Package 'LPI'

May 16, 2025

Version 0.4-0

Date 2025-05-16

Title Lag Profile Inversion

Author Ilkka Virtanen <ilkka.i.virtanen@oulu.fi>

Maintainer Ilkka Virtanen <ilkka.i.virtanen@oulu.fi>

Description Lag Profile Inversion

Depends R (>= 2.14.0), snow, parallel, parallelly, future

Suggests rlips

R topics documented:

`LPI-package` *Lag Profile Inversion*

Description

Incoherent scatter radar lag profile deconvolution by means of statistical inversion

Details

Package:	LPI
Version:	0.4
Date:	2025-05-16
License:	FreeBSD
Depends:	R ()
Built:	

Index:

Author(s)

Ilkka Virtanen (University of Oulu, Finland)
<ilkka.i.virtanen@oulu.fi>

LPI *LPI*

Description

Lag Profile Inversion

Usage

```
LPI( dataInputFunction,
inputPackages=c(),
startTime = 0,
stopTime = 4000000000,
nup = LPIexpand.input( 1 ),
filterLength = LPIexpand.input( 1 ),
decodingFilter = "none",
lagLimits = c(1,2),
rangeLimits = c(1,2),
maxRanges = Inf,
maxClutterRange = 0,
clutterFraction = 1,
timeRes.s = 10,
backgroundEstimate=TRUE,
maxWait.s = -1,
freqOffset = LPIexpand.input( 0 ),
indexShifts = LPIexpand.input( list(c(0,0)) ),
solver = "fishsr",
nBuf = 10000,
fullCovar = FALSE,
```

```
rlips.options = list( type="c" , nbuf=1000 , workgroup.size=128),
remoteRX = FALSE,
normTX = FALSE,
nCode = NA,
ambInterp = FALSE,
resultDir = paste(format(Sys.time(),"%Y-%m-%d_%H:%M"),'LP',sep='_'),
dataEndTimeFunction="currentTimes",
resultSaveFunction = "LPIsaveACF",
paramUpdateFunction="noUpdate" ,
cl=NULL ,
nCores=NULL ,... )
```

Arguments

dataInputFunction	Raw data input function name as a string. The function will get a list of all LPI input arguments as input and it must return a valid list of raw data vectors.
inputPackages	Additional package(s) that contain functions for raw data input, as strings.
startTime	Analysis start time in POSIX format. Default: 0 (1st Jan 1970 00:00 UT)
stopTime	Analysis end time in POSIX format. Default: 4000000000 (2nd Oct 2096 07:00 UT)
timeRes.s	Analysis time resolution (integration time) in seconds. Default: 10
maxWait.s	Maximum time to wait for new data before stopping the analysis, in seconds. Default: -1 (Stop immediately at end of data)
freqOffset	Frequency offsets from baseband. A named vector with elements "RX1", "RX2", "TX1", "TX2" or anything that can be converted into this format with LPIexpand.input. Default: 0.0
nup	Upsampling factors. The data are upsampled by factor 'nup' before applying boxcar filter of length 'filterLength'. A named vector with elements "RX1", "RX2", "TX1", "TX2" or anything that can be converted into this format with LPIexpand.input. Default: 1
filterLength	Length of the boxcar-shaped post-detection filters. The actual filter lengths are filterLength / nup. A named vector with elements "RX1", "RX2", "TX1", "TX2" or anything that can be converted into this format by the function LPIexpand.input. The ratio filterLength / nup must be identical for all four data vectors. Default: 1
lagLimits	Limits of time-lag gates. Default: c(1,2) (The first lag)
rangeLimits	Limits of range gates. Default: c(1,2) (One range gate very close to ground)

`maxRanges` Maximum range for each lag profile. Allows the number of range gates to be limited at selected lags. See details.

Default: Inf

`nCode` Number of pulses in a full modulation cycle. If positive, the value is used for lag profile pre-averaging before the actual inversion. Use NA to dispable the averaging.

Default: NA

`backgroundEstimate`

Logical, if TRUE, an additional unknown is modeled in each lag profile in order to absorb time-stationary background noise.

Default: TRUE

`fullCovar` If TRUE, full covariance matrices of each lag profile are stored, if FALSE, only variances are stored.

Default: FALSE

`decodingFilter` Amplitude domain decoding filter selection, accepted values are a complex vector of filter coefficients and character strings "none", "matched", and "inverse". See details.

Default: "none"

`maxClutterRange`

Maximum range at which ground clutter will be subtracted. Set below min(rangeLimits) in to disable the clutter suppression. See details.

Default: 0

`clutterFraction`

The fraction of a full integration period used for the coherent ground clutter profile estimation. A float from interval (0,1]. Values smaller than 1 should be used carefully.

Default: 1.0

`remoteRX` Logical, if TRUE, all receiver samples flagged as usable by 'dataInputFunction' are used in analysis. If FALSE, only those lagged products whose range ambiguity function is zero in the range interval [0 min(rangeLimits.km)] are used. Should usually be FALSE in monostatic operation and TRUE in bistatic measurements. Notice that zero range ambiguity requirement does not completely cut off signal from small ranges, use indexShifts for this purpose.

Default: FALSE

`ambInterp` Logical, if TRUE, the range ambiguity functions are calculated from transmitter samples that are oversampled by factor 11. The oversampling is performed by means of linear interpolation of decimated transmitter samples. If FALSE, the range ambiguity functions are calculated as simple products of the decimated data.

Default: FALSE

`'minNpower'` Minimum number of samples to average in power profile calculation. The average power profile is used for error estimation. In data points with the number of averaged samples less than minNpower, the average over all data points is used.

Default: 100

'noiseSpikeThreshold'
: Threshold for noise spike detection. Data points with squared amplitude larger than noiseSpikeThreshold times the corresponding average power are rejected.

  Default: 10

indexShifts
: Additional adjustements to TX and RX indices, see details.

  Default: list(TX1=c(0,0) , TX2=c(0,0) , RX1=c(0,0) , RX2=c(0,0))

normTX
: Logical, if TRUE, the transmitter sample amplitudes are normalized to their mean value after filtering and decimation. May be useful with low-quality transmitter samples, but must be used with care.

  Default: FALSE

solver
: Inverse problem solver selection, accepted values are "fishsr", "decor", "dummy", "rlips", "fishs", "deco", and "ffts". See details.

  Default: "fishsr"

nBuf
: Number of theory matrix rows to buffer before calling the solver function.

  Default: 10000

rlips.options
: Additional options to the 'rlips' solver. See rlips help for details.

  Default: list(type="c" , nbuf=1000 , workgroup.size=128)

resultDir
: Output directory. Use non-character value (e.g NA) if 'resultSaveFunction' does not write to files.

  Default: paste(format(Sys.time(),"

resultSaveFunction
: Function that is used for saving the analysis results. The function gets as input the full LPI input argument list ('LPIparam'), the integration period number ('intPeriod'), and the resolved ACF ('ACF'). The function may ouput anything but the output will not be used anywhere.

  The resolved ACF list (argument 'ACF') has the following components:

  'range' A vector of range gate centres

  'lag' A vector of lag gate centres

  'ACF' length(range) x length(lag) complex lag profile matrix

  'var' length(range) x length(lag) real matrix of lag profile variances

  'backgroundACF' length(lag) complex vector of background ACF

  'backgroundvar' length(lag) real vector of background ACF variances

  'covariance' (length(range) + 1) x (length(range) + 1) x length(lag) complex array of lag profile covariances, or NULL if LPIparam$fullCovar=FALSE. The extra elements are from the additional background ACF estimates that are added to the end of each lag profile.

  The integration period number (argument 'intPeriod') starts from one for the first period, and is incremented by one for each successive period.

  The LPI input argument list (argument 'LPIparam') contains all LPI input arguments, including an expansion of the optional ones ('...'). Any information needed in the resultSaveFunction can thus be passed via the '...' argument of LPI.

  Default: LPIsaveACF

`dataEndTimeFunction`

Function that returns sampling times of last available sample in each data vector. The function will get the full LPI input argument list as argument and it must return a named numeric vector with elements "RX1", "RX2", "TX1", and "TX2".

Default: currentTimes (5s ago)

`paramUpdateFunction`

An optional function for updating the parameter list for each integration period. The function gets the full LPI input argument list as input and must return a valid LPI parameter list. Within each integration period, the analysis is continued until NULL is returned by paramUpdateFunction, subsequent calls to paramUpdateFunction will get the *modified* LPIparam from the previous call as input.

Default: "noUpdate"

`...`

Additional arguments to be collected in the LPI parameter list. All input arguments of LPI are collected in an "LPI parameter list", which is passed to 'dataInputFunction', 'dataEndTimeFunction', 'resultSaveFunction', and 'paramUpdateFunction'.

Details

'clusterNodes' Definition of the computer cluster. Following inputs are accepted:

1. clusterNodes = NA No parallelism, everything is run sequentially.
2. clusterNodes = n, n = 1, 2, 3, ... Run n integration periods in parallel on the local computer.
3. clusterNodes = list(computer1=n1 , computer2=n2, ... , computerM=nM) Run M integration periods in parallel in machines "computer1", "computer2", ... The lag profiles in each integration period are divided in between n1 processes in computer1, in between n2 processes in computer2, etc. The list may also contain an entry "localControl", if localControl=FALSE the data is read from files at the remote computers, otherwise at the local machine and then transferred to other computers. If localControl=FALSE, all computers in the clusterNodes list must have exactly identical directory hierarchies. This option requires passwordless ssh connection in between the machines computer1, computer2, etc., i.e. generation of ssh keys.

As an example, the default cluster definition is list(tesla1=8 , tesla2=8 , tesla3=8 , tesla4=8 , tesla5=8) and it is used in a computer cluster that contains computers tesla1 ... tesla5, each of which has 8 cores. The analysis first starts length(clusterNodes) control nodes in the same machine where the main LPI process is running. Each of these control nodes then connects to nodes running in the remote computers, in this case to tesla1 ... tesla5. Finally, each of the remote nodes connects to a number of 'calculation slave' nodes running in their own local computers. In the default case, 8 nodes would be running on each computer. The main process that the user controls from the command line generates input argument lists for length(clusterNodes) integration periods. Each local control node receives one set of arguments, reads in the corresponding raw data, performs necessary filtering, decimation, etc. and passes the data to the remote computers. In each remote node, the data is copied to all calculation slaves. Each remote node then requests its calculation nodes to solve lag profiles until all time lags are solved. The remote nodes then collect the profiles together and send the result back to the control nodes, which call 'resultSaveFunction' and signal the main process about a completed job. When all control nodes are done, a new set of integration periods is selected. In principle, a remote node could have computing slaves in computers other than its own as

well. This is done simply by replacing the number of computing slaves with a list of computer names.

'lagLimits' A vector of time lag limits, lagLimits = c(l1, l2, l3, ... , lN). The analysis integrates lags l1, l1+1, ... , l2-1 (of decimated data) into lag profile 1, lags l2, l2+1, ... , l3-1 into lag profile 2, etc. The steps in lagLimits do not need to be uniform, it is possible to use coarser lag resolution at longer lags by selecting e.g. lagLimits = c(1,2,3,5,7,9,11).

'maxRanges' Allows one to reduce the number of range-gates at selected lags. If the vector is shorter than lagLimits its last value is repeated as necessary. Defaults to Inf, which means that all lags are solved at all range gates defined in rangeLimits.

maxRanges may be used in combined D/E/F-region experiments when correlations longer than certain limit are known to exist only in D-region. As an example, in order to solve first 30 lags at all range gates, but the longer ones only below 100 samplese range, one can use maxRanges = c(rep(Inf , 30) , 100).

Another use case is a dedicated D region experiment that makes use of voltage level decoding. Most time lags will then have zero range ambiguity functions and can be safely skipped. Assuming that modulation bit length is 1 decimated sample and inter-pulse period is 1000 samples, one could use e.g. the following combination: decodingFilter='matched', lagLimits=c(1000,1001,2000,2001,3000,3001), maxRanges=c(Inf,0,Inf,0,Inf) which would solve three pulse-to-pulse lags and save the analysis from the work of inspecting all time lags (only 1 in 1000 of them can actually be measured).

'endTime' Analysis end time in POSIX format. The analysis will be stopped without reaching endTime if all existing data is analysed and the process has waited maxWait.s seconds for new data.

'freqOffset' Frequency offset from baseband to the signal centre frequency. Currently only one frequency per data vector, i.e. either a single offset for all data, or different shifts for "RX1", "RX2", "TX1", and "TX2". The frequencies are not in SI units, but time is measured in decimated sample intervals and the frequecies should be scaled accordingly.

'indexShifts' Additional adjustments to the TX (and) RX indices returned by 'dataInputFunction'. The user input is converted into a list with entries "RX1", "RX2", "TX1", and "TX2". Each entry is a two-element vector c(shift1, shift2) where, in case of TX, shift1 is the adjustment at rising edge of each pulse, and shift2 is the adjustment at falling edge of each pulse. For RX, shift1 is the adjustment at each start of reception, and shift2 at end of reception. All shifts are positive forwards, use negative values to adjust towards earlier times. As an example, if the recorded TX bit for "TX1" starts 10 samples *before* the actual pulse, and ends 20 samples *after* the actual end of the pulse, one should have indexShifts["TX1"] == c(10,-20).

'decodingFilter' Voltage level coherent decoding with given filter coefficients or by means of matched or inverse filtering using measured samples of transmitted waveform. The decoding is performed for a single inter-pulse period at a time. Combining the "matched" or "inverse" filters with range gates extending above the range corresponding the shortest inter-pulse period will thus usually lead to meaningless results. With the "matched" and "inverse" filters data index vectors are modified in the decoding as if the decoding would perfectly suppress code sidelobes. Setting decodingFilter="matched" will thus generally generate range sidelobes. The "matched" and "inverse" filters are intended to be used for D-region pulse-to-pulse correlations together with solver="dummy", which enables fast calculation of lag profiles when high lag resolution is not required. More complicated decoding filters that gnerally require specific information about the modulation should be implemented in dataInputFunction

'solver' LPI allows the user to select an inverse problem solver that best suits for the problem at hand from among the following options:

1. "rlips", R Linear Inverse Problem Solver, is an R package for solving large linear inverse problems. The software exploits GPUs providing massive parallelism. The solver is most suitable for solving large problems (large number of range gates). The rlips package must be installed in all computers of the analysis cluster.

2. "fishs" is a simple inverse problem solver provided as an integral part of the LPI package. It can be used as an CPU-based alternative for "rlips". Relative speed in between "rlips" and "fishs" depends on several factors, such as size of the inverse problem and the computer hardware.

3. "deco" is a modification of "fishs" that essentially performs matched filter decoding of lag profiles. It can thus be used as a faster alternative when alternating codes or long cycles of random codes are used as transmitter modulation.

4. "ffts" is a fast solver that gains its speed from exploiting FFT. The FFT-based solution is not reliable if the received signal is not continuous. The solver is thus mainly intended for bistatic operations. Background noise subtraction cannot be combined with "ffts".

5. "dummy" is a dummy solver that calculates simple averages of lag profiles without actually decoding them. The solver is intended to be used together with voltage level decoding in D-region measurements.

'inputPackages' A list of packages that contain the data input functions. LPI does not need to be listed. The packages must be installed on all computers of the analysis cluster.

'dataInputFunction' Name of a function that returns one integration period of a raw voltage level data. The package containing this function and all functions that this function calls must be installed in all computers of the analysis cluster and listed in 'inputPackages'. All LPI input arguments are collected in a list and passed to this function.

The function is called by means of

eval(as.name(LPIparam[["dataInputFunction"]]))(LPIparam , intPeriod)

and it must return a list with elements "RX1", "RX2", "TX1", "TX2", and "success". The first four elements are lists themselves, consisting of a complex data vector 'cdata', a logical "index vector" 'idata', and integer "ndata". Denoting complex samples with 'cn' and logical samples with 'in' the output list is of the form

list(RX1=list(cdata=c(c1,c2,c3,...,cn1) , idata=c(i1,i2,i3,...,in1) , ndata=n1), RX2=list(...), TX1=list(...), TX2=list(...), success=TRUE/FALSE)

where '...' denotes a set of vectors similar to that in 'RX1'. The input argument 'intPeriod' is the integration period, counted in steps of 'timeRes.s' from 'startTime'. The period starting exactly at 'startTime' is period number 1.

If success=FALSE in the output list the analysis will completely skip the integration period. E.g. in case of missing data one can ignore the other elements and simply return list(success=FALSE). The integration period will be skipped also if any of the 'idata' vectors has only FALSE elements.

The input argument list 'LPIparam' is the full LPI input argument list, containing also the additional arguments, with the following modifications:

1. The entries 'nup', 'filterLength', 'maxClutterRange', 'clutterFraction', 'freqOffset', and 'indexShifts' are expanded to the LPI internal format with named elements "RX1", "RX2", "TX1", and "TX2" using LPIexpand.input.

2. Storage mode of 'nup', 'filterLength', 'lagLimits', 'rangeLimits', 'maxClutterRange', 'indexShifts', and 'nCode' is set to "integer".

3. Elements 'lastIntPeriod', 'iscluster', 'dataEndTimes', and 'maxIntPeriod' are added. 'dataEndTimes' is the return value of 'dataEndTimeFunction' and 'maxIntPeriod' is the corresponding integration period number. 'lastIntPeriod' is the integration period number corresponding 'stopTime'. The main analysis loop will update 'dataEndTimes' and 'maxIntPeriod' repeatedly. In addition, the optional 'paramUpdateFunction' may modify any other element of the list as well.

All additional arguments to LPI are collected as is to the parameter list.

'dataEndTimeFunction' Name of a function that returns sampling times of latest recorded samples. The function gets a list of all LPI input arguments as input and must return a named vector of POSIX times ("TX1", "TX2", "RX1", and "RX2"). The function should be reasonably fast, because it is called regularly.

Default: "currentTimes"

'cl' A cluster environment returned e.g. by getMPIcluster. Integration periods are divided between the cluster nodes. Each cluster runs nCores time lags in parallel. Use NULL for sequential analysis.

Default: NULL

'nCores' Number of cores available to each cluster nodes. If NULL, the number of cores is queried by parallelly::availableCores.

Default: NULL

Because LPI does not have its own I/O routines it cannot actually check availability of data, but it assumes that all integration periods from 'startTime' to 'dataEndTimes' are available. It will stop only if everything from 'startTime' to 'stopTime' is already analysed, or if everything from 'startTime' to 'dataEndTimes' is analysed and 'dataEndTimes' has not progressed despite waiting for 'maxWait.s' seconds. In real-time analysis it is extremely important that 'dataEndTimes' is never ahead of the actual sampling time of the latest sample that is available for 'dataInputFunction'.

Author(s)

Ilkka Virtanen (University of Oulu, Finland)
<ilkka.i.virtanen@oulu.fi>

`LPIexpand.input` *LPIexpand.input*

Description

Expand vectors or lists to the LPI internal format with names "RX1", "RX2", "TX1", and "TX2"

Usage

```
LPIexpand.input( parvec)
```

Arguments

parvec	A vector or list. See details.

Details

The input may be in several formats:

1. If parvec is readily in the internal format it is returned as such.

2. If parvec does not have named elements, it is repeated / truncated to length 4 and the elements are named in order "RX1", "RX2", "TX1", "TX2".

3. If parvec does not contain all the internally used names, but it contains also unnamed elements, the unnamed ones are repeated and named to fill the output.

4. Names 'RX' and 'TX' are expanded in an obvious way.

5. Names 'TR1' and 'TR2' are expanded in an obvious way.

Value

A named vector or list of the internally used format

Author(s)

Ilkka Virtanen (University of Oulu, Finland)
<ilkka.i.virtanen@oulu.fi>

Examples

```
# Expansion of a single unnamed value
LPIexpand.input( 4 )

# A combination of named and unnamed elements is accepted
LPIexpand.input( c( RX1=2, 4) )

# The RX and TX entries
LPIexpand.input(c(RX=2,TX=3))

# The TR1 and TR2 entries
LPIexpand.input(c(TR1=2,TR2=3))
```

---

stripACF *stripACF*

Description

Strip unwanted ranges and lags from ACFs

Usage

```
stripACF(ACFlist, rgates, lags, fullCovar = FALSE)
```

Arguments

ACFlist	An ACF list read from an LPI output file or returned by LPIrun.
rgates	Indices of range gates to preserve
lags	Indices of lags to preserve
fullCovar	Logical, TRUE if the ACF list contains full covariance matrices.

Value

An ACF list with only the selected ranges and lags maintained.

Author(s)

Ilkka Virtanen (University of Oulu, Finland)
<ilkka.i.virtanen@oulu.fi>

Examples

```
## Not run:

  # Load an LPI result file
  load( "1345541810000LP.Rdata" )

  # There will now be an ACF list on the workspace.
  # Check if it contains a full covariance matrix
  fullCov <- ifelse( is.null(ACF$covariance) , FALSE , TRUE )
  # Select range gates 30 - 40 and lags 1, 4, and 5.
  ACFstrip <- stripACF( ACF , seq(30,40) , c(1,4,5) , fullCov )

## End(Not run)
```

Index