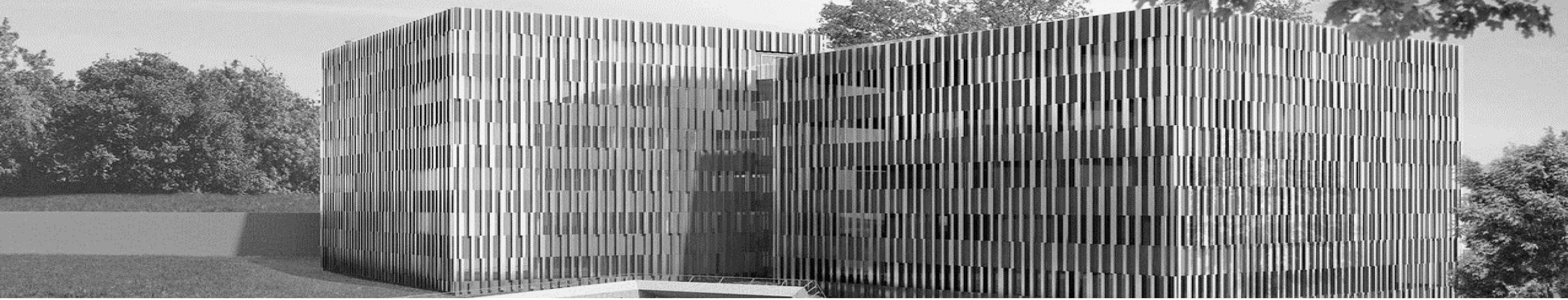


Signal Processing for Mobile Communications – Extended Transceiver Design

Programming Exercise 5: **Power and Rate Adaptation 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` , `Channel.c`

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

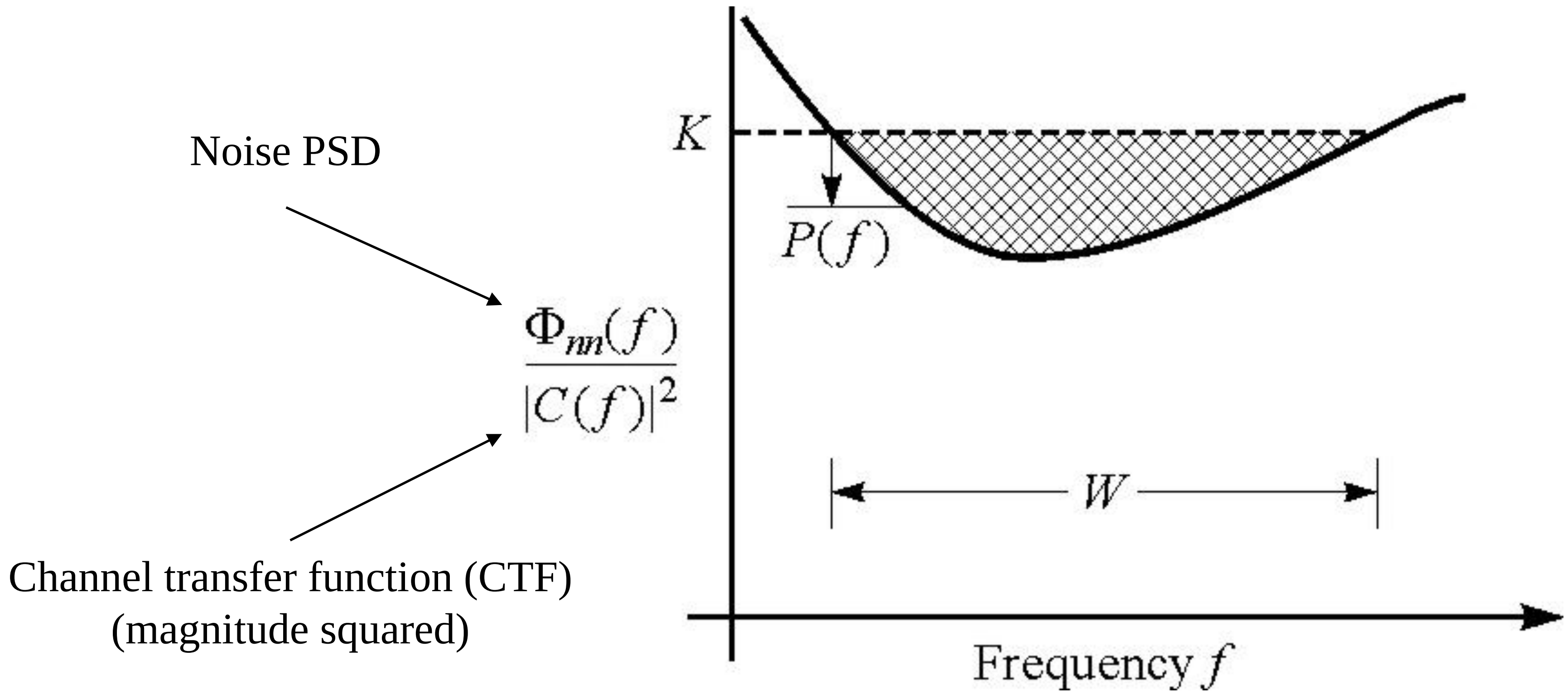
Tasks

1. Implement **Power and rate allocation** algorithm:
   - Use simple **Waterfilling** for power allocation;
   - Choose the highest modulation order that satisfies Bit Error Rate (BER) requirements, among:
     - BPSK, QPSK, 16QAM, 64QAM and 256QAM

3. Evaluate system performance for both approaches
   - Plot BER versus energy per bit E_b/N_0;
   - Plot average throughput (in bits per block / OFDM symbol) versus energy per bit E_b/N_0;
   - Assume **perfect channel knowledge** at both the transmitter and the receiver;
   - Consider frequency-selective fading (e.g., *8 channel taps)*;
   - Compare system performance (BER and throughput) against the non-adaptive case.

Power adaptation: Waterfilling

- Available power is allocated to subcarriers according to signal to noise ratio (SNR), where the subcarriers with higher SNR get more power.
- Subcarriers with SNR below a certain threshold are deactivated.



Waterfilling algorithm

- The power allocation is obtained as the solution of the following optimization problem:

$$\underset{P_0 \,\dots\, P_{N-1}}{\text{maximize}} \ \sum_{k=0}^{N-1} \log_2\left(1 + |H_k|^2 \frac{P_k}{P_N}\right)$$

subject to

$$\sum_{k=0}^{N-1} P_k = P_{tot}$$

$$P_k \geq 0, \quad \forall k$$

N - number of subcarriers;
P_k - power on k-th subcarrier;
H_k - channel transfer function;
P_N - noise power;
P_{tot} - total available power;

C_k - SNR factor;
C_{th} - SNR threshold which needs to be exceeded for the subcarrier to be used.

- Solution:

$$P_k = \max\left(\frac{1}{C_{th}} - \frac{1}{C_k}, 0\right)$$

$$\frac{1}{C_{th}} = \frac{P_{tot} + \sum_{k=0}^{N-1} C_k^{-1}}{N} \qquad C_k = \frac{|H_k|^2}{P_N}$$

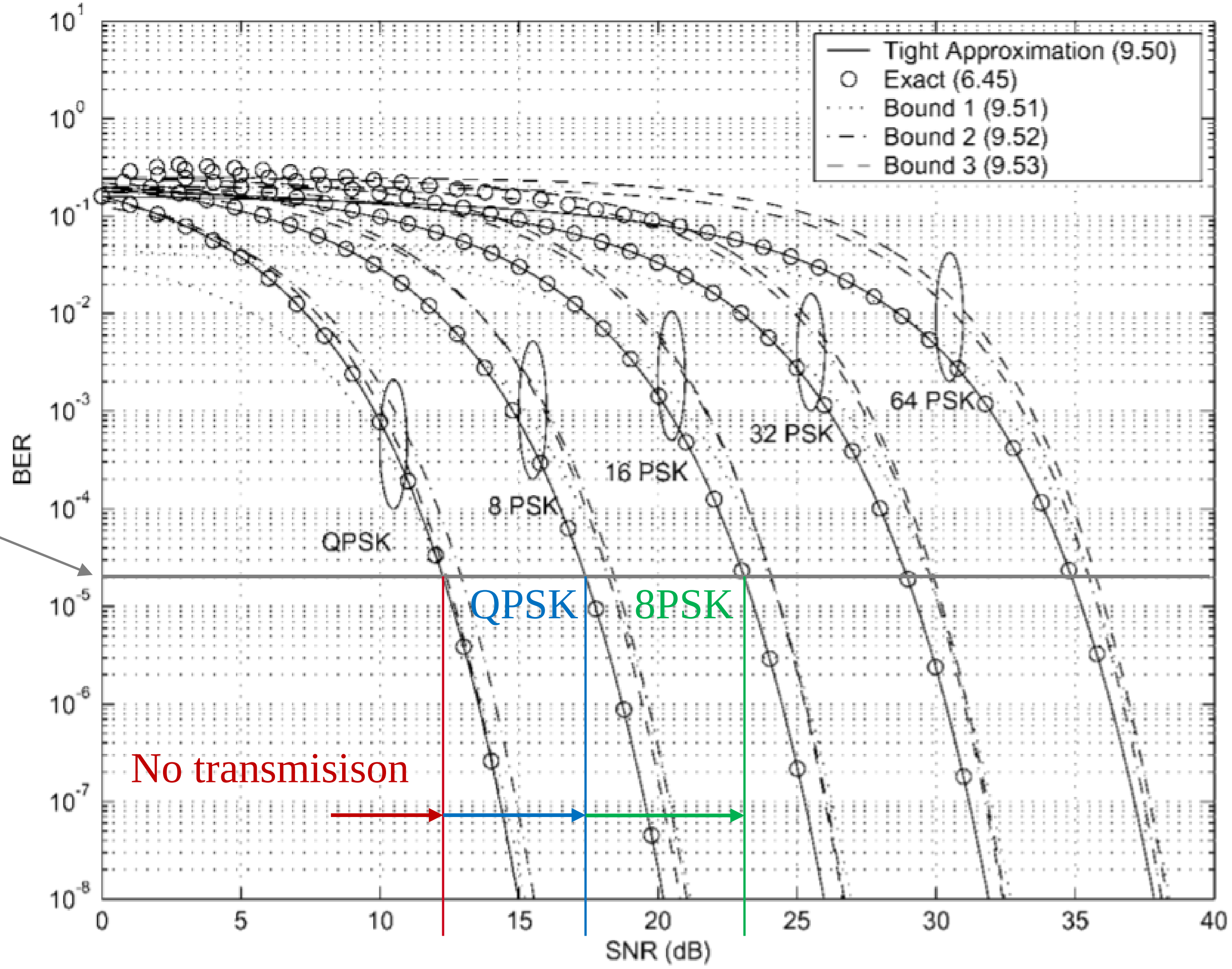
Rate adaptation: Modulation order selection

- Select maximum modulation order that satisfies target BER.
- The receiver (Rx) power required to achieve target BER (or better) with different modulation orders, can be obtained from approximate bit error probability for M-QAM.

$$\underbrace{P_b}_{\text{Target BER}} \leq 0.2e^{-1.5\overbrace{\gamma}^{\text{SNR}}/(\underbrace{M}_{\text{Number of bits per symbol}}-1)}$$

- The modulation order is determined by comparing the Rx power levels with γ levels available for modulations.

Target BER



We consider:
QPSK, 16QAM, 64QAM, 256QAM

[Source: Goldsmith2005]

Bit adaptation (`Transmitter.c`)

- Implement a function that allocates bits as described before, that assigns maximum available modulation order for which target BER is satisfied.

Bit allocation array

Channel SNR

```
int bit_allocation(int bitAlloc[], double gamma[], int numCarriers,
                   double snrAvg, int modBitsPerSym[], double snrTh[], int numMods)
```

Total number of allocated bits per block

Bits Per Symbol for modulations

SNR thresholds array

Number of considered modulations

Power adaptation (`Transmitter.c`)

- Implement a function that allocates power according to the waterfilling algorithm for a fixed modulation format.

Number of Active Subcarriers

Power allocation array

Channel SNR

```
int power_allocation(double powAlloc[], double gamma[], int numCarriers, double
        sqrtSNR)
```

Square root of SNR

Modify existing functions (Transmitter.c and Receiver.c)

- **generate_asymbol** is modified to support case with 0 bits - set I and Q signals to zero.

```
int generate_asymbol(unsigned char* bits, int len, double* symbol_I, double* symbol_Q);
```

- Write a new version of **generate_symbols**, to support variable modulation order:

```
int generate_symbols_ra(unsigned char txBits[], int bitsPerSymbol[], int howManySymbols,
        double txSymI[], double txSymQ[])
```

Array with number of bits per symbol
for all subcarriers (after adaptive loading)

- Modification is applied on this:

```
int decode_asymbol(double* recSymI, double* recSymQ, unsigned char* recBits, int bitsPerSymbol);
```

- Make appropriate modifications for the corresponding decoding functions, i.e.

```
void decode_symbols_ra(double* rxSymI, double* rxSymQ, int howManySymbols, int bitsPerSymbol[],
        unsigned char* rxBits);
```

Submission

- Submit the report before 13:00 on 13.06.2022.

- Attach your code.

- Write a few meaningful sentences about the obtained results.
  - How does BER with adaptation behave compared to the non-adaptive case?
  - What about the throughput?
  - Comment on the achievable system performance improvement with link adaptation.