

User Manual

The Vienna 5G Link Level Simulator v1.2

Institute of Telecommunications,
TU Wien

Authors

Stefan Pratschner, Bashar Tahir, Ronald Nissel, Ljiljana Marijanovic,
Mariam Mussbah, Kiril Kirev, Stefan Schwarz and Markus Rupp

Vienna, May 27, 2020

Institute of Telecommunications, TU Wien

Gusshaussstrasse 25/389
A-1040 Vienna
Austria

web: http://www.nt.tuwien.ac.at/

Contents

1 Introduction

Our research group has a long and successful history of developing and sharing open-source cellular communications simulators. The implementation of our work-horse of the past eight years, that is, the Vienna LTE Simulators, started back in 2009. Although, the evolution of this project was not straight forward from the beginning, in total three reliable simulators evolved. The system level simulator and the up- and downlink link level simulators attained quite some attention from academia as well as from industry. Over the years, many simulator versions, including new features according to the LTE standard and several bug fixes, were released. Many of these bugs were reported by an online community through our simulator forum. Today the Vienna LTE Simulators count more than 50 000 downloads in total. This historical development shows the need for a standard compliant reliable simulation tool for performance evaluation and comparison. We therefore extend our simulator suite and evolve to the next generation of mobile communication by introducing new 5G simulators.

In this user manual we will describe the general idea and scope, as well as implementation details and usage of the Vienna 5G Link Level Simulator. As an introduction, we explain the concept and functionality of the simulator in this document.

The Vienna 5G Link Level Simulator is the newest member of the family of Vienna Cellular Communications Simulators (VCCS). Although, as of this writing, there exists no definite 5G specification, the standardization process within 3rd Generation Partnership Project (3GPP) is ongoing and is already taking shape. Work on the first set of 5G standards is expected to start by the second half of 2017 within Long Term Evolution (LTE) Rel. 15. While it is not yet decided which physical layer methods will be standardized for 5G, there are several hot candidates for physical layer waveforms and channel coding schemes. Performance evaluation and comparison of these candidate physical layer methods is currently subject to scientific work and topic of many publications over the past years. Through our simulators, we intend to offer a unifying platform for performance evaluation as well as co-existence investigation of candidate 5G physical layer schemes. Since there exists no concrete specification yet, we provide great flexibility by supporting a broad range of simulation parameters. Thus, many different combinations of physical layer settings are comparable by our 5G link level simulator.

In general the purpose of link level simulations of communication systems is to evaluate the average performance of the physical layer transceiver architecture. Correspondingly, the focus of our 5G link level simulator is on point-to-point simulations. Nevertheless, there exists the abstract concept of

cells within our simulator. However, this is realized without implementing an underlying cellular geometry. There is neither a physical cell size nor a distance to the user being considered; the path-loss to a user is rather specified as an input parameter, leading to an average signal to noise ratio. The concept of a cell can be thought of a group of nodes, that is, one base station and several users are grouped within a cell.

To obtain an average system performance as simulation result, we perform Monte Carlo simulations and average over many random channel realizations. As a result, system performance in terms of throughput, Bit Error Ratio (BER) or Frame Error Ratio (FER), is calculated and plotted together with confidence intervals that indicate their statistical reliability.

In this current initial version, our 5G link level simulator supports both, up- and downlink simulations. The simulator includes parameter settings for Long Term Evolution-Advanced (LTE-A) compliant simulations and additionally further user defined settings for the simulation of future 5G cellular communications systems. While currently up- and downlink are implemented for Frequency Division Duplex (FDD) mode only, our simulator structure allows for future implementation of Device-to-Device (D2D) communications as well as a Time Division Duplex (TDD) frame structure. Further, for simulation of 5G communications systems, we offer high flexibility in choosing desired physical layer methods. Not only simulation parameters that were free to choose within the full LTE-A specification, such as channel model, bandwidth or receiver type, but also very basic parameters, such as sampling rate and frame duration, are adjustable in our simulator. To enable investigation of 5G physical layer candidate methods, we support features such as new PHY waveforms like filtered or windowed Orthogonal Frequency Division Multiplexing (OFDM), Filter Bank Multicarrier (FBMC) or Universal Filtered Multicarrier (UFMC), and different channel codes like Turbo coding, LDPC coding or Polar coding. All of these schemes support any combination of channel coding rate and Quadrature Amplitude Modulation (QAM) alphabet size, which results in many different Modulation and Coding Scheme (MCS). In addition, the employed physical layer schemes can be different for users of different cells such that their co-existence can be simulated.

2 Quick Start

The Vienna 5G Link Level Simulator may be used in two different ways. In the simplest case, a pre defined or user defined simulation scenario is simulated. Simulation scenarios are further described in Section 5.2. Alternatively, the implemented classes and functions may be re-used within another framework as explained in Section 5.1.

This quick start guide describes how to run a pre defined simulation scenario.

1. Open the script `main.m` in the simulator root directory.

2. Select a simulation scenario:

```
% select scenario
simulationScenario = 'genericScenario';
```

3. In case you want to specifically set simulation parameters or define your own simulation scenario, edit or create scenario files in the `/Scenarios` directory.

4. Depending on whether you want to simulate in parallel mode, change the loop over the sweep parameter to `for` or `parfor`, accordingly:

```
% loop over sweep parameter
for iSweep = 1:length(simParams.simulation.sweepValue)
            % this may be 'for' or 'parfor'
```

5. Run the `main.m` script. Plots will be shown as the simulation is finished. All results are stored in the `downlinkResults` and `uplinkResults` objects.

function	usage	toolbox	comment
`SphereDecoder()`	`Modulator.m`	Communications System Toolbox	sphere decoding for MIMO
`parfor`	`main.m`	Parallel Computing Toolbox	parallelized simulation loop
`parfor`	`main.m`	Distributed Computing Server	parallelized simulation loop

Table 1: Employed functions requiring MATLAB toolboxes.

3 System Requirements

The simulator is implemented using MATLAB. The current minimum required version is 2016b.

3.1 Employment of MEX Files

The channel decoders as well as other channel coding functionalities are implemented in C++ and compiled to `mex` files. This yields a significant performance gain. We offer compiled `mex` files for Microsoft Windows, included in the simulator download package. Unfortunately, these files are platform dependent and may not work on all systems. However, we also offer the C++ source files, such that each user may compile them to obtain `mex` files for other operating systems. To compile the `mex` files, simply enter `Coding.compileMEX` in the MATLAB command window while being in the main directory of the simulator.

3.2 Matlab Toolboxes

Our implementation does not rely on the use of toolboxes. The majority of the implementation is done relying only on MATLAB built-in functions, supported without any toolbox. Only few methods employ functions from toolboxes. Those functions are listed in Table 1. Note that the usage of those functions are optional, and therefore are not required in order to run the simulator.

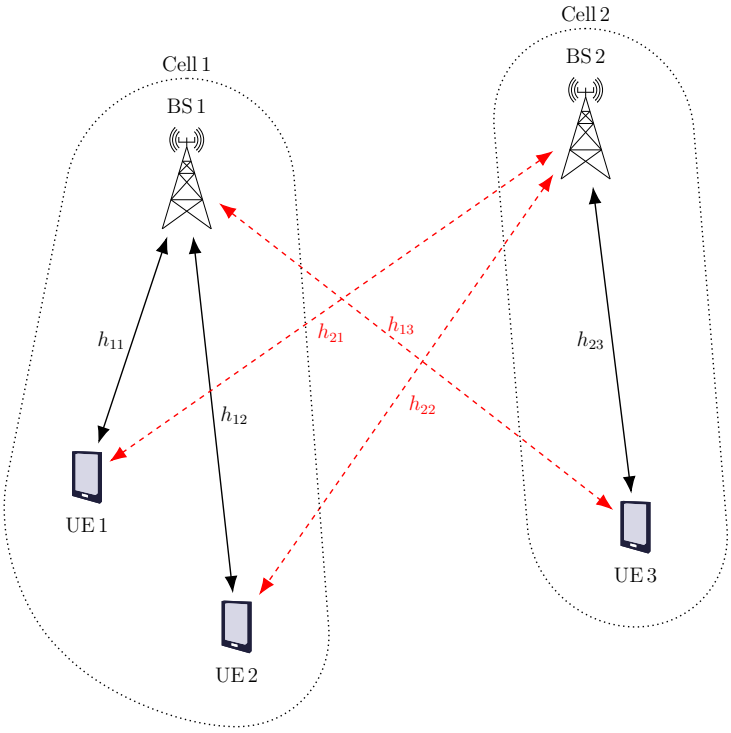


Figure 1: An exemplary network topology.

4 Simulation Methodology

With our 5G link level simulator we want to enable performance evaluation of future physical layer access schemes. For this we aim to maintain high flexibility of simulation scenarios and parameter settings. We not only support various waveforms and channel codes in general, but also allow these parameters to be different from cell to cell. This facilitates investigation of co-existence and interference of 4G and possible 5G physical layer schemes. Again, in our link level simulator there exists no underlying geometry. A cell should be thought of a collection of nodes (one base station and several users) rather than a physical area.

To further explain simulations with different waveforms in different cells, consider the exemplary cellular network topology as shown in Fig. 1. How such a topology is set up, is described in more detail in Section 7.1. Here we assume two cells, each with one base station. Users one and two are attached to base station one and therefore belong to cell one while user three is attached to base station two and belongs to cell two. The wireless channels $h_{i,j}$ are indicated with double arrows where the first subscript i indicates the base station and the second subscript j indicates the user. Desired or primary

channels are shown in solid black while interference or secondary channels are shown in dashed red.

While all nodes belonging to a cell must have the same waveform, channel code, total number of subcarriers[1] and number of symbols per frame (frame duration), these settings might be different for nodes of another cell. To enable discrete simulation of several nodes and many wireless channels, the sampling rate is a common parameter for all cells. Further, the frame duration, that is the number of samples per frame, has to be equal for all cells, independent of the employed waveform and modulation such that interference and desired signals can be superimposed.

For our example, assume that OFDM is used in both cells, but cell one employs a subcarrier spacing of 15 kHz while cell two employs a subcarrier spacing of 30 kHz. In order to obtain the same total bandwidth, the number of subcarriers in the first cell must be double the one in the second cell. Let's assume 72 subcarriers for cell one and 36 subcarriers for cell 2. Similarly, the number of symbols per frame in the second cell must be double the number of symbols in the first cell. As we choose the first cell to be LTE-A compliant, there are 15 total symbols (including the guard symbol for Cyclic Prefix (CP)) per frame in cell one and 30 symbols per frame in cell two. While there is a 15^{th} OFDM symbol in an LTE-A frame spent on CP in cell one, we follow this idea and use two symbols of a frame for CP in cell two to obtain the same number of samples per frame. Still, the sampling rate has to be the same for all nodes and is a common parameter. The parameters described above are entered in a scenario file the following way

```
scStr.modulation.numerOfSubcarriers = [72, 36];      % per BS
scStr.modulation.subcarrierSpacing  = [15e3, 30e3]; % per BS
scStr.modulation.nSymbolsTotal      = [15, 30];      % per BS
scStr.modulation.nGuardSymbols      = [1, 2];        % per BS
scStr.modulation.samplingRate       = 15e3*72*2;
```

where the array index corresponds to the base station index, except for the sampling rate, which has to be the same for all nodes.

As the number of subcarriers and users are chosen individually for each cell, also the schedule has to be adapted accordingly. The schedule is fixed over time, that is, it stays constant for all frames, and for all symbols within a frame. For user assignment, blocks of subcarriers are designated to users of a cell. This schedule is then considered for up- and downlink. The number of users and total subcarriers has to correspond to the topology and modulation settings. The schedule has to be entered in a scenario file the following form

```
scStr.schedule.fixedSchedule{1} = [ 'UE1:36,UE2:36' ];  % BS1
```

[1] Of course, the scheduled bandwidth may be different from user to user.

```
2 scStr.schedule.fixedSchedule{2} = [ 'none:18,UE3:18' ]; % BS2
```

There is a schedule for each base station or cell. In our example there are two users attached to base station one, namely user one and user two. The total number of subcarriers is 72 and we chose to share them equally on the two users. Since the subcarrier spacing is doubled in cell two, the total number of subcarriers is 36 here. The first half of this bandwidth is left unassigned by the keyword *none* while the second half is assigned to user three. In this way, user three is scheduled on the exact same frequency resources as user two.

Depending on the desired simulation scenario, the interference channel from user three to base station one is critical in this setup. If a high attenuation of interference channels is set according to Section 7.1, cell one and cell two will not influence each other. If a low attenuation is selected, significant interference from user three to users one and two will occur. In this setting, users three and one will interfere although they are not overlapping in frequency since they are not orthogonal due to the different subcarrier spacings. If the co-existence of users with different subcarrier spacings is subject of investigation, the interference link attenuation may be set to the same value as the channel's path loss. In this case, interference channels and desired channels are generated following the same statistics with the same average channel power and are therefore equivalent. Considering the received signals at base station one, it is not distinguishable if user three is within cell one or cell two. By this method, users within one cell that employ different modulation schemes can be simulated.

5 Simulation Examples and Scenarios

We include several pre-defined simulation scenarios within the download package of our simulator. The easiest way to define simulation settings is via the scenario files located in the `Scenarios` folder. We provide some ready-to-simulate scenario files with the simulator, that are described in Section 5.2.

However, there might be simulations and comparisons which cannot be obtained by the 5G Link Level Simulator in its original form directly. Since the simulator is implemented in a modular way, using object oriented programming, parts of it might be reused within a different simulation script. We provide simulation examples in the `Examples` folder, that re-use parts of the simulator by exploiting objects. These examples are described in Section 5.1.

5.1 Simulation Scripts

This section describes simulation scripts that are provided in the `Examples` folder and do not exploit the whole implemented Vienna 5G Link Level Simulator structure. To run this example simulations, execute the desired script directly in MATLAB.

5.1.1 Channel Coding with Short Block Length

The example script `shortBlockChannelCoding.m` performs a comparison between the channel coding schemes of convolutional, turbo, LDPC, and polar codes for the case of short block length combined with a low code rate. Such scenario has its importance in various applications, such as the control channels in cellular systems, and also in the Fifth Generation (5G) Massive Machine Type Communication (MMTC) and Ultra-Reliable and Low-Latency Communication (URLLC) use cases. As mentioned in the introduction of the section, this example uses the coding object in a separate manner from the simulator. With this way, we set the target length and code rate irrespectively of the number of scheduled resources or the target Channel Quality Indicator (CQI) code rate. The simulation is then setup according to Table 2. Once the simulation is finished, the script calculates the confidence intervals and plots the results. It is clearly visible in the results that the polar code is the clear winner in such scenario. At the FER of 10^{-2}, we observe a lead for the polar code of about 1 dB against the LDPC code and of 1.5 dB against the turbo and convolutional codes. This result, however, does not alone rule out the choice of the coding scheme, as there are other considerations with respect to the decoding latency, hardware implementation, etc.

parameter	**value**			
channel code	convolutional	turbo	LDPC	polar
decoder	MAX-Log-MAP	Linear-Log-MAP	PWL-Min-Sum	CRC-List-SC
iterations/list size	-	16	32	32
block length	64 bits (48 info + 16 CRC)			
code rate	1/6			
modulation	4 QAM			
channel	AWGN			

Table 2: The simulation parameters of the channel coding for short block lengths example. Four different channel codes are compared for the same short block length.

5.1.2 Comparison of Waveforms

The example script `Comparison_Waveforms.m` provides a comparison of different waveforms that are supported by our simulator. Specifically, it shows results for OFDM, FBMC, Weighted OverLap and Add (WOLA), filtered OFDM (f-OFDM) and UFMC. The waveforms are compared in terms of their pulse shape in time and frequency domain. Further, simulation results for the BER over a frequency selective channel are generated.

5.1.3 LTE Comparison

The script `LTE_comparison.m` simulates all CQI values and saves the results. This results can then be compared to results obtained with the Vienna LTE-A simulator. Together with the `plotLTEComp.m` script, this enables to reproduce the comparison shown in Section 6.

5.2 Simulation Scenarios

Pre-defined simulation scenarios, defined via a corresponding scenario file, are described in this section. Choose the scenario to simulate via

```
1 simulationScenario = 'genericScenario'; % select a simulation
      scenario
```

in the `main.m` simulation script.

5.2.1 Generic Scenario

This scenario file does actually not fit any specific simulation setup. It serves as a reference, as all possible parameter settings are listed and describes. We chose to make this scenario file as self-explaining as possible rather than

describing each single parameter in this documentation. Further, this scenario is as a good starting point to define your own simulation setup.

5.2.2 LTE-Advanced

This simulation scenario sets up a single cell downlink transmission with LTE-A standard compliant settings. This means a subcarrier spacing of 15 kHz and 14 OFDM symbols within a frame of 1 ms duration. Also the LTE downlink pilot pattern is employed together with pilot aided Least Squares (LS) channel estimation and linear interpolation. A 2×2 Multiple-Input Multiple-Output (MIMO) transmission with a fixed identity precoder is employed.

5.2.3 Multi-Link Simulation

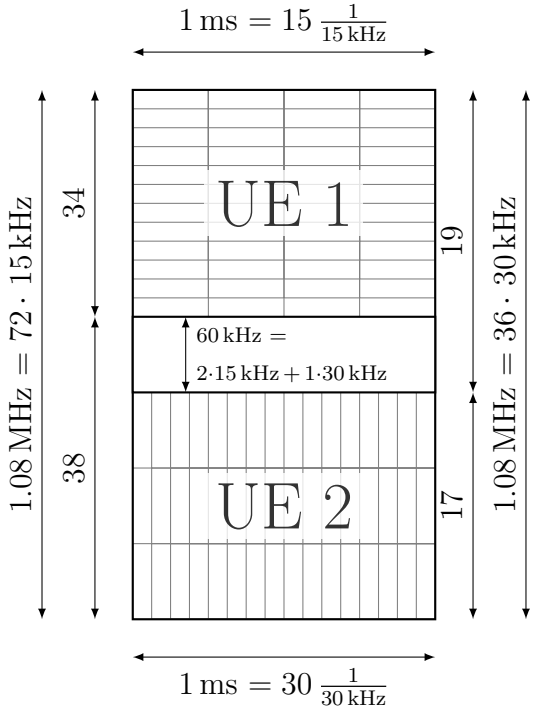


Figure 2: User resource allocation within a cell.

In this scenario we want to investigate the impact of interference between two users. We consider only uplink transmissions for this example. We assume two users that employ different numerologies (subcarrier spacings) and are therefore non-orthogonal to each other. The two users are scheduled next

to each other in frequency as shown in Fig. 2, such that they experience interference due to the high Out Of Band (OOB) emissions of OFDM. As already described in Section 4, there may only be one numerology, waveform and channel coding method per cell. Therefore, the topology setup is as follows. There are two base stations and two users in total, one user per cell. UE1 is assigned to BS1 and UE2 is assigned to BS2. However, to virtually place these two users within one cell, we exploit the interference links and set the inter-cell attenuation to the same value, as the channel path loss. By this, the interference channels and desired channels become indistinguishable. This enables to investigate the impact of inter-user interference. These topology settings are obtained by the following parameters:

```
scStr.topology.nodes                    = ['BS1,BS2,UE1,UE2'];
   % 2 cells with one user each
scStr.topology.primaryLinks             = ['UE1:BS1,UE2:BS2'];
   % downlink links only
scStr.topology.interferenceGeneration = 'Automatic';
   % automatic generation of interference links
scStr.topology.attenuation              = 107;
   % channel attenuation fits channel path loss
```

The interfering links from UE2 to BS1 and from UE1 to BS2 are generated automatically due to these settings. The transmit power of UE2 is swept over to obtain inter-user interference of different strengths.[2]

```
scStr.simulation.sweepParam     = {'simulation.txPowerUser'};
   % sweep over a user's transmit power
scStr.simulation.sweepValue      = 10:5:60;
   % transmit power of UE2
scStr.simulation.applySweepingTo = [0,1];
   % apply sweep to second user only
scStr.simulation.pathloss        = 107;
   % channel path loss for an SNR of 40dB
```

While a subcarrier spacing of 15 kHz is used in the first cell, 30 kHz are used in the second cell. The total number of subcarriers of 72 and 36 are chosen such that the total utilized bandwidths are equal in both cells , i.e., $15\,\text{kHz} \cdot 72 = 30\,\text{kHz} \cdot 36$. As the simulator performs numerical computations on a frame basis, frame durations of both cells must match. Therefore, the number of symbols is chosen such that both frames have the same duration, i.e., $1\,\text{ms} = 15\frac{1}{15\,\text{kHz}} = 30\frac{1}{30\,\text{kHz}}$. Also the guard duration (CP length) is chosen accordingly. In an LTE like system, one out of 15 symbols is spit up upon the

[2]In this scenario we consider UE1 to BS1 as the desired link and UE2 to BS1 as the interfering link. Of course, this is an arbitrary interpretation and only serves the description of the scenario.

remaining 14 symbols for CP duration of $\frac{1}{15\,\text{kHz}\cdot 14} = 4.76\,\mu\text{s}$. For matching frame durations, 2 symbol durations are shared among the remaining 28 symbols for a CP duration of $\frac{2}{30\,\text{kHz}\cdot 28} = 2.38\,\mu\text{s}$. Obviously, the factor 2 also appears between the two CP durations of cell one and cell two. The numerology is obtained by the following settings:

```
scStr.modulation.subcarrierSpacing   = [15e3, 30e3];
   % 15kHz subcarrier spacing in cell 1 and 30kHz in cell 2
scStr.modulation.numerOfSubcarriers  = [72, 36];
   % total bandwidth is equal for both cells
scStr.modulation.nSymbolsTotal       = [15, 30];
   % number of symbols such that frame durations match
scStr.modulation.nGuardSymbols       = [1, 2];
   % guard durations for matching frame duration
scStr.modulation.samplingRate        = 30*72*2;
            % a suitable sampling rate
```

For a fair comparison, both users have the same number of total allocated resource elements. A guard band in frequency domain is employed between the users by means of scheduling. This guard band is split among the users such that they both lose the same number of resource elements. This means, UE1 looses 2 subcarriers to the guard band while UE2 looses one subcarrier compared to half the total bandwidth. The schedule is obtained by the following:

```
scStr.schedule.fixedSchedule{1} = [ 'UE1:34,none:38' ];      %
    UE1 gets the upper half of BS1's schedule
scStr.schedule.fixedSchedule{2} = [ 'none:19,UE2:17' ];      %
    UE2 gets the lower half of BS2's schedule
```

Simulation parameters are summarized in Table 3. The effects of interference between users is clearly visible in the simulation results. While the sweep of UE2's transmit power is carried out up to very high values of 60 dBm, this power has to be seen relative to the transmit power of UE1. Since UE1 has a transmit power of 30 dBm, a transmit power of 60 dBm of UE2 means that UE2 is stronger than UE1 by 30 dB. This could correspond to a near UE2 and a far UE1 in an uplink transmission.

The pre-defined waveform for this scenario is OFDM. You may want to run the simulation with this parameters and then change the waveform to f-OFDM or FBMC. Comparing the simulation results, you will observe that the weaker UE2 profits from a waveform with quickly decreasing sidelobes.

5.2.4 Flexible Numerology

Flexible numerology is proposed by 3GPP for New Radio (NR) physical layer design. Numerology refers to the parametrization of the multicarrier

parameter	**value**		
waveform	OFDM	f-OFDM	FBMC
filter type/length	-	7.14 µs	PHYDYAS-OQAM
CP length	4.76 µs	4.76 µs	-
subcarrier spacing	User 1: 15 kHz, User 2: 30 kHz		
guard band	$2\times15\,\text{kHz}+1\times30\,\text{kHz}=60\,\text{kHz}$		
bandwidth per user	$34\times15\,\text{kHz}=17\times30\,\text{kHz}=0.51\,\text{MHz}$		
modulation/coding	64 QAM/LDPC, r = 0.65 (CQI 12)		
channel model	block fading Pedestrian A		

Table 3: Multi-link scenario simulation parameters overview.

scheme. It means that we are flexible to choose different subcarrier spacing and thus symbol and CP duration in order to fulfill different service and user demands. The goal of this scenario file is to show how different numerology behaves according to the different channel conditions, i.e. Doppler shift and delay spread of the channel. We assume a Single-Input Single-Output (SISO) OFDM downlink transmission with parameters as summarized in Table 4. In terms of numerology, we specify subcarrier spacing and number of subcarriers for the desired bandwidth and according to the subcarrier spacing we choose appropriate symbol as well as CP duration [1].

```
scStr.modulation.numerOfSubcarriers = [48];
   % 384, 96, 48
scStr.modulation.subcarrierSpacing  = [120e3];
   % 15e3, 60e3, 120e3
scStr.modulation.nSymbolsTotal      = [120];
   % 14, 56, 112
scStr.modulation.nGuardSymbols      = [8];
   % 1, 4, 8
```

When the total number of subcarriers is changed, the downlink schedule needs to be adapted accordingly.

```
scStr.schedule.fixedScheduleDL{1} = ['UE1:48'];
   % ['UE1:384'], ['UE1:96'] or ['UE1:48']
```

In order to change the Doppler shift we sweep over different user velocities with the carrier frequency of 5.9 GHz that is typical for vehicular communications.

```
scStr.simulation.sweepParam = {'simulation.userVelocity'};
   % sweep over user velocity
scStr.simulation.sweepValue = [5, linspace(50,300,6)./3.6];
   % user velocity values
```

At the same time we employ the Tap Delay Line (TDL)-A channel model with desired Root Mean Square (RMS) delay spread [2].

```
scStr.channel.powerDelayProfile = 'TDL-A_45ns';
   %'TDL-A_45ns' or 'TDL-A_250ns'
```

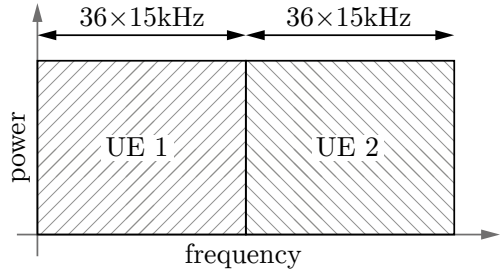
The final results are determined by interplay between Inter-Carrier Interference (ICI) and Inter-Symbol Interference (ISI). Namely, larger subcarrier spacings are more robust to ICI and hence they outperform the smaller subcarrier spacings. Of course, this holds true only with small RMS delay spread channels, since the CP durations of larger subcarrier spacings are sufficient compared to the maximum channel delay spread. If we have high delay spread channels, then due to the insufficient CP duration for large subcarrier spacings we have ISI present, deteriorating our performance. With 120 kHz subcarrier spacing and above, ISI is superior over ICI for the entire range of considered user velocities, making these curves flat.

parameter	**value**		
subcarrier spacing	15 kHz	60 kHz	120 kHz
number of symbols per frame	14	56	112
CP duration	4.76 µs	1.18 µs	0.59 µs
bandwidth	5.76 MHz		
carrier frequency	5.9 GHz		
modulation alphabet	64 QAM		
channel model	TDL-A		
channel RMS delay spread	45 ns or 250 ns		

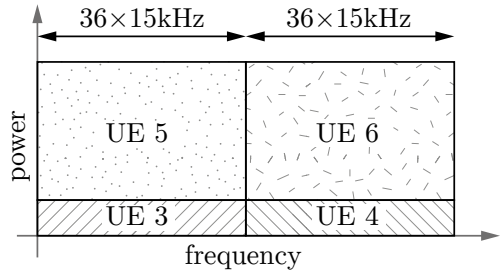
Table 4: Parameters for the flexible numerology simulation example. Three different numerologies are employed.

5.2.5 Non-Orthogonal Multiple Access

The purpose of this scenario is to show the operation of Non-Orthogonal Multiple Access (NOMA) in the simulator, and demonstrate the gain offered by the 3GPP Multi-User Superposition Transmission (MUST). For this, we set up two cells, the first one operates with Orthogonal Multiple Access (OMA), and the other one with MUST. In each cell, the Base Station (BS) splits the bandwidth equally between two User Equipments (UEs) that have good channel conditions (strong users); however, since the second BS supports MUST, it can superimpose those two strong UEs with two other cell-edge UEs (weak users). In this ideal case, a cell overloading of 200% is achieved. The topology of this scenario is setup according to



User schedule for the OMA case.



User schedule for the NOMA case.

Figure 3: User assignment for the NOMA simulation scenario.

```
scStr.topology.nodes              = ['BS1,BS2,UE1,UE2,UE3,UE4,
    UE5,UE6'];
scStr.topology.primaryLinks       = ['BS1:UE1,'...
                                     'BS1:UE2,'...
                                     'BS2:UE3,'...
                                     'BS2:UE4,'...
                                     'BS2:UE5,'...
                                     'BS2:UE6'
                                    ];
```

Next we setup the strong and the weak users. This is done by adjusting their corresponding path loss

```
scStr.simulation.pathloss         = [80, 90, 80, 90, 110, 115];
```

Therefore, UE1 link gets a path loss of 80 dB, UE2 gets 90 dB, etc. Next, we set the multiuser mode of the BSs, which is achieved via

```
scStr.schedule.multiuserMode.Downlink    = {'none', 'MUST'};
```

This indicates that the first BS does not employ any multiuser transmission mode, while the second BS employs MUST. Also, we set the power ratios for each BS

```
scStr.modulation.MUSTIdx          = [0, 2];
```

which indicates that the second BS employs the second power ratio from the standard. For the first BS, it does not matter what ratio we assign (here it is 0), because its multiuser mode is set to 'none' anyway, and thus it is ignored. Finally, we set the schedule for the superimposed users. This is achieved by setting the downlink schedule as follows

```
% BS1 does Orthogonal Multiple Access
scStr.schedule.fixedScheduleDL{1}    = ['UE1:36,UE2:36'];

% BS2 does MUST operation
scStr.schedule.fixedScheduleDL{2}    = ['UE3:36,UE4:36,UE5:UE3
    ,UE6:UE4'];
```

The assignment `UE5:UE3` indicates that `UE5` uses the same resources as `UE3`, or in terms of MUST, `UE5` is superimposed on top of `UE3`. Similar is the case for `UE6:UE4`. The user assignment is illustrated in Fig. 3. Note that even though we use the notion of "superimposed", the bits to symbols mapping for both users is done in a joint manner, such that the resultant multiuser constellation is Gray-mapped. Additional information is provided in section Section 16.

Table 5 summarizes the simulation parameters for this scenario. The resulting curves show that MUST allows the BS to support more users, and when combined with a sufficiently high transmit power, it offers a higher downlink spectral efficiency.

parameter	**value**	
cells	OMA	NOMA
number of users	2	4 (2 strong, 2 cell-edge)
path-loss	80, 90 dB	strong: 80, 90 dB cell-edge: 110, 115 dB
NOMA receiver	-	ML
MUST power-ratio	-	fixed (second ratio)
bandwidth	1.4 MHz (72 subcarriers)	
waveform/coding	OFDM, LDPC	
MIMO mode	2×2 CLSM	
modulation/code rate	adaptive (CQI based)	
feedback delay	no delay (ideal)	
channel model	Pedestrian A	

Table 5: The simulation parameters for the NOMA example scenario.

5.2.6 Massive MIMO

In this simulation scenario, the massive MIMO capabilities of the Vienna 5G Link Level simulator are demonstrated. To this extent, a multi-user MIMO scenario with two BSs is set up, with four users per cell. Users 1-4 are associated with cell one whereas users 5-8 are connected to cell two, both for downlink as well as for uplink. The two cells only serve the purpose to enable simulation of two different multi-user MIMO modes in one simulation to facilitate comparison of simulation results. Therefore, to disable the interference between the two cells, the attenuation for the interfering links is set to 300 dB

```
scStr.topology.attenuation                  = 300;
```

This scenario may be run with a FDD frame structure or in TDD mode.

```
scStr.simulation.frameStructure             = 'TDD';    % may
    be 'TDD' or 'FDD'
```

For the TDD mode, the time correlation of the channel between frames needs to be enabled

```
scStr.channel.correlatedFrames              = true;
```

to obtain a continuous channel trace. In this context, a user velocity of 15 km/h is considered in this scenario

```
scStr.simulation.userVelocity               = [15/3.6];
```

To keep the simulation time to a reasonable duration, the channel trace is sub-sampled in time by a factor of 10 in this scenario

```
scStr.channel.timeSubsamplingFactor         = 10;
```

The simulation results in the scenario show the advantage of Zero Forcing (ZF) beamforming over Maximum Ratio Transmission (MRT) beamforming at a small number of antennas. While both sum troughput curves, from cell one and cell two, do of course increase with an increasing number of antennas, the ZF cell one throughput reaches the saturation value for a smaller number of antennas compared to the MRT cell two.

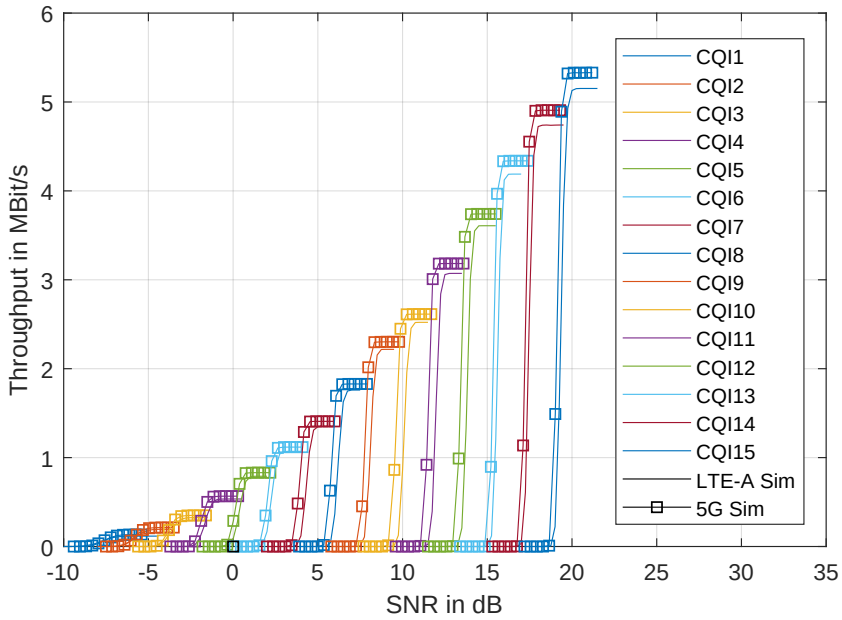
Since the wireless channel is estimated only in the uplink in TDD mode, a higher user velocity will lead to a decreased performance since the channel is changing from uplink to downlink frames over time. If the user velocity is chosen too small, the channel varies only slowly over time such that a high number of simulation frames are required in order to obtain average results for the selected scenario. When a high user velocity is selected, switching between the FDD and the TDD frame structure shows the performance impact of the

outdated channel state information in TDD mode. A better performance is achieved with FDD mode at high velocities since the channel is estimated in the uplink and the downlink for every frame.

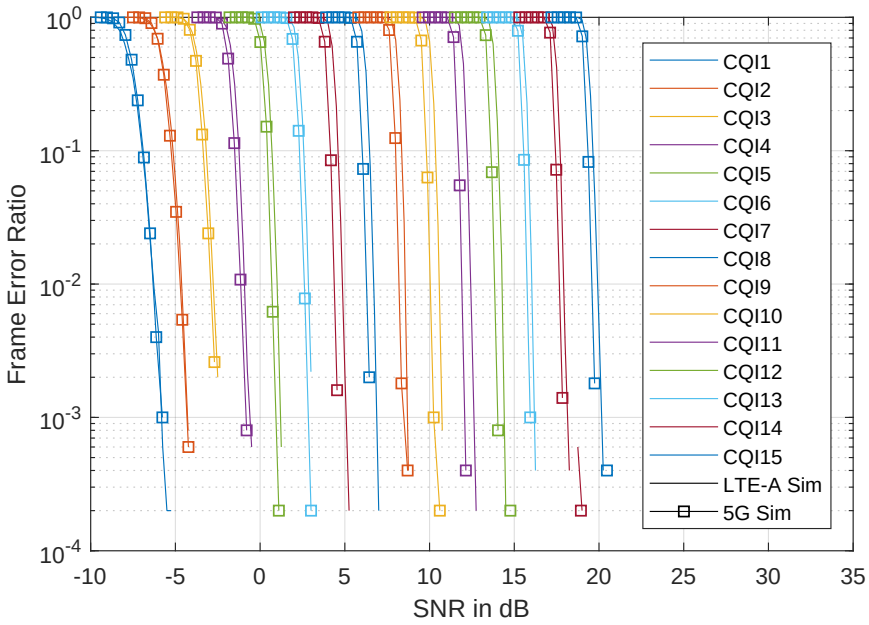
In this comparison, one has to keep in mind that the same number of pilots is actually multiplexed for both, the FDD and the TDD mode. Please note, that although the channel estimation method `'Approximate-Perfect'` is selected, there are still pilot symbols multiplexed at the transmitter side to account for the symbol overhead. For this multi-user MIMO scenario, the `'LTE Uplink'` pilot pattern is employed for the downlink and the uplink. In this case, the pilot symbols for all transmit antennas are allocated on overlapping time-frequency resources, see Section 14. Therefore, the symbol overhead due to pilot symbols does not scale with the number of antennas. Changing the pilot pattern for the downlink to `'Diamond'` leads to a significant decrease in downlink sum throughput since the pilot overhead is increased. Since there are more and more pilots multiplexed when the number of antennas is increased. the throughput even decreases with the number of transmit antennas in this example. This shows the necessity for orthogonal pilot symbols allocated on the very same resources for all antennas in a large scale MIMO system.

6 Comparison to LTE-A

We offer an extremely flexible simulator, that is able to simulate almost any multi-carrier system. Therefore, physical layer methods for future wireless communications systems can be investigated and compared. On the other hand, the Vienna 5G Link Level Simulator is also able to perform LTE-A compliant simulations. To justify this claim, reference simulations for 15 CQI values were carried out on a SISO Additive White Gaussian Noise (AWGN) channel. Simulation results obtained with the Vienna LTE-A Downlink Link Level Simulator and the Vienna 5G Link Level Simulator are shown in terms of throughput and FER in Fig. 4. These results are reproducible via a simulation example, see Section 5.1.3.



(a) Throughput comparison.



(b) Frame Error Ratio comparison.

Figure 4: LTE-A simulator to 5G simulator comparison on SISO AWGN channel for 15 CQI.

7 Simulator Structure

The simulator is built upon a flexible structure that can run complicated tasks, yet allows for easy addition of new features later on. The basis of the structure is the Link object, which carries all the information and functionalities of the connection between nodes (base stations and users) in the cellular map.

7.1 Topology Configuration

The generation of the links between the nodes is based on the supplied 'topology', which basically tells how the nodes are connected together. For example, in order to configure the simulator for the topology shown in Figure 1, open the file Scenarios.genericScenario.m, and modify topology.nodes and topology.primaryLinks as follows

```
scStr.topology.nodes         = ['BS1,BS2,UE1,UE2,UE3'];
scStr.topology.primaryLinks = [ 'BS1:UE1,' ...
                                'BS1:UE2,' ...
                                'BS2:UE3,' ...
                                'UE1:BS1,' ...
                                'UE2:BS1,' ...
                                'UE3:BS2'];
scStr.topology.interferenceGeneration   = 'Automatic';
scStr.topology.attenuation              = 30; % in dB
```

The first property topology.nodes contains all the nodes in the network. The nodes names must be entered in an ascending order, i.e. UE1,UE2,UE3 and the node number must be larger than zero, i.e. BS0 or UE0 are not allowed. Once the participating nodes are entered, the next step is to define the connected links. The property topology.primaryLinks takes care of that, where each line indicates a connection. For each line, the first entry is the transmitting end, the second entry is the receiving end. For example, 'BS1:UE1' indicates a downlink from BS1 to UE1, 'UE2:BS1' indicates an uplink from UE2 to BS1, etc. The two entries are separated by a colon, and each line is terminated by a comma.

Interference from other cells is defined in the same manner using additional links. These interference links can be setup automatically by setting topology.interferenceGeneration to 'Automatic', in which all possible interference links between the nodes in the different cells are automatically generated. The strength of the interference can be controlled by introducing attenuation to the interference links. This is set in dB through topology.attenuation, where low values indicate high amount of inter-cell interference. Alternatively, custom configuration of the interference links is also possible. To see how the attenuation property works, check out the example scenario in Section 5.2.3.

7.2 Links Generation

Once the desired topology is entered, the links are generated using the function Topology.getTopology. The output of this function are three collections: BS, UE, Links. The BS collection contains the base stations, UE contains the users, and Links contains the generated Link objects. All the generated nodes (base stations and users) have unique IDs. These IDs are used to access the associated links between these nodes.

The Links collection has a very specific structure, namely, it is a 2D cell, where the rows indicate the transmitters ID, and the columns indicate the receivers ID. Assume you would like to access the downlink and uplink between BS1 and UE2, this can be done as follows

```
1 downlink = Links{BS{1}.ID, UE{2}.ID};
2 uplink   = Links{UE{2}.ID, BS{1}.ID};
```

As stated above, the row index is the transmitter ID, and the column index is the receiver ID. Such unique ID association is important to avoid conflict between the uplink and downlink, otherwise they would share the same index.

7.3 The Link Object

As mentioned above, the building block of the structure is the Link object. On the one hand it contains all the properties that define a connection between two nodes, like the connected nodes IDs, scheduled resources, path loss, receiver SNR, velocity, etc. It also contains all the generated signals throughout the transmission, like the data bit stream, modulation symbols, soft bits, etc. On the other hand, it offers signal processing functionalities like channel coding, modulation, MIMO processing, the transmission channel, equalization, feedback, etc.

Such structure simplifies the simulation of multiple base stations and users, since each link (connection) can have a different set of parameters, which are grouped together and accessed under the same package. This in turn, greatly reduces the processing overhead, since all the information is available locally to the object.

7.4 Some Rules

Almost every parameter can be applied either to each node or link individually, or it can be applied globally. This can greatly simplify the way the parameters are entered in the scenario file. Let us consider the following example. Assume you have the following topology

```
scStr.topology.nodes = ['BS1,BS2,UE1,UE2'];
```

Now you would like to have both base stations transmitting with 30 dBm power. This can be either entered individually

```
scStr.simulation.txPowerBaseStation = [30,30];
```

or, since both of them have the same value, it can be entered globally. To do so, you only have to enter the parameter once, i.e.,

```
scStr.simulation.txPowerBaseStation = [30];
```

and the simulator will take care of applying the parameter to all the nodes or links. The `txPowerBaseStation` is a per BS parameter. Similar thing can be applied to a per link parameter such as the `pathloss`. For example, assume you have the following links

```
scStr.topology.primaryLinks     = ['BS1:UE1,'...
                                   'BS1:UE2,'...
                                   'BS2:UE3,'...
                                   'BS2:UE4,'...
                                   'BS2:UE5,'...
                                   'BS2:UE6'
                                  ];
```

Then setting the `pathloss` according to

```
scStr.simulation.pathloss         = [80,90,70,88,110,115];
```

indicates that `'BS1:UE1'` has a pathloss of 80 dB, `'BS1:UE2'` has a pathloss of 90 dB, `'BS2:UE3'` of 70 dB, etc. Is the parameter per node, per BS, per UE, or per link. In the next subsection we mention some of them and how they are used, and also in the scenario files that are supplied with the simulator, we give a description for each parameter and its type.

7.5 General Simulation Parameters

This subsection is dedicated to the category of the general simulation parameters. They allow you to control the flow of your simulation, and in certain cases allow you to explore new simulation ideas. For the other set of parameters, such as the channel, MIMO, modulation, coding, etc, you get to know them throughout the other sections of the manual.

Let start by listing all of them below

```
%% General Simulation Parameters
scStr.simulation.simulateDownlink  = true;
scStr.simulation.simulateUplink    = false;
```

```
scStr.simulation.simulateD2D        = false;
scStr.simulation.frameStructure     = 'FDD';

scStr.simulation.plotResultsFor     = [1];
scStr.simulation.plotOverSNR        = true;
scStr.simulation.plotPAPR           = false;
scStr.simulation.saveData           = false;

scStr.simulation.sweepParam         = {'simulation.pathloss'};
scStr.simulation.sweepValue         = linspace(130,80,6);
scStr.simulation.applySweepingTo    = [1];

scStr.simulation.nFrames            = 100;
```

The first three parameters `simulateDownlink`, `simulateUplink`, `simulateD2D` control the direction in which the simulation is carried out. These can be useful when the loaded scenario has a lot of connections in the downlink, uplink, and D2D, and then at certain point you are interested in simulating only a single direction. The parameter `frameStructure` specified whether the transmission is FDD or TDD. For more information, check out Section 8.

The next set of parameters control the results processing. The parameter `plotResultsFor` manages which nodes will get their results plotted. When you set this parameter to 1, then all the nodes will get their results plotted, while if you set it to zero, then no plots will be shown. It is also possible to show the plots of only a certain number of nodes. For example, assume your topology is defined as follows

```
scStr.topology.nodes = ['BS1,BS2,UE1,UE2'];
```

and now if you want to show the results for only BS2 and UE2, then you need to set their corresponding positions in the `plotResultsFor` to 1, while set the others to 0, i.e.,

```
scStr.simulation.plotResultsFor     = [0,1,0,1];
```

The parameter `plotOverSNR` plots the final results in terms of the Signal to Noise Ratio (SNR) when it is set to 1. This only works when the sweep parameter is the pathloss. Parameter `plotPAPR` indicates whether the Peak-to-Average Power Ratio (PAPR) of the transmit signals is plotted or not. The last parameter in this set is `saveData`, and it controls whether the whole generated signals (input bits, symbols, decoded bits, etc) are saved in the results, or only the instantaneous (per frame) FER and BER are saved. It is recommended to keep it turned off, especially if you are going for a lot of realizations and simulation points.

The next group of parameters is the sweep parameter set. The first entry `sweepParam` selects the parameter that will be swept over. In the

generic scenario file, you can find a description in front of each parameter that indicates whether the parameter is per BS, UE, or link. The values of sweeping are set through the parameter `sweepValue`; this works for most of the values as long as they are valid. The next parameter `applySweepingTo` is the most exciting here, as it allows you to setup more advanced scenarios. As the name suggests, it allows you select which nodes or links are swept over. The selection is done in a similar way as with `plotResultsFor`; you basically insert a 1 in the position of the link or node, and set the rest to zero. Then only the selected links/nodes will be swept over, while other nodes will use the default values that are entered globally. Entering only a single 1 means that all the links/nodes will be swept over.

To demonstrate how it works, let us consider the following setup (showing only the relevant parameters)

```
scStr.topology.nodes                = ['BS1,UE1,UE2'];
scStr.topology.primaryLinks         = ['BS1:UE1,' ...
                                       'BS1:UE2'];

scStr.simulation.sweepParam         = {'simulation.pathloss'};
scStr.simulation.sweepValue         = linspace(130,80,6);
scStr.simulation.applySweepingTo    = [0,1];

scStr.simulation.pathloss           = [80];
```

The pathloss is a per link parameter, and therefore setting `applySweepingTo` to `[0,1]` means that link `BS1:UE1` will not be swept over, but rather it will take the value that is entered in `simulation.pathloss`. In our example it is `80` and it will continue to have this value until the simulation end. Link `BS1:UE2` on the other hand will be swept over and its path loss value is given by the current sweep value in the simulation loop. Let us now consider another example that uses `userVelocity`. Consider the following setup (again showing only the relevant parameters)

```
scStr.topology.nodes                = ['BS1,UE1,UE2,UE3'];
scStr.topology.primaryLinks         = ['BS1:UE1,' ...
                                       'BS1:UE2,' ...
                                       'BS1:UE3'];

scStr.simulation.sweepParam         = {'simulation.
   userVelocity'};
scStr.simulation.sweepValue         = linspace(0,250,10);
scStr.simulation.applySweepingTo    = [0,0,1];

scStr.simulation.userVelocity       = [5];
```

In this setup, only UE3 will have its velocity changing during the simulation.

UE1 and UE2 will have their velocity fixed to 5 m/s. Check out the scenario in Section 5.2.3 to see how useful this can be.

The last parameter `nFrames` determines the number of simulated frames. 100 frames is typically a good starting points for a quick simulation with tolerable accuracy.

8 Framestructure (Duplexing Mode)

Currently we offer a FDD and a TDD frame structure. In the classical FDD mode, the uplink and downlink transmissions are performed on different frequency bands, at the same time. Assuming that the separation in frequency domain between those uplink and downlink bands is large, there is no correlation between their wireless channels. Therefore, we assume the uplink and downlink channels to be independent and generate them separately for the FDD mode. Please note that our simulator considers complex baseband signals for transmission. The described uplink and downlink bands at different frequencies are not explicitly implemented in our link level simulator. The simulation parameter carrier frequency is only used to calculate the maximum Doppler shift from the user velocity and thereby influences the generation of the wireless channel.

In the TDD mode, the uplink and downlink transmissions are performed at the same frequency band but at different times. This means that transmission frames are assigned to uplink transmissions and downlink transmissions in an alternating fashion. In this case, the time correlation of the wireless channel may be exploited since the frame duration is small. This is currently considered in the context of massive MIMO. The wireless channel is estimated from the pilots in the uplink transmission. Due to the correlation in time domain, the channel will not change much during the frame duration. The channel estimated in the uplink phase is then used to calculate the feedbak parameters (beamforming) for the downlink transmission. This facilitates multi-user MIMO transmissions with a high number of antennas, as there is no pilot overhead needed in the downlink transmission phase.

In our simulator, we achieve the generation of a continuous channel trace for uplink/downlink channel pairs, by assigning the same random number seed to a uplink/downlink link pair. The feedback parameters and the precoder for the downlink transmission is calculated from the estimated uplink channel. When a feedback delay of n is selected, the feedback is calculated from the uplink frame $2n+1$ frames prior to the considered downlink frame. Therefore, to enable TDD transmission with feedback and multi-user MIMO, a time-correlated channel generation must be selected and a user speed greater than zero should be used. Currently, only the custom feedback mode is supported for TDD multi-user MIMO. Exemplary simulation parameters:

```
% set duplexing mode (frame structure)
scStr.simulation.frameStructure    = 'TDD';   % 'TDD' or 'FDD'

scStr.simulation.userVelocity      = [5/3.6]; % per UE;
    velocity in m/s
```

```
scStr.channel.correlatedFrames      = true;     % channel time
    correlation between frames

% MIMO mode
scStr.modulation.transmissionMode = 'custom';
scStr.schedule.multiuserMode      = {'ZF-MUMIMO'};
```

In the current implementation, odd frames are considered as uplink frames and even frame are considered as downlink frames. Both currently have the same length that is set via the simulation parameters. When the TDD mode is selected, the same links must be scheduled for uplink and downlink, that is, for each link there must be a downlink/uplink pair. For example:

```
scStr.topology.nodes         = ['BS1,UE1,UE2,UE3'];

% Primary (desired) links
scStr.topology.primaryLinks = ['BS1:UE1,' ...
                               'BS1:UE2,' ...
                               'BS1:UE3,' ...
                               'UE1:BS1,' ...
                               'UE2:BS1,' ...
                               'UE3:BS1'  ...
                              ];
```

9 Channel Models

As the aim of LL simulation is acquisition of the average link performance, many random channel realizations are necessary per scenario. There exists no network geometry and therefore no path loss model. A link's path loss is an input parameter, determining the user's average SNR. Therefore, this parameter dictates the average channel power while the channel model only includes small scale fading effects. The simulator employs two types of channel modelling: TDL and Spatial Channel Model (SCM), supporting features such as time and frequency selectivity, fading statistics, spatial and temporal correlation, directional information, etc. Features of the TDL-based models are introduced first, while the SCM is described in Section 9.1.

The frequency selectivity is adjusted via the chosen tapped delay line model. Currently we offer implementations for the Pedestrian A, Pedestrian B, Vehicular A [3], from TDL-A to TDL-C [2], Extended Pedestrian A and Extended Vehicular A [4] channel models. For the TDL-A to -C channels, the effective RMS delay spread can be set explicitly the following way:

```
scStr.channel.powerDelayProfile = 'TDL-A_45ns';
```

In order to turn on the channel's time selectivity, the user velocity must be greater than zero, i.e:

```
scStr.simulation.userVelocity = [14];  % velocity in m/s
```

To model it, the fading taps change over time to fit a certain Doppler spectrum. The supported spectra are uniform Dopper and Jakes', both continuous as well as discrete. Using discrete spectra is more computationally efficient as it assumes discrete Doppler shifts, but is only valid for high user velocity. Otherwise, using a (true) continuous spectrum is recommended.

Additionally one can specify the fading statistic of the impulse response via the Two-Wave with Diffuse Power (TWDP) model [5] which generalizes the Rayleigh and Rician fading models. In contrast to the Rayleigh one, where only diffuse components are considered, and the Rician model, where a single specular component is added, two specular components together with multiple diffuse components are considered in TWDP. This allows for a more general description of small-scale fading phenomena in which the two dominant components can interfere destructively. In order to use the TWDP model, the user velocity must be set to 0, which corresponds to the time-invariant case. In case of time-selectivity, Rayleigh fading will be automatically set. The two key parameters for this model are K and Δ. They can be added to a scenario file the following way:

```
scStr.channel.K = 4;
```

Fading statistic	K	Δ
No fading	∞	0
Rician	> 0	0
Rayleigh	0	-
Hyper-Rayleigh	∞	$\approx$1

Table 6: Parameters of the TWDP fading model.

```
scStr.channel.delta = 0;
```

Similar to the Rician fading model, the parameter K represents the power ratio between the specular and diffuse components and is always positive. The parameter Δ is related to the ratio between peak and average specular power and thus describes the power relationship between the two specular components, and is limited to the interval from 0 to 1. By the clever choice of K and Δ, the TWDP fading model is able to characterize small scale fading for a wide range of propagation conditions, from no fading to hyper-Rayleigh fading. Table 6 shows typical parameter combinations and their corresponding fading statistic, with exemplary plots of the Empirical Cumulative Distribution Function (ECDF) of the corresponding impulse response envelope $|h|$ shown in Fig. 5. The analytical Rayleigh reference has a variance of $\sigma^2 = \sqrt{0.5}$. The TWDP fading model allows for destructive interference between two dominant specular components which results in worse than Rayleigh fading, depending on the model parameters.

The continuous Jakes' spectrum as well as the SCM support temporal correlation between frames which is controlled over the logical parameter:

```
scStr.channel.correlatedFrames = true;
```

Setting this parameter to true leads to a continuous channel trace not only during, but also between frames. Spatial correlation of MIMO channels is supported by every channel model and achieved in different ways, depending on the model choice. The SCM achieves it by incorporating users' spatial information in the model, see Section 9.1. In this sense it is inherent in the model. All other channel variations achieve spatial correlation via a Kronecker correlation model with correlation matrices according to [6]. It allows for three distinguished levels of correlation, namely low, medium and high, and is controlled over the parameter:

```
scStr.channel.spatialCorrelation = 'none';
```

For time-varying channels, the generation of the channel realizations can be time consuming. The slowdown occurs due to the calculation of the

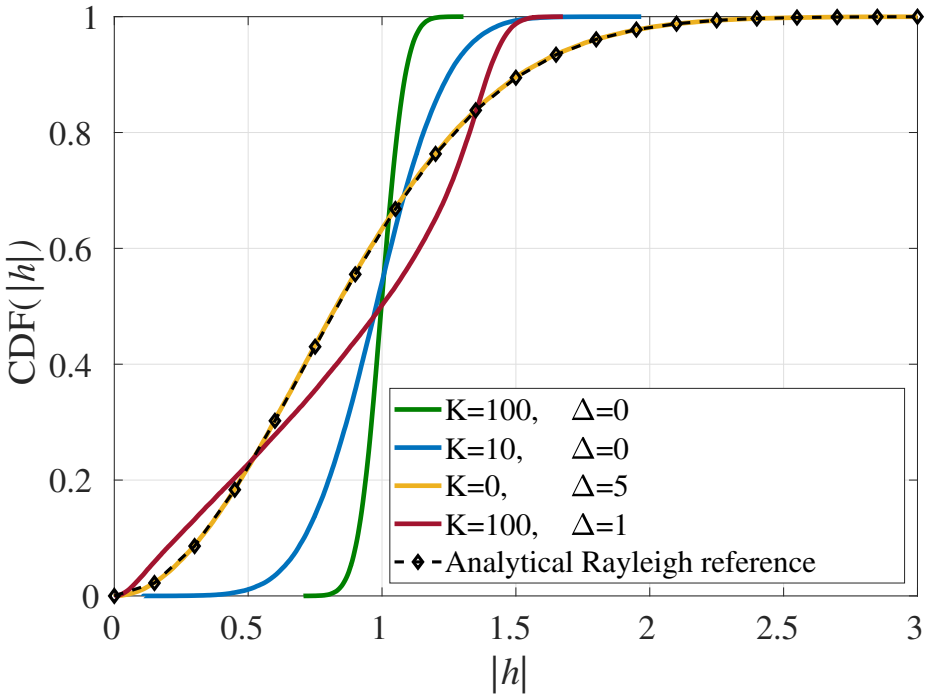


Figure 5: Empirical CDF of the impulse response envelope $|h|$.

impulse response across a large number of time samples. In order to speed up the process, subsampling of the impulse response is employed. Instead of calculating the impulse response at each time sample, it is only calculated for a subset (equidistantly spaced), and then the impulse response at the remaining samples is obtained via nearest-neighbor interpolation. This, of course, can cause a loss in the accuracy of the results, especially for very fast time-varying channels; however, the speed up in the simulation time is definitely worth it. The subsampling is controlled via the parameter `timeSubsamplingFactor`, for which a factor of one indicates that no subsampling is performed. A factor of ten is a good starting point, and indicates that each ten samples of the original impulse response are represented by single sample in the subsampled one. To illustrate the impact of subsampling, an example trace of the impulse response for a user moving at 100 m/s is shown in Fig. 6 with and without subsampling (factor of ten).

As a final remark, here is one possible initialization of the channel object:

```
Channel.FastFading(...
        2.94e6,... % sampling rate
        'Flat', ... % Power delay profile
        200, ... % total number of samples
```

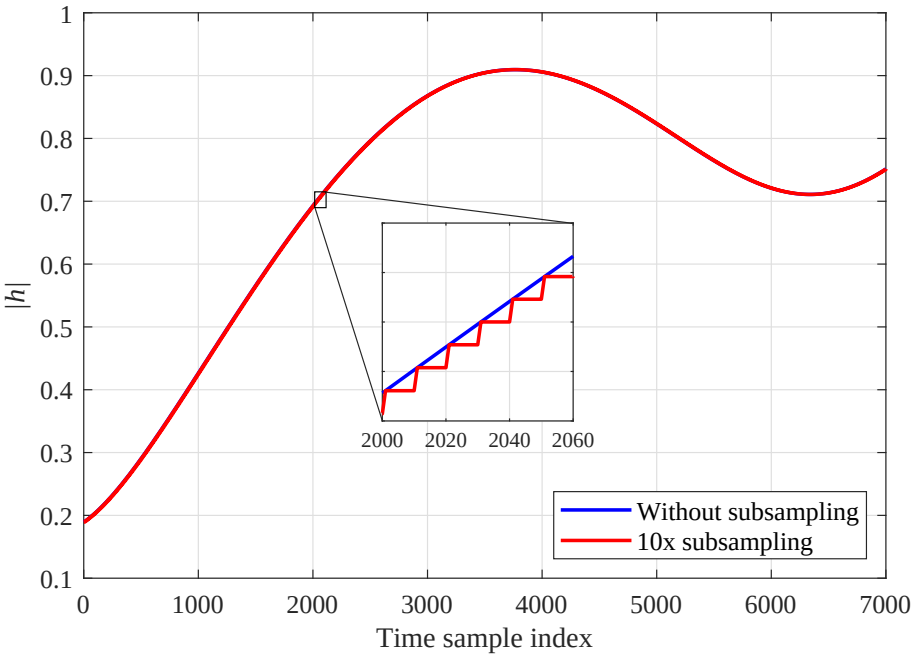


Figure 6: Impact of subsampling on the resultant impulse response.

```
        0, ... % Maximum Doppler shift
        'Jakes',... % Doppler model
        200,... % number of paths
        false,... % time correlated fading
        'none',... % spatial correlation of MIMO channels
        0,... % spatial correlation coeff TX
        0,... % spatial correlation coeff RX
        1,... % number of transmit antennas
        1,... % number of receive antennas
        true,... % show checks
        10,... % TWDP model parameter K
        0.5... % TWDP model parameter Delta
        false,... % Spatial channel model parameters
        [1, 1],... % N of Tx horizontal and vertical antennas
        [1, 1],... % N of Rx horizontal and vertical antennas
        28e6,...   % Carrier frequency
        10... % Subsampling factor for time-varying channels
);
```

9.1 3D Spatial Channel Model

The LL simulator also supports a directional channel model inspired by 3GPP's 3D SCM [2] in the sense that transmitter locations are defined based on angles of received and transmitted signals, called Angles Of Arrival (AOAs) and Angles Of Departure (AODs), respectively. These angles are modelled stochastically with a certain probability distribution. The SCM can be selected by the following parameter:

```
scStr.channel.spatialChannelModel = true;
```

The Receiver (RX) location is fixed at the origin of a Cartesian Global Coordinate System (GCS) and using the random spherical zenith θ and azimuth ϕ angles, which are defined as shown in Fig. 7, the Transmitter (TX)'s angular position can be calculated over the spherical unit vector $\boldsymbol{p}$:

$$\boldsymbol{p}_{RX} = \begin{pmatrix} x_{RX} \\ y_{RX} \\ z_{RX} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{9.1}$$

$$\boldsymbol{p}_{TX} = \begin{pmatrix} x_{TX} \\ y_{TX} \\ z_{TX} \end{pmatrix} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix} \tag{9.2}$$

The channel impulse response is modelled as a combination of a single Line-Of-Sight (LOS) and multiple Non Line-of-Sight (NLOS) components, whose power ratio is governed by a Rician K-factor, an input parameter. The LOS component models the direct RX-TX path and therefore arrival and departure angles are deterministically determined over the Exterior Angle Theorem. Thus only LOS AOAs are drawn from the following Normal distributions:

$$\theta_{AOA} \sim \mathcal{N}\left(\mu_{\theta_{AOA}}, \sigma^2_{\theta_{AOA}}\right), \tag{9.3}$$

$$\phi_{AOA} \sim \mathcal{N}\left(\mu_{\phi_{AOA}}, \sigma^2_{\phi_{AOA}}\right), \tag{9.4}$$

where the sufficient statistic (mean and standard deviation) is an input parameter, entered in a scenario file the following way:

```
scStr.channel.angleMeanAOA = [90]; % per Link; in Degrees;
   Mean Azimuth angle Of Arrival (AOA); Serves as receiver
   orientation and angular distribution for the LOS part,
   between [0,360]
scStr.channel.angleMeanZOA = [45]; % per Link; in Degrees;
   Mean Zenith angle Of Arrival (ZOA); Serves as receiver
   orientation and angular distribution for the LOS part,
   between [0,180]
```

```
scStr.channel.angleSigmaAOA = [5]; % per Link; in Degrees;
    Standard Deviation of the AOA distribution
scStr.channel.angleSigmaZOA = [5]; % per Link; in Degrees;
    Standard Deviation of the ZOA distribution
```

The mean values represent the transmitter's azimuth and zenith orientation, relative to the receiver, and the standard deviations represent the angular spread of the incoming waves. The NLOS components are assumed indepen-

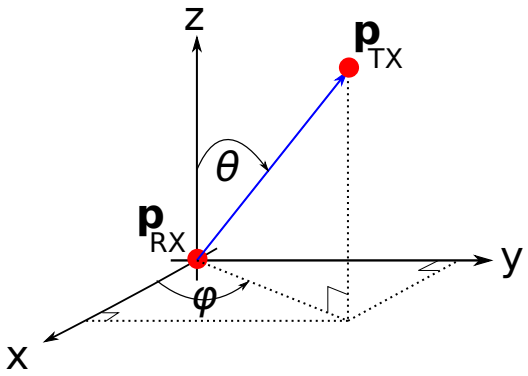


Figure 7: Spherical angles definition in a Cartesian coordinate system.

dent as they model multipath effects such as reflection and scattering. Both NLOS AOAs and AODs are sampled from the following uniform distributions:

$$\theta_{AOA,AOD} \sim \mathcal{U}\,[0, 360) \tag{9.5}$$

$$\phi_{AOA,AOD} \sim \mathcal{U}\,[0, 180) \tag{9.6}$$

The number of NLOS multipath components is an input parameter. Heuristically, a minimum of 10 paths is recommended.

The supported antenna array configurations at the BS are the Uniform Linear Array (ULA) and Uniform Planar Array (UPA). They are deterined by the number of antenna elements positioned along the x-axis, N_x, and the number of antenna elements along the y-axis, N_y, which can be entered in a scenario file the following way:

```
scStr.simulation.nAntennasBaseStation = [10]; % per BS;
    number of antennas at the BS
scStr.simulation.antennaConfiguration = {[Nx,Ny]}; % per BS:
    if the 5G codebook is selected, each vector represents the
     number of horizontal and vertical antennas[Nx,Ny]
```

Only the ULA is supported at the UE terminal and the total number of antenna elements can be entered as:

```
scStr.simulation.nAntennasUser = [4]; % per UE; number of
    antennas at the user
```

In terms of polarization the supported antenna configurations at the BS are single-element vertically polarized and double-element cross-polarized. The type of polarization is given by the total number of BS antennas and product of antenna elements $N_x \cdot N_y$. In case of equality single-element vertical polarization is assumed. If the product of antenna elements is double the number of antennas, double-element cross-polarization is assumed. At the UE terminal, only single-element vertical polarization is supported. Additionally, one can also specify the inter antenna element spacing as a multiple of the signal's wavelength λ by specifying the following parameter in one of the scenario files:

```
scStr.channel.antennaSpacing = [1/2];
```

The SCM is able to capture both spatial and temporal correlation. The former one is determined by user position proximity in space and depends on a number of parameters. Fig. 8 illustrates the most prominent ones and their impact on correlation, evaluated as normalized inner product between the user's channel vectors $|\rho|$ for a planar, vertically polarized antenna array with $N_x = N_y = \sqrt{N_{antennas}}$. $|\rho|$ is directly proportional to the K-factor as

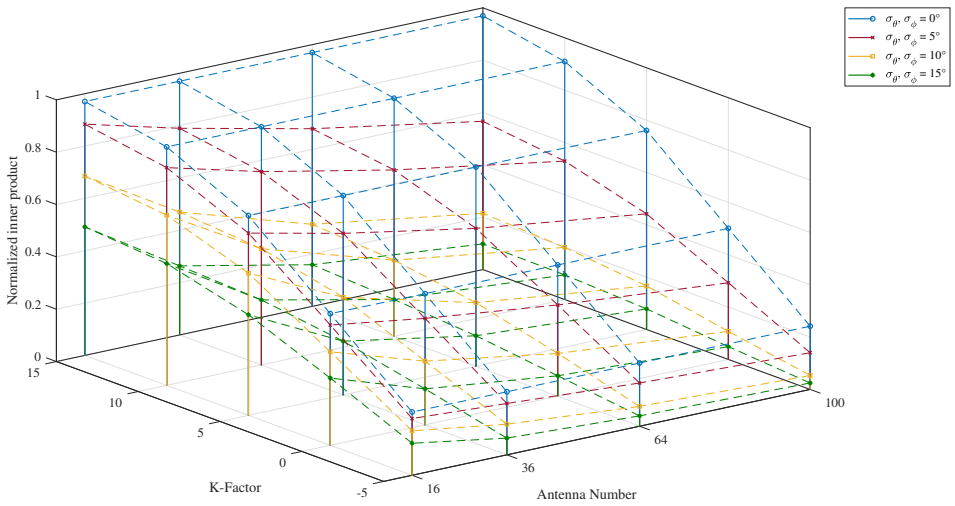


Figure 8: Correlation coefficient parameter dependency.

more energy of the impulse response is concentrated in the LOS part, which

carries the channel's spatial information. The impact of the LOS angular variances $\sigma^2_{\theta_{AOA}}, \sigma^2_{\phi_{AOA}}$ on the correlation coefficient is inversely proportional, as the angular spread of incoming rays increases with higher variance. And finally, the number of antennas determine the spatial resolution of the array. This in term determines how well it can differentiate between signals coming from sources closely positioned in space.

The frequency selectivity is implemented similar to the TDL models, where the NLOS part accumulates the delayed echoes of the signal. How long the delays are, their number and the power distribution between them are modelled after standard Power Delay Profiles (PDPs) and can be selected in a scenario file the following way:

```
scStr.channel.powerDelayProfile = 'PedestrianA';
```

For time selectivity, the user velocity must be set to greater than zero. In that case, the transmitter is assumed moving radially away from the receiver (since only one node is considered moving here, whether the receiver or the transmitter is moving is interchangable). And finally, temporal correlation is also supported by the SCM. It can be switched on as:

```
scStr.channel.correlatedFrames = true;
```

This assures that channel traces will be continuous not only during but also between frames. It should be noted, however, that when temporal correlation across frames is switched on, the user is advised to set the standard deviation of the angular distributions to zero. Otherwise a sufficiently large sample size won't be achieved and results could be ambiguous.

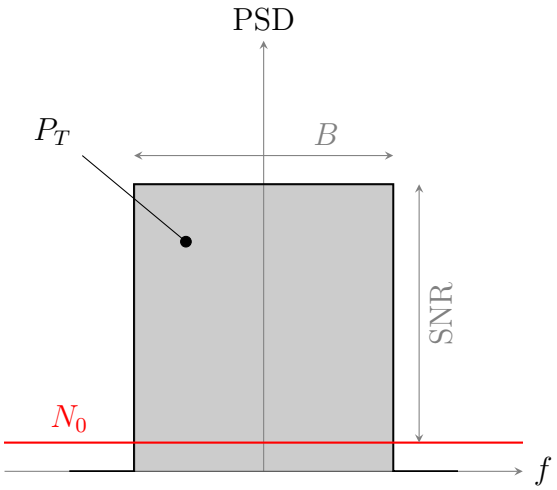


Figure 9: Calculation of SNR.

10 Definition of SNR

With our link level simulator we facilitate simulation of several waveforms, which are considered for 5G. In order to achieve a fair comparison between different waveforms, we fix the transmit power for each node (base station or user). At the receive side, thermal noise of constant Power Spectral Density (PSD) is added to the signal. The resulting SNR then depends on the employed bandwidth of a specific waveform.

The calculation of SNR is illustrated in Fig. 9. The total transmission power is fixed for each node and is denoted by P_T. This power is spend equally on the whole scheduled transmission bandwidth B. Therefore, the signal PSD is scaled such that P_T stays constant for any bandwidth B. This means, the gray shaded region in Fig. 9 corresponds to P_T and has a constant area.

The definition of the transmit power is illustrated in Fig. 10. In this figure, the mean signal power $\mathbb{E}\{|s(t)|^2\}$ is plotted. Here, K denotes the number of symbols per frame and T is the time spacing between two consecutive symbols. The total signal power is then given by

$$P_T = \frac{1}{KT} \int_{-\infty}^{\infty} \mathbb{E}\{|s(t)|^2\} dt \ . \tag{10.1}$$

The thermal noise PSD is denoted by N_0 and constant over frequency. It is given by $N_0 = k_B \vartheta$, with Boltzmann's constant k_B and temperature ϑ.

The SNR is defined as the ratio between signal PSD P_T/B and thermal

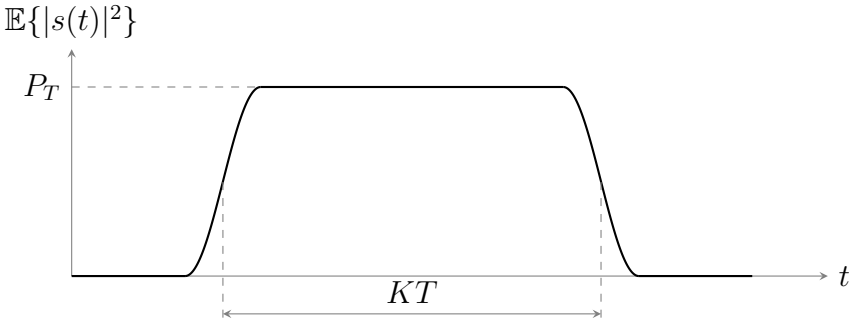


Figure 10: Definition of transmit power.

noise PSD N_0, which yields

$$\mathrm{SNR} = \frac{1}{N_0}\frac{P_T}{B} \ . \tag{10.2}$$

Of course, the received signal power depends on the wireless channel. However, we normalize the channel to have an average power of one. In this sense, the SNR calculated in (10.2) is an average SNR.

To enable sweeps over SNR, several simulation runs are carried out with different transmit powers. While the actual transmission power is fixed for each node, an artificial parameter, referred to as pathloss, is introduced. As there is no network geometry and no large scale fading in our link level simulator, this pathloss value is not a channel property, but only serves the purpose to adapt the transmission power. In this sense, the transmit power P_T in (10.2) is modified as

$$P_T = \mathrm{PL}\, P_T^\star \ , \tag{10.3}$$

where PL denotes the pathloss and $P_T^\star$ is the fixed transmission power.

10.1 Scheduling

For a downlink transmission, B is the total scheduled bandwidth for all users. In this case, the signal PSD increases only when a portion of the whole available bandwidth is unused, that is, when not the whole available bandwidth is scheduled. For an uplink transmission, a user transmits with power P_T on its scheduled bandwidth B. In this case, the signal PSD increases with decreasing scheduled bandwidth B, such that P_T remains constant.

From the previous paragraph it is clear, that the SNR depends on the scheduled bandwidth. A time variant schedule, that is, a resource allocation that changes from transmission frame to transmission frame, therefore means a different resulting SNR in each frame. Predicting the resulting SNR and

the number of simulated frames per SNR value is not straight forward, thus significantly complicating the simulation procedure. Further, obtained results need to be sorted according to the SNR for plotting and interpretation. We therefore do not allow for time variant scheduling in our link level simulator.

11 Channel Coding

The first block in the processing chain is the channel coding, where redundancy is added to provide error correction and detection capabilities for the wireless transmission. The simulator supports the four candidates (or were candidates) of 5G: convolutional, turbo, LDPC, and polar codes. The aim was to have a single structure that can handle the four coding schemes simultaneously, with challenges arising due to the different requirements of the different coding schemes. Table 7 summarizes the supported schemes, their construction, and the corresponding decoding algorithms.

Table 7: Supported channel coding schemes.

scheme	**construction/ encoding**	**decoding algorithms**
turbo	LTE	Log-MAP Linear-Log-MAP Max-Log-MAP
LDPC	5G NR	Sum-Product PWL-Min-Sum Min-Sum
polar	currently custom	List-SC CRC-List-SC
convolutional	LTE	Log-MAP Max-Log-MAP

The procedure of the channel coding is identical across all the aforementioned schemes. We describe in the following subsections the main steps.

11.1 Block Length Calculation and Segmentation

The first step is to determine how many information bits are supported by the current transmission. This depends on how many resources are scheduled, number of spatial layers, modulation order, and the code rate. The modulation order and code rate are obtained using the current CQI. Once this is determined and depending on the chosen coding scheme, filler bits might be added to the block when its length does not match the size of the interleaver (in case of the turbo code) or the dimensions of the parity check matrix (for the LDPC code). If the block length is too long, then code

block segmentation is performed. For the turbo and convolutional codes, the segmentation follows the LTE standard, while for the LDPC code, it follows the current 5G specs. For the polar code, the segmentation is similar to that of the turbo code, but it has more granularity in the selection of the block length, since the polar code does not require a strict set of input lengths. Moreover, for LDPC codes, the Transport Block Size (TBS) is determined according to the standard as well. This ensures that the input length would fit the dimensionality of the party check matrix after segmentation.

11.2 Convolutional Code

The implementation of the convolutional code is based on the LTE standard [7]. More specifically, it is a tail-biting convolutional code, meaning that the starting and ending states of the encoder is the same. In the simulator, we pass this state directly to the decoder (i.e., gene-aided). This should not have an impact on the performance of the code, however, it will reduce the decoding complexity, as the decoder does not have to spend time figuring out that state. The encoder is implemented using a shift register, and it is initialized with the last bits of the information block, guaranteeing that the initial and final states of the encoder are the same, i.e., tail-bitten.

The decoder is based on the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [8], which is the efficient implementation of the bit-wise Maximum A-Posteriori (MAP) decoder. The supported algorithms are the 'Log-MAP', that is the original MAP algorithm in the log domain, and the sub-optimal 'Max-Log-MAP' which provides lower complexity.

11.3 Turbo Code

Similar the convolutional code, the turbo code is also based on the LTE standard [7]. The encoder consists of two recursive constituent convolutional encoders. The encoders are initialized with zeros, and the final states are tracked through trellis-termination.

The iterative (turbo) decoding is based on the BCJR algorithm. It supports the 'Log-MAP', 'Max-Log-MAP', and the 'Linear-Log-MAP' algorithms. The latter uses a linear function to approximate the exponential correction term in the original MAP algorithm. The decoder supports Cyclic Redundancy Check (CRC)-based early stopping criterion. In other words, after each decoding iteration, CRC is performed on the corrected codeword, and if it passes the check, then the iterative decoding is stopped.

11.4 LDPC Code

The LDPC code is based on the current 5G NR specs [9]. The employed code is quasi-cyclic and therefore allows easy adaptation of of the parity check matrix to different input lengths. The standard defines two base parity check matrices, or as it is called, a Base Graph (BG). Depending on the code rate and the input length, either BG 1 or BG 2 is used. One BG is better suited for short lengths and low code rates, while the one is more suited for long lengths and high code rates. Thanks to the diagonal and double-diagonal structures of the parity check matrix, the encoding can be carried out with low complexity. The encoding is systematic, and at the output of the encoder, a certain amount of the systematic bits are punctured. These punctured bits never enter the circular buffer.

The decoder is based on layered Belief Propagation [10] or usually called Sum-Product algorithm. The layering is utilized through the Column Message Passing schedule [11]. The supported decoding algorithms are 'Sum-Product', the lower complexity 'Min-Sum', and the 'PWL-Min-Sum', where PWL stands for Piecewise Linear. Similar to the turbo decoder, it uses linear functions to approximate the correction terms. Also, CRC-based early stopping criterion is supported as well.

11.5 Polar Code

The construction of the polar code (i.e., finding the frozen set) is currently based on method in [12]. Once the set is found, the next power-of-two polarization transform is selected. In case the codeword length does not match the generator matrix (polarization transform) size, then the extra positions from the bottom of the generator matrix are set to zero and then removed at the output. At the decoder side, the Log-Likelihood Ratio (LLR) of these positions is set to a high value, reflecting a $+\infty$ LLR.

The decoder is based on Successive Cancellation (SC). The supported decoding algorithms are 'List-SC' and 'CRC-List-SC' [13]. In List-SC, multiple decoding paths are maintained. For each bit, the two possibilities of being zero or one are considered, and the paths with the highest path-metric are chosen. The maximum number of the paths is equal to the list size. At the end of the decoding, the path with the highest path-metric is chosen as the correct one. CRC-List-SC operates in a similar manner; however, at the end, the CRC determines which path is correct.

11.6 Interleaving and Rate Matching

For the turbo and convolutional codes, the rate matching procedure is identical to the standard. The interleaving is carried out directly on each of the three streams at the encoder output, i.e., subblock-interleaving. After interleaving, the streams are passed to the circular buffer, where puncturing and/or repetition is performed in order to meet the output length, and consequently the target code rate. For the turbo code, part of the systematic stream is skipped at the first transmission from the circular buffer.

For the polar code, the whole non-systematic codeword is interleaved. The main rate adaption is carried out at the input of the generator matrix, by adjusting how many bits are frozen. After interleaving, the codeword is passed to the circular buffer, where further slight puncturing or repetition is performed in case the codeword does not exactly match the target length.

As for the LDPC code, we follow the 5G NR chain, in which the systematic codeword is passed to the circular buffer directly without interleaving. The codeword is then punctured/repeated in order to meet the target length. After the codeword is rate matched, it is then interleaved using a rectangular interleaver. The interleaving pattern depends on the modulation order.

11.7 Object Usage

The object is initialized in the following manner:

```
ChannelCodingObj = Coding.ChannelCoding( ...
        'Turbo', ...                        % coding scheme
        'Linear-Log-MAP', ...               % decoding algorithm
        1/2, ...                            % code rate
        8 ...                               % iterations/list size
);
```

Once the object is initialized, the next step is to update the object parameters based on the required input or output length. This can be done in two ways, the first one is

```
inputLength = ChannelCodingObj.update('Output', N, L, R, Qm,
    SoftBufferRatio);
```

where N, L, R, Q_m, and `SoftBufferRatio` are the output length, number of spatial layers, code rate, modulation order, and soft-buffer rate-matching ratio, respectively. This is used when the output length is known and you would like to obtain then number of information bits that are needed at the input. Alternatively, it can be used in the following way

```
ChannelCodingObj.update('Input', K, L, R, Qm, SoftBufferRatio
    );
```

where K is the input length. In this case, the function `update()` does not return anything and the output length is set automatically based on the input length. Beside calculating the input length, the function `update()` prepares the object for codeblock segmentation, and in the case of LDPC and polar codes, the update function also performs LDPC lifting, and construct the appropriate polar code. This function has to be called every time the code rate or the output code length changes. The rest of the object usage is straightforward, the encoding and decoding is carried out through

```
codedBits = ChannelCodingObj.encode(inputBits);
...
decodedBits = ChannelCodingObj.decode(channelLLRs);
```

The parameter `SoftBufferRatio` controls how much of the circular buffer is used (i.e., soft rate matching). Its value ranges from 0 to 1, with 1 indicating that the full circular buffer is used.

12 Modulation Mapping and MIMO Processing

12.1 Modulation Mapping

For each codeword, the block of scrambled and coded bits are mapped according to the modulation alphabet determined by the CQI index. The interpretation of the CQI indices depends on the CQI table. The simulator supports three CQI tables, these tables are based on the 3GPP technical specification for the physical layer procedures [14]. In Table 8, the supported CQI tables and the corresponding symbol alphabet are summarized.

CQI Table	Symbol Alphabet	Table [14]
0	QPSK to 64-QAM	7.2.3-1
1	QPSK to 256-QAM	7.2.3-2
2	QPSK to 1024-QAM	7.2.3-4

Table 8: CQI Mapping Table.

12.2 MIMO Processing

For multiple antenna transmission, the output of the modulation mappers is jointly processed and mapped to different antennas. The joint processing of the output of the modulation mapper comprises of two steps; namely, codeword to layer mapping and precoding.

12.2.1 Layer Mapping

The complex-valued symbols of each of the codewords are mapped onto one or several layers according to the layer mapping table. Mapping the codewords onto the layers depends on the rules specified in the layer mapping table. The simulator supports three layer mapping modes: LTE, 5G and the custom mode. The LTE mode layer mapping is based on Table 5.3.2A.2-1 in [15] and supports up to 4 layers. The 5G mode is based on Table 7.3.1.3-1 in [1] and supports up to 8 layers. For the third mode, the custom mode, the mapping rules are based on the layer mapping table defined in the corresponding scenario file.

```
% Layer mapping mode selection
scStr.layerMapping.mode = 'custom';

% custom layer mapping table
```

```
scStr.layerMapping.table.Uplink                = {1;2;[1,2]};
scStr.layerMapping.table.Downlink              = {1;2;[2,1]};
```

The n-th entry of the layer mapping table corresponds to the layer mapping rule for n layers and the size of the n-th entry corresponds to the number of codewords for n layers. In the example shown in 12.2.1, the layer mapping table has three entries, which means that the maximum number of layers is three. For one layer one codeword is mapped to one layer. The second entry of the table is of size one; thus, one codeword is mapped to two layers. For three layers, two codewords are mapped to three layers. For the uplink the first codeword is mapped to one layer and the second codeword is mapped to two layers, whereas for the downlink the first codeword is mapped to two layers and the second codeword is mapped to only one layer.

12.3 Precoding

The precoder takes the output of the layer mapper and maps it to the different antennas. The pre-coding can be configured in different ways corresponding to different transmission schemes. In what follows, the different precoder configuration for Single User Multiple-Input Multiple-Output (SU-MIMO) and Multi User Multiple-Input Multiple-Output (MU-MIMO) will be presented.

12.3.1 SU-MIMO

The SU-MIMO mode can be selected by setting the multi-user mode parameters for the downlink and uplink to 'none':

```
scStr.schedule.multiuserMode.Downlink = {'none'};
scStr.schedule.multiuserMode.Uplink   = {'none'};
```

The simulator supports four different single-user transmission modes: Open Loop Spatial Multiplexing (OLSM), Closed Loop Spatial Multiplexing (CLSM), Transmit Diversity (TxD) and a custom mode. The transmission mode can be selected by setting the following parameters:

```
% MIMO mode
scStr.modulation.transmissionMode = 'custom';
% MIMO detector
scStr.simulation.receiverTypeMIMO = 'MMSE';
```

If the custom mode is selected and the Precoding Matrix Indicator (PMI) feedback is deactivated, the following precoding parameters have to be configured:

```
% per Link
```

```
2 scStr.modulation.nStreams   = [2];        % Number of active
    spatial streams
3 scStr.modulation.precodingMatrix{1} = 1/sqrt(2)*eye(2,2); %
    Link 1 employed precoding matrix
```

12.3.2 MU-MIMO

MU-MIMO is activated by setting the multi-user mode parameter to one of the supported multi-user MIMO schemes. For the downlink, three schemes are supported: ZF, Block Diagonalization (BD) and MRT. ZF and MRT are implemented for single antenna users, whereas BD is implemented for multi-antenna users and supports multi-stream transmission. The uplink also supports three schemes: ZF, Minimum Mean Square Error (MMSE) and Maximum Ratio Combining (MRC), and all three schemes are implemented for single-antenna users only.
For MU-MIMO, the schedule in the scenario file is ignored and all users assigned to the base station are scheduled to transmit/receive on all available subcarrriers. Furthermore, the MU-MIMO only supports the custom transmission mode and CQI feedback.

13 Modulation Waveforms

13.1 Orthogonal Frequency-Division Multiplexing

CP-OFDM (CP-OFDM) is the most prominent multicarrier scheme and is applied, for example, in Wireless LAN and LTE-A. CP-OFDM employs a rectangular transmit and receive pulse, which greatly reduce the computational complexity. Furthermore, the CP implies that the transmit pulse is slightly longer than the receive pulse, preserving orthogonality in frequency selective channels. Thus, frequency-selective broadband channels transform into multiple, virtually frequency flat, sub-channels (subcarriers) without interference. This allows the application of simple one-tap equalizers, corresponding to maximum likelihood symbol detection in case of Gaussian noise. Furthermore, the channel estimation process is simplified, adaptive modulation and coding techniques become applicable, and MIMO can be straightforwardly employed. Unfortunately, the rectangular pulse in OFDM leads to high out-of-band emissions. This is one of the biggest disadvantages of CP-OFDM. Additionally, the CP simplifies equalization in frequency-selective channels but also reduces the spectral efficiency. In order to reduce the OOB emissions, 3GPP is currently considering windowing and filtering, see the next subsections.

One of our most important implementation aspects is that we consider a fixed sampling rate f_s instead of a fixed Fast Fourier Transform (FFT) length N_FFT, as often done in literature. The main reason for a fixed sampling rate is to enable a fair comparison between different subcarriers spacings and to guarantee that a specific channel power delay profile fits approximately the sampling rate. Additionally, the sampling rate is often predetermined by real world hardware and cannot be changed easily. The relationship between FFT size, sampling rate and subcarrier spacing F is:

$$N_\mathrm{FFT} = \frac{f_\mathrm{s}}{F}. \tag{13.1}$$

Note that the FFT size must be larger or equal than the number of active subcarriers. In practice, the FFT size will always be larger than the number of active subcarriers. For example, in 10 MHz LTE-A, we have 600 active subcarriers and an FFT size of 1024. We also advise to use a (much) larger FFT size than the number of active subcarriers.

The OFDM object can be initialized by:

```
1 OFDMobject = Modulation.OFDM(...
2  L,...      % Number of subcarriers
3  K,...      % Number OFDM symbols in time
```

```
 F,...      % Subcarrier spacing (Hz)
 fs,...     % Sampling rate (Samples/s)
 fI,...     % Intermediate frequency of the 1st subcarrier (Hz)
 false,... % Transmit real valued signal, true/false
 TCP, ...   % Length of the cyclic prefix (s)
 TZG ...    % Length of the zero guard time (s), (frame)
);
```

whereas we always consider a block transmission of L subcarriers and K OFDM symbols. The intermediate frequency f_{I} corresponds to a circular shift of the FFT.

A given symbol vector $\mathbf{x} \in \mathbb{C}^{L \times K}$, for example chosen from a QAM signal constellation, can then be modulated by:

```
s = OFDMobject.Modulation(x);
```

where $\mathbf{s}$ represents the transmitted signal in time. The demodulation, on the other hand, can be performed by applying the following method on the received time signal $\mathbf{r}$

```
y = OFDMobject.Demodulation(r);
```

In case of a back-to-back transmission, $\mathbf{r} = \mathbf{s}$, we will recover the transmitted data symbols, that is, $\mathbf{y} == \mathbf{x}$.

For a concrete implementation of our OFDM object (without unnecessary overhead), we refer to the example file in "Example/Comparison_5GWaveforms.m".

13.2 WOLA

The windowed OFDM scheme is called OFDM with WOLA [16]. At the transmitter, the edges of the rectangular pulse is replaced by a smoother function (windowing) and neighboring WOLA symbols overlap in time. The receiver also applies windowing but the overlapping and add operation is performed within the same WOLA symbol, reducing the inter-band interference. Fig. 11 illustrates the WOLA concept. Compared to CP-OFDM, the time spacing is increased by $T_{\mathrm{w,tx}} + T_{\mathrm{w,rx}}$. However, the CP can usually be reduced because some small interference is often acceptable.

The WOLA object can be initialized by:

```
WOLAobject = Modulation.WOLA(...
 L,...      % Number of subcarriers
 K,...      % Number OFDM symbols in time
 F,...      % Subcarrier spacing (Hz)
 fs,...     % Sampling rate (Samples/s)
 fI,...     % Intermediate frequency of the 1st subcarrier (Hz)
```

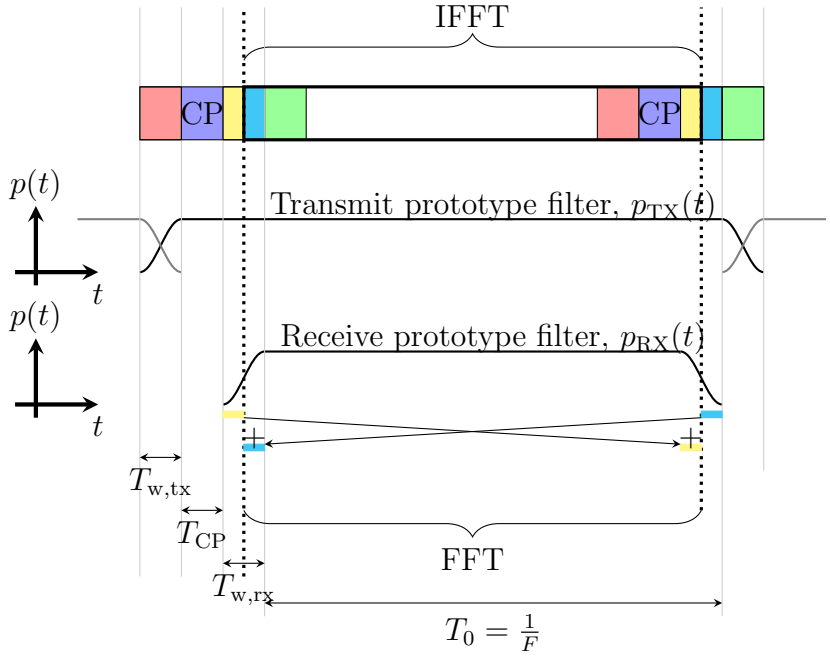


Figure 11: Illustration of WOLA

```
 false,... % Transmit real valued signal, true/false
 TCP, ...  % Length of the cyclic prefix (s)
 TZG ...   % Length of the zero guard time (s), (frame)
 TWTX, ... % Length of the window overlapping (s) at the TX
 TWRX ...  % Length of the window overlapping (s) at the RX
);
```

It works similarly as the OFDM object, see Section 13.1, but has the additional option of $T_{\mathrm{w,tx}}$ and $T_{\mathrm{w,rx}}$, representing the window length at the transmitter and at the receiver. The window function itself is based on a (root) raised cosine function.

13.3 Universal Filtered Multicarrier

UFMC is filtered OFDM technique proposed as one of the candidates for new 5G waveforms below 6 GHz. The main advantage of this scheme is a better fragmentation of spectrum and more suppressed side lobes compared to OFDM.
The implementation of UFMC in this code is based on Nokia's proposal of transceiver structure.[17]

13.3.1 Transmitter

The assigned bandwidth in UFMC is divided into multiple subbands according to different user requirements and services.

```
for m = 1:obj.Nr.ResourceBlocks
n = (m-1)* obj.Nr.SubcarriersPerRb+1:(m-1)*obj.Nr.
    SubcarriersPerRb+obj.Nr.SubcarriersPerRb;
b = reshape(DataSymbolsTemp(n,:),obj.Nr.SubcarriersPerRb,obj.
    Nr.MCSymbols);
end
```

On each of those subbands we apply Inverse Fast Fourier Transform (IFFT) with corresponding length. By doing this we obtain the transmit data in time domain. The choice of the transmit window is arbitrary. We choose Dolph-Chebyshev window since it minimizes the Chebyshev distance of the side lobes for a given main lobe. Before we do subband filtering, we shift each filter to the center frequency of the corresponding subband:

```
freqShift (l) = exp(2*pi*1i*(l-1)*(centralFreq-1)/obj.
    Implementation.FFTSize);
```

We apply this subband filter on the transmit data from the same subband. Unlike OFDM, UFMC uses Zero-Postfix (ZP) instead of CP in order to avoid inter-symbol interference in a case of high delay spread channels. The length of ZP is chosen to be one sample shorter than the filter length. In order to obtain the total transmit signal we summarize all subbands transmit signals together.

13.3.2 Receiver

At the receiver side we apply N-FFT, resulting in the same complexity level as CP-OFDM. In order to apply N-FFT we do some modifications of the received signal. First, we decompose our received signal into two parts, so called *body* and *tail*. Then we transform this received vector by copying the *tail* to the beginning of the signal:

```
receiveSignalTemp(1:guardLength,:) =  ReceivedSignalResh(1:
    guardLength,:) +  ReceivedSignalResh(obj.Implementation.
    FFTSize+1:end,:);
```

13.4 Filtered-OFDM

The second filter-based OFDM scheme considered within 3GPP is f-OFDM [18]. Here, the number of subcarriers for one subband is usually much higher than

in UFMC and often includes all subcarriers belonging to a specific use case. The idea of f-OFDM is quite simple: we modify a conventional CP-OFDM transmission by applying digital filtering at both, transmitter and receiver. If the total CP length is longer than the combined filter length, we restore orthogonality in an AWGN channel. However, some (small) interference is usually acceptable to keep the overhead low. The induced interference can be adjusted by the filter length and the CP length ($T_{\mathrm{CP,f}}$). The filter itself is based on a sinc pulse (perfect rectangular filter) which is multiplied by a Hann window; other filters are also possible [18], but currently not implemented.

The FOFDM object can be initialized by

```
FOFDMobject = Modulation.FOFDM(...
 L,...      % Number of subcarriers
 K,...      % Number OFDM symbols in time
 F,...      % Subcarrier spacing (Hz)
 fs,...     % Sampling rate (Samples/s)
 fI,...     % Intermediate frequency of the 1st subcarrier (Hz)
 false,... % Transmit real valued signal, true/false
 TCP, ...   % Length of the cyclic prefix (s)
 TZG ...    % Length of the zero guard time (s), (frame)
 TFTX, ... % Length of the transmit filter (s)
 TFRX, ... % Length of the receive filter (s)
 TCPF ...   % Length of the additional cyclic prefix (s).
);
```

Again, this is similar to OFDM, see Section 13.1, but with the additional option of $T_{\mathrm{f,tx}}$ and $T_{\mathrm{f,rx}}$, representing the filter length at the transmitter and at the receiver. Furthermore, we have an additional CP with length $T_{\mathrm{CP,f}}$ to combat the effects of filtering. To total CP overhead is then given $T_{\mathrm{CP,total}} = T_{\mathrm{CP}} + T_{\mathrm{CP,f}}$.

13.5 FBMC

Although 3GPP decided that FBMC will not be employed in 5G [19], FBMC still has many advantages over OFDM, namely, much lower OOB emissions and a maximum symbol density (i.e., no CP overhead) [20]. Those advantages, however, come at the price of sacrificing the complex orthogonality condition with the less strict real orthogonality condition. In many cases, however, this has either no, or only a minor influence on the performance. In other cases, such as channel estimation or some MIMO methods, on the other hand, special treatment of the imaginary interference becomes necessary. Fortunately, there exists many efficient methods to deal with those challenges [20]. The signal generation in FBMC is similar to that of windowed OFDM, see Fig. 12, whereas we ignored receive filtering to keep the illustration simple.

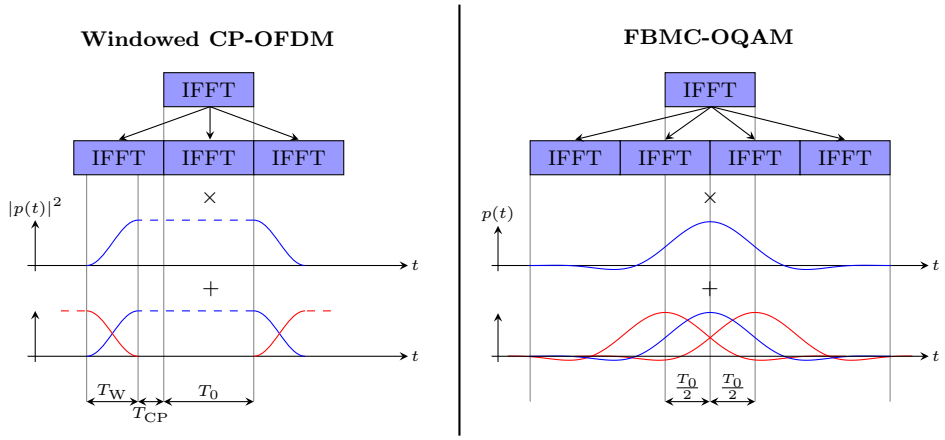


Figure 12: The signal generation in FBMC and windowed OFDM requires the same basic steps [20].

The initialization of the object is similar to OFDM:

```
FBMC = Modulation.FBMC(...
 L,...     % Number of subcarriers
 K,...     % Number FBMC symbols in time
 F,...     % Subcarrier spacing (Hz)
 fs,...    % Sampling rate (Samples/s)
 fI,...    % Intermediate frequency of the 1st subcarrier (Hz)
 false,...% Transmit real valued signal, true/false
 'PHYDYAS-OQAM',... % Prototype filter (Hermite, PHYDYAS, RRC)
 0, ...    % Overlapping factor, e.g., 3,4,5,6,7,8
 0, ...    % Initial phase shift, e.g., pi/2
 true ... % Polyphase implementation, true/false
);
```

14 Channel Estimation

Currently, pilot aided channel estimation is implemented in the Vienna 5G Link Level Simulator. Therefore, pilot symbols are multiplexed with data symbols at the transmit side within the resource grid. Since those pilot symbols are known beforehand, this enables channel estimation at the receiver side. We currently support only LS based channel estimation. The corresponding parameters are:

```
scStr.simulation.channelEstimationMethod    = 'Approximate-Perfect';
% channel estimation method:
% 'Approximate-Perfect'
% 'PilotAided'

scStr.simulation.noisePowerEstimation       = false;
% if the noise and interference power are estimated
% true: estimate interference and noise power at RX
% false: use perfect knowledge of noise power, neglect interference power

scStr.simulation.pilotPatternDownlink       = 'LTE Downlink';
% pilot symbol allocation pattern for Downlink links
scStr.simulation.pilotPatternUplink         = 'LTE Uplink';
% pilot symbol allocation pattern for Uplink links
% 'Rectangular'
% 'Diamond'
% 'LTE Downlink'
% 'LTE Uplink'

scStr.simulation.pilotSpacingFrequency      = 6;
% pilot spacing in frequency domain (may be fractions of subcarriers)
scStr.simulation.pilotSpacingTime           = 3.5;
% pilot spacing in time domain (may be fraction of symbols)

scStr.simulation.pilotSequenceLength        = 12;
% length of orthogonal pilot symbol sequence
% In Multi-user MIMO, this is the maximum number of users supported with channel estimation.
```

The pilot symbols are multiplexed according to the pilot pattern which may be chosen for uplink and downlink separately. Pleas note that even for `channelEstimationMethod = 'Approximate-Perfect'`, pilot symbols are still multiplexed with the data symbols at the transmit side but will not be exploited at the receiver side. This means that the pilot symbol overhead is accounted for, even if perfect channel knowledge is selected.

For the pilot patterns `'LTE Downlink'`, `'Diamond'` and `'Rectangular'` the pilot spacings in frequency and time domain define the density of the multiplexed pilot symbols is defined. For those pilot patterns, the pilot symbols are multiplexed on different (orthogonal) time-frequency resource elements for each transmit antenna. The MIMO channel can therefore be estimated interference free, at the cost of increased pilot overhead for an increasing number of antennas.

For the `'LTE Uplink'` pilot pattern the pilot symbols are allocated on all scheduled subcarriers, on two symbols per frame (one per LTE slot). The time domain spacing parameter and the frequency domain spacing parameter do not have any effect on this pilot pattern. In this case, pilot symbols are allocated overlapping, that is, overlapping (non-orthogonal) in the time-frequency domain for all transmit antennas. In order to enable channel estimation for MIMO transmissions in this case, the pilot symbols are chosen as an orthogonal symbol sequence. Therefore, the orthogonality on the symbol level is exploited at the receiver side to estimate the MIMO channel. For this channel estimation mode, the orthogonal pilot sequence has to be chosen to be a fraction of the number of scheduled subcarriers and must be at least the number of scheduled users in order to enable interference free MIMO channel estimation. As the pilot symbol overhead does not scale with the number of employed transmit antennas, the `'LTE Uplink'` pilot pattern should be used for simulations with a high number of transmit antennas (massive MIMO).

The estimated channel is linearly interpolated in between the assigned pilot positions at the receiver side. Currently a two- dimensional linear interpolation for the real and the imaginary channel coefficients.

15 Feedback

In wireless communications, Channel State Information at the Transmitter (CSIT) is needed to adapt the transmission to the current channel condition, in order to achieve a better performance. The receiver has to feedback the Channel State Information (CSI) to the transmitter. Limited feedback is employed to reduce the overhead. The receiver has to feed back the CQI, the Rank Indicator (RI) and the PMI. The CQI informs the transmitter about the MCS suitable for the current channel conditions. The RI informs the transmitter about the number of useful spatial streams, and the PMI represents the codebook index of the pre-coding matrix.

15.1 Feedback Calculation

The feedback calculation is based on [21, 22]. In this scheme the optimal PMI and RI are calculated jointly. Once a pre-coding matrix is determined, the optimal CQI is calculated.

15.1.1 PMI and RI

In the codebook-based PMI, the receiver performs an exhaustive search, to find the pre-coding matrix $\mathbf{W}_i \in \mathcal{W}$ which maximizes the sum mutual information $I_{k,n}$ over all resource elements (15.1). The mutual information calculation is based on the post-equalization **SINR** (15.2).

$$\mathbf{W}_i = \arg\max_{\mathbf{W}_j \in \mathcal{W}} \sum_{k=1}^{K} \sum_{n=1}^{N} I_{k,n}\left(\mathbf{W}_j\right) \tag{15.1}$$

$$I_{k,n} = \sum_{l=1}^{L} \log_2\left(1 + SINR_{k,n,l}\right) \tag{15.2}$$

15.1.2 CQI

The CQI calculation is based on Effective Signal to Interference and Noise Ratio Mapping (ESM). The post-equalization Signal to Interference and Noise Ratio (SINR) of all scheduled resource elements is mapped to an equivalent SNR value of a SISO AWGN channel, the CQI is then chosen, so that it has the highest value with FER$<$ 0.1 for the equivalent AWGN channel.

15.2 Object Usage

15.2.1 Scenario File

To run a simulation with feedback, the following lines have to be included in the scenario file:

```
1  % Feedback Parameters
2  scStr.feedback.delay          = 1;
3  % Feedback Delay
4  scStr.feedback.averager.Type  = 'miesm';
5  % 'eesm', 'miesm'
6
7  % for the custom transmission mode the following parameters
       are used to configure the feedback
8  scStr.feedback.enable         = true;
9
10 % when the feedback is enabled the following parameters are
       used to configure the individual indicators:
11 scStr.feedback.pmi            = true;
12 scStr.feedback.ri             = true;
13 scStr.feedback.cqi            = true;
```

In line 2 the feedback delay is set; a value of 1 equals a delay of 1 frame duration. A delay larger than 0, only makes sense when subsequent channels are temporarily correlated. The type of averager is chosen in line 4. Line 8 to 13 are relevant for the custom transmission mode, for the CLSM and OLSM the feedback parameters are set automatically and therefore these lines are irrelevant. To enable the feedback, line 8 is set to true. By setting line 11 to true, the PMI and the RI are activated (since the PMI and RI are calculated jointly). To activate the CQI feedback, line 13 has to be set to true. If the feedback is not enabled, line 11 to 13 are ignored.

15.2.2 Feedback Update

Fig. 13 shows how a transmission works. After the channel is generated, the feedback is calculated. If the delay is zero - which means that the channel has to be known before transmitting - the newly generated channel is used for feedback calculation. If the delay is larger than zero, the channel estimated at the receiver in the previous transmission is used for feedback calculation. The size of the delay buffer between the feedback calculation and the transmission corresponds to the delay.

The feedback calculation is done as follows:

```
1  obj.Feedback.updateFeedback(channel,pilotMatrix,...
```

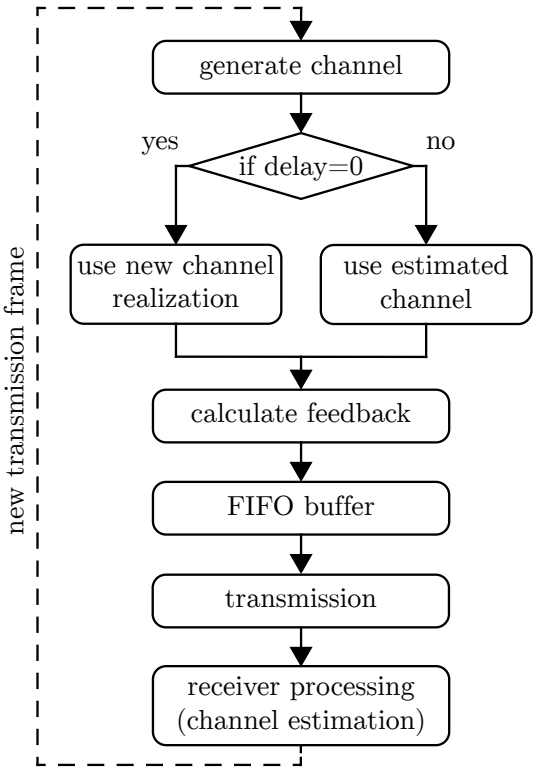


Figure 13: Feedback calculation flowchart. The delay of the feedback channel is implemented by means of a FIFO buffer.

```
2                               simParams.phy.noisePower,...
3                               modulationOrder);
```

the pilotMatrix determines the Resource Element (RE) which are used for the feedback calculation [3], `simParams.phy.noisePower` is the noise power, and the last input argument corresponds to the modulation order of all MCS defined in the CQI table. After the feedback is calculated, the feedback values can be accessed as follows:

```
1 PMI = obj.Feedback.Pmi.pmiArray(1);
2 RI  = obj.Feedback.Ri.riArray(1);
3 CQI = obj.Feedback.Cqi.cqiArray(1);
```

For the transmissions, where no feedback is available due to the feedback delay, the PMI, RI, and CQI are set to 1.

[3] Note that, in order to reduce the simulation time, the SINR is only calculated at the pilot positions.

16 Non-Orthogonal Multiple Access (NOMA)

NOMA is a key technology for next generation communications systems. It allows the users to access the available resources in a non-orthogonal manner, and in turn allowing the system to accommodate more users compared to the case of OMA. This would of course require employing advanced receivers in order to cope with the induced interference between those users. On top of that, many NOMA schemes fit naturally in the context of grant-free access, allowing the users to access the resources more often and therefore reducing the latency. These two aspects; massive connectivity and low latency operation, are main components of the future systems, and NOMA is able to tackle both of these issues in a natural manner.

In the current version of the simulator, we support the 3GPP MUST technique [15]. It is a downlink power-domain NOMA version that works by superimposing a maximum of two users on the same resources in the power-domain. The gain provided by this scheme is maximized when the two imposed users have a large difference in their channel quality, i.e., a user with good channel conditions (NearUE) and a user with bad channel condition (FarUE). A typical example for a FarUE would be a cell-edge user. In the simulator, the notion of strong and weak users can be controlled through `scStr.simulation.pathloss` parameter. The superposition works by assigning the FarUE with most of the transmit power. Then, at the receiving side, Successive Interference Cancelation (SIC) can be used to first detect the high power user, subtract its signal from the total received signal, and then proceed to detect the low power user. Alternatively, one can view the superimposed signal as just a normal signal with symbols being drawn from a super composite constellation. This in turns allow us to perform the detection using a Maximum Likelihood (ML) detector running on the composite constellation. In the standard, FarUE is limited to 4 QAM, while NearUE can use up to 64 QAM. This further simplifies the detection process for the FarUE, since it can treat the received superimposed signal as legacy 4 QAM and proceed to detect its signal as if NearUE is just an extra noise. On the other hand, this also means that the BS does not need to transmit control information to the FarUE about the MUST operation. Furthermore, the standard defines three power ratios that control the allocation of the power between the two superimposed users. This can be controlled using the `MUSTIdx` parameter [15].

To enable MUST transmission in the simulator, consider the following example. Assume there is only a single BS with two users UE1 and UE2. Enabling MUST proceeds as follows: first, indicate that MUST is the multiuser transmission mode for the BS

```
scStr.schedule.multiuserMode.Downlink    = {'MUST'};
```

Then, set the power-ratio for the superposition (e.g., the second power-ratio)

```
scStr.modulation.MUSTIdx          = [2];
```

Those two parameters are per-BS, and so you could run multiple BSs with different configurations. Finally, indicate the schedule of the users. Suppose you would like to have UE2 superimposed on UE1. They need to be scheduled in the following way

```
scStr.schedule.fixedScheduleDL{1}         = ['UE1:72,UE2:UE1'];
```

This way, UE1 will be the NearUE and UE2 is the FarUE.

When it comes to producing results with MUST, it is recommended to use the transmit power of the BS as the sweeping parameter. The path loss parameter can then be used to set the quality of the user channels, as shown in the simulation scenario of Section 5.2.5.

17 Releases and Changelog

3. Vienna 5G LL Simulator **1.2**, released May 2020.
   Newly introduced features:

   - support 1024QAM modulation with feedback
   - implementation of spatially and temporally correlated Spatial Channel Model
   - multi-user MIMO transmission modes with feedback
   - symbol domain orthogonal pilot symbols for multi-user MIMO channel estimation
   - TDD mode with feedback
   - support per-codeword feedback
   - support sum-throughput plotting

   Bug fixes and improvements:

   - bugfix in random number generator initialisation in parfor mode
   - bugfix in feedback parameter calculation for spatial multiplexing
   - bugfix in transmit diversity transmission
   - bugfix in result plots
   - bugfix in noise power calculation for custom feedback
   - bugfix in NOMA for users with different MIMO settings
   - bugfix in transmit power scaling for both downlink and uplink
   - bugfix in single-digit schedule assignment
   - bugfix in TWDP model power normalisation
   - more simulation parameters can be swept over
   - add MUSTIdx to the simulation parameters
   - PA nonlinear model extended to Uplink channels as well
   - Max-Log approximation is now the default LLR calculation method
     - add a fast piecewise implementation of the Max-Log method
     - LLR calculation method is now a per-link simulation parameter
   - improvement of channel coding
     - faster LDPC and polar encoding

  - CRC-based early stopping criterion for LDPC and turbo decoders
  - LDPC TBS is now standard compliant
  - bugfix an error when number of subcarriers is very large
  - CRC operations are now implemented in C++
  - precalculated polar transforms (.mat files) are removed
  - SoftBufferRatio is now a per-link simulation parameter
- speedup of time-varying channels generation via subsampling.
- speedup of f-OFDM implementation
- speedup of PAPR calculation
- remove many of the toolboxes' dependencies

2. Vienna 5G LL Simulator **1.1**, released June 2018.
   Newly introduced features:

   - introduce various input parameter checks for improved usability
   - introduce time correlated Rayleigh fading channel
   - implement spatial correlation for MIMO channels according to TS36.101 Annex B
   - implementation of NOMA (3GPP MUST)
   - implement feedback and transmission modes CLSM and OLSM
   - support for 256 QAM with feedback
   - TWDP fading model for static channels
   - non-linear power amplifier model
   - add PAPR (signal power CCDF) as simulation result
   - add channel estimation mean squared error as simulation result
   - new channel coding algorithms (additional decoding algorithms)

   Bug fixes and improvements:

   - make LLR value calculation numerically robust
   - bugfix in the transmitter signal generation
   - bugfix in the superposition of signals in the channel
   - bugfix in the automatic sampling rate calculation
   - bugfix in the parameter check for FBMC support

1. Vienna 5G LL Simulator **1.0**, released June 2017.

Acknowledgements

The financial support by the Austrian Federal Ministry of Science, Research and Economy, the National Foundation for Research, Technology and Development, and by TU Wien is gratefully acknowledged. This work has been co-financed by A1 Telekom Austria AG, Kathrein-Werke KG and Nokia Solutions and Networks.

References

[1] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Radio Access Network; NR; Physical channels and modulation*. TS 38.211. 3rd Generation Partnership Project (3GPP), Dec. 2017.

[2] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Radio Access Network; Study on channel model for frequencies from 0.5 to 100GHz*. TR 38.901. 3rd Generation Partnership Project (3GPP), Dec. 2017.

[3] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Radio Access Network; High Speed Downlink Packet Access: UE Radio Transmission and Reception*. TR 25.890. 3rd Generation Partnership Project (3GPP), May 2002.

[4] Technical Specification Group Radio Access Network. *Evolved Universal Terrestrial Radio Access: Base Station radio transmission and reception*. TR 36.104. 3rd Generation Partnership Project (3GPP), Dec. 2017.

[5] Erich Zöchmann, Sebastian Caban, Christoph F. Mecklenbräuker, Stefan Pratschner, Martin Lerch, Stefan Schwarz, and Markus Rupp. "Better than Rician: modelling millimetre wave channels as two-wave with diffuse power". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (Jan. 2019), p. 21. ISSN: 1687-1499. DOI: 10.1186/s13638-018-1336-6. URL: https://doi.org/10.1186/s13638-018-1336-6.

[6] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception*. TS 36.101. 3rd Generation Partnership Project (3GPP), Dec. 2017.

[7] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding*. TS 36.212. 3rd Generation Partnership Project (3GPP), Dec. 2017.

[8] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)" In: *IEEE Transactions on Information Theory* 20.2 (Mar. 1974), pp. 284–287. ISSN: 0018-9448. DOI: 10.1109/TIT.1974.1055186.

[9] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Radio Access Network; NR; Multiplexing and channel coding*. TS 38.212. 3rd Generation Partnership Project (3GPP), Dec. 2019.

[10] D. J. C. MacKay. “Good error-correcting codes based on very sparse matrices”. In: *IEEE Transactions on Information Theory* 45.2 (Mar. 1999), pp. 399–431. ISSN: 0018-9448. DOI: 10.1109/18.748992.

[11] P. Radosavljevic, A. de Baynast, and J. R. Cavallaro. “Optimized Message Passing Schedules for LDPC Decoding”. In: *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*. Oct. 2005, pp. 591–595. DOI: 10.1109/ACSSC.2005.1599818.

[12] Bashar Tahir and Markus Rupp. “New construction and performance analysis of Polar codes over AWGN channels”. In: *24th International Conference on Telecommunications (ICT)*. May 2017, pp. 1–4. DOI: 10.1109/ICT.2017.7998250.

[13] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg. “LLR-Based Successive Cancellation List Decoding of Polar Codes”. In: *IEEE Transactions on Signal Processing* 63.19 (2015), pp. 5165–5179.

[14] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. TS 36.213. 3rd Generation Partnership Project (3GPP), Jan. 2015.

[15] 3rd Generation Partnership Project (3GPP). *Evolved Universal Terrestrial Radio Access (E-UTRA) physical channels and modulation*. TS 36.211. 3rd Generation Partnership Project (3GPP), Jan. 2020.

[16] Qualcomm Incorporated. “Waveform Candidates”. In: *3GPP TSG-RAN WG1 84b*. Busan, Korea, Apr. 2016.

[17] “R1-165014, Subband-wise filtered OFDM for New Radio below 6 GHz”. In: *3GPP TSG-RAN WG1 85* (May 2016).

[18] Xi Zhang, Ming Jia, Lei Chen, Jianglei Ma, and Jing Qiu. “Filtered-OFDM-Enabler for Flexible Waveform in The 5th Generation Cellular Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2015, pp. 1–6.

[19] Technical Specification Group Radio Access Network3rd Generation Partnership Project (3GPP). *Study on New Radio Access Technology; Physical Layer Aspects*. TR. 3rd Generation Partnership Project (3GPP), Mar. 2017.

[20] Ronald Nissel, Stefan Schwarz, and Markus Rupp. “Filter bank multicarrier modulation schemes for future mobile communications”. In: *IEEE Journal on Selected Areas in Communications* 35.8 (2017), pp. 1768–1782.

[21] Stefan Schwarz, Christian Mehlführer, and Markus Rupp. “Calculation of the spatial preprocessing and link adaption feedback for 3GPP UMTS/LTE”. In: *6th conference on Wireless advanced (WiAD)*. IEEE. 2010, pp. 1–6.

[22] Stefan Schwarz, Martin Wrulich, and Markus Rupp. “Mutual information based calculation of the precoding matrix indicator for 3GPP UMTS/LTE”. In: *International ITG Workshop on Smart Antennas (WSA)*. IEEE. 2010, pp. 52–58.