Package 'koopman.dmd'

September 14, 2026

Type Package

Title Koopman Operator and Dynamic Mode Decomposition for Dynamical Systems

Version 0.2.2

Description Dynamic Mode Decomposition (DMD) with Koopman operator theory extensions, powered by a Rust backend via 'extendr'. Provides standard DMD as described in Schmid (2010) <doi:10.1017/S0022112010001217>, DMD with control for forced linear systems following Proctor, Brunton, and Kutz (2016) <doi:10.1137/15M1013857>, Extended DMD with lifting functions, Hankel-DMD via time-delay embedding, Generalized Laplace Analysis for direct eigenfunction computation, and harmonic time averages and mesochronic harmonic plots for phase space analysis as developed in Mezic (2020) <doi:10.48550/arXiv.2009.05883>. Includes built-in area-preserving and chaotic maps for experimentation.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements Cargo (Rust's package manager), rustc (>= 1.85)

Depends R (>= 4.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/jimeharrisjr/rust-dmd, https://jimeharrisjr.github.io/rust-dmd/

BugReports https://github.com/jimeharrisjr/rust-dmd/issues

NeedsCompilation yes

Biarch false

Config/testthat/edition 3

Config/rextendr/version 0.3.1

RoxygenNote 7.3.2

Author James Harris [aut, cre, cph], The authors of the dependency Rust crates [ctb] (see inst/AUTHORS file for details)

Maintainer James Harris <jimeharrisjr@gmail.com>

Repository CRAN

Date/Publication 2026-09-14 15:20:02 UTC

Contents

`koopman.dmd-package` *koopman.dmd: Koopman Operator and Dynamic Mode Decomposition for Dynamical Systems*

Description

Dynamic Mode Decomposition (DMD) with Koopman operator theory extensions, powered by a Rust backend via 'extendr'. Provides standard DMD as described in Schmid (2010) doi:10.1017/S0022112010001217, DMD with control for forced linear systems following Proctor, Brunton, and Kutz (2016) doi:10.1137/15M1013857, Extended DMD with lifting functions, Hankel-DMD via time-delay embedding, Generalized Laplace Analysis for direct eigenfunction computation, and harmonic time averages and mesochronic harmonic plots for phase space analysis as developed in Mezic (2020) doi:10.48550/arXiv.2009.05883. Includes built-in area-preserving and chaotic maps for experimentation.

Author(s)

Maintainer: James Harris <jimeharrisjr@gmail.com> [copyright holder]

Other contributors:

- The authors of the dependency Rust crates (see inst/AUTHORS file for details) [contributor]

See Also

Useful links:

- https://github.com/jimeharrisjr/rust-dmd
- https://jimeharrisjr.github.io/rust-dmd/
- Report bugs at https://github.com/jimeharrisjr/rust-dmd/issues

---

`classify_phase_space` *Classify phase space points by HTA magnitude*

Description

Classify phase space points by HTA magnitude

Usage

```
classify_phase_space(
  hta_magnitudes,
  resonating_threshold = 0.01,
  chaotic_threshold = 1e-04
)
```

Arguments

`hta_magnitudes`	Numeric vector of |HTA| values.
`resonating_threshold`	Threshold for resonating. Default 0.01.
`chaotic_threshold`	Threshold for chaotic. Default 0.0001.

Value

Integer vector (1=resonating, 2=chaotic, 3=non-resonating).

Examples

```
# Classify orbits from their HTA magnitudes
mags <- c(0.5, 0.001, 1e-06, 0.1)
classify_phase_space(mags)

# Thresholds are adjustable
classify_phase_space(mags, resonating_threshold = 0.1, chaotic_threshold = 0.001)
```

---

dmd *Dynamic Mode Decomposition*

Description

Perform Dynamic Mode Decomposition on time-series data.

Usage

```
dmd(
  X,
  rank = NULL,
  center = FALSE,
  dt = 1,
  lifting = NULL,
  lifting_param = NULL
)
```

Arguments

X	Numeric matrix (n_vars x n_time).
rank	Integer truncation rank, or NULL for automatic.
center	Logical, center data by subtracting row means.
dt	Numeric time step.
lifting	Character lifting type or NULL.
lifting_param	Integer lifting parameter or NULL.

Value

An S3 object of class "dmd".

Examples

```
# One row per variable, one column per time step
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))

d <- dmd(X, rank = 2, dt = t[2] - t[1])
print(d)
summary(d)

# Extended DMD: lift into a higher-dimensional space where nonlinear
# dynamics become approximately linear
d2 <- dmd(X, lifting = "polynomial", lifting_param = 2)
```

---

dmdc *Dynamic Mode Decomposition with Control (DMDc)*

Description

Identify the forced linear system $x_{t+1} = Ax_t + Bu_t$ from snapshot pairs and control inputs, following Proctor, Brunton and Kutz (2016). Unlike dmd, which takes one contiguous trajectory, dmdc takes explicit pair matrices: X1 holds states at time t, X2 the states one step later, and U the control input applied during each transition. Columns may therefore come from many concatenated trajectories.

Usage

```
dmdc(
  X1,
  X2,
  U = NULL,
  rank_input = NULL,
  rank_output = NULL,
  dt = 1,
  known_B = NULL
)
```

Arguments

X1	Numeric matrix of states at time t (n_states x n_pairs).
X2	Numeric matrix of states at time t+1 (n_states x n_pairs).
U	Numeric matrix of control inputs during each transition (n_inputs x n_pairs), or NULL for an autonomous multi-trajectory fit. A vector is taken as a single input row.
rank_input	Integer truncation rank for the regression-input SVD, or NULL for automatic (99 percent cumulative variance).
rank_output	Integer rank of the output basis (SVD of X2), or NULL to keep everything full-order.

dt	Numeric time step between snapshot pairs.
known_B	Known input matrix B (n_states x n_inputs), or NULL to estimate B jointly with A.

Details

Two identification modes are available. With known_B = NULL (the default), A and B are solved jointly from the stacked regression $[A\ B] = X_2[X_1; U]^+$; this requires the input to be persistently exciting and exogenous (not state feedback). When the input coupling is known by construction, pass it as known_B and only A is estimated.

Value

An S3 object of class "dmdc" with components a, b, a_tilde, b_tilde, basis, eigenvalues_re, eigenvalues_im, singular_values, rank_input, rank_output, dt, n_states, and n_inputs.

References

Proctor, J. L., Brunton, S. L., and Kutz, J. N. (2016). Dynamic Mode Decomposition with Control. *SIAM Journal on Applied Dynamical Systems*, 15(1), 142-161. doi:10.1137/15M1013857

See Also

dmd for autonomous systems, predict.dmdc to simulate the identified system.

Examples

```
# Simulate x_{t+1} = A0 x_t + B0 u_t with a persistently exciting input
A0 <- matrix(c(0.9, 0, 0.1, 0.8), 2, 2)
B0 <- matrix(c(0.5, 1), 2, 1)
m <- 120
X1 <- matrix(0, 2, m)
X2 <- matrix(0, 2, m)
U <- matrix(0, 1, m)
x <- c(1, -0.5)
for (t in seq_len(m)) {
  u_t <- sin(0.7 * (t - 1)) + 0.5 * cos(2.3 * (t - 1) + 1)
  X1[, t] <- x
  U[, t] <- u_t
  x <- as.numeric(A0 %*% x + B0 * u_t)
  X2[, t] <- x
}

# Identify A and B jointly; both are recovered to machine precision
fit <- dmdc(X1, X2, U, rank_input = 3)
round(fit$a, 6)
round(fit$b, 6)

# Known B: pin the input matrix and estimate only A
fit2 <- dmdc(X1, X2, U, rank_input = 2, known_B = B0)
round(fit2$a, 6)
```

dmdc_spectrum *DMDc spectrum analysis*

Description

Per-mode frequency, growth rate and stability for the operator identified by dmdc. DMDc has no mode amplitudes, so the amplitude column is reported as 0.

Usage

```
dmdc_spectrum(object, dt = NULL)
```

Arguments

object	A dmdc object.
dt	Time step. Uses the stored dt by default.

Value

Data frame with mode information.

Examples

```
A0 <- matrix(c(0.9, 0, 0.1, 0.8), 2, 2)
B0 <- matrix(c(0.5, 1), 2, 1)
m <- 60
X1 <- matrix(0, 2, m); X2 <- matrix(0, 2, m); U <- matrix(0, 1, m)
x <- c(1, -0.5)
for (t in seq_len(m)) {
  u_t <- sin(0.7 * (t - 1)) + 0.5 * cos(2.3 * (t - 1) + 1)
  X1[, t] <- x
  U[, t] <- u_t
  x <- as.numeric(A0 %*% x + B0 * u_t)
  X2[, t] <- x
}
fit <- dmdc(X1, X2, U, rank_input = 3)

dmdc_spectrum(fit)
```

dmdc_stability *DMDc stability analysis*

Description

Classify the stability of the operator identified by dmdc from the eigenvalues of $\tilde{A}$.

Usage

```
dmdc_stability(object, tol = 1e-06)
```

Arguments

object	A dmdc object.
tol	Tolerance for marginal classification.

Value

List with stability information (is_stable, is_unstable, is_marginal, spectral_radius).

Examples

```
A0 <- matrix(c(0.9, 0, 0.1, 0.8), 2, 2)
B0 <- matrix(c(0.5, 1), 2, 1)
m <- 60
X1 <- matrix(0, 2, m); X2 <- matrix(0, 2, m); U <- matrix(0, 1, m)
x <- c(1, -0.5)
for (t in seq_len(m)) {
  u_t <- sin(0.7 * (t - 1)) + 0.5 * cos(2.3 * (t - 1) + 1)
  X1[, t] <- x
  U[, t] <- u_t
  x <- as.numeric(A0 %*% x + B0 * u_t)
  X2[, t] <- x
}
fit <- dmdc(X1, X2, U, rank_input = 3)

dmdc_stability(fit)
```

dmd_dominant_modes *DMD dominant modes*

Description

DMD dominant modes

Usage

```
dmd_dominant_modes(
  object,
  n = 3,
  criterion = c("amplitude", "energy", "stability")
)
```

Arguments

object	A dmd object.
n	Number of modes.
criterion	"amplitude", "energy", or "stability".

Value

Integer vector of 1-based mode indices.

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

# Indices of the most significant modes
dmd_dominant_modes(d, n = 1)
```

---

dmd_error *DMD reconstruction error*

Description

DMD reconstruction error

Usage

```
dmd_error(object)
```

Arguments

object	A dmd object.

Value

List with error metrics (rmse, mae, mape, relative_error).

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

# Reconstruction error metrics
dmd_error(d)
```

`dmd_reconstruct` *Reconstruct data from DMD modes*

Description

Reconstruct data from DMD modes

Usage

```
dmd_reconstruct(object, n_steps = NULL, modes = NULL)
```

Arguments

`object`	A dmd object.
`n_steps`	Number of time steps. Defaults to original length.
`modes`	Integer vector of mode indices (1-based), or NULL for all.

Value

Numeric matrix.

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

recon <- dmd_reconstruct(d, n_steps = ncol(X))
dim(recon)
```

dmd_residual *DMD residual analysis*

Description

DMD residual analysis

Usage

```
dmd_residual(object)
```

Arguments

object	A dmd object.

Value

List with residual_norm and residual_relative.

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

dmd_residual(d)
```

dmd_spectrum *DMD spectrum analysis*

Description

DMD spectrum analysis

Usage

```
dmd_spectrum(object, dt = NULL)
```

Arguments

object	A dmd object.
dt	Time step. Uses stored dt by default.

Value

Data frame with mode information.

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

# Frequency, growth rate, amplitude and stability of each mode
dmd_spectrum(d)
```

`dmd_stability` *DMD stability analysis*

Description

DMD stability analysis

Usage

```
dmd_stability(object, tol = 1e-06)
```

Arguments

`object`	A dmd object.
`tol`	Tolerance for marginal classification.

Value

List with stability information.

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

dmd_stability(d)
```

extended_standard_map *Extended standard map (3D)*

Description

Extended standard map (3D)

Usage

```
extended_standard_map(state, epsilon = 0.01, delta = 0.001)
```

Arguments

state	Numeric vector of length 3.
epsilon	Perturbation parameter.
delta	Coupling parameter.

Value

Updated state vector.

Examples

```
extended_standard_map(c(0.1, 0.2, 0.3))
```

froeschle_map *Froeschle map (4D coupled standard maps)*

Description

Froeschle map (4D coupled standard maps)

Usage

```
froeschle_map(state, epsilon1 = 0.02, epsilon2 = 0.02, eta = 0.01)
```

Arguments

state	Numeric vector of length 4.
epsilon1	First perturbation.
epsilon2	Second perturbation.
eta	Coupling parameter.

Value

Updated state vector.

Examples

```
froeschle_map(c(0.1, 0.2, 0.3, 0.4))
```

`generate_trajectory` *Generate a trajectory from a built-in map*

Description

Generate a trajectory from a built-in map

Usage

```
generate_trajectory(map_name, initial_condition, n_iter, ...)
```

Arguments

`map_name`	Character: "standard", "froeschle", "extended_standard", "henon", or "logistic".
`initial_condition`	Numeric vector.
`n_iter`	Integer number of iterations.
`...`	Map parameters passed as named arguments.

Value

Numeric matrix (n_dim x n_iter+1).

Examples

```
# Chirikov standard map with default parameters
traj <- generate_trajectory("standard", c(0.1, 0.2), 100)
dim(traj)

# Map parameters are passed through `...`
henon <- generate_trajectory("henon", c(0, 0), 100, a = 1.4, b = 0.3)
logistic <- generate_trajectory("logistic", 0.5, 100, r = 3.9)
```

gla *Generalized Laplace Analysis*

Description

Generalized Laplace Analysis

Usage

```
gla(y, eigenvalues = NULL, n_eigenvalues = 5, tol = 1e-06, max_iter = NULL)
```

Arguments

y	Numeric matrix (n_obs x n_time).
eigenvalues	Complex vector of known eigenvalues, or NULL.
n_eigenvalues	Number of eigenvalues to estimate.
tol	Convergence tolerance.
max_iter	Maximum iterations, or NULL.

Value

An S3 object of class "gla".

Examples

```
t <- seq(0, 10, length.out = 200)
y <- rbind(sin(t), cos(t))

g <- gla(y, n_eigenvalues = 2)
print(g)
```

gla_reconstruct *Reconstruct from GLA*

Description

Reconstruct from GLA

Usage

```
gla_reconstruct(object, modes_to_use = NULL)
```

Arguments

object	A gla object.
modes_to_use	Integer vector of mode indices (1-based), or NULL.

Value

Numeric matrix.

Examples

```
t <- seq(0, 10, length.out = 200)
y <- rbind(sin(t), cos(t))
g <- gla(y, n_eigenvalues = 2)

recon <- gla_reconstruct(g)
dim(recon)
```

`hankel_dmd` *Hankel-DMD (Time-Delay Embedding DMD)*

Description

Hankel-DMD (Time-Delay Embedding DMD)

Usage

```
hankel_dmd(y, delays = NULL, rank = NULL, dt = 1)
```

Arguments

`y`	Numeric matrix (n_obs x n_time).
`delays`	Integer number of delays, or NULL for automatic.
`rank`	Integer truncation rank, or NULL for automatic.
`dt`	Numeric time step.

Value

An S3 object of class "hankel_dmd".

Examples

```
# A scalar signal, as a 1-row matrix
t <- seq(0, 4 * pi, length.out = 200)
y <- matrix(sin(t), nrow = 1)

h <- hankel_dmd(y, delays = 10)
print(h)
```

hankel_reconstruct *Reconstruct from Hankel-DMD*

Description

Reconstruct from Hankel-DMD

Usage

```
hankel_reconstruct(object, n_steps)
```

Arguments

object	A hankel_dmd object.
n_steps	Number of time steps.

Value

Numeric matrix.

Examples

```
t <- seq(0, 4 * pi, length.out = 200)
y <- matrix(sin(t), nrow = 1)
h <- hankel_dmd(y, delays = 10)

recon <- hankel_reconstruct(h, 50)
```

harmonic_time_average *Harmonic Time Average*

Description

Harmonic Time Average

Usage

```
harmonic_time_average(
  map_name,
  initial_condition,
  observable = "sin_pi",
  omega = 0.1,
  n_iter = 10000,
  ...
)
```

Arguments

map_name	Character map name.
initial_condition	Numeric vector.
observable	Character observable name.
omega	Numeric frequency.
n_iter	Integer iterations.
...	Map parameters.

Value

List with magnitude, phase, hta_re, hta_im.

Examples

```
# Harmonic time average of an orbit of the Chirikov standard map
harmonic_time_average("standard", c(0.1, 0.2), "sin_pi", 0.1, 500)
```

---

henon_map *Henon map (2D dissipative)*

Description

Henon map (2D dissipative)

Usage

```
henon_map(state, a = 1.4, b = 0.3)
```

Arguments

state	Numeric vector c(x, y).
a	Parameter a.
b	Parameter b.

Value

Updated state vector.

Examples

```
henon_map(c(0, 0))

# Classic chaotic parameters
henon_map(c(0, 0), a = 1.4, b = 0.3)
```

`hta_convergence` *HTA convergence analysis*

Description

HTA convergence analysis

Usage

```
hta_convergence(
  map_name,
  initial_condition,
  observable = "sin_pi",
  omega = 0.1,
  n_iter = 10000,
  ...
)
```

Arguments

`map_name`	Character map name.
`initial_condition`	Numeric vector.
`observable`	Character observable name.
`omega`	Numeric frequency.
`n_iter`	Integer iterations.
`...`	Map parameters.

Value

List with times, hta_magnitudes, convergence_rate, dynamics_type.

Examples

```
# How the time average converges along the orbit
conv <- hta_convergence("standard", c(0.1, 0.2), "sin_pi", 0.1, 500)
str(conv)
```

logistic_map *Logistic map (1D)*

Description

Logistic map (1D)

Usage

```
logistic_map(state, r = 3.9)
```

Arguments

state	Numeric scalar.
r	Growth rate parameter.

Value

Updated state value.

Examples

```
logistic_map(0.5)

# In the chaotic regime
logistic_map(0.5, r = 3.9)
```

mesochronic_compute *Mesochronic harmonic plot computation*

Description

Mesochronic harmonic plot computation

Usage

```
mesochronic_compute(
  map_name,
  x_range = c(0, 1),
  y_range = c(0, 1),
  resolution = 100,
  observable = "sin_pi",
  omega = 0.1,
  n_iter = 30000,
  ...
)
```

Arguments

map_name	Character map name.
x_range	Numeric vector c(min, max).
y_range	Numeric vector c(min, max).
resolution	Integer grid resolution.
observable	Character observable name.
omega	Numeric frequency.
n_iter	Integer iterations.
...	Map parameters.

Value

List with hta_matrix, phase_matrix, x_coords, y_coords.

Examples

```
# Mesochronic plot over a coarse grid. Raise `resolution` and `n_iter`
# for publication-quality figures; both cost time roughly linearly.
mhp <- mesochronic_compute("standard", c(0, 1), c(0, 1), 10, "sin_pi", 0.1, 100)
str(mhp)
```

---

predict.dmd *Predict from DMD model*

Description

Predict from DMD model

Usage

```
## S3 method for class 'dmd'
predict(object, n_ahead = 10, x0 = NULL, method = c("modes", "matrix"), ...)
```

Arguments

object	A dmd object.
n_ahead	Number of steps to predict.
x0	Optional initial condition vector.
method	"modes" (default) or "matrix".
...	Additional arguments (ignored).

Value

Numeric matrix of predictions.

Examples

```
t <- seq(0, 10, length.out = 100)
X <- rbind(sin(t), cos(t))
d <- dmd(X, rank = 2, dt = t[2] - t[1])

# Forecast 10 steps beyond the input
pred <- predict(d, n_ahead = 10)
dim(pred)
```

`predict.dmdc` *Predict from a DMDc model*

Description

Simulate the identified system $x_{t+1} = Ax_t + Bu_t$ forward from an initial state under a given control input sequence.

Usage

```
## S3 method for class 'dmdc'
predict(object, U = NULL, x0 = NULL, n_ahead = NULL, ...)
```

Arguments

`object`	A dmdc object.
`U`	Numeric matrix of control inputs (n_inputs x n_steps); the number of columns sets the prediction horizon. A vector is taken as a single input row. NULL applies zero input for `n_ahead` steps.
`x0`	Initial state vector. Defaults to the first stored snapshot.
`n_ahead`	Number of steps when `U` is NULL; if both are given it must match `ncol(U)`.
`...`	Additional arguments (ignored).

Value

Numeric matrix of predicted states $x_1 \dots x_k$ (n_states x k).

Examples

```
A0 <- matrix(c(0.9, 0, 0.1, 0.8), 2, 2)
B0 <- matrix(c(0.5, 1), 2, 1)
m <- 120
X1 <- matrix(0, 2, m)
X2 <- matrix(0, 2, m)
U <- matrix(0, 1, m)
x <- c(1, -0.5)
for (t in seq_len(m)) {
  u_t <- sin(0.7 * (t - 1)) + 0.5 * cos(2.3 * (t - 1) + 1)
```

```
  X1[, t] <- x
  U[, t] <- u_t
  x <- as.numeric(A0 %*% x + B0 * u_t)
  X2[, t] <- x
}
fit <- dmdc(X1, X2, U, rank_input = 3)

# Replaying the training inputs reproduces the observed successors
pred <- predict(fit, U = U)
max(abs(pred - X2))

# Zero-input (free) response from a chosen state
free <- predict(fit, x0 = c(1, 1), n_ahead = 10)
dim(free)
```

`predict.gla` *Predict from GLA*

Description

Predict from GLA

Usage

```
## S3 method for class 'gla'
predict(object, n_ahead = 10, ...)
```

Arguments

`object`	A gla object.
`n_ahead`	Number of steps to predict.
`...`	Additional arguments (ignored).

Value

Numeric matrix.

Examples

```
t <- seq(0, 10, length.out = 200)
y <- rbind(sin(t), cos(t))
g <- gla(y, n_eigenvalues = 2)

pred <- predict(g, n_ahead = 5)
```

predict.hankel_dmd *Predict from Hankel-DMD*

Description

Predict from Hankel-DMD

Usage

```
## S3 method for class 'hankel_dmd'
predict(object, n_ahead = 10, ...)
```

Arguments

object	A hankel_dmd object.
n_ahead	Number of steps to predict.
...	Additional arguments (ignored).

Value

Numeric matrix.

Examples

```
t <- seq(0, 4 * pi, length.out = 200)
y <- matrix(sin(t), nrow = 1)
h <- hankel_dmd(y, delays = 10)

pred <- predict(h, n_ahead = 10)
```

standard_map *Standard map (Chirikov)*

Description

Standard map (Chirikov)

Usage

```
standard_map(state, epsilon = 0.12)
```

Arguments

state	Numeric vector c(x, y).
epsilon	Perturbation parameter.

Value

Updated state vector.

Examples

```
# One iteration from a given state
standard_map(c(0.1, 0.2))

# Stronger nonlinearity
standard_map(c(0.1, 0.2), epsilon = 0.5)
```

Index