Wendy Ide
03/26/18
ECE-408
Professor Hoerning

Standards Project – LTE

1. Introduction

The goal of this project was to simulate a wireless standard. For the simulation, LTE was chosen because in the evolution of wireless cellular standards, all paths over history eventually led to some form of LTE or a WiMax alternative. Hence, as of today, LTE is the most relevant and ubiquitous cellular standard. LTE is specifically designed for packet data communications, where the emphasis of the technology is high spectral efficiency, high peak data rates, low latency, and frequency flexibility [7]. LTE was made possible by utilizing recent DSP techniques developed around the turn of the century, and by simplifying the network architecture to an all IP-based system.

2. Specifications of LTE

This project is a MatLab simulation of the wireless standard that is colloquially known as 4G-LTE or LTE. The standard will be simulated using the Technical Specification (TS) 36 series Release 8 documents from the 3GPP website. Release 8 was chosen because it was frozen in December 2008 and was the basis for the first wave of LTE equipment. The final version of Release 8 will be used for all subdocuments of the TS 36 series referenced.

The main highlights [7] of LTE Release 8 are:

- Up to 300 Mbps downlink and 75 Mbps uplink
- Latency as low as 10 ms
- Bandwidth sized in 1.4, 3, 5, 10, 15, or 20 MHz blocks to allow for a variety of deployment scenarios
- Orthogonal frequency domain multiple access (OFDMA) downlink
- Single-carrier frequency domain multiple access (SC-FDMA) uplink
- Multiple-input multiple-output (MIMO) antennas
- Flat radio network architecture, with no equivalent to the GSM base station controller (BSC) or UMTS radio network controller (RNC), and functionality pushed down and distributed among the base stations (enhanced NodeBs). The flatter architecture results in lower latency.
- All IP core network - System Architecture Evolution (SAE)

The two main documents referenced throughout this report are

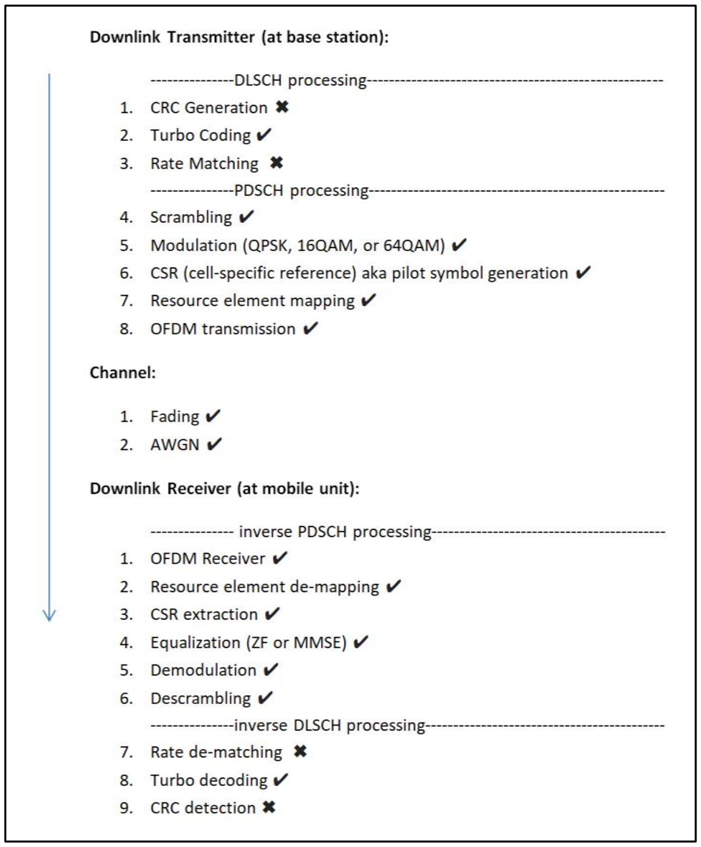
3GPP TS 36.211 V8.9.0 (2009-12): detailing the Physical Channels and Modulation. [1]

3GPP TS 36.212 V8.8.0 (2009-12): detailing the Multiplexing and Channel Coding. [2]

3. Overall Simulation Design

The simulation was performed for the case of a SISO downlink.

The overall components of the system that were implemented in accordance with the 3GPP standard are shown as follows:



Downlink Transmitter (at base station):

---------------DLSCH processing---

1. CRC Generation ✖
2. Turbo Coding ✔
3. Rate Matching ✖

---------------PDSCH processing---

4. Scrambling ✔
5. Modulation (QPSK, 16QAM, or 64QAM) ✔
6. CSR (cell-specific reference) aka pilot symbol generation ✔
7. Resource element mapping ✔
8. OFDM transmission ✔

Channel:

1. Fading ✔
2. AWGN ✔

Downlink Receiver (at mobile unit):

--------------- inverse PDSCH processing---

1. OFDM Receiver ✔
2. Resource element de-mapping ✔
3. CSR extraction ✔
4. Equalization (ZF or MMSE) ✔
5. Demodulation ✔
6. Descrambling ✔

---------------inverse DLSCH processing--

7. Rate de-matching ✖
8. Turbo decoding ✔
9. CRC detection ✖

Figure 1: Overall System

Note that although CRC detection and Rate Matching were not implemented according to the standard guidelines; some custom preprocessing was used for maximum utilization of the resource grids. This will be discussed later on.

LTE Release 8 can either use FDD or TDD for allocating the channel to uplink and downlink. Since this simulation was only simulating downlink, this was not a concern. In the simulation, a long bit stream was mapped to multiple sub frames and frames. This was successful in simulating BER. One could argue running the simulation with different lengths bit streams while changing the range of the slot numbers (2 slots per sub frame) is like allocating different parts of the resource grid to different users in time.

The focus of the simulation was mainly the PDSCH (PHY layer) and parts of the DLSCH (transport channel).

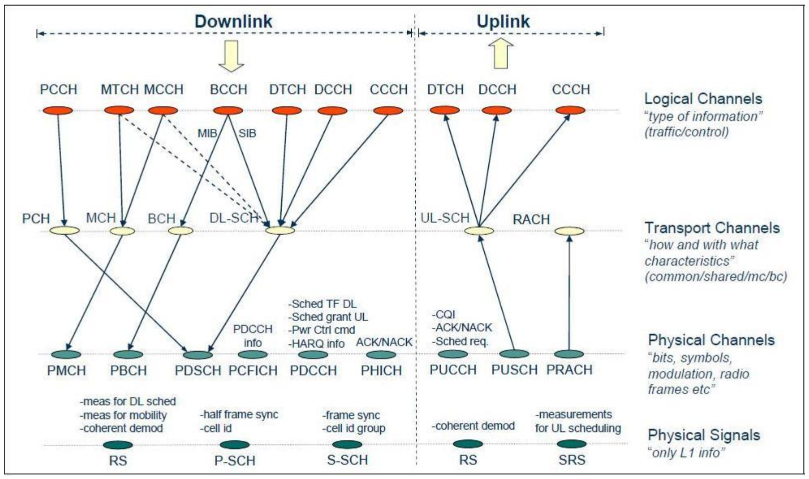


Figure 2: Overview of LTE channel structure and mapping

3.1 LTE Resource Grid

The LTE resource grid structure is shown in the following figures. There are 10 subframers per frame. Each subframe is divided into 2 slots. Each subframe contains 14 OFDM symbols. There is one resource block per slot. One resource block is made up of 12x7 = 84 resource elements, or 7 OFDM symbols. Different bandwidths use different numbers of resource blocks. The example in Figure 5 is for a 1.4 MHz bandwidth, which uses 6 resource blocks. Therefore the number of resource elements in one *sub*frame for 1.4 MHz is 12*6*14 = 1,008.

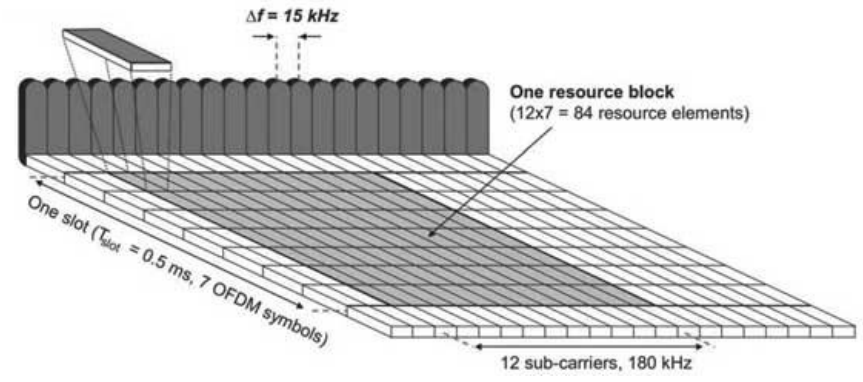

Δf = 15 kHz
One resource block
(12x7 = 84 resource elements)
One slot (Tslot = 0.5 ms, 7 OFDM symbols)
12 sub-carriers, 180 kHz

Figure 3: One Resource Block

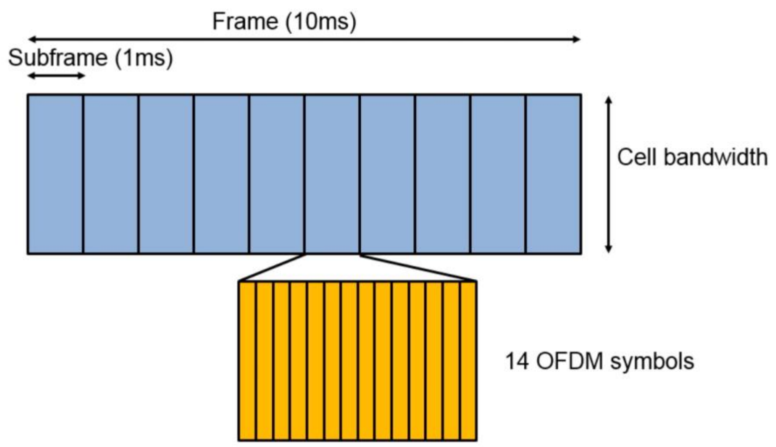


Figure 4: One Frame General

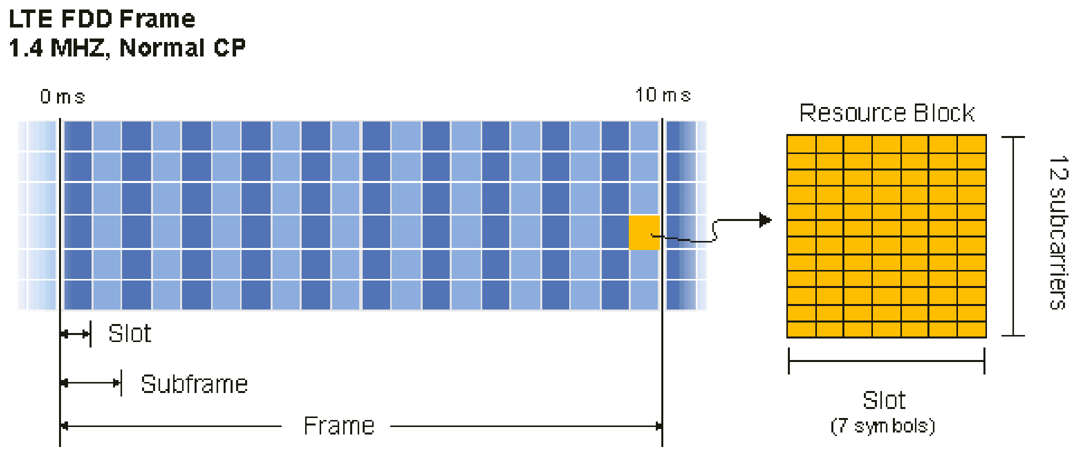


Figure 5: One Frame Detailed

4. Methods and Code Modules

4.1 Transmitter

-File: prms.PDSCH.m

The transmitter properties are stored in the object prmLTE which is generated in the function prmsPDSCH. Parameters defined by the standard include the bandwidth, number of resource blocks, cyclic prefix lengths, channel sampling rate, FFT length, subcarrier spacing (defaulted to 15kHz), number of subcarriers per resource block (defaulted to 12), and number of OFDM symbols in a slot (defaulted to 7). Since this is a SISO case many other parameters do not have to be stored.

- Note that `chanBW` inputs of {1,2,3,4,5,6} correspond to {1.4 MHz, 3 MHz,5 MHz,10 MHz,15 MHz, 20 MHz).
- Note than `modType` inputs of {1,2,3} correspond to {QPSK, 16QAM, 64QAM}.
- `contReg` (control region) determines the total number of OFDM symbols (in one sub frame) reserved for the PDCCH (that carries the DCI). In more detail, the PDCCH takes up the first to the third time-domain columns of each sub frame. The first column is shared with CSR’s.
    - `contReg` is defaulted to 1 in the functions SISO_E2E.m and SISO_E2Efun.m, but can be changed.
- `NCellID` is always defaulted to 0.
- The code is written in terms of the slot number `nS`. Since there are 2 slots per subframe and 10 subframes in 1 frame, `nS` goes from (0 to 20 by 2) modulo 20. That is, slot 10 corresponds to subframe 5, slot 20 corresponds to sub frame 0, etc.
- LTE is 0 indexed, but for the simulation everything was adjusted to 1 indexed to adhere to Matlab standard.
- Assumed normal Cyclic Prefix everywhere.

4.1,1 Turbo Encoder:

-File: TurboEncoder.m

-File: IntrlvrIndices.m

Turbo coders were a natural choice for LTE given the near-Shannon-bound performance of turbo coders, their excellent BER performance compared to other types of convolutional coders, and that they lend themselves to adaptation [5].

According to section 5.1.3.2.1 of TS 36.212:

The scheme of turbo encoder is a Parallel Concatenated Convolutional Code (PCCC) with two 8-state constituent encoders and one turbo code internal interleaver. The coding rate of turbo encoder is 1/3. The structure of turbo encoder is illustrated in the below figure.

The transfer function of the 8-state constituent code for the PCCC is:

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D)}\right],$$

where

$$g_0(D) = 1 + D^2 + D^3,$$
$$g_1(D) = 1 + D + D^3.$$

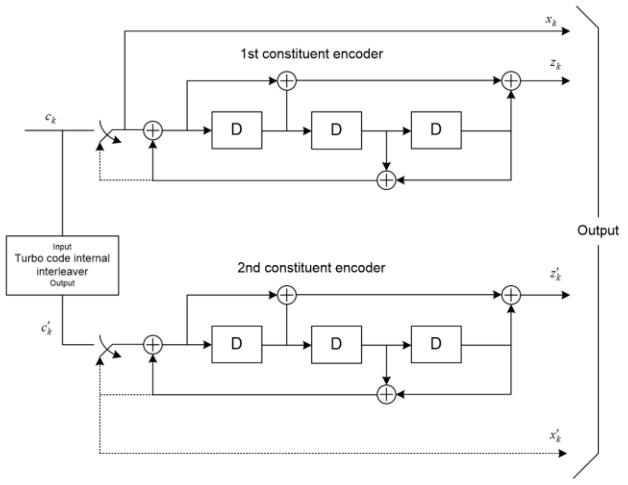


Figure 6 Structure of Turbo Encoder

Each of three streams is terminated by four tail bits. So if the input of the coder is x bits, the output is 3x+12 bits long. The octal representations of 1011 and 1101 are 13 and 15 respectively. The encoder has a constraint length of 4 and feedback connection polynomial of length 13. The simulation uses the MatLab default of poly2trellis(4, [13 15], 13) for the Trellis structure.

The interleaver ensures that two permutations of the same input data are encoded to produce two different parity sequences, hence ensuring robust performance in an unknown channel. The LTE interleaver is based on a simple Quadratic Polynomial Permutation (QPP) scheme. The interleaver permutes the indices of the input bits. The relationship between the output index Π (*i*) and the input index *i* is described by the following quadratic polynomial expression:

$$\Pi(i) = \left(f_1 \cdot i + f_2 \cdot i^2\right) \bmod K$$

LTE allows 188 different values for the input block size *K*. The smallest block size is 40 and largest is 6144. The parameters f_1 and f_2 depend on the block size *K* and are summarized in Table 5.1.3-3 of Section 5.1.3.2.3 of TS 36.212. This table is stored in the variable Iparams.mat.

4.1.2 Scrambling

-File: Scrambler.m

Different scrambling sequences are used in neighboring cells to ensure that the interference is randomized and that transmission from different cells are separated prior to decoding. In order to achieve this, data bits are scrambled with a sequence that is unique to each cell by initializing the sequence generators in the cell based on the PHY cell identity. There are 504 cell identities defined in LTE, organized into 168 groups, each of which contains three unique identities [5]. As stated previously, in this simulation `NCellID` is always defaulted to 0.

After Turbo coding the bits were scrambled with a Gold Sequence. Pseudo-random sequences are defined by a length-31 Gold sequence as detailed in TS 36.211 Section 7.2:

The output sequence $c(n)$ of length M_{PN}, where $n = 0,1,\ldots,M_{\mathrm{PN}} - 1$, is defined by

$$\begin{aligned} c(n) &= \left(x_1(n+N_C) + x_2(n+N_C)\right) \bmod 2 \\ x_1(n+31) &= \left(x_1(n+3) + x_1(n)\right) \bmod 2 \\ x_2(n+31) &= \left(x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)\right) \bmod 2 \end{aligned}$$

where $N_C = 1600$ and the first m-sequence shall be initialized with $x_1(0) = 1, x_1(n) = 0, n = 1,2,\ldots,30$. The initialization of the second m-sequence is denoted by $c_{\mathrm{init}} = \sum_{i=0}^{30} x_2(i) \cdot 2^i$ with the value depending on the application of the sequence.

$$c_{\mathrm{init}} = \begin{cases} n_{\mathrm{RNTI}} \cdot 2^{14} + q \cdot 2^{13} + \lfloor n_s/2 \rfloor \cdot 2^9 + N_{\mathrm{ID}}^{\mathrm{cell}} & \text{for PDSCH} \\ \lfloor n_s/2 \rfloor \cdot 2^9 + N_{\mathrm{ID}}^{\mathrm{MBSFN}} & \text{for PMCH} \end{cases}$$

Where n_{RNTI} corresponds to the RNTI (Radio Network Temporary Identifier) associated with the PDSCH transmission as described in Section 7.1 of TS 36.213. Since no CRC was used, n_{RNTI} was set to 1.

Up to two code words can be transmitted in one subframe, i.e., q = {0,1} . In the case of single code word transmission, q is equal to zero.

The output sequence is defined as the output of an exclusive-or operation applied to a specified pair of sequences. The MatLab object `comm.GoldSequence` was used.

4.1.2 Modulation

-File: Modulator.m

For PDSCH, Section 6.3.2 of TS 36.211 specifies modulation of QPSK, 16QAM, or 64 QAM. The different modulations are accomplished using MatLab system objects with custom symbol mapping to match the symbols specified in Sections 7.1.2-7.1.4. of TS 36.211

4.1.3 CSR (Cell Specific Reference) Symbol Generation

-File: CSRgenerator.m

From Section 6.10.1.1 of TS 36.211, the reference-signal sequence   is defined by:

$$r_{l,n_s}(m) = \frac{1}{\sqrt{2}}\left(1-2\cdot c(2m)\right) + j\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m+1)\right), \quad m = 0,1,\ldots,2N_{\mathrm{RB}}^{\max,\mathrm{DL}}-1$$

where n_s is the slot number within a radio frame and $r_{l,n_s}(m)$ is the OFDM symbol number within the slot. The pseudo-random sequence is the same Gold sequence used for the scrambler. The pseudo-random sequence generator shall be initialized with $c_{\mathrm{init}} = 2^{10}\cdot\left(7\cdot\left(n_s+1\right)+l+1\right)\cdot\left(2\cdot N_{ID}^{cell}+1\right)+2\cdot N_{ID}^{cell}+N_{CP}$ at the start of each OFDM symbol where

$$N_{CP} = \begin{cases} 1 & \text{for normal CP} \\ 0 & \text{for extended CP} \end{cases}$$

4.1.4 Resource Element Mapping

-File: Remapper_1Tx.m

-Uses helper function ExpungeFrom.m

The LTE resource grid is mapped as follows:

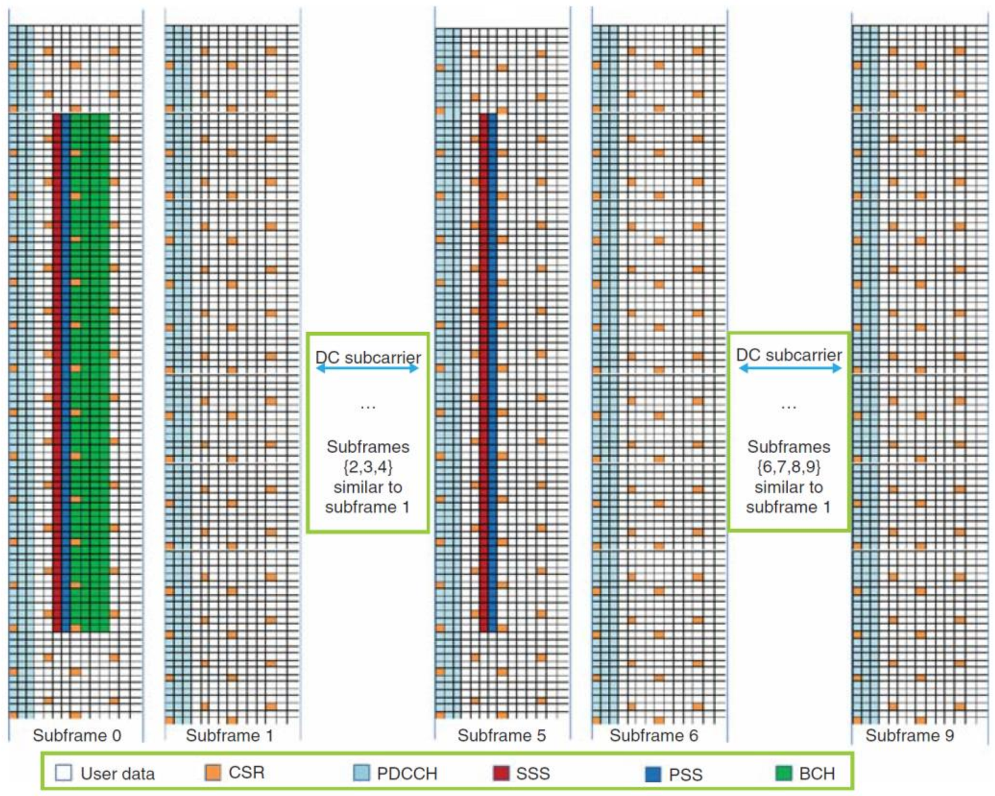


Figure 7: LTE Resource Grid Allocation

Each square is called a resource element and corresponds to one modulated symbol.

Note that in the simulation the PDCHH was defaulted to one column, but can be changed to 2 or 3 columns (like in the Figure 3).

The functions of the non-user data symbols would be as follows:

CSR’s contain the symbols used for channel estimation and equalization. They are present in every subframe.

The DCI (Downlink Control Information) carries the content of the PDCCH, PCFICH (Physical Control Format IndicatorChannel), and PHICH (Physical Hybrid ARQ Indicator Channel). DCI is present in every sub frame.

The BCH (Physical Broadcast Channel) The BCH is only present in Subframe 0 (the first sub frame). The BCH contains the MIB (Master Information Block) which contains important information like the system bandwidth, subframe number, and number of transmit antennas at the eNb.

The PSS (Primary Synchronization Signals) and SSS (Secondary Synchronization Signals) are used by the UE to obtain cell identity and frame timing. PSS and SSS are only present in Subframe 5 (the sixth sub frame).

Note that although space was allocated for the DCI, BCH, PSS, and SSS, nothing was mapped to them. Required information that in reality would be obtained from these information signals was just passed through Matlab functions or saved in the Matlab workspace to simplify the simulation. Besides saving the slot number `nS` in the Matlab workspace, the most notable “cheat” was passing the variable `Control_UserDataLen = length(t2)`, the length of the vector of transmitted symbols, through the functions.

CSR’s were used and mapped to the grid.

4.1.5 OFDM Transmission

-File OFDMTx.m

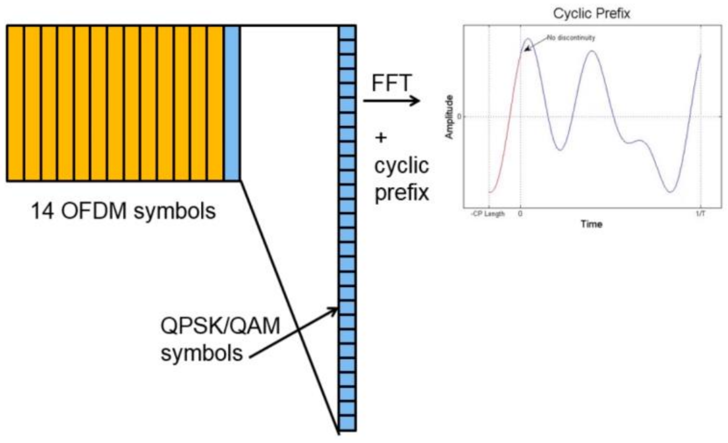


Figure 8

OFDM is a combination of modulation and multiplexing. In this technique, the given resource (bandwidth) is shared among individual modulated data sources. OFDM is very effective for communication over channels with frequency selective fading over a large bandwidth because it converts the entire frequency selective fading channel into small flat fading channels, which can be more easily combated by error correction and equalization techniques.

OFDM is a special case of FDM (Frequency Division Multiplexing). IN FDM the carriers have no special relation to each other. Simply put, if the carriers are harmonics, then they become orthogonal and FDM becomes OFDM. As the carrier spacing is equal to the reciprocal of the symbol period, the carriers will have a whole number of cycles in the symbol period and their contribution will sum to zero - in other words there is no interference contribution. In short, orthogonality property is used to allow much tighter packing of the symbols. Cyclic prefix is added to combat ISI.

According to section 6.12 of 36.211, OFDM signal generation is detailed as follows:

The time-continuous signal $s_l^{(p)}(t)$ on antenna port p in OFDM symbol l in a downlink slot is defined by

$$s_l^{(p)}(t)=\sum_{k=-\lfloor N_{\mathrm{RB}}^{\mathrm{DL}}N_{\mathrm{sc}}^{\mathrm{RB}}/2\rfloor}^{-1} a_{k^{(-)},l}^{(p)}\cdot e^{j2\pi k\Delta f\left(t-N_{\mathrm{CP},l}T_{\mathrm{s}}\right)}+\sum_{k=1}^{\lceil N_{\mathrm{RB}}^{\mathrm{DL}}N_{\mathrm{sc}}^{\mathrm{RB}}/2\rceil} a_{k^{(+)},l}^{(p)}\cdot e^{j2\pi k\Delta f\left(t-N_{\mathrm{CP},l}T_{\mathrm{s}}\right)}$$

for $0\le t<\left(N_{\mathrm{CP},l}+N\right)\times T_{\mathrm{s}}$ where $k^{(-)}=k+\lfloor N_{\mathrm{RB}}^{\mathrm{DL}}N_{\mathrm{sc}}^{\mathrm{RB}}/2\rfloor$ and $k^{(+)}=k+\lfloor N_{\mathrm{RB}}^{\mathrm{DL}}N_{\mathrm{sc}}^{\mathrm{RB}}/2\rfloor-1$. The variable N equals 2048 for $\Delta f=15\,\mathrm{kHz}$ subcarrier spacing and 4096 for $\Delta f=7.5\,\mathrm{kHz}$ subcarrier spacing.

The OFDM symbols in a slot are transmitted in increasing orders of l, beginning at l=0, where l is the column number in the time-domain. OFDM symbol $l>0$ starts at time $\sum_{l'=0}^{l-1}(N_{\mathrm{CP},l'}+N)T_{\mathrm{s}}$ within the slot.

SamplingRate = 30.72 MHz / 2048 × N_{fft} [8].

Nfft is the smallest power of 2 greater than or equal to 12 × N_{RB} / 0.85. It is the smallest FFT that spans all subcarriers and results in a bandwidth occupancy (12 × N_{RB} / N_{fft}) of no more than 85% [8].

The cyclic prefix lengths [8] for one frame for normal cyclic prefix are given below:

N_{fft}	**Cyclic Prefix Lengths**
2048	[160 144 144 144 144 144 144 160 144 144 144 144 144 144]
1024	[80 72 72 72 72 72 72 80 72 72 72 72 72 72]
512	[40 36 36 36 36 36 36 40 36 36 36 36 36 36]
256	[20 18 18 18 18 18 18 20 18 18 18 18 18 18]
128	[10 9 9 9 9 9 9 10 9 9 9 9 9 9]

Hence the lines of code:

```
slotLen = (N*7 + cpLen0 + cpLenR*6);
subframeLen = slotLen*2;
```

Therefore for the single transmit antenna case, the OFDMtx function will return a column vector of length `subframelen.`

4.2. Propagation Channel

-File: Fading_or_ISIChan.m

The propagation channel allows for options of no fading, flat high mobility fading, frequency selective high mobility fading, as well as the two levels of ISI from the first project of the semester. In terms of simulating real life, high mobility fading is most appropriate due to UE (user equipment) movement. These channels use the `comm.RayleighChannel` system object with Maximum Doppler Shift defaulted to 70 Hz. The Maximum Doppler Shift must be smaller than SampleRate/10/fc for each path, where fc represents the cutoff frequency factor of the path [9].

AWGN is always added after the fading channel. AWGN is calculated based on the input SNR.

4.3 Receiver

4.3.1 OFDM Receiver

-File: OFDMRx.m

The OFDM receiver basically undoes what the OFDM transmitter did, i.e. the receiver removed the cyclic prefix, performs the FFT, then reorders the symbols into the original order they were in the transmitted grid.

4.3.2 Resource Grid De-mapping

-File: REdemapper_1Tx.m

The resource grid de-mapper undoes what the mapper did, i.e. the de-mapper takes in the received grid and breaks the symbols into the user-data and the other types of data.

4.3.3 Channel Estimation

-File: ChanEstimate_1Tx.m

-Uses nested function gridResponse_averageSubframe.m

When a cyclic prefix of length Ncp is added to the OFDM symbol, the output of the channel (r) is given by circular convolution of channel impulse response (h) and the transmitted OFDM symbols (x) [10].

$$r = h \circledast x$$

For discrete signals, circular convolution in the time domain translates to multiplication in the frequency domain. Thus, in frequency domain, the above equation translates to

$$\mathbf{R} = \mathbf{H}\mathbf{X}$$

Hence to estimate the transmitted signal from the received signal, the following simple equation is used:

$$\widehat{\mathbf{H}} = \frac{\mathbf{X}}{\mathbf{R}}$$

Because the values of the pilot symbols are known, the channel response at these locations can be determined using the least squares estimate. The least squares estimate is obtained by dividing the received pilot symbols by their expected value.

The function does this by taking the received CSRs aligning them in a grid of length 4 and height of $N_{CSR}/4$, and point wise dividing that grid by the grid of transmitted CSRs, placing the output in the variable called `hp`.

gridResponse_averageSubframe.m then takes `hp` and time averages the pilots symbols then interpolates over the frequency axis forming the channel response for the whole grid. This process is described in Annex F.3.4 in TS 36.141 [4].

4.3.4 Equalization

-File: Equalizer.m

When the channel matrix is represented in block form, the Zero-Forcing equalizer is represented simply by [6]:

$$\mathbf{H}_{\mathbf{ZF}}^{+} = (\mathbf{H}^{\mathbf{H}}\mathbf{H})^{-\mathbf{1}}\mathbf{H}^{\mathbf{H}}$$

The equalized estimate of X can then be computed by $\widehat{\mathbf{X}} = \mathbf{H}_{\mathbf{ZF}}^{+}\mathbf{R}$

The ZF equalizer applies the inverse of the frequency response of the channel. The name Zero Forcing corresponds to bringing down the ISI to zero in a noise free case.

In order account for noise, the inverse matrix solution with the MMSE (Minimum Mean Square Error) criterion is represented by:

$$\mathbf{H}^{+}_{\mathbf{MMSE}} = (\mathbf{H}^{\mathbf{H}}\mathbf{H} + \sigma_n^2\,\mathbf{I})^{-\mathbf{1}}\mathbf{H}^{\mathbf{H}}$$

The MMSE solution trades off the signal separation quality for the noise reduction, so it is not the best option in a low noise environment [6].

4.2.5 Demodulation

-File: DemodulatorSoft.m

The demodulator uses soft-decision decoding to perform demodulation. Its outputs is not bits, but LLRs. The input to the soft demodulator is the equalized signal. The extra information (as opposed to an input of just 1s and 0s) indicates the reliability of each input data point, and is used to form better estimates of the original data. Therefore, a soft-decision demodulator will typically perform better in the presence of corrupted data than a hard-decision demodulator.

4.3.6 Descrambling

-File: DescramblerSoft.m

Since a soft demodulator was used, a soft descrambler must be used. The descrambler uses the same Gold sequence generator (as discussed in the section for the scrambler) to invert the scrambling operation. The descrambler initialization is synchronized with that of the scrambler.

4.3.7 Turbo Decoding

-File: TurboDecoder.m

In the receiver, the turbo decoder inverts the operations performed by the turbo encoder. A turbo decoder is based on the use of two A Posteriori Probability (APP) decoders and two interleavers in a feedback loop. The output of the Turbo Decoder is the output bits. After exiting the while loop in SISO_E2E.m, these bits are compared to the original input bits to calculate number of errors and BER.

4.4. Overall Simulation and Code Block Segmentation Method

-File SISO_E2E.m

Note: Please only use input bit sequence sizes divisible by 2.

While testing earlier versions of the code, it was noticed that without any Rate Matching or Code Block Segmentation, increasing the bandwidth did not improve simulation time at all. This is because without DLSCH preprocessing to the PDSCH, the maximum block size is determined by the interleaver which has a maximum frame input size of 6144 bits. The matlab object Turbo Encoder is rate 1/3 and appends a 12 bit check sequence at the end. Then those bits are modulated with QPSK, 16QAM, or 64QAM to form the actual transmitted symbols.

Therefore the maximum block size (maximum meaning resulting from 6144 bits to the interleaver) for the different modulations is as follows:

Modulation	**Max Block Size**
QPSK	9,210
16QAM	4,605
64QAM	3,070

The available number of user-data resource element space for the different modulations for contReg = 1 is shown in the following table. See Appendix A for calculations and associated Matlab functions. The full table is stored in the variable NuserTable.mat.

	Subframe 0	**Subframe 5 (Slot 10)**	**Other Subframes**
1.4 MHz	480	756	900
3 MHz	1830	2106	2250
5 MHz	3330	3606	3750
10 MHz	7080	7356	7500
15 MHz	10830	11106	11250
20 MHz	14580	14856	15000

Note that at 20Mhz at 64QAM with no code block segmentation, an entire sub frame of 15,000 is wasted on 3,070 symbols. This goes to show the importance of rate matching, code block segmentation, and other DLSCH processes.

The complete processing chain would have been:

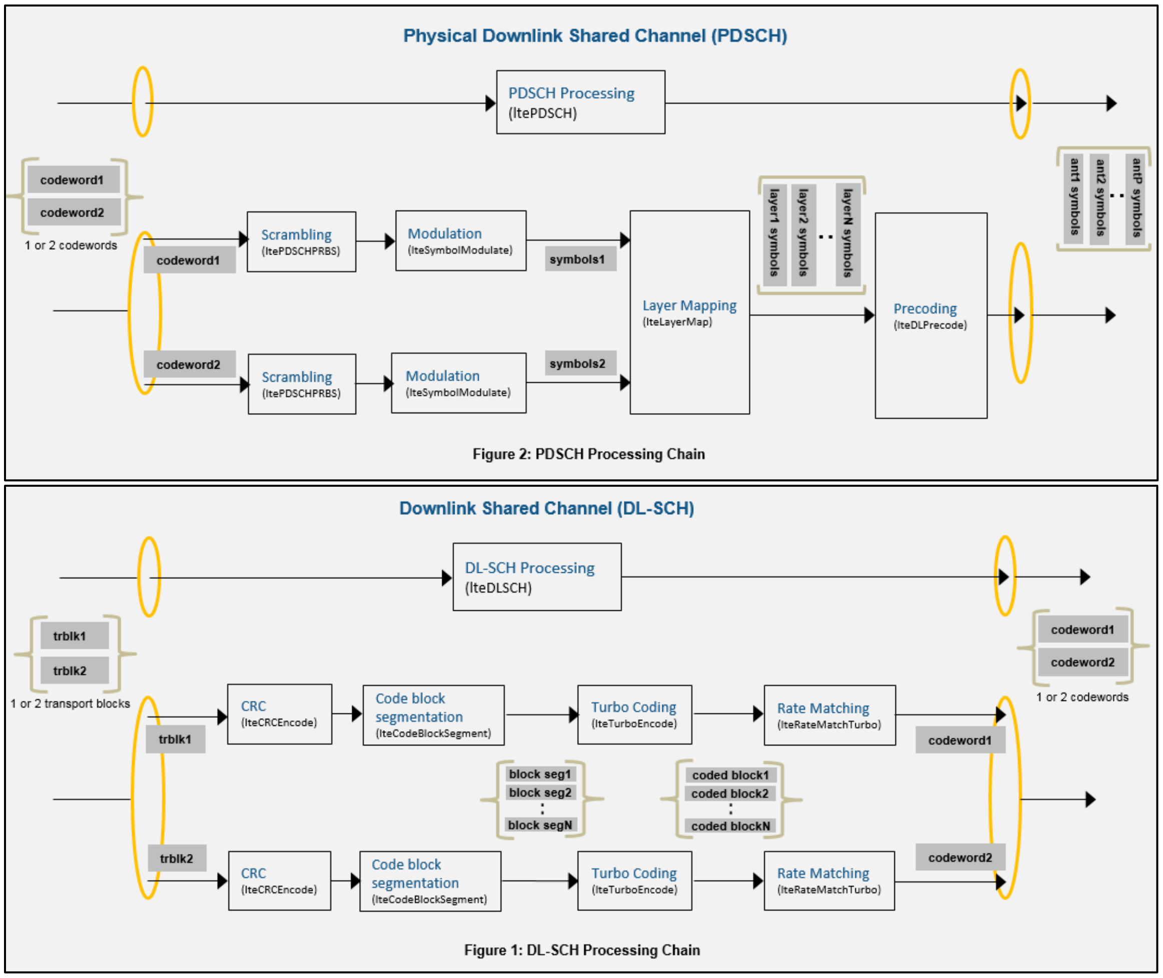

Physical Downlink Shared Channel (PDSCH)
PDSCH Processing (ltePDSCH)
codeword1
codeword2
1 or 2 codewords
codeword1
codeword2
Scrambling (ltePDSCHPRBS)
Scrambling (ltePDSCHPRBS)
Modulation (lteSymbolModulate)
Modulation (lteSymbolModulate)
symbols1
symbols2
Layer Mapping (lteLayerMap)
layer1 symbols
layer2 symbols
layerN symbols
Precoding (lteDLPrecode)
ant1 symbols
ant2 symbols
antP symbols
Figure 2: PDSCH Processing Chain
Downlink Shared Channel (DL-SCH)
DL-SCH Processing (lteDLSCH)
trblk1
trblk2
1 or 2 transport blocks
trblk1
trblk2
CRC (lteCRCEncode)
CRC (lteCRCEncode)
Code block segmentation (lteCodeBlockSegment)
Code block segmentation (lteCodeBlockSegment)
block seg1
block seg2
block segN
Turbo Coding (lteTurboEncode)
Turbo Coding (lteTurboEncode)
coded block1
coded block2
coded blockN
Rate Matching (lteRateMatchTurbo)
Rate Matching (lteRateMatchTurbo)
codeword1
codeword2
codeword1
codeword2
1 or 2 codewords
Figure 1: DL-SCH Processing Chain

Figure 9: Full DL-SCH and PDSCH Design

Rate matching and code block segmentation were not implemented according to the standard in the interest of time. Additionally, the full standard calls for a MIMO or SIMO case when this simulation was of a SISO case. By the standard, each code block has a CRC appended at the end. The first block has zero padding bits prepended. Code block size is determined in Section 7.1.7 of TS 36.213 [3] and concatenation in Section 5.1.5 in TS 36.212.

Although these processes were not simulated strictly to the standard, a form of optimization and codeblock segmentation was written. The rate of the turbo coder is never was changed. However, the input bits are cut into blocks that so that they take up as much possible frame size in the interleaver. Recall the interleaver's maximum frame size is 6144. In order to best fill up the available space in the resource grid, multiple blocks of this size plus a block of a smaller size are all turbo encoded. These turbo encoded blocks are then concatenated and the output vector is fed in the scrambler and modulater as normal. After going through the channel and demodulating and descrambling at the receiver, the resultant vector is again broken up into groups, fed into the turbo decoder, and then the blocks are concatenated again to form the final output bits. This processing is kept outside of a nested function call in order to follow bit movement more easily. Any information that would actually be sent in the BCH, DCI, PSS, or SSS is labeled as a control variable. This implementation is shown in the file SISO_E2E.m.

Note that an older implementation of this processing broke the input bits into blocks then performed scrambling, modulation, and turbo coding on *each* block. This implementation was not realistic for the SISO case and slowed the simulation down greatly. This process was more akin to how code words are processed, like in Figure 9. Processing each block within one subframe in this style did not make sense. Therefore, this implementation was not used to generate BER curves but is included in the zip file as ALTERNATE_SISO_E2E.m.

5. Analysis

5.1 BER

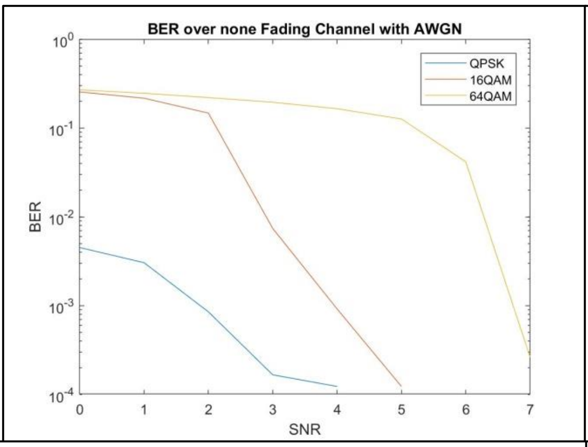
-File: main_E2E.m

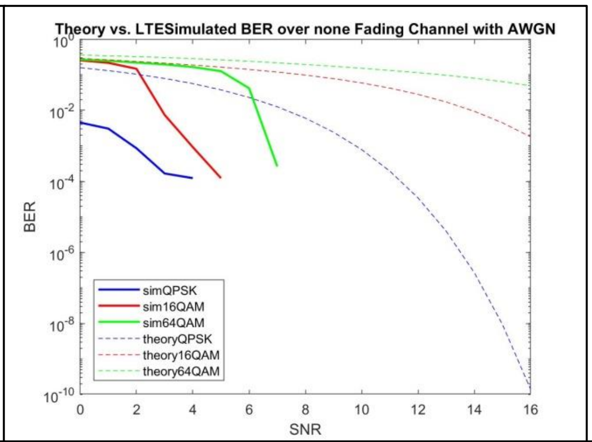
-uses nested function BERsim_E2E.m

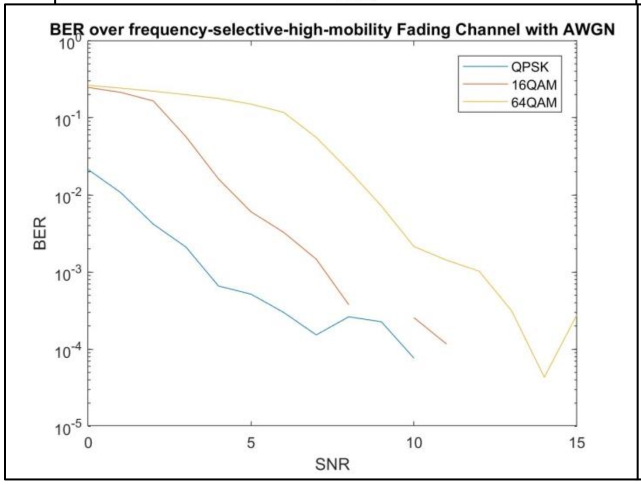
The following BER curves were generated using the following parameters:

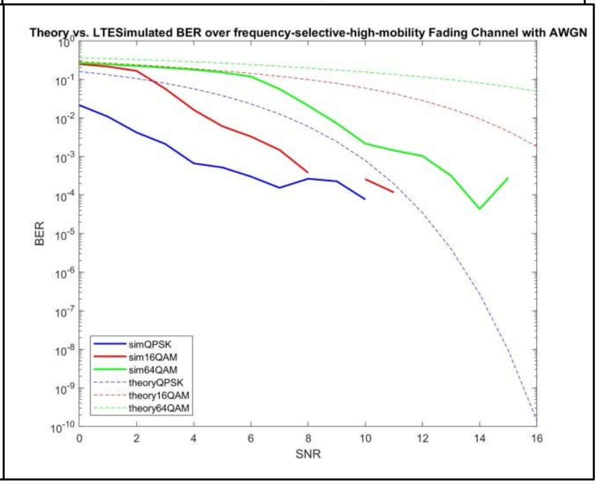
snrVect = [0:16], numTrials= 20, size = 15,000, ZF equalizer, nIterTurboMax = 6:

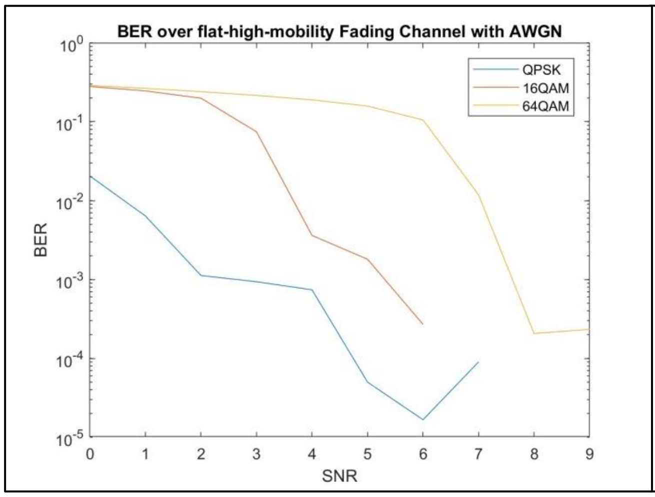
For QPSK at 1.4 Mhz bandwidth:

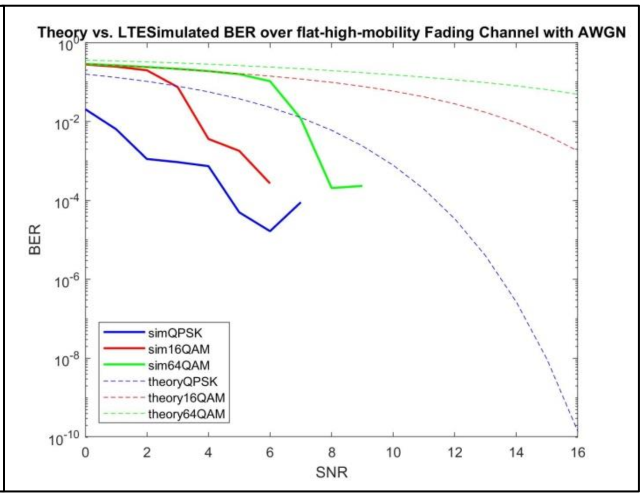


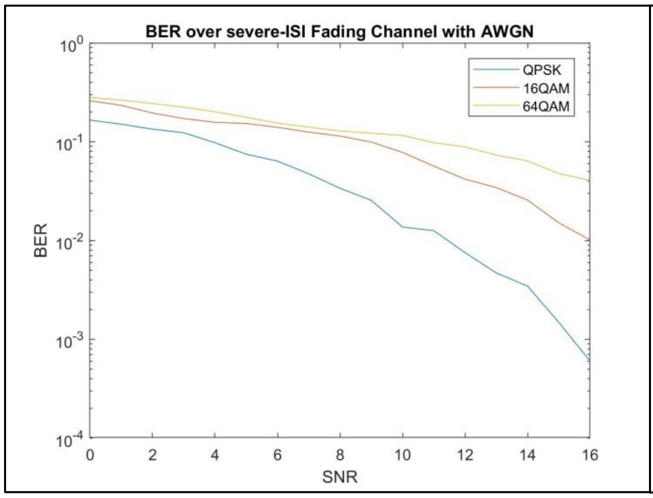


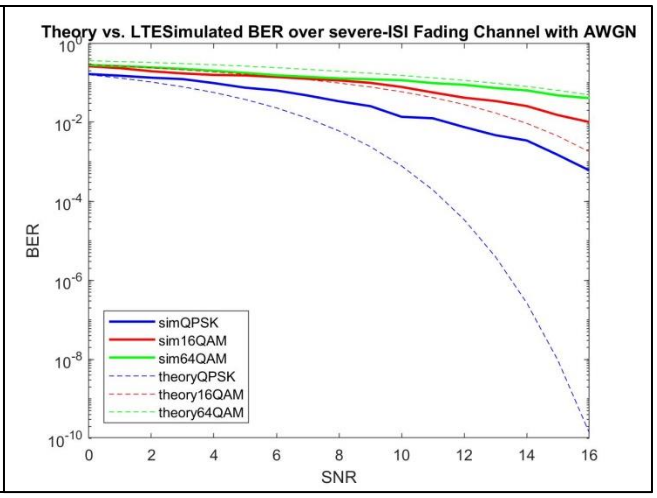


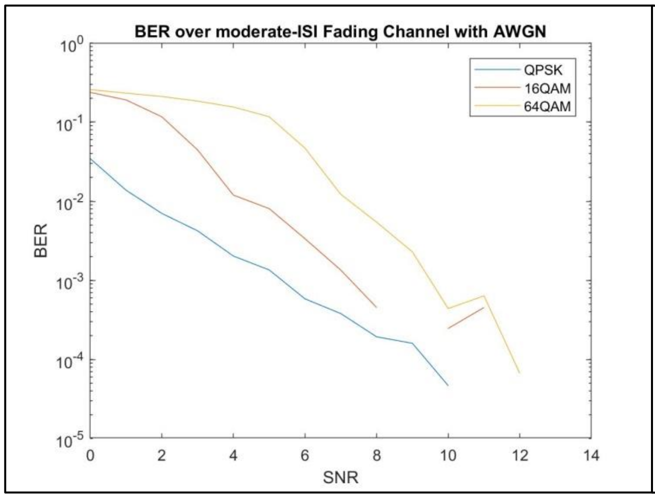


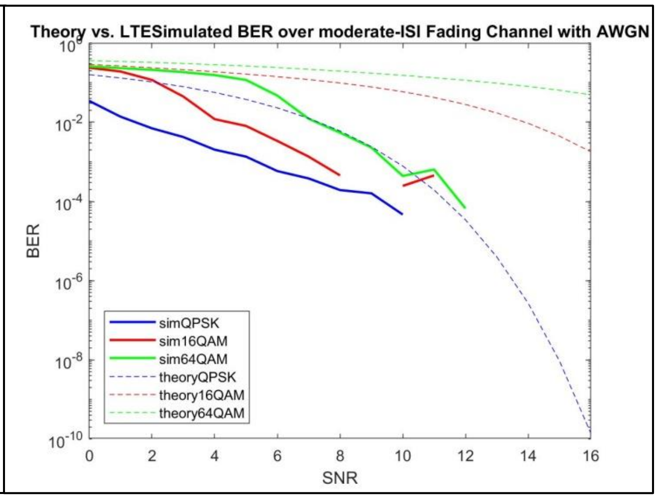

BER over flat-high-mobility Fading Channel with AWGN
QPSK
16QAM
64QAM
BER
SNR

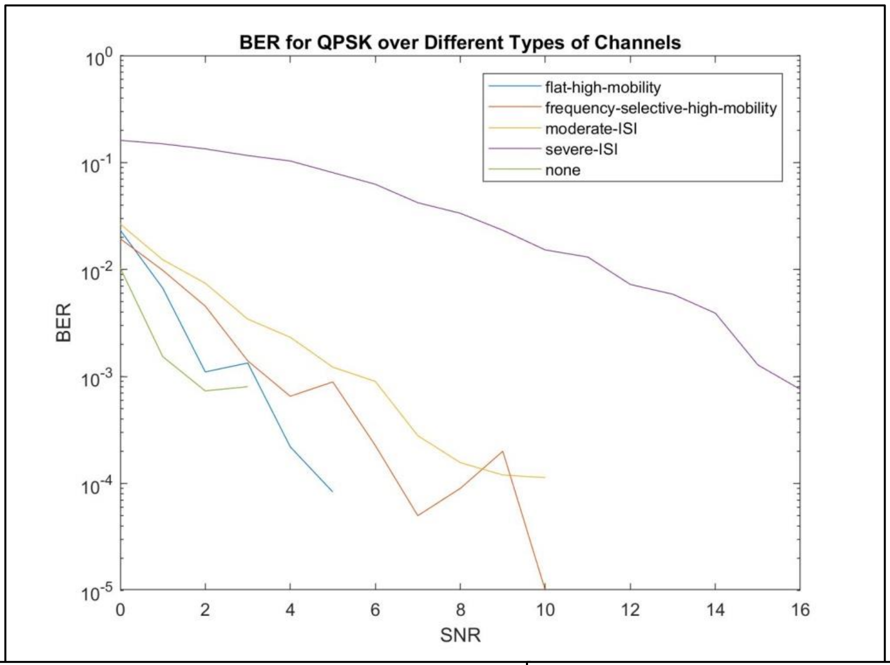

Theory vs. LTESimulated BER over flat-high-mobility Fading Channel with AWGN
simQPSK
sim16QAM
sim64QAM
theoryQPSK
theory16QAM
theory64QAM
BER
SNR

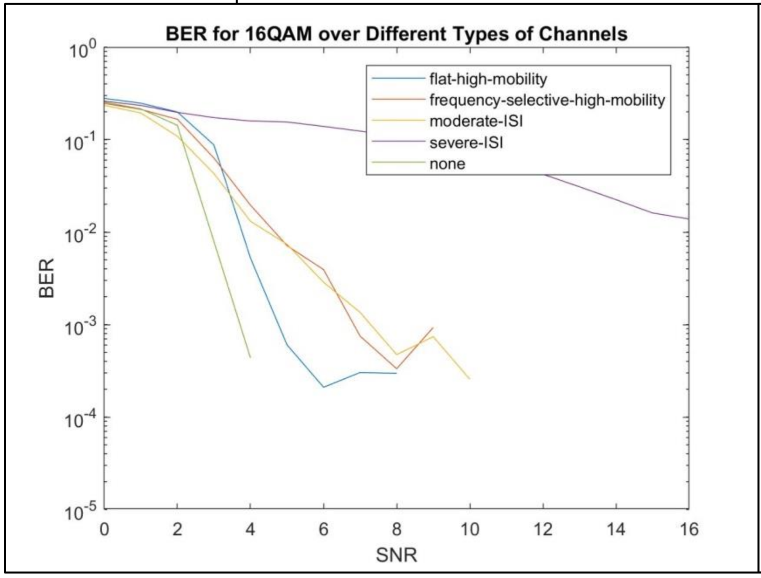

BER over severe-ISI Fading Channel with AWGN
QPSK
16QAM
64QAM
BER
SNR

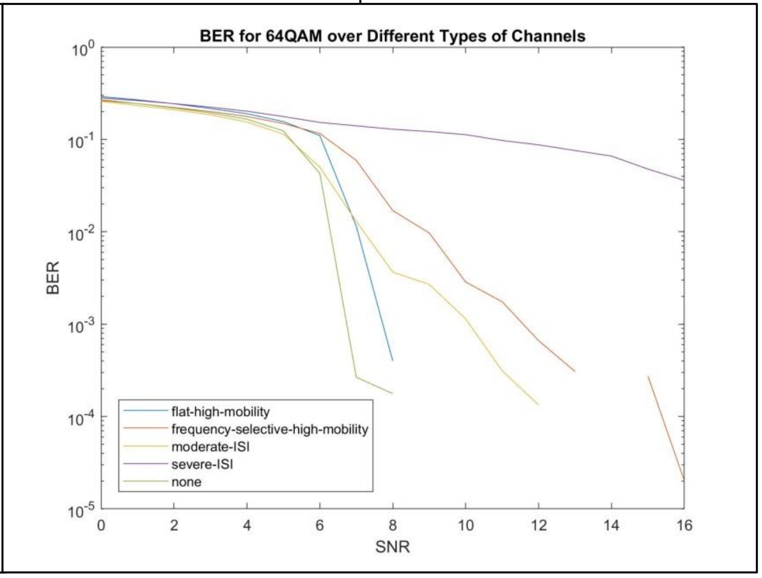

Theory vs. LTESimulated BER over severe-ISI Fading Channel with AWGN
simQPSK
sim16QAM
sim64QAM
theoryQPSK
theory16QAM
theory64QAM
BER
SNR


BER over moderate-ISI Fading Channel with AWGN
QPSK
16QAM
64QAM
BER
SNR


Theory vs. LTESimulated BER over moderate-ISI Fading Channel with AWGN
simQPSK
sim16QAM
sim64QAM
theoryQPSK
theory16QAM
theory64QAM
BER
SNR







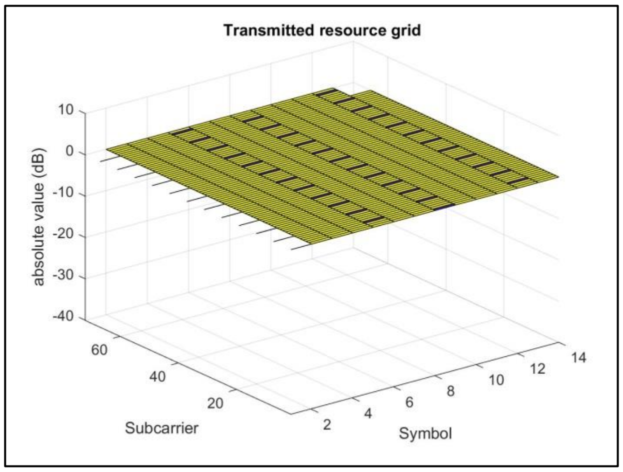
Over the different types of channels, the simulation performed the best under no fading, second best under flat high mobility fading, and worst in the severe ISI case. Perhaps the performance of some of the BER curves could be enhanced by switching to the MMSE equalizer. Since the simulations under the given parameters take a while (~5 minutes each), the values are saved in the zip files as .mat files for reproduction. Reducing the number of trials to 1, even with a high input stream bit length, can run the simulation in under a half a minute in some cases.

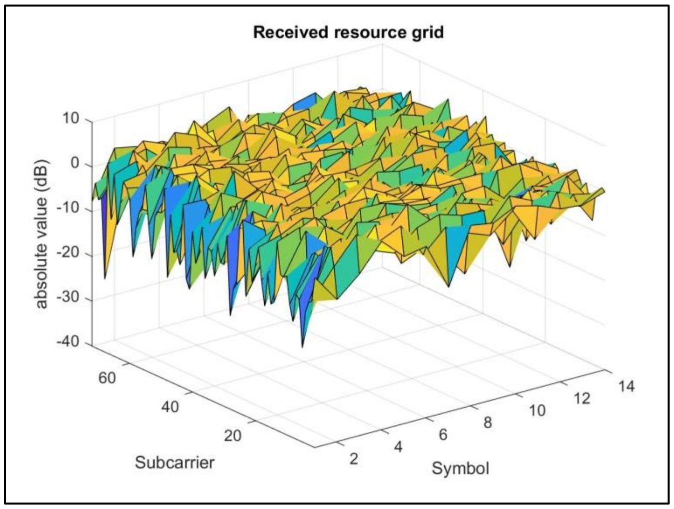
5.2 Equalization

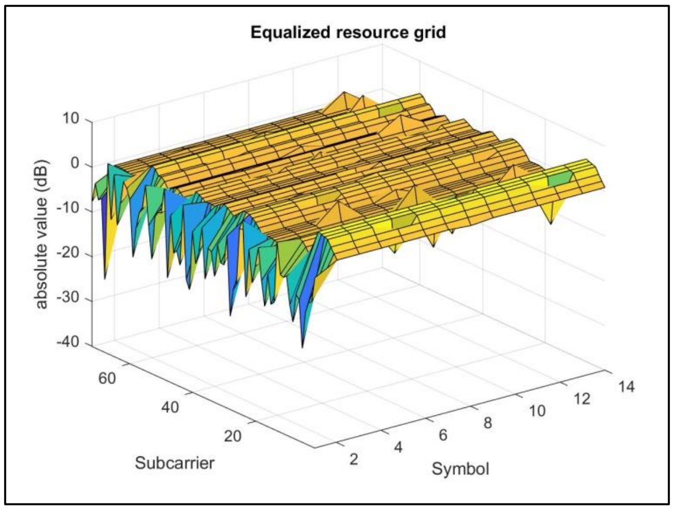
-File: eqPlots.m

Because the equalization function only computed equalization for the number of user-data input symbols, care was taken to generate a simulation where most of the user-data space was filled. In order to form the equalized grid, the equalizer gain vector was then mapped to the rx-grid to replace the rx-grid user-data symbols with the equalized gain values. It is clear that the equalization was very effective in estimating the channel and greatly reduced BER.

The following plots were created using size = 2080, chanBW=1, Mode =1, ZF equalizer, nIterTurbomax=6, and chMdl = frequency-selective-high-mobility':







5.3 Simulation Time

Time shown in charts is in seconds.

The following simulation times were measured using the following parameters:

chanMdl = 'frequency-selective-high-mobility', snrVect = [0:5], numTrials= 1, size = 100,000:

	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
All Modulations	58.3678	37.6368	34.1418	31.2584	30.2942	30.6252
Just QPSK	26.1412	17.0746	13.1682	11.5180	11.1231	11.2670
Just 16QAM	18.3268	11.6926	10.8768	10.3706	9.9735	11.9714
Just 64QAM	15.9424	10.8863	10.3332	9.6898	9.4859	9.6362

When the bit input size was doubled to 200,000:

	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
All Modulations	109.8351	74.6442	67.5206	62.0494	60.2516	60.3724

Note that the as bandwidth increases, the simulation time decreases as a slower rate. NuserTable.mat provides insight to this. It is shown that as bandwidth increases, the user-data available space increases at a diminishing rate. Still, it is an odd result that from 15MHz to 20MHz the simulation time was actually a bit greater. This is probably due to the extreme size of the 20MHz size frame, i.e. when there are only a few bits left, the simulation still has to process a whole subframe. Actual implementation of rate matching would fix this issue. Note that at smaller frame sizes, the positive effect of increased bandwidth is clearly demonstrated (see tables below).

At the following parameters:

chanMdl = 'frequency-selective-high-mobility', snrVect = [1], numTrials= 1, size = 15,000:

	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
Just QPSK	2.2826	0.4430	0.3674	0.3457	0.3338	0.3020

When the SNR was changed to a vector of snr = [0:5] :

	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
Just QPSK	5.6137	2.4705	2.1332	1.8400	1.8286	1.7786

6. Conclusion

LTE is a powerful and complex standard. This simulation models part of the PHY downlink of LTE by employing Turbo Coding, OFDM, and Equalization, among other techniques. The simulation simulated the PDSCH processing and inverse PDSCH processing to the standard of TS 36 series Release 8. Non-standard adherent techniques were used for code block segmentation and maximizing available user-data space on the transmitted OFDM resource grid. Overall, this simulation produced BER within acceptable ranges for different types of fading channels.

7. References

[1]] 3GPP TS 36.211. “Physical channels and modulation.” 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).

[2] 3GPP TS 36.212. “Multiplexing and channel coding.” 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).

[3] 3GPP TS 36.213. “Physical layer procedures.” 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).

[4] 3GPP TS 36.141. “Base Station (BS) conformance testing.” 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA).

[5] Zarrinkoub, Houman. Understanding LTE with MATLAB: From Mathematical Modeling to Simulation and Prototyping. Chichester, West Sussex, John Wiley & Sons, 2014.

[6] Li, Cong, and Yasunori Iwanami. "Linear and Nonlinear Time Domain Block Equalizers on MIMO Frequency Selective Channels." Scientific Research, Nagoya Institute of Technology, 14 Jan. 2014, file.scirp.org/Html/1-9701730_28905.htm. Accessed 25 Mar. 2018.

[7] "What Is LTE?." MathWorks, https://www.mathworks.com/help/lte/gs/what-is-lte.html. Accessed 25 Mar. 2018.

[8] "lteOFDMModulate." MathWorks, https://www.mathworks.com/help/lte/ref/lteofdmmodulate.html. Accessed 26 Mar. 2018.

[9] "comm.RayleighChannel System object." MathWorks, https://www.mathworks.com/help/comm/ref/comm.rayleighchannel-system-object.html. Accessed 26 Mar. 2018.

[10] Mathuranathan. "Introduction to OFDM – orthogonal Frequency division multiplexing." , 6 July 2011, https://www.gaussianwaves.com/2011/07/introduction-to-ofdm-orthogonal-frequency-division-multiplexing-part-4-cyclic-prefix/. Accessed 26 Mar. 2018.

7.1 Figures

Figure Number	Source
1	N/A
2	http://worldtechie.blogspot.com.
3	https://www.tutorialspoint.com/lte/lte_ofdm_technology.htm
4	https://www.mathworks.com/videos/lte-tutorial-understanding-the-lte-resource-grid-99215.html
5	https://www.mathworks.com/videos/lte-tutorial-understanding-the-lte-resource-grid-99215.html
6	Reference [2]
7	Reference [5]
8	https://www.mathworks.com/videos/lte-tutorial-understanding-the-lte-resource-grid-99215.html
9	https://www.mathworks.com/help/lte/examples/lte-dl-sch-and-pdsch-processing-chain.html

Appendix A

NuserTable.mat was generated with the function NuserCalculater.m. See Figure 7 in Section 4.1.4 for the standard's details on how resource elements are mapped to the grid. Values are drawn from NuserTable using the function call NuserLookup.m which uses the command Nuser = NuserTable(chanBW,subframe#,contReg).

Number of user data symbols per sub frame were calculated as follows:

Sub frame 0 (all sources of data present):

Simulation assumes DCI is one column (shares column with CSR's). DCI can be up to three columns total (one sharing space with CSRs, the other two not). Can change to 2 or 3 columns in file SISO_E2E.m.

General: $N_{\text{user data}} = N_{\text{total}} - (N_{\text{CSR}} + N_{\text{DCI}} + N_{\text{PSS}} + N_{\text{SSS}} + N_{\text{BCH}})$

1.4 MHz: $N_{\text{user data}} = (12*6*14) - ((8*6) + (72-12) + 72 + 72 + 276) = 480$

Sub frame 5 :

General: $N_{\text{user data}} = N_{\text{total}} - (N_{\text{CSR}} + N_{\text{DCI}} + N_{\text{PSS}} + N_{\text{SSS}})$

1.4 MHz: $N_{\text{user data}} = 10008 - (48 + 60 + 72 + 72) = 756$

Other frames:

General: $N_{\text{user data}} = N_{\text{total}} - (N_{\text{CSR}} + N_{\text{DCI}})$

1.4 MHz: $N_{\text{user data}} = 1008 - (48 + 60) = 900$

Throughout entire frame (10 subframes)

$N_{\text{user data whole frame}} = 480 + 756 + 8(900) = 8{,}436$.

For 1.4MHz case: Out of the total available space in one frame (10,080 symbols) user data take up 83.7% of it and control and pilot info takes up the rest.