January 2014

An experimental study of OFDM in software defined radio systems using GNU platform and USRP2 devices

Le Chung Tran
University of Wollongong, lctran@uow.edu.au

Duc Toan Nguyen
University of Wollongong, dtn348@uowmail.edu.au

Farzad Safaei
University of Wollongong, farzad@uow.edu.au

Peter J. Vial
University of Wollongong, peterv@uow.edu.au

An experimental study of OFDM in software defined radio systems using GNU platform and USRP2 devices

Abstract

Orthogonal Frequency Division Multiplexing (OFDM) has been well examined in the literature via theoretical simulations. However, implementation of OFDM in actual hardware using Software Defined Radio (SDR) concepts and verification of its performance with different channel estimation methods in various propagation environments have been almost unexplored. The great flexibility feature of SDR systems facilitates the implementation and experimentation of OFDM systems with less cost and effort, compared to the implementation of the whole system in hardware. In this paper, a customized SDR testbed has been developed based on the GNU radio software platform and version-2 Universal Software Radio Peripheral (USRP2) devices to evaluate the practical error performance of OFDM-based systems in both Gaussian and Rician propagation environments. Three different channel interpolation techniques, namely linear interpolation, second-ordered interpolation and cubic spline interpolation, and a blind SNR estimation algorithm have been implemented in our testbed. The performances show that, as opposed to our intuition, linear channel interpolation in some cases might not only be simpler, but also more accurate than the two other non-linear interpolation techniques, implying that channels might change linearly between neighboring subcarriers. The experimental OFDM system on the developed SDR testbed performs very close to the simulated OFDM system, thus the developed testbed can be used to verify advanced signal processing techniques in OFDM systems in various realistic channels by simply developing software, without the need for otherwise complicated hardware developments.

Keywords

platform, devices, gnu, usrp2, systems, radio, defined, software, ofdm, study, experimental

Publication Details

L. Chung. Tran, D. Toan. Nguyen, F. Safaei & P. James. Vial, "An experimental study of OFDM in software defined radio systems using GNU platform and USRP2 devices," in Advanced Technology for Communications (ACT 2014), 2014, pp. 657-662.

An Experimental Study of OFDM in Software Defined Radio Systems Using GNU Platform and USRP2 Devices

Le Chung Tran, Duc Toan Nguyen, Farzad Safaei, and Peter James Vial
School of Electrical, Computer, and Telecommunications Engineering
University of Wollongong, NSW 2522, Australia

***Abstract*— Orthogonal Frequency Division Multiplexing (OFDM) has been well examined in the literature via theoretical simulations. However, implementation of OFDM in actual hardware using Software Defined Radio (SDR) concepts and verification of its performance with different channel estimation methods in various propagation environments have been almost unexplored. The great flexibility feature of SDR systems facilitates the implementation and experimentation of OFDM systems with less cost and effort, compared to the implementation of the whole system in hardware. In this paper, a customized SDR testbed has been developed based on the GNU radio software platform and version-2 Universal Software Radio Peripheral (USRP2) devices to evaluate the practical error performance of OFDM-based systems in both Gaussian and Rician propagation environments. Three different channel interpolation techniques, namely linear interpolation, second-ordered interpolation and cubic spline interpolation, and a blind SNR estimation algorithm have been implemented in our testbed. The performances show that, as opposed to our intuition, linear channel interpolation in some cases might not only be simpler, but also more accurate than the two other non-linear interpolation techniques, implying that channels might change linearly between neighboring subcarriers. The experimental OFDM system on the developed SDR testbed performs very close to the simulated OFDM system, thus the developed testbed can be used to verify advanced signal processing techniques in OFDM systems in various realistic channels by simply developing software, without the need for otherwise complicated hardware developments.**

***Index Terms*— SDR, USRP2, GNU Radio, OFDM, channel estimation, SNR estimation**

I. Introduction

To enhance wireless communication performance numerous techniques have been proposed and validated by simulations in the literature. However, validation of these techniques in actual hardware is much more challenging because implementation of a system in hardware consumes large amounts of time and effort and requires a high cost. The Software Defined Radio (SDR) approach helps researchers reduce the time, effort and cost in implementing a wireless system. The main idea of SDR is to turn radio hardware problems into software problems [1]. Most signal processing techniques in wireless systems thus can be performed via software. This characteristic of SDR offers great flexibility for research and development in wireless communication because various newly-advanced techniques can be implemented and verified with the same hardware.

Several approaches have been mentioned in the literature in regard to the SDR research field, such as Labview software (e.g. [2]) and Matlab Simulink (e.g. [3]). However, open GNU Radio software platform (a free software toolkit for building SDR, written in the Python programming language) and open Universal Software Radio Peripheral (USRP) hardware are the most common low cost software and hardware used in SDR systems respectively. The USRP hardware developed by Ettus Research^{TM} has been originally designed for supporting GNU Radio, which is a software platform for run-time signal processing [4], [5]. With customized C++ codes, the USRP hardware can be easily modified by using the USRP hardware driver for different purposes.

The above advantage of SDR and USRP hardware motivates us to implement a wireless communication system based on SDR techniques using GNU Radio software platform and USRP hardware. OFDM-based systems have been somewhat implemented in the literature, such as [6]–[9]. These works evidenced that OFDM-based systems are possible to be implemented using SDR techniques. However, these works have not experimented the systems in different realistic propagation environments, and lacked the performance comparison of the system with different channel estimation and synchronization methods.

In this paper, an OFDM-based system with three different channel estimation techniques has been developed based on GNU Radio software platform and USRP2 hardware devices. The system performance is then analyzed with different channel estimation techniques in different realistic propagation channels. The paper is structured as follows. Section II briefly introduces GNU Radio software platform and USRP2 hardware. Section III overviews the system structure. Section IV presents an OFDM-based system associated with different modulation and channel estimation techniques. Section V details our experimental results and analyses. Section VI concludes the paper.

II. GNU Platform and USRP2

A. GNU Radio Software Platform

GNU Radio is an open software platform designed to build SDR systems. GNU Radio offers various building blocks for signal processing, as well as a method (called flowgraph) to manipulate the data flow between the blocks. Generally, GNU Radio blocks written in C++ are basic operation units that perform on continuous data streams. Each block includes a set of input and/or output ports. A block receives data from its input port and produces data for its output port. Special blocks called sources and sinks only have output ports and input ports respectively. Examples of sources/sinks are blocks that write/read, respectively, to/from USRP receiver/transmitter ports, sockets and files. Both input and output streams of a block have associated buffers. Each input/output data stream

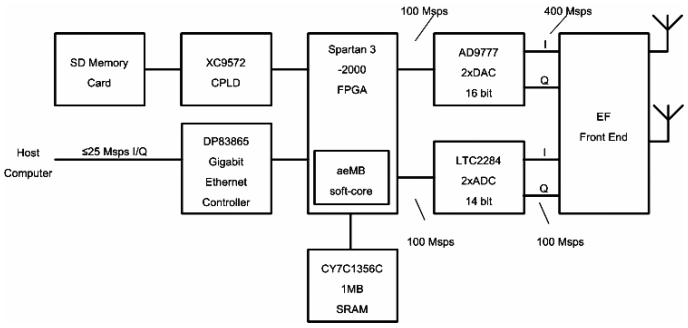

Fig. 1. Block diagram of USRP2, main components and signalling rates.

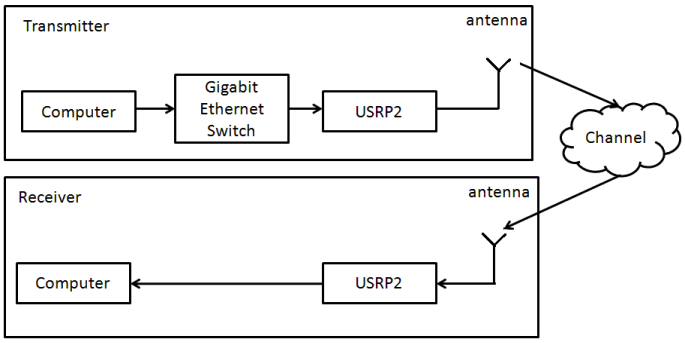

Fig. 2. Structure overview of the testbed.

has a read/write buffer. A block reads data from its read buffers for signal processing. After processing, the block writes data streams to the appropriate write buffers of its outputs. The data buffers are utilized to implement the edges in the flowgraph, i.e. the write buffers for one block are the read buffers of the upstream block. GNU Radio buffers are single writer and multiple reader First In First Out (FIFO) buffers, i.e. one output data stream can connect to one or more input data upstream(s) while one input on the received data stream is only from one output.

Data flows between the blocks are controlled by Python scripts. The use of Python scripts allows easy reconfiguration and manipulation of various functionalities and parameters of the system. Similarly to connecting physically RF building blocks to construct a hardware radio, users can build a SDR system by "wiring" together building blocks. This feature is called the virtual connect function, which indicates how the output data streams of a block connect to the input data stream of one or more upstream blocks. The flowgraph mechanism then automatically constructs the flowgraph. This process is hidden from users. The main function of the flowgraph mechanism is the allocation of data stream buffers to connect blocks. The mechanism takes into account the sizes of input and output data streams used by blocks and the associated data rate at which blocks consume input (or generate output) data streams. After buffers are allocated, they are linked to the input and output data streams of the appropriate blocks. To execute, GNU Radio provides a single thread scheduler to sequentially traverse all of the blocks in a flowgraph.

B. USRP2

USRP is the most common hardware used with GNU Radio to build a SDR system. USRP is a family of hardware devices found by Matt Ettus which facilitate making SDR systems. Each USRP device comprises two main sub-devices, a motherboard and one among various available daughterboards, which can transmit and/or receive signals in different frequency ranges from 0 to 5.9 GHz. The daughterboards can easily be exchanged. This allows USRP to work at various frequencies. Motherboard includes an onboard Field Programmable Gate Array (FPGA), an onboard memory (SDRAM) and a removable memory (SD Card). The primary objective of USRP motherboards is to convert analogue Intermediate Frequency (IF) signals to digital baseband signals and vice versa. Note that RF-IF conversion is done in daughterboards.

Along the transmit path, digital information is sent to the FPGA which is converted to an analogue signal by two high speed digital-to-analogue converters (DAC). Along the receive path, two high speed analogue-to-digital converters (ADC) are used in the motherboard to convert analogue signals to digital signals.

In this paper, we use the version-2 Ettus USRP (USRP2) devices which were built up on the success of the original USRP, offering better performance and increased flexibility. The USRP2 architecture and its features are shown in Fig. 1. It features a Gigabit Ethernet 25 Msps interface to communicate with PC, a Xilinx Spartan 32000 FPGA, a SDRAM 1MB, 2 ADCs at 100 Msps, 14 bits, 2 DACs at 400 Msps, 16 bits, and a Secure Digital (SD) card reader for loading the firmware. The internal USRP2 clock is a 100 MHz Voltage Controlled Oscillator (VCO) with a 10 ppm nominal accuracy. USRP2 also provides a connection to an external reference clock as well as a Pulse-Per-Second (PPS) timing port [6]. USRP2 provides connections to daughter boards which serve as RF front-ends. The RF board used in our testbed is the wideband daughter board WBX, working in a wide range from 50 MHz to 2.2 GHz with the typical transmitted power range of 30-100 mW.

III. System Model

A. System Overview

The schematic diagram of our testbed is illustrated in Fig. 2. The transmitter includes a laptop running the Ubuntu 10.4 operating system, which has been installed via VMware Player; one Gigabit Ethernet switch to synchronise the 100 Mbps port on the laptop and the Gigabit port on USRP2; a USRP2 module with one WBX daughter board; and one omni-directional (whip) antenna operating at 300 MHz. The Gigabit Ethernet switch is used in the transmitter because our PC in the transmitter does not support Gigabit transmission. It facilitates the PC to communicate with the Gigabit Ethernet-featured USRP2 module. The receiver comprises a Gigabit Ethernet-supported PC, running the Ubuntu 10.4 operating system; a USRP2 module with one WBX daughter board; and one whip antenna working at 300 MHz.

Similarly to any other hardware connected via an Ethernet port, each of the USRP2 devices has an IP address which is by default pre-configured to 192.168.10.2/24. The GNU Radio script running on the PCs will use this IP address to identify the connected USRP2 device. Preferably, the Gigabit Ethernet interface of the PC should have some IP address in the same subnet, i.e. 192.168.10.x/24, to avoid the necessity of customized entries in the routing table of the laptop. In this project, the IP addresses of the PCs in both transmitter and receiver are set to 192.168.10.1/24.

B. Baseband Processing At OFDM Transmitter

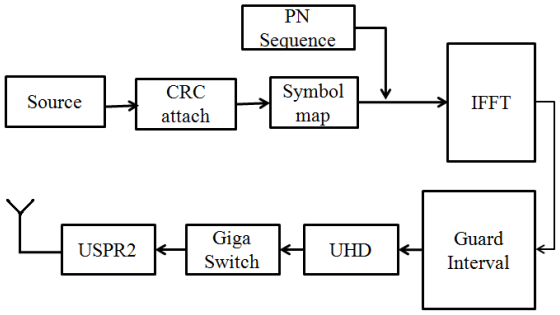

Fig. 3. Block diagram of OFDM transmitter.

Block diagram of the OFDM transmitter in our testbed is presented in Fig. 3. A Cyclic Redundancy Code (CRC) is attached to the vector of information bits. The bit stream is separated into groups of M bits and each group is then fed to the modulator, which is then mapped into a corresponding complex symbol. The stream of modulated symbols is divided into groups of N_d symbols. The N-point Inverse Fast Fourier Transform (IFFT) operation is applied to each group to obtain time-domain OFDM symbols. To reduce Inter-Symbol Inference (ISI) between successive blocks of data, the Cyclic Prefix (CP) of N_g samples will be implemented in each block. The USRP Hardware Driver (UHD) takes care of sending the generated vector of time samples to hardware.

The structure of an OFDM symbol with its preamble is presented in Fig. 4. The preamble includes known modulated symbols referred to as Pseudo-random Noise (PN) sequence. The preamble allows the receiver to obtain the beginning of each frame, channel state information, as well as facilitates the estimation of the Signal-to-Noise Ratio (SNR). The preamble is generated off-line and is known by both transmitter and receiver.

C. Baseband Processing At OFDM Receiver

Block diagram of the OFDM receiver is depicted in Fig. 5. Signal samples are received from USRP2 hardware via UHD, and then fed to a simple Fourier filter. The filtered signal is fed to the synchronization block, which is implemented in the Python file `ofdm_sync_pn.py`. In our testbed, the PN synchronization algorithm proposed in [10] is used. In particular, the OFDM synchronization block includes two processes. The first process is time synchronization to find the start of each frame by observing the signal energy. The second process is to find the fine frequency offset of the frame (which is in fact smaller than the subcarrier spacing). The frequency offset is fed to the frequency modulator to generate a signal proportional to the frequency error of the synchronization block. The signal from the channel filter block is mixed with the signal from the frequency modulator to rotate the received signals back, i.e. to correct the frequency error.

The synchronized signal in the form of vectors of size N each goes to the FFT module, that transfers the received signal into the frequency domain. The output signal includes data subcarriers. Each subcarrier contains information in its phase according to the digital modulation applied to that subcarrier. In our experiments, all data subcarriers are modulated with either BPSK or QPSK modulation. The

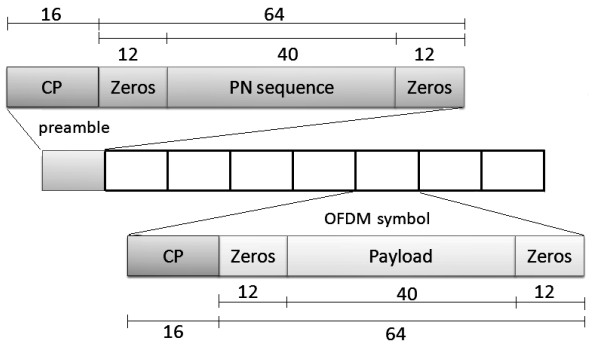

Fig. 4. OFDM frame structure.

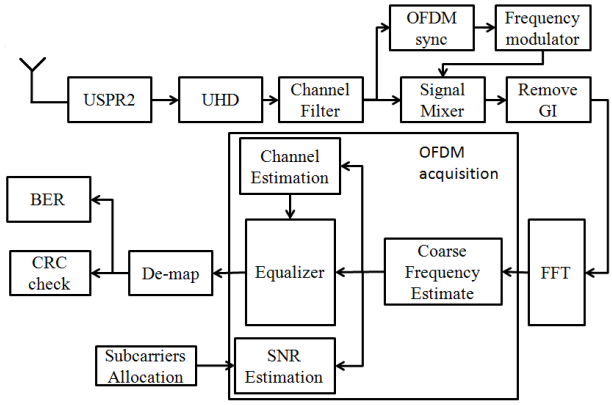

Fig. 5. Block diagram of OFDM receiver.

frame acquisition block in Fig. 5 takes care of finding the coarse carrier frequency offset, equalizing each data subcarrier and estimating SNR. This block is defined in the C++ file `digital_ofdm_frame_acquisition.cc` in GNU Radio.

In the frame acquisition block, the *carrier frequency offset*, which is the frequency error in multiples of the subcarrier spacing, can be easily found after the FFT process by observing the energy of the FFT-ed signals based on the algorithm in [10]. Outputs of this block are vectors of N_d symbols each (unused subcarriers have been removed).

In our testbed, the preamble-based channel estimation algorithm with the Least-Square (LS) approach in the frequency domain proposed by Beek [13] for OFDM systems has been modified in order to estimate the channel coefficient for each subcarrier. Specifically, our modified algorithm uses a linear interpolation, a second-order polynomial interpolation, and a cubic spline interpolation in the frequency domain, rather than just linear interpolation as in [13], to estimate channel coefficients at odd subcarriers (no pilot symbols are transmitted at the odd subcarriers of the preamble). We also use the blind SNR estimation approach proposed in [14] to calculate SNR in order to evaluate the OFDM system performance in different realistic propagation channels. Details of the channel estimation and SNR estimation are presented in the next subsections.

D. Channel Estimation in OFDM Systems

In this subsection, the preamble-based channel estimation algorithm used in our system is presented. It is supposed that channel state information remains static over each frame duration. Therefore, the estimated channel response obtained

from the preamble can be applied to detect coherently the whole frame.

As shown in Fig. 4, an OFDM frame consists totally of 8 OFDM symbols (1 preamble and 7 OFDM data symbols). Each OFDM symbol consists of 64 samples (24 zeros samples for synchronization and 40 data samples) and 16 samples of the guard interval. The preamble symbol has the same structure as other OFDM symbols. The only difference is that all of subcarriers in the preamble are the known Pseudo Noise (PN) sequence used for channel estimation, while these subcarriers in other OFDM symbols are used for data transmission. The PN sequence only consists of ± 1 in the even subcarriers and 0 in the odd subcarriers. Every OFDM frame has one preamble symbol. The received preamble samples at the output of the FFT block can be written as

$$R(k) = C(k) \times H(k) + N(k) \quad k = 0, .., N-1 \qquad (1)$$

where $C(k)$ are PN symbols within the preamble, which are known to the receiver, $H(k)$ are the channel frequency responses, and $N(k)$ are noise samples. The LS estimate of the channel frequency response at the even subcarrires of the preamble, denoted as H_{LS}, can be obtained as follows

$$H_{LS}(2k) = R(2k)/C(2k) \quad k = 0, .., N/2-1 \qquad (2)$$

Because the PN sequence only consists of ± 1 at the even subcarriers and 0 at the odd subcarriers, the channel frequency responses for the odd subcarriers must be interpolated. A linear interpolation, a second-order polynomial interpolation, and a cubic spline interpolation are used in our testbed to estimate the channel frequency responses at the odd subcarriers. For example, the linear interpolation is given as follows [11, Eq.(18)]

$$\begin{aligned} H_{LS}(2k+1) &= \big[H_{LS}(2k) + H_{LS}(2k+2)\big]/2 \qquad (3) \\ H_{LS}(N-1) &\approx H_{LS}(N-2) \qquad (4) \end{aligned}$$

for $k = 0, .., N/2-2$. Second-order interpolation and cubic spline interpolation can be found in [11, Eq.(19)] and [12] respectively.

E. SNR Estimation in OFDM Systems

One of the main differences between simulating and experimenting the OFDM performance is that SNR at the receiver is known in simulations, while it is unknown in the practical experiments. Hence SNR must be estimated in a practical system. SNR is defined as the ratio of the desired signal power to the background noise power. Most SNR estimation algorithms are performed in the frequency domain. The received OFDM signal after FFT at the j-th OFDM symbol and k-th subcarrier is presented by

$$R_j(k) = C_j(k) \times H_j(k) + N_j(k) \qquad (5)$$

During the preamble interval, $C_j(k)$ and $H_j(k)$ are known. Therefore, the instantaneous SNR ψ within the interval J and the set of subcarriers K could be estimated as

$$\begin{aligned} \hat{\psi} &= \frac{\sum_{j\in J}\sum_{k\in K} |C_j(k) \times \hat{H}_j(k)|^2}{\sum_{j\in J}\sum_{k\in K} |N_j(k)|^2} \\ &= \frac{\sum_{j\in J}\sum_{k\in K} |C_j(k) \times \hat{H}_j(k)|^2}{\sum_{j\in J}\sum_{k\in K} |R_j(k) - C_j(k) \times \hat{H}_j(k)|^2} \qquad (6) \end{aligned}$$

TABLE I
MAIN EXPERIMENTAL SETTINGS.

Parameters	Value
Carrier frequency	300 MHz
FFT/IFFT size N	64
Occupied tones N_d	40
Cyclic Prefix N_g	16
Baseband sampling rate	500kps
Number of OFDM symbols per frame	8
TX gain	15 dB
RX gain	15 dB
Modulation	BPSK

where $\hat{H}_j(k)$ is the estimated channel frequency response. Thus the accuracy of the SNR estimation depends on the accuracy of the channel estimation.

Eq. (6) can be used to estimate SNR in various scenarios. However, it cannot be used if a LS channel estimation method is used to estimate the channel coefficients (as in our testbed). It is because, in the LS channel estimation, $\hat{H}_j(k)$ is estimated as

$$\hat{H}_j(k) = \frac{R_j(k)}{C_j(k)} \qquad (7)$$

From (6) and (7), it is clear that with the LS channel estimation in place, (6) cannot be used for SNR estimation in the preamble since the denominator of (6) is canceled, leading to the infinite estimated SNR $\hat{\psi} = \infty$.

Therefore, we decide to use the blind SNR estimation algorithm proposed in [14], where we do not need to know $\hat{H}_j$ to estimate SNR. If Q is defined to be the set of subcarriers in an OFDM symbol where no signal is transmitted, the frequency domain samples for the subcarriers in Q at the receiver are primarily the noise samples

$$R_j(k) = N_j(k) \quad \text{for } \forall k \in Q \qquad (8)$$

The Noise-to-Noise ratio (NNR) between noise power on the null subcarriers (denoted as Set Q) and noise power on the data subcarriers (denoted as Set K) is defined as follows

$$\eta = \frac{\sum_{j\in J}\sum_{q\in Q} |N_j(q)|^2}{\sum_{j\in J}\sum_{k\in K} |N_j(k)|^2} \qquad (9)$$

If the noise is white noise, such as AWGN, the expected value of NNR approaches a constant value

$$\bar{\eta} = \frac{\|Q\|}{\|K\|} \qquad (10)$$

where $\|(.)\|$ denotes the number of subcarriers in the corresponding set. The estimated SNR $\hat{\psi}$ can be derived as below

$$\begin{aligned} \hat{\psi} &= \frac{\sum_{j\in J}\sum_{k\in K} |R_j(k)|^2}{\sum_{j\in J}\sum_{q\in Q} |R_j(q)|^2} \times \bar{\eta} - 1 \\ &= \frac{\sum_{j\in J}\sum_{k\in K} |R_j(k)|^2}{\sum_{j\in J}\sum_{q\in Q} |R_j(q)|^2} \times \frac{\|Q\|}{\|K\|} - 1 \qquad (11) \end{aligned}$$

All terms in Eq. (11) are known, thus the SNR can be estimated.

IV. OFDM PERFORMANCE EVALUATION

This section evaluates the performance of our OFDM-based SDR system with three interpolation schemes for channel estimation, namely linear interpolation, second-order polynomial interpolation, and cubic spline interpolation in both

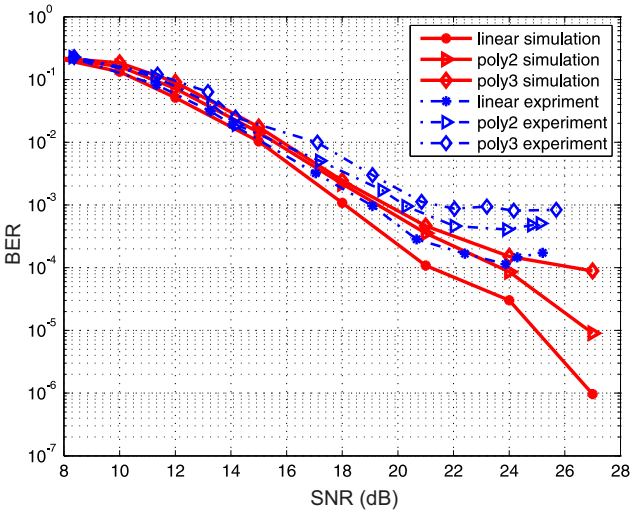

Fig. 6. Comparison of simulated OFDM performances and experimental OFDM performances in an AWGN channel.

Gaussian and Rician realistic propagation channels. Important setting parameters are listed in Table I. The experimental performances of the OFDM testbed are also compared to the simulated OFDM performances in GNU Radio. Parameters of both simulated OFDM system and experimental OFDM SDR system are chosen to be the same for a fair comparison. The estimated channel coefficients received at the output of the channel estimation block (cf. Fig. 5) will be used as the the channel impulse responses in our simulations.

A. OFDM system in AWGN channel

In this subsection, the OFDM SDR system is evaluated in an AWGN channel. The experimental AWGN channel is created by connecting two USRP2 devices via an optical cable, while the theoretical AWGN channel used in our simulation is generated by the block `gr.channel_model` in GNU Radio. The performances of the simulated OFDM system and the implemented OFDM system are illustrated in Fig. 6. Readers may notice that the BER performance of OFDM systems is only obtained for the SNR higher than 8 dB. This is due to the fact that the synchronization blocks (cf. Fig. 5) failed to perform accurately at lower SNRs because no external reference clock, such as GPS signals, was used in our experiments. The BER performance of the experimental OFDM system starts to experience an error floor at SNR higher than 23 dB because of the power saturation of the receiver.

The figure shows the relatively good agreement between the simulated performance and the experimental one, especially at lower SNRs. Interestingly, as opposed to one's intuition, the linear interpolation scheme is better than the second-order polynomial interpolation and cubic spline interpolation in both simulation and experimental cases. This means that estimating the channel response at a subcarrier based on a linear polynomial over its two neighbouring subcarriers is more accurate than estimating it based on non-linear functions over several subcarriers. This reflects that the propagation channels between the stationary transmitter and the stationary receiver in our experiments more likely experience a linear variation between neighbouring subcarriers.

Fig. 7. Laboratory environment is used in experiments.

B. OFDM system in Rician channel

This subsection presents the simulated performance of OFDM systems, and the experimental performance of our OFDM SDR testbed in our laboratory room with no obstruction between the transmit and receive antennas, which can be considered as a Rician wireless propagation environment. The distance between the two antennas is 5 meters. The laboratory environment is depicted in Fig. 7.

To model the multipath effects of a Rician channel, a Finite Impulse Response (FIR) filter provided in the `gr.fir_filter_ccc` block in GNU Radio is added in front of the `gr.channel_model` block. Once the FIR filter has been added, its filter coefficients need to be set up. In our simulations, similarly to the case of AWGN channels, the channel coefficients estimated by the channel estimation block in Fig. 5 will be analyzed to predict the realistic multipath propagation channel between the transmit and receive antennas, i.e. to find the FIR filter coefficients. By analyzing the file of channel coefficients collected from the channel estimation block for many transmission trials, we realized that the estimated average number of paths in our laboratory room is approximate 3 and the estimated powers of these three multipaths are $[0.7448 \;\; 0.1866 \;\; 0.0671]$.

Fig. 8 shows the experimental performance of our OFDM SDR testbed in the Rician channel in comparison with the simulated performance as a benchmark. It is important to recall that, due to the limitation of our hardware, the experimental BER performance can only be obtained for SNR higher than 8 dB, and that the experimental BER performance starts to experience the error floor at SNR = 23 dB onwards because the receiver is saturated. Therefore, we just focus our analysis on the SNR range from 8 to 23 dB. Fig. 8 shows that the experimental OFDM SDR system performances are very close to the simulated ones. In addition, the performance of OFDM systems with the linear channel interpolation is slightly better than those with the second-ordered interpolation and spline cubic interpolation, but the difference between the three interpolation schemes is much negligible compared to the AWGN case.

Note that a multipath Rayleigh fading channel is a special case of the Rician fading channel, where the direct path

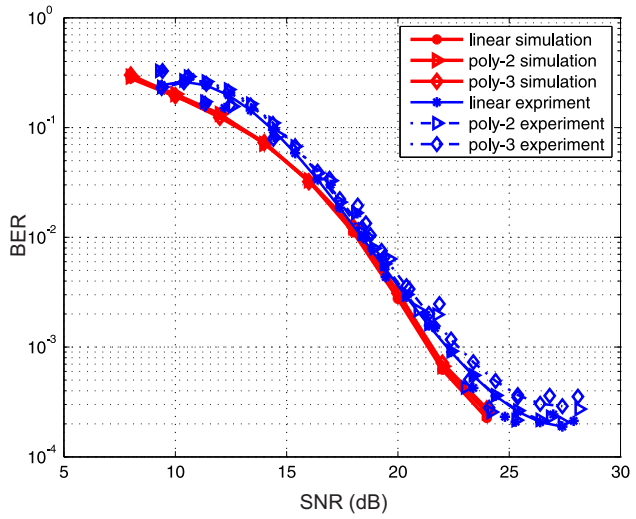

Fig. 8. Comparison of simulated OFDM performances and experimental OFDM performances in a Rician propagation channel.

between the transmit and receive antennas is blocked by an obstruction. Thus comparison between the simulated and the experimented performances can be done in a similar way for a Rayleigh channel.

From Figs. 6 and 8, an interesting observation can be drawn is that linear channel interpolation is not only simpler, but also better than two other non-linear interpolation schemes in the two considered scenarios, where both transmitter and receiver stay stationary. It is our conjecture that the second-order interpolation and spline cubic interpolation might have their advantages if the receiver moves quickly relatively to the transmitter. Validation of this conjecture in realistic propagation environments would be our future works. In addition, while these figures indicates a relatively good agreement between the simulated performance and the implemented one, they also presents slight differences between them. These differences exist due to the fact that SNR values are exactly known in the former, while they are estimated in the latter, and that channel impulse responses are hypothetically assumed to be constant during one OFDM frame in the former, while they may in fact vary over time in the latter.

V. Conclusions

In this paper, we have presented a customized OFDM-SDR testbed using GNU Radio software platform and USRP2 devices. Both baseband signal processing and hardware implementation are detailed. The customized LS channel estimation with the linear interpolation, second-order polynomial interpolation and cubic spline interpolation is experimented in the testbed. BER performances of the developed testbed have been examined in both Gaussian and Rician propagation environments in comparison with the corresponding performances simulated in GNU Radio, which are used as benchmarks. Our results show that, in case transmitter and receiver are stationary, LS channel estimation with the linear channel interpolation might not only be simpler, but also better than that with the two other non-linear interpolations in both AWGN channel and Rician channels. The results also show that there is a good agreement between the simulated and the experimented system performance. This confirms the validity of our testbed. Therefore, other advanced signal processing techniques, such as channel coding, OFDM block spreading, multiple antennas (or Multiple Input Multiple Output - MIMO), space-time coding [15], [16], [17] and space-time-frequency coding [18], [19], [20], can be applied to improve further the BER performance of the developed OFDM-SDR system. These tasks would be our future works.

References

[1] X. Li, W. Hu, H. Yousefi'zadeh, and A. Qureshi, "A case study of a MIMO SDR implementation," *Proc. IEEE Military Communications Conference*, pp. 1–7, Nov. 2008.

[2] "Building an Affordable 8x8 MIMO Testbed with NI USRP," Available: `http://www.ni.com/white-paper/14311/en/`, Accessed: 25 Aug 2014.

[3] M. Braun, M. Muller, M. Fuhr, and F. K. Jondral, "A USRP-based testbed for OFDM-based radar and communication systems," *Proc. 22nd Virginia Tech. Symp. Wireless Commun.*, USA, Jun. 2012.

[4] "GNU Radio," Available: `http://gnuradio.org/redmine/projects/gnuradio/wiki`, Accessed: 30 Sept 2013.

[5] "USRP hardware driver," Available: `http://ettus-apps.sourcerepo.com/redmine/ettus/projects/uhd/wiki`, Accessed: 29 Sept 2013.

[6] G. Berardinelli, et al., "An SDR architecture for OFDM transmission over USRP2 boards," *Conference Record of the 45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pp. 965–969, 2011.

[7] A. Marwanto, et al., "Experimental study of OFDM implementation utilizing GNU Radio and USRP-SDR," *Proc. IEEE 9th Malaysia International Conference on Communications (MICC)*, pp. 132–135, Dec. 2009.

[8] R. K. Ganti, et al., "Implementation and experimental results of superposition coding on software radio," *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–5, May 2010.

[9] M. Tichy and K. Ulovec, "OFDM system implementation using a USRP unit for testing purposes," *Proc. 22nd International Conference Radioelektronika (RADIOELEKTRONIKA)*, pp. 1–4, April 2012.

[10] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, Dec. 1997.

[11] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "Channel Estimation Techniques Based on Pilot Arrangement in OFDM Systems," *IEEE Transactions on Broadcasting*, vol. 48, no. 3, pp. 223–229, Sept. 2002.

[12] C. de Boor, *A practical guide to splines*, vol 27, Springer, New York, USA, 2001.

[13] J.-J. van de Beek, et al., "On channel estimation in OFDM systems," *Proc. 45th IEEE Vehicular Technology Conference*, vol. 2, pp. 815–819, July 1995.

[14] Y. Li, "Blind SNR estimation of OFDM signals," *Proc. International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, pp. 1792–1796, May 2010.

[15] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Select. Areas Commun.*, vol. 16, no. 8, pp. 1451 – 1458, Oct. 1998.

[16] L. C. Tran, T. A. Wysocki, A. Mertins, and J. Seberry, *Complex Orthogonal Space-Time Processing in Wireless Communications*, Springer, New York, USA, 2006.

[17] L. C. Tran, et al., "Novel Constructions of Improved Square Complex Orthogonal Designs for Eight Transmit Antennas," *IEEE Trans. Inform. Theory*, vol. 55, no. 10, pp. 4439-4448, Oct. 2009.

[18] L. C. Tran, and A. Mertins, "Space-time-frequency code implementation in MB-OFDM UWB communications: design criteria and performance," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 701–713, Feb. 2009.

[19] L. C. Tran, and A. Mertins, "Unitary Differential Space-Time-Frequency Codes for MB-OFDM UWB Wireless Communication," *IEEE Trans. Wireless Commun.*, vol. 12, no. 2, pp. 862-876, Feb. 2013.

[20] L. C. Tran, A. Mertins, and T. A. Wysocki, "Quasi-orthogonal space-time-frequency codes in MB-OFDM UWB," *Elsevier Journal on Computers and Electrical Engineering*, vol. 36, no. 4, pp. 766–774, July 2010.