My Project

Generated by Doxygen 1.13.2

Chapter 1

Telecommunications software engineering - Digital transmitter design and signal modulation

1.1 Project Goal

The goal of this project is to design a basic digital transmitter, focusing on:

- Signal generation,
- Implementation of various modulation techniques,
- Analysis of a communication channel with noise,
- Evaluation of the efficiency and robustness of different modulation schemes.

1.2 Project Description

This project explores the concept of a digital transmitter, how it operates, and how signals are generated and prepared for transmission through a communication channel. It delves into the practical aspects of signal modulation and the challenges of transmitting signals through noisy channels.

1.2.1 Main Activities

1. *Designing a digital transmitter:*
    - Developing a foundational digital transmitter using Python, focusing on modular design and clear signal processing steps.
2. *Implementation of modulation schemes:*
    - Amplitude Modulation (AM),
    - Amplitude Shift Keying (ASK),
    - Frequency Modulation (FM),
    - Frequency Shift Keying (FSK),

- Phase Shift Keying (PSK),
- Binary Phase Shift Keying (BPSK),
- Quadrature Amplitude Modulation (QAM),
- Quadrature Phase Shift Keying (QPSK).

3. *Simulation of a communication channel with noise:*
   - Implementing a channel with Additive White Gaussian Noise (AWGN).
   - Analyzing the Signal-to-Noise Ratio (SNR).
4. *Evaluation of efficiency and robustness:*
   - Assessing the energy efficiency of each modulation scheme.
   - Analyzing noise resistance.

1.3 Repository Structure

Directory/File name	Description
AM-ASK+tests	This directory contains all code files necessary for AM and ASK modulations, as well as their respective test files and modulation images.
FM-FSK+tests	This directory contains all code files necessary for FM and FSK modulations, as well as their respective test files and modulation images.
PSK-BPSK+tests	This directory contains all code files necessary for PSK and BPSK modulations, as well as their respective test files and modulation images.
QAM-QPSK+tests	This directory contains all code files necessary for QAM and QPSK modulations, as well as their respective test files and modulation images.
Images	This directory contains a variety of images related to modulations and tests.
.github	This directory contains test.yml file which checks all tests and will notify the user if the tests don't pass (useful for debugging)
.coverage	This file was generated with `pytest -v -cov`, it run all Python tests in the root directory
README.md	Project details.

1.4 Modulation techniques

1.4.1 Amplitude Modulation (AM)

Amplitude Modulation (AM) is a modulation technique where the amplitude of the carrier signal is varied in proportion to the instantaneous amplitude of the message signal. It's a simple method but is susceptible to noise.
Amplitude Modulation (AM) image

Figure 1: Amplitude modulation

1.4.2 Amplitude Shift Keying (ASK)

Amplitude Shift Keying (ASK) is a form of amplitude modulation that represents digital data as variations in the amplitude of a carrier wave. In its simplest form, the presence of a carrier wave represents a binary 1, and the absence represents a binary 0.
Amplitude Shift Keying (ASK) image

Figure 2: Amplitude shift keying

1.4.3 Frequency Modulation (FM)

Frequency Modulation (FM) is a modulation technique where the frequency of the carrier signal is varied in proportion to the instantaneous amplitude of the message signal. FM is less susceptible to noise than AM.
Frequency Modulation (FM) image

Figure 3: Frequency modulation

1.4.4 Frequency Shift Keying (FSK)

Frequency Shift Keying (FSK) is a form of frequency modulation that represents digital data as variations in the frequency of a carrier wave. Different frequencies are used to represent different binary values.
Frequency Shift Keying (FSK) image

Figure 4: Frequency shift keying

1.4.5 Phase Shift Keying (PSK)

Phase Shift Keying (PSK) is a modulation technique where the phase of the carrier signal is varied to represent digital data. The amplitude and frequency of the carrier signal remain constant.
Phase Shift Keying (PSK) image

Figure 5: Phase shift keying

1.4.6 Binary Phase Shift Keying (BPSK)

Binary Phase Shift Keying (BPSK) is a form of phase shift keying where the phase of the carrier signal is varied to represent binary data. It uses two phases to represent binary 0 and 1.
Binary Phase Shift Keying (BPSK) image

Figure 6: Binary phase shift keying

1.4.7 Quadrature Amplitude Modulation (QAM)

Quadrature Amplitude Modulation (QAM) is a modulation technique that combines both amplitude and phase modulation to transmit more data per symbol. It uses multiple amplitude levels and phase shifts to encode data.
Quadrature Amplitude Modulation (QAM) image

Figure 6: Quadrature amplitude modulation

1.4.8 Quadrature Phase Shift Keying (QPSK)

Quadrature Phase Shift Keying (QPSK) is a form of phase shift keying where the phase of the carrier signal is varied to represent digital data. It uses four phases to represent two bits of data.
Quadrature Phase Shift Keying (QPSK) image

Figure 7: Quadrature phase shift keying

1.5 Pytest

The pytest framework is used to run the tests for each modulation technique. The tests can be run with: 'pytest -v -cov 'file.py'`.
Figure 8 shows how pytest output looks when running the tests for all the modulation techniques:

Figure 8: Code coverage

1.6 Doxygen

Doxygen is a documentation generator used across several programming languages, including Python. It parses specially-formatted comments within the code to produce documentation in various formats like HTML, LaTeX, and man pages.

1.6.1 Doxyfile

A **Doxyfile** is Doxygen's configuration file, controlling the documentation generation process. It specifies settings such as input files, output directories, and documentation style. Doxyfile can be generated with: `doxygen -g Doxyfile`, and it can later be edited with a text editor, such as vim with: `vim Doxyfile`. Once we've configured Doxfile the way we want it to be, we can finally type: `doxygen Doxyfile` which will create `html` and `latex` directories, and later, `latex` directory can be used to compile a PDF document from there.

1.6.1.1 Key Doxyfile Settings

The Doxyfile uses a variety of settings to configure the documentation generation. Here are some of the key settings:

1.6.1.1.1 Project Information

- **`PROJECT_NAME`**: The name of the project.
- **`PROJECT_NUMBER`**: The version number of the project.

1.6.1.1.2 Input and Output

- **`OUTPUT_DIRECTORY`**: The directory where the generated documentation will be placed.
- **`INPUT`**: Specifies the input files or directories to be processed by Doxygen.
- **`FILE_PATTERNS`**: Specifies the file patterns to be included in the documentation.
- **`RECURSIVE`**: Whether to recursively search for input files in subdirectories.
- **`HTML_OUTPUT`**: Specifies the directory where the HTML documentation will be placed.
- **`LATEX_OUTPUT`**: Specifies the directory where the LaTeX documentation will be placed.

1.6.1.1.3 Extraction Options

- **`EXTRACT_ALL`**: Whether to extract documentation from all code elements, even if they are not explicitly documented.
- **`EXTRACT_PRIVATE`**: Whether to extract documentation from private members.
- **`EXTRACT_STATIC`**: Whether to extract documentation from static members.

1.6.1.1.4 Preprocessing and Includes

- **`ENABLE_PREPROCESSING`**: Whether to enable preprocessing of the input files.
- **`MACRO_EXPANSION`**: Whether to expand macros in the input files.
- **`SEARCH_INCLUDES`**: Whether to search for include files in the input files.
- **`INCLUDE_PATH`**: The path to the include files.

1.6.1.1.5 Other Settings

- **IMAGE_PATH**: The path to the images used in the documentation.
- **EXAMPLE_PATH**: The path to the example files.
- **GENERATE_HTML**: Whether to generate HTML documentation.
- **GENERATE_LATEX**: Whether to generate LaTeX documentation.
- **GENERATE_TREEVIEW**: Whether to generate a tree view of the documentation.

The Doxyfile is central to configuring Doxygen. The `INPUT` setting is used to specify the files or directories to be processed, and the `RECURSIVE` setting is important for projects with subdirectories. The current specified input is "." which means that all files in the current directory will be processed, and the Doxyfile was generated in the root directory. The `FILE_PATTERNS` setting allows for specific file types to be included in the documentation.

1.7 Technologies Used

- **Programming Language:** Python
- **Libraries:**
  - `numpy` - Numerical data processing.
  - `matplotlib` - Visualization of simulation results.
  - `scipy` - Noise generation and SNR analysis. .
  - `pytest` - Testing framework.
  - `coverage` - Code coverage analysis.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

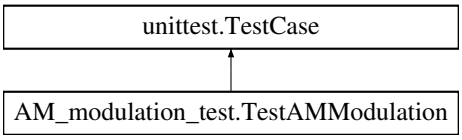
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 AM_modulation_test.TestAMModulation Class Reference

Inheritance diagram for AM_modulation_test.TestAMModulation:



Public Member Functions

- test_am_vector_lengths (self)
- test_carrier_signal_amplitude (self)
- test_am_signal_amplitude (self)
- test_am_signal_high_sampling_rate (self)
- test_am_signal_low_sampling_rate (self)
- test_am_large_duration (self)
- test_am_different_carrier_frequency (self)
- test_am_zero_carrier_frequency (self)
- test_am_signal_zero_duration (self)
- test_am_signal_edge_case_amplitude (self)

4.1.1 Member Function Documentation

4.1.1.1 test_am_different_carrier_frequency()

```
AM_modulation_test.TestAMModulation.test_am_different_carrier_frequency (
              self)
```

```
Test 7: The AM modulation function with a different carrier frequency.
```

4.1.1.2 test_am_large_duration()

```
AM_modulation_test.TestAMModulation.test_am_large_duration (
              self)
```

```
Test 6: The AM modulation function with a very large duration.
```

4.1.1.3 test_am_signal_amplitude()

```
AM_modulation_test.TestAMModulation.test_am_signal_amplitude (
              self)
```

Test 3: Verify that the amplitude-modulated signal remains within the range [-1, 1].

4.1.1.4 test_am_signal_edge_case_amplitude()

```
AM_modulation_test.TestAMModulation.test_am_signal_edge_case_amplitude (
              self)
```

Test 10: Edge case for amplitude boundary conditions.

4.1.1.5 test_am_signal_high_sampling_rate()

```
AM_modulation_test.TestAMModulation.test_am_signal_high_sampling_rate (
              self)
```

Test 4: The AM modulation function with a very high sampling rate.

4.1.1.6 test_am_signal_low_sampling_rate()

```
AM_modulation_test.TestAMModulation.test_am_signal_low_sampling_rate (
              self)
```

Test 5: The AM modulation function with a very low sampling rate.

4.1.1.7 test_am_signal_zero_duration()

```
AM_modulation_test.TestAMModulation.test_am_signal_zero_duration (
              self)
```

Test 9: The AM modulation function with zero duration.

4.1.1.8 test_am_vector_lengths()

```
AM_modulation_test.TestAMModulation.test_am_vector_lengths (
              self)
```

Test 1: Verify that all generated signal vectors have the correct length.

4.1.1.9 test_am_zero_carrier_frequency()

```
AM_modulation_test.TestAMModulation.test_am_zero_carrier_frequency (
              self)
```

Test 8: The AM modulation function with a zero carrier frequency.

4.1.1.10 test_carrier_signal_amplitude()

```
AM_modulation_test.TestAMModulation.test_carrier_signal_amplitude (
              self)
```

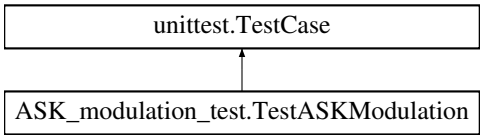
Test 2: Verify that the carrier signal's amplitude remains within the range [-1, 1].

The documentation for this class was generated from the following file:

- Modulation_files/AM-ASK+tests/AM_modulation_test.py

4.2 ASK_modulation_test.TestASKModulation Class Reference

Inheritance diagram for ASK_modulation_test.TestASKModulation:



Public Member Functions

- test_ask_modulation_array_lengths (self)
- test_ask_modulation_values (self)
- test_ask_large_binary_sequence (self)
- test_ask_single_bit_sequence (self)
- test_ask_all_zeros_sequence (self)
- test_ask_alternating_sequence (self)
- test_ask_long_same_bit_sequence (self)
- test_ask_modulation_basic_sequence (self)
- test_ask_empty_binary_sequence (self)
- test_ask_empty_input (self)

4.2.1 Member Function Documentation

4.2.1.1 test_ask_all_zeros_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_all_zeros_sequence (
              self)
```

Test 6: Verify handling of a sequence with all zeros.

4.2.1.2 test_ask_alternating_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_alternating_sequence (
              self)
```

Test 7: Verify handling of an alternating sequence.

4.2.1.3 test_ask_empty_binary_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_empty_binary_sequence (
              self)
```

Test 3: Verify that an empty binary sequence returns empty arrays.

4.2.1.4 test_ask_empty_input()

```
ASK_modulation_test.TestASKModulation.test_ask_empty_input (
              self)
```

Test 9: Verify that an empty input raises an error.

4.2.1.5 test_ask_large_binary_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_large_binary_sequence (
              self)
```

Test 4: Verify handling of a very large binary sequence.

4.2.1.6 test_ask_long_same_bit_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_long_same_bit_sequence (
            self)
```

Test 8: Verify handling of a long sequence of the same bit.

4.2.1.7 test_ask_modulation_array_lengths()

```
ASK_modulation_test.TestASKModulation.test_ask_modulation_array_lengths (
            self)
```

Test 1: Verify the lengths of all arrays returned by the `ask_modulation` function.

4.2.1.8 test_ask_modulation_basic_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_modulation_basic_sequence (
            self)
```

Test 10: Verify that a basic sequence [0, 1, 0] works.

4.2.1.9 test_ask_modulation_values()

```
ASK_modulation_test.TestASKModulation.test_ask_modulation_values (
            self)
```

Test 2: Verify the values of the binary wave generated during ASK modulation.

4.2.1.10 test_ask_single_bit_sequence()

```
ASK_modulation_test.TestASKModulation.test_ask_single_bit_sequence (
            self)
```

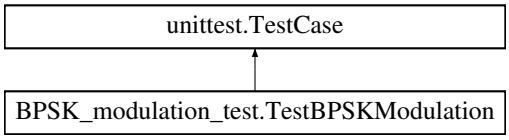
Test 5: Verify handling of a single-bit binary sequence.

The documentation for this class was generated from the following file:

- Modulation_files/AM-ASK+tests/ASK_modulation_test.py

4.3 BPSK_modulation_test.TestBPSKModulation Class Reference

Inheritance diagram for BPSK_modulation_test.TestBPSKModulation:



Public Member Functions

- test_basic_case (self)
- test_bit_duration (self)
- test_high_frequency (self)
- test_large_binary_data (self)
- test_single_bit (self)
- test_invalid_binary_data (self)
- test_zero_bit_duration (self)
- test_negative_sample_rate (self)
- test_negative_bit_duration (self)
- test_empty_binary_data (self)

4.3.1 Member Function Documentation

4.3.1.1 test_basic_case()

```
BPSK_modulation_test.TestBPSKModulation.test_basic_case (
              self)
```

Test the basic functionality of the BPSK modulation function.

Verifies:
- The lengths of all generated signals are consistent.
- The modulating signal corresponds correctly to the binary data input.

4.3.1.2 test_bit_duration()

```
BPSK_modulation_test.TestBPSKModulation.test_bit_duration (
              self)
```

Test the function with varying bit durations.

Verifies:
- The total duration of the generated signal matches the expected value.

4.3.1.3 test_empty_binary_data()

```
BPSK_modulation_test.TestBPSKModulation.test_empty_binary_data (
              self)
```

Test the function with empty binary data.

Verifies:
- The function returns empty arrays when the binary data is empty.

4.3.1.4 test_high_frequency()

```
BPSK_modulation_test.TestBPSKModulation.test_high_frequency (
              self)
```

Test the function with a high carrier frequency.

Verifies:
- The carrier signal is a normalized sine wave with amplitude in the range [-1, 1].

4.3.1.5 test_invalid_binary_data()

```
BPSK_modulation_test.TestBPSKModulation.test_invalid_binary_data (
              self)
```

Test the function with invalid binary input data.

Verifies:
- The function raises a ValueError for non-binary input data.

4.3.1.6 test_large_binary_data()

```
BPSK_modulation_test.TestBPSKModulation.test_large_binary_data (
              self)
```

Test the BPSK modulation function with a large binary data input.

Verifies:
- The function scales appropriately without errors.

4.3.1.7 test_negative_bit_duration()

```
BPSK_modulation_test.TestBPSKModulation.test_negative_bit_duration (
              self)

Test the function with a negative bit duration.

Verifies:
- The function raises a ValueError for a negative bit duration.
```

4.3.1.8 test_negative_sample_rate()

```
BPSK_modulation_test.TestBPSKModulation.test_negative_sample_rate (
              self)

Test the function with a negative sample rate.

Verifies:
- The function raises a ValueError for a negative sample rate.
```

4.3.1.9 test_single_bit()

```
BPSK_modulation_test.TestBPSKModulation.test_single_bit (
              self)

Test the BPSK modulation function with a single-bit binary data input.

Verifies:
- The generated signals correspond to the single bit.
```

4.3.1.10 test_zero_bit_duration()

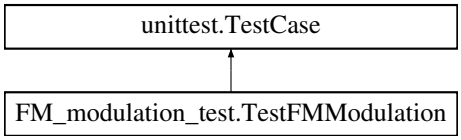
```
BPSK_modulation_test.TestBPSKModulation.test_zero_bit_duration (
              self)

Test the BPSK modulation function with zero bit duration.

Verifies:
- The function returns empty arrays.
```

The documentation for this class was generated from the following file:

- Modulation_files/PSK-BPSK+tests/BPSK_modulation_test.py

4.4 FM_modulation_test.TestFMModulation Class Reference

Inheritance diagram for FM_modulation_test.TestFMModulation:



Public Member Functions

- test_increasing_carrier_frequency (self)
- test_frequency_deviation_change (self)
- test_short_signal_duration (self)
- test_low_sample_rate (self)

- test_high_sample_rate (self)
- test_signal_amplitude_variation (self)
- test_large_duration (self)
- test_different_message_frequencies (self)
- test_invalid_input (self)
- test_negative_frequency_deviation (self)

4.4.1 Member Function Documentation

4.4.1.1 test_different_message_frequencies()

```
FM_modulation_test.TestFMModulation.test_different_message_frequencies (
              self)
```

```
Test FM modulation with different frequencies in the message signal.

Verifies:
- The FM signal reflects the changing message signal frequency.
```

4.4.1.2 test_frequency_deviation_change()

```
FM_modulation_test.TestFMModulation.test_frequency_deviation_change (
              self)
```

```
Test FM modulation with varying frequency deviation.

Verifies:
- The maximum instantaneous frequency deviation matches or exceeds the set value.
```

4.4.1.3 test_high_sample_rate()

```
FM_modulation_test.TestFMModulation.test_high_sample_rate (
              self)
```

```
Test FM modulation with a very high sampling rate.

Verifies:
- The function handles high-resolution signals correctly.
```

4.4.1.4 test_increasing_carrier_frequency()

```
FM_modulation_test.TestFMModulation.test_increasing_carrier_frequency (
              self)
```

```
Test FM modulation with a standard carrier frequency and parameters.

Verifies:
- Signal lengths are consistent across all generated components.
```

4.4.1.5 test_invalid_input()

```
FM_modulation_test.TestFMModulation.test_invalid_input (
              self)
```

```
Test FM modulation with invalid inputs.

Verifies:
- The function raises a ValueError for invalid parameter combinations.
```

4.4.1.6 test_large_duration()

```
FM_modulation_test.TestFMModulation.test_large_duration (
              self)
```

Test FM modulation with a very large signal duration.

Verifies:
- The function handles long signals without errors.

4.4.1.7 test_low_sample_rate()

```
FM_modulation_test.TestFMModulation.test_low_sample_rate (
              self)
```

Test FM modulation with a lower sampling rate.

Verifies:
- Signal lengths are consistent despite the lower resolution.

4.4.1.8 test_negative_frequency_deviation()

```
FM_modulation_test.TestFMModulation.test_negative_frequency_deviation (
              self)
```

Test FM modulation with a negative frequency deviation.

Verifies:
- The function raises a ValueError.

4.4.1.9 test_short_signal_duration()

```
FM_modulation_test.TestFMModulation.test_short_signal_duration (
              self)
```

Test FM modulation with a shorter signal duration.

Verifies:
- The total duration of the generated signal is as expected.

4.4.1.10 test_signal_amplitude_variation()

```
FM_modulation_test.TestFMModulation.test_signal_amplitude_variation (
              self)
```

Test FM modulation when the message signal amplitude changes.

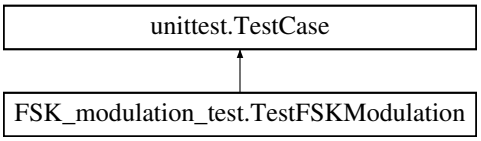
Verifies:
- The FM signal adapts correctly to varying amplitudes.

The documentation for this class was generated from the following file:

- Modulation_files/FM-FSK+tests/FM_modulation_test.py

4.5 FSK_modulation_test.TestFSKModulation Class Reference

Inheritance diagram for FSK_modulation_test.TestFSKModulation:



Public Member Functions

- test_basic_case (self)
- test_bit_duration (self)
- test_high_frequencies (self)
- test_large_binary_data (self)
- test_single_bit (self)
- test_invalid_binary_data (self)
- test_zero_bit_duration (self)
- test_negative_sample_rate (self)
- test_zero_carrier_frequencies (self)
- test_empty_binary_data (self)

4.5.1 Member Function Documentation

4.5.1.1 test_basic_case()

```
FSK_modulation_test.TestFSKModulation.test_basic_case (
              self)
```

Test the basic functionality of the FSK modulation function.

Verifies:
- The lengths of all generated signals are consistent.
- The modulating signal corresponds correctly to the binary data input.

4.5.1.2 test_bit_duration()

```
FSK_modulation_test.TestFSKModulation.test_bit_duration (
              self)
```

Test the function with varying bit durations.

Verifies:
- The total duration of the generated signal matches the expected value.

4.5.1.3 test_empty_binary_data()

```
FSK_modulation_test.TestFSKModulation.test_empty_binary_data (
              self)
```

Test the function with empty binary data.

Verifies:
- The function returns empty arrays when the binary data is empty.

4.5.1.4 test_high_frequencies()

```
FSK_modulation_test.TestFSKModulation.test_high_frequencies (
              self)
```

Test the function with high carrier frequencies.

Verifies:
- The carrier signals are normalized sine waves with amplitudes in the range [-1, 1].

4.5.1.5 test_invalid_binary_data()

```
FSK_modulation_test.TestFSKModulation.test_invalid_binary_data (
              self)
```

Test the function with invalid binary input data.

Verifies:
- The function raises a ValueError for non-binary input data.

4.5.1.6 test_large_binary_data()

```
FSK_modulation_test.TestFSKModulation.test_large_binary_data (
              self)
```

Test the FSK modulation function with a large binary data input.

Verifies:
- The function scales appropriately without errors.

4.5.1.7 test_negative_sample_rate()

```
FSK_modulation_test.TestFSKModulation.test_negative_sample_rate (
              self)
```

Test the function with a negative sample rate.

Verifies:
- The function raises a ValueError for a negative sample rate.

4.5.1.8 test_single_bit()

```
FSK_modulation_test.TestFSKModulation.test_single_bit (
              self)
```

Test the FSK modulation function with a single-bit binary data input.

Verifies:
- The generated signals correspond to the single bit.

4.5.1.9 test_zero_bit_duration()

```
FSK_modulation_test.TestFSKModulation.test_zero_bit_duration (
              self)
```

Test the FSK modulation function with zero bit duration.

Verifies:
- The function returns empty arrays.

4.5.1.10 test_zero_carrier_frequencies()

```
FSK_modulation_test.TestFSKModulation.test_zero_carrier_frequencies (
              self)
```

Test the function with zero carrier frequencies.

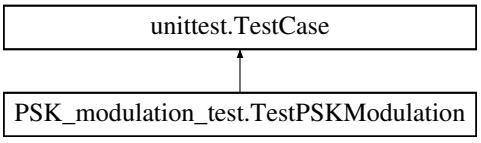
Verifies:
- The function does not raise an error, but the output signals are all zeros.

The documentation for this class was generated from the following file:

- Modulation_files/FM-FSK+tests/FSK_modulation_test.py

4.6 PSK_modulation_test.TestPSKModulation Class Reference

Inheritance diagram for PSK_modulation_test.TestPSKModulation:



Public Member Functions

- test_basic_case (self)
- test_bit_duration (self)
- test_high_frequency (self)
- test_large_binary_data (self)
- test_single_bit (self)
- test_invalid_binary_data (self)
- test_zero_bit_duration (self)
- test_negative_sample_rate (self)
- test_negative_bit_duration (self)
- test_empty_binary_data (self)

4.6.1 Member Function Documentation

4.6.1.1 test_basic_case()

```
PSK_modulation_test.TestPSKModulation.test_basic_case (
              self)
```

Test the basic functionality of the PSK modulation function.

Verifies:
- The lengths of all generated signals are consistent.
- The modulating signal corresponds correctly to the binary data input.

4.6.1.2 test_bit_duration()

```
PSK_modulation_test.TestPSKModulation.test_bit_duration (
              self)
```

Test the function with varying bit durations.

Verifies:
- The total duration of the generated signal matches the expected value.

4.6.1.3 test_empty_binary_data()

```
PSK_modulation_test.TestPSKModulation.test_empty_binary_data (
              self)
```

Test the function with empty binary data.

Verifies:
- The function returns empty arrays when the binary data is empty.

4.6.1.4 test_high_frequency()

```
PSK_modulation_test.TestPSKModulation.test_high_frequency (
              self)
```

Test the function with a high carrier frequency.

Verifies:
- The carrier signal is a normalized sine wave with amplitude in the range [-1, 1].

4.6.1.5 test_invalid_binary_data()

```
PSK_modulation_test.TestPSKModulation.test_invalid_binary_data (
              self)
```

Test the function with invalid binary input data.

Verifies:
- The function raises a ValueError for non-binary input data.

4.6.1.6 test_large_binary_data()

```
PSK_modulation_test.TestPSKModulation.test_large_binary_data (
              self)
```

Test the PSK modulation function with a large binary data input.

Verifies:
- The function scales appropriately without errors.

4.6.1.7 test_negative_bit_duration()

```
PSK_modulation_test.TestPSKModulation.test_negative_bit_duration (
              self)
```

Test the function with a negative bit duration.

Verifies:
- The function raises a ValueError for a negative bit duration.

4.6.1.8 test_negative_sample_rate()

```
PSK_modulation_test.TestPSKModulation.test_negative_sample_rate (
              self)
```

Test the function with a negative sample rate.

Verifies:
- The function raises a ValueError for a negative sample rate.

4.6.1.9 test_single_bit()

```
PSK_modulation_test.TestPSKModulation.test_single_bit (
              self)
```

Test the PSK modulation function with a single-bit binary data input.

Verifies:
- The generated signals correspond to the single bit.

4.6.1.10 test_zero_bit_duration()

```
PSK_modulation_test.TestPSKModulation.test_zero_bit_duration (
              self)
```

Test the PSK modulation function with zero bit duration.

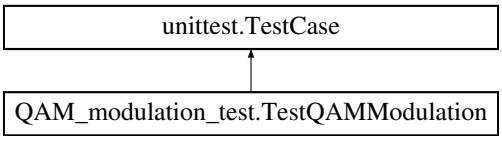
Verifies:
- The function returns empty arrays.

The documentation for this class was generated from the following file:

- Modulation_files/PSK-BPSK+tests/PSK_modulation_test.py

4.7 QAM_modulation_test.TestQAMModulation Class Reference

Inheritance diagram for QAM_modulation_test.TestQAMModulation:



Public Member Functions

- test_basic_case (self)
- test_bit_duration (self)
- test_high_frequency (self)
- test_large_binary_data (self)
- test_single_symbol (self)
- test_invalid_binary_data_length (self)
- test_invalid_constellation_points (self)
- test_invalid_constellation_points_number (self)
- test_negative_sample_rate (self)
- test_empty_binary_data (self)

4.7.1 Member Function Documentation

4.7.1.1 test_basic_case()

```
QAM_modulation_test.TestQAMModulation.test_basic_case (
              self)
```

Test the basic functionality of the QAM modulation function.

Verifies:
- The lengths of all generated signals are consistent.
- The modulating signal has the correct length.

4.7.1.2 test_bit_duration()

```
QAM_modulation_test.TestQAMModulation.test_bit_duration (
              self)
```

Test the function with varying bit durations.

Verifies:
- The total duration of the generated signal matches the expected value.

4.7.1.3 test_empty_binary_data()

```
QAM_modulation_test.TestQAMModulation.test_empty_binary_data (
              self)
```

Test the function with empty binary data.

Verifies:
- The function returns empty arrays when the binary data is empty.

4.7.1.4 test_high_frequency()

```
QAM_modulation_test.TestQAMModulation.test_high_frequency (
              self)
```

Test the function with a high carrier frequency.

Verifies:
- The carrier signals are normalized sine and cosine waves with amplitude in the range [-1, 1].

4.7.1.5 test_invalid_binary_data_length()

```
QAM_modulation_test.TestQAMModulation.test_invalid_binary_data_length (
              self)
```

Test the function with invalid binary input data length.

Verifies:
- The function raises a ValueError for binary data length not a multiple of bits per symbol.

4.7.1.6 test_invalid_constellation_points()

```
QAM_modulation_test.TestQAMModulation.test_invalid_constellation_points (
              self)
```

Test the function with invalid constellation points.

Verifies:
- The function raises a ValueError for non-complex constellation points.

4.7.1.7 test_invalid_constellation_points_number()

```
QAM_modulation_test.TestQAMModulation.test_invalid_constellation_points_number (
              self)
```

Test the function with invalid number of constellation points.

Verifies:
- The function raises a ValueError for number of constellation points not a power of 2.

4.7.1.8 test_large_binary_data()

```
QAM_modulation_test.TestQAMModulation.test_large_binary_data (
              self)
```

Test the QAM modulation function with a large binary data input.

Verifies:
- The function scales appropriately without errors.

4.7.1.9 test_negative_sample_rate()

```
QAM_modulation_test.TestQAMModulation.test_negative_sample_rate (
              self)
```

Test the QAM modulation function with a negative sample rate.

Verifies:
- The function raises a ValueError when sample_rate is negative.

4.7.1.10 test_single_symbol()

```
QAM_modulation_test.TestQAMModulation.test_single_symbol (
              self)
```

Test the QAM modulation function with a single symbol binary data input.

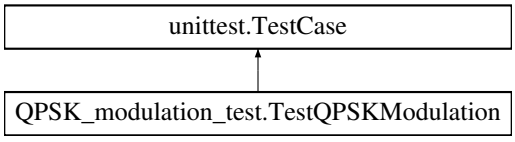
Verifies:
- The generated signals correspond to the single symbol.

The documentation for this class was generated from the following file:

- Modulation_files/QAM-QPSK+tests/QAM_modulation_test.py

4.8 QPSK_modulation_test.TestQPSKModulation Class Reference

Inheritance diagram for QPSK_modulation_test.TestQPSKModulation:



Public Member Functions

- test_basic_case (self)
- test_bit_duration (self)
- test_high_frequency (self)
- test_large_binary_data (self)
- test_single_symbol (self)
- test_invalid_binary_data_length (self)
- test_negative_sample_rate (self)
- test_empty_binary_data (self)
- test_zero_carrier_frequency (self)
- test_uneven_sample_rate_bit_duration (self)

4.8.1 Member Function Documentation

4.8.1.1 test_basic_case()

```
QPSK_modulation_test.TestQPSKModulation.test_basic_case (
              self)
```

Test the basic functionality of the QPSK modulation function.

Verifies:
- The lengths of all generated signals are consistent.
- The modulating signal has the correct length.

4.8.1.2 test_bit_duration()

```
QPSK_modulation_test.TestQPSKModulation.test_bit_duration (
              self)
```

Test the function with varying bit durations.

Verifies:
- The total duration of the generated signal matches the expected value.

4.8.1.3 test_empty_binary_data()

```
QPSK_modulation_test.TestQPSKModulation.test_empty_binary_data (
              self)
```

Test the function with empty binary data.

Verifies:
- The function returns empty arrays when the binary data is empty.

4.8.1.4 test_high_frequency()

```
QPSK_modulation_test.TestQPSKModulation.test_high_frequency (
              self)
```

Test the function with a high carrier frequency.

Verifies:
- The carrier signals are normalized sine and cosine waves with amplitude in the range [-1, 1].

4.8.1.5 test_invalid_binary_data_length()

```
QPSK_modulation_test.TestQPSKModulation.test_invalid_binary_data_length (
              self)
```

Test the function with invalid binary input data length.

Verifies:
- The function raises a ValueError for binary data length not a multiple of 2.

4.8.1.6 test_large_binary_data()

```
QPSK_modulation_test.TestQPSKModulation.test_large_binary_data (
              self)

Test the QPSK modulation function with a large binary data input.

Verifies:
- The function scales appropriately without errors.
```

4.8.1.7 test_negative_sample_rate()

```
QPSK_modulation_test.TestQPSKModulation.test_negative_sample_rate (
              self)

Test the QPSK modulation function with a negative sample rate.

Verifies:
- The function raises a ValueError when sample_rate is negative.
```

4.8.1.8 test_single_symbol()

```
QPSK_modulation_test.TestQPSKModulation.test_single_symbol (
              self)

Test the QPSK modulation function with a single symbol binary data input.

Verifies:
- The generated signals correspond to the single symbol.
```

4.8.1.9 test_uneven_sample_rate_bit_duration()

```
QPSK_modulation_test.TestQPSKModulation.test_uneven_sample_rate_bit_duration (
              self)

Test the function with a sample rate that is not a multiple of the bit duration.

Verifies:
- The function still generates signals without errors.
```

4.8.1.10 test_zero_carrier_frequency()

```
QPSK_modulation_test.TestQPSKModulation.test_zero_carrier_frequency (
              self)

Test the function with a zero carrier frequency.

Verifies:
- The carrier signals are constant (all zeros).
```

The documentation for this class was generated from the following file:

- Modulation_files/QAM-QPSK+tests/QPSK_modulation_test.py

Index