Wireless network

Emilie Daniel - Mathis Boultoureau

Git repository : https://github.com/mboultoureau/wireless-network

📒 Lesson notes / General knowledge

💻 See code in the git project / notes on how we coded the function

⚙️ Awnser to direct question

👉 Conclusion

2.1 Extraction of the time frequency matrix.

We loaded the CSV file and observed the matrix.

- What is the size of the matrix ? Why such dimensions ?

⚙️ We can see that this is a matrix composed or 14 lines and 1024 rows. Each line correspond to a symbol and each row correspond to a subcarrier.

- Explain the new dimensions of the matrix.

⚙️ After removing the unallocated subcarriers we have a matrix composed of 14 lines and 624 rows. We kept the rows 1 to 313 and to 712 to 1024. This is becaus according to section 1.1.2 of the subject Only the 312 first subcarriers and the 312 last subcarriers are allocated. Furthermore the first subcarrier wont carry any data as it's a continuous current.

Here's the formula we used, with Sa the vector of allocated subcarriers, N the FFT size (1024) and Nre the number of allocated subcarriers (624).

$$S_a = U\left\{\left(\left(1 : \frac{N_{\mathrm{Re}}}{2}\right)\right); \left(N - \frac{N_{\mathrm{Re}}}{2} + 1 : N\right)\right\}$$

We then removed the 0 and 1 rows as the are used for synchronization and channel emission.

Conclusion

👉 At this stage, you should have the following elements

A working Python environment
3 defined Matrix with different sizes

2.2 PBCH decoding

Carefully read the section related to the PBCH description in Section 1.3.2. As stated in this
section, the PBCH has a fixed format: the symbol are BPSK and the FEC used is a Hamming748

2.2.1 BPSK decoding

1. Check that the first symbols associated to the PBCH belongs to a BSPK (or 2-QAM).

   ⚙️ We know that BPSK does not use imaginary numbers. So we have to display the first symbol of the first 5 or 6 subcarriers (once the lines 0 and 1 are removed) and check for imaginary numbers. As we can see there are no imaginary numbers so the symbols associated to the PBCH do belong to a BPSK.

2. Create a Python function that demodulates a BPSK stream into a binary stream. The
   function should have the following template
   Check that the function gives the correct output. This can be done by defining a unit test.
   For BPSK, this is done by defining the test_bpsk function defined in tests_modulation.py. Open the function and have a look on the source code. Extract the function and call it in
   your workspace.

   💻 See code

3. If you have a look on Section 1.3.2, the PBCH have a variable size, depending on the
   number of allocated users. We need first to decode the 48 bits to extract the cell and users
   information. From the qamMatrix variable, create the symbol stream associated to the first
   48 bits, and decode the bitstream.

   💻 See code

   We coded a function that demodulates a BPSK stream into a binary stream using the 1.2.2.1 section and especialle the 7th figure

📒 BPSK stands for Binary Phase Shift Keying, which is a type of digital modulation technique used in wireless communication systems. In BPSK, the amplitude and phase of a carrier signal are varied to represent digital 1 and 0. The received signal is then demodulated to extract the digital data. In the context of this document, BPSK decoding is used to extract information from the PBCH (Physical Broadcast Channel) of the wireless network.

2.2.2 Hamming748 decoder (18 users)

We now have a bitstream of 48 elements that contains the cell and user information. This
information is protected with the use of a Hamming748 coding system, as explained in Section
1.3.2. We need to create a Hamming decoding method.

1. Propose a Python function hamming748_decode that returns a decoded binary sequence
   that have been coded with a Hamming coder. The function should follow the following

   💻 See code

2. Check that your method is OK with its associated unitary test defined in tests_hammning.py

   💻 See code

📒 Hamming748 is an error-correcting code that can correct one error in every 748 bits. It’s based on the Hamming74 to which we had a parity bit that allows us to check for one more error and the output is now exactly double the input (24bits will come out as 48 bits).

2.2.3 PBCH decoding

We can now recover the useful information.

1. Create a dictionary associated to the first slot of the PBCH decoding that contains the
   useful information (cell size and number of users). What is the cell ident ? How many users
   are allocated in our network ? To convert a binary information into a value readable for
   humans, you can use the given function bin2dec

   💻 See code

2. Deduce the size of the full PBCH channel

   ⚙️ Full size of the PBCH channel :

   base frame (cell indent + number of users) + PBCHU size * number of users =

   48 + 48*18 = 912 (before decoding)

   = 456 after decoding with hamming

3. Proceed to decode all the PBCHU. Keep only the one associated to your user ident.

💻 See code

Our user ident is 11

4. Create a dictionary associated to the PBCHU of interest (i.e your PBCHU).

💻 See code

Conclusion

👉 At this stage you should know the following elements:

A functional BPSK demodulator and Hamming748 decoder : Yes (see code)

- The cell ident : It’s 12345
- The number of users that are allocated : We have 18 users allocated
- If you are allocated in the proposed frame : We can get our frame
- Where is located your PDCCHU : Symbole 4, RB 52
- What MCS your PDCCHU uses : 2 —> QPSK

2.3 PDCCH decoding

We want now to recover the information contains in the PDCCH. Read carefully the section related
to the PDCCH channel. The PBCH contains the location of each PDDCHU and we will decode
only the PDCCHU of interest. This channel uses a specific channel coder and a specific MCS.
You can recover the link between the MCS flag (in your PBCHU) and the chosen FEC/MCS in
Table 1.2.

2.3.1 MCS decoding

In this lab we will only use the Hamming748 system for our FEC. As a consequence, there is no
need to build a specific function as the FEC is exactly the same as the one used in the PBCH.On
the contrary, if the PBCH only uses BPSK, the PDCCHUs can use both BPSK and QPSK. We
thus need to define a QPSK demodulation function.

1. Create a template function qpsk_demod

   💻 See code

2. Form the Voronoi region defined in Section 1.2.2, propose and code a QPSK demodulation
   function.

   💻 See code

3. Check that your method works with the function test_qpsk.

   💻 See code

4. Built a function that can both decode a PDCCHU stream depending on the MCS flag value.
   This function should call the 2 decoding function, related to BPSK and QPSK decoding.

   💻 See code

2.3.2 PDCCHU decoding

We can now recover the stream.

1. Extract the QAM sequence of the appropriate length to recover the PDCCHU at the correct
   position in the TF grid

   💻 See code

2. Decode the sequence using the correct format

   💻 See code

3. Extract the information and create a Python dictionary that gather all the useful information.

   💻 See code

Conclusion

👉 At this stage you should have the following element
• The position of the user PDSCH : Symbol 10, RB 10
• The RB allocation of the PDSCH : 27
• The MCS used in the PDSCH : 7 = Convolutional Code and 16 QAM
• the CRC type used in the PDSCH : 0 polynom used = $1 + X + X^2 + X^8$
and we are now ready to decode the payload content.

2.4 PDSCH decoding

We can now decode the payload content. We first have to proposed some additional function to

be able to decode higher MCS, other FEC types and be able to use the CRC to decode the data.

2.4.1 FEC and MCS overall functions

We use a specific MCS for the PDSCH. This MCS leads to the support of
• Higher QAM format. In the project we only support BPSK, QPSK, and 16QAM. Support
of 64QAM and 256QAM will not be implemented.
• Use of CC FEC in addition to the Hamming748 FEC.

1. We provide the 16-QAM decoding function. Check that the test test_qam16.m passes.

   💻 See code

2. build a function PBCHU_demod(qamSeq,mcs) which apply the appropriate constellation demodulation based on the MCS flag obtained from the PDCCHU.

   💻 See code

3. Built a function PBCHU_fec(qamSeq,mcs) which apply the appropriate FEC decoder based
   on the associated MCS flag

   💻 See code

4. Built a function PBCHU_crc(qamSeq,crcSize) that uses the provided CRC functions and
   that apply the appropriate CRC check to the PDSCH data. The CRC decoding functions
   are provided and you will have to use crcDecode and getCRCPoly. Test with the unitary
   test function test_crcDecode.

💻 See code

2.4.2 PDSCH decoding

We are now ready to decode our PDSCH channel and recover the payload data.

1. Extract the symbol sequence related to the PDSCH

   💻 See code

2. Apply the Symbol demodulation to recover the bit sequence

   💻 See code

3. Apply the appropriate FEC decoder scheme. For the Convolutional decoder you can use the
   proposed decoding function from the python module (see the convolutional description to
   create c1)

   💻 See code

4. You Now have the payload ! Check its integrity with the use of the CRC check. The CRC
   has then to be removed from the payload content.

   💻 See code

5. Recover the ASCII sequence. The message is protected by a encrypted system. As this is out
   of scope of the project, the decoding method is given in the file

binary_transformation.py.
The decoding be done with the use of the provided functions (bitToByte then cesarDecode
and finally toASCII). For cesarDecode, you will need your user identifier.

💻 See code

Conclusion

👉 The project is now terminated. From the T/F matrix you should now have

• Decoded the payload associated to your user :

User 11 : your key is 61

Never gonna give you up

Never gonna let you down

…

N

• Print the payload text into the Python terminal : See code

• Discover the secret key ! : 61

Some last things before the end:

• Redo the decoding steps with the use of tf_matrix_2.mat. What happens ? What
can we do to address this issue ? The matrix seems to have been damaged because of the propagation channel.

• Redo the decoding steps with the use of tf_matrix_3.mat.What is the new key ? : It’s 52, the message is shorter because we have less ressources allocated. We only get

User 11 : your key is 52

Never gon

📕 In order to understand the concept of the lab we used this youtube vidéo : https://youtu.be/dQw4w9WgXcQ