SECURITY THROUGH FREQUENCY DIVERSITY IN THE 5G NR STANDARD

by

Joshua D. Weitz
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Electrical Engineering

Committee:

_____ Dr. Brian Mark, Thesis Director

_____ Dr. Kai Zeng, Committee Member and Co-advisor

_____ Dr. Zhi Tian, Committee Member

_____ Dr. Monson Hayes, Chairman, Department of Electrical and Computer Engineering

Date: _____ Summer Semester 2022
George Mason University
Fairfax, VA

Security Through Frequency Diversity in The 5G NR Standard

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Joshua D. Weitz
Bachelor of Science
George Mason University, 2013

Director: Dr. Brian Mark, Associate Professor
Department of Electrical and Computer Engineering

Summer Semester 2022
George Mason University
Fairfax, VA

Dedication

I dedicate this thesis to my wife Abigail, without her support this would not have been possible, and my children Charles, Arthur, and Penelope, who bring me so much joy.

Acknowledgments

I would like to thank the following people who made this possible, Dr. Brian Mark, Dr. Kai Zeng, all the faculty at Electrical and Computer Engineering department of the College of Engineering and Computing at George Mason University, who have taught me so much over the course of my higher education. I would also like to thank my employer Rincon Research Corporation for supporting my studies.

This work was supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit `cyberinitiative.org`.

This project was supported by resources provided by the Office of Research Computing at George Mason University (URL: https://orc.gmu.edu) which is funded in part by grants from the National Science Foundation (Awards Number 1625039 and 2018631).

Table of Contents

List of Tables

List of Figures

Abstract

SECURITY THROUGH FREQUENCY DIVERSITY IN THE 5G NR STANDARD

Joshua D. Weitz

George Mason University, 2022

Thesis Director: Dr. Brian Mark

This thesis explores the use of pseudo-random frequency hopping for added security in the 5G New Radio specification. Frequency hopping makes it more difficult for an attacker to intercept, detect, or jam a wireless connection in a 5G network. Current 5G resource allocation options are examined, and the state-of-the-art literature regarding Orthogonal Frequency Division Multiple Access (OFDMA) frequency hopping under various channel conditions is reviewed. Computer simulations were conducted to compare the throughput performance of the frequency hopping technique vs. static resource allocation. It is shown that under certain channel conditions and power allocation schemes, the aggregate user throughput under frequency hopping is within 95% of that of static allocation, although less under more realistic power allocations, while the probabilities of intercept and detection is significantly reduced.

Chapter 1: Introduction

1.1 Research Question

As our world becomes more connected, new technology standards are required to meet the changing needs of mobile networks. The 5G New Radio (NR) specification is the latest mobile network standard offered by the Third Generation Partnership Project (3GPP), building on the LTE standard to meet increasing bandwidth requirements and minimizing latency, while offering maximum flexibility. Future users of 5G networks will have increased security requirements as well. While the 5G NR specification offers many security options, one overlooked area is security through frequency diversity. A security conscious user may be willing to trade off throughput and network efficiency for a radio frequency (RF) signature that is harder for an attacker to detect and intercept. The purpose of this research to propose additions to the existing frequency hopping modes and resource allocation in the 5G NR standard which enhance the security aspects through frequency diversity while maintaining a usable throughput level.

Chapter 2: Background

2.1 5G NR Physical Layer

2.1.1 Overview

5G NR is designed for high speed, efficiency, and flexibility over a wide range of frequency bands, equipment constraints, and usage scenarios. Orthogonal Frequency Division Multiplexing (OFDM) was chosen as the primary modulation method to meet these demands[4]. The primary use case has a mobile User Equipment (UE) device communicating with fixed network equipment. Control and data channels are defined for both the uplink and the downlink in this use case[4].

2.1.2 Modulation

In 5G NR, a variety of modulation schemes are available for the underlying symbols in the OFDMA, including QPSK, 16QAM, 64QAM, and 256QAM.[4] The symbol modulation is chosen depending on the signal power level and the required throughput. These symbols are grouped together into an OFDMA sequence. The parameters of this OFDMA sequence are configurable and can differ for each UE. Subcarriers have a base spacing of 15 KHz$*2^n$, up to 240 KHz. This 2^n scaling factor is base on the numerology n, which defines the subcarrier spacing and the symbol width.[1]

For any given numerology, a slot is defined as 14 symbols. A frame is defined as 10 ms of transmitted symbols. A frame is further divided into 10 subframes of 1 ms duration, each of which has 2^n slots. The 5G NR specification allows a transmission to start on any

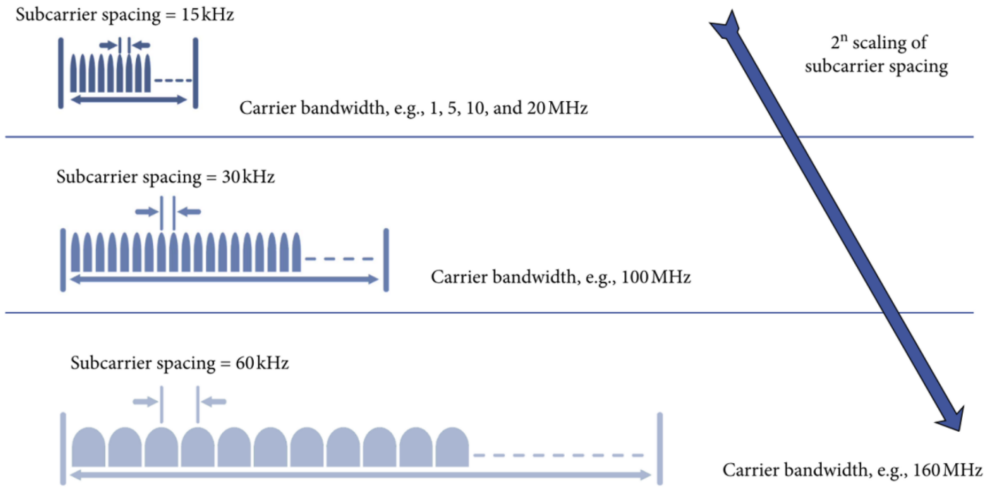

Figure 2.1: 5G Numerology Bandwidth Changes

OFDM symbol and last for as many symbols as are required. Tables 2.1, 2.2, and 2.3 show the key 5G NR modulation specifications and parameters.

2.1.3 Channels

The 5G NR specification defines the physical channels used in NR implementations[4].

- Physical Broadcast Channel (PBCH) is used to transmit synchronization signals from the network to the UE as well as provide initial control information to the UE.

Table 2.1: Numerologies in 5G NR [1]

Numerology(n)	0	1	2	3	4
Subcarrier Spacing (KHz)	15	30	60	120	240
OFDMA Symbol Duration (μs)	66.67	33.33	16.67	8.33	4.17
Cyclic Prefix Duration (μs)	4.69	2.34	1.17	0.57	0.29
OFDMA Symbol including Cyclic Prefix Duration (μs)	71.35	35.68	17.84	8.92	4.46

Table 2.2: Numerology Structures in 5G NR [1]

Numerology(n)	0	1	2	3	4
Subcarrier Spacing (KHz)	15	30	60	120	240
Slot Duration (ms) $1ms/2^n$	1	0.5	0.25	0.125	0.0625
Number of slots/subframe	1	2	4	8	16
Number of slots/frame	10	20	40	80	160
FFT Size (Max)	4096	4096	4096	4096	4096

Table 2.3: Parameters for Different Frequency Ranges in 5G NR [1]

Frequency Range Designation	FR1 (410 - 7125 MHZ)	FR2 (24250 - 52600 MHZ)
Bandwidth per Carrier (MHz)	5,10,15,20,25 50,60,70,80,90,100	50,100,200,500
Subcarrier Spacing (KHz)	15,40,60	60,120,240
Maximum Number of Subcarriers	3300 (FFT 4096)	
Carrier Aggregation	Up to 16 carriers	
Modulation Schemes	QPSK, 16QAM, 64QAM, 256QAM, $\pi/2$BPSK on uplink	

- Physical Downlink Control Channel (PDCCH) is used to transmit control and configuration information for both the downlink and the uplink from the network to the UE.
- Physical Downlink Shared Channel (PDSCH) is used to transmit data from the network to the UE.
- Physical Random Access Channel (PRACH) is used for channel access information from the UE to the network.
- Physical Uplink Control Channel (PUCCH) is used to transmit control and configuration information for both the downlink and the uplink from the UE to the network.
- Physical Uplink Shared Channel (PUSCH) is used to transmit data from the UE to the network.

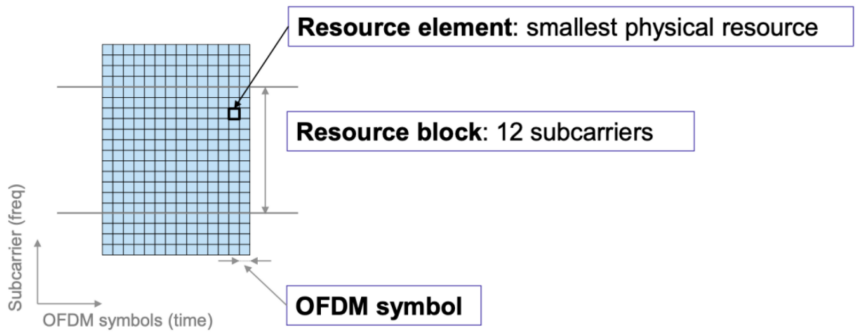

Figure 2.2: 5G Physical Resource Blocks

2.1.4 Resources

The channels are allocated for use in sets of subcarriers are assigned in blocks of 12 called Physical Resource Blocks (PRB).[1]. Each PRB can be as short as one OFDM symbol.

2.1.5 Resource Allocation

The 5G NR specification defines the concept of a Bandwidth Part (BWP). A BWP is a contiguous set of resource blocks inside a channel[2]. Bandwidth parts are used as the first step in resource allocation on the PDSCH and the PUSCH. Each UE is assigned at least one BWP by the network. More can be assigned, but only one is allowed to be active at a time. Not every resource in the active BWP is used by the UE, further allocation specifies exactly which Resource Block (RB) at which time slot is allocated to a given UE.[4]

PDSCH Allocation

Control messages on the PDCCH inform the UE exactly which RB in which slots the network will use to transmit data on the PDSCH to the UE. The network sends a `PDSCH-Config` message, which contains both the time domain and the frequency domain allocations for

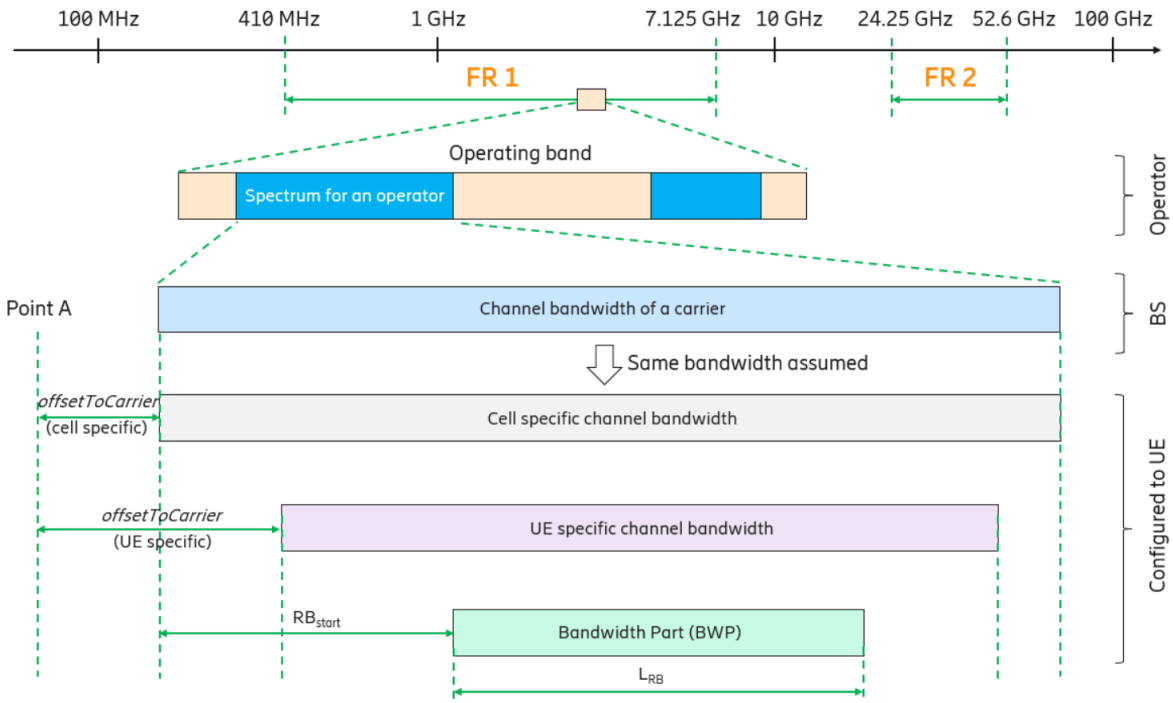

Figure 2.3: 5G Bandwidth Part Concept [2]

upcoming transmissions[4]. The `PDSCH-TimeDomainResourceAllocation` section is defined as[4]

```
PDSCH-TimeDomainResourceAllocation ::= SEQUENCE {
    k0 INTEGER(0..32) OPTIONAL, -- Need S
    mappingType ENUMERATED {typeA, typeB},
    startSymbolAndLength INTEGER (0..127) // SLIV
}

PDSCH-TimeDomainResourceAllocation-r16 ::= SEQUENCE {
    k0-r16 INTEGER(0..32) OPTIONAL, -- Need S
    mappingType-r16 ENUMERATED {typeA, typeB},
    startSymbolAndLength-r16 INTEGER (0..127),
    repetitionNumber-r16 ENUMERATED {n2, n3, n4, n5, n6, n7, n8, n16}
        ↪ OPTIONAL,
                                    -- Cond Formats1-0and1-1
...
}
```

The `startSymbolAndLength` parameter is used to by the UE to know which symbol in a slot is allocated, while K_0 determines which slot the allocation is for. Repetition allows the same symbols to be allocated for multiple slots. Frequency domain allocation is indicated in the `RateMatchPattern` section, which is defined as

```
RateMatchPattern ::= SEQUENCE {
    rateMatchPatternId RateMatchPatternId,
    patternType CHOICE {
        bitmaps SEQUENCE {
            resourceBlocks BIT STRING (SIZE (275)),
            symbolsInResourceBlock CHOICE {
```

```
            oneSlot BIT STRING (SIZE (14)),
            twoSlots BIT STRING (SIZE (28))
        },
        periodicityAndPattern CHOICE {
            n2 BIT STRING (SIZE (2)),
            n4 BIT STRING (SIZE (4)),
            n5 BIT STRING (SIZE (5)),
            n8 BIT STRING (SIZE (8)),
            n10 BIT STRING (SIZE (10)),
            n20 BIT STRING (SIZE (20)),
            n40 BIT STRING (SIZE (40))
    } OPTIONAL, -- Need S
    ...
  },
  controlResourceSet ControlResourceSetId
},
subcarrierSpacing SubcarrierSpacing OPTIONAL, -- Cond CellLevel
dummy ENUMERATED { dynamic, semiStatic },
...,
[[
  controlResourceSet-r16 ControlResourceSetId-r16 OPTIONAL -- Need R
]]
}
```

The `resourceBlocks` bitmap parameter allows the network to set the resource blocks in the most flexible manner possible.

PUSCH Allocation

Control messages on the PUCCH inform the UE exactly which RB in which slots the UE will use to transmit data on the PUSCH to the network. After the UE registers with the networt, the network sends a `UplinkGrantConfig` message, which contains both the time domain and the frequency domain allocations for upcoming transmissions[4]. The `rrc-ConfiguredUplinkGrant` section is defined as[4]

```
rrc-ConfiguredUplinkGrant SEQUENCE {
    timeDomainOffset INTEGER (0..5119),
    timeDomainAllocation INTEGER (0..15),
    frequencyDomainAllocation BIT STRING (SIZE(18)),
    antennaPort INTEGER (0..31),
    dmrs-SeqInitialization INTEGER (0..1) OPTIONAL, -- Need R
    precodingAndNumberOfLayers INTEGER (0..63),
    srs-ResourceIndicator INTEGER (0..15) OPTIONAL, -- Need R
    mcsAndTBS INTEGER (0..31),
    frequencyHoppingOffset INTEGER (1.. maxNrofPhysicalResourceBlocks-1)
        ↪ OPTIONAL, -- Need R
    pathlossReferenceIndex INTEGER (0..maxNrofPUSCH-PathlossReferenceRSs-1)
        ↪ ,
    ...,
}
```

The `timeDomainOffset` and `timeDomainAllocation` parameters are used to by the UE to know which symbol in a slot is allocated, and the `frequencyDomainAllocation` bitstring parameter defines which resource blocks are allocated for transmission by the UE.

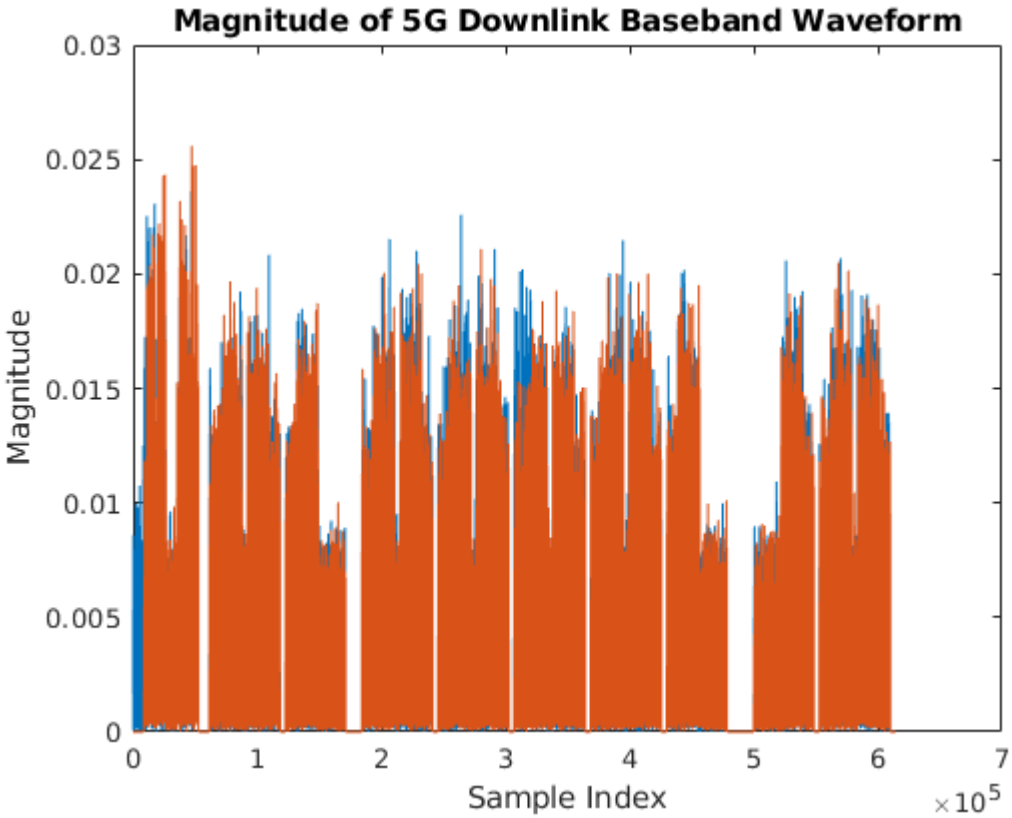

Figure 2.4: MATLAB Example 5G Downlink Waveform Magnitude

2.2 MATLAB 5G Toolbox

Mathworks has created a simulation and testing toolbox for 5G NR as an add on on to MATLAB[5]. This toolbox can generate uplink and downlink waveforms, including PDCCH, PDSCH, PUCCH, PUSCH, PRACH, and PBCH. In addition it has tools for link level simulation, tests and measurement, testing search procedures, and more. This is an excellent resource for testing current specification compliant 5G implementations. Figures 2.4 and 2.5 show the magnitude and spectrogram of a downlink signal including PDCCH and PDSCH generated with the MATLAB 5G toolbox.

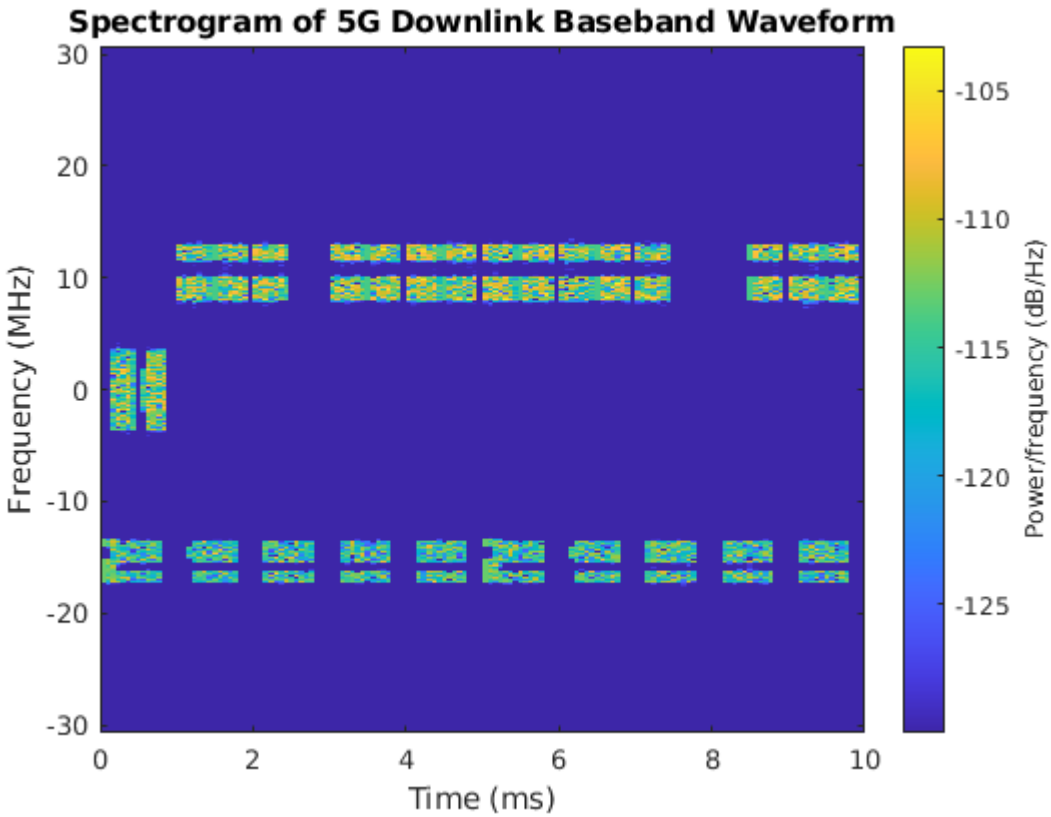

Figure 2.5: MATLAB Example 5G Downlink Waveform Spectorgram

Chapter 3: Literature Review

3.1 Frequency Hopping Performance

Although the specific question of efficient frequency hopping on the 5G NR standard has not been studied, closely related research was found.

Deng et al.(2011) [6] studied the combined effect of transmit diversity and frequency hopping for DFT-precoded OFDMA in uplink frequency-selective fading channels and showed the relative efficiency of of FH-OFDMA in fading channels. They showed frequency hopping is a good choice for maintaining throughput and investigated the potential of open loop (i.e., no channel information feedback) frequency hopping compared to closed loop channel selection. The frequency hopping model employed in the open loop design is as follows: After FFT and symbol mapping, a vector $\mathbf{s}^{Txb,(l)}$ is obtained for the lth FFT block. This vector is multiplied by a matrix $\mathbf{Q}_{N_{FFT}\times M}$, where M is the number of data modulated symbols in an FFT block. For FH, $\mathbf{Q}$ is constructed as follows:

$$\mathbf{Q}_{N_{FFT}\times M} = \begin{pmatrix} \mathbf{0}_{q\times M} \\ \mathbf{I}_M \\ \mathbf{0}_{(N_{FFT}-q-M)\times M} \end{pmatrix} \text{when } l < N_{sbf}/2$$

$$\mathbf{Q}_{N_{FFT}\times M} = \begin{pmatrix} \mathbf{0}_{[(q+N_{FH})modN_{FFT}]\times M} \\ \mathbf{I}_M \\ \mathbf{0}_{[(N_{FFT}-q-N_{FH}-M)modN_{FFT}]\times M} \end{pmatrix} \text{when } l \geq N_{sbf}/2$$

where $q \in \{0, ..., (N_{FFT} - M - N_{FH} - 1)\}$ is the first subcarrier index in the M subcarrier system assigned and N_{FH} defines the number of subcarriers to hop forward each block. This will generate a cyclic hopping scheme.

Oturak et al. (2013)[7] studied the performance of fast frequency hopping OFDM under Doppler spread. They showed that in a fading channel, FH-OFDMA and traditional OFDMA perform the same, but there is a limit at which the bit error rate no longer reduces even as signal power increases. This is due to the Doppler fading. The frequency hopping model employed in this paper is also cyclic. After FFT and symbol mapping, a vector $\mathbf{d}$ is obtained for the each FFT block. This vector is multiplied by a matrix $\mathbf{P}_{N_C}$, where N_C is the number of subcarriers. $\mathbf{P}_{N_C}$ has the elements $e^{j2\pi(k-1)(r-1)/N_C}/\sqrt{N_C}$ for the kth row and rth column. The non-hopped OFDM symbols can be found by

$$\mathbf{s_{classic}} = \mathbf{P}_{N_C}\mathbf{d}$$

If instead, $\mathbf{P}_{N_C H}$ is constructed such that

$$\mathbf{P}_{N_C H} = \mathbf{P}_{N_C}\mathbf{C}_{N_C H}$$

where the elements of $\mathbf{P}_{N_C}\mathbf{C}_{N_C H}$ are $e^{j2\pi(k-1)\Psi_{N_C}/N_C}/\sqrt{N_C}$ and $\Psi_{N_C} = \mod(f_k + r - 1, N_C)$, for $k, r = 1, ..., N_C$. The frequency hopping pattern can be selected by generating a vector $\mathbf{f}$ of length N_C, in which each f_k represents the subcarrier index each symbol will be assigned to. $\mathbf{f}$ is a randomly shuffled version of $\mathbf{l} = (0, ..., N_C)$.

Zhou and Giannakis (2000)[8] studied generalized frequency hopping OFDMA through unknown frequency-selective multi-path channels. In their paper they model OFDMA as a CDMA process with a spreading code

$$c_m(k; n) = e^{-j(\frac{2\pi}{M}m + \theta_k)(P-1-n)}$$

where M is the number of users, $m \in \{1, M\}$, unique for each user, P is the code length k is the symbol index, and $n \in \{0, P-1\}$ is the chip index. If $\theta_k = 0$, this becomes OFDMA, and if θ_k is time varying, then it becomes a FH-OFDMA sequence. They then go on to use a cyclic sequence, $\theta_k = (k \bmod M)\frac{2\pi}{M}$. They showed that under frequency selective multi-path channels, GFH-OFMDA can perform better than OFDMA.

Jung and Lim (2011) [3] studied chaotic map based frequency hopping OFMDA for low probability of intercept. They proposed a scrambling method which allows multiple transmitted symbols from the same user to be in the same time slot, or none at all, with the order of the symbols scrambled as well. The scrambling technique is called a Chaotic Standard Map (CSM). It starts by allocating a seed matrix with a symbol assigned to every user in every time slot, then permutes the seed matrix to come up with the CSM matrix (see Fig. 3.1. This poses an additional burden on an observer to correctly reorder the symbols before they can be decoded, if than can be detected at all.

3.2 LPI/LPD

In order to characterize the security of our proposed frequency hopping scheme, it is first necessary to have a qualitative measurement for the related terms Low Probability of Intercept (LPI) and Low Probability of Detection (LPD). Low probability of detection is the idea of transmitting in such a way that there is a low probability an observer will detect that a transmission has occurred[9]. Low probability of intercept is the idea of transmitting in such a way that there is a low probability an observer can capture the signal and analyze to recover the transmitted data[10]. Once these measurement methods have been established, our proposed frequency hopping method can be compared to the non-hopping method and the relative gain in security can be ascertained. The question of security can be divided into two parts. Does the observer know that a transmission is occurring? A detection probability metric is defined with the goal of minimizing the probability of detection. Secondly, can the observer demodulate the symbols to recover the message contents?

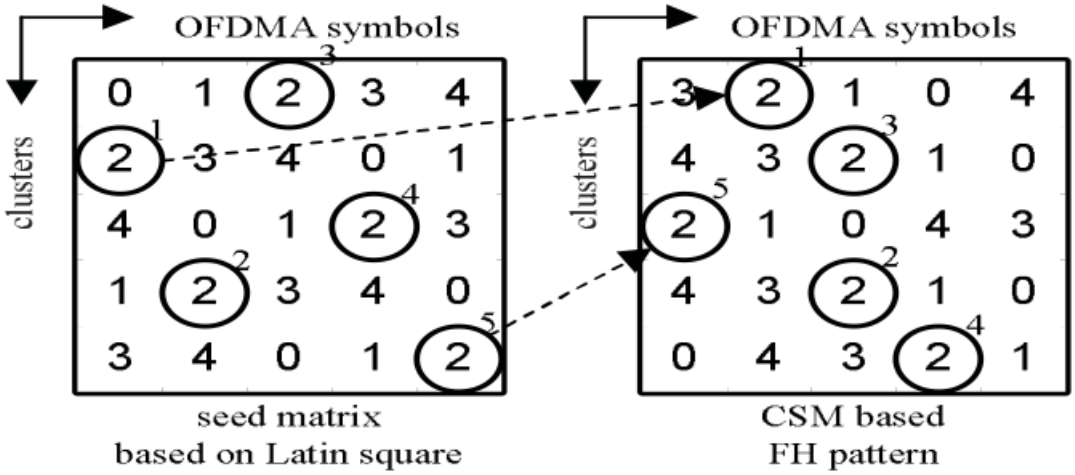

Figure 3.1: Chaotic Standard Map[3]

An interception probability metric is defined, once again with the goal of minimizing the probability of interception.

3.2.1 LPD Metrics

Many papers that propose an LPD metric for a specific signal and signal conditions end up referring back to "Energy Detection of Unknown Deterministic Signals"[11] written by Harry Urkowitz in 1967. He proposes a generic method for measuring the probability of detection for a band limited signal in AWGN. First, a test (decision) statistic V' is proposed such that,

$$V' = (1/N_{02}) \int_0^T y^2(t)\, dt$$

where N_{02} is the two sided noise power spectral density, T is the observation time, and $y(t)$ is the observed signal + noise. A threshold value V'_T is set and a detection is determined

when $V' > V'_T$. Two cases are considered. When the signal is not present, a false alarm probability is calculated:

$$Q_0 = \frac{1}{\sqrt{8\pi TW}} \int_{V'_T}^{\infty} e^{-\frac{(x-2TW)^2}{8TW}} dx$$

$$Q_0 = \frac{1}{2}\text{erfc}\left[\frac{V'_T - 2TW}{2\sqrt{2}\sqrt{TW}}\right]$$

where W is the bandwidth under observation. When the signal is present, the detection probability is given by

$$Q_D = \frac{1}{2}\text{erfc}\left[\frac{V'_T - 2TW - \lambda}{2\sqrt{2}\sqrt{TW} + \lambda}\right]$$

where

$$\lambda = \frac{E_s}{N_{02}}$$

and E_s is the signal energy in the observed bandwidth.

3.2.2 LPI Metrics

Calculating an intercept probability is highly dependent on the characteristics of the signal under observation. One method is defined in "Chaotic Standard Map Based Frequency Hopping OFDMA for Low Probability of Intercept by Junwoo Jung and Jaesung Lim" [3]. For their chaotic frequency hopping method, they have worked out an intercept probability metric already. They define three different frequency hopping techniques and use combinatoric methods to compute the probability of intercept. In order for the observer to successfully intercept the message, the observer must be able to detect the exact frequency hopping pattern a specific user uses. They assume that the observer knows the exact bandwidth, center frequency, and FFT size used in the OFDMA system. In a symbol by symbol

hopping strategy, such a latin squares method, where there is one symbol per user per time slot, the intercept probability for K users with N FFT size is[3]

$$P_I^{SBS} = \begin{cases} \frac{1}{N(N-1)^{N-1}} & K = N \\ \frac{1}{K^N} & K < N \end{cases}$$

If instead, a cat map based pattern is used, where the there is one symbol per user per time slot, but the transmission order is mixed, then the intercept probability is[3]

$$P_I^{CAT} = \begin{cases} \frac{1}{N(N-1)^{N-1}N!} & K = N \\ \frac{1}{K^N N!} & K < N \end{cases}$$

Finally, if the chaotic standard map is used, where there can be multiple or no symbols per user per time slot, and the transmission order is mixed, the intercept probability is[3]

$$P_I^{CSM} = \frac{1}{(NK)!/(NK-N)!}$$

3.3 PDCCH Security

Gardner and Roth (2020)[12] studied an efficient method for descrambling the control channel. The PDCCH is already vulnerable to a brute force descrambling attack. The scrambling is set by the cell-radio network temporary identifier (C-RNTI), which is assigned to the UE by the network, and may vary with time. The C-RNTI is 16 bits long, so an observer could decode any given message on the PDCCH by trying each possible C-RNTI and checking the CRC on the message for validity. Gardner and Roth showed that by exploiting the properties of the polar encoding forward error correction that is applied to the PDCCH, this descrambling could be accomplished with much more efficiency. Once the C-RNTI is

recovered, an observer can view all the PDCCH assignments to that user, and would be aware of the resource blocks assigned for upcoming transmissions.

Chapter 4: Proposed Frequency Hopping Resource Allocation

4.1 Chaotic Standard Map OFMDA Allocation

With the goal of adding security to the 5G NR signal by adding frequency hopping, the choice of hopping pattern becomes critical. Following the work done by Manuel Jiménez Martínez in his research report titled "5G LPI Communications"[1], the frequency hopping pattern chosen for this research is a Chaotic Standard Map (CSM) based pattern. Jung and Lim (2011) [3] studied CSM patterns for resource allocation in an OFDMA system to enhance the LPI characteristics. The CSM method allows the transmitted symbols to be mixed up in order, as well as allows multiple, or no, symbols to be sent in a given time slot. This greatly increases the work of an observer to intercept the transmitted message. The proposed frequency hopping scheme is a four step process[3].

1. "Choose a prime number N, where N is the number of available OFDMA resource blocks"[3]. N becomes the maximum number of available users as well.

2. "Construct a $N \times N$ Latin square seed matrix"[3]. This will have a resource block in each time slot allocated to each user. The method of construction is

$$L_\alpha(i,j) = (\alpha i + j) \mod N, \; i,j = 0,...,N-1$$

   "where α is a pattern index, $(\alpha = 2,...,N-1)$ and i and j are the column and row indices of the seed matrix"[3].

3. "Generate the FH pattern S_α^k for a given user k for a period N points long composed of consecutive points (i,j) for which $L_\alpha(i,j) = k$[3]. The clusters $s_\alpha^k(j)$ occupied by

pattern S_α^k at column j satisfy,"[3]

$$s_\alpha^k(j) = ((k-j)\alpha^{-1}) \mod N \; j = 0, ..., N-1.$$

4. "Convert the FH pattern S_α^k into the CSM based FH pattern C_α^k , consisting of N consecutive points $(c_{\alpha,x}^k(j), c_{\alpha,y}^k(j))$, $j = 0, ..., N-1$ C_α^k is,

$$\begin{cases} c_{\alpha,x}^k(j) = (s_\alpha^k(j) + j + r_x + r_y) \mod N \\ c_{\alpha,y}^k(j) = (j + r_y + [K_C \sin(\frac{c_{\alpha,x}^k(j)N}{2\pi})]) \mod N \end{cases}$$

where $c_{\alpha,x}^k(j)$ and $c_{\alpha,y}^k(j)$ denote row and column of the CSM matrix, the same as the cluster and symbol indexes for user k, respectively. $r_x$and r_y are integers varying from 0 to $N-1$, and K_C is a positive integer. $[x]$ is the nearest integer to x."[3]

Figure 3.1 shows an example mapping from L_α to C_α.

4.2 Chaotic Standard Map Allocation in 5G

In this research, we propose methods for using CSM based hopping patterns for the PUSCH and the PDSCH. The primary unit of frequency allocation in 5G is a resource block. Each block consists of twelve contiguous subcarriers. Thus the frequency allocation unit of the CSM matrix will be defined as a resource block. Ideally, the time allocation unit of the CSM matrix would be a symbol, as this would give the finest granularity for transmission. This could be challenging for some existing hardware, as the current specification allows hops either on a per slot basis, or at most one time midway through a slot, not on a symbol by symbol basis.

4.2.1 Slot Based Hopping

Slot based hopping could be achieved within the current 5G standard without any modification the the specification. The change would be in the allocation scheme at the network, where r_x and r_y would be pre-selected. When a user with this frequency hopping mode connects, the network would assign a bandwidth part that consists of a prime number N of resource blocks. Then, for each N slots, the allocation is as follows:

1. For the upcoming N slots, K_C is chosen, and a C_α matrix for CSM allocation is generated. Depending on the required throughput for a user, a set of user indices k would be assigned to the connected user.

2. The allocation resource blocks for each slot for each of the k user indices would be determined, creating the resource block allocation for each slot.

3. Both the PDSCH and PUSCH control messages would be used to communicate the upcoming N slot resource block allocations to the user.

4. After the N slots are complete, the next N slots can be computed, changing K_C to change the frequency pattern so that it is non-repeating.

Multiple users could be accommodated in the same bandwidth part, with different sets of k generating different allocations for each uses to avoid overlapping. The downside of this approach is the greatly increased traffic on the control channels. Since there is no longer slot repetition in the allocation, a control message detailing the resource block allocation for every slot would have to be sent for both the PDSCH and the PUSCH for every user with this mode. The allocation in the control messages would also be vulnerable if an observer could descramble the control channel and acquire the allocation, thereby losing the LPI gains which the allocation method is supposed to give.

4.2.2 Control Channel Updates

In order to maintain security in the control channel, and to keep overhead lower, the PDSCH and the PUSCH control messages could be modified to communicate the CSM parameters

and assigned user index to a UE, and then both the network and the UE could independently generate the hopping sequence for the upcoming N time periods. α, r_x, and r_y will be chosen ahead of time programmed into the network and the UE. This will not be shared over the air to avoid security vulnerabilities. In addition, an offset for K_C called K'_C will be chosen ahead of time and programmed into both the network and the UE With these parameters pre-configured, the changes to the grant config PUSCH message are as follows.

```
frequencyHopping ENUMERATED {intraSlot, interSlot}
```

will become

```
frequencyHopping ENUMERATED {intraSlot, interSlot, CSM}
```

When in CSM mode,

```
frequencyHoppingOffset
```

will be set to such that $K_C = K'_C + \text{frequencyHoppingOffset}$ for the K_C which the network used to generate the CSM table. With α, r_x, and r_y pre-shared, the UE can now calculate the same table as the network.

In addition, in the CSM hopping mode

```
timeDomainOffset
```

will be repurposed as the cluster index of the CSM map. Each set of clusters assigned to a specific UE for a hopping period will be sent in the grant config message. The UE now knows which resource clusters to transmit on for the next N OFDMA symbols. A new grant must be received before N OFDMA slot periods pass.

When the Radio Resource Controller (RRC) at the network is deciding how to allocate the channels, it will first allocate all the connected authorized CSM users, then take the

remaining channels and assign them to the rest of the users in the system by whatever method it normally would. PDSCH config messages would be updated in a like manner, with a different α, r_x, r_y, and K'_C.

These changes would greatly reduce the overhead in scheduling CSM frequency hopping sequences and maintain security in the allocation information on the control channel.

4.2.3 Symbol Based Hopping

Symbol based hopping could be achieved with the proposed control channel updates, instead of allocating each slot with a column in the CSM matrix, each column could represent a symbol. This uses more control channel overhead, since control updates would need to be sent more frequently, but would provide the most enhancement to the LPI properties of the transmission.

4.2.4 Detection Probabilities For A Given Resource Block

A key measure of improvement to security is probability of detection. In their study of CSM based FH-OFDMA patterns, Jung et al.[13] calculate the detection probability for various frequency hopping patterns. They define the probability of detection as the probability that an observer can both tell that a transmission from a specific user is occurring, and recover all the transmitted symbols. This is analyzed by hypothesis testing. Assume the observer sees M symbols, where M is a multiple of the number of symbols in a frame, T_d is the symbol duration, and T_c is the cyclic prefix duration. Under h_0, the null hypothesis, a user is not present in the system, and under h_1 a user is present.

$$\begin{cases} h_0: & y(t) = w(t) \\ h_1: & y(t) = s(t) + w(t) \end{cases}$$

where $y(t)$ is the received signal, $w(t)$ is additive white Gaussian noise (AWGN) with zero mean and variance σ_w^2 and $s(t)$ is the FH-OFDMA signal. With a sufficiently large IFFT size, under the central limit theorem $s(t)$ is also Gaussian with zero mean and variance σ_s^2, therefore

$$\begin{cases} h_0: & y(t) \sim \mathcal{N}_c(0, \sigma_w^2) \\ h_1: & y(t) \sim \mathcal{N}_c(0, \sigma_w^2 + \sigma_s^2) \end{cases}$$

Under these two hypotheses, the auto-correlation coefficients for delay $\tau = \pm T_d$ are

$$\begin{cases} h_0: & \rho(\pm T_d) = 0 \\ h_1: & \rho(\pm T_d) = \rho_D \end{cases}$$

where

$$\rho_D = \frac{T_c}{T_d + T_c} \frac{\sigma_s^2}{\sigma_w^2 + \sigma_s^2} = \frac{T_c}{T_d + T_c} \frac{\Gamma}{1 + \Gamma}$$

where Γ is the signal to noise ratio σ_s^2/σ_w^2 with real valued $\rho(\tau)$, assuming circular symmetry, and maximum likelihood detection estimates,

$$\begin{cases} h_0: & y(t) \sim \mathcal{N}_r(0, \frac{1}{2M}) \\ h_1: & y(t) \sim \mathcal{N}_r(\rho_D, \frac{(1-\rho_D^2)^2}{2M}) \end{cases}$$

Using a Neyman-Pearson detection strategy to satisfy the false alarm constraint, for a generic Gaussian random variable $R \sim \mathcal{N}_r(\mu_r, \sigma_r^2)$

$$P(R > \eta_r) = \frac{1}{2} \text{erfc} \left(\frac{\eta_r - \mu_r}{\sqrt{2}\sigma_r} \right)$$

Therefore, the false alarm rate is

$$P_{fa} = (\hat{\rho}_{ML} > \eta_l | h_0) = \frac{1}{2}\text{erfc}\left(\sqrt{M}\eta_l\right)$$

and the detection threshold can be chosen to suit a false alarm rate

$$\eta_l = \frac{1}{\sqrt{M}}\text{erfc}^{-1}(2P_{fa})$$

The overall detection probability then is[13]

$$P_d^{FT} = P_I^{FH} P(\hat{\rho}_{ML} > \eta_l | h_1) = P_I^{FH}\frac{1}{2}\text{erfc}\left(\sqrt{M}\frac{\eta_l - \rho_D}{1-\rho_D^2}\right)$$

where P_I^{FH} is the intercept probability.

4.2.5 Pattern Interception Probabilities

Jung and Lim [3] calculated the probability that an observer of the entire hopping bandwidth could reassemble a given user's transmission pattern as

$$P_I^{CSM_{sym}} = \frac{1}{(NK)!/(NK-N)!}$$

where N is the number of resource blocks in the hopping back and $K, K \leq N$ is the number of hopping patterns in use. Assuming this as the pattern intercept probability for the symbol based hopping case, in the slot based hopping case, with the same observation period, N is divided by the number of symbols S in a slot, and the pattern intercept probability becomes

$$P_I^{CSM_{slot}} = \frac{1}{((N/S)K)!/((N/S)K-(N/S))!}$$

Symbol based hopping provides the most improvement to the pattern intercept probability. However as an intermediate step slot based hopping is still better than the non-hopping transmission, which, following the same combinatorial logic has the pattern intercept probability of

$$P_I^{Fixed} = \frac{1}{K}$$

4.2.6 Overall Intercept Probability

Given these intercept probabilities, the overall intercrpt probability for the CSM hopping case is

$$P_d^{FT} = \frac{1}{(NK)!/(NK-N)!}\frac{1}{2}\text{erfc}\left(\sqrt{M}\frac{\eta_l - \rho_D}{1-\rho_D^2}\right)$$

and the overall detection probability for the non-hopping case is

$$P_d^{FT} = \frac{1}{K}\frac{1}{2}\text{erfc}\left(\sqrt{M}\frac{\eta_l - \rho_D}{1-\rho_D^2}\right)$$

CSM frequency hopping then presents a significant improvement in overall intercept probability over non-hopping based methods.

Chapter 5: CSM OFDMA Simulation

5.1 Pattern Generation and Throughput Analysis

Our simulation work started with code that Manuel Jiménez Martínez wrote for his George Mason University ECE 798 Research Report Project titled"5G LPI Communications"[1]. The original code was a MATLAB user interface. As part of this project, the code was separated into functions and modifications and additions were made to support the throughput analysis for this research. `chaoticmap.m` (see appendix B.1) is the code used to generate the CSM allocation matrix and a more traditional allocation matrix which assigns a channel to each user for the same amount of time as the CSM allocation matrix, based on the best available Channel State Information (CSI) in the first time slot.

The CSI is generated with `csg_gen.m` (see appendix B.2). The channel model is a first order autoregressive time varying fading channel. The initial CSI for a subcarrier CSI_n, $n = 0$ is calculated as

$$CSI_0 = \frac{|h_0|^2}{\sigma^2}$$

where h_0 is a Gaussian random variable with zero mean and variance σ^2. Each subsequent CSI value for a channel is calculated as

$$CSI_n = e^{-d_s/d_0} \cdot CSI_{n-1} + g_n$$

where g_n is a zero mean Gaussian random variable with variance $(1 - (e^{-d_s/d_0})^2) \cdot \sigma^2$, d_s is the distance moved between each CSI sample, and d_0 is the decay factor. This CSI simulates a moving UE relative to the network, and different movement rates can be calculated.

To obtain the maximum throughput for each channel during the simulation, power is allocated to each subcarrier using a water-filling optimization method, in `waterfilling.m` ↪ (see appendix B.3). The method to maximize the throughput is as follows taken from Martínez's paper[1]:

$$C_{NC} := \max_{P_n} \sum_{n=0}^{N_c-1} \log_2(1 + P_n c_n)(\frac{\text{bps}}{\text{Hz}})$$

such that

$$\sum_{n=0}^{N_c-1} P_n = P_{tot},\ P_n \geq 0$$

where C_{NC} is the maximized throughput, P_n is the power allocated to each subcarrier, N_C is the number of subcarriers, P_{tot} is the total power per channel, and c_n is the CSI for that subcarrier. To solve this problem, the Lagrangian is used:

$$\mathcal{L}(P_0, ..., P_{N_c-1}, \lambda_1, ..., \lambda_{M_C-1}) = \sum_{n=0}^{N_c-1} \log_2(1 + P_n c_n) - \sum_{n=0}^{N_c-1} \lambda_n P_n - \mu \sum_{n=0}^{N_c-1} P_n - P_{tot}$$

Solving for:

$$\frac{d\mathcal{L}}{dP_n} = 0$$

Then the optimal power is:

$$P_n^* = \frac{1}{\lambda_n + \mu} - \frac{1}{c_n} \geq 0$$

Setting λ_n to 0:

$$P_n^* = \left(\frac{1}{\mu} - \frac{1}{c_n}\right)^+$$

To find the optimal power in the MATLAB code, for a range of μ from 0 to $\max c_n$, P_n is calculated for a given user and time slot CSI value. Then, using this P_n vector and the CSI, $\mathcal{L}(P_0, ..., P_{N_c-1}, \lambda_1, ..., \lambda_{M_C-1})$ is calculated. The minimum is found, and the value of μ at that point gives P_n^*. This must be repeated for each of the N^2 allocation slots if the ideal power is to be calculated each time. As N increases this becomes an extremely computationally intensive process.

All these parts are brought together in `primes_test.m` (see appendix B.4), which was a test of generating a series of CSM allocations for an increasing N. For a given prime number of resource blocks N, CSI is generated, CSM allocation, static allocation, and frequency hopping with the best available channel chosen via user priority allocation are generated. Power allocation is simulated twice for each frequency allocation. The first case is power allocation on the first time slot, with this allocation being used for all the simulated N time slots. The second is an updated power allocation in each time slot. Throughput is calculated for all six allocation cases, then compared. For the CSM allocation, K_c, α, r_x, and r_y are randomly selected for each simulation, within the specified boundaries. Listing 5.1 shows an example test run for $N = 3, 5, 7, 11, 13, 17, 19, 23$.

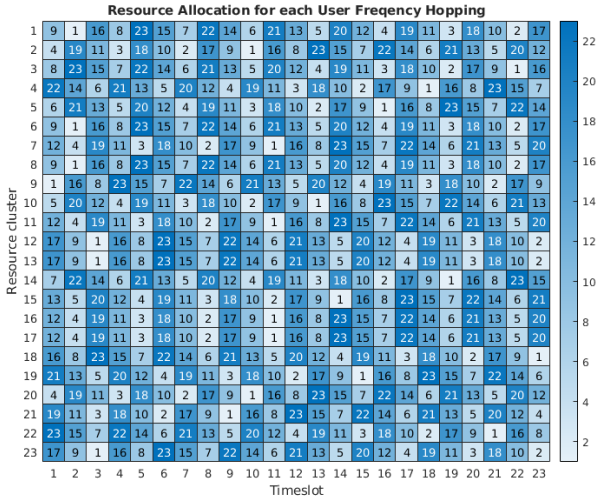

Figure 5.1: CSM Resource Block Allocation

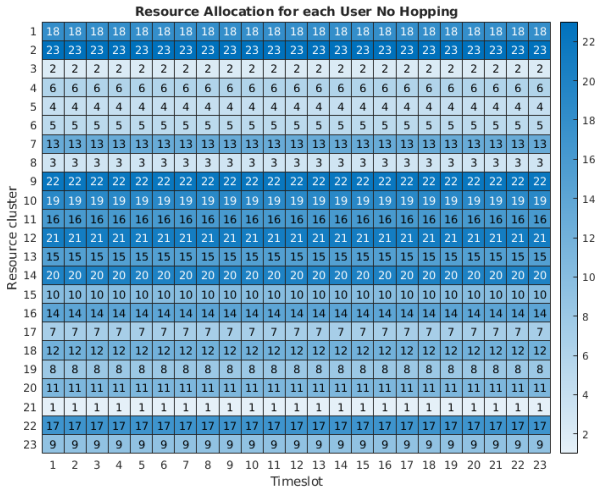

Figure 5.2: Standard Resource Block Allocation

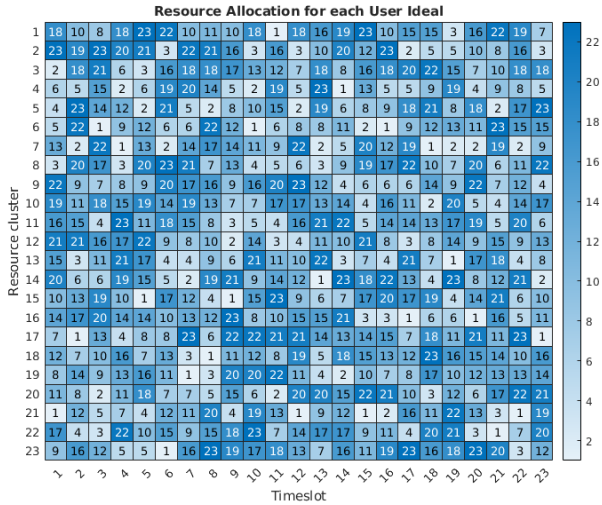

Figure 5.3: Ideal FH Resource Block Allocation

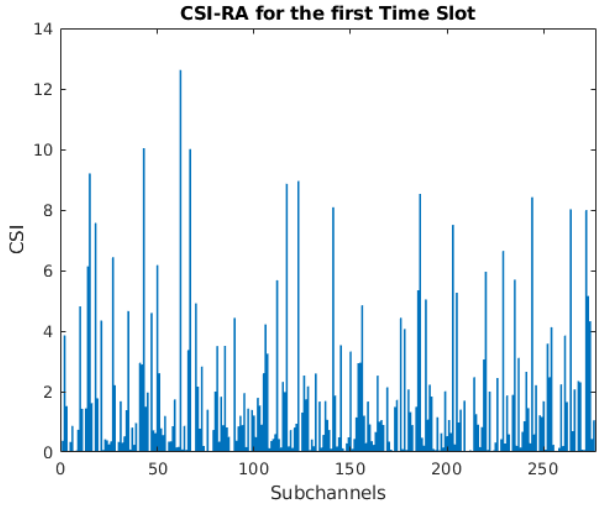

Figure 5.4: First Slot CSI

Figure 5.5: Power Allocation

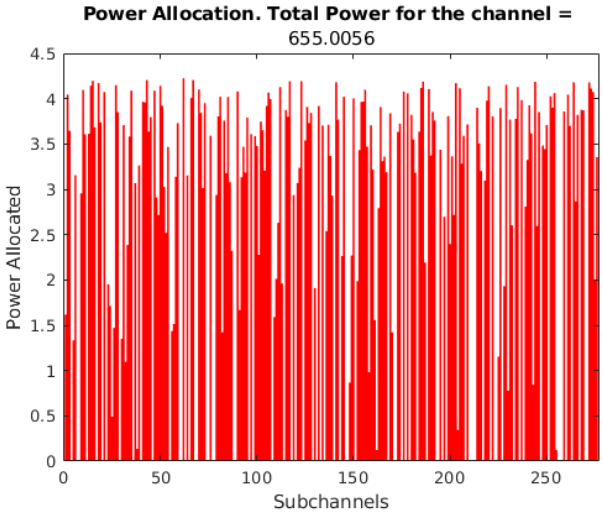

Figure 5.6: Power Allocation

Listing 5.1: Example Test Output

```
RBs 1 Nclu 3 alpha 2 rx 2 ry 2 Kc 4500201
FH average throughput per slot 80.110499 bps/Hz
Non FH average throughput per slot 117.051647 bps/Hz
Ideal FH average throughput per slot 100.080215 bps/Hz
Ideal FH Continous Power average throughput per slot 111.151855 bps/Hz
Non FH Continous Power average throughput per slot 117.051647 bps/Hz
FH Continous Power average throughput per slot 102.266916 bps/Hz
RBs 1 Nclu 5 alpha 4 rx 3 ry 4 Kc 2007829
FH average throughput per slot 93.107455 bps/Hz
Non FH average throughput per slot 152.986992 bps/Hz
Ideal FH average throughput per slot 114.563538 bps/Hz
Ideal FH Continous Power average throughput per slot 148.816901 bps/Hz
Non FH Continous Power average throughput per slot 152.986992 bps/Hz
FH Continous Power average throughput per slot 143.327416 bps/Hz
RBs 1 Nclu 7 alpha 3 rx 3 ry 0 Kc 1046943
FH average throughput per slot 106.264238 bps/Hz
Non FH average throughput per slot 187.533869 bps/Hz
Ideal FH average throughput per slot 127.105374 bps/Hz
Ideal FH Continous Power average throughput per slot 182.844455 bps/Hz
Non FH Continous Power average throughput per slot 187.533869 bps/Hz
FH Continous Power average throughput per slot 174.688099 bps/Hz
RBs 1 Nclu 11 alpha 9 rx 2 ry 0 Kc 1275646
FH average throughput per slot 129.582926 bps/Hz
Non FH average throughput per slot 231.673593 bps/Hz
Ideal FH average throughput per slot 136.162288 bps/Hz
Ideal FH Continous Power average throughput per slot 232.883792 bps/Hz
Non FH Continous Power average throughput per slot 231.673593 bps/Hz
```

```
FH Continous Power average throughput per slot 229.211199 bps/Hz
RBs 1 Nclu 13 alpha 4 rx 12 ry 6 Kc 9230289
FH average throughput per slot 135.975798 bps/Hz
Non FH average throughput per slot 256.439057 bps/Hz
Ideal FH average throughput per slot 147.807107 bps/Hz
Ideal FH Continous Power average throughput per slot 255.270065 bps/Hz
Non FH Continous Power average throughput per slot 256.439057 bps/Hz
FH Continous Power average throughput per slot 251.621898 bps/Hz
RBs 1 Nclu 17 alpha 8 rx 4 ry 9 Kc 3002700
FH average throughput per slot 153.205455 bps/Hz
Non FH average throughput per slot 298.093681 bps/Hz
Ideal FH average throughput per slot 163.597278 bps/Hz
Ideal FH Continous Power average throughput per slot 299.450359 bps/Hz
Non FH Continous Power average throughput per slot 298.093681 bps/Hz
FH Continous Power average throughput per slot 293.167494 bps/Hz
RBs 1 Nclu 19 alpha 9 rx 0 ry 17 Kc 9014816
FH average throughput per slot 159.067142 bps/Hz
Non FH average throughput per slot 325.747108 bps/Hz
Ideal FH average throughput per slot 178.859993 bps/Hz
Ideal FH Continous Power average throughput per slot 318.827081 bps/Hz
Non FH Continous Power average throughput per slot 325.747108 bps/Hz
FH Continous Power average throughput per slot 313.127362 bps/Hz
RBs 1 Nclu 23 alpha 9 rx 14 ry 4 Kc 7158884
FH average throughput per slot 172.321340 bps/Hz
Non FH average throughput per slot 347.796051 bps/Hz
Ideal FH average throughput per slot 182.823967 bps/Hz
Ideal FH Continous Power average throughput per slot 351.399945 bps/Hz
Non FH Continous Power average throughput per slot 347.796051 bps/Hz
```

```
FH Continous Power average throughput per slot 346.493863 bps/Hz
```

In `loss_test.m` (see appendix B.5), the combined throughput for all users is simulated for a range of d_s channel values. For each value of d_s, 50 simulations are averaged for CSM allocation, static allocation, and frequency hopping with the best available channel chosen via user priority. Power allocation is simulated as described before in `primes_test`
↪ `.m` Figures 5.7, 5.8, 5.9,5.10, 5.11, 5.12,5.13, 5.14, and 5.15 show the results. If the power allocation is done once and then kept over the entire hopping period N, the average throughput loss is between 50 and 60 percent. This is due to the power allocation being non-ideal once the frequency allocation moves away from the power allocation conditions. Both the CSM and the"ideal" frequency hopping versions suffer from this. However, if the power is reallocated each time slot, the total throughput for the CSM frequency hopping is comparable to the non-hopping version. This suggests a trade-off between power allocation update times and frequency hopping is available.

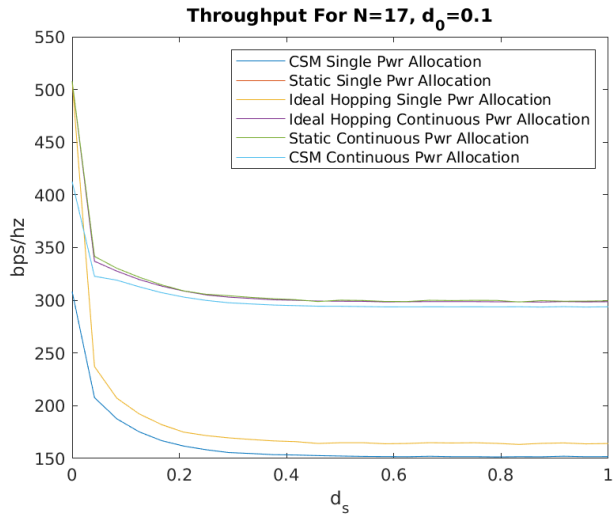

Figure 5.7: Throughput vs Channel d_s for $N = 17$, $d_0 = 0.1$

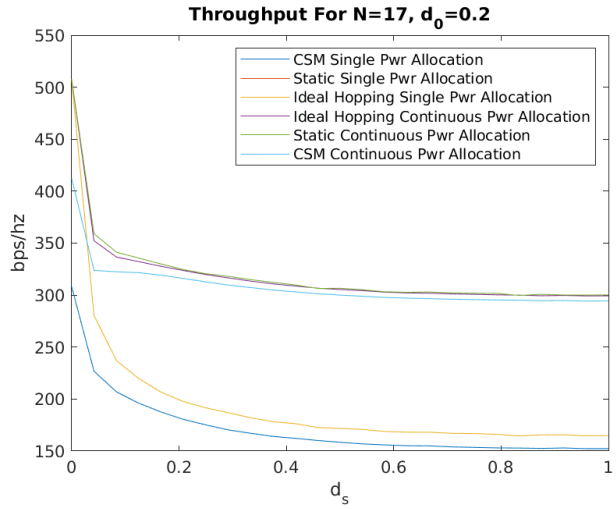

Figure 5.8: Throughput vs d_s for $N = 17$, $d_0 = 0.2$

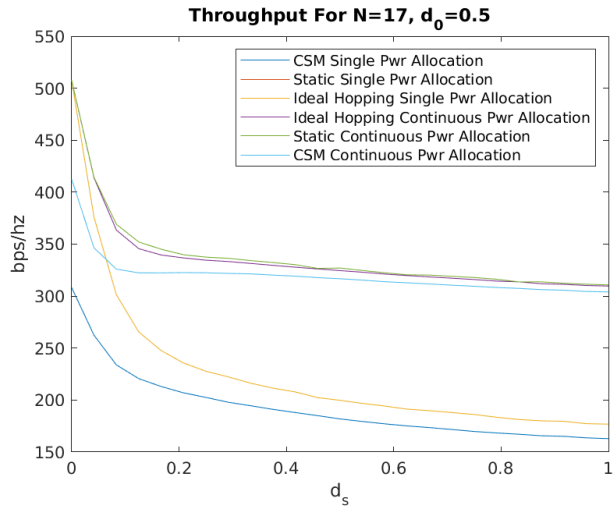

Figure 5.9: Throughput vs d_s for $N = 17$, $d_0 = 0.5$

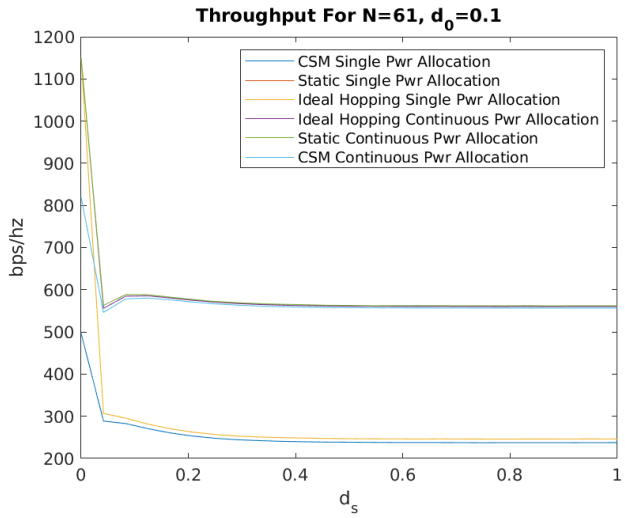

Figure 5.10: Throughput vs d_s for $N = 61$, $d_0 = 0.1$

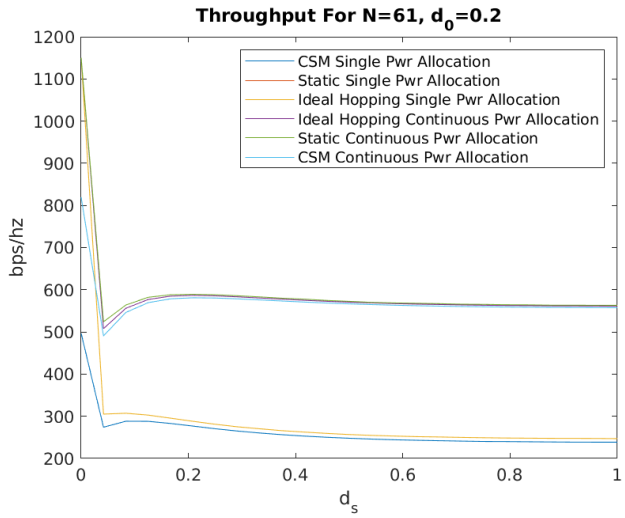

Figure 5.11: Throughput vs Channel d_s for $N = 61$, $d_0 = 0.2$

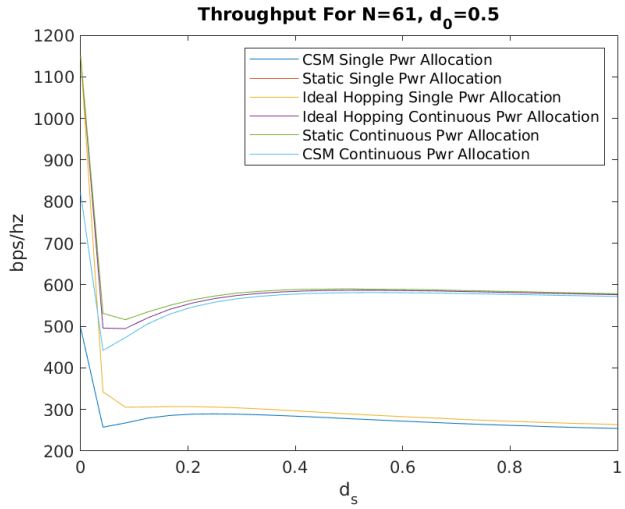

Figure 5.12: Throughput vs d_s for $N = 61$, $d_0 = 0.5$

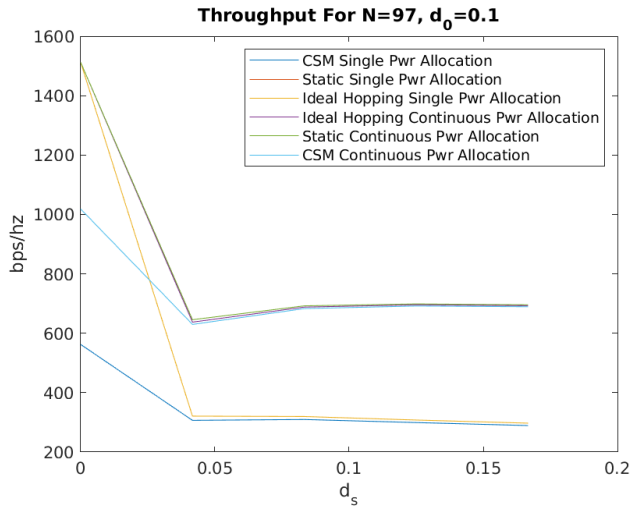

Figure 5.13: Throughput vs d_s for $N = 97$, $d_0 = 0.1$

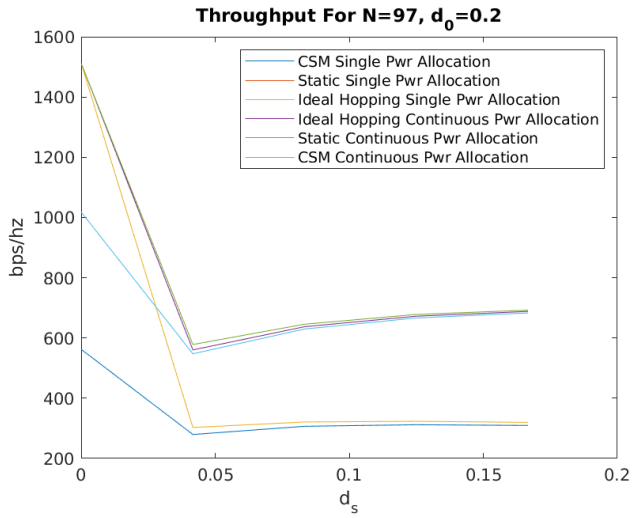

Figure 5.14: Throughput vs d_s for $N = 97$, $d_0 = 0.2$

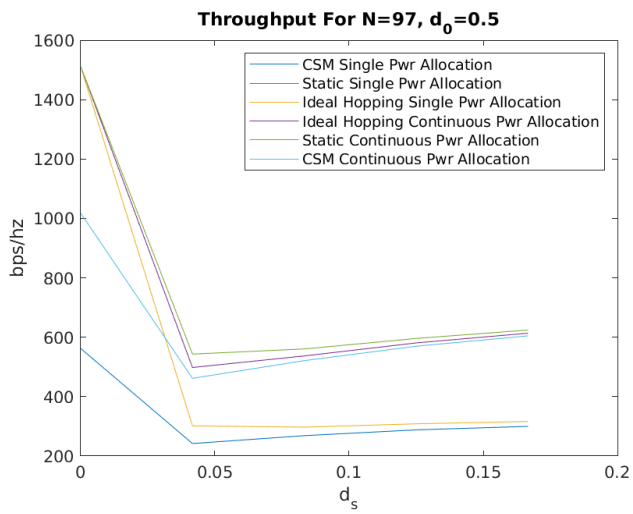

Figure 5.15: Throughput vs d_s for $N = 97$, $d_0 = 0.5$

5.2 5G Toolbox Simulation

The MATLAB 5G toolbox provides the opportunity to test out these concepts in a simulated 5G NR environment. Since the toolbox follows the current specification, symbol by symbol CSM hopping was not achievable in the current framework. However slot based CSM hopping was possible. Since the proposed changes to the configuration messages do not exist in the 5G specification currently, the allocation had to be done with a new configuration for each time slot. Figure 5.16 shows the spectrogram of a CSM slot based hopping allocation for a single K in a spread pattern with $N = 61$ on a QPSK downlink with 15 KHz subcarriers. Figure 5.17 shows the same in the time domain power. Figures 5.18, 5.19, 5.20, and 5.21 show the same configuration with cyclic hopping and static allocation respectively, the results are included for comparison.

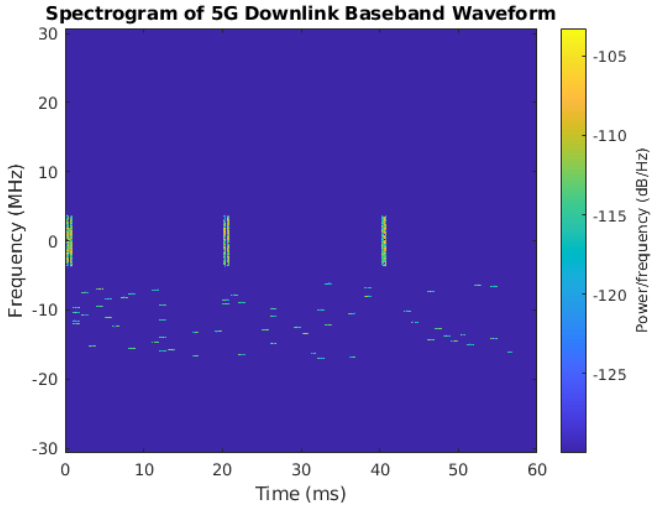

Figure 5.16: 5G Toolbox Spectrogram Slot Based CSM Hopping

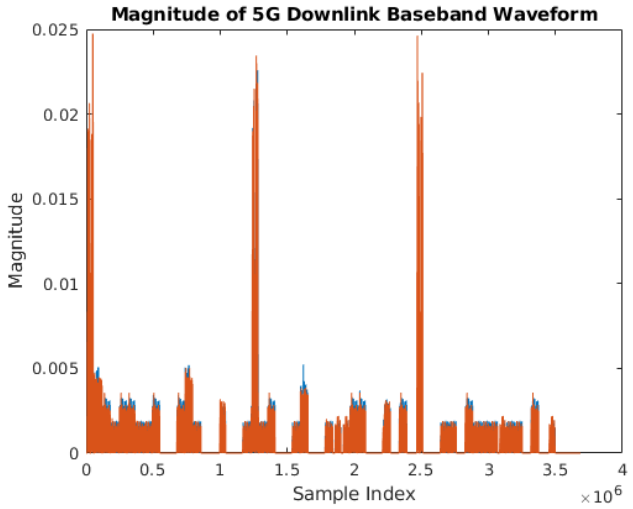

Figure 5.17: 5G Toolbox Time Domain Slot Based CSM Hopping

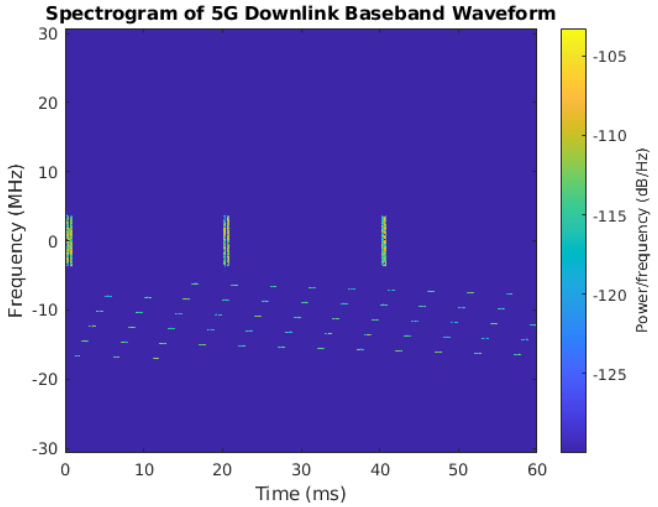

Figure 5.18: 5G Toolbox Spectrogram Slot Based Cyclic Hopping

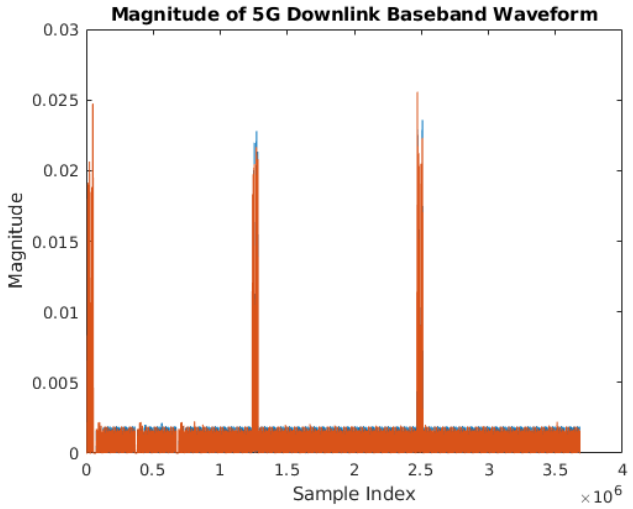

Figure 5.19: 5G Toolbox Time Domain Slot Based Cyclic Hopping

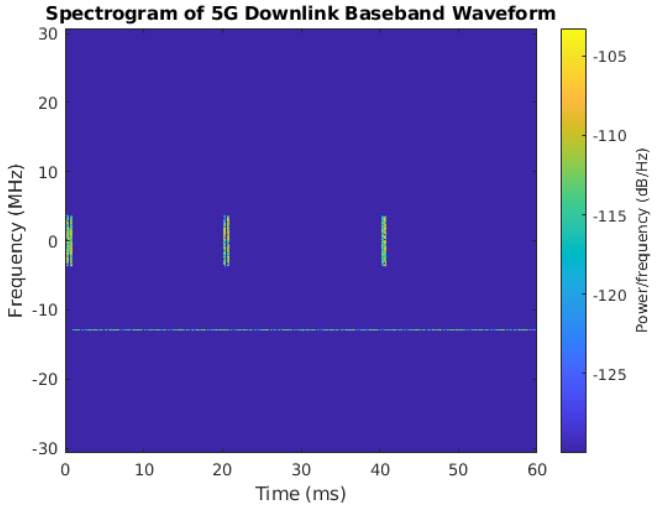

Figure 5.20: 5G Toolbox Spectrogram Static Channel

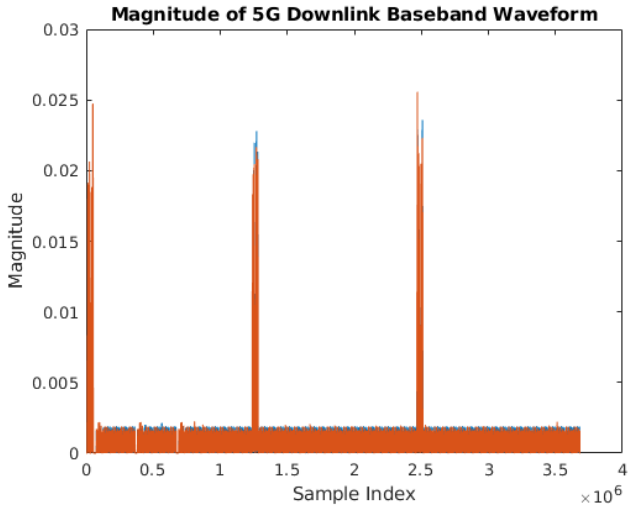

Figure 5.21: 5G Toolbox Time Domain Static Channel

Chapter 6: Summary and Future Work

6.1 Summary

A new model of frequency hopping for 5G NR networks based on a chaotic standard map has been proposed. It has been shown that in fast fading channels, the throughput loss is tolerable for the improved LPI characteristics. Slot based hopping could be implemented under the current specification, and with revisions to the specification the control channel overhead could be reduced, allocation security could be increased, and symbol based hopping could be implemented.

6.2 Future Work

A future effort with this work could involve modifying the MATLAB 5G toolbox to simulate the CSM based hopping sequence and measuring the throughput in the channel models provided there. A real world test could be performed using existing equipment and specification with software modification to the allocation algorithm at the network base station. Future research to identify what algorithms and schemes are used by 5G operators for resource block and power allocations would make more realistic comparisons between the CSM hopping and existing solutions.

List of Abbreviations

3GPP Third Generation Partnership Project

AWGN additive white Gaussian noise

BWP Bandwidth Part

CSI Channel State Information

CSM Chaotic Standard Map

LPD Low Probability of Detection

LPI Low Probability of Intercept

NR New Radio

OFDM Orthogonal Frequency Division Multiplexing

OFDMA Orthogonal Frequency Division Multiple Access

PBCH Physical Broadcast Channel

PDCCH Physical Downlink Control Channel

PDSCH Physical Downlink Shared Channel

PRACH Physical Random Access Channel

PRB Physical Resource Blocks

PUCCH Physical Uplink Control Channel

PUSCH Physical Uplink Shared Channel

RB Resource Block

RF radio frequency

RRC Radio Resource Controller

UE User Equipment

Appendix A: Data

Table A.1: Throughput Loss Simulation Results

$N = 17$ and $d_0 = 0.1$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	308.3078	507.9242	507.9242	507.9242	507.9242	412.5856
0.0417	207.7136	341.6755	237.1506	337.0484	341.6755	322.7849
0.0833	187.5746	330.3883	207.1567	327.6902	330.3883	319.1060
0.1250	175.2386	321.9381	192.2724	319.7240	321.9381	312.8325
0.1667	166.9293	314.6632	182.1134	313.5150	314.6632	307.2564
0.2083	161.7224	308.9491	174.9337	308.7711	308.9491	303.0540
0.2500	158.2691	305.7097	171.7034	305.1178	305.7097	299.9229
0.2917	155.6145	304.3980	169.5498	303.0271	304.3980	297.6912
0.3333	154.6898	302.7978	168.0100	301.7788	302.7978	296.5686
0.3750	153.5939	301.5321	166.6527	300.5087	301.5321	295.5450
0.4167	153.2783	300.6008	165.9670	300.0728	300.6008	294.9828
0.4583	152.7074	298.8312	164.2349	299.3180	298.8312	294.4348
0.5000	152.3050	300.1606	164.9064	299.1417	300.1606	294.3451
0.5417	151.9449	299.8601	164.9429	299.0300	299.8601	294.0784
0.5833	151.7834	298.9575	163.9301	298.5082	298.9575	293.7589
0.6250	151.6547	298.8833	164.2150	298.6292	298.8833	293.7888
0.6667	152.0653	300.0451	164.9467	298.8152	300.0451	293.8625
0.7083	151.7000	299.7710	164.6671	298.7479	299.7710	293.8257
0.7500	151.7161	299.9837	164.9103	298.7674	299.9837	293.8637
0.7917	151.4386	299.9107	164.3139	298.6008	299.9107	293.8058
0.8333	151.6794	298.4622	163.2920	298.6121	298.4622	293.8820
0.8750	151.5902	299.6951	164.3093	298.3579	299.6951	293.6323
0.9167	152.1163	299.1635	164.7347	298.8112	299.1635	293.9798
0.9583	151.6650	299.2676	163.8330	298.5067	299.2676	293.6321
1	151.7440	299.5431	164.1836	298.6834	299.5431	293.8697

Table A.2: Throughput Loss Simulation Results

$N = 17$ and $d_0 = 0.2$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	308.3078	507.9242	507.9242	507.9242	507.9242	412.5856
0.0417	226.8133	359.1355	280.3247	352.3578	359.1355	323.7916
0.0833	206.9793	341.2678	237.1725	336.6253	341.2678	322.4494
0.1250	196.0572	335.6564	219.7761	332.2315	335.6564	321.5934
0.1667	187.5513	329.9229	206.6247	327.6862	329.9229	319.1572
0.2083	180.4772	324.5612	197.5924	323.6397	324.5612	316.0116
0.2500	175.1269	320.5854	191.5885	319.7754	320.5854	312.7315
0.2917	170.3400	318.0821	186.8354	316.5753	318.0821	309.6900
0.3333	167.1105	314.9084	182.0353	313.4724	314.9084	307.2929
0.3750	164.0955	312.2880	178.2294	310.7936	312.2880	304.9005
0.4167	162.2096	309.7377	176.3040	308.7262	309.7377	303.0975
0.4583	160.1622	306.4094	172.6915	306.6821	306.4094	301.3037
0.5000	158.4146	306.6057	171.7875	305.3751	306.6057	300.0311
0.5417	156.9248	305.1531	170.7816	304.3359	305.1531	298.8889
0.5833	155.8849	303.2255	168.8167	302.8497	303.2255	297.7646
0.6250	155.1164	302.6964	168.1727	302.1018	302.6964	297.1132
0.6667	154.9277	303.0141	168.0913	301.8258	303.0141	296.5941
0.7083	154.0011	302.1053	167.0480	301.1273	302.1053	296.0343
0.7500	153.5327	301.8265	166.8099	300.7581	301.8265	295.7165
0.7917	153.0400	301.5988	166.0528	300.2365	301.5988	295.3263
0.8333	152.8971	299.6919	164.5604	299.9674	299.6919	295.1262
0.8750	152.5482	300.7741	165.4387	299.3832	300.7741	294.6262
0.9167	152.9737	300.1096	165.7160	299.6910	300.1096	294.7891
0.9583	152.2934	300.0812	164.6479	299.2337	300.0812	294.2741
1	152.2852	300.1935	164.8434	299.2770	300.1935	294.4009

Table A.3: Throughput Loss Simulation Results

$N = 17$ and $d_0 = 0.5$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	308.3078	507.9242	507.9242	507.9242	507.9242	412.5856
0.0417	262.3846	414.3265	375.9795	413.8162	414.3265	346.4562
0.0833	233.8494	369.1325	301.3300	363.5435	369.1325	325.9499
0.1250	220.6351	352.0980	265.5234	345.6618	352.0980	322.3765
0.1667	213.0738	345.1302	247.6841	339.5766	345.1302	322.3511
0.2083	206.9666	339.7950	235.6576	336.7247	339.7950	322.6181
0.2500	202.5655	337.5232	227.7584	334.5789	337.5232	322.4381
0.2917	197.9245	336.3612	222.3279	333.2810	336.3612	321.8333
0.3333	194.5481	334.1934	216.2514	331.4507	334.1934	321.3938
0.3750	191.0647	332.3208	211.4906	329.6087	332.3208	320.2211
0.4167	187.9897	330.2425	207.7890	327.8207	330.2425	319.2155
0.4583	185.0039	326.6550	202.3473	326.1741	326.6550	317.7813
0.5000	181.8619	327.0178	199.7642	324.5284	327.0178	316.7017
0.5417	179.4094	324.6445	196.7788	322.9836	324.6445	315.4006
0.5833	177.1289	322.4542	194.2419	321.1903	322.4542	313.8649
0.6250	175.1373	320.4991	191.2535	319.7690	320.4991	312.7954
0.6667	173.5194	320.2226	189.8288	318.4474	320.2226	311.6762
0.7083	171.6832	319.0232	187.9546	317.2295	319.0232	310.4979
0.7500	169.7921	317.8909	186.1128	315.9458	317.8909	309.4708
0.7917	168.3559	316.2077	183.4633	314.4490	316.2077	308.3214
0.8333	167.1295	313.6775	181.2794	313.7125	313.6775	307.4166
0.8750	165.6719	313.6992	180.0364	311.8779	313.6992	306.2448
0.9167	165.1301	312.3867	179.5615	311.3345	312.3867	305.6988
0.9583	163.6468	311.4526	177.2990	310.2257	311.4526	304.4449
1	162.9162	310.7675	176.8683	309.6845	310.7675	304.0200

Table A.4: Throughput Loss Simulation Results

$N = 61$ and $d_0 = 0.1$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	496.6757	1.1501e+03	1.1501e+03	1.1501e+03	1.1501e+03	819.5123
0.0417	288.6294	563.1935	306.7423	556.0782	563.1935	546.3143
0.0833	282.6688	588.7333	295.5443	584.8017	588.7333	578.4351
0.1250	270.5474	588.4262	281.2571	585.5640	588.4262	580.4291
0.1667	260.3796	582.4241	270.4131	580.4758	582.4241	575.9677
0.2083	253.1190	576.8503	262.3362	574.9514	576.8503	570.8207
0.2500	247.8026	572.2043	256.8222	570.6547	572.2043	566.6314
0.2917	244.3728	569.3034	253.1279	567.3192	569.3034	563.5254
0.3333	242.1963	566.9764	250.8917	565.0046	566.9764	561.2381
0.3750	240.5421	565.5353	249.2348	563.4291	565.5353	559.7026
0.4167	239.5148	564.6535	248.3001	562.3773	564.6535	558.6555
0.4583	238.7879	563.2702	247.4226	561.5386	563.2702	557.9089
0.5000	238.4950	562.9479	246.8949	561.0761	562.9479	557.4768
0.5417	238.1489	562.1251	246.5250	560.8598	562.1251	557.1968
0.5833	237.8045	562.2641	246.4899	560.5744	562.2641	556.9046
0.6250	237.6795	562.4298	246.1836	560.4514	562.4298	556.8050
0.6667	237.6946	562.3581	246.3866	560.4813	562.3581	556.7822
0.7083	237.5559	561.9901	246.1269	560.2750	561.9901	556.6823
0.7500	237.3080	562.1572	246.0503	560.1634	562.1572	556.5560
0.7917	237.5559	561.9174	245.9506	560.1234	561.9174	556.5502
0.8333	237.5794	562.2748	246.0964	560.2329	562.2748	556.6118
0.8750	237.5504	561.7911	246.0509	560.0769	561.7911	556.5265
0.9167	237.4299	562.0482	246.2575	560.2087	562.0482	556.5728
0.9583	237.4900	562.1096	246.3293	560.2258	562.1096	556.5822
1	237.6798	562.1748	246.0714	560.1899	562.1748	556.5645

Table A.5: Throughput Loss Simulation Results

$N = 61$ and $d_0 = 0.2$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	496.6757	1.1501e+03	1.1501e+03	1.1501e+03	1.1501e+03	819.5123
0.0417	274.0919	524.1606	305.1495	507.8716	524.1606	490.7142
0.0833	288.2888	563.8542	307.1952	556.0717	563.8542	546.2325
0.1250	288.0197	582.1880	302.8290	576.9968	582.1880	569.4423
0.1667	282.9802	588.2452	295.5303	584.8747	588.2452	578.5522
0.2083	276.8050	589.2950	288.1267	586.6710	589.2950	581.0264
0.2500	270.4619	588.3168	281.1136	585.6264	588.3168	580.4755
0.2917	264.9870	585.8741	275.0534	583.2641	585.8741	578.4994
0.3333	260.3263	583.0995	270.3564	580.4581	583.0995	575.9577
0.3750	256.3243	579.9797	265.8437	577.6638	579.9797	573.3294
0.4167	252.9926	577.7895	262.5060	575.0240	577.7895	570.8394
0.4583	250.2019	574.7140	259.3867	572.6176	574.7140	568.5475
0.5000	247.9275	572.5951	256.7411	570.5596	572.5951	566.6097
0.5417	245.9348	570.0516	254.6747	568.8521	570.0516	564.9016
0.5833	244.3472	568.8232	253.1752	567.2510	568.8232	563.4237
0.6250	243.1173	567.9878	251.7447	566.0164	567.9878	562.1973
0.6667	242.0854	567.1245	250.9514	565.0835	567.1245	561.2825
0.7083	241.1438	565.8507	249.9295	564.1239	565.8507	560.3717
0.7500	240.2892	565.3934	249.1454	563.3035	565.3934	559.6199
0.7917	240.0271	564.5738	248.4789	562.7067	564.5738	559.0591
0.8333	239.5639	564.2883	248.0752	562.3705	564.2883	558.6449
0.8750	239.1173	563.5584	247.7184	561.8044	563.5584	558.1996
0.9167	238.7188	563.4986	247.5898	561.6145	563.4986	557.9461
0.9583	238.5923	563.2465	247.4211	561.3740	563.2465	557.7020
1	238.5558	563.1385	246.9745	561.1235	563.1385	557.4649

Table A.6: Throughput Loss Simulation Results

$N = 61$ and $d_0 = 0.5$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	496.6757	1.1501e+03	1.1501e+03	1.1501e+03	1.1501e+03	819.5123
0.0417	257.5043	531.2153	342.2228	495.6841	531.2153	441.8163
0.0833	267.4624	515.8823	305.5131	494.5232	515.8823	473.0166
0.1250	279.0943	534.5093	306.0070	520.4284	534.5093	505.9544
0.1667	285.6914	550.5963	306.9858	541.1672	550.5963	529.7146
0.2083	288.7527	563.5923	306.9446	556.0488	563.5923	546.3045
0.2500	289.2965	572.8775	305.6869	566.8287	572.8775	558.1458
0.2917	288.6789	579.9565	303.9582	574.1775	579.9565	566.3741
0.3333	287.2027	584.1833	301.2567	579.3205	584.1833	572.0474
0.3750	285.1875	587.1235	298.6439	582.6622	587.1235	575.9452
0.4167	282.9526	589.0664	295.5869	584.8692	589.0664	578.5388
0.4583	280.5182	589.3401	292.3990	585.9769	589.3401	580.0421
0.5000	277.8221	590.1181	289.2723	586.5531	590.1181	580.8546
0.5417	275.4921	589.0249	286.5060	586.6477	589.0249	581.1027
0.5833	272.8181	588.8445	283.8009	586.2004	588.8445	580.8841
0.6250	270.5420	588.5462	281.0780	585.5581	588.5462	580.4155
0.6667	268.2239	587.4783	278.8594	584.8869	587.4783	579.8134
0.7083	266.0220	586.0668	276.2905	583.7304	586.0668	578.9069
0.7500	263.8871	585.3100	274.0956	582.6750	585.3100	577.9295
0.7917	262.1574	583.8601	272.0093	581.5203	583.8601	576.8854
0.8333	260.4840	583.0579	270.1761	580.4302	583.0579	575.9458
0.8750	258.5330	581.4656	268.3607	579.2285	581.4656	574.8114
0.9167	256.9049	580.6239	266.6508	578.2108	580.6239	573.8050
0.9583	255.5952	579.2754	265.2700	577.1640	579.2754	572.8367
1	254.3871	578.2164	263.4357	576.0456	578.2164	571.7822

Table A.7: Throughput Loss Simulation Results

$N = 97$ and $d_0 = 0.1$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	562.3798	1.5123e+03	1.5123e+03	1.5123e+03	1.5123e+03	1.0184e+03
0.0417	306.7072	645.5608	320.9741	637.4081	645.5608	629.7174
0.0833	309.5877	692.2557	319.6309	688.0392	692.2557	683.0130
0.1250	299.2912	698.7365	307.8070	695.6708	698.7365	691.6125
0.1667	289.2313	695.7743	297.2740	692.7673	695.7743	689.1660
0.2083	281.6858	690.3358	289.2112	687.8243	690.3358	684.5243
0.2500	276.2731	685.8301	283.6248	683.3478	685.8301	680.1441
0.2917	272.7085	682.1705	279.7494	679.8788	682.1705	676.8271
0.3333	270.1436	679.7067	277.2972	677.3759	679.7067	674.3614
0.3750	268.4150	678.1157	275.5343	675.6997	678.1157	672.6934
0.4167	267.1702	676.2167	274.1329	674.4578	676.2167	671.5039

Table A.8: Throughput Loss Simulation Results

$N = 97$ and $d_0 = 0.2$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	562.3798	1.5123e+03	1.5123e+03	1.5123e+03	1.5123e+03	1.0184e+03
0.0417	279.5612	578.2343	302.8874	560.3483	578.2343	547.4574
0.0833	306.8284	645.6968	321.0298	637.4294	645.6968	629.7555
0.1250	311.9305	678.0623	323.3911	672.3093	678.0623	666.4188
0.1667	309.5389	692.4237	319.7483	688.0878	692.4237	683.0213
0.2083	304.7950	697.9118	313.8941	694.2861	697.9118	689.7819
0.2500	299.1578	699.1286	307.8859	695.6994	699.1286	691.5932
0.2917	293.9335	697.7613	302.1826	694.7817	697.7613	690.9779
0.3333	289.3507	695.4535	297.2791	692.7631	695.4535	689.1402
0.3750	285.2469	692.9769	292.8552	690.3369	692.9769	686.8926
0.4167	281.6597	689.8020	289.0473	687.8244	689.8020	684.4894

Table A.9: Throughput Loss Simulation Results

$N = 97$ and $d_0 = 0.5$						
	Single Power Allocation			Cont. Power Allocation		
d_s	CSM Th	NH Th	FH Ideal Th	FI Ideal Th	NH Th	CSM Th
0	562.3798	1.5123e+03	1.5123e+03	1.5123e+03	1.5123e+03	1.0184e+03
0.0417	242.1055	543.2977	301.7014	498.0970	543.2977	461.7137
0.0833	268.6770	560.6035	297.2643	537.1221	560.6035	521.2543
0.1250	288.1697	595.9166	308.5526	580.8997	595.9166	569.7990
0.1667	299.8673	624.2952	316.5706	613.6926	624.2952	604.6876
0.2083	306.7492	645.8691	321.2152	637.3980	645.8691	629.7437
0.2500	310.1611	661.7317	323.2291	654.7413	661.7317	647.9128
0.2917	311.7450	673.4742	323.7354	667.3708	673.4742	661.2251
0.3333	311.8959	681.9357	322.9001	676.5963	681.9357	670.8823
0.3750	310.9100	688.0373	321.6029	683.3222	688.0373	677.9920
0.4167	309.4808	691.5867	319.4720	688.0020	691.5867	682.9967

Appendix B: MATLAB Code

B.1 chaoticmap.m

```
function [L,C,L_nh,C_nh,C_ideal] = chaoticmap(Nu,Nsc,RBs,alpha,rx,ry,Kc,
    ↪ Rb_size, csi)

 %When Frequency Hopping is activated, this code is executed
% after run button is pushed. Currently, Chaotic Map Technique
% is used but other techniques can be implemented in future
% It would be interesting to implement a list of techniques
% to make the user able to select the one desired.
%We should create a listBox with a set of FH techniques
% Then execute the FH code corresponding with the FH technique selected.



%Frequency Hopping Technique based on chaotic map technique


%DATA in needed to perform the technique
% ---------------------------------------------------------------------
%--kusers is the number of users in the system
%--Nsc is the number of subcarriers in the system
%--Rbs is the number of Resource Blocks for each user
%--alpha represents a pattern index for alpha=2,...,Nclu-1. alpha=2 is an
% example used for simulations
%--rx, ry are integers varying from 0 to Nclu-1.(rx,ry)=(3,1) are examples
% used for simulations
%--Kc is a positive integer. 1000 is an example used for simulations.
```

```
% ----------------------------------------------------------------------



%DATA out
% ----------------------------------------------------------------------
%--CSM_all contains the FH pattern for each user K, it means the Frequency
% hops from the subcarriers in matrix L of user K to the subcarriers in
% matrix C
%--L represents the first FH assignation to the users, each value of the
% matrix represents the assignation of Nclu clusters
% to the user marked for the matrix. Seed matrix.
%--C represents the change in the FH assignation based on chaotic map
% ----------------------------------------------------------------------

    Nclu=floor(Nsc/(RBs*Rb_size)); %Number of clusters in the channel
                                   %It is RBs * Rb_size because each RB has Rb_size
                                   %subcarriers

    %csi = cell(1,Nu);
    %for i=1:Nu
    % %csi{1,i}(1:Nsc) = rand(1,Nsc);
    % %distance = normrnd(1.0,0.5);
    % %temp = ((abs(normrnd(1,0.5,[1,Nsc])).^2)./0.5)*distance;
    % var = 1.5;
    % temp = ((abs(normrnd(0,var,[1,Nsc])).^2)/var);
    % %temp(temp < 0)=0.000001;
    % %temp(temp > 1)=1;
    % csi{1,i}(1:Nsc) = temp;
```

```
%end

%Next code will be executed only if FH is activated

%if useHop==1

    L=zeros(Nclu,Nclu); %Seed Matrix

    %Compute alpha inverse
    alpha_inv=0;
    for n=0:Nclu-1
        if mod(alpha*n,Nclu)==1
            alpha_inv=n;
        end
    end

    %Construct an Nclu x Nclu seed matrix using Latin
    %Square
    for i = 1:Nclu
        for j=1:Nclu
            L(i,j)=mod((alpha*(i-1)+(j-1)),Nclu);
        end
    end
    kusers=max(max(L))+1; %Max users in the system
    S_FH_all=cell(1,kusers); %FH pattern matrix for each user
    s_fh_all=cell(1,kusers); %FH pattern matrix for each user
    CSM_all=cell(1,kusers); %CSM FH pattern matrix for each user
```

```
%S_FH represents the positions in L for user k
for k=1:kusers
    row=1;
    column=1;
    for i = 1:Nclu
        for j=1:Nclu
            if L(i,j)+1==(k)
                S_FH(row,column)=i;
                S_FH(row,column+1)=j;
                column=1;
                row=row+1;
            end
        end
    end

    s_fh=zeros(length(S_FH),1);
    for j=0:Nclu-1
        s_fh(j+1)=(mod(floor(((k-1)-(j))*alpha_inv), Nclu));
    end
    s_fh=s_fh';

    %Convert the FH pattern into CSM FH pattern
    %Rows CSM Matrix
    for j=0:Nclu-1
        CSM_row(j+1)=(mod(s_fh(j+1)+j+rx+ry,Nclu));
    end
    %Columns CSM Matrix
    for j=0:Nclu-1
```

```
        CSM_column(j+1)=(mod(j+ry+round((Kc*sin((CSM_row(j+1)*Nclu)
            ↪ /(2*pi)))),Nclu));
    end
    %CSM Matrix
    CSM=zeros(size(S_FH,1),size(S_FH,2));
    CSM(:,1)=CSM_row(:)+1; %Scaled to +1 because matlab does not
        ↪ have 0 indexes
    CSM(:,2)=CSM_column(:)+1;

    S_FH_all(1,k)={S_FH};
    s_fh_all(1,k)={s_fh};
    CSM_all(1,k)={CSM};
end

%Seed Matrix converted according to the FH pattern
C=L;
for i=1:length(CSM_all)
    for j=1:size(S_FH_all{1,i},1)
        C(CSM_all{1,i}(j,1),CSM_all{1,i}(j,2))=L(S_FH_all{1,i}(j,1),
            ↪ S_FH_all{1,i}(j,2));
    end
end

%If Nusers is lower than Nclusters, we randomly assign
%the free subcarriers between the users.
if Nu<Nclu
    for i=1:size(C,1)
        for j=1:size(C,2)
```

```
                if Nu<=C(i,j)
                    C(i,j)=randi([1 Nu],1);
                end
            end
        end
    %We cannot have more users than clusters
    elseif Nu>Nclu
       error('Max␣Users␣allowed␣=␣#Subcarriers/(#RB*Rb_size)');
    end

    %Not zeros within the matrix (user one instead of user zero)
    C=C+1;



%If not FH is desired
%else

    C_nh=zeros(Nclu,Nclu);
    C_nh=C_nh+3301;
    %First RA approach when no FH is activated
    %It would be interesting to compare the CSI between all
    %the users but it requires a high computation cost when
    %there are a lot of subcarriers.
    cluster_avail = ones(1,Nclu);
    for column=1:Nu
        cluster_qual = zeros(1,Nclu);
        for row=1:Nclu
```

```
        cluster_qual(row) = sum(csi{1,column}((row-1)*RBs*Rb_size+1:
            ↪ row*RBs*Rb_size));
    end
    cluster_qual(cluster_avail==0)=0;
    [M, I] = max(cluster_qual);
    C_nh(I,:)=column;
    cluster_avail(I) = 0;

end

if Nu<Nclu
    for i=1:size(C_nh,1)
        for j=1:size(C_nh,2)
            if Nu<=C_nh(i,j)
                C_nh(i,j)=randi([1 Nu],1);
            end
        end
    end

elseif Nu>Nclu
   error('Max␣Users␣allowed␣=␣#Subcarriers/(#RB*Rb_size)');
end

%C_nh=C_nh+1;
L_nh=C_nh;

C_ideal=zeros(Nclu,Nclu);
C_ideal=C_ideal+3301;
```

```
for slot=1:Nu
    cluster_avail = ones(1,Nclu);
    for column=1:Nu
        cluster_qual = zeros(1,Nclu);
        for row=1:Nclu
            cluster_qual(row) = sum(csi{slot,column}((row-1)*RBs*
                ↪ Rb_size+1:row*RBs*Rb_size));
        end
        cluster_qual(cluster_avail==0)=-10000000;
        [M, I] = max(cluster_qual);
        C_ideal(I,slot)=column;
        cluster_avail(I) = 0;

    end
end

if Nu<Nclu
    for i=1:size(C_ideal,1)
        for j=1:size(C_ideal,2)
            if Nu<=C_ideal(i,j)
                C_ideal(i,j)=randi([1 Nu],1);
            end
        end
    end

elseif Nu>Nclu
   error('Max␣Users␣allowed␣=␣#Subcarriers/(#RB*Rb_size)');
```

```
        end


    %end
    %When there are less users than clusters we assign randomly
    %the free clusters to the users
    if Nu<Nclu
        for i=1:size(C,1)
            for j=1:size(C,2)
                if Nu<=C(i,j)
                    C(i,j)=randi([1 Nu],1);
                end
            end
        end


    elseif Nu>Nclu
        error('Max␣Users␣=␣#Subcarriers/(#RB*Rb_size)');
    end



end
```

B.2 csigen.m

```
function [csi] = csi_gen(Nu,Nsc,var,ds,d0)
%Channel State Information Generateror Fuction
%Generates random CSI for a given number of users and subcarriers
%Time varying fading is according to a first order auto-regressive moodel
```

```
%DATA in needed to perform the technique
% ----------------------------------------------------------------------
%--Nu is the number of users in the system
%--Nsc is the number of subcarriers in the system
%--var is the csi variance.
%--ds is the distance moved between each csi sample
%--d0 is the decay factor
% ----------------------------------------------------------------------



%DATA out
% ----------------------------------------------------------------------
%--csi cell data of csi for all users
% ----------------------------------------------------------------------
    csi = cell(Nsc,Nu);
    alpha = exp(-(ds/d0));
    fade_var=(1-alpha^2)*var;
    for i=1:Nu
        temp = ((abs(normrnd(0,var,[1,Nsc])).^2)/var);
        csi{1,i}(1:Nsc) = temp;
        for j=2:Nsc
            %First order autoregressive time varying fading channel
            csi{j,i}(1:Nsc) = alpha*csi{j-1,i}(1:Nsc)+normrnd(0,fade_var,[1,
                ↪ Nsc]);
            %Don't allow csi to be less than 0.
            csi{j,i}(1:Nsc) = max(csi{j,i}(1:Nsc),0);
        end
    end
```

```
end
```

B.3 waterfilling.m

```
function [Pn_opt,csi_ra, Cn] = waterfilling(csi,C, RBs, Pt, Rb_size,
    ↪ update_period)

    % Water Filling Algorithm
    % ---------------------------------------------------------------

    % - Water Filling algorithm to find the optimal value to allocate power
        ↪  for
    % different channels.
    % - CSI (Subchannel information) is assumed to be known. csi is
    % a cell containing CSI information for all the users in each
    % subcarrier
    % - C is the Matrix with the allocation for each user
    % - RBs is the number of resoruce blocks per user
    % - Pt is total power in the system
    % - Maximum Throughput, Cn, for each Time Slot
    % - In future, Bandwidth can be included (B), where B represents the
        ↪ bandwidth used for
    % subchannels.

    % ---------------------------------------------------------------

    % Instead of using loops, we are going to use matrix vector computation
        ↪  because
```

```
% MATLAB is very slow using loops but very fast when using matrix
    ↪ vector
% computation.



%CSI of each user is computed for the RA selected.
Nclu = size(C,1);

csi_ra=cell(Nclu,Nclu); %CSI for the current RA
for column=1:Nclu
    for row=1:Nclu
        for user=1:Nclu
            csi_ra{user,column}((row-1)*RBs*Rb_size+1:row*RBs*Rb_size) =
                ↪ csi{user,C(row,column)}((row-1)*RBs*Rb_size+1:row*RBs*
                ↪ Rb_size);
        end
    end
end

%mc = length(csi); % Number of subchannels/subcarriers
M = 2e3; % Number of grid points you want to compute the lagrangian
    ↪ dual g of mu
mu_axis= linspace(1e-15,max(max(cell2mat(csi))),M);

% Power allocation for each Time Slot
Pn = cell(1,Nclu); % Power
g = cell(1,Nclu); % Cost Function
Pn_opt = cell(1,Nclu); % Optimal power
```

```
    Cn = cell(1,Nclu); % Maximum Rate reached


    %For each user
    for i=1:Nclu
        Cn{1,i}=0;
        %For each time slot
        for slot=1:Nclu
            if mod(slot-1,update_period)==0
                %If it's the update period, update the power allocation for
                    ↪ the slot
                Pn{1,i} = max((1./mu_axis - (1./csi_ra{i,slot})'),0); % A
                    ↪ value less than 0 will be replaced by 0.
                %% Lagrange Dual Function
                g{1,i} = sum(log(1+Pn{1,i}.*(repmat(csi_ra{i,slot}',[1 M]))))
                    ↪  - mu_axis.*(sum(Pn{1,i}) - Pt);
                %We have to find the minimum of g
                [ind]= find(g{1,i}==min(g{1,i})); %We need the index to
                    ↪ compute the optimal mu
                mu = mu_axis(ind);
                %Compute the optimal Powers based on the optimal mu value.
                Pn_opt{1,i} = max(1./mu - 1./csi_ra{i,slot},0);
            end

            Cn{1,i} = Cn{1,i}+sum(1*log2(1+Pn_opt{1,i}.*csi_ra{i,slot})); %
                ↪ Max Throughput without considering the bandwidth, B
        end
    end
end
```

B.4 primestest.m

```
clear;
runs = primes(15);
%runs = [0,17];
output = [];
d0 = 0.1;
ds = 0.5;
var = 1.5;
for r=runs(2:end)
    s.Rb_size=12;
    s.Nu = r;
    p_aloc_rate = s.Nu;
    s.Nsc = s.Nu*s.Rb_size;
    s.RBs = floor(s.Nsc/s.Rb_size/s.Nu);
    s.Nclu=floor(s.Nsc/(s.RBs*s.Rb_size));
    s.alpha = randi([2,s.Nclu-1],1);
    s.rx = randi([0,s.Nclu-1],1);
    s.ry = randi([0,s.Nclu-1],1);
    s.Kc = randi([0,10000000],1);
    s.Pt=1000;
    fprintf("RBs %d Nclu %d alpha %d rx %d ry %d Kc %d\n", s.RBs, s.Nclu, s
        ↪ .alpha, s.rx, s.ry, s.Kc);
    s.csi = csi_gen(s.Nu,s.Nsc,var,ds,d0);
    [s.L,s.C,s.L_nh,s.C_nh,s.C_ideal] = chaoticmap(s.Nu,s.Nsc,s.RBs,s.alpha
        ↪ ,s.rx,s.ry,s.Kc,s.Rb_size,s.csi);
    [s.Pn_opt,s.csi_ra,s.Cn] = waterfilling(s.csi,s.C,s.RBs,s.Pt,s.Rb_size,
        ↪ p_aloc_rate);
```

```
[s.Pn_opt2,s.csi_ra2,s.Cn2] = waterfilling(s.csi,s.C_nh,s.RBs,s.Pt,s.
    ↪ Rb_size,p_aloc_rate);
[s.Pn_opt3,s.csi_ra3,s.Cn3] = waterfilling(s.csi,s.C_ideal,s.RBs,s.Pt,s
    ↪ .Rb_size,p_aloc_rate);
[s.Pn_opt4,s.csi_ra4,s.Cn4] = waterfilling(s.csi,s.C_ideal,s.RBs,s.Pt,s
    ↪ .Rb_size,1);

s.count = zeros(1,s.Nu);
for i=1:s.Nu
    s.count(i)=sum(sum(s.C==i));
    %fprintf("user %d clusters %d\n", i, s.count(i));
end

%use the throughput for user 1
%s.Th=sum(cell2mat(s.Cn))/s.Nu/s.Nclu;
s.Th=sum(s.Cn{1})/s.Nclu;
fprintf("FH average throughput per slot %f bps/Hz\n", s.Th);
%s.Th2=sum(cell2mat(s.Cn2))/s.Nu/s.Nclu;
s.Th2=sum(s.Cn2{1})/s.Nclu;
s.Nts=size(s.C,2);
fprintf("Non FH average throughput per slot %f bps/Hz\n", s.Th2);
%s.Th3=sum(cell2mat(s.Cn3))/s.Nu/s.Nclu;
s.Th3=sum(s.Cn3{1})/s.Nclu;
fprintf("Ideal FH average throughput per slot %f bps/Hz\n", s.Th3)
s.Th4=sum(s.Cn4{1})/s.Nclu;
fprintf("Ideal FH Continous Power average throughput per slot %f bps/Hz
    ↪ \n", s.Th4)
output = [output s];
```

```
end
%save('newstruct.mat', 'output')

%Plotting results for first time slot
f1 = figure(1);
clf(f1);
%hold on;
sz=size(s.C);
%sz=[4,5]
%for i=1:sz(2)
% scatter(C(:,i),1:1:size(C,1),'filled','DisplayName',['timeslot ',num2str(
    ↪ i)])
%end
heatmap(s.C);
title('Resource␣Allocation␣for␣each␣User␣Freqency␣Hopping')
ylabel('Resource␣cluster')
xlabel('Timeslot')
%legend();

f2 = figure(2);
clf(f2);
bar(s.Pn_opt{1,1},1,'r')
xlabel('Subchannels')
ylabel('Power␣Allocated')
title('Power␣Allocation.␣Total␣Power␣for␣the␣channel␣=␣',num2str(sum(s.
    ↪ Pn_opt{1,1})))

f3 = figure(3);
```

```
clf(f3);
bar(s.csi_ra{1,1},1)
xlabel('Subchannels')
ylabel('CSI')
title('CSI-RA␣for␣the␣first␣Time␣Slot')

f4 = figure(4);
clf(f4);
%hold on;
sz=size(s.C_nh);
%sz=[4,5]
%for i=1:sz(2)
% scatter(C(:,i),1:1:size(C,1),'filled','DisplayName',['timeslot ',num2str(
    ↪ i)])
%end
heatmap(s.C_nh);
title('Resource␣Allocation␣for␣each␣User␣No␣Hopping')
ylabel('Resource␣cluster')
xlabel('Timeslot')
%legend();

f5 = figure(5);
clf(f5);
%hold on;
sz=size(s.C_ideal);
heatmap(s.C_ideal);
title('Resource␣Allocation␣for␣each␣User␣Ideal')
ylabel('Resource␣cluster')
```

```
xlabel('Timeslot')


%f1 = figure(1);
%clf(f1);
% set(f1, 'Color', [1 1 0.1])
%bar(s.C(:,1),1,'g')
%title('Resource Allocation for the first Symbol Period')
%xlabel('Subchannels/Subcarriers')
%ylabel('Users')
```

B.5 losstest.m

```
function loss_test(d0, N)
    d0 = str2num(d0);
    N = str2num(N);
    output = [];
    output2 = [];
    output3 = [];
    output4 = [];
    th1 = [];
    th2 = [];
    th3 = [];
    th4 = [];
    th5 = [];
    th6 = [];
    ds_array = linspace(0,1,25);
    var = 1.5;
    k = 50; % inner loop count
```

```
for ds = ds_array
    percentage = 0;
    percentage2 = 0;
    percentage3 = 0;
    percentage4 = 0;
    t1 = 0;
    t2 = 0;
    t3 = 0;
    t4 = 0;
    t5 = 0;
    t6 = 0;
    for i = 1:k
        s.Rb_size=12;
        s.Nu = N;
        p_aloc_rate = s.Nu;
        s.Nsc = s.Nu*s.Rb_size;
        s.RBs = floor(s.Nsc/s.Rb_size/s.Nu);
        s.Nclu=floor(s.Nsc/(s.RBs*s.Rb_size));
        s.alpha = randi([2,s.Nclu-1],1);
        s.rx = randi([0,s.Nclu-1],1);
        s.ry = randi([0,s.Nclu-1],1);
        s.Kc = randi([0,10000000],1);
        s.Pt=1000;
        fprintf("RBs %d Nclu %d alpha %d rx %d ry %d Kc %d\n", s.RBs, s.
            ↪ Nclu, s.alpha, s.rx, s.ry, s.Kc);
        s.csi = csi_gen(s.Nu,s.Nsc,var,ds,d0);
        [s.L,s.C,s.L_nh,s.C_nh,s.C_ideal] = chaoticmap(s.Nu,s.Nsc,s.RBs,
            ↪ s.alpha,s.rx,s.ry,s.Kc,s.Rb_size,s.csi);
```

```
[s.Pn_opt,s.csi_ra,s.Cn] = waterfilling(s.csi,s.C,s.RBs,s.Pt,s.
    ↪ Rb_size,p_aloc_rate);
[s.Pn_opt2,s.csi_ra2,s.Cn2] = waterfilling(s.csi,s.C_nh,s.RBs,s.
    ↪ Pt,s.Rb_size,p_aloc_rate);
[s.Pn_opt3,s.csi_ra3,s.Cn3] = waterfilling(s.csi,s.C_ideal,s.RBs
    ↪ ,s.Pt,s.Rb_size,p_aloc_rate);
[s.Pn_opt4,s.csi_ra4,s.Cn4] = waterfilling(s.csi,s.C_ideal,s.RBs
    ↪ ,s.Pt,s.Rb_size,1);
[s.Pn_opt5,s.csi_ra5,s.Cn5] = waterfilling(s.csi,s.C_nh,s.RBs,s.
    ↪ Pt,s.Rb_size,1);
[s.Pn_opt6,s.csi_ra6,s.Cn6] = waterfilling(s.csi,s.C,s.RBs,s.Pt,
    ↪ s.Rb_size,1);

s.count = zeros(1,s.Nu);
for i=1:s.Nu
    s.count(i)=sum(sum(s.C==i));
    %fprintf("user %d clusters %d\n", i, s.count(i));
end

%use the throughput for user 1
s.Th=sum(cell2mat(s.Cn))/s.Nu/s.Nclu;
fprintf("FH average throughput per slot %f bps/Hz\n", s.Th);
s.Th2=sum(cell2mat(s.Cn2))/s.Nu/s.Nclu;
s.Nts=size(s.C,2);
fprintf("Non FH average throughput per slot %f bps/Hz\n", s.Th2)
    ↪ ;
s.Th3=sum(cell2mat(s.Cn3))/s.Nu/s.Nclu;
```

```
    fprintf("Ideal FH average throughput per slot %f bps/Hz\n", s.
        ↪ Th3)
    s.Th4=sum(cell2mat(s.Cn4))/s.Nu/s.Nclu;
    fprintf("Ideal FH Continous Power average throughput per slot %f
        ↪  bps/Hz\n", s.Th4)
    s.Th5=sum(cell2mat(s.Cn5))/s.Nu/s.Nclu;
    fprintf("Ideal FH Continous Power average throughput per slot %f
        ↪  bps/Hz\n", s.Th5)
    s.Th6=sum(cell2mat(s.Cn6))/s.Nu/s.Nclu;
    fprintf("FH Continous Power average throughput per slot %f bps/
        ↪ Hz\n", s.Th6)
    %fprintf("Hopping Throughput Loss %f%%\n", 100.0-(s.Th/s.Th2)
        ↪ *100.0);
    percentage = percentage + (100.0-(abs(s.Th)/abs(s.Th2))*100.0);
    percentage2 = percentage2 + (100.0-(abs(s.Th)/abs(s.Th3))*100.0)
        ↪ ;
    percentage3 = percentage3 + (100.0-(abs(s.Th)/abs(s.Th4))*100.0)
        ↪ ;
    percentage4 = percentage4 + (100.0-(abs(s.Th)/abs(s.Th5))*100.0)
        ↪ ;
    %output = [output s];
    t1 = t1 + s.Th;
    t2 = t2 + s.Th2;
    t3 = t3 + s.Th3;
    t4 = t4 + s.Th4;
    t5 = t5 + s.Th5;
    t6 = t6 + s.Th6;
end
```

```
percentage = percentage/k;
percentage2 = percentage2/k;
percentage3 = percentage3/k;
percentage4 = percentage4/k;
t1 = t1/k;
t2 = t2/k;
t3 = t3/k;
t4 = t4/k;
t5 = t5/k;
t6 = t6/k;
fprintf("After %d runs, for ds = %f:\n", k, ds);
fprintf("CSM vs Non Hopping Throughput Loss %f%%\n", percentage);
fprintf("CSM vs Ideal Hopping Throughput Loss %f%%\n", percentage2);
fprintf("CSM vs Ideal Hopping Continuous Power Allocation Throughput
    ↪ Loss %f%%\n", percentage3);
fprintf("CSM vs Non Hopping Continuous Power Allocation Throughput
    ↪ Loss %f%%\n", percentage4);
output = [output percentage];
output2 = [output2 percentage2];
output3 = [output3 percentage3];
output4 = [output4 percentage4];
th1 = [th1 t1];
th2 = [th2 t2];
th3 = [th3 t3];
th4 = [th4 t4];
th5 = [th5 t5];
th6 = [th6 t6];
```

```
        f = sprintf('loss_comp_N%d_d0_0p%d.mat', N, round(10*abs(d0 - fix(d0
            ↪ ))));
        save(f, 'd0', 'ds_array', 'output', 'output2', 'output3', 'output4',
            ↪ 'N', 'k', 'th1', 'th2', 'th3', 'th4', 'th5', 'th6')
    end
    %figure(1);
    %plot(ds_array, output);
    %title('Hopping Throughput Loss vs d_{s}');
    %ylabel('Througput Loss %');
    %xlabel('d_{s}');
end
```

B.6 nrdownlinkgen.m

```
%% 5G NR Downlink Vector Waveform Generation
% This example shows how to configure and generate a 5G NR downlink vector
% waveform for a baseband component carrier by using the
% <docid:5g_ref#mw_function_nrWaveformGenerator nrWaveformGenerator>
    ↪ function.

% Copyright 2018-2021 The MathWorks, Inc.

%% Introduction
% This example shows how to parameterize and generate a 5G New Radio (NR)
% downlink waveform by using the |nrWaveformGenerator| function. The
    ↪ generated
% waveform contains these channels and signals:
%
```

```
% * PDSCH and its associated DM-RS and PT-RS
% * PDCCH and its associated DM-RS
% * PBCH and its associated DM-RS
% * PSS and SSS
% * CSI-RS
%
% This example demonstrates how to parameterize and generate a baseband
% component carrier waveform characterized by multiple subcarrier spacing
% (SCS) carriers and bandwidth parts (BWP). You can generate multiple
% instances of the physical downlink shared channel (PDSCH), the physical
% downlink control channel (PDCCH), and the channel state information
% reference signal (CSI-RS) over the different BWPs. You can configure sets
% of control resource sets (CORESETs) and search space monitoring
% opportunities for mapping the PDCCHs. This example does not apply
% precoding to the physical channels and signals.

%% Waveform and Carrier Configuration
% The baseband waveform is parameterized by the
% <docid:5g_ref#mw_object_nrDLCarrierConfig nrDLCarrierConfig> object and
% a set of additional objects associated with the waveform channels and
% signals.
%
% With the |nrDLCarrierConfig| object, you can set these downlink carrier
% configuration parameters.
%
% * Label for this DL carrier configuration
% * SCS carrier bandwidth in resource blocks
% * Carrier cell ID
```

```
% * Length of the generated waveform in subframes
% * Windowing
% * Sample rate of the OFDM modulated waveform
% * Carrier frequency for symbol phase compensation
%
% You can control SCS carrier bandwidths and guardbands using the |
    ↪ NStartGrid|
% and |NSizeGrid| properties of the <docid:5g_ref#
    ↪ mw_object_nrSCSCarrierConfig
% nrSCSCarrierConfig> object.

waveconfig = nrDLCarrierConfig(); % Create an instance of the waveform's
    ↪ parameter object
waveconfig.Label = 'DL␣carrier␣1'; % Label for this downlink waveform
    ↪ configuration
waveconfig.NCellID = 0; % Cell identity
waveconfig.ChannelBandwidth = 40; % Channel bandwidth (MHz)
waveconfig.FrequencyRange = 'FR1'; % 'FR1' or 'FR2'
waveconfig.NumSubframes = 60; % Number of 1ms subframes in generated
    ↪ waveform (1,2,4,8 slots per 1ms subframe, depending on SCS)
waveconfig.WindowingPercent = 0; % Percentage of windowing relative to FFT
    ↪ length
waveconfig.SampleRate = []; % Sample rate of the OFDM modulated waveform
waveconfig.CarrierFrequency = 0; % Carrier frequency in Hz. This property
    ↪ is used for symbol phase
                                % compensation before OFDM modulation, not
                                    ↪ for upconversion
```

```
% Define a set of SCS specific carriers, using the maximum sizes for a
% 40 MHz NR channel. See TS 38.101-1 for more information on defined
% bandwidths and guardband requirements
scscarriers = {nrSCSCarrierConfig(),nrSCSCarrierConfig()};
scscarriers{1}.SubcarrierSpacing = 15;
scscarriers{1}.NSizeGrid = 216;
scscarriers{1}.NStartGrid = 0;

scscarriers{2}.SubcarrierSpacing = 30;
scscarriers{2}.NSizeGrid = 106;
scscarriers{2}.NStartGrid = 1;

%% SS Burst
% In this section you can set the parameters for the signal synchronization
% (SS) burst. The numerology of the SS burst can be different from other
% parts of the waveform. This is specified via the block pattern parameter,
% as specified in TS 38.213 Section 4.1. A bitmap specifies the blocks to
% transmit in a 5 ms half-frame burst. You can also set the periodicity in
% milliseconds and the power of the burst. For a full list of configurable
% SS burst properties, see <docid:5g_ref#mw_object_nrWavegenSSBurstConfig
% nrWavegenSSBurstConfig>.

% SS burst configuration
ssburst = nrWavegenSSBurstConfig();
ssburst.Enable = 1; % Enable SS Burst
ssburst.Power = 0; % Power scaling in dB
ssburst.BlockPattern = 'Case␣B'; % Case B (30kHz) subcarrier spacing
```

```
ssburst.TransmittedBlocks = [1 1 1 1]; % Bitmap indicating blocks
    ↪ transmitted in a 5ms half-frame burst
ssburst.Period = 20; % SS burst set periodicity in ms (5, 10, 20, 40, 80,
    ↪ 160)
ssburst.NCRBSSB = []; % Frequency offset of SS burst (CRB), use [] for the
    ↪ waveform center

%% Bandwidth Parts
% A BWP is formed by a set of contiguous resources sharing a numerology on
% a given carrier. You can define multiple BWPs using a cell array. Each
% element in the cell array of <docid:5g_ref#mw_object_nrWavegenBWPConfig
% nrWavegenBWPConfig> objects defines a BWP. For each BWP, you can specify
% the SCS, the cyclic prefix (CP) length and the bandwidth. The
% |SubcarrierSpacing| property maps the BWP to one of the SCS specific
% carriers defined earlier. The |NStartBWP| property controls the location
% of the BWP in the carrier, relative to point A. This is expressed in
% common resource blocks (CRB) in terms of the BWP numerology. Different
% BWPs can overlap with each other.
%
% <<../bwp.png>>

% Bandwidth parts configurations
bwp = {nrWavegenBWPConfig(),nrWavegenBWPConfig()};
%bwp = {nrWavegenBWPConfig()};
bwp{1}.BandwidthPartID = 1; % Bandwidth part ID
bwp{1}.Label = 'BWP␣@␣15␣kHz'; % Label for this BWP
bwp{1}.SubcarrierSpacing = 15; % BWP subcarrier spacing
bwp{1}.CyclicPrefix = 'Normal'; % BWP cyclic prefix for 15 kHz
```

```
bwp{1}.NSizeBWP = 100; % Size of BWP in PRBs
bwp{1}.NStartBWP = 12; % Position of BWP, relative to point A (i.e. CRB)

 bwp{2}.BandwidthPartID = 2; % Bandwidth part ID
 bwp{2}.Label = 'BWP␣@␣30␣kHz'; % Label for this BWP
 bwp{2}.SubcarrierSpacing = 30; % BWP subcarrier spacing
 bwp{2}.CyclicPrefix = 'Normal'; % BWP cyclic prefix for 30 kHz
 bwp{2}.NSizeBWP = 61; % Size of BWP in PRBs
 bwp{2}.NStartBWP = 31; % Position of BWP, relative to point A (i.e. CRB)

%% CORESET and Search Space Configuration
% This section specifies the CORESET and the PDCCH search space
% configuration. The CORESET and search spaces specify the possible
% locations (in time and frequency) of the control channel transmissions
% for a given numerology. Each element in the cell array of
% <docid:5g_ref#mw_object_nrCORESETConfig nrCORESETConfig> objects defines
% a CORESET and each element in the cell array of
% <docid:5g_ref#mw_object_nrSearchSpaceConfig nrSearchSpaceConfig> objects
% defines a search space.
%
% Set these parameters for each CORESET and search space:
%
% * The OFDM symbols which specify the first symbol of each CORESET
% monitoring opportunity in a slot
% * The duration of the block of allocated slots within a period
% * Periodicity of the allocation pattern
% * The CORESET duration in symbols, either 1, 2 or 3
% * A bitmap defining the allocated physical resource blocks (PRB) of the
```

```
% CORESET. The CORESET frequency allocation is defined in blocks of 6 PRBs,
% aligned in CRB numbering, relative to point A. Each bit in the bitmap
% selects all 6 PRBs in the CRB aligned block that contains it
% * CCE-to-REG mapping which can be 'interleaved' or 'noninterleaved'
% * Resource element group (REG) bundle size (L), either (2,6) or
% (3,6), based on CORESET duration
% * Interleaver size, either 2, 3, or 6
% * Shift index, a scalar value in the range 0...274
%
% The figure below shows the meaning of some of the CORESET parameters.
%
% <<../coresetAlloc.png>>

% CORESET and search space configurations
coresets = {nrCORESETConfig()};
coresets{1}.CORESETID = 1; % CORESET ID
coresets{1}.Duration = 3; % CORESET symbol duration (1,2,3)
coresets{1}.FrequencyResources = [1 1 0 1]; % Bitmap indicating blocks of 6
    ↪  PRB for CORESET (RRC - frequencyDomainResources)
coresets{1}.CCEREGMapping = 'noninterleaved'; % Mapping: 'interleaved' or '
    ↪ noninterleaved'
coresets{1}.REGBundleSize = 3; % L (2,6) or (3,6)
coresets{1}.InterleaverSize = 2; % R (2,3,6)
coresets{1}.ShiftIndex = waveconfig.NCellID; % Set to NCellID

searchspaces = {nrSearchSpaceConfig()};
searchspaces{1}.SearchSpaceID = 1; % Search space ID
searchspaces{1}.CORESETID = 1; % CORESET associated with this search space
```

```
searchspaces{1}.SearchSpaceType = 'ue'; % Search space type, 'ue' or '
    ↪ common'
searchspaces{1}.SlotPeriodAndOffset = [5,0]; % Allocated slot period and
    ↪ slot offset of search space pattern
searchspaces{1}.Duration = 2; % Number of slots in the block of slots in
    ↪ pattern period
searchspaces{1}.StartSymbolWithinSlot = 0; % First symbol of each CORESET
    ↪ monitoring opportunity in a slot
searchspaces{1}.NumCandidates = [8 8 4 2 0]; % Number of candidates at each
    ↪  AL (set to 0 if the AL doesn't fit in CORESET)

%% PDCCH Instances Configuration
% This section specifies the the set of PDCCH instances in the waveform by
% using a cell array. Each element in the cell array of
% <docid:5g_ref#mw_object_nrWavegenPDCCHConfig nrWavegenPDCCHConfig>
% objects defines a sequence of PDCCH instances.
%
% Set these parameters for each PDCCH sequence:
%
% * Enable or disable this PDCCH sequence
% * Specify a label for this PDCCH sequence
% * Specify the BWP carrying the PDCCH. The PDCCH uses the SCS specified
% for this BWP
% * Power scaling in dB
% * Enable or disable downlink control information (DCI) channel coding
% * Allocated search spaces within the CORESET monitoring occasion sequence
% * Search space (and CORESET) that carries the PDCCH instances
% * Period of the allocation in slots. Empty period indicates no repetition
```

```
% * The aggregation level (AL) of the PDCCH (number of control channel
    ↪ elements (CCEs))
% * The allocated candidate which specifies the CCE used for the
% transmission of the PDCCH
% * RNTI
% * Scrambling NID for this PDCCH and its associated DM-RS
% * DM-RS power boosting in dB
% * DCI message payload size
% * DCI message data source. You can use an array of bits or one of these
% standard PN sequences: |'PN9-ITU'|, |'PN9'|, |'PN11'|, |'PN15'|,
% |'PN23'|. The seed for the generator can be specified using a cell array
% in the form |{'PN9', seed}|. If no seed is specified, the generator is
% initialized with all ones

pdcch = {nrWavegenPDCCHConfig()};
pdcch{1}.Enable = 1 ; % Enable PDCCH sequence
pdcch{1}.Label = 'PDCCH␣@␣15␣kHz'; % Label for this PDCCH sequence
pdcch{1}.BandwidthPartID = 2; % Bandwidth part of PDCCH transmission
pdcch{1}.Power = 1.1; % Power scaling in dB
pdcch{1}.Coding = 1; % Enable DCI coding
pdcch{1}.SearchSpaceID = 1; % Search space
pdcch{1}.SlotAllocation = 1; % Allocated slots indices for PDCCH sequence
pdcch{1}.Period = 5; % Allocation period in slots
pdcch{1}.AggregationLevel = 8; % Aggregation level (1,2,4,8,16 CCEs)
pdcch{1}.AllocatedCandidate = 1; % PDCCH candidate in search space (1 based
    ↪ )
pdcch{1}.RNTI = 0; % RNTI
pdcch{1}.DMRSScramblingID = 1; % PDCCH and DM-RS scrambling NID
```

```
pdcch{1}.DMRSPower = 0; % Additional DM-RS power boosting in dB
pdcch{1}.DataBlockSize = 20; % DCI payload size
pdcch{1}.DataSource = 'PN9'; % DCI data source

%% PDSCH Instances Configuration
% This section specifies the set of PDSCH instances in the waveform by
% using a cell array. Each element in the cell array of <docid:5g_ref#
    ↪ mw_object_nrWavegenPDSCHConfig
% nrWavegenPDSCHConfig> objects defines a sequence of PDSCH instances. This
% example defines two PDSCH sequences.
%
% *General Parameters*
%
% Set these parameters for each PDSCH sequence:
%
% * Enable or disable this PDSCH sequence
% * Specify a label for this PDSCH sequence
% * Specify the BWP carrying the PDSCH. The PDSCH uses the SCS specified
% for this BWP
% * Power scaling in dB
% * Enable or disable the DL-SCH transport channel coding
% * Transport block data source. You can use an array of bits or one of
% these standard PN sequences: |'PN9-ITU'|, |'PN9'|, |'PN11'|, |'PN15'|,
% |'PN23'|. The seed for the generator can be specified using a cell array
% in the form |{'PN9', seed}|. If no seed is specified, the generator is
% initialized with all ones
% * Target code rate used to calculate the transport block sizes
% * Overhead parameter
```

```
% * Symbol modulation
% * Number of layers
% * Redundancy version (RV) sequence
% * Enable or disable the interleaving of the virtual to physical resource
% block mapping
% * Bundle size for the interleaved map, specified by the higher layer
% parameter vrb-ToPRB-Interleaver

pdsch = {nrWavegenPDSCHConfig()};
pdsch{1}.Enable = 1; % Enable PDSCH sequence
pdsch{1}.Label = 'PDSCH␣@␣15␣kHz'; % Label for this PDSCH sequence
pdsch{1}.BandwidthPartID = 1; % Bandwidth part of PDSCH transmission
pdsch{1}.Power = 0; % Power scaling in dB
pdsch{1}.Coding = 1; % Enable the DL-SCH transport channel coding
pdsch{1}.DataSource = 'PN9'; % Channel data source
pdsch{1}.TargetCodeRate = 0.4785; % Code rate used to calculate transport
    ↪ block sizes
pdsch{1}.XOverhead = 0; % Rate matching overhead
pdsch{1}.Modulation = 'QPSK'; % 'QPSK', '16QAM', '64QAM', '256QAM'
pdsch{1}.NumLayers = 2; % Number of PDSCH layers
pdsch{1}.RVSequence = [0,2,3,1]; % RV sequence to be applied cyclically
    ↪ across the PDSCH allocation sequence
pdsch{1}.VRBToPRBInterleaving = 0; % Disable interleaved resource mapping
pdsch{1}.VRBBundleSize = 2; % vrb-ToPRB-Interleaver parameter

%%
% *Allocation*
%
```

```
% This diagram represents some of the parameters used in the PDSCH
    ↪ allocation.
%
% <<../pdschAlloc.png>>
%
% You can set the following parameters to control the PDSCH allocation.
% These parameters are relative to the BWP. The specified PDSCH allocation
% will avoid the locations used for the SS burst.
%
% * Symbols in a slot allocated to each PDSCH instance
% * Slots in a frame used for the sequence of PDSCH
% * Period of the allocation in slots. Empty period indicates no repetition
% * The allocated PRBs relative to the BWP
% * RNTI. This value is used to link the PDSCH to an instance of the PDCCH
% * NID for scrambling the PDSCH bits

pdsch{1}.SymbolAllocation = [0,13]; % First symbol and length
pdsch{1}.SlotAllocation = [0]; % Allocated slot indices for PDSCH sequence
pdsch{1}.Period = 60; % Allocation period in slots
pdsch{1}.PRBSet = [0]; % PRB allocation
pdsch{1}.RNTI = 40; % RNTI
pdsch{1}.NID = 20; % Scrambling for data part

%%
% CORESETs and sets of PRB can be specified for rate matching around, if
    ↪ required
%
% * The PDSCH can be rate matched around one or more CORESETs
```

```
% * The PDSCH can be rate matched around other resource allocations

pdsch{1}.ReservedCORESET = 1; % Rate matching pattern, defined by CORESET
    ↪ IDs
pdsch{1}.ReservedPRB{1}.PRBSet = []; % Rate matching pattern, defined by
    ↪ set of PRB (RRC 'bitmaps')
pdsch{1}.ReservedPRB{1}.SymbolSet = [];
pdsch{1}.ReservedPRB{1}.Period = [];

%%
% *PDSCH DM-RS Configuration*
%
% Set the DM-RS parameters.

% Antenna port and DM-RS configuration (TS 38.211 section 7.4.1.1)
pdsch{1}.MappingType = 'A'; % PDSCH mapping type ('A'(slot-wise),'B'(non
    ↪ slot-wise))
pdsch{1}.DMRSPower = 0; % Additional power boosting in dB

pdsch{1}.DMRS.DMRSConfigurationType = 2; % DM-RS configuration type (1,2)
pdsch{1}.DMRS.NumCDMGroupsWithoutData = 1; % Number of DM-RS CDM groups
    ↪ without data. The value can be one of the set {1,2,3}
pdsch{1}.DMRS.DMRSPortSet = []; % DM-RS antenna ports used ([] gives port
    ↪ numbers 0:NumLayers-1)
pdsch{1}.DMRS.DMRSTypeAPosition = 2; % Mapping type A only. First DM-RS
    ↪ symbol position (2,3)
pdsch{1}.DMRS.DMRSLength = 1; % Number of front-loaded DM-RS symbols (1(
    ↪ single symbol),2(double symbol))
```

```
pdsch{1}.DMRS.DMRSAdditionalPosition = 0; % Additional DM-RS symbol
    ↪ positions (max range 0...3)
pdsch{1}.DMRS.NIDNSCID = 1; % Scrambling identity (0...65535)
pdsch{1}.DMRS.NSCID = 0; % Scrambling initialization (0,1)

%%
% *PDSCH PT-RS Configuration*
%
% Set the PT-RS parameters.

% PT-RS configuration (TS 38.211 section 7.4.1.2)
pdsch{1}.EnablePTRS = 0; % Enable or disable the PT-RS (1 or 0)
pdsch{1}.PTRSPower = 0; % Additional PT-RS power boosting in dB

pdsch{1}.PTRS.TimeDensity = 1; % Time density (L_PT-RS) of PT-RS (1,2,4)
pdsch{1}.PTRS.FrequencyDensity = 2; % Frequency density (K_PT-RS) of PT-RS
    ↪ (2,4)
pdsch{1}.PTRS.REOffset = '00'; % PT-RS resource element offset
    ↪ ('00','01','10','11')
pdsch{1}.PTRS.PTRSPortSet = 0; % PT-RS antenna ports must be a subset of DM
    ↪ -RS ports

%%
% When PT-RS is enabled, the DM-RS ports must be in the range from 0 to 3
% for DM-RS configuration type 1, and in the range from 0 to 5 for DM-RS
% configuration type 2. Nominally, the antenna port of PT-RS is the lowest
% DM-RS port number.
N=61;
```

```
d0 = 1;
ds = 0.5;
var = 1.5;
s.Rb_size=12;
s.Nu = N;
s.Nsc = s.Nu*s.Rb_size;
s.RBs = floor(s.Nsc/s.Rb_size/s.Nu);
s.Nclu=floor(s.Nsc/(s.RBs*s.Rb_size));
s.alpha = randi([2,s.Nclu-1],1);
s.rx = randi([0,s.Nclu-1],1);
s.ry = randi([0,s.Nclu-1],1);
s.Kc = randi([0,10000000],1);
s.Pt=1000;
fprintf("RBs %d Nclu %d alpha %d rx %d ry %d Kc %d\n", s.RBs, s.Nclu, s.
    ↪ alpha, s.rx, s.ry, s.Kc);
s.csi = csi_gen(s.Nu,s.Nsc,var,ds,d0);
[s.L,s.C,s.L_nh,s.C_nh] = chaoticmap(s.Nu,s.Nsc,s.RBs,s.alpha,s.rx,s.ry,s.
    ↪ Kc,s.Rb_size,s.csi);

%%
% *Specifying Multiple PDSCH Instances*
%
% Specify the second PDSCH sequence for the second BWP.

for i =2:61
    pdsch{i} = pdsch{1};
    pdsch{i}.Enable = 1;
    pdsch{i}.Label = 'PDSCH␣@␣30␣kHz';
```

```
    pdsch{i}.BandwidthPartID = 1; % PDSCH mapped to 2nd BWP
    pdsch{i}.SymbolAllocation = [0,13];
end
for i =1:61
    %n = randi(60,[10,1]);
    pdsch{i}.SlotAllocation = [i];
    %pdsch{i}.PRBSet = [i]; % PRB allocation, relative to BWP
    %pdsch{i}.PRBSet = [(find(s.C(:,i)==10))];
    %pdsch{i}.PRBSet = [(find(s.L(:,i)==10))];
    pdsch{i}.PRBSet = [(find(s.C_nh(:,i)==10))];
    %pdsch{i}.PRBSet = [0:60];
end

%% CSI-RS Instances Configuration
% This section configures CSI-RS in the waveform. Each element in the cell
% array of <docid:5g_ref#mw_object_nrWavegenCSIRSConfig
    ↪ nrWavegenCSIRSConfig>
% objects defines a set of CSI-RS resources associated with a BWP. This
% example defines two sets of CSI-RS resources that are disabled.
%
% *General Parameters*
%
% Set these parameters for a set of CSI-RS resources:
%
% * Enable or disable this set of CSI-RS resources
% * Specify a label for this set of CSI-RS resources
% * Specify the BWP carrying this set of CSI-RS resources. The CSI-RS
% resource(s) configuration uses the SCS specified for this BWP
```

```
% * Specify the power scaling in dB. Providing a scalar defines the power
% scaling for a single CSI-RS resource or all configured CSI-RS resources.
% Providing a vector defines a separate power level for each of the CSI-RS
% resources.

csirs = {nrWavegenCSIRSConfig()};
csirs{1}.Enable = 0;
csirs{1}.Label = 'CSI-RS␣@␣15␣kHz';
csirs{1}.BandwidthPartID = 1;
csirs{1}.Power = 3; % Power scaling in dB

%%
% *CSI-RS Configuration*
%
% You can configure these parameters for one or more zero-power (ZP) or
% non-zero-power (NZP) CSI-RS resource configurations.
%
% * Type of CSI-RS resource(s) ('nzp','zp')
% * Row number corresponds to CSI-RS resource(s) as defined in TS 38.211
% Table 7.4.1.5.3-1 (1...18)
% * Frequency density of CSI-RS resource(s). It can be |'one'|, |'three'|,
% |'dot5even'|, or |'dot5odd'|
% * Subcarrier locations of CSI-RS resource(s) within a resource block (RB)
% * Number of RBs allocated to CSI-RS resource(s) (1...275)
% * Starting RB index of CSI-RS resource(s) allocation relative to the
% carrier resource grid (0...274)
% * OFDM symbol locations of CSI-RS resource(s) within a slot
% * The period and offset of slots (0-based) of CSI-RS resource(s). This
```

```
% parameter can be a vector or a cell array of vectors. In the latter case,
% each cell corresponds to an individual CSI-RS resource. In case of a
% vector, the same set of slots is used for all CSI-RS resources
% * Scrambling identity corresponds to CSI-RS resource(s) for pseudo-random
% sequence generation (0...1023)

csirs{1}.CSIRSType = {'nzp','zp'};
csirs{1}.RowNumber = [3 5];
csirs{1}.Density = {'one','one'};
csirs{1}.SubcarrierLocations = {6,4};
csirs{1}.NumRB = 25;
csirs{1}.RBOffset = 12;
csirs{1}.SymbolLocations = {13,9};
csirs{1}.CSIRSPeriod = {[5 0],[5 0]};
csirs{1}.NID = 5;

%%
% *Specifying Multiple CSI-RS Instances*
%
% Specify the second set of CSI-RS resources for the second BWP.

csirs{2} = nrWavegenCSIRSConfig();
csirs{2}.Enable = 0;
csirs{2}.Label = 'CSI-RS␣@␣30␣kHz';
csirs{2}.BandwidthPartID = 1;
csirs{2}.Power = 3; % Power scaling in dB
csirs{2}.CSIRSType = {'nzp','nzp'};
csirs{2}.RowNumber = [1 1];
```

```
csirs{2}.Density = {'three','three'};
csirs{2}.SubcarrierLocations = {0,0};
csirs{2}.NumRB = 50;
csirs{2}.RBOffset = 50;
csirs{2}.SymbolLocations = {6,10};
csirs{2}.CSIRSPeriod = {[10,1],[10,1]};
csirs{2}.NID = 0;

%% Waveform Generation
% This section assigns all the channel and signal parameters into the main
% carrier configuration object |nrDLCarrierConfig|, then generates and
% plots the waveform.

waveconfig.SSBurst = ssburst;
waveconfig.SCSCarriers = scscarriers;
waveconfig.BandwidthParts = bwp;
waveconfig.CORESET = coresets;
waveconfig.SearchSpaces = searchspaces;
waveconfig.PDCCH = pdcch;
waveconfig.PDSCH = pdsch;
waveconfig.CSIRS = csirs;

% Generate complex baseband waveform
[waveform,info] = nrWaveformGenerator(waveconfig);

%%
% Plot the magnitude of the baseband waveform for the set of antenna ports
    ↪ defined.
```

```
figure;
plot(abs(waveform));
title('Magnitude␣of␣5G␣Downlink␣Baseband␣Waveform');
xlabel('Sample␣Index');
ylabel('Magnitude');

%%
% Plot spectogram of waveform for first antenna port.

samplerate = info.ResourceGrids(1).Info.SampleRate;
nfft = info.ResourceGrids(1).Info.Nfft;
figure;
spectrogram(waveform(:,1),ones(nfft,1),0,nfft,'centered',samplerate,'yaxis'
    ↪ ,'MinThreshold',-130);
title('Spectrogram␣of␣5G␣Downlink␣Baseband␣Waveform');

%%
% The waveform generator function returns the time-domain waveform and a
% structure |info|, which contains the underlying resource element grid and
% a breakdown of the resources used by all the PDSCH and PDCCH instances in
% the waveform.
%
% The |ResourceGrids| field is structure array, which contains these fields
    ↪ :
%
% * The resource grid corresponding to each BWP
% * The resource grid of the overall bandwidth containing the channels and
```

```
% signals in each BWP
% * An info structure with information corresponding to each BWP. The
% contents of this info structure for the first BWP are shown below.

disp('Modulation information associated with BWP 1:')
disp(info.ResourceGrids(1).Info)

%%
% Note that the generated resource grid is a 3D matrix where the different
% planes represent the antenna ports. For the different physical channels
% and signals the lowest port is mapped to the first plane of the grid.
```

Bibliography

Bibliography

[1] M. J. Martinez, " 5G LPI Communications ," 2021, ECE 798 Research Project.

[2] X. Lin, D. Yu, and H. Wiemann, "A Primer on Bandwidth Parts in 5G New Radio," *CoRR*, vol. abs/2004.00761, 2020. [Online]. Available: https://arxiv.org/abs/2004.00761

[3] J. Jung and J. Lim, "Chaotic Standard Map Based Frequency Hopping OFDMA for Low Probability of Intercept," *IEEE Communications Letters*, vol. 15, no. 9, pp. 1019–1021, 2011.

[4] 3GPP, " 5G; NR; Specifications ," 3rd Generation Partnership Project (3GPP), Technical Specification (TS), 2022, version 16.8.0. [Online]. Available: https://www.3gpp.org/specifications/specification-numbering

[5] MathWorks, "5G Toolbox- MATLAB," last accessed 25 June 2022. [Online]. Available: https://www.mathworks.com/products/5g.html

[6] L. Deng, T. Kawamura, H. Taoka, and M. Sawahashi, "Combined Effect of Transmit Diversity and Frequency Hopping for DFT-Precoded OFDMA in Uplink Frequency-Selective Fading Channels," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011, pp. 1–5.

[7] A. Oturak and E. Öztürk, "Performance of fast frequency hopping OFDM under Doppler spread," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2013, pp. 884–888.

[8] S. Zhou and G. Giannkis, "Generalized frequency hopping OFDMA through unknown frequency-selective multipath channels," in *2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540)*, vol. 1, 2000, pp. 56–60 vol.1.

[9] S. Yan, X. Zhou, J. Hu, and S. V. Hanly, "Low Probability of Detection Communication: Opportunities and Challenges," *IEEE wireless communications*, vol. 26, no. 5, pp. 19–25, 2019.

[10] B. N. Corporation, "Understanding and Calculating Probability of Intercept," last accessed 13 August 2022. [Online]. Available: https://www.berkeleynucleonics.com/sites/default/files/products/resources/7550-app-poi-10-23-17.pdf

[11] H. Urkowitz, "Energy detection of unknown deterministic signals," *Proceedings of the IEEE*, vol. 55, no. 4, pp. 523–531, 1967.

[12] B. Gardner and J. Roth, "An Efficient Methodology to De-Anonymize the 5G-New Radio Physical Downlink Control Channel," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 2917–2921.

[13] J. Jung, J. Lim, S. Park, H. Kang, and S. Kwon, "Analysis of Low Probability of Detection Capability for Chaotic Standard Map-Based FH-OFDMA System," *Applied sciences*, vol. 11, no. 5, pp. 2198–, 2021.

Curriculum Vitae

Joshua Weitz received his Bachelor of Science in Electrical Engineering from George Mason University in 2013. He has worked in embedded software and digital signal processing for the last 13 years.