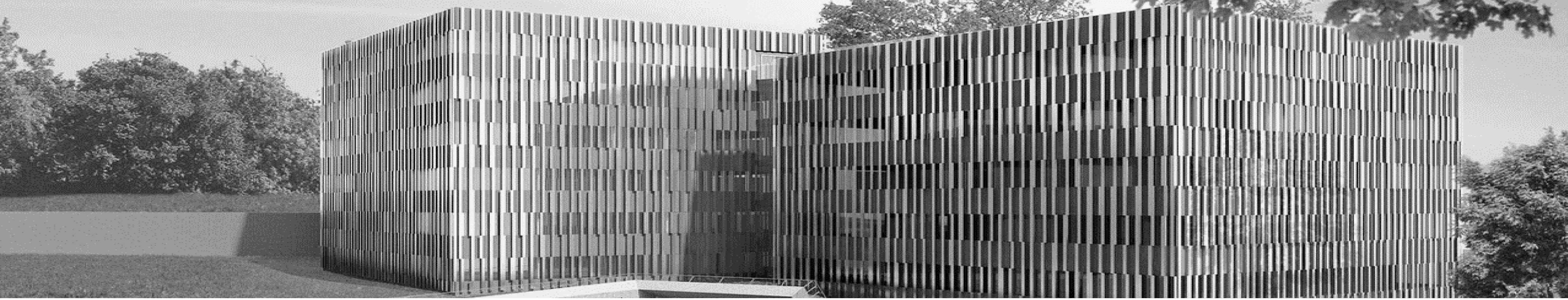


Signal Processing for Mobile Communications – Extended Transceiver Design

Programming Exercise 6: **Time and Frequency Synchronization in OFDM**

DSP Chair for Distributed Signal Processing
RWTH AACHEN UNIVERSITY

About the programming exercise

- The general simulator structure (`main` files) is provided.

- We will discuss the new functionalities you have to implement, and function interfaces are provided.

- You will build on top of what you have already implemented in previous exercises:
  - Add the new functions' interfaces to `OFDM.h`;
  - Add their implementations to `Transmitter.c`, `Receiver.c`

- You should analyze the provided simulator structure and understand the signal processing chains at the transmitter and the receiver.

Tasks

1. Implement **Time and Frequency Estimation** algorithm:
   - Use autocorrelation by Cyclic Prefix for timing offset estimation;
   - Compensate the timing offset;
   - Use autocorrelation and cross-correlation to estimate fractional and integer frequency offset;
   - Compensate fractional and integer frequency offset.

2. Evaluate system performance for both approaches
   - Plot BER versus energy per bit E_b/N_0;
   - Assume **perfect channel knowledge** at both the transmitter and the receiver;
   - Consider flat fading;
   - Compare system performance (BER) against the perfect case (No timing and frequency offset).

M-sequence

- Proposed in 5G NR standard for the Primary Synchronization Sequence.
- Generated by using a BPSK modulated sequence of length 127.
- Three different *m*-sequences are possible. (Cell ID)
- The sequence $d_{PSS}(n)$:

  - $d_{PSS}(n) = 1 - 2x(m)$
  - $m = (n + 43{N_{ID}}^{(2)}) \text{ mod} 127$
  - $0 \leq n < 127$

  - Where $x(i+7) = \big(x(i+4) + x(i)\big) mod2$
  - Initial state: $[x(6)\ x(5)\ x(4)\ x(3)\ x(2)\ x(1)\ x(0)] = [1\ 1\ 1\ 0\ 1\ 1\ 0]$

Time Synchronization (Autocorrelation)

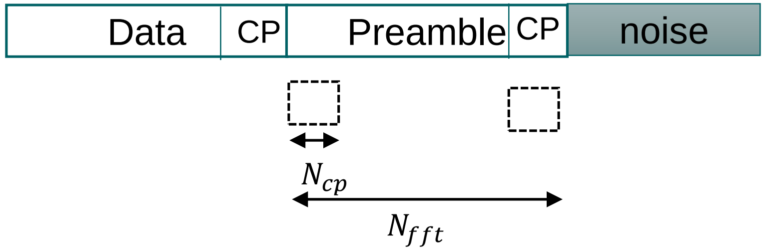
- For timing offset estimation, autocorrelation based on CP is suggested :

$$R_{rr}[l] = \sum_{i=0}^{N_{cp}-1} r[l+i]r^*[l+i+N_{fft}]$$

$$T[l] = \frac{|R_{rr}[l]|^2}{\sum_{k=0}^{N_{cp}-1} \left|r[l+k+N_{fft}]\right|^2}$$

- Timing offset

$$\hat{l} = \underset{l}{\operatorname{argmax}}\{T[l]\}$$



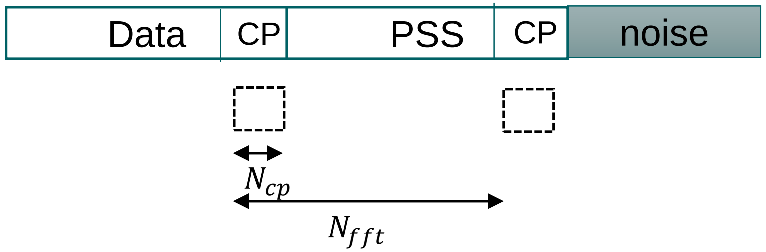
Frequency Offset Estimation – Fractional part

- Frequency Estimation includes two part: Fractional frequency estimate and Integer frequency estimate:

$$\Delta f_o = \Delta f_I + \Delta f_\varepsilon$$

- For fractional frequency offset estimation, autocorrelation based on CP is suggested:

- Fractional frequency offset $\Delta\hat{f}_\varepsilon = -\frac{1}{2\pi}\angle R_{rr}\,[\hat{l}]$



Frequency Offset Estimation – Integer part

- Integer frequency estimation:

$$\Delta\hat{f}_I = \underset{l}{\operatorname{argmax}} \left| \sum_{i=0}^{N_{fft}-1} R[i+l]P^*[i] \right|$$

- $R[i]$ is the i-th subcarrier of FFT of received signal and $P[i]$ is the i-th subcarrier of preamble in frequency domain.
- Cross-Correlation is applied to estimate integer frequency offset.

M-sequence(Transmitter.c)

- Implement a function that generates the m-sequence as described before, it will be used as a preamble in transmitting slots.

```
void m_seq(double preambleI[], double preambleQ[], int len, int Nid_2)
```

I-component of preamble array

Q-component of preamble array

preamble array length

Cell ID value

Time Offset Estimation(Receiver.c)

- Implement a function that estimates timing offset as described before, that estimates timing offset based on the autocorrelation of cyclic prefix part and the signal.

```
int timing_synch(double inI[], double inQ[], int lenIN)
```

Timing offset value

I-component of received signal

Q-component of received signal

Length of input array

```
void auto_corr(double inI[], double inQ[], int lenIN, double outI[], double outQ[], double
normalizedValue[])
```

Normalized Energy of output

I-component of output array

Q-component of output array

```
int find_max_index(double inputArray[], int len)
```

maximum value index

Input array

Input array length

Fractional Frequency Estimation(Receiver.c)

- Implement a function that estimates the fractional frequency offset based on the autocorrelation as described before.

```
double carrier_freq_estimation_fractional(double inI[], double inQ[], int lenIN)
```

Estimated fractional frequency

```
void auto_corr(double inI[], double inQ[], int lenIN, double outI[], double outQ[], double normalizedValue[])
```

Integer Frequency Estimation(Receiver.c)

- Implement a function that estimates the integer frequency offset based on the cross-correlation as described before.

```
int carrier_freq_estimation_integer(double rxSymI[], double rxSymQ[], int seqLen,
int syncSigStartIndex, int Nid_2)
```

Estimated integer frequency

First subcarrier to put m-sequence

Second cell ID value

I-component of received signal

Q-component of received signal

m-sequence length

I-component of first signal

Q-component of first signal

First signal length

I-component of second signal

Q-component of second signal

```
void cross_corr(double* hI, double* hQ, int lenH, double* inI, double* inQ,
      int lenIN, double* outI, double* outQ, int lenOUT)
```

Second signal length

I-component of cross-correlation output

Q-component of cross-correlation output

Cross-correlation output length

```
int find_peak_index(double* inI, double* inQ, int lenIN)
```

maximum value index

I-component of input array

Q-component of input array

Input array length

Submission

- Submit the report before 13:00 on 27.06.2022.

- Attach your code.

- Write a few meaningful sentences about the obtained results.
  - How does BER with imperfections behave compared to the perfect case?
  - How the timing and frequency estimation algorithms are working?