Package ‘rLifting’

February 20, 2026

Title High-Performance Wavelet Lifting Transforms

Version 0.9.0

Date 2026-02-17

Description Performs Wavelet Lifting Transforms focusing on signal denoising and functional data analysis (FDA). Implements a hybrid architecture with a zero-allocation 'C++' core for high-performance processing. Features include unified offline (batch) denoising, causal (real-time) filtering using a ring buffer engine, and adaptive recursive thresholding.

License MIT + file LICENSE

URL https://github.com/mkyou/rLifting

BugReports https://github.com/mkyou/rLifting/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

VignetteBuilder knitr

LinkingTo Rcpp

Imports Rcpp

Suggests testthat (>= 3.0.0), knitr, rmarkdown, dplyr, ggplot2, microbenchmark

Config/testthat/edition 3

Depends R (>= 4.1.0)

NeedsCompilation yes

Author Moises da Silva [aut, cre]

Maintainer Moises da Silva <moisesdff8@gmail.com>

Repository CRAN

Date/Publication 2026-02-20 10:00:10 UTC

Contents

benchmark_causal . . . 2
benchmark_offline . . . 3
compute_adaptive_threshold . . . 3
custom_wavelet . . . 4
denoise_signal_causal . . . 5
denoise_signal_offline . . . 6
diagnose_wavelet . . . 7
doppler_example . . . 7
ilwt . . . 8
leakage_results . . . 8
lifting_scheme . . . 9
lift_step . . . 9
lwt . . . 10
new_wavelet_stream . . . 11
plot.adaptive_thresholds . . . 12
plot.lifting_scheme . . . 12
plot.lwt . . . 13
print.adaptive_thresholds . . . 13
print.lifting_scheme . . . 14
print.lwt . . . 14
print.wavelet_diagnosis . . . 15
print.wavelet_stream . . . 15
rLifting . . . 16
threshold . . . 16
threshold_hard . . . 17
threshold_semisoft . . . 17
threshold_soft . . . 18
validate_compact_support . . . 18
validate_orthogonality . . . 19
validate_perfect_reconstruction . . . 19
validate_shift_sensitivity . . . 20
validate_vanishing_moments . . . 20
visualize_wavelet_basis . . . 21

Index **22**

`benchmark_causal` *Causal Benchmark Results*

Description

Comparison of execution time between `rLifting`'s optimized causal mode and a naive sliding-window implementation using `wavethresh`.

Usage

```
data(benchmark_causal)
```

Format

A list containing:

rLifting_Time_Avg Average time (seconds) for rLifting.

Wavethresh_Naive_Time Time (seconds) for naive sliding window.

Speedup_Factor Ratio of Naive Time to rLifting Time.

`benchmark_offline` *Offline Benchmark Results*

Description

Comparison of execution time and reconstruction error (MSE) between `rLifting` and other packages (wavethresh, wavelets) using Haar wavelet.

Usage

```
data(benchmark_offline)
```

Format

A data frame with the following columns:

Pkg Package name.

Time Execution time in seconds.

MSE Mean Squared Error.

`compute_adaptive_threshold` *Calculate Adaptive Threshold (Recursive)*

Description

Estimates the optimal noise threshold based on current window statistics. Implements the recursive formula from Liu et al. (2014). Accelerated with 'C++'.

Usage

```
compute_adaptive_threshold(lwt_obj, alpha = 0.3, beta = 1.2)
```

Arguments

`lwt_obj`	Object returned by `lwt()`.
`alpha`	Recursive adjustment parameter (Eq. 9).
`beta`	Initial threshold scale factor (Eq. 9).

Value

Object of class `adaptive_thresholds` (a list of thresholds).

`custom_wavelet` *Create a custom wavelet*

Description

Wrapper to create a `lifting_scheme` object from manual steps.

Usage

```
custom_wavelet(name, steps, norm = c(1, 1))
```

Arguments

`name`	Identifier name for the wavelet.
`steps`	List of steps created via `lift_step`.
`norm`	Normalization vector c(K, 1/K).

Value

An object of class `lifting_scheme`.

Examples

```
p1 = lift_step("predict", c(1), position = "center")
u1 = lift_step("update", c(0.5), position = "center")
w = custom_wavelet("HaarManual", list(p1, u1), c(1.41, 0.707))
```

`denoise_signal_causal` *Causal Batch Denoising (Turbo Simulation)*

Description

Processes a complete signal simulating the sequential arrival of data. Uses the specialized 'C++' class `WaveletEngine` to perform causal filtering efficiently on a historical dataset.

Usage

```
denoise_signal_causal(
  signal,
  scheme,
  levels = 1,
  window_size = 256,
  alpha = 0.3,
  beta = 1.2,
  method = "semisoft",
  extension = "symmetric",
  update_freq = 1
)
```

Arguments

`signal`	Complete vector of the noisy signal.
`scheme`	`lifting_scheme` object.
`levels`	Decomposition levels.
`window_size`	Window size.
`alpha`	Threshold decay parameter (Eq 9).
`beta`	Threshold gain factor (Eq 9).
`method`	Thresholding method ("soft", "hard", "semisoft").
`extension`	Boundary treatment ('symmetric', 'periodic').
`update_freq`	Frequency of threshold updates.

Value

Filtered vector (same length as input).

denoise_signal_offline — *Offline Denoising (Global Batch)*

Description

Performs denoising on the entire signal at once using a non-causal approach. Uses global statistics for recursive threshold calculation (Eq. 9). This function is fully optimized in 'C++' (Zero-Allocation).

Usage

```
denoise_signal_offline(
  signal,
  scheme,
  alpha = 0.3,
  beta = 1.2,
  levels = 3,
  method = "semisoft",
  extension = "symmetric"
)
```

Arguments

`signal`	Numeric vector containing the complete signal.
`scheme`	A `lifting_scheme` object.
`alpha`	Recursive threshold parameter.
`beta`	Threshold scale factor.
`levels`	Number of decomposition levels.
`method`	Thresholding method ("hard", "soft", "semisoft").
`extension`	Extension mode ("symmetric", "periodic", "zero").

Value

Filtered numeric vector (same length as input).

`diagnose_wavelet` *Complete Wavelet Diagnosis*

Description

Runs a battery of physical and mathematical tests on a wavelet.

Usage

```
diagnose_wavelet(wavelet_name, config, verbose = TRUE, plot = TRUE)
```

Arguments

`wavelet_name`	Name string or a `lifting_scheme` object.
`config`	Configuration list (is_ortho, vm_degrees, max_taps).
`verbose`	Print results to console handling? (Defaults to TRUE).
`plot`	Boolean. Visualize basis functions during diagnosis? (Defaults to TRUE).

Value

An object of class `wavelet_diagnosis` (S3), which is a list containing the results of each test. The object has a dedicated `print` method.

`doppler_example` *Noisy Doppler Signal Example*

Description

A synthetic dataset containing a Doppler signal contaminated with Gaussian noise. Used in the "General Usage" vignette.

Usage

```
data(doppler_example)
```

Format

A data frame with 2048 rows and 3 columns:

index Time index.

original The pure Doppler signal.

noisy The signal with added Gaussian noise (sd=0.5).

ilwt *Inverse Lifting Wavelet Transform ('C++' Accelerated)*

Description

Reconstructs the original signal from wavelet coefficients. Optimized with 'C++' backend.

Usage

```
ilwt(lwt_obj, scheme = NULL)
```

Arguments

`lwt_obj`	Object of class 'lwt' returned by `lwt()`.
`scheme`	(Optional) `lifting_scheme` object. If NULL, uses the one from `lwt_obj`.

Value

Numeric vector containing the reconstructed signal.

Examples

```
s = c(1, 2, 3, 4)
sch = lifting_scheme("haar")
fwd = lwt(s, sch)
rec = ilwt(fwd)
print(rec) # Should match s
```

leakage_results *Leakage (Impulse Response) Results*

Description

Measurement of energy leakage into the "past" when processing an impulse signal. Used to demonstrate the zero-lookahead property of the causal mode.

Usage

```
data(leakage_results)
```

Format

A data frame with:

Method Method description (e.g. "rLifting causal").

Leakage Sum of squared differences (leakage energy).

`lifting_scheme` *Lifting Scheme Constructor*

Description

Creates an S3 object containing the prediction (P) and update (U) steps required for the Lifting Transform.

Usage

```
lifting_scheme(wavelet = "haar", custom_steps = NULL, custom_norm = NULL)
```

Arguments

`wavelet`	Wavelet name (string). Options: "haar", "db2", "cdf53", "cdf97", "dd4", "lazy".
`custom_steps`	List of custom steps (optional). If provided, ignores internal lookup.
`custom_norm`	Normalization vector (optional).

Value

An object of class `lifting_scheme`.

`lift_step` *Create an individual Lifting Step*

Description

Helper function to create prediction (P) or update (U) steps, abstracting the complexity of index management.

Usage

```
lift_step(
  type = c("predict", "update"),
  coeffs,
  start_idx = NULL,
  position = "center"
)
```

Arguments

`type`	Step type: "predict" (P) or "update" (U).
`coeffs`	Numeric vector containing the filter coefficients.
`start_idx`	(Optional) Manual start index. If provided, ignores the `position` parameter. Use this for fine-grained control.
`position`	Automatic index adjustment (used only if `start_idx` is NULL): • "center": Centers the filter (default). • "left": Causal filter (looks into the past). • "right": Anti-causal filter (looks into the future).

Value

A list formatted for the internal lifting engine.

`lwt` *Lifting Wavelet Transform (Forward)*

Description

Performs the Forward Wavelet Transform using the Lifting Scheme. Optimized with 'C++' back-end.

Usage

```
lwt(signal, scheme, levels = 1, extension = "symmetric")
```

Arguments

`signal`	Numeric vector containing the input signal.
`scheme`	A `lifting_scheme` object.
`levels`	Integer. Number of decomposition levels.
`extension`	Boundary extension mode: "symmetric" (default), "periodic", or "zero".

Value

An object of class 'lwt'. It is a list containing 'coeffs' (list of details d1..dn and approximation an) and 'scheme' (the scheme object used).

Examples

```
data = c(1, 2, 3, 4, 5, 6, 7, 8)
sch = lifting_scheme("haar")
res = lwt(data, sch, levels = 2)
print(res)
```

`new_wavelet_stream` *Create an Adaptive Wavelet Stream Processor ('C++' Core)*

Description

Generates a stateful function backed by a high-performance 'C++' Ring Buffer engine. It implements Sliding Window + Lifting Decomposition + Adaptive Thresholding in constant amortized time (O(1)) per sample.

Usage

```
new_wavelet_stream(
  scheme,
  window_size = 256,
  levels = 1,
  alpha = 0.3,
  beta = 1.2,
  method = "semisoft",
  extension = "symmetric",
  update_freq = 1
)
```

Arguments

`scheme`	A `lifting_scheme` object.
`window_size`	Sliding window size (W). Must be > 8.
`levels`	Decomposition levels (default 1).
`alpha`	Threshold decay parameter (Eq 9).
`beta`	Threshold gain factor (Eq 9).
`method`	Shrinkage method: "hard", "soft", "semisoft".
`extension`	Boundary handling ('symmetric', 'periodic', 'zero').
`update_freq`	How often to recompute threshold statistics (default 1).

Value

A closure function `processor(new_sample)` that accepts a single numeric value and returns the filtered value immediately.

`plot.adaptive_thresholds` *Plot method for Adaptive Thresholds*

Description

Plot method for Adaptive Thresholds

Usage

```
## S3 method for class 'adaptive_thresholds'
plot(x, ...)
```

Arguments

`x`	Object of class `adaptive_thresholds`.
`...`	Additional arguments.

Value

Invisibly returns `NULL`.

`plot.lifting_scheme` *Plot method for Lifting Scheme*

Description

Plot method for Lifting Scheme

Usage

```
## S3 method for class 'lifting_scheme'
plot(x, ...)
```

Arguments

`x`	An object of class `lifting_scheme`.
`...`	Additional arguments passed to `visualize_wavelet_basis`.

Value

Invisibly returns `NULL`.

plot.lwt *Plot method for LWT Decomposition*

Description

Plot method for LWT Decomposition

Usage

```
## S3 method for class 'lwt'
plot(x, ...)
```

Arguments

x	An object of class `lwt`.
...	Additional arguments.

Value

Invisibly returns `NULL`.

print.adaptive_thresholds *Print method for Adaptive Thresholds*

Description

Print method for Adaptive Thresholds

Usage

```
## S3 method for class 'adaptive_thresholds'
print(x, ...)
```

Arguments

x	Object of class `adaptive_thresholds`.
...	Additional arguments.

Value

Invisibly returns `x`.

`print.lifting_scheme` *Print method*

Description

Print method

Usage

```
## S3 method for class 'lifting_scheme'
print(x, ...)
```

Arguments

`x`	object of class `lifting_scheme`.
`...`	additional arguments.

Value

Invisibly returns `NULL`. Called for side effects (printing).

`print.lwt` *Print method for LWT*

Description

Print method for LWT

Usage

```
## S3 method for class 'lwt'
print(x, ...)
```

Arguments

`x`	An object of class lwt.
`...`	Additional arguments.

Value

Invisibly returns `NULL`. Called for side effects (printing).

`print.wavelet_diagnosis` *Print method for Wavelet Diagnosis*

Description

Print method for Wavelet Diagnosis

Usage

```
## S3 method for class 'wavelet_diagnosis'
print(x, ...)
```

Arguments

`x`	Object of class `wavelet_diagnosis`.
`...`	Additional arguments.

Value

Invisibly returns `x`.

`print.wavelet_stream` *Print method for Wavelet Stream Processor*

Description

Print method for Wavelet Stream Processor

Usage

```
## S3 method for class 'wavelet_stream'
print(x, ...)
```

Arguments

`x`	Object of class `wavelet_stream`.
`...`	Additional arguments.

Value

Invisibly returns `x`.

rLifting — *rLifting: High-Performance Wavelet Lifting Transforms*

Description

A unified framework for Wavelet Transforms using the Lifting Scheme. It provides robust tools for offline signal analysis and functional data analysis (FDA), while also enabling high-performance causal processing for real-time applications via a specialized 'C++' core.

Author(s)

Maintainer: Moises da Silva <`moisesdff8@gmail.com`>

See Also

Useful links:

- `https://github.com/mkyou/rLifting`
- Report bugs at `https://github.com/mkyou/rLifting/issues`

threshold — *General Thresholding Wrapper*

Description

General Thresholding Wrapper

Usage

```
threshold(x, lambda, method = "soft")
```

Arguments

`x`	Input vector.
`lambda`	Threshold value.
`method`	Method: "hard", "soft" or "semisoft".

Value

Numeric vector of the same length as `x` with thresholded coefficients.

`threshold_hard` *Hard Thresholding*

Description

Sets coefficients below the threshold to zero, keeping others unchanged. Known as the "keep or kill" policy.

Usage

```
threshold_hard(x, lambda)
```

Arguments

`x`	Vector of coefficients (details).
`lambda`	Positive threshold value.

Value

Processed vector.

`threshold_semisoft` *Semisoft Shrinkage (Hyperbolic)*

Description

Implementation based on Liu et al. (2014). Combines the stability of Soft Thresholding with the amplitude precision of Hard Thresholding. Function: `sign(x) * sqrt(x^2 - lambda^2)` for values above lambda.

Usage

```
threshold_semisoft(x, lambda)
```

Arguments

`x`	Vector of coefficients.
`lambda`	Positive threshold value.

Value

Processed vector.

`threshold_soft` *Soft Thresholding*

Description

Sets coefficients below the threshold to zero and shrinks others towards zero. Reduces noise variance but introduces amplitude bias.

Usage

```
threshold_soft(x, lambda)
```

Arguments

`x`	Vector of coefficients.
`lambda`	Positive threshold value.

Value

Processed vector.

`validate_compact_support` *Validate Compact Support (FIR Compliance)*

Description

Verifies if the impulse response is finite (FIR Filter).

Usage

```
validate_compact_support(scheme, max_width)
```

Arguments

`scheme`	Object of class `lifting_scheme`.
`max_width`	Maximum expected width (number of taps).

Value

List with status and number of active taps.

`validate_orthogonality` *Validate Orthogonality (Energy Conservation)*

Description

Verifies Parseval's Theorem. Only applicable for orthogonal wavelets.

Usage

```
validate_orthogonality(scheme, expected = TRUE, tol = 1e-09)
```

Arguments

`scheme`	Object of class `lifting_scheme`.
`expected`	Boolean. If TRUE, expects orthogonality.
`tol`	Tolerance (default 1e-9).

Value

List with status and energy ratio (Out/In).

`validate_perfect_reconstruction` *Validate Perfect Reconstruction (Stress Test)*

Description

Verifies wavelet invertibility against a battery of signals.

Usage

```
validate_perfect_reconstruction(scheme, tol = 1e-09)
```

Arguments

`scheme`	Object of class `lifting_scheme`.
`tol`	Numerical error tolerance (default 1e-9).

Value

List with global status and maximum error found.

`validate_shift_sensitivity`

Validate Shift Sensitivity (Shift Variance)

Description

Decimated wavelets are not translation invariant. This test quantifies the variation in detail energy when shifting the input signal by 1 sample.

Usage

```
validate_shift_sensitivity(scheme)
```

Arguments

`scheme`	Object of class `lifting_scheme`.

Value

List with status and percentage variation.

`validate_vanishing_moments`

Validate Vanishing Moments

Description

Verifies if the wavelet cancels polynomials of a specific degree.

Usage

```
validate_vanishing_moments(scheme, degree = 0, tol = 1e-09)
```

Arguments

`scheme`	Object of class `lifting_scheme`.
`degree`	Polynomial degree (0=Constant, 1=Ramp, 2=Parabola...).
`tol`	Residual energy tolerance (default 1e-9).

Value

List with status and residual energy.

visualize_wavelet_basis

Visualize Basis Functions (Scaling and Wavelet)

Description

Plots the waveform by iterating the reconstruction over several levels.

Usage

```
visualize_wavelet_basis(scheme, plot = TRUE, levels = 8)
```

Arguments

`scheme`	Object of class `lifting_scheme`.
`plot`	Boolean.
`levels`	Number of cascade levels.

Value

Invisibly returns `NULL`. Called for side effects (plotting).

Index

∗ **datasets**
    benchmark_causal, 2
    benchmark_offline, 3
    doppler_example, 7
    leakage_results, 8

benchmark_causal, 2
benchmark_offline, 3

compute_adaptive_threshold, 3
custom_wavelet, 4

denoise_signal_causal, 5
denoise_signal_offline, 6
diagnose_wavelet, 7
doppler_example, 7

ilwt, 8

leakage_results, 8
lift_step, 9
lifting_scheme, 9
lwt, 10

new_wavelet_stream, 11

plot.adaptive_thresholds, 12
plot.lifting_scheme, 12
plot.lwt, 13
print.adaptive_thresholds, 13
print.lifting_scheme, 14
print.lwt, 14
print.wavelet_diagnosis, 15
print.wavelet_stream, 15

rLifting, 16
rLifting-package *(*rLifting*)*, 16

threshold, 16
threshold_hard, 17
threshold_semisoft, 17
threshold_soft, 18

validate_compact_support, 18
validate_orthogonality, 19
validate_perfect_reconstruction, 19
validate_shift_sensitivity, 20
validate_vanishing_moments, 20
visualize_wavelet_basis, 21