP with $N_r - L_r = 7$,$N_r - L_r = 5$,$N_r - L_r = 3$ and $N_r - L_r = 1$ receive antennas. We report results at two different SNR levels, respectively 5 dB and 100 db in order to highlight effects of noise over message passing algorithms. Notice that users transmitting power has been kept constant, i.e. different SNR means different noise level. Results are shown in figure 5.6.
We can notice that the relative error grows as the number of receive antennas decreases after ≈ 3 RGMP iterations for low SNR values (except for 1 receive antenna, wich leads to approximatively the same relative error obtained with 5), whereas for high SNR values we can notice that the single receive antenna sparsification method leads to approximatively the same relative error obtained with pure RGMP and with a lower number of message passing iterations. We can also notice that for high SNR values RGMP run over matrices with different number of receive antennas can reach approximatively the same relative error, with the only difference in the number of required message passing iterations.

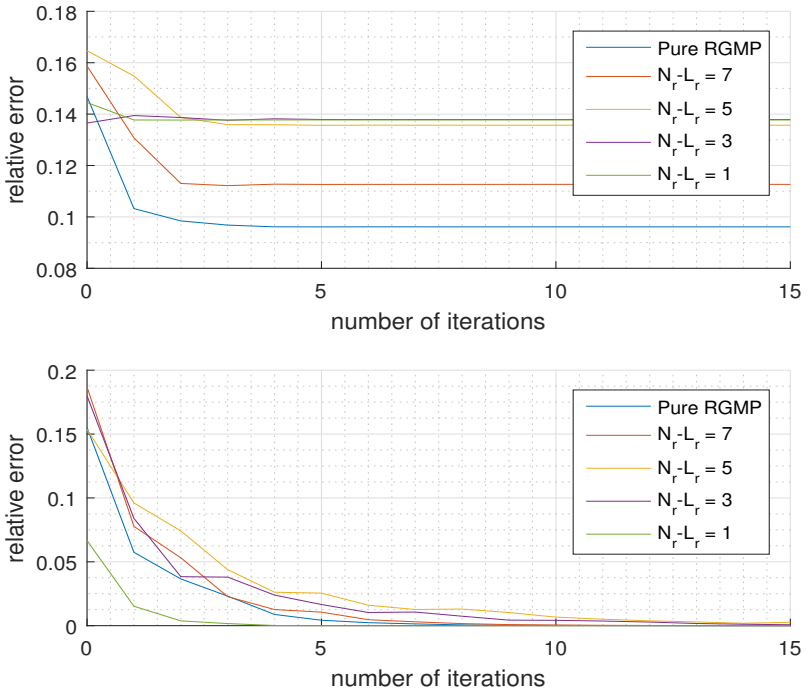

Figure 5.6: Relative error vs. number of iterations for CBM with different number of receive antennas: upper Fig. with 5 dB SNR, lower with 100 dB SNR.

We now discuss the sparsification level. Table 5.3 reports the number of coefficients of the channel matrix which are different from zero after that CBM antenna selection has been applied. Since we eliminate L_r rows from each sub channel matrix, table's results are easily obtainable.
We can hence state that if we know the SNR at the receiver we can make a better choice. In fact, as previously discussed, in a high SNR regime not only single receive antenna implementation leads to a minimum relative error with a lower number of message passing iterations, it also leads to the biggest sparsification of the channel matrix.

	Full H	$L_r = 1$	$L_r = 3$	$L_r = 5$	$L_r = 7$
number of $H(n,k) \neq 0$	8192	7168	5120	3072	1024

Table 5.3: Sparsification levels for CBM antenna selection.

5.2.6 Achievable Sum-Rate: CBM

We now report simulation results obtained for achievable sum-rate when antenna selection is applied. We tested systems where the number of receiving antennas is 8, 7, 5, 3 and 1 as done for relative error and sparsification level. Results are shown in figure 5.7. We can see that the achievable sum-rate suffers the decreasing in receiving antennas number. In fact we notice that decreasing the number of receive antennas decreases the achievable sum-rate. Hence we can state that, equivalently from what stated in relative error and channel sparsification level, the best approach is here to keep all receiving antennas on. We can also state that in real system a threshold exists, i.e. in order to take a decision on which approach to follow we must consider both sparsification level and channel capacity. The choice depends on the performance measure we are interested in.

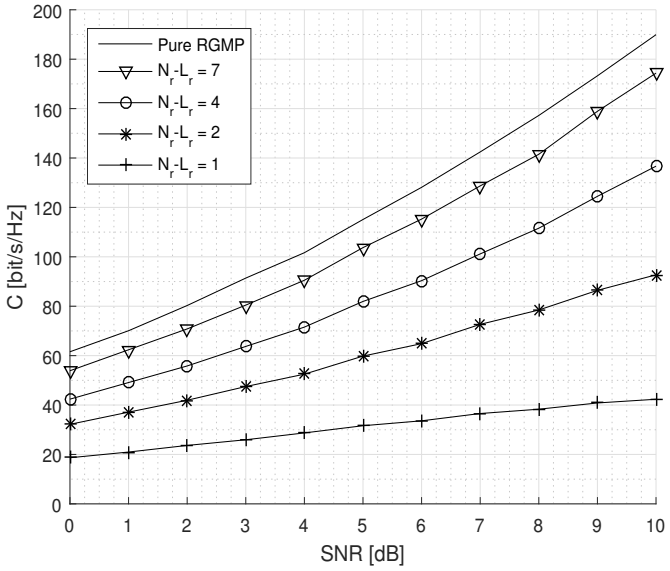

Figure 5.7: Capacity with antenna selection sparsification: CBM.

5.2.7 Relative Error and Sparsification Level: MIBM

We now discuss the relative error obtained with MIBM sparsification with different number of receive antennas and compare them to relative error obtained with RGMP with a full channel matrix. The relative error versus the number of RGMP iteration for both 5 dB and 100 dB SNR values are shown in Fig. 5.8, where users transmitting power has been kept constant, while noise power varied.
We notice that relative error, for low SNR values, grows as the number of receive antennas decrease after a certain number of message passing iterations, as for CBM antenna selection. However with 7 receive antennas we obtain the minimum among relative error values with a lower number of message passing iterations than pure RGMP's. Instead, at high SNRs all implementations reach the same relative error after 15 message passing iterations. Furthermore as for CBM, for 100 dB SNR the single receive antenna implementation reaches the minimum relative error (equal to pure RGMP's one) with the lower number of message passing iterations, even smaller than pure RGMP's.

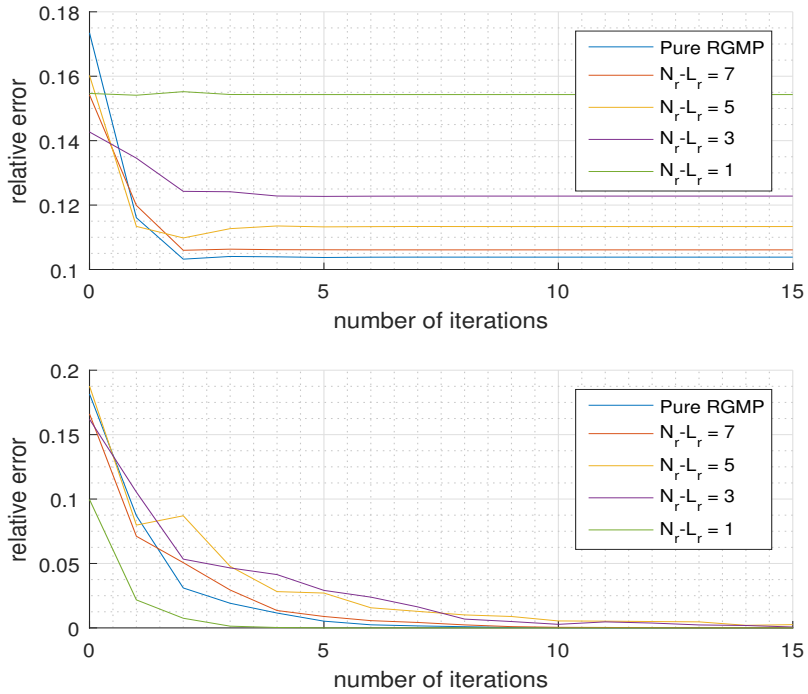

Figure 5.8: Relative error vs. number of iterations for MIBM with different number of receive antennas: upper Fig. with 5 dB SNR, lower with 100 dB SNR.

Sparsification levels are the same shown in 5.3. Consideration and results are the same made for CBM antenna selection. Even in this case the knowledge of the SNR value at the receiver influences our parameter choice. In fact, as for CBM, if we are transmitting in a high SNR receiver the single antenna sparsification leads to the lowest relative error with the lower number of RGMP iterations.

5.2.8 Achievable Sum-Rate: MIBM

We now consider MIBM achievable sum-rate. Fig. 5.9 shows results obtained wit 8, 7, 5, 3 and 1 receive antenna out of 8. Although difference in relative errors was low, difference in achievable sum-rate is significant when varying the number of selected receive antennas. In particular we can state that capacity decreases with the number of selected receive antennas. Hence, although from a relative error and sparsification level prospective the single antenna MIBM is the best choice, from a channel capacity point of view is the worse. Therefore we have a trade-off between complexity and achievable sum-rate.

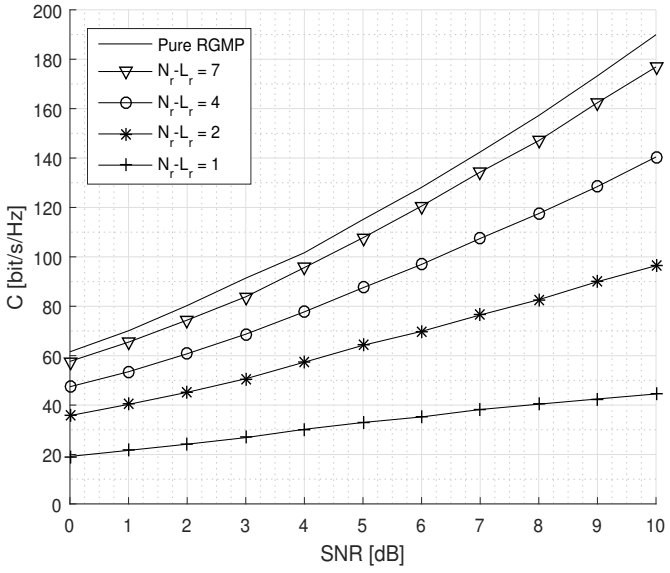

Figure 5.9: achievable sum-rate with antenna selection sparsification: MIBM.

5.3 Pre-Coding Sparsification

We have seen in previous sections sparsification methods that try to analyse the channel matrix at the receiver and, before feeding it to the receiver, set some coefficients to zero according to a certain rule. The problem of this approach is that we however need the entire channel matrix at the BBU pool, besides signal coming from the 8 antennas of each BS. We now porpose to reduce the amount of information flowing from RRHs to the BBU pool by performing receive beamforming of each BS before forwarding signals to the central BBU Pool. Recalling the received signal (2.1), if the considered BS has N_a antennas, we want to multiply the received signal by a matrix $\mathbf{B}$ of the type $\mathbb{C}^{N_r \times N}$, in order to reduce the number of signals that will be passed to the central BBU pool from N_a to N_r. Matrix $\mathbf{B}$ can assume different forms and consequently different meanings. The first implemented method tries to reduce the number of signal coming from BS's antennas by multiplication by a matrix whose rows are equal to the number of users located inside the considered cell. We hence recall the definition of sub-channel matrix given in previous sections, and consider the sub-channel matrix belonging to the n^{th} cell and denote it as $\mathbf{H}_n$. If the number of users located inside such a cell is N_u received signal at the n^{th} cell can be written as

$$\mathbf{y}_n = P\mathbf{H}_n(n_1, ...n_{N_a}; u_1, ..., u_{N_u})\mathbf{x} + \mathbf{H}_n(n_1, ..., n_{N_a}; u_{N_u+1}, ..., u_K)P\mathbf{i} + \mathbf{w}, \tag{5.11}$$

where $\mathbf{x}$ is the vector containing signals coming from users located inside the considered cell, $\mathbf{i}$ is the vector containing signals coming from users located outside the considered cell, $\mathbf{w}$ is additive white Gaussian noise with zero mean and variance N_0 at the N receivers and $\mathbf{H}_n(n_1, ...n_{N_a}; u_1, ...u_{N_u})$ is the sub channel matrix of cell n with $\{n_1, ..., n_{N_a}\} \in \mathcal{A}(n)$ being the set of indexes of antennas located in n and $\{u_1, ...u_{N_u}\} \in \mathcal{U}(c)$ being the set of indexes of users of interest for cell n (concept that will be later discussed). The same form is assumed if we do not consider interferers all users located outside the cell, but only some less powerful ones. The only difference is the dimensions of matrix H_n that will multiply useful signals and interferers.
Multiplying the received signal by matrix $\mathbf{B}$ provides

$$\mathbf{B}\mathbf{y}_n = \mathbf{B}\mathbf{H}_n(n_1, ...n_{N_a}; u_1, ..., u_{N_u})P\mathbf{x} + \mathbf{B}\mathbf{H}_n(n_1, ..., n_{N_a}; u_{N_u+1}, ..., u_K)P\mathbf{i} + \mathbf{B}\mathbf{w}. \tag{5.12}$$

We can see that noise vector entries are correlated and that the message passing algorithms must be modified. Since noise power remain the same in all branches we can modify the power level passed to the message passing algorithm as follows

$$N_0(n) = N_0 \sum_{k=1}^{N_u} |B_{n;k}|^2 \tag{5.13}$$

Therefore we pass to the message passing algorithms a vector of noise power values.
In what follows we are first going to present three different approaches in order to choose a set of users over which matrix B will be built. For each one of these three methods we choose both the set user and matrix operation that will lead to the required form of $\mathbf{B}$. Then we are going to test all presented approaches in terms of relative error and channel capacity, in order to derive system's performance.

5.3.1 Known Users Pre-Coding

Suppose that each BS is equipped with N_a receive antennas. Then the received signal at the n^{th} BS that has to be passed to the central pool in order to decode messages sent by all users can be represented as a vector of length N_a. As we discussed in previous section, this amount of information can be prohibitive to be sent over links from BS to BBU. Let us assume we now users location and, in particular, in which cell a user is located. Hence we know exactly which signals represent the useful information and which ones are interferers in (5.11) and hence we also know which columns of the sub-channel matrix of the considered cell are representative of users and interferers. Let us consider the set of columns of the sub-channel matrix of the considered cell relative to users located inside this cell and denote it as $\mathbf{H}_n$, which is $\mathbb{C}^{N_a \times N_u}$ matrix. In order to obtain B, we can consider different matrix operations over matrix $\mathbf{H}_n$. In particular we will consider:

1. $\mathbf{B} = \mathbf{H}_n^H$;
2. $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1} \mathbf{H}_n^H$.

This two different types of matrix assume different meaning. The first one is a matched receiver and will be denoted as TC, whereas the latter is the pseudo-inverse of matrix $\mathbf{H}_n$ and is a zero-forcing receiver and will be denoted as ZF. Matrix $\mathbf{B}$ will always be of the type $\mathbb{C}^{N_u \times N_a}$ and the pre-coding operation will be applied as in

(5.12).
We can see that this type of approach tries to take advantage of the knowledge of the users located in a cell in order to apply a pre-coding method that will hopefully be advantageous for such a set of users.
The message passing algorithm that will be applied for the decoding is the modified version presented in previous section.

5.3.2 Power Based Pre-Coding

We now consider another approach based on received power. As in known users pre-coding, in order to build matrix $\mathbf{B}$, we need a matrix $\mathbf{H}_n$ derived from a sub-set of columns of the sub-channel matrix for a certain cell, but in this case we do not assume to know users location. Instead we choose the set of N_u users with the higher received power and build $\mathbf{H}_n$ as the set of columns of the sub-channel matrix relative to such users. Therefore, recalling (5.11), the set of N_u most powerful users will be considered as useful signal, whereas other users will be considered as interferers. Hence we can multiply the received signal at the considered BS by this B matrix as in (5.12), and obtain a representative vector to be passed to the central pool of length N_u.
In formulas, consider the set $\mathcal{A}(c)$ of indexes of antennas located in cell c. Given the channel from user k to the BS in c, we compute

$$p(k) = \sum_{n \in \mathcal{A}(c)} |H(n,k)|^2 \tag{5.14}$$

for each user in the cellular network, and consider the N_p users with highest $p(k)$ for building matrix $\mathbf{B}$.
As for known users pre-coding the massage passing algorithm to be used is the modified version.

5.3.3 Known Users plus Powerful Ones

This approach exploits the theory presented for known users pre-coding with the difference that we do not consider only the users located inside the cell, but even a certain number of users located outside the considered cell and that reach the considered BS with the highest power. We then assume to know users' locations and decide a number of additional users to consider, denoted as N_{add}. Then, among all the users located outside the cell, we consider the N_{add} ones that arrive to the cell's BS with the highest power. The matrix $\mathbf{H}_n$ will hence be composed by the columns of the sub-channel

matrix associated to users inside the cell and by the columns of the sub-channel matrix associated to the N_{add} selected users. Matrix B will then be built with the previously depicted different approaches. In formulas, we consider users located in cell c and compute $p(k)$ (5.14) for each one of the users located outside the cell and choose the N_{add} most powerful. The matrix $\mathbf{B}$ will be built based on users located inside the cell and the N_{add} most powerful, chosen based on their $p(k)$ value.

5.4 Simulation Results for Pre-Coding Sparsification Methods

We are here going to present simulation results obtained with the different methods explained in previous section. First we will analyse and discuss results regarding known users pre-coding, then power based pre-coding and then compare the two methods.

5.4.1 Known users pre-coding: relative error and channel capacity

We here want to analyse results obtained for knonw users pre-conding. We assumed that 4 users with the same transmitting power and 8 receiving antennas are present in each cell, hence matrix $\mathbf{B}$ is a $\mathbb{C}^{4\times 8}$ matrix. Simulation have been run for 2 different types of matrices $\mathbf{B}$ and results are compared in the following figures. Notice that we denoted as pure RGMP results obtained with no pre-coding, with $\mathbf{H}_n^H$ results obtained with pre-coding matrix $\mathbf{B} = \mathbf{H}_n^H$ and with ZF results obtained with pre-coding matrix $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1}\mathbf{H}_n^H$.
Notice that relative error formula for the pre-coding case is different from the one presented for pure RGMP in (3.30), since we must take into account that multiplication by B has been applied. Hence relative error for pre-coding results in being

$$\epsilon_{pre-c} = \frac{||P\sum_{c=1}^{N_c}\mathbf{B}_c\mathbf{H}_c x_c^{(t)} - P^{\frac{1}{2}}\sum_{c=1}^{N_c}\mathbf{B}_c\mathbf{y}_c||}{||P^{\frac{1}{2}}\sum_{c=1}^{N_c}\mathbf{B}_c\mathbf{y}_c||} \tag{5.15}$$

where $[]_c$ denotes vectors and matrices of cell c.
Figure 5.10 shows the relative error results obtained for the different types of $\mathbf{B}$ matrices. We show results obtained for two SNR values, respectively 5 dB and 100 dB in order to show effects of noise over results. We can see that ZF results in having a relative error higher than other methods in 5 dB case, whereas for 100 dB SNR we see

that all methods converge to the same relative error after 5 MP iterations.

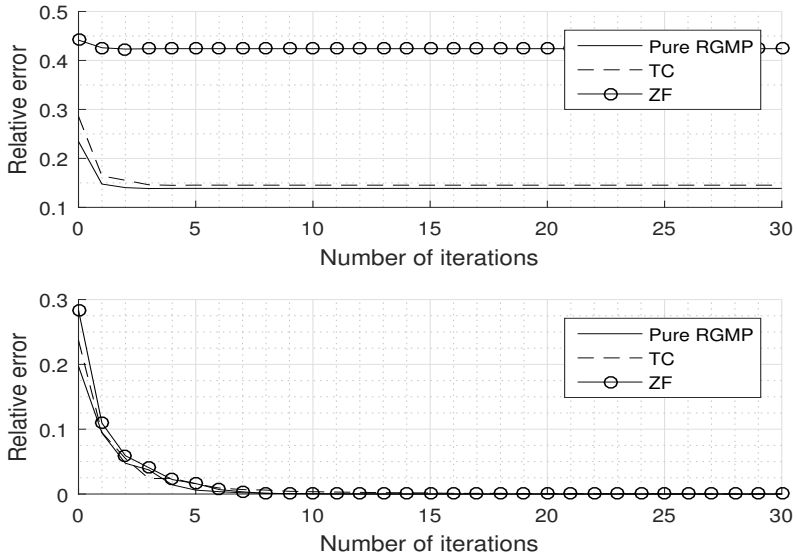

Figure 5.10: Relative error with different pre-coding matrices: upper case with 5 dB SNR, lower case with 100 dB SNR.

Fig. 5.11 reports the achievable sum-rate results obtained with the previously discussed pre-coding matrices $\mathbf{B}$ with the same notation used for relative error. Every method presents a lower achievable sum-rate than pure RGMP's one. We can also notice that there is an SNR threshold value, before which H^H,i.e. TC, has a bigger achievable sum-rate than ZF and after which ZF has a bigger achievable sum-rate than H^H.

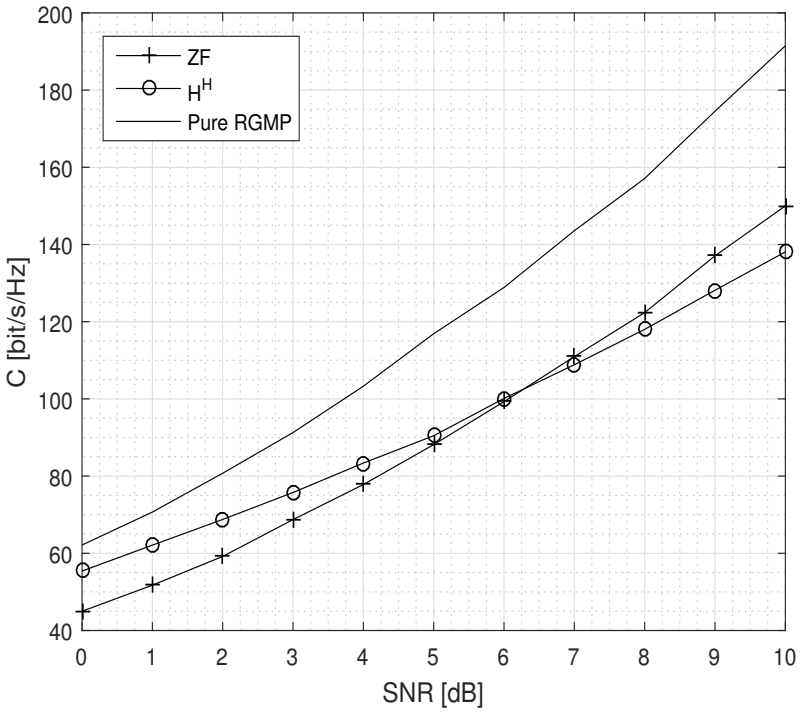

Figure 5.11: achievable sum-rate with different pre-coding matrices.

5.4.2 Power Based Pre-Coding: Relative Error and Achievable Sum-Rate

We want here to analyse results obtained for power based pre-conding. We assumed that 4 users with the same transmitting power and 8 receiving antennas are present in each cell, and that power based pre-coding is performed by choosing the N_u most powerful signals coming from each one of the users present in the cellular network. We show results relative to different number of most powerful users, in particular 1, 3, 5 and 7 and test the different types of matrices that we presented in previous sections.

Case 1 :$\mathbf{B} = \mathbf{H_n}$

Figure 5.12 presents relative error results for power-based pre-coding with $\mathbf{B} = \mathbf{H}_n^H$ for the different number of selected users compared to pure RGMP,i.e. without pre-coding, for both 5 and 100 dB SNR values. We can see that, when noise level is high (i.e. 5dB), relative error increases with the number of selected users and that, overall, is higher than pure RGMP's one. When noise level is low however we notice that every realization reaches approximatively the same relative error obtained with pure RGMP with different numbers of message passing iterations.
Figure 5.13 presents a closer vision to the obtained results. In the 5 dB SNR case, for 1 message passing iteration, both 1 user and 3 users power based pre-coding exhibit a lower relative error than pure RGMP's one, but with a higher number of iteration pure RGMP reaches the lowest relative error. When SNR is 100 dB however, power based pre-coding with a single users reaches a relative error which is the minimum among all realizations and iteration numbers with a lower number of iteration with respect to pure RGMP.

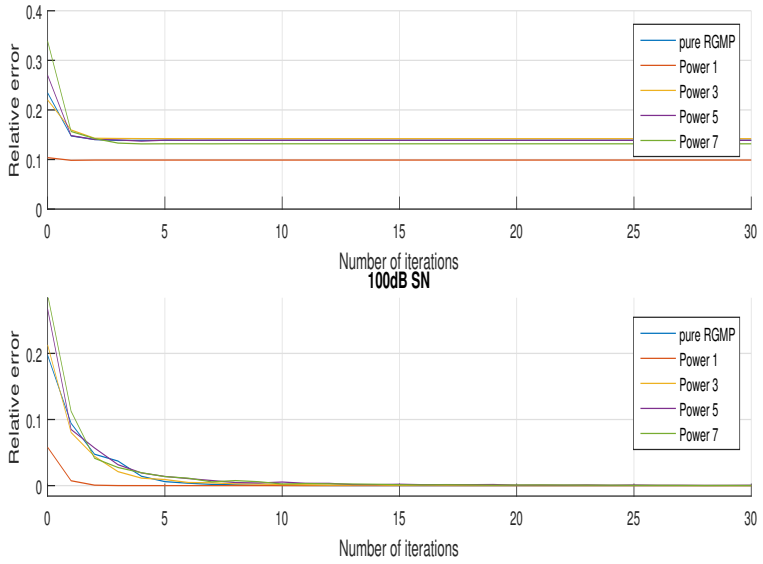

Figure 5.12: Relative error with different number of powerful users with $B = H_u^H$: upper case with 5 dB SNR, lower case with 100 dB SNR.

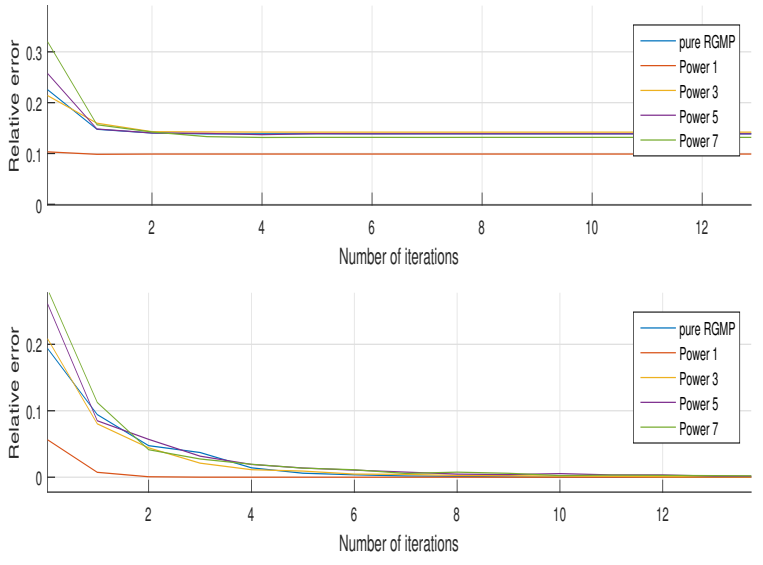

Figure 5.13: Relative error with different number of powerful users with $B = H_u^H$: upper case with 5 dB SNR, lower case with 100 dB SNR, zoom.

Figure 5.14 compares in terms of relative error between known users and power based pre-coding with 4 users for the selected type of B matrix for both 5 dB and 100 dB SNR. We can see that, when noise is high, known users pre-coding presents a lower relative error than power based, whereas, when the noise level is low, the two pre-coding approaches exhibit approximatively the same relative error after 4 message passing iterations.
Figure 5.15 shows the capacity results obtained with power based pre-coding with different number of selected users. We can see that, as in previous, case system's capacity decreases as the number of selected users decreases.
Figure 5.19 shows the capacity comparison between power based with 4 selected users and known users pre-coding. We can see that both methods exhibit approximatively the same capacity, except for high SNR values, where known users pre-coding reaches higher achievable sum-rate values. In order to confirm this results we can see in Fig. 5.16, 5.17 and 5.18 the cumulative distribution functions (CDFs) of achievable sum-rate values respectively 0, 2 and 4 dB SNR values. Results are confirmed and we state that this pre-coding method achieves higher sum-rate values for the considered SNR values and number of users.

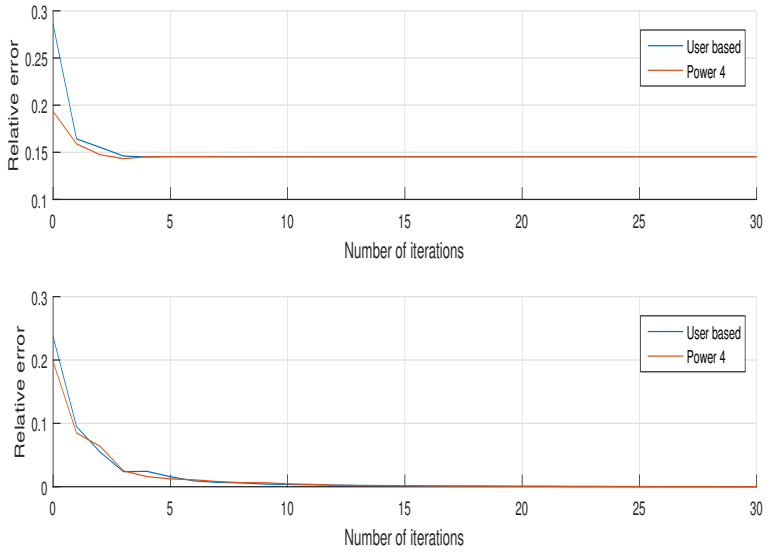

Figure 5.14: Relative error comparison between known users and power based pre-coding with $B = H_n^H$: upper case with 5 dB SNR, lower case with 100 dB SNR.

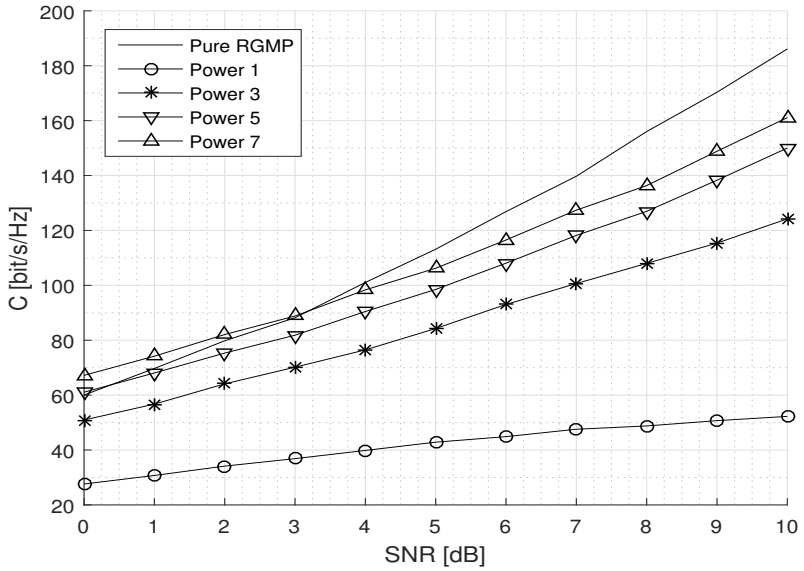

Figure 5.15: achievable sum-rate with different number of powerful users with $B = H_u^{\dagger}$.

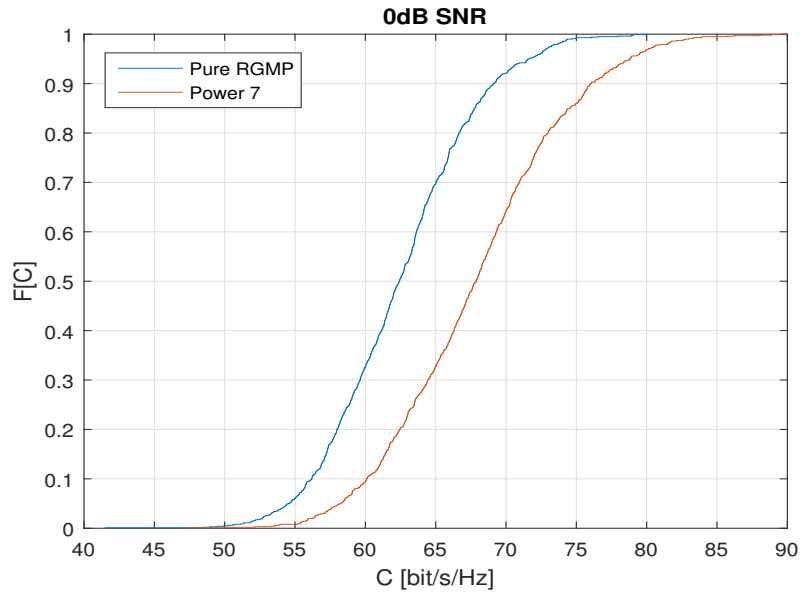

Figure 5.16: CDF of achievable sum-rate values for pure RGMP and power based pre-coding with 7 powerful users: $B = H_n^H$.

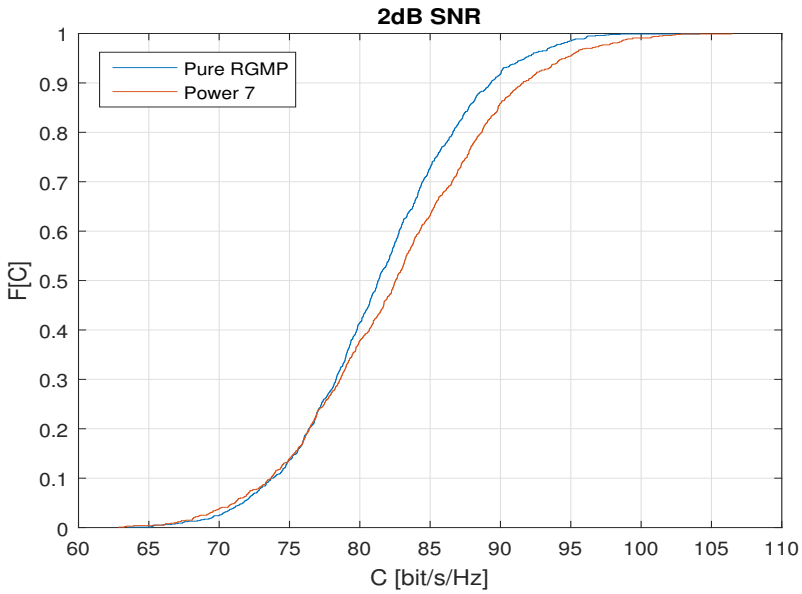

Figure 5.17: CDF of achievable sum-rate values for pure RGMP and power based pre-coding with 7 powerful users: $B = H_n^H$.

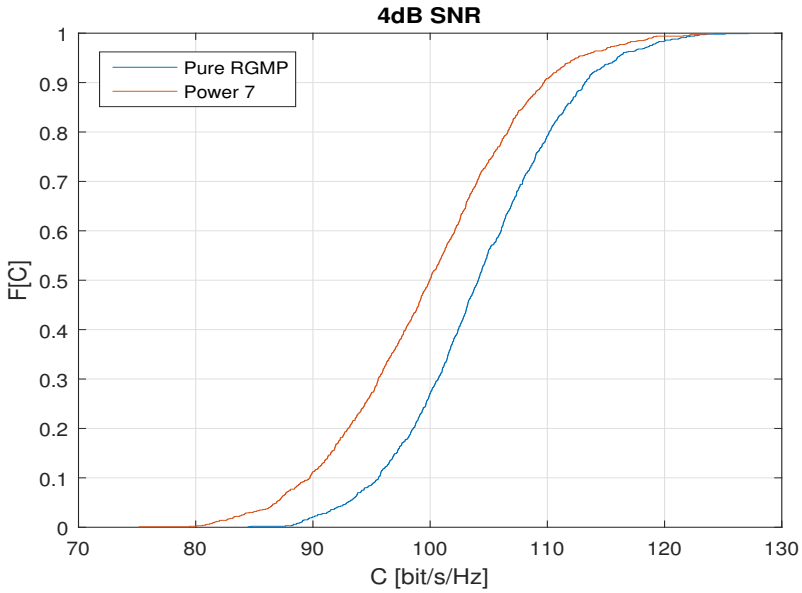

Figure 5.18: CDF of achievable sum-rate values for pure RGMP and power based pre-coding with 7 powerful users: $B = H_n^H$

.

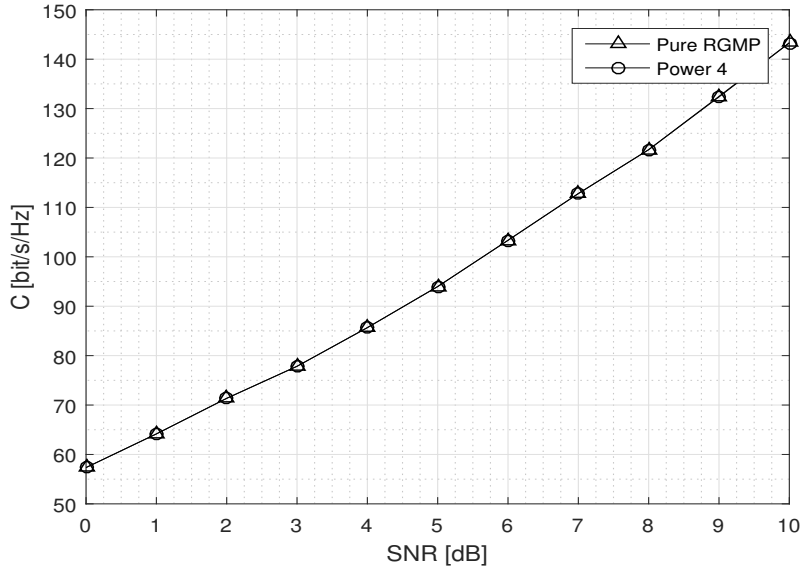

Figure 5.19: achievable sum-rate comparison between known users and power based pre-coding: $B = H_n^H$.

Case 2 :B $= (\mathbf{H_u^H H_u})^{-1}\mathbf{H_u^H}$

We investigate now relative error results when matrix B is of the form $(\mathbf{H}_n^H\mathbf{H}_n)^{-1}\mathbf{H}_n^H$.
Figure 5.20 reports relative error results for both 5 dB and 100 dB SNR values. We can see that in 5 dB SNR case relative error increases with the number of selected users and that each one of such realizations present a relative error over the number of message passing iterations higher respect to pure RGMP's one. We obtain different results at low noise regime, where we can see that all realization present a relative error that converges to pure RGMP's one at approximatively 10 message passing iterations. Figure 5.21 presents a closer vision to the relative error obtained results. We can notice hat power based pre-coding with a single user reaches pure RGMP's relative error with two message passing iterations and decreases as the number of iterations increases.

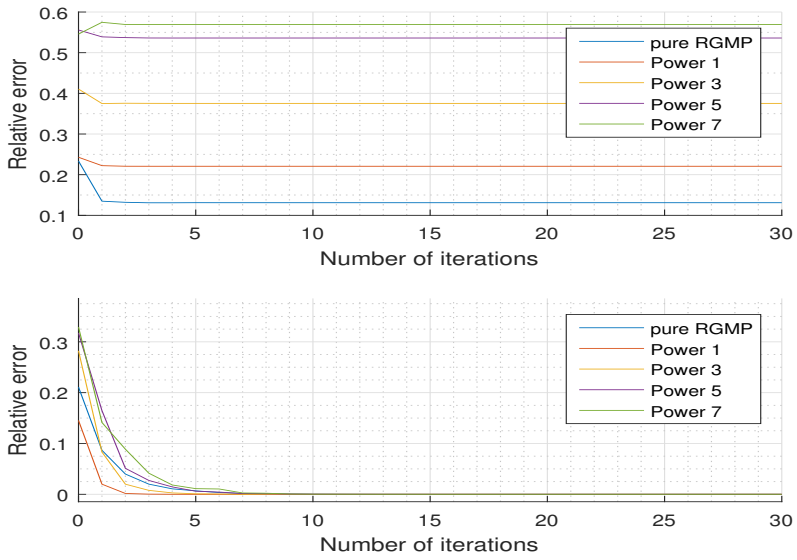

Figure 5.20: Relative error with different number of powerful users with $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1} \mathbf{H}_n^H$: upper case 5 dB SNR, lower case 100 dB SNR.

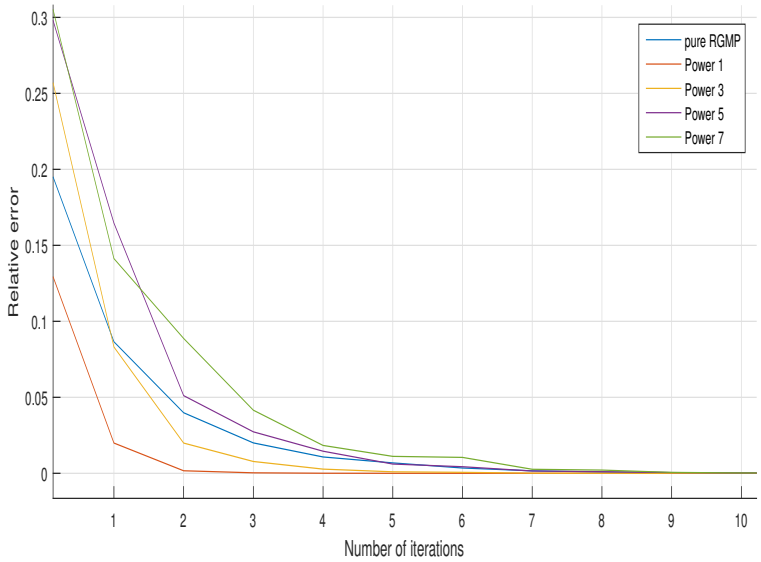

Figure 5.21: Relative error with different number of powerful users with $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1} \mathbf{H}_n^H$: 5 dB SNR, zoom.

Figure5.22 presents the difference in relative error for both 5 db and 100 dB SNR for known users and power based pre-coding with 4 selected users. We can see that power based for this B implementation exhibits a higher relative error. In particular we can see that in 5 dB case power based pre-coding does not reach the same relative error of known users, whereas in 100 dB case in reaches the same relative error with approximatively 30 message passing iterations.
In this case the known users pre-coding is preferable over the power based pre-coding with 4 users if we are interested in relative error minimization.

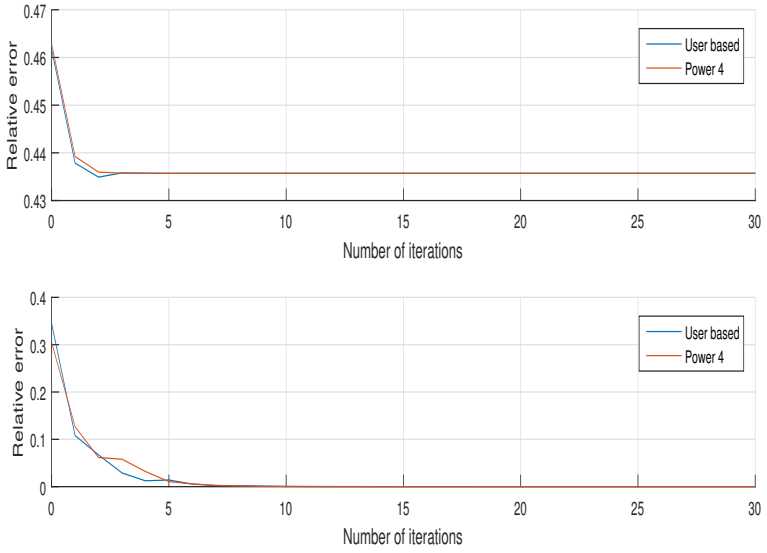

Figure 5.22: Relative error comparison between known users and power based pre-coding for $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1} \mathbf{H}_n^H$: upper figure with 5 dB SNR, lower figure with 100 dB SNR.

Figure 5.23 presents cchievable sum-rate results obtained with power based pre-coding with 1,3,5 and 7 selected users compared to pure RGMP's. We can notice that there is a threshold value on the number of considered users over which achievable sum-rate stops to grow with the number of users and starts to decrease with them. We can in fact notice that achievable sum-rate grows when passing from 1 user to 3, but then decreases for low SNR values passing from 3 to 5 users, and is lower for all SNR values when passing from 3 to 7 users.

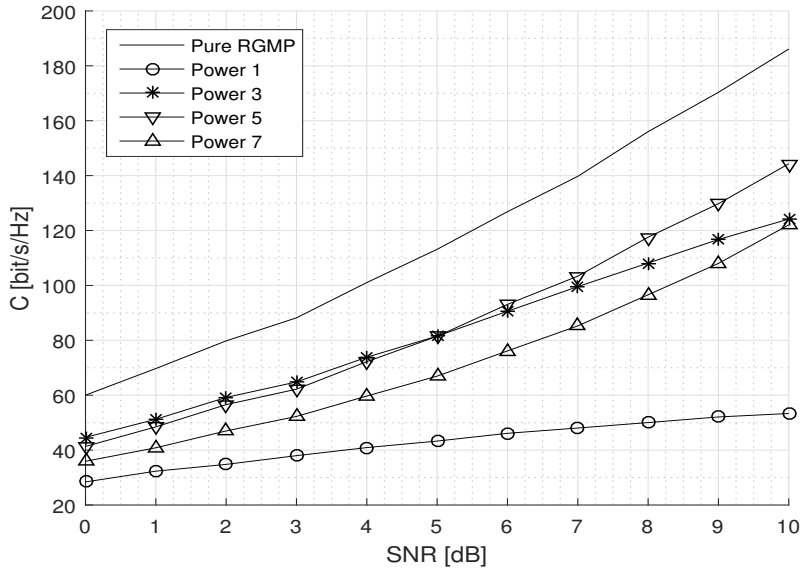

Figure 5.23: Achievable sum-rate with different number of powerful users with $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1} \mathbf{H}_n^H$

Figure 5.24 shows achievable sum-rate results comparison for known users and power based pre-coding with 4 selected users. We can see that the two implementations achieve the same sum-rate values.

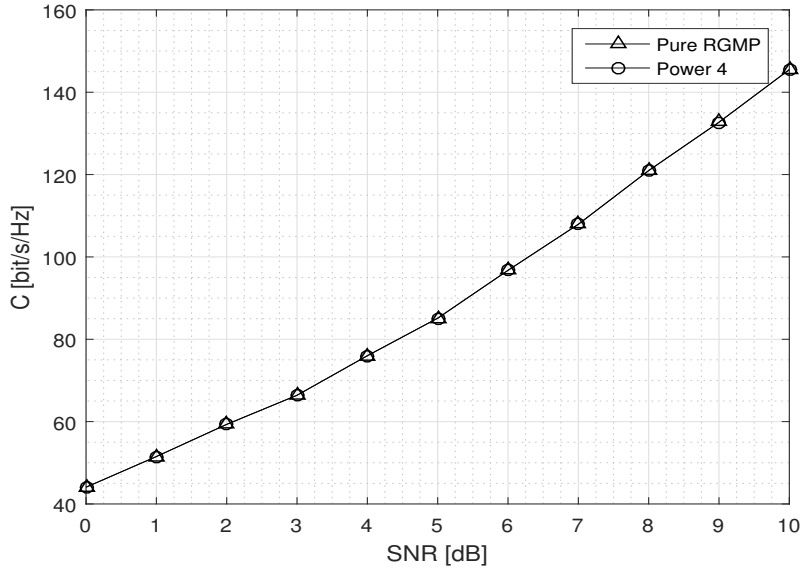

Figure 5.24: Achievable sum-rate comparison between known users and power based pre-coding: $\mathbf{B} = (\mathbf{H}_n^H \mathbf{H}_n)^{-1} \mathbf{H}_n^H$

5.4.3 Known Users plus Powerful Ones Pre-Coding: Relative Error and Achievable Sum-Rate

Case 1 :$\mathbf{B} = \mathbf{H_n^H}$

As previously discussed $\mathbf{B}$ matrix can be implemented in several ways. In this section we are going to analyse obtainable results with known users plus powerful ones pre-coding when $\mathbf{B} = \mathbf{H}_n^H$.
Figure 5.25 presents relative error results for pre-coding methods compared to pure RGMP's one. Both $5dB$ and $100dB$ SNR cases are presented in this figure. We can notice that additional users worsen relative error in $5dB$ SNR case, whereas in $100dB$ case we can notice that all methods reach approximatively the same relative error after ≈ 0 message passing iterations.
From figure 5.26 we can however notice that none of the presented methods can reach the same relative error reached by pure RGMP.

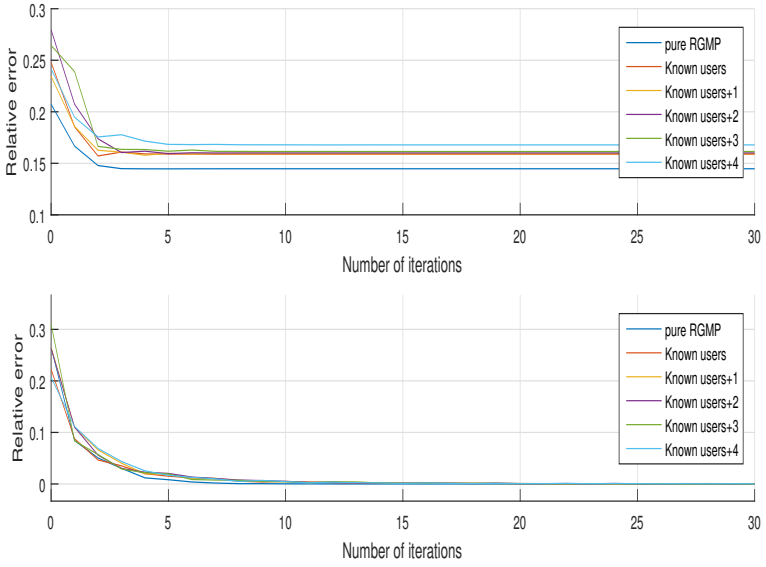

Figure 5.25: Relative error comparison with known users plus powerful ones for $\mathbf{B} = \mathbf{H}_n^H$: upper figure with 5 dB SNR, lower figure with 100 dB SNR

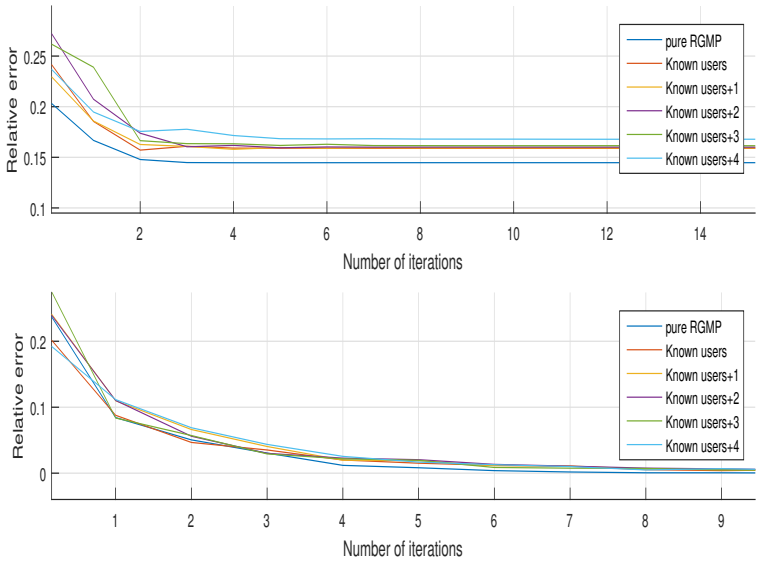

Figure 5.26: Relative error comparison with known users plus powerful for $\mathbf{B} = \mathbf{H}_n^H$: upper figure with 5 dB SNR, lower figure with 100 dB SNR, zoom.

Figure 5.27 shows achievable sum-rate results obtained with the presented method compared to the one obtainable with pure RGMP. We can notice that for low SNR values known users plus 2, 3 and 4 powerful users pre-coding achieves higher sum-rate values than pure RGMP. After ≈ 3 dB SNR we however notice that pure RGMP is the best performing, while achievable sum-rate for the presented pre-coding method decreases with the number of considered users.

Figures 5.28, 5.29, 5.30 reports CDFs for achievable sum rates of pure RGMP and pre-coding sparsification with parameters that in Fig. 5.27 achieved higher sum-rate values than pure RGMP. We see that CDFs confirm the results obtained in Fig. 5.27, where for low SNR values, known users pre-coding plus 2, 3 and 4 users achieves higher sum-rate values.

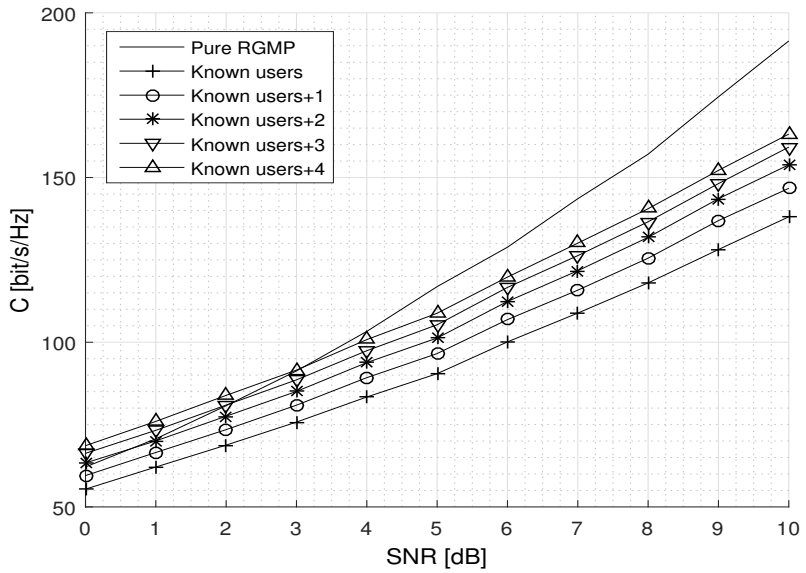

Figure 5.27: Achievable sum-rate comparison with known users plus powerful for $\mathbf{B} = \mathbf{H}_n^H$

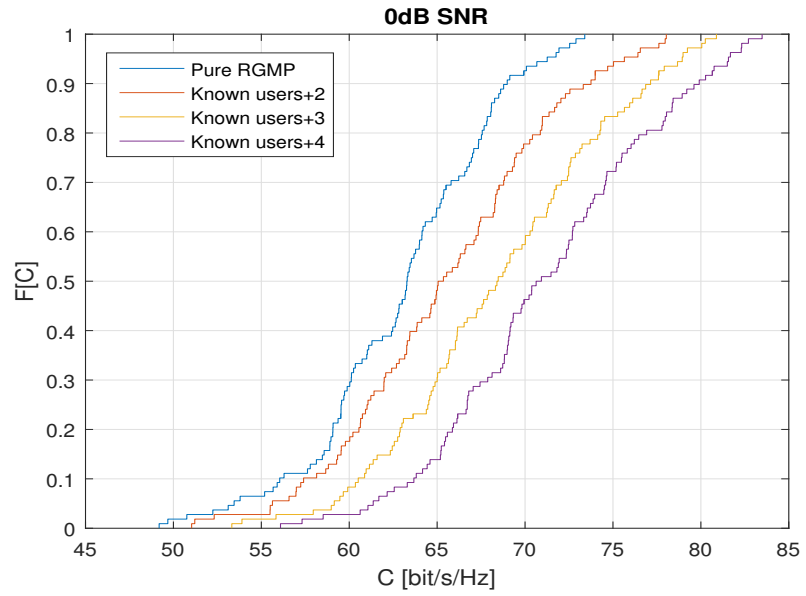

Figure 5.28: CDF of achievable sum-rate values for pure RGMP and known plus powerful users: $\mathbf{B} = \mathbf{H}_n^H$

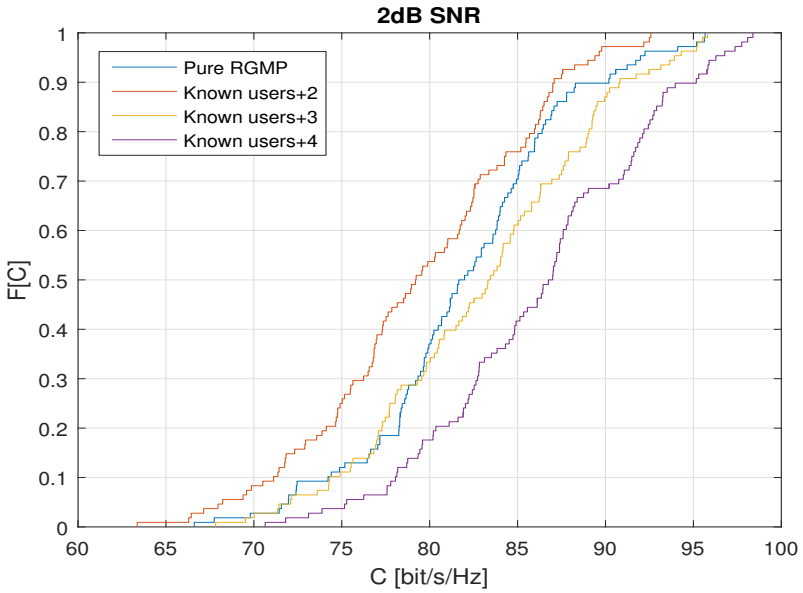

Figure 5.29: CDF of achievable sum-rate values for pure RGMP and known plus powerful users: $\mathbf{B} = \mathbf{H}_n^H$

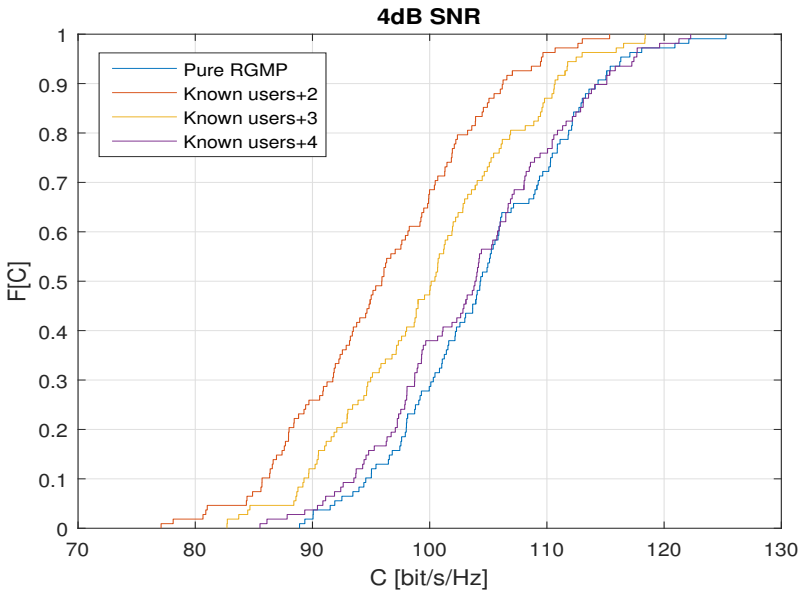

Figure 5.30: CDF of achievable sum-rate values for pure RGMP and known plus powerful users: $\mathbf{B} = \mathbf{H}_n^H$

Case 2 :B $= (\mathbf{H_n^H H_n})^{-1}\mathbf{H_n^H}$

We now discuss results obtained for know users plus powerful ones pre-coding when **B** is ZF. Fig. **??** shows relative error results. We see that in 5 dB SNR case relative error obtained with pre-coding methods is higher than pure RGMP's and, in particular, that it grows with the number of considered users (except for known users + 4). In 100 dB SNR case we notice that all methods converge to approximatively the same relative error obtained with pure RGMP after 10 MP iterations. In particular Fig. 5.32 shows that after seven MP ioterations all methods except for known users+4 reach approximatively the same relative error, whereas known users + 4 need 5 more MP iterations to reach this relative error value.

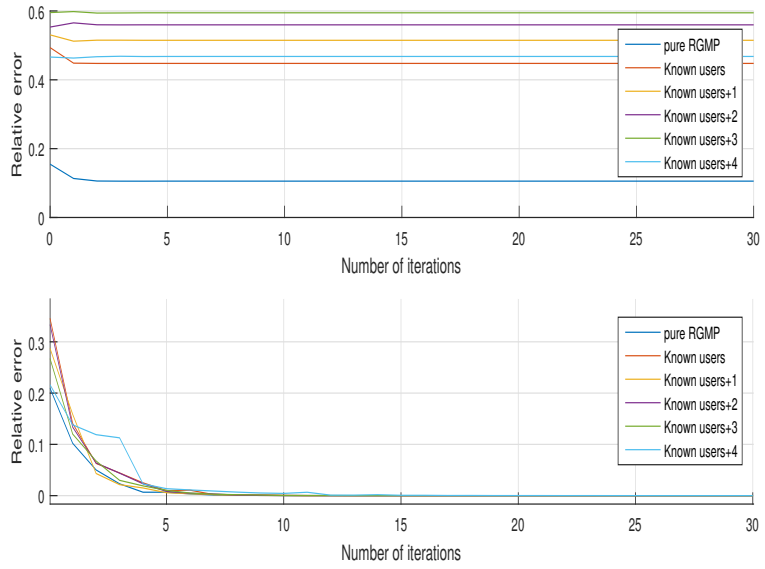

Figure 5.31: Relative error comparison with known users plus powerful for $B = (H_n^H H_n)^{-1} H_n^H$: upper figure is for 5 dB SNR, lower figure for 100 dB SNR.

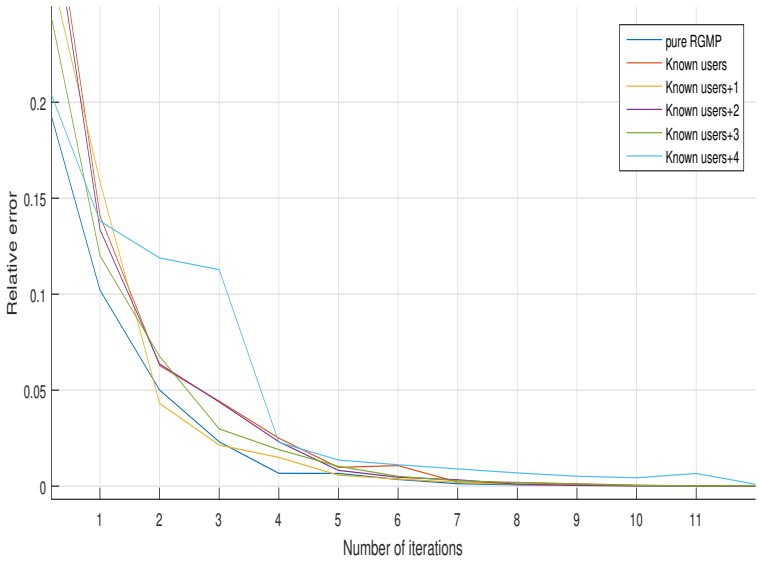

Figure 5.32: Relative error comparison with known users plus powerful for $B = (H_n^H H_n)^{-1} H_n^H$: 100 dB SNR case, zoom.

Figure 5.33 shows achievable sum-rate results obtained with known-users pre-coding plus powerful ones with **B** of type ZF. We notice that achievable sum rate decreases with the number of considered users, and hence the best performing strategy is keeping only known users.

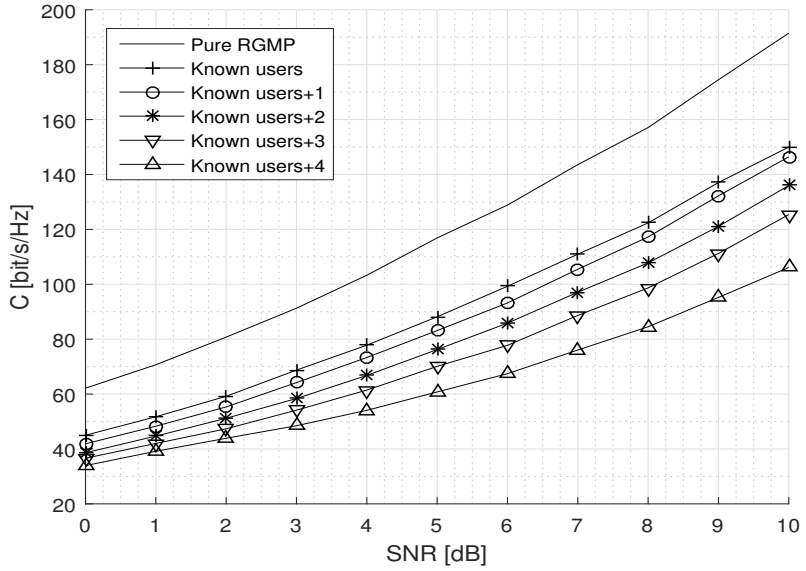

Figure 5.33: Capacity comparison with known users plus powerful for $B = (H_n^H H_n)^{-1} H_n^H$

5.5 Decoding Computational Complexity Analysis

We recall that all presented sparsification methods aim at reducing the computational complexity of the decoding algorithm while maintaining the same system performance.
In order to state if proposed algorithms are effectively reducing the computational complexity we have first to obtain a way to measure it. Since all sparsification methods are performed before RGMP algorithm we analyse its computational complexity by looking at steps of Algorithm 2. We see that, after checking whether the considered channel matrix entry is or not equal to zero, two sums over the total number K of users are performed if the considered entry is different from zero. Hence the total number of operations at this point is given by the multiplication of the number of channel matrix entries different from zero and 2 times the number of users in the considered scenario. Then this number of operation is repeated until stopping criterion is satisfied. Hence the total number of operations required for decoding are

$$N_{op} = 2\,K\,s\,I \tag{5.16}$$

where s denotes the number of channel matrix entries different from 0, and I the number of message passing iterations needed to satisfy

the stopping criterion.
Since the decoding algorithm is iterative we stated that convergence is not assured and furthermore we do not know whether computed marginals are correct. We hence decided to set a stopping criterion based on relative error. Considering iteration t we decide to stop here the decoding process is future values of relative error do not increment or decrement of a value equal to the 1% of relative error at iteration t. This criterion allows us to save computational time, since further efforts in the decoding process do not lead to lower errors, i.e. a more precise decoding.
In following tables we report, for all of the presented methods and for each analysed parameter value, the sparsification level and the computational complexity in terms of number of required operation for the decoding process for three different SNR values. Each table reports the used sparsification method, its parameter and the corresponding value, the sparsification level, the number of MP iterations to satisfy the stopping criterion and the total number of operations. We see that, among all presented methods and corresponding parameter values, some of them allow a reduction of the computational complexity analysis respect to pure RGMP.

$SNR = 0dB$				
Sparsification method	Parameter	Spasification level	Number of MP iterations	Total number of operations
Pure RGMP	.	8192	2	2097152
Power-based	$P_{min} = 0.001$	4121	3	1582464
Power-based	$P_{min} = 0.005$	1345	3	516480
Power-based	$P_{min} = 0.01$	755	3	289920
Power-based	$P_{min} = 0.1$	154	2	39424
Orthogonal users	$T_{prod} = 0.005$	6728	2	1722368
Orthogonal users	$T_{prod} = 0.0075$	5560	3	2135040
Orthogonal users	$T_{prod} = 0.01$	4288	2	1097728
Orthogonal users	$T_{prod} = 0.05$	648	2	165888
CBM	$L_r = 1$	7168	2	1835008
CBM	$L_r = 3$	5120	3	1966080
CBM	$L_r = 5$	3072	3	1179648
CBM	$L_r = 7$	1024	2	262144
MIBM	$L_r = 1$	7168	1	917504
MIBM	$L_r = 3$	5120	2	655360
MIBM	$L_r = 5$	3072	2	786432
MIBM	$L_r = 7$	1024	2	262144
Power-based pre-coding	$H^H, 1usr.$	1024	2	262144
Power-based pre-coding	$H^H, 3usr.$	3072	3	786432
Power-based pre-coding	$H^H, 4usr.$	4096	3	1572864
Power-based pre-coding	$H^H, 5usr.$	5120	3	1966080
Power-based pre-coding	$H^H, 7usr.$	7168	4	3670016
Power-based pre-coding	$(H^H H)^{-1} H^H, 1usr.$	1024	2	262144
Power-based pre-coding	$(H^H H)^{-1} H^H, 3usr.$	3072	1	393216
Power-based pre-coding	$(H^H H)^{-1} H^H, 4usr.$	4096	1	524288
Power-based pre-coding	$(H^H H)^{-1} H^H, 5usr.$	5120	1	655360
Power-based pre-coding	$(H^H H)^{-1} H^H, 7usr.$	7168	2	1835008
Power-based plus pow.ones pre-c	$H^H, 4usr.$	4096	2	1048576
Power-based plus pow.ones pre-c	$H^H, 5usr.$	5120	4	2621440
Power-based plus pow.ones pre-c	$H^H, 6usr.$	6144	2	1572864
Power-based plus pow.ones pre-c	$H^H, 7usr.$	7168	2	1835008
Power-based plus pow.ones pre-c	$H^H, 8usr.$	8192	5	5242880
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 4usr.$	4096	1	524288
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 5usr.$	5120	1	655360
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 6usr.$	6144	2	1572864
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 7usr.$	7168	2	1835008
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 8usr.$	8192	2	2097152

Table 5.4: Decoding computational complexity for presented sparsification methods: $0dB$ SNR case

$SNR = 5dB$				
Sparsification method	Parameter	Spasification level	Number of MP iterations	Total number of operations
Pure RGMP	.	8192	4	4194304
Power-based	$P_{min} = 0.001$	4121	3	1582464
Power-based	$P_{min} = 0.005$	1345	4	688640
Power-based	$P_{min} = 0.01$	755	3	289920
Power-based	$P_{min} = 0.1$	154	2	39424
Orthogonal users	$T_{prod} = 0.005$	6728	4	3444736
Orthogonal users	$T_{prod} = 0.0075$	5560	3	2135040
Orthogonal users	$T_{prod} = 0.01$	4288	3	1646592
Orthogonal users	$T_{prod} = 0.05$	648	3	248832
CBM	$L_r = 1$	7168	4	3670016
CBM	$L_r = 3$	5120	5	3276800
CBM	$L_r = 5$	3072	2	786432
CBM	$L_r = 7$	1024	2	262144
MIBM	$L_r = 1$	7168	3	2752512
MIBM	$L_r = 3$	5120	2	1310720
MIBM	$L_r = 5$	3072	4	1572864
MIBM	$L_r = 7$	1024	2	262144
Power-based pre-coding	$H^H, 1usr.$	1024	2	262144
Power-based pre-coding	$H^H, 3usr.$	3072	4	1572864
Power-based pre-coding	$H^H, 4usr.$	4096	4	2097152
Power-based pre-coding	$H^H, 5usr.$	5120	4	2621440
Power-based pre-coding	$H^H, 7usr.$	7168	7	6422528
Power-based pre-coding	$(H^H H)^{-1} H^H, 1usr.$	1024	2	262144
Power-based pre-coding	$(H^H H)^{-1} H^H, 3usr.$	3072	2	786432
Power-based pre-coding	$(H^H H)^{-1} H^H, 4usr.$	4096	2	1048576
Power-based pre-coding	$(H^H H)^{-1} H^H, 5usr.$	5120	3	1966080
Power-based pre-coding	$(H^H H)^{-1} H^H, 7usr.$	7168	2	1835008
Power-based plus pow.ones pre-c	$H^H, 4usr.$	4096	2	1048576
Power-based plus pow.ones pre-c	$H^H, 5usr.$	5120	4	2621440
Power-based plus pow.ones pre-c	$H^H, 6usr.$	6144	5	3932160
Power-based plus pow.ones pre-c	$H^H, 7usr.$	7168	7	6422528
Power-based plus pow.ones pre-c	$H^H, 8usr.$	8192	5	5242880
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 4usr.$	4096	3	1572864
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 5usr.$	5120	2	1310720
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 6usr.$	6144	2	1572864
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 7usr.$	7168	3	2752512
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 8usr.$	8192	4	4194304

Table 5.5: Decoding computational complexity for presented sparsification methods: $5dB$ SNR case

$SNR = 10dB$				
Sparsification method	Parameter	Spasification level	Number of MP iterations	Total number of operations
Pure RGMP	.	8192	6	6291456
Power-based	$P_{min} = 0.001$	4121	31	16352128
Power-based	$P_{min} = 0.005$	1345	31	5336960
Power-based	$P_{min} = 0.01$	755	31	2995840
Power-based	$P_{min} = 0.1$	154	31	611072
Orthogonal users	$T_{prod} = 0.005$	6728	7	6028288
Orthogonal users	$T_{prod} = 0.0075$	5560	5	3558400
Orthogonal users	$T_{prod} = 0.01$	4288	6	3293184
Orthogonal users	$T_{prod} = 0.05$	648	5	414720
CBM	$L_r = 1$	7168	5	4587520
CBM	$L_r = 3$	5120	6	3932160
CBM	$L_r = 5$	3072	5	1966080
CBM	$L_r = 7$	1024	2	262144
MIBM	$L_r = 1$	7168	5	4587520
MIBM	$L_r = 3$	5120	6	3932160
MIBM	$L_r = 5$	3072	6	2359296
MIBM	$L_r = 7$	1024	3	393216
Power-based pre-coding	$H^H, 1usr.$	1024	3	393216
Power-based pre-coding	$H^H, 3usr.$	3072	7	2752512
Power-based pre-coding	$H^H, 4usr.$	4096	7	3670016
Power-based pre-coding	$H^H, 5usr.$	5120	8	5242880
Power-based pre-coding	$H^H, 7usr.$	7168	10	9175040
Power-based pre-coding	$(H^H H)^{-1} H^H, 1usr.$	1024	3	393216
Power-based pre-coding	$(H^H H)^{-1} H^H, 3usr.$	3072	3	1179648
Power-based pre-coding	$(H^H H)^{-1} H^H, 4usr.$	4096	2	1048576
Power-based pre-coding	$(H^H H)^{-1} H^H, 5usr.$	5120	4	2621440
Power-based pre-coding	$(H^H H)^{-1} H^H, 7usr.$	7168	2	1835008
Power-based plus pow.ones pre-c	$H^H, 4usr.$	4096	3	1572864
Power-based plus pow.ones pre-c	$H^H, 5usr.$	5120	10	6553600
Power-based plus pow.ones pre-c	$H^H, 6usr.$	6144	11	8650752
Power-based plus pow.ones pre-c	$H^H, 7usr.$	7168	11	10092544
Power-based plus pow.ones pre-c	$H^H, 8usr.$	8192	13	13631488
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 4usr.$	4096	4	2097152
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 5usr.$	5120	3	1966080
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 6usr.$	6144	2	1572864
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 7usr.$	7168	3	2752512
Power-based plus pow.ones pre-c	$(H^H H)^{-1} H^H, 8usr.$	8192	5	5242880

Table 5.6: Decoding computational complexity for presented sparsification methods: $10dB$ SNR case

5.5.1 Comparison of Presented Sparsification Methods

Results presented in previous section do not take into account the loss in achievable sum-rate that we encountered when discussing simulation results. In this section we compare the different sparsification methods in terms of bot computational complexity and achievable sum-rate. Three different SNR values have been used, respectively 0, 5 and 10 dB.
Fig. 5.34 shows results obtained with centralised sparsification methods (i.e. power-based, orthogonal users-based, CBM, MIBM) for 0 dB SNR. We notice that with orthogonal users-based sparsification we obtain the best performing system, with an achievable sum rate of 60 bit/s/Hz with a computational complexity of $1.0977 \cdot 10^6$ operations. However notice that if a small loss in achievable sum-rate is acceptable for system's requirements, with MIBM antenna selection we can obtain a system with an achievable sum-rate of approximatively 57 bit/s/Hz with a computational complexity of $0.9175 \cdot 10^6$ operations.

Figure 5.35 shows results obtained with centralised sparsification method when $SNR = 5$ dB. We see that with power-based sparsification we obtain a system with achievable sum-rate of approximatively 120 bit/s/Hz and with a computational complexity of $1.5525 \cdot 10^6$ operations. Notice however that if we accept a loss in achievable sum-rate of approximatively 20 bit/s/Hz with orthogonal users-based sparsification we can reduce the computational complexity to $0.2488 \cdot 10^6$ operations.

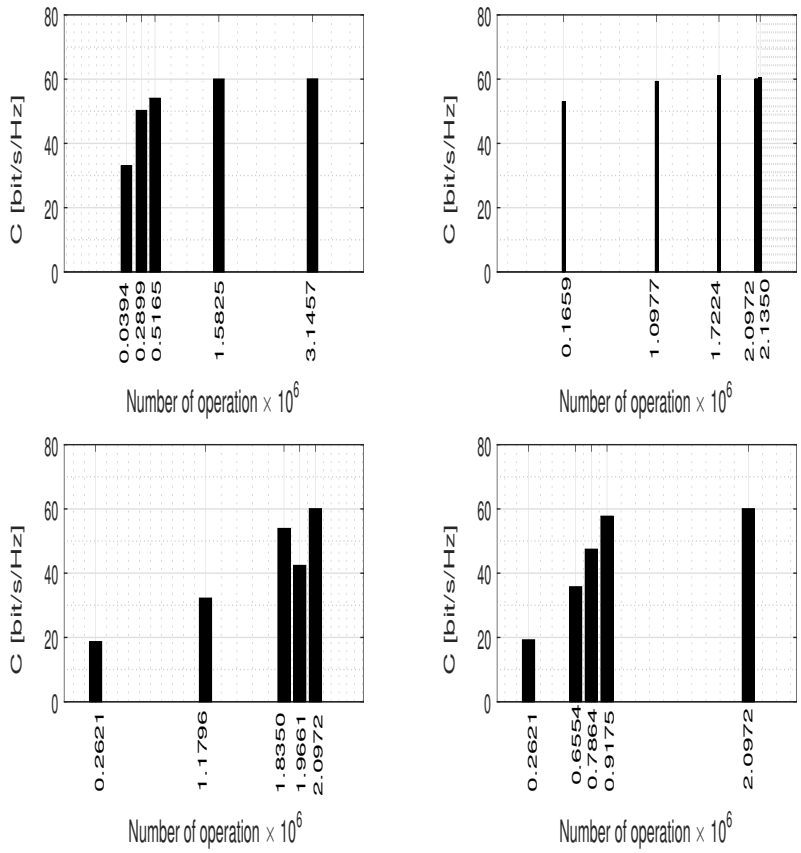

Figure 5.34: Achievable sum-rate versus number of operations for different methods 0 dB SNR: upper left power-based, upper right orthogonal-users, lower left CBM, lower right MIBM.

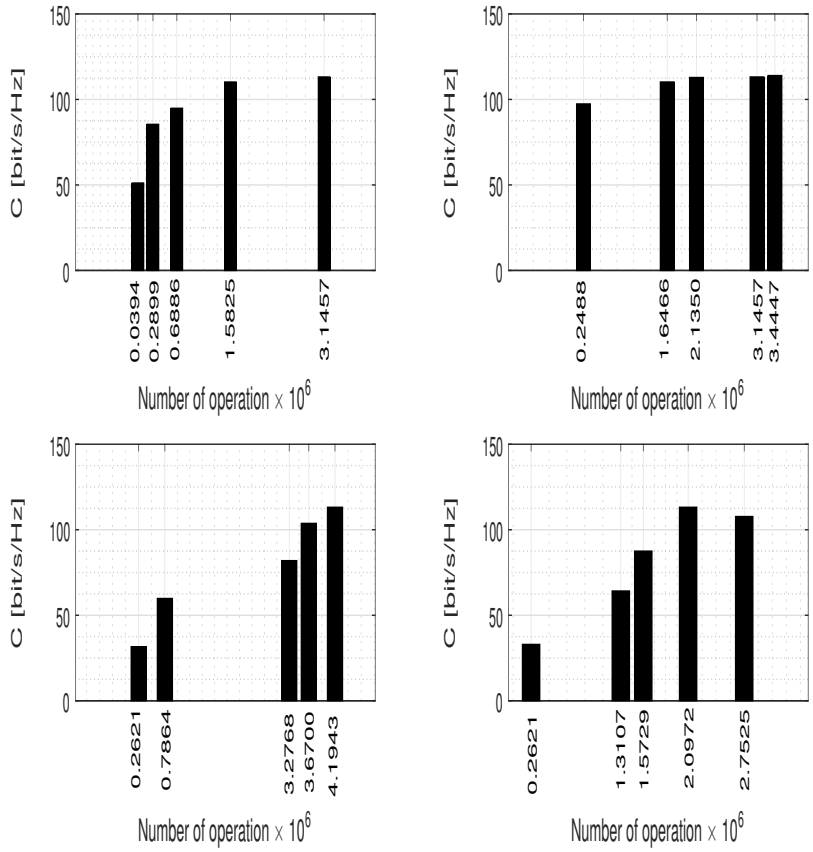

Figure 5.35: Achievable sum-rate versus number of operations for different methods 5 dB SNR: upper left power-based, upper right orthogonal-users, lower left CBM, lower right MIBM.

Figure 5.36 shows results obtained for dcentralysed sparsification methods for 10 dB SNR. We see that with power-based sparsification we obtain a system with an achievable sum-rate of approximatively 175 bit/s/Hz with a computational complexity of $1.6352 \cdot 10^6$ operations. Notice that this is the best result, since in order to reduce computational complexity, we must accept a loss in achievable sum-rate of dozens of bit/s/Hz.

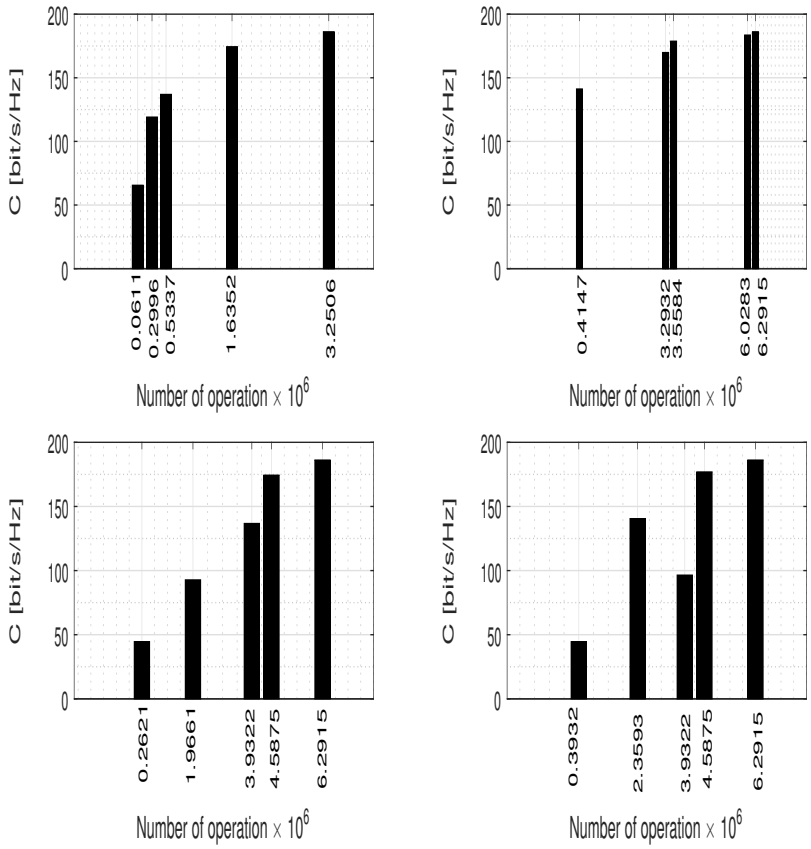

Figure 5.36: Achievable sum-rate versus number of operations for different methods 10 dB SNR: upper left power-based, upper right orthogonal-users, lower left CBM, lower right MIBM.

We now analyse results obtained for distributed sparsification methods (i.e. pre-coding with different types of matrices). We compare the different methods in terms of both achievable sum-rate and computational complexity for SNR values of 0, 5 and 10 dB.
Figure 5.37 shows results for $SNR = 0$ dB. We see that the best compromise between achievable sum-rate and computational complexity is obtained for known users plus powerful ones with ZF matrix, which presents an achievable sum-rate of approximatively 45 bit/s/Hz with a computational complexity of $0.3932 \cdot 10^6$ operations. In order to increase achievable sum-rate we notice that other methods can achieve higher values for this parameter, however the computational complexity significantly grows.

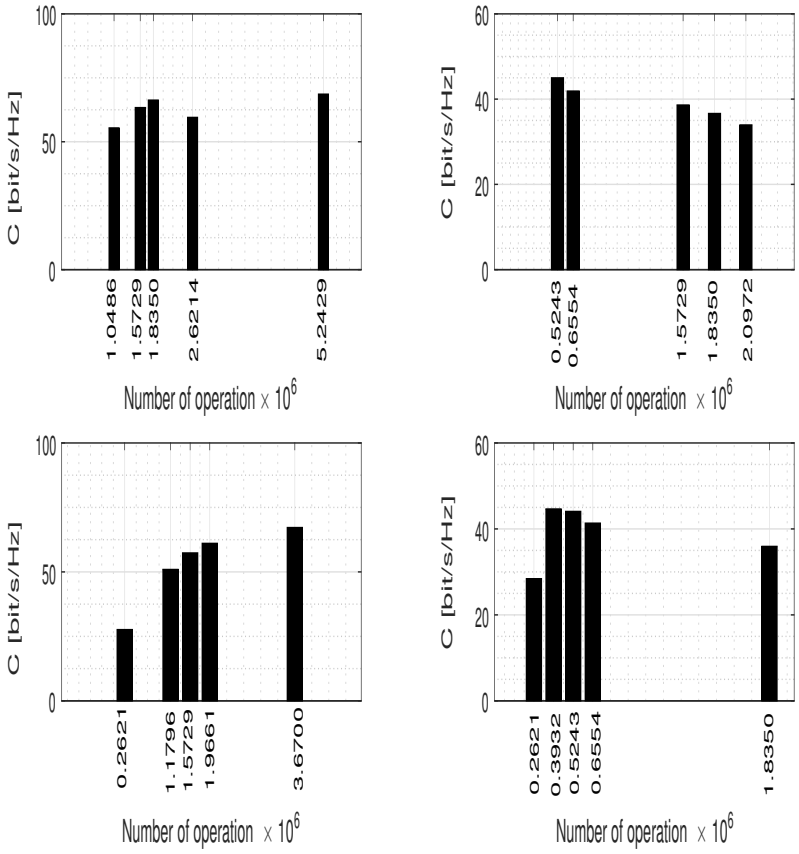

Figure 5.37: Achievable sum-rate versus number of operations for different methods 0 dB SNR: upper left power based TC, upper right power based ZF, lower left known users plus powerful ones TC, lower right known users plus powerful ones ZF.

Figure 5.38 shows results obtained for distributed sparsification

methods when $SNR = 5$ dB. We see that the best compromise is obtained with power-based pre-coding with TC matrix, obtaining a siystem with achievable sum rate of approximatively 90 bit/s/Hz and a computational complexity of $1.0488 \cdot 10^6$ operations. Notice that, in order to augment achievable sum-rate of 15 bit/s/Hz (maximum obtaiable values among all methods) computational complexity shall be increased to $5.2429 \cdot 10^6$ operations.

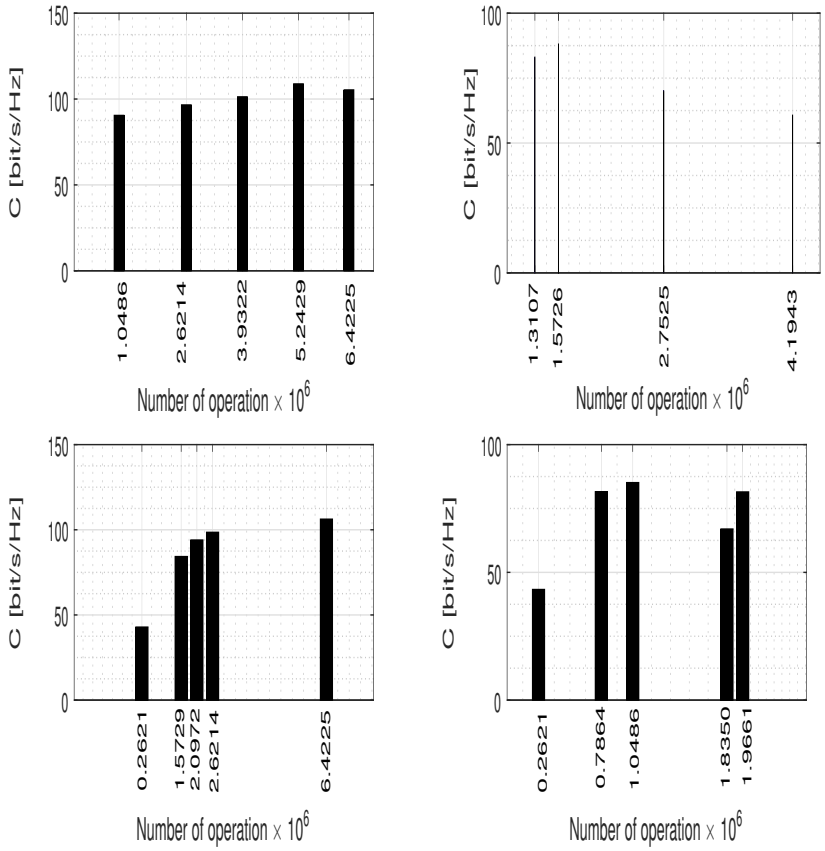

Figure 5.38: Achievable sum-rate versus number of operations for different methods 5 dB SNR: upper left power based TC, upper right power based ZF, lower left known users plus powerful ones TC, lower right known users plus powerful ones ZF.

Figure 5.39 shows results obtained with distributed sparsification methods when $SNR = 10$ dB. We see that the best approach in this case is known users plus poiwerful ones with ZF matrix, which presents an achievable sum-rate pof approximatively 150 bit/s/Hz and a computational complexity of $1.0488 \cdot 10^6$ operations. Notice that the maximum achevable sum-rate among all methods is ap-

proximatively 160 bit/s/Hz, obtainable with a computational complexity of $9.1750 \cdot 10^6$ operations, which justifies our choice for the best approach to follow.

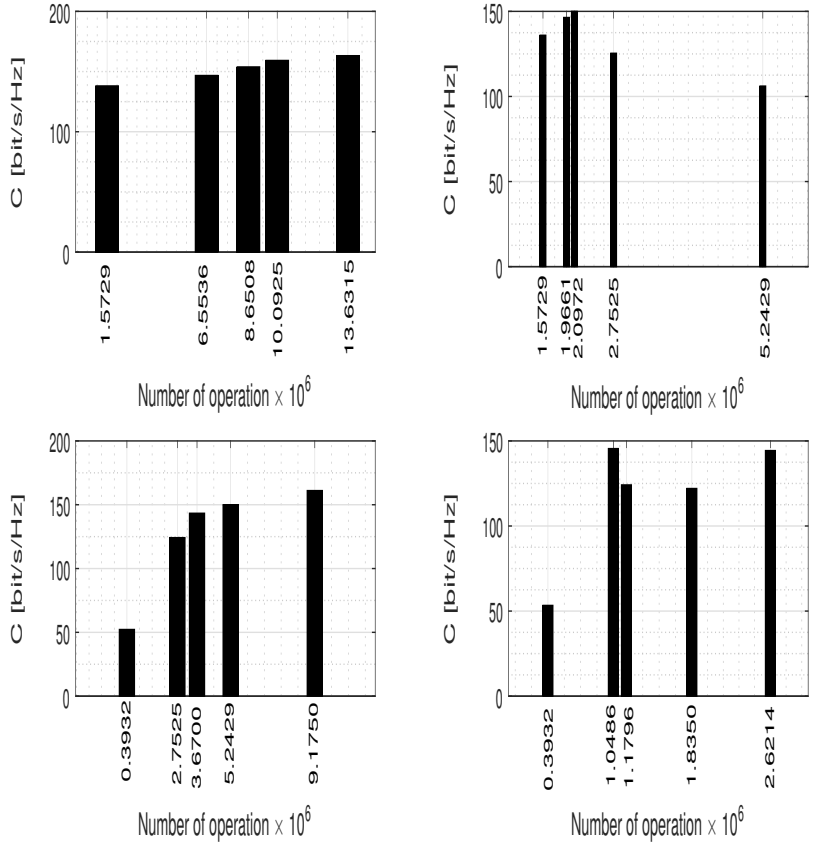

Figure 5.39: Achievable sum-rate versus number of operations for different methods 10 dB SNR: upper left power based TC, upper right power based ZF, lower left known users plus powerful ones TC, lower right known users plus powerful ones ZF.

Chapter 6

Conclusions

In this work we presented C-RAN, a promising architecture to be implemented on 5G networks. We then introduced a model for communication over this architecture, where we interpreted communications between users and BSs as MIMO channels. We then reviewed MP algorithms of [3] and discussed their convergence properties as well as their computational complexity. As stated before, this algorithms require a computational complexity that grows quadratically with the number of users and RRHs in the network and result hence in a prohibitively high computational complexity. Subsequently, we introduced methods to overcome such an issue by sparsifying the channel matrix, as we observed in Algorithms 1 and 2 that the number of operations required for the decoding process depends on the number of entries of the channel matrix that are different from zero. We introduced and tested two different approaches: a centralised one, in which sparsification is applied at the central BBU Pool and a distributed one, in which sparsification is performed as pre-coding at the BS of each cell. Different approaches have been proposed for both distributed and centralised sparsification and each one has been tested in terms of relative error, sparsification level (i.e. number of channel matrix coefficients that are different from zero) and achievable sum-rate.
We presented first the power-based centralised channel sparsification, in which we set a threshold value P_{min} on the squared module of coefficients of the channel matrix and set to zero those which were below P_{min}. Secondly, we introduced orthogonal-users based sparsification, another distributed sparsification method, in which channel matrix entries of users located outside the considered cell have been set to zero in case their transmission were orthogonal to the one of users inside the cell. Lastly, the third and fourth centralised sparsifcation methods, based on receive antenna selection,

were introduced. The third method performs antenna selection by computing correlation values among the rows of the channel matrix related to a BS, whereas the fourth replaces correlation measures with normalised mutual information measures.
As regards the category of distributed sparsification methods, we considered three main users selection approaches, which are known users, powerful users and known users plus powerful ones. In the first method we assume the knowledge of the position of users and, in particular, whether they belong to a specific cell or not. In the second we consider only a number N_p of users that reach the BS with the highest power. The last one considers both known users and the N_p powerful ones. All distributed sparsification methods has been tested for two different pre-coding matrices, implementing a matched and a zero forcing receiver.
In the last two chapters we presented the obtained results for computational complexity and compared the different methods in terms of both computational complexity and achievable sum-rate. We showed that we can decrease the total number of operations required for the decoding process while maintaining values of achievable sum-rate similar to the ones obtained without channel sparsification. The selection of the method to be used as well as its parameter value is however a choice that depends on the SNR value and furthermore a trade-off between achievable sum-rate and computational complexity must be always performed. In fact we have seen that the best choice in achievable sum-rate does not always match the computational complexity one.

Bibliography

[1] A. Checko, H.L. Christiansen, Y. Yang, L. Scolari, G. Kardaras, M.S. Berger, L .Dittman; *Cloud RAN for Mobile Networks–A Technology Overview*, IEEE Communication survey and tutorials, Vol.17, No.1, 2015.

[2] D. Gesbert, S. Hanly, H. Huang, S.S. Shitz, O. Simeone, W. Yu; *Multi Cell MIMO Cooperative Networks: A New Look at Interference*, IEEE Journal on selected areas in communications, Vol.28, No.9, 2010.

[3] C. Fan, X. Yuan, Y.J.A. Zhang; *Scalable Uplink Signal Detection in C-RANs via Randomized Gaussian Message Passing*, 2015.

[4] A. Checko, H. Holm, H. Christiansen; *Optimizing small cell deployments by the use of C-RANs*, 20th Eur.Wireless Conf.EW, pp. 1-6.

[5] T. Nakamura et al. *Trends in small cell enhancements in LTE advanced*, IEEE communication magazine, Vol.51, no.2, pp. 98-105, Feb. 2013.

[6] A.F. Molish, M.Z. Win, Y.S. Choi, J.H. Winters; *Capacity of MIMO Systems with Antenna Selection*, IEEE Transactions on Wireless Communications, Vol.4, no.4, July 2005.

[7] F.R. Kschischang, B.J. Frey, H.A. Loeliger; *Factor Graphs and the Sum-Product Algorithm*, IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 47, NO. 2, FEBRUARY 2001

[8] J.M. Mooij, H.J. Kappen; *Sufficient Conditions for the Convergence of the Sum-Product Algorithm*, IEEE Transaction on Information Theory, Vol.53, no.12 December 2007

[9] J.W. Lee, H.N. Cho, H.J. Park, Y.H. Lee; *Sum-Rate Capacity of Correlated Multi-User MIMO Channels* School of Electrical Engineering and INMC, Seoul National University

[10] T.L. Narasmihan, A. Chockalingam; Channel Hardening-Exploiting Message Passing (CHEMP) Receiver in Large MIMO Systems IEEE WCNC '14, Track 1, 2014

[11] S. Rosati, S. Tomasin, M. Butussi, B. Rimoldi; *LLR Compression for BICM Systems Using Large Constellations* IEEE Transactions on Communications, Vol.61, No. 7, July 2013

[12] F.R. Farrokhi, K.J.R. Liu, L. Tassiulas; *Transmit Beamforming and Power Control for Cellular Wireless Systems* IEEE Journal on Selected Areas in Communications, Vol.16, No. 8, October 1998