LDPC 5G

This module contains the API for using the GPU-accelerated LDPC coding chain from the cuPHY library. This includes encoding and decoding as well as rate matching. Code block segmentation and concatenation are implemented in Python. Additionally, this module contains a number of utility functions for example for determining the LDPC base graph, transport block size, etc.

class aerial.phy5g.ldpc.decoder.LdpcDecoder

LDPC decoder.

This class supports decoding of LDPC code blocks encoded following TS 38.212. It uses cuPHY accelerated LDPC decoding routines under the hood.

__init__(*num_iterations=10*, *num_profiling_iterations=0*, *half_precision=False*, *throughput_mode=False*, *max_num_code_blocks=152*, *cuda_stream=None*)

Initialize LdpcDecoder.

Initialization does all the necessary memory allocations for cuPHY.

Parameters

- **num_iterations** (*int*) – Number of LDPC decoder iterations. Default: 10.
- **num_profiling_iterations** (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0.
- **half_precision** (*bool*) – Enable FP16 processing. Default: False.
- **throughput_mode** (*bool*) – Enable throughput mode. Default: False.
- **max_num_code_blocks** (*int*) – Maximum number of code blocks. Memory is allocated based on this. Default: 152.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

Return type

None

decode(*input_llr*, *tb_size*, *code_rate*, *redundancy_version*, *rate_match_len*)

Decode function for LDPC decoder.

The decoder outputs decoded code blocks which can be further concatenated into the received transport block using `code_block_desegment()`.

Parameters

- **input_llr** (*np.ndarray*) – Input LLRs as a N x C array of 32-bit floats, N being the number of LLRs per code block and C being the number of code blocks.
- **tb_size** (*int*) – Transport block size in bits, without CRC.
- **code_rate** (*float*) – Target code rate.
- **redundancy_version** (*int*) – Redundancy version, i.e. 0, 1, 2, or 3.
- **rate_match_len** (*int*) – Number of rate matching output bits, the same as N.

Returns

The decoded bits in a numpy array.

Return type

np.ndarray

set_half_precision(*half_precision*)

Set half precision (float16) for decoding algorithm.

Parameters

half_precision (*bool*) – Enable (True) half precision for decoding.

Return type

None

set_num_iterations(*num_iterations*)

Set a particular value for the number of iterations to be run.

Parameters

num_iterations (*int*) – Value of the number of iterations.

Return type

None

set_profiling_iterations(*num_profiling_iterations*)

Set a particular value for the number of profiling iterations to be run.

Parameters

num_profiling_iterations (*int*) – Value of the number of profiling iterations.

Return type

None

set_throughput_mode(*throughput_mode*)

Enable throughput mode.

Parameters

throughput_mode (*bool*) – Enable (True) throughput mode.

Return type

None

class aerial.phy5g.ldpc.encoder.LdpcEncoder

LDPC encoder.

This class provides encoding of transmitted transport block bits using LDPC coding following TS 38.212. The encoding process is GPU accelerated using cuPHY routines. As the input, the transport blocks are assumed to be attached with the CRC and segmented to code blocks (as per TS 38.212).

__init__(*num_profiling_iterations=0*, *puncturing=True*, *max_num_code_blocks=152*, *cuda_stream=None*)

Initialize LdpcEncoder.

Initialization does all the necessary memory allocations for cuPHY.

Parameters

- **num_profiling_iterations** (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0.
- **puncturing** (*bool*) – Whether to puncture the systematic bits (2Zc). Default: True.
- **max_num_code_blocks** (*int*) – Maximum number of code blocks. Memory is allocated based on this. Default: 152.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

Return type

None

encode(*input_data*, *tb_size*, *code_rate*, *redundancy_version*)

Encode function for LDPC encoder.

The input to this function is code blocks, meaning that the code block segmentation is expected to be done before calling this function. Code block segmentation can be done using `code_block_segment()`.

Parameters

- **input_data** (*np.ndarray*) – The input code blocks as a K x C array where K is the number of input bits per code block (including CRCs) and C is the number of code blocks. The dtype of the input array must be *np.float32*.
- **tb_size** (*int*) – Transport block size in bits, without CRC.
- **code_rate** (*float*) – Target code rate.

- **redundancy_version** (*int*) – Redundancy version, 0, 1, 2, or 3.

Returns

Encoded bits as a N x C array where N is the number of encoded bits per code block.

Return type

np.ndarray

set_profiling_iterations(*num_profiling_iterations*)

Set a particular value for the number of profiling iterations to be run.

Parameters

num_profiling_iterations (*int*) – Value of the number of profiling iterations.

Return type

None

set_puncturing(*puncturing*)

Set puncturing flag.

Parameters

puncturing (*bool*) – Whether to puncture the systematic bits (2*Zc). Default: True.

Return type

None

class aerial.phy5g.ldpc.rate_match.LdpcRateMatch

LDPC rate matching.

__init__(*enable_scrambling=True*, *num_profiling_iterations=0*, *max_num_code_blocks=152*, *cuda_stream=None*)

Initialize LdpcRateMatch.

Initialization does all the necessary memory allocations for cuPHY.

Parameters

- **enable_scrambling** (*bool*) – Whether to enable scrambling after code block concatenation.
- **num_profiling_iterations** (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0 (no profiling).
- **max_num_code_blocks** (*int*) – Maximum number of code blocks. Memory will be allocated based on this number.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

Return type

None

rate_match(*input_data*, *tb_size*, *code_rate*, *rate_match_len*, *mod_order*, *num_layers*, *redundancy_version*, *cinit*)

LDPC rate matching function.

This function does rate matching of LDPC code blocks following TS 38.212. If scrambling is enabled, it also scrambles the rate matched bits. In this case the *c_init* value needs to be set to an appropriate scrambling sequence initialization value.

Parameters

- **input_data** (*np.ndarray*) – Input bits as a N x C numpy array with dtype *np.float32*, where N is the number of bits per code block and C is the number of code blocks.
- **tb_size** (*int*) – Transport block size in bits without CRC.
- **code_rate** (*float*) – Code rate.
- **rate_match_len** (*int*) – Number of rate matching output bits.
- **mod_order** (*int*) – Modulation order.

- **num_layers** (*int*) – Number of layers.
- **redundancy_version** (*int*) – Redundancy version, i.e. 0, 1, 2, or 3.
- **cinit** (*int*) – The *c_init* value used for initializing scrambling.

Returns

Rate matched bits.

Return type

np.ndarray

set_profiling_iterations(*num_profiling_iterations*)

Set a particular value for the number of profiling iterations to be run.

Parameters

num_profiling_iterations (*int*) – Value of the number of profiling iterations.

Return type

None

class aerial.phy5g.ldpc.derate_match.LdpcDeRateMatch

LDPC derate matching.

__init__(*enable_scrambling=True*, *num_profiling_iterations=0*, *max_num_code_blocks=152*, *cuda_stream=None*)

Initialize LdpcDeRateMatch.

Initialization does all the necessary memory allocations for cuPHY.

Parameters

- **enable_scrambling** (*bool*) – Whether to descramble the bits before derate matching. Default: True.
- **num_profiling_iterations** (*int*) – Number of profiling iterations. Set to 0 to disable profiling. Default: 0 (no profiling).

- **max_num_code_blocks** (*int*) – Maximum number of code blocks. Memory will be allocated based on this number. Default: 152.

- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

Return type

None

derate_match(*input_data*, *tb_size*, *code_rate*, *rate_match_len*, *mod_order*, *num_layers*, *redundancy_version*, *ndi*, *cinit*)

LDPC derate matching function.

Parameters

- **input_data** (*np.ndarray*) – Input LLRs as a N x 1 numpy array with dtype *np.float32*, where N is the number of LLRs coming from the equalizer. Ordering of this input data is *bitsPerQam x numLayers x numSubcarriers x numDataSymbols*.

- **tb_size** (*int*) – Transport block size in bits without CRC.

- **code_rate** (*float*) – Code rate.

- **rate_match_len** (*int*) – Number of rate matching output bits, the same as N.

- **mod_order** (*int*) – Modulation order.

- **num_layers** (*int*) – Number of layers.

- **redundancy_version** (*int*) – Redundancy version, i.e. 0, 1, 2, or 3.

- **ndi** (*int*) – New data indicator.

- **cinit** (*int*) – The *c_init* value used for initializing scrambling.

Returns

Derate matched LLRs.

Return type

np.ndarray

set_profiling_iterations(*num_profiling_iterations*)

Set a particular value for the number of profiling iterations to be run.

Parameters

num_profiling_iterations (*int*) – Value of the number of profiling iterations.

Return type

None

aerial.phy5g.ldpc.util.get_mcs(*mcs*, *table_idx=2*)

Get modulation order and code rate based on MCS index.

Parameters

- **mcs** (*int*) – MCS index pointing to the table indicated by *table_idx*.
- **table_idx** (*int*) – Index of the MCS table in TS 38.214 section 5.1.3.1. Values: - 1: TS38.214, table 5.1.3.1-1. - 2: TS38.214, table 5.1.3.1-2. - 3: TS38.214, table 5.1.3.1-3.

Returns

A tuple containing:

- *int*: Modulation order.
- *float*: Code rate * 1024.

Return type

int, float

aerial.phy5g.ldpc.util.get_tb_size(*mod_order*, *code_rate*, *dmrs_syms*, *num_prbs*, *start_sym*, *num_symbols*, *num_layers*)

Get transport block size based on given parameters.

Determine transport block size as per TS 38.214 section 5.1.3.2.

Parameters

- **mod_order** (*int*) – Modulation order.
- **code_rate** (*float*) – Code rate * 1024 as in section 5.1.3.1 of TS 38.214.
- **dmrs_syms** (*List[int]*) – List of binary numbers indicating which symbols contain DMRS.
- **num_prbs** (*int*) – Number of PRBs.
- **start_sym** (*int*) – Starting symbol.
- **num_symbols** (*int*) – Number of symbols.
- **num_layers** (*int*) – Number of layers.

Returns

Transport block size in bits.

Return type

int

aerial.phy5g.ldpc.util.get_base_graph(*tb_size*, *code_rate*)

Get LDPC base graph.

Parameters

- **tb_size** (*int*) – Transport block size in bits, without CRC.
- **code_rate** (*float*) – Code rate.

Returns

Base graph, 1 or 2.

Return type

int

aerial.phy5g.ldpc.util.max_code_block_size(*base_graph*)

Get maximum LDPC code block size based on base graph.

Parameters

base_graph (*int*) – Base graph, 1 or 2.

Returns

Maximum code block size.

Return type

int

aerial.phy5g.ldpc.util.find_lifting_size(*base_graph*, *tb_size*)

Find lifting size for base graph.

Parameters

- **base_graph** (*int*) – Base graph, 1 or 2.
- **tb_size** (*int*) – Transport block size in bits without CRC.

Returns

Lifting size.

Return type

int

aerial.phy5g.ldpc.util.get_num_info_nodes(*base_graph*, *tb_size*)

Get number of information nodes.

Note: This is the value *K_b* in TS 38.212.

Parameters

- **base_graph** (*int*) – Base graph, 1 or 2.
- **tb_size** (*int*) – Transport block size without any CRCs.

Returns

The number of information nodes (*K_b*).

Return type

int

aerial.phy5g.ldpc.util.get_code_block_num_info_bits(*base_graph*, *tb_size*)

Get number of information bits in a code block.

This is the number K’ in TS 38.212, i.e. the number of information bits without the filler bits.

Parameters

- **base_graph** (*int*) – Base graph, 1 or 2.
- **tb_size** (*int*) – Transport block size in bits, without CRC.

Returns

Number of information bits in a code block.

Return type

int

aerial.phy5g.ldpc.util.get_code_block_size(*tb_size*, *code_rate*)

Get code block size.

This is the number K in TS 38.212, i.e. the number of information bits including filler bits.

Parameters

- **tb_size** (*int*) – Transport block size in bits, without CRC.

- **code_rate** (*float*) – Code rate.

Returns

Code block size.

Return type

int

aerial.phy5g.ldpc.util.get_num_code_blocks(*tb_size*, *code_rate*)

Return the number of code blocks for a transport block.

Parameters

- **tb_size** (*int*) – Transport block size in bits, without CRC.
- **code_rate** (*float*) – Code rate.

Returns

The number of code blocks (C).

Return type

int

aerial.phy5g.ldpc.util.code_block_segment(*tb_size*, *transport_block*, *code_rate*)

Do code block segmentation.

This function does code block segmentation as per TS 38.212 section 5.2.2. Randomly generated 24-bit string is attached to each code block to emulate code block CRC if there is more than one code block.

Parameters

- **tb_size** (*int*) – Transport block size in bits, without CRC.
- **transport_block** (*np.ndarray*) – Transport block in bits, CRC included.
- **code_rate** (*float*) – Code rate.

Returns

The code blocks.

Return type

np.ndarray

aerial.phy5g.ldpc.util.code_block_desegment(*code_blocks*, *tb_size*, *code_rate*, *return_bits=True*)

Concatenate code blocks coming from LDPC decoding into a transport block.

This function desegments code blocks into a transport block as per TS 38.212, and removes the CRCs, i.e. does the opposite of `code_block_segment()`.

Parameters

- **code_blocks** (*np.ndarray*) – The code blocks coming out of the LDPC decoder as a N x C array.
- **tb_size** (*int*) – Transport block size, without CRC.
- **code_rate** (*float*) – Code rate.
- **return_bits** (*bool*) – If True (default), give the return value in bits. Otherwise convert to bytes.

Returns

The transport block with CRC, in bits or bytes depending on the value of *return_bits*.

Return type

np.ndarray

aerial.phy5g.ldpc.util.add_crc_len(*tb_size*)

Append CRC length to transport block size.

Parameters

tb_size (*int*) – Transport block size in bits without CRC.

Returns

Transport block size in bits with CRC.

Return type

int

aerial.phy5g.ldpc.util.random_tb(*mod_order*, *code_rate*, *dmrs_syms*, *num_prbs*, *start_sym*, *num_symbols*, *num_layers*, *return_bits=False*)

Generate a random transport block.

Generates random transport block according to given parameters. The transport block size is first determined as per TS 38.214 section 5.1.3.2.

Parameters

- **mod_order** (*int*) – Modulation order.
- **code_rate** (*float*) – Code rate * 1024 as in section 5.1.3.1 of TS 38.214.
- **dmrs_syms** (*List[int]*) – List of binary numbers indicating which symbols contain DMRS.
- **num_prbs** (*int*) – Number of PRBs.
- **start_sym** (*int*) – Starting symbol.
- **num_symbols** (*int*) – Number of symbols.
- **num_layers** (*int*) – Number of layers.
- **return_bits** (*bool*) – Whether to return the transport block in bits (True) or bytes (False).

Returns

Random transport block payload.

Return type

np.ndarray

aerial.phy5g.ldpc.util.get_crc_len(*tb_size*)

Return CRC length based on transport block size.

Parameters

tb_size (*int*) – Transport block size in bits without CRC.

Returns

CRC length (either 16 or 24 bits).

Return type

int