Receiver algorithms

This module contains a number of receiver algorithms implemented in cuPHY, thus using GPU acceleration.

class aerial.phy5g.algorithms.channel_estimator.ChannelEstimator

Channel estimator class.

This class implements traditional MMSE-based channel estimation on the DMRS symbols of the received slot signal. It outputs the channel estimates for all resource elements in the DMRS symbols. Similarly to many other classes in pyAerial, this class handles groups of UEs sharing the same time-frequency resources with one call, i.e. it supports MU-MIMO.

__init__(*num_rx_ant*, *cuda_stream=None*, *chest_filter_h5=None*, *w_freq_array=None*, *w_freq4_array=None*, *w_freq_small_array=None*, *shift_seq_array=None*, *unshift_seq_array=None*, *shift_seq4_array=None*, *unshift_seq4_array=None*)

Initialize ChannelEstimator.

The channel estimation filters can be given as an H5 file or directly as Numpy arrays. If neither is given, the channel estimator is using default filters.

Parameters

- **num_rx_ant** (*int*) – Number of receive antennas.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.
- **chest_filter_h5** (*str*) – Filename of an HDF5 file containing channel estimation filters.
- **w_freq_array** (*np.ndarray*) –
- **w_freq4_array** (*np.ndarray*) –
- **w_freq_small_array** (*np.ndarray*) –
- **shift_seq_array** (*np.ndarray*) –
- **unshift_seq_array** (*np.ndarray*) –

- **shift_seq4_array** (*np.ndarray*) –
- **unshift_seq4_array** (*np.ndarray*) –

Return type

None

estimate(*rx_slot*, *num_ues*, *slot*, *num_dmrs_cdm_grps_no_data*, *dmrs_scrm_id*, *start_prb*, *num_prbs*, *dmrs_syms*, *dmrs_max_len*, *dmrs_add_ln_pos*, *start_sym*, *num_symbols*, *scids*, *layers*, *dmrs_ports*)

Run the channel estimation.

This runs the cuPHY channel estimation for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

Parameters

- **rx_slot** (*np.ndarray*) – Input received data as a frequency x time x Rx antenna Numpy array with type *np.complex64* entries.
- **num_ues** (*int*) – Number of UEs in the UE group.
- **layers** (*List[int]*) – Number of layers for each UE. The length of the list equals the number of UEs.
- **scids** (*List[int]*) – DMRS sequence initialization SCID [TS38.211, sec 7.4.1.1.2] for each UE. Value is 0 or 1.
- **dmrs_ports** (*List[int]*) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.
- **slot** (*int*) – Slot number.
- **dmrs_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.

- **dmrs_scrm_id** (*int*) – DMRS scrambling ID.
- **dmrs_max_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos** (*int*) – Number of additional DMRS positions.
- **num_dmrs_cdm_grps_no_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **start_prb** (*int*) – Start PRB index of the UE allocation.
- **num_prbs** (*int*) – Number of allocated PRBs for the UE group.
- **start_sym** (*int*) – Start symbol index for the UE group allocation.
- **num_symbols** (*int*) – Number of symbols in the UE group allocation.

Returns

The channel estimates as a Rx ant x layer x frequency x time Numpy array, per UE group. Note: Currently this only supports a single UE group, i.e. the length of the list is one.

Return type

List[np.ndarray]

class aerial.phy5g.algorithms.noise_intf_estimator.NoiseIntfEstimator

Noise and interference estimator class.

This class implements an algorithm for noise and interference estimation. It calls the corresponding cuPHY algorithms and provides the estimates as needed for cuPHY equalization and soft demapping.

It needs channel estimates as its input, along with the received data symbols.

__init__(*num_rx_ant*, *eq_coeff_algo*, *cuda_stream=None*)

Initialize NoiseIntfEstimator.

Parameters

- **num_rx_ant** (*int*) – Number of receive antennas.
- **eq_coeff_algo** (*int*) –

  Algorithm used to compute equalizer coefficients.

  - 0: Zero-forcing equalizer.
  - 1: MMSE with noise variance only.
  - 2: MMSE-IRC.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

Return type

None

estimate(*rx_slot*, *channel_est*, *num_ues*, *slot*, *num_dmrs_cdm_grps_no_data*, *dmrs_scrm_id*, *start_prb*, *num_prbs*, *dmrs_syms*, *dmrs_max_len*, *dmrs_add_ln_pos*, *start_sym*, *num_symbols*, *scids*, *layers*, *dmrs_ports*)

Estimate noise and interference.

This runs the cuPHY noise and interference estimation for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

Parameters

- **rx_slot** (*np.ndarray*) – Input received data as a frequency x time x Rx antenna Numpy array with type *np.complex64* entries.
- **channel_est** (*List[np.ndarray]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy array, per UE group. Note: Currently this only supports a single UE group, i.e. the length of the list is one.
- **num_ues** (*int*) – Number of UEs in the UE group.
- **slot** (*int*) – Slot number.

- **num_dmrs_cdm_grps_no_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **dmrs_scrm_id** (*int*) – DMRS scrambling ID.
- **start_prb** (*int*) – Start PRB index of the UE allocation.
- **num_prbs** (*int*) – Number of allocated PRBs for the UE group.
- **dmrs_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.
- **dmrs_max_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos** (*int*) – Number of additional DMRS positions.
- **start_sym** (*int*) – Start symbol index for the UE group allocation.
- **num_symbols** (*int*) – Number of symbols in the UE group allocation.
- **scids** (*List[int]*) – DMRS sequence initialization SCID [TS38.211, sec 7.4.1.1.2] for each UE. Value is 0 or 1.
- **layers** (*List[int]*) – Number of layers for each UE. The length of the list equals the number of UEs.
- **dmrs_ports** (*List[int]*) – DMRS ports for each UE. The format of each entry is in the SCF FAPI format as follows: A bitmap (mask) starting from the LSB where each bit indicates whether the corresponding DMRS port index is used.

Returns

A tuple containing:

- *List[np.ndarray]*: Inverse of the Cholesky decomposition of the noise/interference covariance matrix per PRB, per UE group. The size of each entry in this list is number of Rx antennas x number of Rx antennas x number of PRBs. ote that since only one UE group is currently supported, the length of this list is one.

- *np.ndarray*: Pre-equalization wideband noise variance estimate per UE group, i.e. one value per UE group averaged over the whole frequency allocation. This value is in dB.

Return type

List[np.ndarray], np.ndarray

class aerial.phy5g.algorithms.channel_equalizer.ChannelEqualizer

Channel equalizer class.

This class implements MMSE-based channel equalization along with soft demapping to get log-likelihood ratios for channel decoding.

It needs channel estimates and noise and interference estimates as its input, along with the received data symbols.

__init__(*num_rx_ant*, *eq_coeff_algo*, *enable_pusch_tdi*, *cuda_stream=None*)

Initialize ChannelEqualizer.

Parameters

- **num_rx_ant** (*int*) – Number of receive antennas.
- **eq_coeff_algo** (*int*) –

  Algorithm used to compute equalizer coefficients.

  - 0: Zero-forcing equalizer.
  - 1: MMSE with noise variance only.
  - 2: MMSE-IRC.
- **enable_pusch_tdi** (*int*) – Whether to use time-domain interpolation.
- **cuda_stream** (*int*) – The CUDA stream. If not given, one will be created.

Return type

None

equalize(*rx_slot*, *channel_est*, *lw_inv*, *noise_var_pre_eq*, *num_ues*, *num_dmrs_cdm_grps_no_data*, *start_prb*, *num_prbs*, *dmrs_syms*, *dmrs_max_len*, *dmrs_add_ln_pos*, *start_sym*, *num_symbols*, *layers*, *mod_orders*)

Run equalization and soft demapping.

This runs the cuPHY equalization for a single UE group sharing the same time-frequency resources, i.e. having the same PRB allocation, and the same start symbol and number of allocated symbols.

Parameters

- **rx_slot** (*np.ndarray*) – Input received data as a frequency x time x Rx antenna Numpy array with type *np.complex64* entries.
- **channel_est** (*List[np.ndarray]*) – The channel estimates as a Rx ant x layer x frequency x time Numpy array, per UE group. Note: Currently this only supports a single UE group, i.e. the length of the list is one.
- **lw_inv** (*List[np.ndarray]*) – Inverse of the Cholesky decomposition of the noise/interference covariance matrix per PRB, per UE group. The size of each entry in this list is number of Rx antennas x number of Rx antennas x number of PRBs.
- **noise_var_pre_eq** (*np.ndarray*) – Average pre-equalizer noise variance in dB. One value per UE group.
- **num_ues** (*int*) – Number of UEs in the UE group.
- **num_dmrs_cdm_grps_no_data** (*int*) – Number of DMRS CDM groups without data [3GPP TS 38.212, sec 7.3.1.1]. Value: 1->3.
- **start_prb** (*int*) – Start PRB index of the UE allocation.
- **num_prbs** (*int*) – Number of allocated PRBs for the UE group.
- **dmrs_syms** (*List[int]*) – For the UE group, a list of binary numbers each indicating whether the corresponding symbol is a DMRS symbol. The length of the list equals the number of symbols in the slot. 0 means no DMRS in the symbol and 1 means the symbol is a DMRS symbol.

- **dmrs_max_len** (*int*) – The *maxLength* parameter, value 1 or 2, meaning that DMRS are single-symbol DMRS or single- or double-symbol DMRS.
- **dmrs_add_ln_pos** (*int*) – Number of additional DMRS positions.
- **start_sym** (*int*) – Start symbol index for the UE group allocation.
- **num_symbols** (*int*) – Number of symbols in the UE group allocation.
- **layers** (*List[int]*) – Number of layers for each UE.
- **mod_orders** (*List[int]*) – QAM modulation order for each UE.

Returns

A tuple containing:

- *List[np.ndarray]*: Log-likelihood ratios for the received bits to be fed into decoding (rate matching). One Numpy array per UE group and the size of each Numpy array is 8 x number of layers x number of subcarriers x number of data symbols. The size of the first dimension is fixed to eight as modulations up to 256QAM are supported and cuPHY returns the same size independently of modulation. Only the first entries corresponding to the actual number of bits are used.
- *List[np.ndarray]*: Equalized symbols, one Numpy array per UE group. The size of each Numpy array is equal to number of layers x number of subcarriers x number of data symbols.

Return type

List[np.ndarray], List[np.ndarray]

class aerial.phy5g.algorithms.demapper.Demapper

This class provides demapping of symbols to log-likelihood ratios.

The algorithm used is the exact log-MAP mapping, which is computationally intensive. Note also that this is currently implemented purely in Python so it may be slow.

__init__(*mod_order*)

Initialize demapper.

Parameters

mod_order (*int*) – Modulation order. Supported values: 2, 4, 6, 8.

Return type

None

demap(*syms*, *noise_var_inv*)

Run demapping.

Parameters

- **syms** (*np.ndarray*) – An array of modulation symbols.
- **noise_var_inv** (*np.ndarray*) – Inverse of noise variance per subcarrier. The size of this array must broadcast with *syms*.

Returns

Log-likelihood ratios. The first dimension is modulation order, otherwise the dimensions are the same as those of *syms*.

Return type

np.ndarray

class aerial.phy5g.algorithms.srs_channel_estimator.SrsCellPrms

SRS cell parameters.

A list of SRS cell parameters is given to the SRS channel estimator as input, one entry per cell.

Parameters

- **slot_num** (*np.uint16*) – Slot number.
- **frame_num** (*np.uint16*) – Frame number.

- **srs_start_sym** (*np.uint8*) – SRS start symbol.
- **num_srs_sym** (*np.uint8*) – Number of SRS symbols.
- **num_rx_ant_srs** (*np.uint16*) – Number of SRS Rx antennas.
- **mu** (*np.uint8*) – Subcarrier spacing parameter, see TS 38.211.

class aerial.phy5g.algorithms.srs_channel_estimator.UeSrsPrms

UE SRS parameters.

A list of UE SRS parameters is given to the SRS channel estimator as input, one entry per UE.

Parameters

- **cell_idx** (*np.uint16*) – Index of cell user belongs to.
- **num_ant_ports** (*np.uint8*) – Number of SRS antenna ports. 1,2, or 4.
- **num_syms** (*np.uint8*) – Number of SRS symbols. 1,2, or 4.
- **num_repetitions** (*np.uint8*) – Number of repititions. 1,2, or 4.
- **comb_size** (*np.uint8*) – SRS comb size. 2 or 4.
- **start_sym** (*np.uint8*) – Starting SRS symbol. 0 - 13.
- **sequence_id** (*np.uint16*) – SRS sequence ID. 0 - 1023.
- **config_idx** (*np.uint8*) – SRS bandwidth configuration idndex. 0 - 63.
- **bandwidth_idx** (*np.uint8*) – SRS bandwidth index. 0 - 3.
- **comb_offset** (*np.uint8*) – SRS comb offset. 0 - 3.
- **cyclic_shift** (*np.uint8*) – Cyclic shift. 0 - 11.
- **frequency_position** (*np.uint8*) – Frequency domain position. 0 - 67.
- **frequency_shift** (*np.uint16*) – Frequency domain shift. 0 - 268.

- **frequency_hopping** (*np.uint8*) – Freuqnecy hopping options. 0 - 3.
- **resource_type** (*np.uint8*) – Type of SRS allocation. 0: Aperiodic. 1: Semi-persistent. 2: Periodic.
- **periodicity** (*np.uint16*) – SRS periodicity in slots. 0, 2, 3, 5, 8, 10, 16, 20, 32, 40, 64, 80, 160, 320, 640, 1280, 2560.
- **offset** (*np.uint16*) – Slot offset value. 0 - 2569.
- **group_or_sequence_hopping** (*np.uint8*) – Hopping configuration. 0: No hopping. 1: Group hopping. 2: Sequence hopping.
- **ch_est_buff_idx** (*np.uint16*) – Index of which buffer to store SRS estimates into.
- **srs_ant_port_to_ue_ant_map** (*np.ndarray*) – Mapping between SRS antenna ports and UE antennas in channel estimation buffer: Store estimates for SRS antenna port i in srs_ant_port_to_ue_ant_map[i].
- **prg_size** (*np.uint8*) – Number of PRBs per PRG.

class aerial.phy5g.algorithms.srs_channel_estimator.SrsReport

SRS output report.

This report is returned by the SRS channel estimator.

Parameters

- **to_est_micro_sec** (*np.float32*) – Time offset estimate in microseconds.
- **wideband_snr** (*np.float3*) – Wideband SNR.
- **wideband_noise_energy** (*np.float32*) – Wideband noise energy.
- **wideband_signal_energy** (*np.float32*) – Wideband signal energy.
- **wideband_sc_corr** (*np.complex64*) – Wideband subcarrier correlation.
- **wideband_cs_corr_ratio_db** (*np.float32*) –
- **wideband_cs_corr_use** (*np.float32*) –

- **wideband_cs_corr_not_use** (*np.float32*) –

class aerial.phy5g.algorithms.srs_channel_estimator.SrsChannelEstimator

SrsChannelEstimator class.

This class implements SRS channel sounding for 5G NR.

__init__(*chest_params=None*)

Initialize SrsChannelEstimator.

Parameters

chest_params (*dict*) – Dictionary of channel estimation filters and parameters. Set to None to use defaults.

Return type

None

estimate(*rx_data*, *num_srs_ues*, *num_srs_cells*, *num_prb_grps*, *start_prb_grp*, *srs_cell_prms*, *srs_ue_prms*)

Run SRS channel estimation.

Parameters

- **rx_data** (*np.ndarray*) – Input RX data, size num_subcarriers x num_srs_sym x num_rx_ant.
- **num_srs_ues** (*int*) – Number of UEs.
- **num_srs_cells** (*int*) – Number of SRS cells.
- **num_prb_grps** (*int*) – Number of PRB groups.
- **start_prb_grp** (*int*) – Start PRB group.
- **srs_cell_prms** (*List[SrsCellPrms]*) – List of SRS cell parameters, one per cell.
- **srs_ue_prms** (*List[UeSrsPrms]*) – List of UE SRS parameters, one per UE.

Returns

A tuple containing:

- *List[np.ndarray]*: A list of channel estimates, one per UE. The channel estimate is a num_prb_grps x num_rx_ant x num_tx_ant numpy array.
- *np.ndarray*: SNRs per RB per UE.
- *List[SrsReport]*: A list of SRS wideband statistics reports, one per UE.

Return type

List[np.ndarray], np.ndarray, List[SrsReport]