Package: xemd (via r-universe)

November 8, 2026

Type Package

Title eXtended Empirical Mode Decomposition

Version 0.1.5

Description Signals are decomposed into oscillatory modes by empirical mode decomposition and its ensemble, complementary ensemble, complete ensemble, masked, bivariate and multivariate variants, by variational mode decomposition and its multivariate and successive variants, and by local mean decomposition, iterative filtering, time varying filter empirical mode decomposition and the empirical wavelet transform; the synchrosqueezed wavelet transform is also provided. Empirical mode decomposition is due to Huang et al. (1998) <doi:10.1098/rspa.1998.0193>, its ensemble variant to Wu and Huang (2009) <doi:10.1142/S1793536909000047> and variational mode decomposition to Dragomiretskiy and Zosso (2014) <doi:10.1109/TSP.2013.2288675>. The modes can be described by their instantaneous amplitude and frequency and their Hilbert spectrum, tested as intrinsic mode functions or against white noise, selected by energy and compared through orthogonality, mode splitting, reconstruction quality, ensemble dispersion and similarity measures. Acceleration, velocity and displacement records stored as a long table of time, value, quantity and component are decomposed for every record and quantity in one call, with a warning and a regularization of the time grid when it is not uniform.

URL https://kanameishi.github.io/xemd/

License GPL-3

Copyright Authors, copyright notices and licenses of the third-party code adapted in xemd are listed in file inst/NOTICE.

Encoding UTF-8

Depends R (>= 4.1.0)

Imports data.table, pracma, splines, stats, utils

Suggests hht, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Alejandro Verri Kozlowski [aut, cre, cph, trl] (ORCID: <https://orcid.org/0000-0002-8535-1170>), G. Rilling [ctb] (emd.m code), Marcelo A. Colominas [cph] (CEEMDAN distribution of emd.m), Pierre Masselot [ctb] (emdr code), Vinícius Rezende Carvalho [ctb, cph] (ewtpy code), Eduardo Mazoni [cph] (ewtpy code), Jerome Gilles [ctb, cph] (pyewt code; MATLAB EWT code translated in ewtpy), Kathryn Heal [ctb] (MATLAB EWT boundary detection translated in ewtpy), Emanuele Papini [ctb, cph] (fifpy code), Antonio Cicone [ctb] (MATLAB iterative filtering code translated in fifpy), Shubhra Prakash [ctb] (LMD code), Stefano Bianchi [ctb, cph] (pytvfemd code), Alessandro Longo [ctb, cph] (pytvfemd code), Eugene Brevdo [ctb] (Synchrosqueezing Toolbox code ported in SynchWave), Hau-Tieng Wu [ctb] (Synchrosqueezing Toolbox code ported in SynchWave), Jianfeng Lu [ctb] (curve_ext of the Synchrosqueezing Toolbox, ported in SynchWave), Dongik Jang [ctb] (SynchWave code), Hee-Seok Oh [ctb] (SynchWave code), Donghoh Kim [ctb] (SynchWave code)

Maintainer Alejandro Verri Kozlowski <averri@fi.uba.ar>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-08 19:52:49 UTC

RemoteUrl https://github.com/cran/xemd

RemoteRef HEAD

RemoteSha a2a96e4e0d525e50fd1a88b9e1dc5b1840559895

Contents

assessFractionalNoise . . . 3
assessIMF . . . 5
assessWhiteNoise . . . 6
buildHilbertSpectrum . . . 8
computeInstantaneousAttributes . . . 9
decomposeBEMD . . . 10
decomposeCEEMD . . . 12
decomposeCEEMDAN . . . 14
decomposeEEMD . . . 16
decomposeEMD . . . 18
decomposeEWT . . . 20
decomposeICEEMDAN . . . 23
decomposeIF . . . 25
decomposeIteratedMaskedEMD . . . 27
decomposeLMD . . . 29
decomposeMaskedEMD . . . 31

decomposeMEMD . . . 33
decomposeMVMD . . . 35
decomposeNAMEMD . . . 37
decomposeSVMD . . . 38
decomposeTSL . . . 40
decomposeTVFEMD . . . 43
decomposeVMD . . . 45
describeModule . . . 47
estimateMaskFrequency . . . 47
extractSSTModes . . . 49
listDecompositions . . . 51
listOperations . . . 51
measureDispersion . . . 52
measureDTW . . . 53
measureFCF . . . 54
measureModesTSL . . . 55
measureOrthogonality . . . 57
measurePMSI . . . 58
measureQRF . . . 59
reconstructTSL . . . 60
resiftModes . . . 61
selectModesEnergy . . . 63
selectModesTSL . . . 64
transformSST . . . 65

Index **68**

`assessFractionalNoise` *Fractional Gaussian Noise Test of Intrinsic Mode Functions*

Description

Compares the energy of each intrinsic mode function, from the second on, with the threshold that fractional Gaussian noise of a tabulated Hurst exponent gives at a tabulated level, with the first function as the noise scale.

Usage

```
assessFractionalNoise(modes, hurst, probability)
```

Arguments

`modes`	Finite numeric matrix of 2048 samples by at least two intrinsic mode functions, in their order of extraction and without the residue.
`hurst`	Hurst exponent H: `0.2`, `0.5` or `0.8`. No default.
`probability`	Level p of the threshold: `0.95` or `0.99`. No default.

Details

The energy of function k is $W[k] = \sum_n c_k[n]^2$. From $k = 2$ on, the reference energy is

$$W_H[k] = \frac{W[1]}{\beta_H}\rho_H^{-2(1-H)k}, \quad \rho_H = 2.01 + 0.2(H - 0.5) + 0.12(H - 0.5)^2,$$

and the threshold is $T_{H,p}[k] = W_H[k]\, 2^{2^{a_{H,p}} k + b_{H,p}}$, with β_H, $a_{H,p}$ and $b_{H,p}$ of table 3.1 of Flandrin et al. (2005). A function lies outside when $W[k] > T_{H,p}[k]$. The coefficients come from simulations of 10000 realizations of 2048 samples and are not interpolated to other exponents nor recalibrated for other lengths. The first function is taken to carry mainly noise; the function does not verify it.

The result is conditional on that noise model and calibration. Exceeding the threshold does not prove that a component is free of noise or identify its physical source.

Value

A list of vectors with one element per function:

`energy`	The energies $W[k]$.
`reference`	The reference energies $W_H[k]$, `NA` for the first function.
`upper`	The thresholds $T_{H,p}[k]$, `NA` for the first function.
`outside`	Logical, `energy > upper`, `NA` for the first function.

Conditions

A first function of zero energy stops the function with an error of class `xemdZeroEnergy`.

References

Flandrin, P., Gonçalvès, P. and Rilling, G. (2004). Detrending and denoising with empirical mode decompositions. *EUSIPCO 2004*, 1581–1584.

Flandrin, P., Gonçalvès, P. and Rilling, G. (2005). EMD equivalent filter banks, from interpretation to applications. In N. E. Huang and S. S. P. Shen (eds.), *Hilbert-Huang Transform and Its Applications*, World Scientific, 57–74.

See Also

`assessWhiteNoise`, `decomposeEMD`

Examples

```
Time <- seq_len(2048) / 1000
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
assessFractionalNoise(Tones, hurst = 0.5, probability = 0.95)
# The third tone as the noise scale.
assessFractionalNoise(Tones[, c(3, 1, 2)], hurst = 0.5,
                      probability = 0.95)$outside
```

`assessIMF` *Intrinsic Mode Function Conditions*

Description

Evaluates numerical diagnostics of the intrinsic mode function conditions for each column of a matrix: the balance of extrema and zero crossings, and a double threshold on the envelope mean relative to its amplitude.

Usage

```
assessIMF(modes, thresholds = c(0.05, 0.5), fraction = 0.05,
          boundary = "reflection")
```

Arguments

`modes`	Finite numeric matrix, samples by columns. Attributes other than `dim` do not reach the outputs.
`thresholds`	Two dimensionless numbers $0 < \theta_1 < \theta_2$; default `c(0.05, 0.5)`.
`fraction`	The admitted fraction α in $[0, 1)$; default `0.05`.
`boundary`	`"reflection"` (default) or `"extrapolation"`, the extension of the envelopes at the borders, as in `decomposeEMD`. The defaults are those of emd.m, shared with the sifting controls.

Details

The ideal envelope condition is a zero local mean. This diagnostic uses tolerances and a specified boundary rule; passing it does not establish recovery of a physical component. For methods defined by other conditions, such as a bandwidth objective, it is an additional EMD-based diagnostic.

`count` tests $|N_e - N_z| \leq 1$, with the extrema N_e and zero crossings N_z counted as in `decomposeEMD`. `deviation` applies to

$$\sigma = \frac{|m_e|}{a_e},$$

m_e and a_e the half sum and the half difference of the envelopes, the double threshold that accepts a mode under `stopRule = "doubleThreshold"`: the fraction of samples with $\sigma \geq \theta_1$ is at most α and no sample has $\sigma \geq \theta_2$.

Value

A list of two logical vectors with one element per column:

`count`	The counting condition.
`deviation`	The double threshold, `NA` where the boundary rule leaves the column without envelopes.

References

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

decomposeEMD, assessWhiteNoise

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
OUT <- decomposeEMD(Signal, sifting = list(stopRule = "doubleThreshold"))
# The signal, its modes and its residue as columns.
assessIMF(cbind(Signal, OUT$modes, OUT$residue))
```

---

assessWhiteNoise *White-Noise Significance Test of Modes*

Description

Compares the energy of each mode with reference bounds from a white-noise model at the mode's mean period (test of Wu and Huang).

Usage

```
assessWhiteNoise(modes, probabilities, bound, rescalingMode)
```

Arguments

`modes`	Finite numeric matrix, samples by modes. The caller supplies the components of a normalized signal; see Details for the assumptions of the reference distribution.
`probabilities`	Two numbers $0 < p_- < p_+ < 1$, the probabilities of the lower and upper bounds. No default; PyEMD uses $\alpha = 0.95$, that is $(1 \mp \alpha)/2$.
`bound`	`"gaussian"`, the line of the logarithmic quantile, or `"chiSquare"`, the quantile of the energy. No default; PyEMD has only the gaussian bound.
`rescalingMode`	`NULL` for the test without rescaling, or the index r of the column whose energy is placed on its upper bound. No default; PyEMD rescales on the first component.

Details

The energy of mode k is $E_k = \sum_n c_k[n]^2/N$ and its mean period T_k is N divided by its number of maxima, or N without maxima. The bound of probability p is, with `"gaussian"` and z_p the standard normal quantile,

$$\ln E_p = -\ln T + z_p\sqrt{2/N}\,T^{1/2},$$

and with `"chiSquare"` $E_p = q(p;r)/N$, the chi-square quantile with $r = N/T$ degrees of freedom. With `rescalingMode = r`, the energy of component r is placed on its upper bound and every other energy is multiplied by $E_{p_+}(T_r)/E_r$. A mode lies outside when its energy is below the lower bound or above the upper one.

The reference distribution is conditional on the noise model, normalization and EMD procedure. Interpreting the rescaled test assumes that the component selected by `rescalingMode` is dominated by noise; its mode index does not establish that assumption. Exceeding a bound does not identify a physical source or prove that a component is free of noise. A white-noise interpretation of an acceleration record does not transfer unchanged to modes of its integrated displacement.

Value

A list of vectors with one element per mode:

`energy`	The energies, rescaled when `rescalingMode` is given.
`period`	The mean periods, in samples.
`lower`, `upper`	The bounds at those periods.
`outside`	Logical, `energy < lower | energy > upper`.

Conditions

A column of zero energy stops the function with an error of class `xemdZeroEnergy`.

See Also

`assessIMF`, `decomposeEMD`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
# The four summands of the signal, divided by its population deviation.
Summands <- cbind(sin(pi * Time), sin(2 * pi * Time), sin(6 * pi * Time),
                  0.5 * Time) / sqrt(mean((Signal - mean(Signal))^2))
assessWhiteNoise(Summands, probabilities = c(0.025, 0.975),
                 bound = "gaussian", rescalingMode = NULL)$outside
assessWhiteNoise(Summands, probabilities = c(0.025, 0.975),
                 bound = "chiSquare", rescalingMode = 1)$outside
```

`buildHilbertSpectrum` *Hilbert Spectrum by Frequency Cells*

Description

Accumulates the instantaneous amplitude, or its square, of every mode in the frequency cell of its instantaneous frequency, sample by sample, and integrates the result over time.

Usage

```
buildHilbertSpectrum(frequency, amplitude, edges, exponent, samplingInterval)
```

Arguments

`frequency`	Numeric matrix, samples by modes, of instantaneous frequencies in Hz, finite or `NA`.
`amplitude`	Finite non-negative matrix with the dimensions of `frequency`, the instantaneous amplitudes in the units of the signal.
`edges`	At least two finite, strictly increasing cell edges in Hz; cell b is $[e_b, e_{b+1})$. Negative values are admitted. No default.
`exponent`	The power p: 1 for the amplitude spectrum, 2 for the spectrum weighted by squared amplitude. No default.
`samplingInterval`	Sampling interval Δt in seconds, a finite number greater than zero.

Details

The spectrum holds the mass of each cell,

$$H[n, b] = \sum_{k:\, f_k[n] \in [e_b, e_{b+1})} a_k[n]^p,$$

in units of the signal to the power p. A frequency outside the edges or `NA` adds nothing. The marginal spectrum of a cell is the time integral $\Delta t \sum_n H[n, b]$. No scaling or division by the cell width is applied: the values are masses, not densities. The inputs are usually the `frequency` and `amplitude` of `computeInstantaneousAttributes`.

Value

A list with

`spectrum`	Double matrix, samples by cells, in units of the signal to the power p.
`marginal`	Numeric vector by cell, in units of the signal to the power p times seconds.

See Also

`computeInstantaneousAttributes`, `decomposeEMD`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
Modes <- decomposeEMD(Signal,
                      sifting = list(stopRule = "doubleThreshold"))$modes
Attributes <- computeInstantaneousAttributes(Modes,
                                             samplingInterval = 9 / 2999)
OUT <- buildHilbertSpectrum(Attributes$frequency, Attributes$amplitude,
                            edges = c(1, 2, 3, 4, 5), exponent = 2,
                            samplingInterval = 9 / 2999)
dim(OUT$spectrum)
OUT$marginal
```

---

computeInstantaneousAttributes

Instantaneous Amplitude, Phase and Frequency

Description

Computes, for each column of a matrix of modes, the analytic signal and its instantaneous amplitude, continuous phase and frequency.

Usage

```
computeInstantaneousAttributes(modes, samplingInterval)
```

Arguments

`modes`	Finite double matrix, samples by modes, with at least three rows; it may have zero columns. Attributes other than `dim` do not reach the outputs.
`samplingInterval`	Sampling interval Δt in seconds, a finite number greater than zero.

Details

The analytic signal $z = c + i\mathcal{H}c$ of each column is computed with the discrete Fourier transform, multiplying each bin by $1 + \operatorname{sgn}(\omega)$: positive frequencies are doubled, negative ones removed, and the zero bin and, for an even number of samples, the Nyquist bin keep the factor 1, so $\operatorname{Re} z = c$. The transform implies the periodic extension of the record. The amplitude is $a = |z|$. Within each run of samples with $|z| > 0$, the phase starts at the principal value of $\arg z$ and adds the increments $\arg(z_{n+1}\bar{z}_n)$ taken in $(-\pi, \pi]$; a zero of z leaves its sample undefined (`NA`) and the next run starts again at its principal value. The frequency is

$$f_n = \frac{\phi_{n+1} - \phi_{n-1}}{4\pi\Delta t}$$

in Hz, the central difference of $\omega = \phi'$; it is `NA` where one of the three phases is, and each border repeats its neighbour. These local undefined values raise no condition.

Value

A list of three matrices, samples by modes:

`amplitude`	Instantaneous amplitude, in the units of the modes.
`phase`	Continuous phase in radians, `NA` at zeros of the analytic signal.
`frequency`	Instantaneous frequency in Hz, `NA` where the phase is undefined.

See Also

`buildHilbertSpectrum`, `decomposeEMD`, `estimateMaskFrequency`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
# Three tones of 2, 24 and 288 Hz as columns.
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
OUT <- computeInstantaneousAttributes(Tones,
                                      samplingInterval = SamplingInterval)
apply(OUT$amplitude, 2, range)
apply(OUT$frequency, 2, range)
```

`decomposeBEMD` *Bivariate Empirical Mode Decomposition*

Description

Decomposes a two-channel signal, read as the complex signal $z = x_1 + ix_2$, by successive extraction of bivariate modes with envelopes built on projections onto equally spaced directions of the plane.

Usage

```
decomposeBEMD(signal, directionCount = 64, envelope, sifting)
```

Arguments

`signal`	Numeric matrix of finite real values with two columns, the real and the imaginary parts, samples in rows.
`directionCount`	Whole number $D \geq 1$ of directions $\varphi_d = 2\pi(d-1)/D$; default `64`, the default of Rlibeemd `bemd` and of emdr `memd`.
`envelope`	Mean of the sifting, without default: `"trajectory"` averages the envelopes of the trajectory itself at the maxima of each projection; `"projection"` sums the envelopes of the projections, oriented on their directions, with the factor $2/D$.
`sifting`	Named list of sifting controls, as in `decomposeEMD`, with `stopRule = "fixed"` only. The trajectory envelope admits only `boundary = "reflection"`; the projection envelope also admits `"extrapolation"`.

Details

The projection on direction φ_d is $p_d = x_1 \cos\varphi_d + x_2 \sin\varphi_d$. With `"trajectory"`, e_d interpolates the two-dimensional samples at the maxima of p_d and the mean is

$$m = \frac{1}{D} \sum_{d=1}^{D} e_d;$$

with `"projection"`, the scalar upper envelope of p_d is oriented on $(\cos\varphi_d, \sin\varphi_d)$ and the sum is scaled by $2/D$. Each extraction subtracts the mean `fixedSift` times. Before extraction k the decomposition ends with `"terminalResidue"` when every projection of r_{k-1} has at most one extremum, with `"modeLimit"` when $k-1$ equals `sifting$maxModes`, and with `"insufficientExtrema"` when a direction has no envelope. One extraction serves both channels, so they share the number of modes. If a direction loses its envelope before fixed sifting completes, the decomposition stops before adding that vector candidate and preserves the residue from before the attempt. A candidate that reaches the iteration limit of an adaptive rule is kept and flagged `FALSE`. Directions are not dropped from the mean.

Value

A list with

`modes`	Double array, samples by modes by channels.
`residue`	Numeric matrix, samples by channels.
`iterations`	Integer vector, the subtractions of each mode.
`converged`	Logical vector by mode: `NA` after the fixed number of subtractions. Interrupted attempts are reported in `termination`.
`stopReason`	`"terminalResidue"`, `"modeLimit"` or `"insufficientExtrema"`.
`termination`	Data frame with one row: `mode`, the next extraction index after the retained modes; `direction`, the one-based index of the first direction without an envelope, or `NA` when none failed; `iterations`, subtractions of the interrupted attempt, zero if it did not start; and `stopReason`.

References

Rilling, G., Flandrin, P., Gonçalves, P. and Lilly, J. M. (2007). Bivariate Empirical Mode Decomposition. *IEEE Signal Processing Letters*, 14, 936–939.

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeMEMD`, `decomposeEMD`

Examples

```
Time <- 2 * pi * (0:511) / 512
z <- cos(0.3 * Time) * exp(2i * Time) +
    0.3 * abs(sin(2.3 * Time)) * exp(17i * Time)
Signal <- cbind(Re(z), Im(z))
OUT <- decomposeBEMD(Signal, directionCount = 64, envelope = "projection",
                     sifting = list(stopRule = "fixed", fixedSift = 10,
                                    boundary = "extrapolation",
                                    maxModes = 3))
dim(OUT$modes)
OUT$stopReason
max(abs(apply(OUT$modes, c(1, 3), sum) + OUT$residue - Signal))
```

---

decomposeCEEMD *Complementary Ensemble Empirical Mode Decomposition*

Description

Decomposes pairs of copies of a signal perturbed with the same noise realization of opposite signs and averages the modes of all the copies by extraction order.

Usage

```
decomposeCEEMD(signal, noise, amplitudeRatio = 0.2, sifting)
```

Arguments

`signal`	Numeric vector of at least two finite values without `dim`, in the units of the quantity.
`noise`	Finite numeric matrix with one row per sample of `signal` and one column per realization, $M \geq 1$; column m is a realization $z^{(m)}$ of unit variance, used as supplied. The function draws no random numbers.
`amplitudeRatio`	Dimensionless ratio $\varepsilon \geq 0$; the perturbation is $az^{(m)}$ with $a = \varepsilon\,\mathrm{sd}(x)$ (divisor $N-1$). Default `0.2`, as in `decomposeEEMD`.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected.

Details

For each column, the inputs $x + az^{(m)}$ and $x - az^{(m)}$ are decomposed by scalar sifting, the positive input first. Each stops before adding an extraction whose sifting lost its envelopes and keeps its input in the residue. Accepted adaptive extractions, completed fixed sifting and extractions that reached the iteration limit are retained. The $2M$ decompositions are extended with zero modes to the largest number of modes and averaged by extraction order,

$$d_j = \frac{1}{2M} \sum_{m=1}^{M} \left(c_j^{(m,+)} + c_j^{(m,-)} \right),$$

and the residue is the mean of the $2M$ residues, so the perturbations cancel in the reconstruction: `rowSums(modes) + residue` equals `signal` up to rounding. The method has no outer termination and returns neither `meanNoise` nor `stopReason`.

Value

A list with

`modes`	Double matrix, samples by modes.
`residue`	Numeric vector, the mean residue of the $2M$ decompositions.
`iterations`	Integer vector by mode, the sum of the subtractions of the decompositions that have the mode.
`converged`	Logical vector by mode: `TRUE` only when every decomposition that has the mode accepted it; `NA` for completed fixed sifting. Failed attempts are reported in `termination`.
`termination`	Data frame with one row per signed realization: `realization`, its column index in `noise`; `sign`, 1 or -1; and `mode`, `iterations` and `stopReason` as in `decomposeEEMD`. A stopped realization stays in its pair and in the divisor $2M$; its remaining content stays in the mean residue.

References

Yeh, J.-R., Shieh, J.-S. and Huang, N. E. (2010). Complementary ensemble empirical mode decomposition: A novel noise enhanced data analysis method. *Advances in Adaptive Data Analysis*, 2(2), 135–156. doi:10.1142/S1793536910000422

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeEEMD`, `decomposeCEEMDAN`, `decomposeICEEMDAN`

Examples

```
Time <- seq(0, 9, length.out = 1000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
set.seed(628)
Noise <- t(matrix(stats::rnorm(10L * length(Signal)), nrow = 10L,
                  byrow = TRUE))
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = Inf)
# Two of the ten realizations keep the example short.
OUT <- decomposeCEEMD(Signal, noise = Noise[, 1:2], amplitudeRatio = 0.2,
                      sifting = Sifting)
dim(OUT$modes)
max(abs(rowSums(OUT$modes) + OUT$residue - Signal))
```

`decomposeCEEMDAN` *Complete Ensemble Empirical Mode Decomposition with Adaptive Noise*

Description

Extracts each mode as the average first mode of the common residue perturbed by the modes of supplied noise realizations, with a constant ratio of the noise to the residue at each stage.

Usage

```
decomposeCEEMDAN(signal, noise, amplitudeRatio = 0.2, sifting)
```

Arguments

`signal`	Numeric vector of at least two finite values without `dim`, in the units of the quantity.
`noise`	Finite numeric matrix with one row per sample of `signal` and one column per realization, $M \geq 1$; column m is a realization $w^{(m)}$ of unit variance, used as supplied. The function draws no random numbers.
`amplitudeRatio`	Dimensionless ratio $\varepsilon \geq 0$ of the perturbation to the standard deviation of the residue at each stage (divisor $N - 1$). Default `0.2`, as in `decomposeEEMD`.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected. The same list decomposes the noise realizations.

Details

Each column of `noise` is decomposed once with `sifting`, $\eta_k^{(m)} = E_k(w^{(m)})$, and $\eta_0^{(m)} = w^{(m)}$. From $r_0 = x$, before stage k the common residue r_{k-1} ends the decomposition, in this order, with `"terminalResidue"` when it has at most one extremum, with `"modeLimit"` when $k - 1$ equals `sifting$maxModes` and with `"insufficientExtrema"` when its envelopes cannot be built. Stage k perturbs r_{k-1} with $p^{(m)} = \beta_{k-1}^{(m)} \eta_{k-1}^{(m)}$, $\beta_{k-1}^{(m)} = \varepsilon \, \mathrm{sd}(r_{k-1}) / \mathrm{sd}(\eta_{k-1}^{(m)})$, and computes

$$d_k = \frac{1}{M} \sum_{m=1}^{M} E_1 \left(r_{k-1} + p^{(m)} \right), \qquad r_k = r_{k-1} - d_k.$$

A realization without η_{k-1} contributes a null perturbation, and a perturbed residue without a first mode contributes zero; both stay in the divisor M.

Value

A list with

`modes`	Double matrix, samples by modes.
`residue`	Numeric vector, the final common residue.

`iterations` Integer vector by mode, the sum of the subtractions of the M extractions of the stage.

`converged` Logical vector by mode: `TRUE` when the M extractions accepted, `FALSE` when one ended without accepting or could not begin, `NA` when all completed fixed sifting.

`stopReason` `"terminalResidue"`, `"modeLimit"` or `"insufficientExtrema"`.

Conditions

A noise mode of zero standard deviation leaves its scale undefined and stops the function with an error of class `xemdZeroEnergy`.

References

Torres, M. E., Colominas, M. A., Schlotthauer, G. and Flandrin, P. (2011). A complete ensemble empirical mode decomposition with adaptive noise. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4144–4147. doi:10.1109/ICASSP.2011.5947265

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeICEEMDAN`, `decomposeEEMD`, `decomposeCEEMD`

Examples

```
Time <- seq(0, 9, length.out = 1000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
set.seed(628)
Noise <- t(matrix(stats::rnorm(10L * length(Signal)), nrow = 10L,
                  byrow = TRUE))
# Two of the ten realizations and three modes keep the example short.
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = 3)
OUT <- decomposeCEEMDAN(Signal, noise = Noise[, 1:2], amplitudeRatio = 0.2,
                        sifting = Sifting)
dim(OUT$modes)
OUT$stopReason
max(abs(rowSums(OUT$modes) + OUT$residue - Signal))
```

`decomposeEEMD` *Ensemble Empirical Mode Decomposition*

Description

Decomposes noise-perturbed copies of a signal by scalar sifting and averages their completed modes by extraction order.

Usage

```
decomposeEEMD(signal, noise, amplitudeRatio = 0.2, divisor, sifting)
```

Arguments

`signal`	Numeric vector of at least two finite values without `dim`, in the units of the quantity.
`noise`	Finite numeric matrix with one row per sample of `signal` and one column per realization, $M \geq 1$. Column m is a realization $z^{(m)}$ of unit variance, used as supplied: the function neither generates nor normalizes it, and it draws no random numbers.
`amplitudeRatio`	Dimensionless ratio $\varepsilon \geq 0$ of the noise standard deviation to the standard deviation of `signal` (divisor $N - 1$). Default `0.2`, the value common to libeemd and emd 0.8.1.
`divisor`	Categorical, without default: `"realizations"` divides the sum of the modes of order j by M; `"available"` divides it by the number M_j of realizations that have that mode.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected.

Details

Realization m is $x + w^{(m)}$ with $w^{(m)} = \varepsilon \, \mathrm{sd}(x) \, z^{(m)}$; it is decomposed by scalar sifting with `sifting`, in the order of the columns of `noise`. Each realization stops before adding an extraction whose sifting lost its envelopes and keeps the input of that attempt in its residue. Accepted adaptive extractions, completed fixed sifting and extractions that reached the iteration limit are retained, the last ones flagged `FALSE`. Absent modes are extended with zeros, and mode j is

$$D_j = \frac{1}{M} \sum_{m=1}^{M} c_j^{(m)}$$

with `"realizations"`, or the same sum divided by M_j with `"available"`. The residue is the mean of the M final residues and `meanNoise` the mean of the added perturbations, so that with `"realizations"` the modes, the residue and `signal` satisfy `rowSums(modes) + residue = signal + meanNoise` up to rounding. A realization without modes contributes zeros to the modes and its input to the residue. Matching by extraction order does not establish a physical correspondence between the modes of different realizations.

The method has no outer termination, so the result has no `stopReason`. The number of realizations has no default; the audited libraries use 100 (PyEMD), 250 (libeemd) and 4 (emd 0.8.1). hht divides by M and PyEMD by M_j.

Value

A list with

`modes`	Double matrix, samples by modes, with as many columns as the largest number of modes among the realizations.
`residue`	Numeric vector, the mean final residue.
`meanNoise`	Numeric vector, the mean added perturbation, in the units of `signal`.
`iterations`	Integer vector by mode, the sum of the subtractions of the realizations that have the mode.
`converged`	Logical vector by mode: `TRUE` only when every realization that has the mode accepted it; `NA` for completed fixed sifting. Failed attempts are reported in `termination`.
`termination`	Data frame with one row per realization: `realization`, its column index in `noise`; `mode`, the next extraction index after the retained modes; `iterations`, subtractions of the interrupted attempt, zero if it did not start; and `stopReason`, `"terminalResidue"`, `"modeLimit"` or `"insufficientExtrema"`. These rows describe individual decompositions, not the averaged mode indices.

References

Wu, Z. and Huang, N. E. (2009). Ensemble empirical mode decomposition: A noise-assisted data analysis method. *Advances in Adaptive Data Analysis*, 1(1), 1–41. doi:10.1142/S1793536909000047

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeCEEMD`, `decomposeCEEMDAN`, `decomposeICEEMDAN`, `resiftModes`, `measureDispersion`

Examples

```
Time <- seq(0, 9, length.out = 1000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
set.seed(628)
Noise <- t(matrix(stats::rnorm(10L * length(Signal)), nrow = 10L,
                  byrow = TRUE))
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = Inf)
# Two of the ten realizations keep the example short.
OUT <- decomposeEEMD(Signal, noise = Noise[, 1:2], amplitudeRatio = 0.2,
                     divisor = "realizations", sifting = Sifting)
```

```
dim(OUT$modes)
OUT$iterations
max(abs(rowSums(OUT$modes) + OUT$residue - Signal - OUT$meanNoise))
```

`decomposeEMD` *Empirical Mode Decomposition*

Description

Decomposes a signal by successive extraction: each component is obtained by sifting the current residue and is subtracted from that residue. Its numerical acceptance depends on the selected stopping rule.

Usage

```
decomposeEMD(signal, sifting)
```

Arguments

`signal`	Numeric vector of finite real values without `dim`, in the units of the quantity. Positions are sample indices, so no time vector or sampling interval is needed. Attributes such as names, `tsp` or class do not reach the outputs.
`sifting`	Named list of sifting controls: `stopRule` and the fields of that rule, described in section 'Sifting controls'. Fields with a default may be omitted; every other field is required, and a field that does not belong to the rule is an error.

Details

With $r_0 = x$, extraction k computes the mode $c_k = E_1(r_{k-1})$ by sifting and updates the residue

$$r_k = r_{k-1} - c_k,$$

so that $x = \sum_{k=1}^{K} c_k + r_K$ up to rounding. Before extraction k the decomposition ends, in this order, with `"terminalResidue"` when r_{k-1} has at most one extremum, with `"modeLimit"` when $k - 1$ equals `sifting$maxModes`, and with `"insufficientExtrema"` when the boundary rule cannot build the envelopes of r_{k-1}.

Sifting starts from $h_0 = r_{k-1}$ and repeats $h_{j+1} = h_j - m_j$, where m_j is the mean of the upper and lower envelopes of h_j. The envelopes interpolate the maxima and the minima, extended at both borders by `sifting$boundary`: a cubic spline with not-a-knot end conditions through four or more nodes, the parabola through three and the line through two. A run of equal interior samples is one extremum at the exact centre of the run. Each state h_j is examined once, in this order: the stopping rule accepts it (`converged` `TRUE`); j equals `fixedSift` under `"fixed"` (`converged` `NA`); j equals `maxSift` under an adaptive rule, or the envelopes of h_j cannot be built (`converged` `FALSE`); otherwise one more subtraction follows. Non-convergence is reported in `converged`, without a warning.

Value

A list with

`modes`	Double matrix, samples by modes, in the units of `signal`; it has zero columns when no mode is extracted. No names.
`residue`	Numeric vector r_K of the input length; equal to `signal` when no mode is extracted.
`iterations`	Integer vector with the number of subtractions of each mode.
`converged`	Logical vector by mode: `TRUE` when the stopping rule accepted the mode, `FALSE` when `maxSift` was reached or the envelopes could not be built, `NA` under `"fixed"`.
`stopReason`	`"terminalResidue"`, `"modeLimit"` or `"insufficientExtrema"`.

Sifting controls

The same list configures the extractor of `decomposeEEMD`, `decomposeCEEMD`, `decomposeCEEMDAN`, `decomposeICEEMDAN`, `resiftModes`, `decomposeMaskedEMD`, `decomposeIteratedMaskedEMD`, `estimateMaskFrequency` and, with the restrictions stated in their help pages, `decomposeBEMD`, `decomposeMEMD` and `decomposeNAMEMD`. Counts are accepted as doubles with a whole value.

`stopRule` Categorical, without default: `"SD"`, `"Snumber"`, `"doubleThreshold"` or `"fixed"`. The rule `"consecutive"` belongs to the vector sifting of `decomposeMEMD` and is rejected here.

`tolerance` Under `"SD"`: dimensionless threshold, greater than zero, without default. From $j = 1$ the state is accepted when $\sum_n (h_{j-1}[n] - h_j[n])^2 / h_{j-1}[n]^2$ is finite and below it; the sum runs over all samples, without the factor $1/N$, and a null denominator is not regularized. The 1998 formulation proposes values from 0.2 to 0.3.

`stableSift` Under `"Snumber"`: the count $S \geq 1$, without default. From $j = 1$ the state is accepted when the counts of extrema E and zero crossings Z have stayed equal over S consecutive states and $|E - Z| \leq 1$. The 2003 formulation reports typical values from three to five.

`thresholds` Under `"doubleThreshold"`: two dimensionless numbers $0 < \theta_1 < \theta_2$; default `c(0.05, 0.5)`.

`fraction` Under `"doubleThreshold"`: the admitted fraction α in $[0, 1)$; default `0.05`. With $\sigma = |m/a|$, m the half sum and a the half difference of the envelopes, the state, the signal included, is accepted when $|E - Z| \leq 1$, the fraction of samples with $\sigma \geq \theta_1$ is at most α and no sample has $\sigma \geq \theta_2$. Where $m = a = 0$ the ratio is undefined and counts as no exceedance; where only $a = 0$ it is infinite.

`fixedSift` Under `"fixed"`: the count $J \geq 1$ of subtractions of every extraction, without default.

`maxSift` Under `"SD"`, `"Snumber"` and `"doubleThreshold"`: the count $M \geq 1$ of subtractions after which an extraction ends without acceptance; under `"Snumber"` $M \geq S$. Default `2000` under `"doubleThreshold"`, none under the other rules. The rule `"fixed"` has no `maxSift`.

`maxModes` Count ≥ 1 or `Inf`, the limit of extractions; default `Inf`, which extracts until the residue ends the decomposition.

`boundary` Categorical: `"reflection"` or `"extrapolation"`; default `"reflection"` under `"doubleThreshold"` and `"fixed"`, none under `"SD"` and `"Snumber"`. `"reflection"` mirrors two extrema of each family per side about the first extremum or the border sample and needs three extrema; `"extrapolation"` anchors the upper envelope at each border at the larger of the border sample and the line through the two nearest maxima, and the lower envelope at the smaller of the border sample and the line through the two nearest minima.

References

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.-C., Tung, C. C. and Liu, H. H. (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A*, 454, 903–995. doi:10.1098/rspa.1998.0193

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

assessIMF, decomposeEEMD, decomposeTSL

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = Inf)
OUT <- decomposeEMD(Signal, sifting = Sifting)
dim(OUT$modes)
OUT$iterations
OUT$stopReason
max(abs(rowSums(OUT$modes) + OUT$residue - Signal))

# The rule alone: the other fields take their defaults.
identical(decomposeEMD(Signal, sifting = list(stopRule = "doubleThreshold")),
          OUT)

# Ten fixed subtractions per mode, at most three modes.
Fixed <- decomposeEMD(Signal, sifting = list(stopRule = "fixed",
                                             fixedSift = 10, maxModes = 3))
Fixed$converged
Fixed$stopReason
```

---

decomposeEWT *Empirical Wavelet Transform*

Description

Splits the spectrum of a sampled signal at the midpoints between its dominant spectral maxima, or at the boundaries of a detection rule of the EWT code of Gilles, and returns the components of an adapted filter bank, which add up to the signal.

Usage

```
decomposeEWT(signal, samplingInterval, bandCount, detection = NULL)
```

Arguments

`signal` Numeric vector of at least four finite values without `dim`, in the units of the quantity.

`samplingInterval`
Sampling interval Δt in seconds, a finite number greater than zero.

`bandCount` Whole number ≥ 2, the largest number of bands, or `NULL` for the automatic count of `detection`. No default; ewtpy uses 5, and pyewt, the package of the author of the method, requires the value.

`detection` `NULL`, the default, for the midpoint boundaries, or a named list with these fields:

`detector` `"locmax"`, `"locmaxmin"` or `"locmaxminf"`; required.

`logarithm` `TRUE` to work on the natural logarithm of the spectral magnitude, which must then be positive; default `FALSE`.

`regularization` `"none"`, `"average"` or `"gaussian"`; default `"gaussian"`.

`window` Whole number $L \geq 1$ of cells of the regularization filter; default `10`.

`sigma` Standard deviation $\sigma > 0$ of the gaussian filter, in cells; default `1.5`.

`thresholdRatio` Number α in $(0, 1)$, given only with `bandCount = NULL`. No default; Gilles (2013) tries 0.3 and 0.4.

The defaults are the initial values of the EWT code of Gilles; its initial detector, `"scalespace"`, is not available. With `bandCount = NULL` the list must give `detector = "locmax"`, `regularization = "none"`, `logarithm = FALSE` and `thresholdRatio`.

Details

The maxima of the one-sided magnitude of the discrete Fourier transform are ranked, and the `bandCount - 1` largest are retained, or all of them when fewer exist, so the number of bands adapts to the spectrum. With $b_0 = 0$ and $b_1 < \cdots < b_{K-1}$ the bins of the retained maxima, the boundaries are the midpoints $\Omega_j = \pi(b_{j-1} + b_j)/N$ rad per sample, and the last band closes at π. The record is extended by mirroring $\lceil N/2 \rceil - 1$ samples before it and $\lceil N/2 \rceil$ after it. On that extension the bank holds an approximation filter and band filters with cosine transitions of half width $\gamma\Omega_j$, γ below the bound that keeps neighbouring transitions apart. The analysis coefficients are $W_j = \mathcal{F}^{-1}[\hat{x}\,\overline{H_j}]$, and each component applies the dual filter $H_j / \sum_i H_i^2$,

$$c_j = \mathcal{F}^{-1}\left[\frac{\hat{W}_j H_j}{\sum_i H_i^2}\right],$$

restricted to the record, so the components add up to the signal. The components are defined by the filter bank, without imposing the IMF conditions of EMD. The method has no iterative stopping rule.

A detection list works on the one-sided magnitude s, or on its natural logarithm, and on its regularization p, the central n values of the full convolution of s with the weights h_r, $r = 0, \ldots, L-1$, from position $\lfloor L/2 \rfloor + 1$: $h_r = 1/L$ for `"average"` and $h_r \propto \exp[-u_r^2/(2\sigma^2)]$, $u_r = r - (L-1)/2$, for `"gaussian"`; `"none"` keeps $p = s$. `"locmax"` places the midpoints above on the maxima of p. `"locmaxmin"` and `"locmaxminf"` need `bandCount - 1` positive maxima of p and retain the largest; on the interval from the first cell to the first retained maximum and on those between consecutive retained maxima, endpoints included, the boundary is the lowest minimum of p for `"locmaxmin"` or of s for `"locmaxminf"`, the upper central one among equal minima, and a minimum at position

q gives $\Omega = 2\pi(q-1)/N$. No boundary is completed. With `bandCount = NULL`, the number of bands is the number of maxima of the magnitude whose height exceeds $M_m + \alpha(M_1 - M_m)$, M_1 and M_m the largest and smallest heights, and the midpoint boundaries follow; one band returns the signal.

Value

A list with

`modes`	Double matrix, samples by bands.
`boundaries`	Numeric vector of the $K-1$ boundaries in Hz.

Conditions

Equal magnitudes at the last retained maximum and the first discarded one leave the selection undefined and stop with an error, as does a frequency where every filter of the bank vanishes. With a detection list, a zero magnitude under `logarithm = TRUE`, fewer than `bandCount - 1` positive maxima of p or an interval without minimum for `"locmaxmin"` and `"locmaxminf"`, boundaries that do not increase and an automatic count of zero bands stop with an error.

References

Gilles, J. (2013). Empirical wavelet transform. *IEEE Transactions on Signal Processing*, 61(16), 3999–4010.

Gilles, J. *Empirical Wavelets*, MATLAB toolbox, revision 0381208d40352237876b79912013b0f68337133c, https://github.com/jegilles/Empirical-Wavelets.

The detection of the boundaries, the mirror extension and the filter bank are adapted from ewtpy, revision f9ff38bc5cd7859eea58c5b29099f8a503792b63, copyright (c) 2019 Vinícius Carvalho and Eduardo Mazoni, MIT License; the synthesis through the dual bank is adapted from pyewt, revision 9d60a340f34e92356ed4f7a6a8b7044f524455e3, copyright (c) 2024 Jerome Gilles, MIT License. See the NOTICE file of the installed package.

See Also

`transformSST`, `decomposeVMD`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
OUT <- decomposeEWT(Signal, samplingInterval = SamplingInterval,
                    bandCount = 3)
OUT$boundaries
dim(OUT$modes)
max(abs(rowSums(OUT$modes) - Signal))
# Automatic count: only the 2 Hz maximum exceeds the threshold of 0.3.
dim(decomposeEWT(Signal, samplingInterval = SamplingInterval,
                 bandCount = NULL,
```

```
                detection = list(detector = "locmax", regularization = "none",
                                 thresholdRatio = 0.3))$modes)
# Lowest minima between the maxima of the regularized magnitude.
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
decomposeEWT(Signal, samplingInterval = 9 / 2999, bandCount = 3,
             detection = list(detector = "locmaxmin"))$boundaries
```

`decomposeICEEMDAN` *Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise*

Description

Computes each residue as the average local mean of the previous residue perturbed by a mode of supplied noise realizations, and each mode as the difference between consecutive residues.

Usage

```
decomposeICEEMDAN(signal, noise, amplitudeRatio = 0.2, normalizeNoise,
                  sifting)
```

Arguments

`signal`	Numeric vector of at least two finite values without `dim`, in the units of the quantity.
`noise`	Finite numeric matrix with one row per sample of `signal` and one column per realization, $M \geq 1$; column m is a realization $w^{(m)}$ of unit variance, used as supplied. The function draws no random numbers.
`amplitudeRatio`	Dimensionless ratio $\varepsilon_0 \geq 0$. Default `0.2`, as in `decomposeEEMD`.
`normalizeNoise`	`TRUE` or `FALSE`, without default. From the second stage on, `TRUE` divides each noise mode by its standard deviation before scaling; `FALSE` keeps its amplitude. `normalizeNoise = TRUE` corresponds to `SNRFlag = 2` of ceemdan.m.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected. The same list decomposes the noise realizations.

Details

Each column of `noise` is decomposed once with `sifting`, $\eta_k^{(m)} = E_k(w^{(m)})$. From $r_0 = x$, before stage k the common residue r_{k-1} ends the decomposition, in this order, with `"terminalResidue"` when it has at most one extremum, with `"modeLimit"` when $k - 1$ equals `sifting$maxModes` and with `"insufficientExtrema"` when its envelopes cannot be built. Stage k perturbs r_{k-1} with $p^{(m)} = \beta_{k-1}^{(m)} \eta_k^{(m)}$, where $\beta_0^{(m)} = \varepsilon_0 \, \mathrm{sd}(x) / \mathrm{sd}(\eta_1^{(m)})$ and, for $k \geq 2$, $\beta_{k-1} = \varepsilon_0 \, \mathrm{sd}(r_{k-1})$, divided by $\mathrm{sd}(\eta_k^{(m)})$ when `normalizeNoise` is `TRUE`; standard deviations use the divisor $N - 1$. With A the

average first mode of $r_{k-1} + p^{(m)}$ and $\bar{p}$ the mean perturbation, the new residue is the mean local mean and the mode the difference of residues:

$$r_k = r_{k-1} + \bar{p} - A, \qquad d_k = r_{k-1} - r_k.$$

A realization without η_k contributes a null perturbation and stays in the divisor M.

Value

A list with

`modes`	Double matrix, samples by modes.
`residue`	Numeric vector, the final common residue.
`iterations`	Integer vector by mode, the sum of the subtractions of the M extractions of the stage.
`converged`	Logical vector by mode: `TRUE` when the M extractions accepted, `FALSE` when one ended without accepting or could not begin, `NA` when all completed fixed sifting.
`stopReason`	`"terminalResidue"`, `"modeLimit"` or `"insufficientExtrema"`.

Conditions

A noise mode of zero standard deviation leaves its scale undefined and stops the function with an error of class `xemdZeroEnergy`.

References

Colominas, M. A., Schlotthauer, G. and Torres, M. E. (2014). Improved complete ensemble EMD: A suitable tool for biomedical signal processing. *Biomedical Signal Processing and Control*, 14, 19–29. doi:10.1016/j.bspc.2014.06.009

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeCEEMDAN`, `decomposeEEMD`, `decomposeCEEMD`

Examples

```
Time <- seq(0, 9, length.out = 1000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
set.seed(628)
Noise <- t(matrix(stats::rnorm(10L * length(Signal)), nrow = 10L,
                  byrow = TRUE))
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = Inf)
# Two of the ten realizations keep the example short.
```

```
OUT <- decomposeICEEMDAN(Signal, noise = Noise[, 1:2], amplitudeRatio = 0.2,
                         normalizeNoise = TRUE, sifting = Sifting)
dim(OUT$modes)
OUT$stopReason
max(abs(rowSums(OUT$modes) + OUT$residue - Signal))
```

`decomposeIF` *Iterative Filtering*

Description

Decomposes a signal by successive extraction of components, each obtained by iterating the subtraction of a moving average whose filter is a supplied prototype resampled to a support set by the extrema of the residue.

Usage

```
decomposeIF(signal, filter, boundary, tolerance, maxIterations, stretch,
            maxModes = Inf)
```

Arguments

`signal`	Numeric vector of finite values without `dim`, in the units of the quantity. Positions are sample indices.
`filter`	Prototype filter, a numeric vector of finite weights of odd length, at least 3. No default. The package tests use the 13 weights that fifpy obtains from its prototype for a half width of 6.
`boundary`	`"zero"` extends the record with zeros; `"periodic"` extends it periodically and evaluates the iteration spectrally (fast iterative filtering). No default.
`tolerance`	Bound of the relative change between consecutive iterates, greater than zero and less than one. No default; the package tests use $\sqrt{0.001}$.
`maxIterations`	Whole number ≥ 1 of iterations of one extraction. No default; the package tests use 200.
`stretch`	Number greater than zero that scales the support of the filter. No default; the package tests use 1.6.
`maxModes`	Whole number ≥ 1 or `Inf`, the limit of extractions; default `Inf`.

Details

Before extraction k the decomposition ends with `"terminalResidue"` when r_{k-1} has at most one extremum and with `"modeLimit"` when $k-1$ equals `maxModes`. Otherwise the half width is $l_k = 2\,\mathrm{round}(\mathrm{stretch}\,N/E)$ for the E extrema of r_{k-1}; from the second extraction a value that does not exceed the previous half width becomes $\lceil 1.1\,l_{k-1} \rceil$. The prototype is resampled to $2l_k + 1$ weights and the iteration

$$f^{(q)} = f^{(q-1)} - \mathcal{L}_w f^{(q-1)}, \qquad f^{(0)} = r_{k-1},$$

with $\mathcal{L}_w$ the moving average under `boundary`, stops at the first q with $\|f^{(q)}-f^{(q-1)}\|_2/\|f^{(q-1)}\|_2 \leq$ `tolerance` (`converged` `TRUE`) or after `maxIterations` (`converged` `FALSE`). Then $c_k = f^{(Q)}$ and $r_k = r_{k-1} - c_k$. A shrinking prototype is integrated over rescaled bins; an enlarged one is interpolated linearly and normalized to a unit absolute sum. The components are not shown to satisfy the counting conditions of intrinsic mode functions.

Value

A list with

`modes`	Double matrix, samples by components.
`residue`	Numeric vector, the final residue.
`iterations`	Integer vector, the iterations Q of each component.
`converged`	Logical vector by component.
`stopReason`	`"terminalResidue"` or `"modeLimit"`.

References

Cicone, A. and Zhou, H. (2018). Numerical Analysis for Iterative Filtering with New Efficient Implementations Based on FFT. arXiv:1802.01359.

The support rule, the resampling of the prototype and both iterations are adapted from fifpy, revision 26e51e7d45e551950dea8be37ece447fc4340296, copyright (c) 2022 Emanuele Papini, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeLMD`, `decomposeEMD`

Examples

```
Time <- seq_len(1000) / 1000
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
Filter <- c(6.6306270449357867e-06, 0.00053827687799438354,
            0.0082858777882808092, 0.044245208647455882, 0.11927281750086954,
            0.20513332487239222, 0.24503568511284984, 0.20513332487239222,
            0.11927281750086952, 0.044245208647455903, 0.0082858777882808109,
            0.00053827687799438213, 6.6306270449358087e-06)
OUT <- decomposeIF(Signal, filter = Filter, boundary = "periodic",
                   tolerance = sqrt(0.001), maxIterations = 200,
                   stretch = 1.6, maxModes = 3)
OUT$iterations
OUT$stopReason
Zero <- decomposeIF(Signal, filter = Filter, boundary = "zero",
                    tolerance = sqrt(0.001), maxIterations = 200,
                    stretch = 1.6, maxModes = 3)
Zero$iterations
```

`decomposeIteratedMaskedEMD` *Iterated Masking Empirical Mode Decomposition*

Description

Iterates a uniform-phase masked decomposition, updating each mask frequency to the amplitude-weighted mean instantaneous frequency of its mode until a relative-change test passes or the iteration limit is reached (itEMD).

Usage

```
decomposeIteratedMaskedEMD(signal, samplingInterval, initialFrequencies,
                           phaseCount, amplitudeRatio, tolerance = 0.1,
                           maxIterations = 15, sifting)
```

Arguments

`signal`	Numeric vector of finite real values without `dim`, in the units of the quantity.
`samplingInterval`	Sampling interval in seconds, a finite number greater than zero, without default.
`initialFrequencies`	Numeric vector of $K \geq 1$ initial mask frequencies in Hz, each in $(0, 1/(2\Delta t)]$, without default; its length is the number of masks.
`phaseCount`	Whole number $P \geq 2$ of mask phases, without default; the package tests use 4.
`amplitudeRatio`	Dimensionless $\varepsilon_0 > 0$, without default; the mask amplitude of each stage is ε_0 times the standard deviation of the residue with divisor N. The package tests use 1.
`tolerance`	Threshold of the largest relative change of the mask frequencies, a finite number greater than zero; default `0.1`.
`maxIterations`	Whole number ≥ 1 of frequency updates; default `15`. The defaults of `tolerance` and `maxIterations` are those of `iterated_mask_sift` in emd 0.8.1, the code of a coauthor of itEMD.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected.

Details

Iteration q decomposes the signal with the uniform-phase masks of `decomposeMaskedEMD`, one stage per frequency of $\mathbf{f}^{(q)}$, phases $2\pi p/P$ and the amplitude above. Each frequency is then updated to the mean instantaneous frequency of its mode weighted by the squared instantaneous amplitude, over the samples where the frequency is defined,

$$f_j^{(q+1)} = \frac{\sum_n A_j[n]^2 \, \nu_j[n]}{\sum_n A_j[n]^2},$$

with A_j and ν_j from computeInstantaneousAttributes. The iteration stops with "tolerance" when every relative change $|f_j^{(q+1)} - f_j^{(q)}|/f_j^{(q)}$ is below tolerance, or with "iterationLimit" after maxIterations updates. The result is the last decomposition together with the frequencies that produced it.

Value

A list with the fields of decomposeMaskedEMD for the last decomposition (modes, residue, iterations, converged) and

stopReason "tolerance" or "iterationLimit".

maskFrequencies
: Numeric vector, the K mask frequencies in Hz that produced the returned decomposition.

maskIterations Integer, the number of frequency updates.

innerStopReason
: The termination of the returned masked decomposition, with the vocabulary of decomposeMaskedEMD.

Conditions

When the masked decomposition has fewer modes than masks the update is undefined and the function stops with an error of class xemdInsufficientModes. A mode with zero squared amplitude where its frequency is defined stops with an error of class xemdZeroEnergy. An updated frequency that is not positive stops with an error when the iteration continues.

References

Fabus, M. S., Quinn, A. J., Warnaby, C. E. and Woolrich, M. W. (2021). Automatic decomposition of electrophysiological data into distinct nonsinusoidal oscillatory modes. *Journal of Neurophysiology*, 126(5), 1670–1684. doi:10.1152/jn.00315.2021

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

decomposeMaskedEMD, computeInstantaneousAttributes

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = 3)
# Three dyadic initial frequencies, with one sample per second.
OUT <- decomposeIteratedMaskedEMD(Signal, samplingInterval = 1,
```

```
                                initialFrequencies = 55 / 3000 / 2 / 2^(0:2),
                                phaseCount = 4, amplitudeRatio = 1,
                                tolerance = 0.1, maxIterations = 15,
                                sifting = Sifting)
OUT$maskFrequencies
OUT$maskIterations
OUT$stopReason
```

`decomposeLMD` *Local Mean Decomposition*

Description

Decomposes a signal by successive extraction of product functions, each obtained by repeated subtraction of a smoothed local mean and division by a smoothed local magnitude.

Usage

```
decomposeLMD(signal, maxIterations, envelopeTolerance, changeTolerance,
             maxSmoothPasses, maxModes = Inf)
```

Arguments

`signal`	Numeric vector of finite values without `dim`, in the units of the quantity. Positions are sample indices.
`maxIterations`	Whole number ≥ 1 of normalizations allowed in one product function. No default; the package tests use 200.
`envelopeTolerance`	Number ≥ 0; an extraction ends when the mean of $\lvert 1 - a_q \rvert$ over the samples is at most this value. No default; the package tests use 0.01.
`changeTolerance`	Number ≥ 0; an extraction ends when the mean of $\lvert s_{q-1} - s_q \rvert$ is at most this value. No default; the package tests use 0.01.
`maxSmoothPasses`	Whole number ≥ 1 of passes of the moving average that smooths the local levels. No default; 12 is the number of passes that LMD 1.2.1 reaches by default.
`maxModes`	Whole number ≥ 1 or `Inf`, the limit of product functions; default `Inf`.

Details

The local levels of a function use as nodes its interior extrema and its first and last samples: consecutive nodes n_i, n_{i+1} give the mean $m_i = (n_i + n_{i+1})/2$ and the magnitude $a_i = |n_i - n_{i+1}|/2$, held between the two nodes. Both levels are smoothed by a repeated uniform moving average whose half width follows from the largest gap between nodes and shrinks at the borders; passes repeat while two consecutive samples are equal, up to `maxSmoothPasses`.

From $s_0 = r_{p-1}$, iteration q forms

$$s_q = \frac{s_{q-1} - m_q}{a_q}$$

with the smoothed levels of s_{q-1}. The extraction ends without that normalization, with `converged` `TRUE`, when one of the two tolerances is met; it ends with `converged` `FALSE` when s_{q-1} has at most one extremum or after `maxIterations` normalizations. With Q normalizations, the product function is $a_1 \cdots a_Q s_Q$ and $r_p = r_{p-1} - PF_p$. Before extraction p the decomposition ends with `"terminalResidue"` when r_{p-1} has at most one extremum and with `"modeLimit"` when $p - 1$ equals `maxModes`. A product function combines an accumulated amplitude envelope with a normalized oscillation; this construction does not impose the IMF envelope conditions of EMD.

In the first iteration a_1 and $s_0 = r_{p-1}$ carry the units of the signal, so both tolerances compare magnitudes in those units with dimensionless quantities; the result depends on the unit only when the initial magnitude approaches 1 in that unit.

Value

A list with

`modes`	Double matrix, samples by product functions.
`residue`	Numeric vector, the final residue.
`iterations`	Integer vector, the number Q of normalizations of each product function.
`converged`	Logical vector by product function.
`stopReason`	`"terminalResidue"` or `"modeLimit"`.

Conditions

A local magnitude that is not positive leaves the normalization undefined and stops the function with an error of class `xemdZeroEnvelope`.

References

Smith, J. S. (2005). The local mean decomposition and its application to EEG perception data. *Journal of the Royal Society Interface*, 2(5), 443–454. doi:10.1098/rsif.2005.0058

The local levels and the extraction of a product function are adapted from LMD 1.2.1 (Shubhra Prakash), revision ec11b721ec7b05f0137eeb67a619b28f62ef85a2, Apache License (>= 2); see the NOTICE file of the installed package.

See Also

`decomposeIF`, `decomposeEMD`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
OUT <- decomposeLMD(Signal, maxIterations = 200, envelopeTolerance = 0.01,
                    changeTolerance = 0.01, maxSmoothPasses = 12,
```

```
                    maxModes = 8)
dim(OUT$modes)
OUT$iterations
OUT$stopReason
```

`decomposeMaskedEMD` *Masked Empirical Mode Decomposition*

Description

Decomposes a signal by successive extraction in which each mode is the average first mode of the residue plus sinusoidal masks of several phases, under one of three mask rules: the opposite sine pair of 2005, dyadic masks (EMD-DMS) and uniformly distributed phases (UPEMD).

Usage

```
decomposeMaskedEMD(signal, samplingInterval, masking, sifting)
```

Arguments

`signal`	Numeric vector of finite real values without `dim`, in the units of the quantity.
`samplingInterval`	Sampling interval in seconds, a finite number greater than zero, without default. Mask frequencies are in Hz.
`masking`	Named list with the field `rule` and exactly the fields of that rule, without defaults: `"pair"` `frequency`, in Hz, in $(0, 1/(2\Delta t)]$, for instance the value of `estimateMaskFrequency`; `amplitude`, greater than zero, in the units of the signal. `"dyadic"` `firstFrequency`, in Hz, in $(0, 1/(2\Delta t)]$; `amplitudeRatio`, the dimensionless $V_m > 0$. The example of the EMD-DMS article uses $V_m = 10$. `"uniformPhase"` `amplitudeRatio`, the dimensionless $\varepsilon_0 > 0$; `phaseCount`, the whole number $P \geq 2$ of phases. The package tests use $\varepsilon_0 = 1$ and $P = 4$.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected.

Details

From $r_0 = x$, before stage k the decomposition ends, in this order, with `"terminalResidue"` when r_{k-1} has at most one extremum, with `"modeLimit"` when $k - 1$ equals `sifting$maxModes` or the schedule of masks is exhausted, and with `"insufficientExtrema"` when the envelopes of r_{k-1} cannot be built. Otherwise the masks of stage k are $a_k \cos(2\pi f_k n \Delta t + \theta_p)$, $n = 0, \ldots, N-1$, with the first sample as phase origin, and the mode is the average of the first sifted component of r_{k-1} plus each mask,

$$c_k = \frac{1}{P} \sum_{p=1}^{P} E_1 \left(r_{k-1} + w_p \right), \qquad r_k = r_{k-1} - c_k.$$

A stage is retained unless one of its phases has no first mode or loses its envelopes during sifting; then the decomposition stops before adding the stage mean or subtracting it from the residue. A phase that reaches the iteration limit is kept and flags the stage `FALSE`. The phases and divisor are unchanged. The rules fix the phases, frequencies and amplitudes:

`"pair"` Phases $-\pi/2$ and $\pi/2$, so the masks are $\pm a \sin(2\pi f n \Delta t)$ and the mode is $(E_1(x+w) + E_1(x-w))/2$. The schedule has one stage: the decomposition ends with `"modeLimit"` after one extraction unless the residue is terminal.

`"dyadic"` Phases 0 and π; $f_k = 2^{1-k} f_1$; amplitude V_m times the mean absolute value of the maxima of the residue.

`"uniformPhase"` Phases $2\pi p/P$, $p = 0, \ldots, P-1$; $f_k = 1/(2^k \Delta t)$; amplitude ε_0 times the standard deviation of the residue with divisor N.

The masks are deterministic: the function draws no random numbers.

Value

A list with

`modes` Double matrix, samples by modes.

`residue` Numeric vector, the final residue.

`iterations` Integer vector by mode, the sum of the subtractions of the extractions of the stage.

`converged` Logical vector by mode: `TRUE` when every extraction of the stage accepted; `NA` when all completed fixed sifting. Interrupted stages are reported in `termination`.

`stopReason` `"terminalResidue"`, `"modeLimit"`, `"insufficientExtrema"` or `"incompleteExtraction"`, the last when at least one phase fails.

`maskFrequencies` Numeric vector, the mask frequency of each mode in Hz.

`termination` Data frame with `mode`, the next stage index after the retained modes; `phase`, its one-based index in the phase sequence; `iterations`, subtractions of that failed phase; and `stopReason`, `"terminalResidue"`, `"modeLimit"`, `"insufficientExtrema"` or `"iterationLimit"`. One row is returned for each phase of the interrupted stage that did not accept, including the phases that reached the iteration limit. If the decomposition stops before a stage starts, one row carries its `stopReason`, `phase = NA` and zero iterations.

References

Deering, R. and Kaiser, J. F. (2005). The use of a masking signal to improve empirical mode decomposition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 4, 485–488.

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

estimateMaskFrequency, decomposeIteratedMaskedEMD, decomposeEMD

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = 3)
# Uniform phases, with one sample per second.
OUT <- decomposeMaskedEMD(Signal, samplingInterval = 1,
                          masking = list(rule = "uniformPhase",
                                         amplitudeRatio = 1, phaseCount = 4),
                          sifting = Sifting)
OUT$maskFrequencies
OUT$stopReason

# The pair of 2005 at the estimated mask frequency.
Frequency <- estimateMaskFrequency(Signal, samplingInterval = 1,
                                   sifting = Sifting)
Pair <- decomposeMaskedEMD(Signal, samplingInterval = 1,
                           masking = list(rule = "pair", frequency = Frequency,
                                          amplitude = sqrt(mean((Signal -
                                              mean(Signal))^2))),
                           sifting = Sifting)
dim(Pair$modes)
Pair$stopReason
```

`decomposeMEMD` *Multivariate Empirical Mode Decomposition*

Description

Decomposes a signal of two or more channels by successive extraction of vector modes, with vector envelopes built at the maxima of its projections onto directions of the channel space.

Usage

```
decomposeMEMD(signal, directionCount = 64, sifting)
```

Arguments

`signal`	Numeric matrix of finite real values, samples by channels, with two or more columns, as composed by the caller and without standardization.
`directionCount`	Whole number $D \geq 1$ of Hammersley points; default `64`, common to Rlibeemd and emdr. The projections use the $2D$ unit vectors formed by those points and their antipodes.

`sifting` Named list of sifting controls, as in `decomposeEMD`, with the rules `"fixed"`, `"consecutive"` and `"doubleThreshold"` and `boundary = "reflection"` only. A missing `stopRule` is `"doubleThreshold"`, whose `thresholds`, `fraction` and `maxSift` take the defaults of `decomposeEMD`; its extra field `amplitudeRule`, `"globalMean"` or `"envelopePairs"` (default), selects the amplitude of the ratio. Under `"consecutive"`, `consecutiveCount` (a count, default 2) and `maxSift` (default 1000, at least `consecutiveCount`) apply. The default rule, the default amplitude and the defaults of `"consecutive"` are those of memd.m by Rehman and Mandic.

Details

The directions map D Hammersley points, built from the radical inverses in the bases of the first $C-1$ primes, to unit vectors $\mathbf{v}_d$ through the angles of `get.projections` of emdr 1.2.3 (Pierre Masselot); their antipodes complete $2D$ directions. For each direction, $\mathbf{e}_d$ interpolates the vector samples at the maxima of the projection $p_d = \mathbf{v}_d^T\mathbf{x}$, extended by reflection, and the sifting subtracts the mean of the $2D$ envelopes,

$$\mathbf{h}_{s+1} = \mathbf{h}_s - \frac{1}{2D}\sum_{d=1}^{2D}\mathbf{e}_d[\mathbf{h}_s].$$

The adaptive rules act on the projections. `"consecutive"` accepts when every projection keeps $|E-Z| \le 1$ in `consecutiveCount` successive states after a subtraction, the run restarting when one does not. `"doubleThreshold"` accepts a state, the signal included, whose projections all keep $|E-Z| \le 1$ and whose ratio $|\mathbf{m}|/a$ meets the double threshold of `decomposeEMD`; a is the mean distance of the envelopes to their mean with `"globalMean"`, or the mean half distance between the envelopes of antipodal directions with `"envelopePairs"`. Before extraction k the decomposition ends with `"terminalResidue"` when every projection of $\mathbf{r}_{k-1}$ has at most one extremum, with `"modeLimit"` when $k-1$ equals `sifting$maxModes`, and with `"insufficientExtrema"` when a direction has no envelope. One extraction serves all the channels, so they share the number of modes.

Value

A list with

`modes` Double array, samples by modes by channels, in the order of the columns of `signal`.

`residue` Numeric matrix, samples by channels.

`iterations` Integer vector, the subtractions of each mode.

`converged` Logical vector by mode, as in `decomposeEMD`.

`stopReason` `"terminalResidue"`, `"modeLimit"` or `"insufficientExtrema"`.

References

Rehman, N. and Mandic, D. P. (2010). Multivariate empirical mode decomposition. *Proceedings of the Royal Society A*, 466, 1291–1302. doi:10.1098/rspa.2009.0502

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

decomposeNAMEMD, decomposeBEMD, decomposeMVMD, decomposeEMD

Examples

```
Time <- 2 * pi * (0:255) / 256
z <- cos(0.3 * Time) * exp(2i * Time) +
    0.3 * abs(sin(2.3 * Time)) * exp(17i * Time)
Signal <- cbind(Re(z), Im(z))
OUT <- decomposeMEMD(Signal, directionCount = 64,
                     sifting = list(stopRule = "fixed", fixedSift = 10,
                                    boundary = "reflection", maxModes = 3))
dim(OUT$modes)
OUT$stopReason
```

---

decomposeMVMD *Multivariate Variational Mode Decomposition*

Description

Decomposes the channels of a sampled multichannel signal into modes whose channel coordinates share one center frequency.

Usage

```
decomposeMVMD(signal, samplingInterval, initialCenters, alpha, tau,
              fixedZeroCenter, tolerance, maxIterations = 499)
```

Arguments

signal	Numeric matrix of finite values, samples by channels, with at least one row and one column, as composed by the caller and without standardization.
samplingInterval	Sampling interval Δt in seconds, a finite number greater than zero.
initialCenters	Numeric vector of initial center frequencies in Hz, each in $[0, 1/(2\Delta t)]$; its length is the number of modes K. No default.
alpha	Penalty $\alpha > 0$ in s^2, as in decomposeVMD. No default; PySDKit takes it on cycles per sample without the factor 2.
tau	Dual ascent step $\tau \geq 0$; `0` solves the penalized problem with zero multipliers. No default.
fixedZeroCenter	`TRUE` keeps the first center at 0, while its channel coordinates are still updated, and requires `initialCenters[1] == 0`. No default.

`tolerance`	Threshold $\epsilon > 0$ of the relative change summed over modes and channels. No default; the `1e-7` of PySDKit belongs to another measure.
`maxIterations`	Whole number ≥ 1 of sweeps; default `499`, the limit of 500 states of the authors' MVMD.m 1.0.1.

Details

Each channel is extended and transformed as in `decomposeVMD`. In each sweep the coordinate c of mode k is

$$\hat{u}_{k,c} = \frac{\hat{x}_c - \sum_{i \neq k} \hat{u}_{i,c} + \hat{\lambda}_c/2}{1 + 2\alpha(\omega - \omega_k)^2},$$

with a denominator common to the channels; the new center weights $|\omega|$ by the power summed over the channels, and the multiplier of each channel is updated with step τ. The iteration stops with `"tolerance"` when $\sum_k \sum_c \|\hat{u}_{k,c}^{(n+1)} - \hat{u}_{k,c}^{(n)}\|^2 / \|\hat{u}_{k,c}^{(n)}\|^2 < \epsilon$, which cannot happen before the second sweep, or with `"iterationLimit"` after `maxIterations` sweeps. A coordinate without energy in the previous sweep leaves its ratio undefined, so the test fails until `maxIterations`. The channels keep the order received. The joint bandwidth formulation does not impose the geometrical IMF conditions of EMD. A shared spectral center does not require identical amplitudes or instantaneous frequencies across channels.

Value

A list with

`modes`	Double array, samples by modes by channels.
`fitResidual`	Numeric matrix, samples by channels, $x_c - \sum_k u_{k,c}$.
`centers`	Numeric vector of the K shared centers in Hz.
`iterations`	Integer vector repeating, for every mode, the number of sweeps.
`converged`	Logical vector repeating, for every mode, the result of the joint stopping test.
`stopReason`	`"tolerance"` or `"iterationLimit"`.

Conditions

A mode without energy over all its channels leaves its center undefined and stops the function with an error of class `xemdZeroEnergy`.

References

ur Rehman, N. and Aftab, H. (2019). Multivariate Variational Mode Decomposition. arXiv:1907.04509.

See Also

`decomposeVMD`, `decomposeSVMD`, `decomposeTSL`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
Signal <- cbind(cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
                  0.0625 * cos(2 * pi * 288 * Time),
                cos(2 * pi * 2 * Time),
                0.25 * cos(2 * pi * 24 * Time))
OUT <- decomposeMVMD(Signal, samplingInterval = SamplingInterval,
                     initialCenters = (0:2) / (6 * SamplingInterval),
                     alpha = 2000 * SamplingInterval^2 / (8 * pi^2),
                     tau = 0.5, fixedZeroCenter = FALSE, tolerance = 1e-6,
                     maxIterations = 98)
dim(OUT$modes)
OUT$centers
OUT$stopReason
```

`decomposeNAMEMD` *Noise-Assisted Multivariate Empirical Mode Decomposition*

Description

Decomposes a signal together with supplied noise channels by one multivariate decomposition and keeps the coordinates of the signal.

Usage

```
decomposeNAMEMD(signal, noiseChannels, directionCount = 64, sifting)
```

Arguments

`signal`	Numeric matrix of finite real values, samples by channels, with one or more columns.
`noiseChannels`	Numeric matrix of finite real values with the rows of `signal` and $Q \geq 1$ columns, in the units of the signal, supplied by the caller: the function draws no random numbers. emdr draws its noise with standard deviation 0.02 on standardized channels.
`directionCount`	Whole number of Hammersley points, as in `decomposeMEMD`; default `64`.
`sifting`	Named list of sifting controls, as in `decomposeMEMD`.

Details

The function runs `decomposeMEMD` on the augmented matrix $\mathbf{y} = [\mathbf{x}, \mathbf{w}]$ and keeps the first C coordinates of every mode, $\mathbf{c}_k = P_x \tilde{\mathbf{c}}_k$, and of the residue. Projections, stopping, iterations and termination are those of the augmented decomposition. In `decomposeTSL` the noise channels are drawn by group with `noiseChannelCount` and scaled by `noiseRatio` times the reference deviation of the group.

Value

A list with the fields of decomposeMEMD; modes has the C channels of signal and residue its C columns.

References

ur Rehman, N. and Mandic, D. P. (2011). Filter Bank Property of Multivariate Empirical Mode Decomposition. *IEEE Transactions on Signal Processing*, 59(5), 2421–2426. doi:10.1109/TSP.2011.2106779

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

decomposeMEMD, decomposeTSL

Examples

```
Time <- 2 * pi * (0:511) / 512
z <- cos(0.3 * Time) * exp(2i * Time) +
    0.3 * abs(sin(2.3 * Time)) * exp(17i * Time)
Signal <- cbind(Re(z), Im(z))
set.seed(628)
# Two noise channels with 0.02 times the reference deviation of the signal.
Noise <- 0.02 * sqrt(mean(apply(Signal, 2, stats::sd)^2)) *
    matrix(stats::rnorm(512 * 2), nrow = 512)
OUT <- decomposeNAMEMD(Signal, noiseChannels = Noise, directionCount = 64,
                       sifting = list(stopRule = "fixed", fixedSift = 1,
                                      boundary = "reflection",
                                      maxModes = 3))
dim(OUT$modes)
dim(OUT$residue)
```

---

decomposeSVMD *Successive Variational Mode Decomposition*

Description

Extracts variational modes one at a time from the whole signal, keeping the centers of the modes already extracted fixed, and stops after a known number of modes or when the residual power meets a known noise power.

Usage

```
decomposeSVMD(signal, samplingInterval, initialCenter = 0, alpha, maxAlpha,
              tau, tolerance, maxIterations = 299, stopping)
```

Arguments

`signal`	Non-empty numeric vector of finite values without `dim`, in the units of the quantity.
`samplingInterval`	Sampling interval Δt in seconds, a finite number greater than zero.
`initialCenter`	Initial center of every mode in Hz, in $[0, 1/(2\Delta t)]$; default `0`.
`alpha`	Initial penalty $\alpha > 0$ in s^2. No default.
`maxAlpha`	Final penalty, at least `alpha`, in s^2; equal to `alpha` gives a fixed penalty. No default. PySDKit uses 10 and 20000 for its denominator $1 + 2\alpha_\nu(\nu - \nu_L)^2$ on cycles per sample ν; a value α_ν there corresponds to $\alpha_\nu \Delta t^2/(4\pi^2)$.
`tau`	Dual ascent step $\tau \geq 0$. No default; PySDKit uses 0.
`tolerance`	Threshold $\epsilon_1 > 0$ of the relative change of the mode spectrum between inner iterations. No default; PySDKit uses `1e-6`.
`maxIterations`	Whole number ≥ 1 of inner iterations for each value of the penalty; default `299`. The defaults of `initialCenter` and `maxIterations` are those of the authors' svmd.m 1.1.1.
`stopping`	Named list with the field `stopRule` and exactly the fields of that rule: `"modeCount"` `modeCount`, the whole number of modes to extract, without default. `"noisePower"` `noisePower`, the noise power $\sigma^2 > 0$ in squared units of the signal; `noiseTolerance`, the dimensionless $\epsilon_2 > 0$; `maxModes`, a whole number ≥ 1 or `Inf`, default `Inf`.

Details

The signal is extended and transformed as in `decomposeVMD`. Mode L is extracted from the whole signal with the centers $\omega_1, \ldots, \omega_{L-1}$ of the previous modes fixed. Its penalty starts at `alpha` and doubles while below `maxAlpha`; when doubling would exceed `maxAlpha`, one last computation runs at `maxAlpha`. Each penalty restarts the inner iteration from the spectrum and center reached, with a zero multiplier and the penalty $H_{L-1}(\omega) = \sum_{i<L} 1/(\alpha^2(\omega - \omega_i)^4)$. With $A = \alpha^2(\omega - \omega_L)^4$, the inner update is

$$\hat{u}_L \leftarrow \frac{\hat{x} + A\hat{u}_L + \hat{\lambda}/2}{(1 + A)\,(1 + 2\alpha(\omega - \omega_L)^2 + H_{L-1})},$$

followed by the new center and the dual ascent, and it stops when the relative change of $\hat{u}_L$ is below `tolerance` or after `maxIterations`. Extraction stops with `"modeCount"` after `modeCount` modes; with `"noisePower"` when $|\sigma^2 - P_L|/\sigma^2 < \epsilon_2$, P_L the mean square of the residual on the record; or with `"modeLimit"` after `maxModes` modes without meeting it. With `maxModes = Inf` the extraction under `"noisePower"` continues until the condition holds. The modes keep the extraction order. Their bandwidth formulation does not impose the geometrical IMF conditions of EMD.

Value

A list with

`modes`	Double matrix, samples by modes, in extraction order.

`fitResidual`	Numeric vector $x - \sum_L u_L$.
`centers`	Numeric vector of the centers in Hz.
`iterations`	Integer vector by mode, the inner iterations of all its penalties.
`converged`	Logical vector by mode, `TRUE` when every inner loop of the mode met the change test.
`stopReason`	`"modeCount"`, `"noisePower"` or `"modeLimit"`.

Conditions

A mode without spectral energy leaves its center undefined and stops the function with an error of class `xemdZeroEnergy`.

References

Nazari, M. and Sakhaei, S. M. (2020). Successive variational mode decomposition. *Signal Processing*, 107610. doi:10.1016/j.sigpro.2020.107610

See Also

`decomposeVMD`, `decomposeMVMD`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
OUT <- decomposeSVMD(Signal, samplingInterval = SamplingInterval,
                     initialCenter = 0,
                     alpha = 10 * SamplingInterval^2 / (4 * pi^2),
                     maxAlpha = 20000 * SamplingInterval^2 / (4 * pi^2),
                     tau = 0, tolerance = 1e-6, maxIterations = 299,
                     stopping = list(stopRule = "modeCount", modeCount = 5))
OUT$centers
OUT$stopReason
```

---

decomposeTSL *Decomposition of Time-Series Records*

Description

Decomposes the series of a long table of records, grouped by record and quantity, with one method of the package, and returns homogeneous tables for all the methods.

Usage

```
decomposeTSL(TSL, method, control, by)
```

Arguments

TSL	Long table, a data frame or a **data.table**, with rows, unique column names and the columns t, the time in seconds; s, the sample; ID, the quantity, such as "AT", "VT" or "DT"; and OCID, the component. t and s are finite numbers; ID and OCID are character without NA. The other columns are metadata; they cannot take the names mode, iterations, converged, stopReason, innerStopReason, frequency or boundary. Also reserved for the selected method are realization for EEMD; realization and sign for CEEMD; phase for MaskedEMD; and direction for BEMD. TSL is not modified.
method	One of listDecompositions().
control	Named list of arguments of method, common to all the groups, except those the operation derives for each group: signal, samplingInterval, noise and noiseChannels. The ensembles take realizationCount in place of noise; decomposeNAMEMD takes noiseChannelCount and noiseRatio, a finite number ≥ 0, in place of noiseChannels. Controls in the units of the signal are rejected, because AT, VT and DT do not share units: the rule "pair" of decomposeMaskedEMD and any stopping$stopRule of decomposeSVMD other than "modeCount".
by	Character vector naming the metadata columns that identify a record, or character() for a single record.

Details

A group is a record and an ID; its OCID are the channels of the group and must share one time grid. Groups follow by ascending and the IDs "AT", "VT" and "DT" first, then the other IDs ascending; the OCID of a group ascend. The order of the rows of TSL does not intervene. A series needs two samples or more, a time cannot repeat within a series, and the metadata other than by must be constant within each record, ID and OCID.

The grid of a group is uniform when diff(t) equals the mean step $\Delta t = (t_N - t_1)/(N-1)$ under all.equal; it is then used unchanged, with samplingInterval Δt. A grid that is not uniform is regularized with one warning per group: the step becomes $1/F_s$, with F_s the smallest $2^a 5^b$ not below $1/\Delta t$; the grid is $t_1 + k/F_s$, and each OCID is interpolated on it by stats::splinefun(method = "monoH.FC"). The table signal holds the series actually decomposed.

Scalar methods decompose each OCID separately with the same controls. The coupled methods decompose the matrix of the OCID of a group in ascending order, without standardization: decomposeBEMD needs exactly two OCID and decomposeMEMD at least two; their iterations, converged, stopReason and centers are common and repeat for every OCID. The ensembles draw realizationCount unit normal realizations for each OCID from the random stream of the caller, in the order group, OCID, realization. decomposeNAMEMD draws noiseChannelCount channels for each group and scales them by noiseRatio times $\sqrt{\text{mean}(s_c^2)}$, s_c the sample deviations of the OCID of the group; the noise channels give no rows. A control valid for one group and not for another, such as a center above the Nyquist frequency of a group, stops the operation. Units are not inferred, validated or converted; the metadata are carried to the output.

Channels are coupled within a record and an ID. The operation does not impose a kinematic relationship between separate AT, VT and DT groups. Equal mode indices across those groups do not establish corresponding physical contributions.

Value

A named list of **data.table** tables, each with the by columns, the other metadata, `OCID` and `ID`, followed by

`signal`	`t`, `s`: the series decomposed, on the output grid.
`modes`	`mode`, an integer from 1, `t`, `s`. Each `OCID` keeps its own number of modes, without padding; an `OCID` without modes has no rows.
`residue`, `fitResidual`, `meanNoise`	`t`, `s`, for the fields that the method returns.
`diagnostics`	`mode`, `iterations`, `converged`, when the method returns iterations.
`termination`	`stopReason`, and `innerStopReason` for `decomposeIteratedMaskedEMD`, when the method returns a termination. EEMD, CEEMD, MaskedEMD and BEMD carry the columns and rows of their native `termination` data frame, with the series keys added; the BEMD row repeats for both channels. In these rows `mode` identifies the next attempted extraction, not a returned mode; `diagnostics` continues to describe only returned modes.
`centers`, `maskFrequencies`	`mode`, `frequency` in Hz.
`boundaries`	`boundary`, `frequency` in Hz.

Rows follow the by columns, `OCID`, `ID`, `mode` or `boundary`, and `t`.

References

The regularization of the grid is adapted from `.regularize` and `.rationalize` of gmsp 0.8.0, copyright (c) 2026 Alejandro Verri Kozlowski, MIT License: gmsp starts from the smallest difference of the times and returns the grid from zero, whereas here the step starts from the mean and the grid keeps the origin t_1. See the NOTICE file of the installed package.

See Also

`listDecompositions`, `decomposeEMD`, `decomposeMVMD`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
# The same series under three IDs and three components.
TSL <- data.table::rbindlist(lapply(c("AT", "VT", "DT"), function(id) {
    data.table::rbindlist(lapply(c("H1", "H2", "UP"), function(ocid) {
        data.table::data.table(ID = id, OCID = ocid, t = Time, s = Signal)
    }))
}))
OUT <- decomposeTSL(TSL, method = "decomposeEMD",
                    control = list(sifting = list(stopRule = "doubleThreshold")),
                    by = character())
names(OUT)
OUT$termination
```

```
head(OUT$modes)

# A coupled method: three channels of one record decomposed together.
SamplingInterval <- 1 / 1000
TimeV <- seq_len(1000) * SamplingInterval
Series <- cbind(cos(2 * pi * 2 * TimeV) + 0.25 * cos(2 * pi * 24 * TimeV) +
                  0.0625 * cos(2 * pi * 288 * TimeV),
                cos(2 * pi * 2 * TimeV), 0.25 * cos(2 * pi * 24 * TimeV))
Record <- data.table::data.table(ID = "AT",
                                 OCID = rep(c("H1", "H2", "UP"), each = 1000),
                                 t = rep(TimeV, 3), s = as.vector(Series))
Coupled <- decomposeTSL(Record, method = "decomposeMVMD",
                        control = list(initialCenters = (0:2) / (6 * SamplingInterval),
                                       alpha = 2000 * SamplingInterval^2 / (8 * pi^2),
                                       tau = 0.5, fixedZeroCenter = FALSE,
                                       tolerance = 1e-6, maxIterations = 98),
                        by = character())
Coupled$centers
```

`decomposeTVFEMD` *Time-Varying Filter Empirical Mode Decomposition*

Description

Decomposes a signal by successive extraction of components, each obtained by sifting with a local mean that is a B-spline approximation on knots placed by a time-varying cutoff frequency (TVF-EMD).

Usage

```
decomposeTVFEMD(signal, bandwidthThreshold, splineOrder,
                intermittencyThreshold, maxSift, maxModes = Inf)
```

Arguments

`signal`	Numeric vector of finite real values without `dim`, in the units of the quantity. Positions are sample indices and the cutoff is in radians per sample.
`bandwidthThreshold`	Dimensionless threshold $\xi > 0$ of the relative instantaneous bandwidth. No default; the original study uses 0.1.
`splineOrder`	Whole number $n \geq 1$, the order of the B-spline (degree $n - 1$). No default; the original study uses 28.
`intermittencyThreshold`	Dimensionless threshold $\rho > 0$ of the intermittency test of the cutoff. No default; the original study uses 0.25.
`maxSift`	Whole number ≥ 1 of subtractions of one component. No default; pytvfemd stops after five.
`maxModes`	Whole number ≥ 1 or `Inf`, the limit of components; default `Inf`.

Details

Before extraction k the decomposition ends with `"terminalResidue"` when r_{k-1} has fewer than four extrema, maxima plus minima, and with `"modeLimit"` when $k-1$ equals `maxModes`. A component is obtained by sifting,

$$h_{s+1} = h_s - m_s,$$

where m_s is the periodic least-squares B-spline approximation of order n of h_s on knots at the extrema of $\cos\varphi$, φ the cumulative realigned cutoff. Each iteration extends h_s by symmetric reflection of $\mathrm{round}(N/2)$ samples per side and restores the record afterwards. The cutoff follows from the amplitude A and the frequency of the analytic signal, interpolated at the minima and at the maxima of A; values above 0.9π are lowered to 0.9π, undefined or non-positive values are interpolated, and the zones marked by intermittency events, where the relative variation of the cutoff between consecutive maxima of the signal exceeds ρ, are replaced by interpolation.

The stopping rule aggregates the relative instantaneous bandwidth θ over the samples from $\mathrm{round}(0.05N)$ to $\mathrm{round}(0.95N)$: h_s is accepted when the mean of θ is below $\xi(1+(s+1)/4)$, from $s=1$ in the first component and from $s=0$ in the later ones, or when the samples with $\theta > \xi$ are fewer than $0.2N$. This aggregation of pytvfemd 0.3.14 replaces the local criterion of the monograph. `converged` is `FALSE` when `maxSift` subtractions end without acceptance or when the cutoff or the knots cannot be built. Acceptance refers to the aggregated rule above; it does not establish the pointwise narrow-band condition. Neither that rule nor a returned component guarantees the two classical IMF conditions.

Value

A list with

`modes`	Double matrix, samples by components.
`residue`	Numeric vector, the final residue.
`iterations`	Integer vector, the subtractions of each component.
`converged`	Logical vector by component.
`stopReason`	`"terminalResidue"` or `"modeLimit"`.

References

The stopping rule and the reflection of the sifting, the end of the decomposition, the interpolation, the bounds and the realignment of the cutoff, the relative bandwidth, the knots and the periodic B-spline approximation are adapted from pytvfemd 0.3.14 (Stefano Bianchi and Alessandro Longo), GPL-3; see the NOTICE file of the installed package.

See Also

`decomposeEMD`, `computeInstantaneousAttributes`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
```

```
OUT <- decomposeTVFEMD(Signal, bandwidthThreshold = 0.1, splineOrder = 28,
                       intermittencyThreshold = 0.25, maxSift = 5)
dim(OUT$modes)
OUT$iterations
OUT$stopReason
```

`decomposeVMD` *Variational Mode Decomposition*

Description

Decomposes a sampled signal into a given number of modes, each concentrated around a center frequency, by alternating updates of the mode spectra, the centers and a multiplier.

Usage

```
decomposeVMD(signal, samplingInterval, initialCenters, alpha, tau,
             fixedZeroCenter, tolerance, maxIterations)
```

Arguments

`signal`	Non-empty numeric vector of finite values without `dim`, in the units of the quantity.
`samplingInterval`	Sampling interval Δt in seconds, a finite number greater than zero.
`initialCenters`	Numeric vector of initial center frequencies in Hz, each in $[0, 1/(2\Delta t)]$; its length is the number of modes K. No default.
`alpha`	Penalty $\alpha > 0$ of the bandwidth, in s^2 because it multiplies squared angular frequencies in rad/s. No default. sktime and VMDecomp take α on cycles per sample without the factor 2: their value α_ν corresponds to $\alpha = \alpha_\nu \Delta t^2/(8\pi^2)$.
`tau`	Dual ascent step $\tau \geq 0$, dimensionless. `0` solves the penalized problem with a zero multiplier; a positive value the constrained problem. No default; the package tests use 0 and 0.5.
`fixedZeroCenter`	`TRUE` keeps the first center at 0 and requires `initialCenters[1] == 0`; `FALSE` updates every center. No default.
`tolerance`	Threshold $\epsilon > 0$ of the relative change of the mode spectra between sweeps, dimensionless. No default; the `1e-7` of sktime belongs to an absolute criterion, not to this relative change.
`maxIterations`	Whole number ≥ 1 of sweeps. No default; sktime and VMDecomp allocate 500 states.

Details

The record of N samples is extended to $T = 2N$ samples by reflecting $\lceil N/2 \rceil$ samples at the start and the rest at the end, and transformed with the unnormalized discrete Fourier transform; the grid holds $|\omega|$ in rad/s. Sweeps start from zero mode spectra and multiplier. In each sweep mode k, updated after modes $1, \ldots, k-1$, is

$$\hat{u}_k = \frac{\hat{x} - \sum_{i \neq k} \hat{u}_i + \hat{\lambda}/2}{1 + 2\alpha(\omega - \omega_k)^2},$$

its center becomes the power-weighted mean of $|\omega|$, and the sweep closes with $\hat{\lambda} \leftarrow \hat{\lambda} + \tau(\hat{x} - \sum_k \hat{u}_k)$. The iteration stops with `"tolerance"` when $\sum_k \|\hat{u}_k^{(n+1)} - \hat{u}_k^{(n)}\|^2 / \|\hat{u}_k^{(n)}\|^2 < \epsilon$, which cannot happen before the second sweep, or with `"iterationLimit"` after `maxIterations` sweeps; the last sweep is returned. The modes are the inverse transforms restricted to the record. With `tau = 0` the penalized formulation permits a nonzero fit residual. VMD defines its modes through a bandwidth objective, without imposing the geometrical IMF conditions of EMD.

Value

A list with

`modes`	Double matrix, samples by modes, on the input grid.
`fitResidual`	Numeric vector $x - \sum_k u_k$.
`centers`	Numeric vector of the K centers of the last sweep, in Hz.
`iterations`	Integer vector repeating, for every mode, the number of sweeps.
`converged`	Logical vector repeating, for every mode, the result of the joint stopping test.
`stopReason`	`"tolerance"` or `"iterationLimit"`.

Conditions

A mode without spectral energy leaves its center undefined and stops the function with an error of class `xemdZeroEnergy`.

References

Dragomiretskiy, K. and Zosso, D. (2014). Variational Mode Decomposition. *IEEE Transactions on Signal Processing*, 62(3), 531–544. doi:10.1109/TSP.2013.2288675

See Also

`decomposeMVMD`, `decomposeSVMD`, `decomposeTSL`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
OUT <- decomposeVMD(Signal, samplingInterval = SamplingInterval,
```

```
                    initialCenters = (0:2) / (6 * SamplingInterval),
                    alpha = 2000 * SamplingInterval^2 / (8 * pi^2),
                    tau = 0.5, fixedZeroCenter = FALSE, tolerance = 1e-6,
                    maxIterations = 98)
OUT$centers
OUT$stopReason
max(abs(OUT$fitResidual))
```

`describeModule` *Identity of the Installed Module*

Description

Returns the name and the installed version of the package.

Usage

```
describeModule()
```

Value

A list with `package`, the string `"xemd"`, and `version`, the installed version as a string.

See Also

`listOperations`, `listDecompositions`

Examples

```
describeModule()
```

`estimateMaskFrequency` *Mask Frequency of the 2005 Masking*

Description

Estimates the frequency of the masking signal of 2005 from the instantaneous amplitude and frequency of the first sifted component of a signal.

Usage

```
estimateMaskFrequency(signal, samplingInterval, sifting)
```

Arguments

`signal`	Numeric vector of finite real values without `dim`, in the units of the quantity.
`samplingInterval`	Sampling interval in seconds, a finite number greater than zero, without default.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected.

Details

With A_1 and ν_1 the instantaneous amplitude and frequency, computed by `computeInstantaneousAttributes`, of the first sifted component $y_1 = E_1(x)$ without mask, the estimate is

$$\bar{\nu} = \frac{\sum_n A_1[n]\,\nu_1[n]^2}{\sum_n A_1[n]\,\nu_1[n]},$$

with the sums over the samples where ν_1 is defined. The result is the `frequency` of the rule `"pair"` of `decomposeMaskedEMD`.

Value

A number, the mask frequency in Hz.

Conditions

A signal without a first extracted component stops with an error. When the weights $A_1\nu_1$ sum to zero the estimate is undefined, and the function stops with an error of class `xemdZeroEnergy`.

References

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeMaskedEMD`, `computeInstantaneousAttributes`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = 3)
# One sample per second, so the result is in cycles per sample.
estimateMaskFrequency(Signal, samplingInterval = 1, sifting = Sifting)
# The time step of the signal, 9 / 2999 s.
estimateMaskFrequency(Signal, samplingInterval = 9 / 2999, sifting = Sifting)
```

`extractSSTModes` *Components Along the Ridges of a Synchrosqueezed Transform*

Description

Extracts ridges of a synchrosqueezed transform one after another by dynamic programming and reconstructs a component from the coefficients in a band around each ridge.

Usage

```
extractSSTModes(transform, frequency, modeCount, penalty = 1000,
                exclusionWidth = 4, reconstructionWidth, morletCenter)
```

Arguments

`transform`	Numeric or complex matrix of finite values, samples by frequency cells, such as the `transform` of `transformSST`.
`frequency`	Increasing positive frequencies of the cells, one per column of `transform`, such as the `frequency` of `transformSST`.
`modeCount`	Whole number ≥ 1 of ridges. No default; curve_ext_multi of the Synchrosqueezing Toolbox requires the value.
`penalty`	Dimensionless penalty $\lambda \geq 0$ of a jump between cells; default `1000`, the value that curve_ext_multi assigns.
`exclusionWidth`	Whole number $b \geq 0$ of cells on each side of a ridge that are removed before the next extraction; default `4`, the value that curve_ext_multi assigns.
`reconstructionWidth`	Whole number $w \geq 0$ of cells on each side of a ridge in its reconstruction band. No default; a value not below the number of cells gives the complete inversion, as synsq_cwt_iw does without a band.
`morletCenter`	Center $\mu > 0$ of the Morlet wavelet that produced `transform`, in normalized frequency. No default.

Details

With $T_{n,j}$ the coefficients, f_j the frequencies and $z_j = \log_2 f_j$, the cost of a cell is $E_{n,j} = -\log(|T_{n,j}|^2/Q + 10^{-8})$, Q the sum of $|T|^2$ over the matrix. The accumulated cost is $F_{1,j} = E_{1,j}$ and

$$F_{n,j} = E_{n,j} + \min_k \left[F_{n-1,k} + \lambda(z_j - z_k)^2\right]$$

over every cell k; the ridge ends at the first cell of least $F_{N,j}$, and each earlier cell is the first whose residual in that recurrence is below 10^{-8} in modulus. After each ridge C_n, the cells from $C_n - b$ to $C_n + b$, clipped to the cells, take the value $\sqrt{\epsilon}$, ϵ the machine precision, and the next ridge is extracted with the energy recomputed. Component k sums the original coefficients of its band,

$$c_k[n] = \frac{1}{C}\mathrm{Re}\sum_{j=C_{k,n}-w}^{C_{k,n}+w} T_{n,j},$$

over the cells of the band, with $C = C_\psi/(\sqrt{2\pi}\, 2\ln 2)$ and $C_\psi = \int_0^\infty \hat{\psi}(\xi)/\xi \, d\xi$ for the Morlet wavelet of center `morletCenter`. The bands may overlap, the components need not add up to the signal, and no residue is returned. The components are defined by the selected ridge bands, without imposing the IMF conditions of EMD.

Value

Double matrix, samples by components, one column per ridge.

Conditions

A transform of zero energy stops the function with an error of class `xemdZeroEnergy`. A sample where no cell satisfies the recurrence within the tolerance stops it with an error.

References

Daubechies, I., Lu, J. and Wu, H.-T. (2009). Synchrosqueezed wavelet transforms: a tool for empirical mode decomposition. arXiv:0912.2437.

Brevdo, E. and Thakur, G. *Synchrosqueezing Toolbox*, revision 7e9fec0c6c9ed478dafac4479c0658d24fc6d8ef, functions curve_ext (original code by Jianfeng Lu), curve_ext_multi and synsq_cwt_iw, https://github.com/ebrevdo/synchrosqueezing.

The extraction of the ridges is adapted from curve_ext (original author Jianfeng Lu, maintainer Eugene Brevdo) and curve_ext_multi of the Synchrosqueezing Toolbox, as ported in SynchWave 1.1.2, revision 9ac80b2c3e60b417d0ee5567a66e8e64faa1a979 (port by Dongik Jang), and the inversion from SynchWave 1.1.2 (MATLAB original by Eugene Brevdo; R port by Dongik Jang, Hee-Seok Oh and Donghoh Kim), LGPL-2.0-or-later, which `xemd` distributes under GPL-3 as section 3 of the GNU Library General Public License version 2 permits; see the NOTICE file of the installed package.

See Also

`transformSST`, `decomposeEWT`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
SST <- transformSST(Signal, samplingInterval = SamplingInterval,
                    voiceCount = 16, morletCenter = 2 * pi,
                    threshold = sqrt(.Machine$double.eps))
Modes <- extractSSTModes(SST$transform, frequency = SST$frequency,
                         modeCount = 3, reconstructionWidth = 0,
                         morletCenter = 2 * pi)
dim(Modes)
sqrt(colMeans(Modes^2))
```

listDecompositions *Methods of the Record Operation*

Description

Returns the names of the decomposition methods that `decomposeTSL` accepts.

Usage

```
listDecompositions()
```

Details

The scalar decompositions are applied to each `OCID` of a group and the coupled ones (`decomposeBEMD`, `decomposeMEMD`, `decomposeNAMEMD` and `decomposeMVMD`) to the `OCID` of a group together. `transformSST`, the analysis operations, `resiftModes` and `estimateMaskFrequency` have no record operation.

Value

Character vector of function names: the thirteen scalar methods followed by the four coupled ones.

See Also

`decomposeTSL`, `listOperations`

Examples

```
listDecompositions()
```

listOperations *Exported Operations and Their Arguments*

Description

Lists the exported functions of the package with the names of their formal arguments.

Usage

```
listOperations()
```

Details

The list reads the exports of the loaded namespace; it performs no decomposition and reads no files.

Value

A list with one element per exported function, in the order of `sort()` on their names; each element is a list with `name`, the function name, and `arguments`, the names of its formal arguments.

See Also

`describeModule`, `listDecompositions`

Examples

```
Operations <- listOperations()
length(Operations)
Operations[[1]]
```

---

measureDispersion *Dispersion of Ensemble Members*

Description

Computes, sample by sample and mode by mode, the sample variance of the members of an ensemble around their mean, and that sample variance divided by the number of members. The latter estimates the variance of the mean under the assumptions stated below.

Usage

```
measureDispersion(realizations)
```

Arguments

`realizations`	Finite numeric array with three dimensions, samples by modes by realizations, with at least two realizations. The modes of each realization occupy their extraction order, and a mode absent from a realization is filled with zeros.

Details

With M realizations $C^{(m)}$ and their mean $\bar{C}$, the sample variance is

$$s^2 = \frac{1}{M-1} \sum_{m=1}^{M} \left(C^{(m)} - \bar{C} \right)^2$$

and the returned estimate for the variance of the mean is s^2/M. The first describes dispersion between members. Interpreting the second as Monte Carlo precision assumes comparable components and independent realizations with a common mean and a common finite covariance. Opposite perturbations in a complementary pair are dependent; the two signs do not provide independent realizations for this interpretation.

The function does not establish these statistical assumptions or physical correspondence between equal mode indices. Dispersion under perturbations of a fixed record does not quantify all uncertainty in the physical signal, and increasing the number of members does not remove decomposition bias. Attributes of `realizations` other than `dim` do not reach the outputs.

Value

A list with two matrices, samples by modes:

`variance`	The sample variance s^2, in squared units of the signal.
`meanVariance`	The estimate s^2/M, subject to the assumptions in Details.

See Also

`decomposeEEMD`

Examples

```
# Two realizations of four samples: the first has two modes, the second
# one, so its second mode is filled with zeros.
Realizations <- array(c(2, 4, 6, 8, 1, 1, 1, 1,
                        4, 6, 8, 10, 0, 0, 0, 0), dim = c(4, 2, 2))
measureDispersion(Realizations)
```

---

measureDTW *Dynamic Time Warping Distance of Modes*

Description

Computes the dynamic time warping distance between a signal and each of its modes.

Usage

```
measureDTW(signal, modes)
```

Arguments

`signal`	Non-empty finite numeric vector without `dim`.
`modes`	Finite numeric matrix with one row per sample of `signal` and one column per mode. Attributes other than `dim` do not reach the outputs.

Details

With the local cost $d_{ij} = (x_i - c_j)^2$ and the accumulation

$$G_{ij} = d_{ij} + \min\{G_{i-1,j-1}, G_{i-1,j}, G_{i,j-1}\},$$

$G_{00} = 0$ and infinite cost outside the matrix, the distance is $\sqrt{G_{NN}}$, without normalization by the length of the path. The work grows with the square of the number of samples for each mode.

Value

Numeric vector with one distance per mode, in the units of the signal.

See Also

measureFCF, selectModesEnergy

Examples

```
Time <- seq_len(1000) / 1000
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
measureDTW(signal = rowSums(Tones), modes = Tones)
```

---

measureFCF *Fault Correlation Factor of Modes*

Description

Computes the fault correlation factor between a signal and each of its modes, a normalized inner product without removal of means.

Usage

```
measureFCF(signal, modes)
```

Arguments

signal	Finite numeric vector without dim.
modes	Finite numeric matrix with one row per sample of signal and one column per mode. Attributes other than dim do not reach the outputs.

Details

$$\mathrm{FCF}(x, c) = \frac{\sum_n x[n]c[n]}{\sqrt{\sum_n x[n]^2 \sum_n c[n]^2}}.$$

The means are not removed, so the factor differs from a correlation of centered vectors when the inputs have a nonzero mean.

Value

Numeric vector with one factor per mode.

Conditions

A signal or a mode of zero norm stops the function with an error of class xemdZeroEnergy.

See Also

measureDTW, selectModesEnergy

Examples

```
Time <- seq_len(1000) / 1000
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
measureFCF(signal = rowSums(Tones), modes = Tones)
```

---

measureModesTSL *Metrics of the Modes of Time-Series Records*

Description

Measures every mode and every series of the tables of decomposeTSL, for any of its methods: frequency, bandwidth, energy, energy fraction and mean period of each mode, and number of modes and energy fractions of the remainder and the energy difference described by crossFraction for each series.

Usage

```
measureModesTSL(decomposition)
```

Arguments

decomposition	The list of tables of decomposeTSL, with signal, modes and at most one of residue and fitResidual. Its keys are the columns of signal other than t and s: the by columns, the other metadata, OCID and ID; they cannot take the names mode, frequency, bandwidth, energy, energyFraction, meanPeriod, modeCount, residueFraction or crossFraction. t and s are finite numbers and mode holds whole numbers from 1. Each series of signal has two samples or more at distinct times, and each mode and remainder of a series holds exactly the times of that series. decomposition is not modified.

Details

A mode c of a series x with N samples uses the step $\Delta t = (t_N - t_1)/(N-1)$ of the grid of its group, with its samples ordered by t; the order of the rows does not intervene. Its spectrum is that of the VMD family: the mode is extended to $2N$ samples by reflecting $\lceil N/2 \rceil$ samples at the start and the others at the end, P_j is the squared modulus of the unnormalized discrete Fourier transform of the extension and $|\omega_j| = 2\pi \min(j, 2N-j)/(2N\Delta t)$, $j = 0, \ldots, 2N-1$. Then frequency is the power centroid

$$f = \frac{\sum_j |\omega_j| P_j}{2\pi \sum_j P_j}$$

and bandwidth the power deviation around it on the same grid,

$$b = \left[\frac{\sum_j (|\omega_j|/2\pi - f)^2 P_j}{\sum_j P_j} \right]^{1/2},$$

both in Hz. `frequency` equals the `centers` of `decomposeVMD` and `decomposeSVMD`, except a center fixed at zero, which is not a centroid; in `decomposeMVMD` it is the centroid of each channel, not the common center. A mode of zero energy determines no centroid: its `frequency` and `bandwidth` are `NA`. `energy` is $\Delta t \sum_n c[n]^2$, `energyFraction` is $\sum c^2 / \sum x^2$, and `meanPeriod` is $N\Delta t$ over the number of maxima of the mode, or $N\Delta t$ without maxima.

For each series, `modeCount` is the number of its modes; `residueFraction` is $\sum r^2 / \sum x^2$, with r its `residue` or `fitResidual`, and zero when the method returns neither, as `decomposeEWT`; and `crossFraction` is

$$1 - \frac{\sum_k \sum c_k^2 + \sum r^2}{\sum x^2},$$

the input-energy fraction left after subtracting modal and remainder energies. It equals the normalized cross products when the modes and remainder reconstruct that input. Otherwise it also reflects the reconstruction discrepancy, as with EEMD's retained `meanNoise`. A series without modes has `modeCount` 0, and its remainder is the series.

Energy here is a quadratic signal measure, with squared signal units times seconds. Its identification with physical energy requires a relation for the measured quantity. A large modal energy fraction does not establish that the component is free of contamination.

Value

A named list of two **data.table** tables, each with the keys of `decomposition` followed by

`metrics`	`mode`, `frequency` in Hz, `bandwidth` in Hz, `energy` in the squared units of the series times seconds, `energyFraction` and `meanPeriod` in seconds; one row per mode, in the order of `modes`.
`metricsSeries`	`modeCount`, `residueFraction` and `crossFraction`; one row per series, in the order of `signal`.

Conditions

A series of `signal` with zero energy leaves its energy fractions undefined and stops the function with an error of class `xemdZeroEnergy`.

See Also

`decomposeTSL`, `selectModesTSL`, `reconstructTSL`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
# The same series under three IDs and three components.
TSL <- data.table::rbindlist(lapply(c("AT", "VT", "DT"), function(id) {
    data.table::rbindlist(lapply(c("H1", "H2", "UP"), function(ocid) {
        data.table::data.table(ID = id, OCID = ocid, t = Time, s = Signal)
    }))
}))
Sifting <- list(stopRule = "doubleThreshold")
```

```
Decomposition <- decomposeTSL(TSL, method = "decomposeEMD",
                              control = list(sifting = Sifting),
                              by = character())
OUT <- measureModesTSL(Decomposition)
OUT$metrics[OUT$metrics$OCID == "H1" & OUT$metrics$ID == "AT", ]
OUT$metricsSeries
```

`measureOrthogonality` *Inner Products and Cosines Between Components*

Description

Computes the inner products between the columns of a matrix of components and the cosines of the angles between them.

Usage

```
measureOrthogonality(components)
```

Arguments

`components`	Finite numeric matrix, samples by components. Attributes other than `dim` do not reach the outputs.

Details

The inner products are the Euclidean products $\sum_n v_i[n]v_j[n]$, without the sampling interval of the discrete inner product, which cancels in the cosine

$$\cos\theta_{ij} = \frac{\langle v_i, v_j\rangle}{\|v_i\|_2\|v_j\|_2}.$$

The cosine keeps its sign. The cross products are the terms by which the energy of a sum of components differs from the sum of their energies.

Value

A list of two square matrices, components by components:

`innerProduct`	The inner products.
`cosine`	The cosines.

Conditions

A column of zero norm stops the function with an error of class `xemdZeroEnergy`.

See Also

`measurePMSI`, `measureFCF`

Examples

```
Time <- seq_len(1000) / 1000
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
# The sum of the tones and the tones.
OUT <- measureOrthogonality(cbind(rowSums(Tones), Tones))
round(OUT$cosine, 6)
```

`measurePMSI` *Pseudo-Mode Splitting Index*

Description

Computes the pseudo-mode splitting index of each pair of adjacent modes.

Usage

```
measurePMSI(modes, denominator)
```

Arguments

`modes`	Finite numeric matrix, samples by modes, with at least one column. Attributes other than `dim` do not reach the outputs.
`denominator`	`"homogeneous"`, the sum of the squared norms of the pair, or `"regularized"`, which adds the 10^{-8} of the UPEMD definition, an absolute scale absent from the homogeneous form. No default; emd 0.8.1 has only the regularized form.

Details

For each pair of adjacent columns,

$$\mathrm{PMSI}(c_i, c_{i+1}) = \max\left\{\frac{\langle c_i, c_{i+1}\rangle}{\|c_i\|_2^2 + \|c_{i+1}\|_2^2}, 0\right\},$$

with 10^{-8} added to the denominator under `"regularized"`.

Value

Numeric vector with one value per pair of adjacent modes, of length one less than the number of columns.

Conditions

A pair of adjacent modes with zero energy under `"homogeneous"` stops the function with an error of class `xemdZeroEnergy`.

See Also

`measureOrthogonality`, `decomposeMaskedEMD`

Examples

```
Time <- seq_len(1000) / 1000
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
Mixed <- cbind(rowSums(Tones), Tones)
measurePMSI(Mixed, denominator = "homogeneous")
measurePMSI(Mixed, denominator = "regularized")
```

---

measureQRF *Quality of Reconstruction Factor*

Description

Computes the quality of reconstruction factor of an estimate against a reference component, in decibels.

Usage

```
measureQRF(estimate, reference)
```

Arguments

`estimate`	Finite numeric vector without `dim`.
`reference`	Finite numeric vector without `dim`, of the length of `estimate`, with a nonzero norm.

Details

$$\mathrm{QRF}(\hat{s}, s) = 20 \log_{10} \frac{\|s\|_2}{\|s - \hat{s}\|_2},$$

`Inf` when the estimate equals the reference.

Value

A number in dB.

Conditions

A reference of zero norm is outside the domain of the factor and stops the function with an error.

See Also

`measureFCF`, `measureOrthogonality`

Examples

```
Time <- seq_len(1000) / 1000
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
# The signal without its third tone, against the signal.
measureQRF(estimate = Signal - 0.0625 * cos(2 * pi * 288 * Time),
           reference = Signal)
measureQRF(estimate = Signal, reference = Signal)
```

`reconstructTSL` *Reconstruction of Time-Series Records from Their Modes*

Description

Adds up, for each series of the tables of `decomposeTSL`, the modes that a selection retains and, unless excluded, the remainder of the series.

Usage

```
reconstructTSL(decomposition, selection, includeResidue = TRUE)
```

Arguments

`decomposition`	The list of tables of `decomposeTSL`, as for `measureModesTSL`. It is not modified.
`selection`	A table with the keys and `mode` of every mode of `decomposition`, once each, and `retained`, `TRUE` or `FALSE`, as the table `selection` of `selectModesTSL`. It is not modified.
`includeResidue`	`TRUE`, the default, adds the remainder of each series, `residue` or `fitResidual`; `FALSE` leaves it out.

Details

The reconstruction of a series is the sum of its retained modes and, with `includeResidue`, its remainder. A series without retained modes sums zero, and a method without remainder, as `decomposeEWT`, adds none. With every mode and the remainder, the result follows the reconstruction relation of the selected method. For EMD the sum recovers the input up to rounding. In `decomposeEEMD` with the divisor `"realizations"`, it gives the signal plus `meanNoise`. The order of the rows of the tables does not intervene.

A partial sum implements the declared selection; neither the sum nor the choice to include a remainder establishes that the result is free of noise.

Value

A named list with one **data.table** table, `reconstruction`: the keys of `decomposition`, `t` and `s`, on the grid and in the order of `signal`.

See Also

selectModesTSL, measureModesTSL, decomposeTSL

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
# The same series under three IDs and three components.
TSL <- data.table::rbindlist(lapply(c("AT", "VT", "DT"), function(id) {
    data.table::rbindlist(lapply(c("H1", "H2", "UP"), function(ocid) {
        data.table::data.table(ID = id, OCID = ocid, t = Time, s = Signal)
    }))
}))
Sifting <- list(stopRule = "doubleThreshold")
Decomposition <- decomposeTSL(TSL, method = "decomposeEMD",
                              control = list(sifting = Sifting),
                              by = character())
Metrics <- measureModesTSL(Decomposition)$metrics
# A minimum energy fraction of zero retains every mode: with the residue,
# the reconstruction is the signal.
Every <- selectModesTSL(Decomposition, metrics = Metrics,
                        criterion = list(rule = "energyFraction", minimum = 0,
                                         action = "keep"))
Complete <- reconstructTSL(Decomposition, selection = Every$selection)
max(abs(Complete$reconstruction$s - Decomposition$signal$s))
# The modes in [2, 15] Hz, without the residue.
Band <- selectModesTSL(Decomposition, metrics = Metrics,
                       criterion = list(rule = "band", fMin = 2, fMax = 15,
                                        action = "keep"))
Filtered <- reconstructTSL(Decomposition, selection = Band$selection,
                           includeResidue = FALSE)
head(Filtered$reconstruction)
```

---

resiftModes *Consecutive Post-Processing of Ensemble Modes*

Description

Re-sifts a sequence of modes, typically the averaged modes of an ensemble, by extracting the first sifted component of consecutive sums and carrying each remainder to the next mode.

Usage

```
resiftModes(modes, sifting)
```

Arguments

`modes`	Finite numeric matrix, samples by modes, with at least two columns $D_1, \ldots, D_K$.
`sifting`	Named list of sifting controls, as in `decomposeEMD`; the rule `"consecutive"` is rejected. The chain has $K - 1$ extractions, so `sifting$maxModes` does not apply.

Details

With $u_1 = D_1 + D_2$, stage i extracts $a_i = E_1(u_i)$, keeps the remainder $q_i = u_i - a_i$ and adds the next mode, $u_{i+1} = q_i + D_{i+2}$. The completed chain returns $K - 1$ modes and the residue q_{K-1} with `"chainComplete"`, and

$$\sum_{i=1}^{K-1} a_i + q_{K-1} = \sum_{j=1}^{K} D_j.$$

When $E_1(u_i)$ does not exist, the chain ends with `"insufficientExtrema"`, the modes $a_1, \ldots, a_{i-1}$ and the residue $u_i + \sum_{j>i+1} D_j$, which keeps the same sum.

Value

A list with

`modes`	Double matrix, samples by the extracted modes.
`residue`	Numeric vector, the remainder of the chain.
`iterations`	Integer vector, the subtractions of each extraction.
`converged`	Logical vector by extraction, as in `decomposeEMD`.
`stopReason`	`"chainComplete"` or `"insufficientExtrema"`.

References

The selection of the reflected extrema and the spline interpolation of the envelope nodes are adapted from emd.m (G. Rilling, March 2007), distributed with CEEMDAN revision af9a0f83, copyright (c) 2020 Marcelo A. Colominas, MIT License; see the NOTICE file of the installed package.

See Also

`decomposeEEMD`, `decomposeEMD`

Examples

```
Time <- seq(0, 9, length.out = 1000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
set.seed(628)
Noise <- t(matrix(stats::rnorm(10L * length(Signal)), nrow = 10L,
                  byrow = TRUE))
Sifting <- list(stopRule = "doubleThreshold", thresholds = c(0.05, 0.5),
                fraction = 0.05, maxSift = 2000, boundary = "reflection",
                maxModes = Inf)
# Two of the ten realizations keep the example short.
```

```
Modes <- decomposeEEMD(Signal, noise = Noise[, 1:2], amplitudeRatio = 0.2,
                       divisor = "realizations", sifting = Sifting)$modes
OUT <- resiftModes(Modes, sifting = Sifting)
c(input = ncol(Modes), output = ncol(OUT$modes))
OUT$stopReason
max(abs(rowSums(OUT$modes) + OUT$residue - rowSums(Modes)))
```

`selectModesEnergy` *Selection of Modes by Energy*

Description

Orders the modes by decreasing energy and selects those before the largest drop between consecutive energies.

Usage

```
selectModesEnergy(modes)
```

Arguments

`modes`	Finite numeric matrix, samples by modes, with at least two columns. Attributes other than `dim` do not reach the outputs.

Details

The energy of mode k is $P_k = \sum_n c_k[n]^2$. With the permutation π that orders the energies decreasingly, the selection keeps $\pi(1), \ldots, \pi(v)$ with

$$v = \arg \max_{1 \le m < K} \left(P_{\pi(m)} - P_{\pi(m+1)} \right) .$$

The source fixes no tie-break, so a largest drop that is zero or not unique stops the function with an error.

Value

Integer vector of the selected column indices, the most energetic first.

See Also

`measureFCF`, `measureDTW`

Examples

```
Time <- seq_len(1000) / 1000
Tones <- cbind(cos(2 * pi * 2 * Time), 0.25 * cos(2 * pi * 24 * Time),
               0.0625 * cos(2 * pi * 288 * Time))
selectModesEnergy(Tones)
# The same tones in another order.
selectModesEnergy(Tones[, c(3, 1, 2)])
```

`selectModesTSL` *Selection of the Modes of Time-Series Records*

Description

Selects the modes of the tables of `decomposeTSL` with a declared criterion: explicit mode indices, a minimum energy fraction or a frequency band, to keep the modes that meet it or to remove them.

Usage

```
selectModesTSL(decomposition, metrics, criterion)
```

Arguments

`decomposition`	The list of tables of `decomposeTSL`, as for `measureModesTSL`; its metadata cannot take the names `mode`, `criterion` or `retained`. It is not modified.
`metrics`	The table `metrics` of `measureModesTSL` for the same decomposition: one row for each of its modes, each with an `energyFraction`. It is not modified.
`criterion`	A list with `rule`, exactly the fields of that rule, and `action`, `"keep"` or `"remove"`. The rules are `"modes"` with `modes`, a table with `mode`, whole numbers from 1, and key columns of `decomposition`. Each row names the mode of that index of every series whose keys it matches, so the indices go by group (the by columns and `ID`) or by series (with `OCID`); each row names a mode of the decomposition. `"energyFraction"` with `minimum`, a finite number not below zero: a mode meets it when its `energyFraction` is not below `minimum`. `"band"` with `fMin` and `fMax` in Hz, $0 \leq f_{\min} \leq f_{\max}$: a mode meets the closed band when $f_{\min} \leq$ `frequency` $\leq f_{\max}$. A mode without frequency, of zero energy, is in no band. `"keep"` retains the modes that meet the criterion and `"remove"` the others.

Details

The criterion is supplied by the caller. Mode indices, energy fractions and centroid frequencies do not by themselves identify uncontaminated physical components. Equal mode indices in independently decomposed series need not represent the same physical contribution.

The band compares `frequency` alone, whereas the rules `keep.Fo` and `remove.Fo` of gmsp 0.8.0 require the interval $[F_o - BW, F_o + BW]$ inside the band.

Value

A named list of two **data.table** tables:

`modes`	The rows of `modes` of `decomposition` of the retained modes, in its order.

`selection` Every mode of `decomposition` with its keys, `mode`, `criterion` and `retained`, in the order of the decomposition. `criterion` is the criterion as declared text: `"<action> listed modes"`, `"<action> energyFraction >= <minimum>"` or `"<action> frequency in [<fMin>, <fMax>] Hz"`, such as `"keep frequency in [2, 15] Hz"`.

See Also

`measureModesTSL`, `reconstructTSL`, `decomposeTSL`

Examples

```
Time <- seq(0, 9, length.out = 3000)
Signal <- sin(pi * Time) + sin(2 * pi * Time) + sin(6 * pi * Time) +
    0.5 * Time
# The same series under three IDs and three components.
TSL <- data.table::rbindlist(lapply(c("AT", "VT", "DT"), function(id) {
    data.table::rbindlist(lapply(c("H1", "H2", "UP"), function(ocid) {
        data.table::data.table(ID = id, OCID = ocid, t = Time, s = Signal)
    }))
}))
Sifting <- list(stopRule = "doubleThreshold")
Decomposition <- decomposeTSL(TSL, method = "decomposeEMD",
                              control = list(sifting = Sifting),
                              by = character())
Metrics <- measureModesTSL(Decomposition)$metrics
# Keep the modes whose frequency lies in [2, 15] Hz.
Band <- selectModesTSL(Decomposition, metrics = Metrics,
                       criterion = list(rule = "band", fMin = 2, fMax = 15,
                                        action = "keep"))
Band$selection[Band$selection$OCID == "H1" & Band$selection$ID == "AT", ]
# Remove the first mode of every series of the ID VT.
Listed <- data.frame(ID = "VT", mode = 1)
First <- selectModesTSL(Decomposition, metrics = Metrics,
                        criterion = list(rule = "modes", modes = Listed,
                                         action = "remove"))
table(First$selection$ID, First$selection$retained)
```

---

transformSST *Synchrosqueezed Wavelet Transform*

Description

Computes the Morlet continuous wavelet transform of a sampled signal and reassigns its coefficients to frequency cells by the instantaneous frequency estimated from the transform.

Usage

```
transformSST(signal, samplingInterval, voiceCount = 16,
             morletCenter = 2 * pi, threshold)
```

Arguments

`signal`	Numeric vector of at least four finite values without `dim`, in the units of the quantity.
`samplingInterval`	Sampling interval Δt in seconds, a finite number greater than zero.
`voiceCount`	Whole number ≥ 1 of voices per octave; default `16`, the value of SynchWave.
`morletCenter`	Center $\mu > 0$ of the Morlet wavelet in normalized frequency; default 2π, the value of SynchWave.
`threshold`	Number greater than zero in the units of the signal: only coefficients with $\lvert W\rvert >$ `threshold` are reassigned. No default; SynchWave uses `sqrt(.Machine$double.eps)`.

Details

The record is extended by symmetric padding to $T = 2^{1+\mathrm{round}(\log_2 N)}$ samples, and the transform uses the scales $a_j = 2^{j/\mathrm{voiceCount}}$, $j = 1, \ldots, (\log_2 T - 1)\,\mathrm{voiceCount}$. For each coefficient above the threshold, the frequency

$$\frac{\mathrm{Re}\,\Omega_f(a,b)}{2\pi} = \frac{1}{2\pi}\mathrm{Im}\frac{\partial_b W_f(a,b)}{W_f(a,b)}$$

in Hz goes to the nearest of J logarithmic cells from $1/((N-1)\Delta t)$ to $1/(2\Delta t)$, J the number of scales, where the coefficient adds $Wa^{-1/2}$; the sums are divided by `voiceCount`. Estimates that are not positive or fall outside the cells are not accumulated, and a transform whose energy is below the machine epsilon gives a zero result. The representation recovers the signal only approximately. The threshold is in the units of the signal, so `decomposeTSL` has no operation for this transform.

Value

A list with

`transform`	Complex matrix, samples by frequency cells.
`frequency`	Numeric vector of the cell frequencies in Hz.

References

Daubechies, I., Lu, J. and Wu, H.-T. (2009). Synchrosqueezed wavelet transforms: a tool for empirical mode decomposition. arXiv:0912.2437.

The wavelet transform, the Morlet wavelet and the reassignment are adapted from SynchWave 1.1.2, revision 9ac80b2c3e60b417d0ee5567a66e8e64faa1a979 (MATLAB original by Eugene Brevdo; R port by Dongik Jang, Hee-Seok Oh and Donghoh Kim; the Synchrosqueezing toolbox attribution also names Hau-Tieng Wu), LGPL-2.0-or-later, which `xemd` distributes under GPL-3 as section 3 of the GNU Library General Public License version 2 permits; see the NOTICE file of the installed package.

See Also

`decomposeEWT`, `computeInstantaneousAttributes`

Examples

```
SamplingInterval <- 1 / 1000
Time <- seq_len(1000) * SamplingInterval
Signal <- cos(2 * pi * 2 * Time) + 0.25 * cos(2 * pi * 24 * Time) +
    0.0625 * cos(2 * pi * 288 * Time)
OUT <- transformSST(Signal, samplingInterval = SamplingInterval,
                    voiceCount = 16, morletCenter = 2 * pi,
                    threshold = sqrt(.Machine$double.eps))
dim(OUT$transform)
range(OUT$frequency)
```

Index

assessFractionalNoise, 3
assessIMF, 5, *7*, *20*
assessWhiteNoise, *4*, 6, *6*

buildHilbertSpectrum, 8, *10*

computeInstantaneousAttributes, *8*, 9, *28*, *44*, *48*, *66*

decomposeBEMD, 10, *19*, *35*, *41*, *51*
decomposeCEEMD, 12, *15*, *17*, *19*, *24*
decomposeCEEMDAN, *13*, 14, *17*, *19*, *24*
decomposeEEMD, *12–15*, 16, *19, 20*, *23, 24*, *53*, *60*, *62*
decomposeEMD, *4–8*, *10–12*, *14*, *16*, 18, *23*, *26, 27*, *30, 31*, *33–35*, *42*, *44*, *48*, *62*
decomposeEWT, 20, *50*, *56*, *60*, *66*
decomposeICEEMDAN, *13*, *15*, *17*, *19*, 23
decomposeIF, 25, *30*
decomposeIteratedMaskedEMD, *19*, 27, *33*, *42*
decomposeLMD, *26*, 29
decomposeMaskedEMD, *19*, *27, 28*, 31, *41*, *48*, *58*
decomposeMEMD, *11*, *19*, 33, *37, 38*, *41*, *51*
decomposeMVMD, 35, *35*, *40*, *42*, *46*, *51*, *56*
decomposeNAMEMD, *19*, *35*, 37, *41*, *51*
decomposeSVMD, *36*, 38, *41*, *46*, *56*
decomposeTSL, *20*, *36–38*, 40, *46*, *51*, *55, 56*, *60, 61*, *64–66*
decomposeTVFEMD, 43
decomposeVMD, *22*, *35, 36*, *39, 40*, 45, *56*
describeModule, 47, *52*

estimateMaskFrequency, *10*, *19*, *31*, *33*, 47, *51*
extractSSTModes, 49

listDecompositions, *41, 42*, *47*, 51, *52*
listOperations, *47*, 51, *51*

measureDispersion, *17*, 52
measureDTW, 53, *54*, *63*
measureFCF, 54, *54*, *57*, *59*, *63*
measureModesTSL, 55, *60, 61*, *64, 65*
measureOrthogonality, 57, *58, 59*
measurePMSI, *57*, 58
measureQRF, 59

reconstructTSL, *56*, 60, *65*
resiftModes, *17*, *19*, *51*, 61

selectModesEnergy, *54*, 63
selectModesTSL, *56*, *60, 61*, 64

transformSST, *22*, *49–51*, 65