LuMP version 2.0 is in beta testing

2013 Aug 09

Usage

Setting Up the Environment

On the lofarXN computers at Effelsberg, using a bash shell, do

source /opt/lump/lump_2.0/SETUP.sh
The setup commands for (t)csh are not yet ready, but should become available soon (less than 1 week).

Note that the LuMP 2.0 compatible dspsr is not yet installed on the lofarXN recording computers at Effelsberg.

Quick Overview

From lofarb1, the following command works for recording LOFAR pulsar data in the LuMP1 format:

Basic_LuMP_Recorder.py --port=4346 --clock_speed=200 --beamlets_per_lane=122 --datadir=. --data_type_in=L_intComplex16_t --station_name=Ef --writer_type=LuMP1 --filename_base=test_out --physical_beamlet_array='[0:122]' --subband_array='[12:134]' --rcumode_array='[5]*122' --rightascension_array='[0.929337]*122' --declination_array='[0.952579]*122' --epoch_array='[J2000]*122' --sourcename_array='[B0329+54]*122' --duration=10

The important parts are:

- --port give the port numer to listen to
- --beamlets_per_lane sets the beamlets per lane coming out of the LOFAR station. This should be 61 for the 16 bit mode, 122 for the 8 bit mode, and 244 for the 4 bit mode
- --data_type_in specify the type of data coming out of the station. This should be L_intComplex32_t for the 16 bit mode, L_intComplex16_t for the 8 bit mode, and L_intComplex8_t for the 4 bit mode.
- --station_name which station is being recorded
- --writer_type specify the type of writer format to use. LuMP0 for the original LuMP format (one file per beamlet). LuMP1 for the new LuMP format (multiple beamlets per subband --- requires an updated dspsr to process).
- --filename_base base of the filenames that are generated for writing data
- --physical_beamlet_array the physical beamlets being recorded
- --subband_array the subbands used for the corresponding beamlets

- --rcumode_array the rcumodes for the corresponding beamlets (note that different rcumodes are possible for different beamlets at the same time)
- --rightascension_array the right ascension values (in radians) for the corresponding beamlets
- --declination_array the declination values (in radians) for the corresponding beamlets
- --epoch_array the epoch names (usually J2000 or SUN) for the corresponding beamlets
- --sourcename_array the name of the target source for each corresponding beamlet
- --duration the duration of the recording, in seconds

The start time may be specified with

- --start_date in the format YYYY-MM-DDTHH:MM:SSZ

Detailed Option Help

```
Basic_LuMP_Recorder.py --help
usage: Basic_LuMP_Recorder.py [-h] [--start_date START_DATE] --duration
                              DURATION --port PORT --station_name STATION_NAME
                              --datadir DATADIR [--data_type_in DATA_TYPE_IN]
                              [--clock_speed {200,160,0}]
                              [--beamlets_per_lane BEAMLETS_PER_LANE]
                              [--samples_per_packet SAMPLES_PER_PACKET]
                              [--main_recorder_logfile MAIN_RECORDER_LOGFILE]
                              [--recorder_num_cores RECORDER_NUM_CORES]
                              [--recorder_cache_size RECORDER_CACHE_SIZE]
                              [--recorder_ram_size RECORDER_RAM_SIZE]
                              [--writer_type WRITER_TYPE] --filename_base
                              FILENAME_BASE --physical_beamlet_array
                              PHYSICAL_BEAMLET_ARRAY --subband_array
                              SUBBAND_ARRAY --rcumode_array RCUMODE_ARRAY
                              --rightascension_array RIGHTASCENSION_ARRAY
                              --declination_array DECLINATION_ARRAY
                              --epoch_array EPOCH_ARRAY --sourcename_array
                              SOURCENAME_ARRAY
                              [--data_type_process DATA_TYPE_PROCESS]
                              [--data_type_out DATA_TYPE_OUT]
                              [--num_output_channels NUM_OUTPUT_CHANNELS]
                              [--num_polyphase_filter_taps NUM_POLYPHASE_FILTER_TAPS]
                              [--window_function WINDOW_FUNCTION]
                              [--window_parameter WINDOW_PARAMETER]
                              [--integration_time INTEGRATION_TIME]
                              [--scale_by_inverse_samples {0,1}]
                              [--extra_scale_factor EXTRA_SCALE_FACTOR]
                              [--bounds_check_output {0,1}]
                              [--extra_string_option_0 EXTRA_STRING_OPTION_0]
                              [--extra_string_option_1 EXTRA_STRING_OPTION_1]
                              [--extra_string_option_2 EXTRA_STRING_OPTION_2]
                              [--extra_string_option_3 EXTRA_STRING_OPTION_3]
```

```
                [--extra_string_option_4 EXTRA_STRING_OPTION_4]
                [--verbose] [--echo_only] [--stdin] [--version]

Python program to run LOFAR_Station_Beamformed_Recorder.py for basic LuMP
output mode operation on a single LOFAR recording computer.

optional arguments:
 -h, --help          show this help message and exit
 --start_date START_DATE
                *OPTIONAL* The date and time to begin recording, as a
                UTC ISO date string of the format YYYY-MM-DDTHH:MM:SSZ
                or as a hexadecimal number representing the integer
                Unix timestamp in seconds since the reference epoch
                1970-01-01 00:00:00 +0000 (UTC).
 --duration DURATION   *REQUIRED* Duration of measurment to listen to the
                station, in seconds. Note that this is the duration
                from the start_date, so if recording begins late, the
                actual recorded duration will be smaller.
 --port PORT         *REQUIRED* Port number to listen to for incoming data
                from a LOFAR station. Normally, this should be a
                decimal number. When reading from a file (raw UDP
                dump), this should be the filename, including any
                necessary path, of the input file, preceded by FILE:.
                For example, if your filename is
                ./MYDIR/somedir/myfile.raw then you would specify this
                as --port=FILE:./MYDIR/somedir/myfile.raw in the
                argument list. If the port specification starts with
                UDP: then UDP network data are read from the port
                number following the UDP: key. TCP: specifies that the
                TCP protocol is to be used. The value - specifies that
                the program should read from stdin. By default, the
                program will use UDP access.
 --station_name STATION_NAME
                *REQUIRED* Name of the LOFAR station to record data
                from. This should be of the type DE601, Ef, or EfDE601
 --datadir DATADIR     *REQUIRED* Name of the directory of the main data
                recording area into which this recording will be
                written. For example, suppose that the main data
                recording area is '/media/scratch/observer', your name
                is 'Astronomer', and you are observing on 2010 Dec 25.
                You want all of the data to be recorded to your own
                specific directory area, to not be confused with other
                people's data, and you want to sort things by the date
                of observation. Then you would set --datadir to
                '/media/scratch/observer/Astronomer/20101225'. A
```

relative path name may be specified. The datadir '.' may also be specified.

--data_type_in DATA_TYPE_IN

OPTIONAL data type of station beamforemd data. Defaults to 26. Available options are: 7=L_int8_t (8 bit integer), 10=L_int16_t (16 bit integer), 11=L_int32_t (32 bit integer), 12=L_int64_t (64 bit integer), 14=L_Real16_t (half precision floating point), 15=L_Real32_t (single precision floating point), 16=L_Real64_t (double precision floating point), 17=L_Real80_t (80 bit extended precision precision floating point), 18=L_Real128_t (quad precision floating point), 19=L_Complex32_t (half precision complex floating point, two L_Real16_t values), 20=L_Complex64_t (single precision complex floating point, two L_Real32_t values), 21=L_Complex128_t (double precision complex floating point, two L_Real64_t values), 22=L_Complex160_t (extended precision complex floating point, two L_Real80_t values), 23=L_Complex256_t (quad precision complex floating point, two L_Real128_t values), 24=L_intComplex8_t (complex integer, two L_int4_t values, LOFAR 4 bit mode), 25=L_intComplex16_t (complex integer, two L_int8_t values, LOFAR 8 bit mode), 26=L_intComplex32_t (complex integer, two L_int16_t values, LOFAR 16 bit mode), 27=L_intComplex64_t (complex integer, two L_int32_t values), 27=L_intComplex128_t (complex integer, two L_int64_t values)

--clock_speed {200,160,0}

Clock speed of station, in MHz. Defaults to 200

--beamlets_per_lane BEAMLETS_PER_LANE

OPTIONAL number of beamlets per RSP lane sent out by the station. Defaults to 61

--samples_per_packet SAMPLES_PER_PACKET

OPTIONAL number of samples per packet sent out by the station. Defaults to 16

--main_recorder_logfile MAIN_RECORDER_LOGFILE

Name of the logfile to write out for the main recorder program. Defaults to LOFAR_Station_Beamformed_Recorder.log

--recorder_num_cores RECORDER_NUM_CORES

OPTIONAL number of CPU cores to use for LuMP recording. If 0 is specified, the software will attempt to determine the number of cores available on

the recording computer and use all of them. Also note that this only specifies the number of cores used by the LuMP recording software itself --- CPU utilization by downstream software that may be started by LuMP (dspsr, for example) must be specified in the option arguments to that software separately. Defaults to 0

--recorder_cache_size RECORDER_CACHE_SIZE

OPTIONAL size of the CPU cache, in bytes. This is the size of the full cache per CPU (typically L3 cache), as reported by the 'cache size' listing in /proc/cpuinfo. If specified as 0, LuMP will attempt to determine this information automatically. Defaults to 0

--recorder_ram_size RECORDER_RAM_SIZE

OPTIONAL size of the RAM available for LuMP to use, in bytes. If specified as 0, LuMP will attempt to determine the amount of RAM available on the computer and assume it can use all of that. Default 0

--writer_type WRITER_TYPE

OPTIONAL The enum code of the writer type to use. Defaults to LuMP0. Available options are: 1=RAW (raw voltages, separate files for each beamlet, for ALL beamlets from the RSP board, for each polarization) @@single_thread, 2=RAW0 (raw voltages, one data file with ALL beamlets from the RSP board, all polarizations) @@single_thread, 3=RAW1 (raw voltages, separate files for each beamlet, for selected beamlets from the RSP boardfor each polarization) @@multi_thread, 5=POWER0 (power measuremnts integrated over time, one data file containing the selected beamlets and full polarization information) @@multi_thread, 6=LuMP0 (raw voltage data for selected beamlets in the LuMP output format, full polarization information) @@multi_thread, 7=FFT0 (channelized voltage data using an FFT, single data file for selected beamlets, full polarization) @@multi_thread, 8=PFB0 (channelized voltage data using a polyphase filterbank, single data file for selected beamlets, full polarization) @@multi_thread, 9=POWER_FFT0 (channelized power measurements integrated over time using an FFT, single data file for selected beamlets, full polarization) @@multi_thread, 10=POWER_PFB0 (channelized power measurements integrated over time using a polyphase filterbank, single data file for selected beamlets, full polarization) @@multi_thread,

11=LuMP1 (raw voltage data for selected beamlets in the LuMP output format, full polarization information, single output file for all selected beamlets) @@single_thread, 12=VDIF0 (raw voltage data in the VDIF 2 format, a single data file is written for all selected beamlets, with different threads for different beamlets) @@single_thread,

--filename_base FILENAME_BASE

REQUIRED base string from which the output file names will be generated. Note that this base filename will be extended by a 2 digit hexadecimal number indicating the writer ID number used to write out the data (filename_base="%s.%2.2X"%(filename_base,ID)).

--physical_beamlet_array PHYSICAL_BEAMLET_ARRAY

REQUIRED Python-like array of the physical beamlets to use for this writer. A combination of individual physical beamlets and Python ranges may be used, such as '[0,1,4,7,10:31,60:62]'. Ranges must have both start and end specified as start:end. Note that the notation here is a Python notation for the ranges (start, start+1,start+2,...,end-1), which is different from the ASTRON LOFAR station software.

--subband_array SUBBAND_ARRAY

REQUIRED Python array of the subbands corresponding to the beamlets. A combination of individual physical subbands and Python ranges may be used, such as '[100,101,104,107,110:131,160:162]'. Ranges must have both start and end specified as start:end. Note that the notation here is a Python notation for the ranges (start, start+1,start+2,...,end-1), which is different from the ASTRON LOFAR station software. Python-style array multipliers may be used to repeat subband selections, such as when multiple pointing directions use the same observing frequencies. For example, '[0:2]*3' is equivalent to '[0,1,0,1,0,1]'. The PHYSICAL_BEAMLET_ARRAY and SUBBAND_ARRAY should match beamlet to subband at the same index.

--rcumode_array RCUMODE_ARRAY

REQUIRED Python array of the RCUMODEs corresponding to the beamlets. Python-style array multipliers are allowed, simplifying the standard case where all beamlets have the same RCUMODE. For example, '[5]*244' yields an array of RCUMODE values that is 244 elements long, all with RCUMODE==5. Alternatively, individual RCUMODE values may be specified in the standard Python

array syntax, such as '[5,5,5,5,6,6,7,7]'. Ranges may also be specified as start:end. Note that the notation here is a Python notation for the ranges (start, start+1,start+2,...,end-1), which is different from the ASTRON LOFAR station software. The PHYSICAL_BEAMLET_ARRAY and RCUMODE_ARRAY should match beamlet to RCUMODE at the same index.

--rightascension_array RIGHTASCENSION_ARRAY

REQUIRED Python array of the right ascensions (or other coordinate if the Epoch is not J2000) corresponding to the beamlets. *Note that the right ascension is to be provided in units of radians, as it is specified to the LOFAR beamctl program.* Python-style array multipliers are allowed, simplifying the standard case where all beamlets have the same pointing direction. For example, '[1.23456789]*244' yields an array of right ascension values that is 244 elements long, all with rightascension==1.23456789. Alternatively, individual right ascension values may be specified in the standard Python array syntax, such as '[1,1,1,1,2,2,3,3]'. The PHYSICAL_BEAMLET_ARRAY and RIGHTASCENSION_ARRAY should match beamlet to right ascension at the same index.

--declination_array DECLINATION_ARRAY

REQUIRED Python array of the declinations (or other coordinate if the Epoch is not J2000) corresponding to the beamlets. *Note that the declination is to be provided in units of radians, as it is specified to the LOFAR beamctl program.* Python-style array multipliers are allowed, simplifying the standard case where all beamlets have the same pointing direction. For example, '[1.23456789]*244' yields an array of declination values that is 244 elements long, all with declination==1.23456789. Alternatively, individual declination values may be specified in the standard Python array syntax, such as '[0,0,0,0,0.5,0.5,1.0,1.0]'. The PHYSICAL_BEAMLET_ARRAY and DECLINATION_ARRAY should match beamlet to declination at the same index.

--epoch_array EPOCH_ARRAY

REQUIRED Python array of the epochs (or other coordinate system identifiers) corresponding to the beamlets. Python-style array multipliers are allowed, simplifying the standard case where all beamlets have the same pointing epoch. For example, '[J2000]*244'

yields an array of epoch values that is 244 elements long, all with epoch==J2000. Note that the epoch values do not require string quotation marks. Alternatively, individual epoch values may be specified in the standard Python array syntax, such as '[J2000,J2000,HADEC, AZELGEO, SUN,MOON]'. The PHYSICAL_BEAMLET_ARRAY and EPOCH_ARRAY should match beamlet to epoch at the same index.

--sourcename_array SOURCENAME_ARRAY

REQUIRED Python array of the source names corresponding to the beamlets. Python-style array multipliers are allowed, simplifying the standard case where all beamlets have the same source name. For example, '[Cas A]*244' yields an array of epoch values that is 244 elements long, all with sourcename==Cas A. Note that the source name values do not require string quotation marks. Alternatively, individual source name values may be specified in the standard Python array syntax, such as '[Cas A, Cas A, Cyg A, Cyg A, Hydra A]'. Note that leading and trailing whitespace will be removed. The PHYSICAL_BEAMLET_ARRAY and SOURCENMAE_ARRAY should match beamlet to source name at the same index.

--data_type_process DATA_TYPE_PROCESS

OPTIONAL Data type for internal processing. Defaults to L_intComplex32_t. Available options are: 7=L_int8_t (8 bit integer), 10=L_int16_t (16 bit integer), 11=L_int32_t (32 bit integer), 12=L_int64_t (64 bit integer), 14=L_Real16_t (half precision floating point), 15=L_Real32_t (single precision floating point), 16=L_Real64_t (double precision floating point), 17=L_Real80_t (80 bit extended precision precision floating point), 18=L_Real128_t (quad precision floating point), 19=L_Complex32_t (half precision complex floating point, two L_Real16_t values), 20=L_Complex64_t (single precision complex floating point, two L_Real32_t values), 21=L_Complex128_t (double precision complex floating point, two L_Real64_t values), 22=L_Complex160_t (extended precision complex floating point, two L_Real80_t values), 23=L_Complex256_t (quad precision complex floating point, two L_Real128_t values), 24=L_intComplex8_t (complex integer, two L_int4_t values, LOFAR 4 bit mode), 25=L_intComplex16_t (complex integer, two L_int8_t values, LOFAR 8 bit

mode), 26=L_intComplex32_t (complex integer, two L_int16_t values, LOFAR 16 bit mode), 27=L_intComplex64_t (complex integer, two L_int32_t values), 27=L_intComplex128_t (complex integer, two L_int64_t values)

--data_type_out DATA_TYPE_OUT

OPTIONAL Data type for output to disk. Defaults to L_intComplex32_t. Available options are: 7=L_int8_t (8 bit integer), 10=L_int16_t (16 bit integer), 11=L_int32_t (32 bit integer), 12=L_int64_t (64 bit integer), 14=L_Real16_t (half precision floating point), 15=L_Real32_t (single precision floating point), 16=L_Real64_t (double precision floating point), 17=L_Real80_t (80 bit extended precision precision floating point), 18=L_Real128_t (quad precision floating point), 19=L_Complex32_t (half precision complex floating point, two L_Real16_t values), 20=L_Complex64_t (single precision complex floating point, two L_Real32_t values), 21=L_Complex128_t (double precision complex floating point, two L_Real64_t values), 22=L_Complex160_t (extended precision complex floating point, two L_Real80_t values), 23=L_Complex256_t (quad precision complex floating point, two L_Real128_t values), 24=L_intComplex8_t (complex integer, two L_int4_t values, LOFAR 4 bit mode), 25=L_intComplex16_t (complex integer, two L_int8_t values, LOFAR 8 bit mode), 26=L_intComplex32_t (complex integer, two L_int16_t values, LOFAR 16 bit mode), 27=L_intComplex64_t (complex integer, two L_int32_t values), 27=L_intComplex128_t (complex integer, two L_int64_t values)

--num_output_channels NUM_OUTPUT_CHANNELS

OPTIONAL Number of output channels to make per subband. Default is 1

--num_polyphase_filter_taps NUM_POLYPHASE_FILTER_TAPS

OPTIONAL Number of polyphase filter taps to use. Default is 1

--window_function WINDOW_FUNCTION

OPTIONAL Type of window function to use. Defaults to 0. Available options are: 0=Rectangular 1=Hann 2=Hamming 3=Tukey 4=Cosine 5=Lanczos 6=Barlett0 7=BarlettN0 8=Gaussian 9=Bartlett_Hann 10=Blackman 11=Kaiser 12=Nuttall 13=Blackman_Harris 14=Blackman_Nuttall 15=Flat_Top

--window_parameter WINDOW_PARAMETER
OPTIONAL Extra parameter for specific window functions. Defaults to 0.000000E+00.
--integration_time INTEGRATION_TIME
OPTIONAL The integration time, in seconds, for averaging the total power data. Default=1.00E+00
--scale_by_inverse_samples {0,1}
OPTIONAL Should the total power values should be scaled by the number of samples per integration? 0 No, or 1 Yes.
--extra_scale_factor EXTRA_SCALE_FACTOR
OPTIONAL Extra scaling factor for total power. Default is 1.00
--bounds_check_output {0,1}
OPTIONAL Should the software do bounds checking when converting data types? 0 No, or 1 Yes. This is most useful only for integer output types, whre the default is a simple truncation of bits.
--extra_string_option_0 EXTRA_STRING_OPTION_0
OPTIONAL Extra string option for controlling some writers. Default is ''
--extra_string_option_1 EXTRA_STRING_OPTION_1
OPTIONAL Extra string option for controlling some writers. Default is ''
--extra_string_option_2 EXTRA_STRING_OPTION_2
OPTIONAL Extra string option for controlling some writers. Default is ''
--extra_string_option_3 EXTRA_STRING_OPTION_3
OPTIONAL Extra string option for controlling some writers. Default is ''
--extra_string_option_4 EXTRA_STRING_OPTION_4
OPTIONAL Extra string option for controlling some writers. Default is ''
--verbose, -v write commands to screen as well as executing
--echo_only Only show the commands that would be processed, do not actually run them
--stdin Read the arguments to the program from stdin instead of from the command line. If this option is present, it must be the only option on the command line provided, and all regular options must be passed in via stdin.
--version, -V Print the version number of this software and exit.

See the accompanying manual for more information.