Channel Model

The channel takes an input vector $x \in \mathbb{R}^{2n}$ and returns an output vector $Y \in \mathbb{R}^{2n}$. The input and output are related as

$$Y = Bx + Z,$$

where $B = \text{diag}(\underbrace{A, \ldots, A}_{n})$ is a block diagonal matrix[1], with $A = \begin{bmatrix} 11 & 10 \\ 10 & 11 \end{bmatrix}$, and $Z \sim \mathcal{N}(0, I_{2n})$, i.e., the components of Z are independent (i.i.d.) Gaussians with zero mean and variance 1.

To promote efficient communication, we insist on $n \leq 100$. We also enforce a constraint on the transmitted average energy — with $s = \frac{1}{2n}\sum_{i=1}^{2n} |x_i|^2$, we check if $s \leq 1$. If $s > 1$, then Z will be scaled by $\sqrt{s}$ to maintain the signal-to-noise ratio.

Assignment

a) Solve the theoretical part of the project.

b) Develop a system capable of reliably transmitting *text* messages over *this* channel. Specifically:

- Design a transmitter that reads a text message and returns real-valued samples of an information-bearing signal X.
- You send X to a server that applies channel effects, returning Y.
- Having received Y, your designed receiver must reconstruct the text message.

Submission and Evaluation Rules

- You work in teams of *three or four*.

  Please choose your teammates at the latest by **Friday, May 12** and send an email to `adway.girish@epfl.ch` with the subject "PDC project team" in order to register your team.

- For the theoretical part, please prepare one solution per team, and submit it through Moodle. The deadline for this submission is on **Wednesday, May 31**.
- For the practical part, you may use any programming language, as long as all the code pertaining to the transmitter and receiver is produced by your team.
- During the last session (**June 2**), each group presents their project in 5–10 minutes and gives a demonstration by transmitting a file that we provide.

  (i) You will run the transmitter and the receiver on your own laptop.

[1] `https://en.wikipedia.org/wiki/Block_matrix#Block_diagonal_matrices`

(ii) You must submit your team's code to Moodle before **Friday, June 2, 10 am**. For each team, it is sufficient to submit through the Moodle account of any one member.

(iii) The message which you will be asked to transmit is a text which contains *50 characters*. The text is guaranteed to be ASCII-encodable, i.e., each character is a 7-bit symbol.

(iv) Your presentation should contain a brief explanation of your signaling scheme (this is not expected to be very formal; there is no need to prepare slides, we simply want you to be able to explain your choices in designing your transmitter and receiver), followed by the transmission and decoding of the chosen text (that will be given to you on the spot).

(v) You will be given *two* chances for transmission, i.e., if you fail to transmit the message during your first transmission, you can repeat the transmission once more.

- The grade is based on the solution that you submit for the theoretical part, and the reliability of your scheme during the demonstration. The theoretical part counts for 5 points of the grade and the demonstration part counts for 15 points (together counting for a total of 20 points for the project). The demonstration part will be graded as follows:

  (a) If you manage to transmit the text message without error during one of the two transmissions, you will get full marks (15 pts).

  (b) Otherwise your score will be $\max\{10 - \varepsilon, 0\}$ where ε is the number of incorrect characters in the reproduced text at the receiver. We will grade based on the better of the two transmissions.

  (c) On top of that, the team which uses smallest n will get 5 additional (bonus) points.

  (d) You will be given 0 if you do not attend the presentation and your teammates cannot specify your contribution during the presentation.

Python Client

To simplify communication with the channel server, we provide you a Python script `client.py` that you can download from Moodle. Please read the associated docstrings for more information. You can only connect to the server if you are inside the EPFL network. If you are working outside of EPFL, you can use EPFL's VPN (please visit the following link[2] for more information). Please use the following connection parameters:

- `--srv_hostname=iscsrv72.epfl.ch`
- `--srv_port=80`

Tips and Practical Considerations

- To avoid overloading the server, we only allow each client to connect once every 30 seconds.

[2] `https://www.epfl.ch/campus/services/en/it-services/network-services/remote-intranet-access/`

- The input file for python client should be organized such that the first $2n$ lines of the input file contain the signal components $x_1, \ldots, x_{2n}$.
- The channel model is implemented on the server by the following function:

```
import numpy as np

def channel(sent_signal):
    sent_signal = np.array(sent_signal)
    assert np.size(sent_signal) <= 200, "n must be <= 100"
    n = np.size(sent_signal)//2
    x = sent_signal[0:2*n]
    s = sum(x**2)/np.size(x)
    sigma = 1
    if s > 1:
        sigma = np.sqrt(s)
    Z = np.random.normal(0, sigma, size=(2*n,1))
    A = np.array([[11,10],[10,11]])
    B = np.kron(np.eye(n),A)
    Y = B.dot(x) + Z
    return Y
```

  You can use this function to test your transmission scheme locally, but be sure to also check your scheme with the server using the python client as mentioned above.