Package 'VNDesign'

July 30, 2026

Title Virtual Noise Algorithm for Computing D-Optimal Designs with Correlated Observations

Version 0.1.0

Description Provides an implementation of the Virtual Noise algorithm for D-optimal experimental designs under correlated observations. The package supports flexible covariance structures, multi-dimensional candidate sets, and analytical or numerical computation of regression gradients. It offers a unified framework for constructing design matrices, defining covariance models, and computing optimal design measures.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Àngela Sebastià Bargues [aut, cre], Irene García-Camacha Gutiérrez [aut]

Maintainer Àngela Sebastià Bargues <angela.sebastia@uv.es>

Repository CRAN

Date/Publication 2026-07-30 16:50:24 UTC

Contents

`build_Sigma` *Build a covariance matrix* Σ

Description

Generates a covariance matrix Σ for a set of locations `coords`, using different covariance kernel families (exponential, Gaussian, Matérn, spherical, triangular, Brownian, AR(1), or a user-defined covariance).

Usage

```
build_Sigma(
  coords,
  type = c("exp", "gauss", "matern32", "matern52", "spherical", "triangular",
    "triangular_trunc", "brownian", "ar1", "custom"),
  sigma2 = 1,
  range = 1,
  nugget = 0,
  phi = NULL,
  cov_fun = NULL,
  return_dist = FALSE
)
```

Arguments

`coords`	Coordinates of the observation points. It can be: • a numeric vector (1D case), • or a matrix / `data.frame` of dimension $N \times d$, where d is the dimension of the coordinate space.
`type`	Type of covariance structure. Options: `"exp"`, `"gauss"`, `"matern32"`, `"matern52"`, `"spherical"`, `"triangular"`, `"triangular_trunc"`, `"brownian"`, `"ar1"`, `"custom"`.
`sigma2`	Marginal variance parameter multiplying the covariance kernel. If K denotes the correlation/kernel matrix, the covariance matrix is constructed as $\Sigma = \sigma^2 K$ (up to the nugget effect). Larger values imply greater overall variability. Must be a positive scalar.
`range`	Scale/range parameter. Must be a positive scalar. In distance-based kernels, it controls the rate of decay of the covariance. In `type="ar1"`, it acts as a scaling factor in the power `phi^(D/range)`.

nugget	Independent noise (nugget effect) added to the diagonal of the covariance matrix. Represents measurement error or small-scale variability not explained by the covariance structure, and may improve numerical stability. Must be a non-negative scalar.
phi	AR(1) parameter (used only if type="ar1"). Must satisfy |phi| < 1.
cov_fun	User-defined covariance function (used only if type = "custom"). The function must accept two locations u and v (numeric vectors of length d) and return their covariance $C(u, v)$ as a numeric scalar.
return_dist	If TRUE, returns both the covariance matrix Sigma and the corresponding distance matrix D.

Details

The implemented covariance structures include exponential, Gaussian, Matérn, spherical, triangular, Brownian and AR(1) covariance models.

For distance-based covariance structures, the pairwise distance matrix D is computed as:

- abs(outer(coords, coords, "-")) when coords is one-dimensional,
- as.matrix(dist(coords, method = "euclidean")) when coords is multidimensional.

For type = "brownian", the Brownian covariance kernel $K(u, v) = \min(u, v)/range$ is used, and therefore coords must be one-dimensional.

For type = "custom", the covariance matrix is constructed by evaluating cov_fun(u,v) for each pair of candidate points.

Value

If return_dist = FALSE (default), returns the covariance matrix Sigma of dimension $N \times N$.

If return_dist = TRUE, returns a list containing:

- Sigma: covariance matrix.
- D: pairwise distance matrix.

---

build_Xchi_grad *Build χ using an analytical gradient*

Description

Constructs the candidate regressor matrix Xchi by evaluating the analytical gradient of the predictor $\eta(x, \theta)$ with respect to the parameter vector theta at each candidate point in chi.

Usage

```
build_Xchi_grad(chi, theta, grad_fun)
```

Arguments

chi	Candidate set. Either: • a numeric vector of length N (one-dimensional case), • or a matrix / data.frame of dimension $N \times d$, where N is the number of candidate points and d is the dimension of the coordinate space.
theta	Numeric parameter vector of length p.
grad_fun	Function grad_fun(x, theta) returning the analytical gradient of $\eta(x, \theta)$ with respect to theta. The function must return a numeric vector of length length(theta). If chi is multidimensional, x is a numeric vector of length d.

Value

Numeric matrix Xchi of dimension $N \times p$, where row i contains the gradient evaluated at the candidate point x_i.

Examples

```
## ------------------------------------------
## Example 1D
## ------------------------------------------

chi <- seq(-1, 1, length.out = 5)
theta <- c(0, 1)

# eta(x) = theta1 + theta2 * x
grad_fun <- function(x, theta) {
  c(1, x)
}

build_Xchi_grad(chi, theta, grad_fun)

## ------------------------------------------
## Example 2D
## ------------------------------------------

chi2 <- cbind(x1 = c(0, 1), x2 = c(0.5, 0.8))
theta <- c(0, 1, -1)

grad_fun2 <- function(x, theta) {
  c(1, x[1], x[2])
}

build_Xchi_grad(chi2, theta, grad_fun2)
```

build_Xchi_numeric *Build χ via centered finite differences*

Description

Constructs the candidate regressor matrix `Xchi` by numerically approximating the gradient of the predictor $\eta(x, \theta)$ with respect to the parameter vector `theta` using centered finite differences.

Usage

```
build_Xchi_numeric(chi, theta, eta_fun, step = NULL, rel_step = 1e-06)
```

Arguments

`chi`	Candidate set. Either: • a numeric vector of length N (one-dimensional case), • or a matrix / `data.frame` of dimension $N \times d$, where N is the number of candidate points and d is the dimension of the coordinate space.
`theta`	Numeric parameter vector of length `p`.
`eta_fun`	Function `eta_fun(x, theta)` returning the predictor value $\eta(x, \theta)$ as a numeric scalar. If `chi` is multidimensional, `x` is a numeric vector of length `d`.
`step`	Step size(s) used in the finite-difference approximation. Can be: • `NULL`: step sizes are automatically computed as `rel_step * max(abs(theta_j), 1)`, • a numeric scalar: same step size for all parameters, • a numeric vector of length `p`.
`rel_step`	Relative step size used when `step = NULL`. Default is `1e-6`.

Details

Uses the centered scheme

$$\partial\eta(x, \theta)/\partial\theta_j \approx \{\eta(x, \theta + h_j e_j) - \eta(x, \theta - h_j e_j)\}/(2h_j)$$

which has error order $O(h^2)$.

Value

Numeric matrix $N \times p$ with the finite-difference gradient approximation.

`efficiency_bound` *Compute an efficiency lower bound for a given design*

Description

Computes a lower bound efficiency of a design ξ under the D-optimality criterion. The bound is based on the formulation proposed in the literature for correlated observations, and is used as a stopping criterion for the Virtual Noise algorithm.

Usage

```
efficiency_bound(xi_vec, Xchi, Sigma_chi, M_tilde, kappa, n_target)
```

Arguments

`xi_vec`	Numeric vector of design weights (length N).
`Xchi`	Numeric matrix $N \times p$ of regressors or gradients evaluated at the candidate points.
`Sigma_chi`	Numeric covariance matrix $N \times N$.
`M_tilde`	Current information matrix used by the Virtual Noise algorithm.
`kappa`	Virtual noise tuning parameter.
`n_target`	Target exact design size used in the efficiency diagnostic.

Details

The efficiency is expressed in terms of the information matrix $M(\xi)$ and a deviation term δ derived from the sensitivity function $h(x, \xi)$.

The efficiency diagnostic is computed as

$$\mathrm{eff}_\Phi(\xi) \geq \frac{n\,\Phi(M(\xi))}{n\,\Phi(M(\xi)) + \kappa\,\delta},$$

where $\Phi(M) = |M|^{1/p}$ and $\delta = \max_x h(x, \xi) - \sum_x \xi(x) h(x, \xi)$.

The sensitivity function $h(x, \xi)$ is constructed using the transformed matrix $T(\xi)$ and the gradient of Φ.

This quantity provides a practical measure of convergence and can be used as a stopping rule in iterative design algorithms.

Value

A list with the following components:

- `eff_lb`: lower bound for the efficiency of the design.
- `delta`: deviation term measuring how far the design is from optimality.
- `phi`: value of the D-optimality criterion in its multiplicative form $|M(\xi)|^{1/p}$.
- `h_max`: maximum value of the sensitivity function.
- `h_xi_mean`: weighted average of the sensitivity function under ξ.

Note

This function currently supports only the D-optimality criterion.

`exact_design` *Convert a Virtual Noise design measure into an exact design*

Description

Converts an approximate design measure obtained from `vndesign_Dopt` or `virtual_noise_Dopt` into an exact design with a fixed number of design points.

Usage

```
exact_design(
  res,
  Xchi,
  Sigma_chi,
  chi = NULL,
  n,
  method = c("random_best", "quantile", "quantile_endpoints"),
  B = 100,
  seed = NULL
)
```

Arguments

`res`	Output object returned by `vndesign_Dopt` or `virtual_noise_Dopt`. The object must contain a numeric component `xi` representing the approximate design weights.
`Xchi`	Numeric matrix $N \times p$ containing the regressors or gradients evaluated at the candidate points.
`Sigma_chi`	Numeric covariance matrix of dimension $N \times N$.
`chi`	Optional candidate set. If `NULL`, candidate indices are used instead of candidate coordinates.
`n`	Target size of the exact design, i.e. the number of selected design points.
`method`	Method used to construct the exact design. Use `"random_best"` for repeated weighted random sampling, or one of the quantile-based methods for one-dimensional candidate sets.
`B`	Number of random candidate designs generated when `method = "random_best"`.
`seed`	Optional random seed for reproducibility.

Value

A list with components:

- `idx`: selected candidate indices.
- `points`: selected candidate points.
- `n`: number of selected points.
- `criterion_value`: D-criterion value of the exact design.

Examples

```
chi <- seq(-1, 1, length.out = 10)
Xchi <- cbind(1, chi)
Sigma_chi <- build_Sigma(chi, type = "exp", range = 0.4)
res <- list(xi = rep(1 / length(chi), length(chi)))

exact_design(res, Xchi, Sigma_chi, chi = chi, n = 3, B = 5, seed = 1)
```

---

plot_convergence *Plot convergence of the D-optimality criterion*

Description

Plots the value of the D-optimality criterion at each iteration of the Virtual Noise algorithm in order to visualize the convergence of the approximate design sequence. The initial value at iteration 0 is omitted from the plot.

Usage

```
plot_convergence(res, main = NULL, xlab = "Iteration", ylab = NULL)
```

Arguments

`res`	Output object returned by `vndesign_Dopt` or `virtual_noise_Dopt` containing the sequence of D-criterion values across iterations.
`main`	Optional title of the plot. If `NULL`, no title is shown.
`xlab`	Label for the x-axis. Default is `"Iteration"`.
`ylab`	Label for the y-axis. Default is `"D-criterion value"`.

Value

Produces a base R convergence plot and returns `NULL` invisibly.

Examples

```
## Run the Virtual Noise algorithm
chi <- seq(-1, 1, length.out = 50)
res <- vndesign_Dopt(
  chi = chi,
  theta = c(0, 1),
  grad_fun = function(x, theta) c(1, x),
  kappa = 0.99 / length(chi),
  max_iter = 200
)

## Plot convergence of the D-criterion
plot_convergence(res)
```

`plot_design_weights` *Plot design weights over the candidate set*

Description

Plots the design weights returned by `vndesign_Dopt` or `virtual_noise_Dopt`. One-dimensional designs are shown as vertical lines; two-dimensional designs are shown as points with size proportional to the square root of the weight.

Usage

```
plot_design_weights(res, chi = NULL, main = NULL)
```

Arguments

`res`	Output object returned by `vndesign_Dopt` or `virtual_noise_Dopt`.
`chi`	Optional candidate set. If `NULL`, candidate indices are used.
`main`	Optional title of the plot. If `NULL`, no title is shown.

Value

Produces a base R visualization of the design weights over the candidate set and returns `NULL` invisibly.

Examples

```
## Run the Virtual Noise algorithm
chi <- seq(-1, 1, length.out = 50)
res <- vndesign_Dopt(
  chi = chi,
  theta = c(0, 1),
  grad_fun = function(x, theta) c(1, x),
  kappa = 0.99 / length(chi),
```

```
  max_iter = 200
)

## Plot design weights
plot_design_weights(res, chi = chi)
```

`plot_efficiency` *Plot efficiency diagnostic*

Description

Plots the efficiency diagnostic values stored in an object returned by `vndesign_Dopt` or `virtual_noise_Dopt` with `return_eff_bound = TRUE`. The initial value at iteration 0 is omitted from the plot.

Usage

```
plot_efficiency(res, main = NULL)
```

Arguments

`res`	Output object containing `eff_lb_vals`.
`main`	Optional title of the plot. If `NULL`, no title is shown.

Value

Produces a base R visualization of the efficiency diagnostic over the iterations of the Virtual Noise algorithm and returns `NULL` invisibly.

Examples

```
## Run the Virtual Noise algorithm with efficiency tracking
chi <- seq(-1, 1, length.out = 50)
res <- vndesign_Dopt(
  chi = chi,
  theta = c(0, 1),
  grad_fun = function(x, theta) c(1, x),
  kappa = 0.99 / length(chi),
  max_iter = 200,
  return_eff_bound = TRUE,
  n_target = 3
)

## Plot efficiency diagnostic
plot_efficiency(res)
```

`virtual_noise_Dopt` *Virtual Noise algorithm for D-optimal design*

Description

Implements the Virtual Noise (VN) algorithm for constructing approximate D-optimal experimental designs under correlated observations.

Usage

```
virtual_noise_Dopt(
  Xchi,
  Sigma_chi,
  chi = NULL,
  kappa,
  delta = 0.001,
  max_iter = NULL,
  xi_init = NULL,
  verbose = TRUE,
  vn_alt = 3,
  vn_lambda = 40,
  vn_delta = 1e-06,
  stop_rule = c("max_iter", "efficiency"),
  return_eff_bound = FALSE,
  n_target = NULL,
  eff_tol = NULL
)
```

Arguments

`Xchi`	Numeric matrix of dimension $N \times p$ containing the regressors or gradients evaluated at the candidate points.
`Sigma_chi`	Numeric covariance matrix of dimension $N \times N$ associated with the candidate points.
`chi`	Candidate set used to label the selected and support points in the output. Can be: • a numeric vector of length N, • a matrix / `data.frame` with N rows, • or `NULL`. If `NULL`, candidate indices `1:N` are used.
`kappa`	Virtual-noise tuning parameter controlling the weight penalization and the VN selection rules. Smaller values of `kappa` generally allow larger support sizes, while in practice `kappa` is often chosen close to $0.99/N$, where N is the number of candidate points.
`delta`	Numerical tolerance used in Step 2 of the VN selection rule.

max_iter	Maximum number of iterations of the VN algorithm.
xi_init	Optional initial design measure of length N. If NULL, a uniform design measure is used. If supplied, the vector is normalized to sum to one.
verbose	Logical; if TRUE, prints iteration progress.
vn_alt	Integer in 1:4 selecting the virtual-noise alternative used to construct the additional diagonal penalization term.
vn_lambda	Numeric lambda parameter used by virtual-noise alternatives 3 and 4.
vn_delta	Numeric stabilization parameter used by virtual-noise alternatives 1 and 2.
stop_rule	Stopping rule used by the algorithm. Use "max_iter" to stop after max_iter iterations, or "efficiency" to stop when the D-efficiency diagnostic reaches eff_tol. The efficiency stopping rule automatically enables return_eff_bound.
return_eff_bound	Logical; if TRUE, returns a Pázman-type D-efficiency diagnostic and related values along the iterations.
n_target	Target exact design size used in the computation of the D-efficiency diagnostic. Required when return_eff_bound = TRUE or stop_rule = "efficiency".
eff_tol	Efficiency tolerance in (0,1] used by stop_rule = "efficiency". The algorithm stops when the efficiency diagnostic reaches this value. Required for that stopping rule.

Details

The algorithm iteratively updates a design measure over a candidate set chi by selecting candidate points according to D-optimality sensitivity rules and incorporating an additional virtual-noise penalization mechanism.

At each iteration, the method:

1. Constructs a modified covariance matrix including the virtual-noise term,
2. Computes the corresponding information matrix,
3. Evaluates D-optimality selection quantities over the candidate set,
4. Selects a candidate point according to the VN decision rules,
5. Updates the design measure exactly.

The algorithm updates the weight vector xi over the candidate set by selecting, at each iteration, a candidate point x_s and performing the exact update

$$\xi^{(s)} = \frac{s-1}{s}\xi^{(s-1)} + \frac{1}{s}e_{x_s},$$

where e_{x_s} denotes the unit vector associated with the selected candidate point.

The VN component modifies the covariance structure through an additional penalization term controlled by kappa and the selected VN alternative vn_alt.

The D-criterion is implemented as -log det(M_tilde) where M_tilde = X^T Sigma_tilde^{-1} X and Sigma_tilde = Sigma_chi + diag(sigma2_xi).

This implementation follows the classical VN heuristic. The optional efficiency values returned by return_eff_bound = TRUE are useful convergence diagnostics; they should not be read as certificates from the convex linear-programming formulation of Pázman et al. (2022).

Value

A list with components:

- `xi`: final weights (length N).
- `crit_vals`: D-criterion values over iterations.
- `iterations`: number of executed iterations.
- `chosen_indices`: indices selected at each iteration.
- `chosen_points`: selected points from `chi`.
- `support_idx`: indices with positive weight at the end.
- `support_points`: support points from `chi`.
- `support_weights`: normalized support weights.

If `return_eff_bound = TRUE`, the list also contains `eff_lb_vals`, `delta_vals`, `phi_vals`, `h_max_vals`, and `h_xi_mean_vals`.

References

Pázman, A. (1999). Further developments on virtual noise methods for exact optimal designs.

Pázman, A., Hainy, M., and Müller, W. G. (2022). A convex approach to optimum design of experiments with correlated observations.

López-Fidalgo, J., and Wong, W. K. (2025). Optimal Designs for Correlated Data. *Annual Review of Statistics and Its Application*.

`vndesign_Dopt` *Main interface for constructing D-optimal designs using the Virtual Noise algorithm*

Description

This is the main user-level function of the package. It provides a complete workflow for constructing approximate D-optimal designs under correlated observations using the Virtual Noise (VN) algorithm implemented in `virtual_noise_Dopt`.

Usage

```
vndesign_Dopt(
  chi,
  Xchi = NULL,
  theta = NULL,
  grad_fun = NULL,
  eta_fun = NULL,
  kappa,
  max_iter = NULL,
  xi_init = NULL,
```

```
  delta = 0.001,
  verbose = TRUE,
  stop_rule = c("max_iter", "efficiency"),
  cov_type = c("exp", "gauss", "matern32", "matern52", "spherical", "triangular",
    "triangular_trunc", "brownian", "ar1", "custom"),
  sigma2 = 1,
  range = 1,
  nugget = 0,
  phi = NULL,
  cov_fun = NULL,
  vn_alt = 3,
  vn_lambda = 40,
  vn_delta = 1e-06,
  num_step = NULL,
  num_rel_step = 1e-06,
  return_components = FALSE,
  return_eff_bound = FALSE,
  n_target = NULL,
  eff_tol = NULL
)
```

Arguments

chi	Candidate set. Either: • a numeric vector of length N (one-dimensional case), • or a matrix / data.frame of dimension $N \times d$, where N is the number of candidate points and d is the dimension of the coordinate space.
Xchi	Optional numeric matrix of dimension $N \times p$ containing the regressors or gradients evaluated at the candidate points. If supplied, model-based arguments are ignored.
theta	Numeric parameter vector of length p. Required when Xchi is not provided.
grad_fun	Function grad_fun(x, theta) returning the analytical gradient of $\eta(x, \theta)$ with respect to theta. The function must return a numeric vector of length length(theta).
eta_fun	Function eta_fun(x, theta) returning the predictor value $\eta(x, \theta)$ as a numeric scalar. Used only when grad_fun is NULL, in which case gradients are approximated numerically using centered finite differences.
kappa	Virtual-noise tuning parameter controlling the weight penalization and the VN selection rules. For the classical VN algorithm implemented here, kappa is typically chosen slightly below the initial uniform design weight, for example $0.99/N$, where N is the number of candidate points.
max_iter	Maximum number of iterations of the Virtual Noise algorithm. If stop_rule = "efficiency", this acts as a safety limit.
xi_init	Optional initial design measure of length N. If NULL, a uniform design measure is used. If supplied, the vector is normalized to sum to one.
delta	Numerical tolerance used in Step 2 of the VN selection rule.
verbose	Logical; if TRUE, prints iteration progress information.

`stop_rule` Stopping rule used by the algorithm. Use `"max_iter"` to stop after `max_iter` iterations, or `"efficiency"` to stop when the D-efficiency diagnostic reaches `eff_tol`. The efficiency stopping rule automatically enables `return_eff_bound`.

`cov_type` Covariance structure passed to `build_Sigma`.

`sigma2` Marginal variance parameter multiplying the covariance kernel. If K denotes the covariance kernel, the covariance matrix is constructed as $\Sigma = \sigma^2 K$ (up to the nugget effect). Larger values imply greater overall variability.

`range` Range or scale parameter of the covariance structure.

`nugget` Independent noise (nugget effect) added to the diagonal of the covariance matrix. Represents measurement error or small-scale variability not explained by the covariance structure, and may improve numerical stability.

`phi` AR(1) parameter used only when `cov_type = "ar1"`.

`cov_fun` User-defined covariance function used only when `cov_type = "custom"`. The function must accept two locations `u` and `v` (numeric vectors of length d) and return their covariance $C(u, v)$ as a numeric scalar.

`vn_alt` Integer in `1:4` selecting the virtual-noise alternative used to construct the additional diagonal penalization term.

`vn_lambda` Numerical parameter used by virtual-noise alternatives 3 and 4.

`vn_delta` Numerical stabilization parameter used by virtual-noise alternatives 1 and 2.

`num_step` Step size(s) used in the finite-difference approximation. Can be:

- `NULL`: step sizes are automatically computed as `num_rel_step * max(abs(theta_j), 1)`,
- a numeric scalar: same step size for all parameters,
- a numeric vector of length `length(theta)`.

`num_rel_step` Relative step size used when `num_step = NULL`. Default is `1e-6`.

`return_components`
Logical; if `TRUE`, additionally returns `Xchi`, `Sigma_chi`, and the pairwise distance matrix `D`.

`return_eff_bound`
Logical; if `TRUE`, computes and returns a Pázman-type D-efficiency diagnostic together with additional quantities along the iterations. Automatically enabled when `stop_rule = "efficiency"`.

`n_target` Target exact design size used in the computation of the D-efficiency diagnostic. Required when `return_eff_bound = TRUE` or `stop_rule = "efficiency"`.

`eff_tol` Efficiency tolerance in `(0,1]` used by `stop_rule = "efficiency"`. The algorithm stops when the efficiency diagnostic reaches this value.

Details

The function acts as a high-level wrapper integrating most of the package functionality, including:

- construction or validation of the candidate regressor matrix `Xchi`,
- analytical or numerical gradient evaluation,
- covariance matrix construction via `build_Sigma`,

- execution of the Virtual Noise algorithm,
- computation of optional efficiency diagnostics,
- and generation of visualization-ready outputs.

The candidate set `chi` may be:

- a numeric vector (one-dimensional case),
- or a matrix / `data.frame` of dimension $N \times d$, where each row represents a candidate point.

Two modes of operation are supported:

Direct mode: If `Xchi` is provided, it is used directly and no model specification through `theta`, `grad_fun`, or `eta_fun` is required.

Model-based mode: If `Xchi` is `NULL`, the function constructs it from `theta` using either:

- an analytical gradient function `grad_fun`,
- or numerical finite differences via `eta_fun`.

This function provides a high-level interface to the Virtual Noise methodology implemented in `virtual_noise_Dopt`.

For the D-criterion, the implementation minimizes $-\log|M|$, which is equivalent to maximizing the determinant of the information matrix.

When `return_eff_bound = TRUE`, the function also computes a Pázman-type D-efficiency diagnostic along the iterations. If `stop_rule = "efficiency"`, this diagnostic is additionally used as a stopping criterion, while `max_iter` acts as a safety limit. Since this package implements the classical VN heuristic rather than the convex linear-programming formulation of Pázman et al. (2022), these values should be interpreted as convergence diagnostics.

Value

A list containing the output of `virtual_noise_Dopt`. Main components include:

- `xi`: final design weights over the candidate set.
- `crit_vals`: D-criterion values over the iterations.
- `iterations`: number of executed iterations.
- `support_idx`: indices with positive final weights.
- `support_points`: support points from `chi`.
- `support_weights`: normalized support weights.

If `return_eff_bound = TRUE`, the output also includes the efficiency diagnostic values and related quantities. If `return_components = TRUE`, the output additionally includes `Xchi`, `Sigma_chi`, and `D`.

See Also

`virtual_noise_Dopt`, `efficiency_bound`, `build_Sigma`, `build_Xchi_grad`, `build_Xchi_numeric`

Examples

```
chi <- seq(-1, 1, length.out = 15)
theta <- c(0, 1)
grad_fun <- function(x, theta) c(1, x)
kappa <- 0.99 / length(chi)

res <- vndesign_Dopt(
  chi = chi,
  theta = theta,
  grad_fun = grad_fun,
  kappa = kappa,
  max_iter = 5,
  verbose = FALSE,
  cov_type = "exp",
  range = 0.4,
  vn_alt = 3,
  vn_lambda = 2
)

head(res$xi)
```

Index