Package 'gek'

March 14, 2025

Title Gradient-Enhanced Kriging

Version 1.1.0

Date 2025-03-13

Description
Gradient-Enhanced Kriging as an emulator for computer experiments based on Maximum-Likelihood estimation.

Imports stats, graphics, dfoptim

License GPL (>= 2)

Depends R (>= 3.5.0)

NeedsCompilation yes

Author Carmen van Meegen [aut, cre] (<https://orcid.org/0000-0003-4125-5088>)

Maintainer Carmen van Meegen <vanmeegen@statistik.tu-dortmund.de>

Repository CRAN

Date/Publication 2025-03-14 12:30:01 UTC

Contents

---

banana *Rosenbrock's Banana Function*

Description

Rosenbrock's banana function is defined by

$$f_{\text{banana}}(x_1, ..., x_d) = \sum_{k=1}^{d-1} (100(x_{k+1} - x_k^2)^2 + (x_k - 1)^2)$$

with $x_k \in [-5, 10]$ for $k = 1, ..., d$ and $d \geq 2$.

Usage

```
banana(x)
bananaGrad(x)
```

Arguments

x	a numeric vector of length d or a numeric matrix with n rows and d columns, where d must be greater than 1.

Details

The gradient of Rosenbrock's banana function is

$$\nabla f_{\text{banana}}(x_1, ..., x_d) = \begin{pmatrix} -400(x_2 - x_1)^2 x_1 + 2(x_1 - 1) \\ 200(x_2 - x_1)^2 - 400x_2(x_3 - x_2^2) + 2(x_2 - 1) \\ \vdots \\ 200(x_{d-1} - x_{d-2})^2 - 400x_{d-1}(x_d - x_{d-1}^2) + 2(x_{d-1} - 1) \\ 200(x_d - x_{d-1}^2) \end{pmatrix}.$$

Rosenbrock's banana function has one global minimum $f_{\text{banana}}(x^\star) = 0$ at $x^\star = (1, \ldots, 1)$.

Value

banana returns the function value of Rosenbrock's banana function at x.

bananaGrad returns the gradient of Rosenbrock's banana function at x.

Author(s)

Carmen van Meegen

References

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150-–194. doi:10.1504/IJMMNO.2013.055204.

Rosenbrock, H. H. (1960). An Automatic Method for Finding the Greatest or least Value of a Function. *The Computer Journal*, **3**(3):175–184. doi:10.1093/comjnl/3.3.175.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

Examples

```
# Contour plot of Rosenbrock's banana function
n.grid <- 50
x1 <- seq(-2, 2, length.out = n.grid)
x2 <- seq(-1, 3, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) banana(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Rosenbrock's banana function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

blockChol *Block Cholesky Decomposition*

Description

blockChol calculates the block Cholesky decomposition of a partitioned matrix of the form

$$A = \begin{pmatrix} K & R^{\mathrm{T}} \\ R & S \end{pmatrix}.$$

Usage

```
blockChol(K, R = NULL, S = NULL, tol = NULL)
```

Arguments

K	a real symmetric positive-definite square submatrix.
R	an (optinal) rectangular submatrix.
S	an (optional) real symmetric positive-definite square submatrix.
tol	an (optional) numeric tolerance, see 'Details'.

Details

To obtain the block Cholesky decomposition

$$\begin{pmatrix} K & R^{\mathrm{T}} \\ R & S \end{pmatrix} = \begin{pmatrix} L^{\mathrm{T}} & 0 \\ Q^{\mathrm{T}} & M^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} L & Q \\ 0 & M \end{pmatrix}$$

the following steps are performed:

1. Calculate $K = L^{\mathrm{T}} L$ with upper triangular matrix L.
2. Solve $L^{\mathrm{T}} Q = R^{\mathrm{T}}$ via forward substitution.
3. Compute $N = S - Q^{\mathrm{T}} Q$ the Schur complement of the block K of the matrix A.
4. Determine $N = M^{\mathrm{T}} M$ with upper triangular matrix M.

The upper triangular matrices L and M in step 1 and 4 are obtained by `chol`. Forward substitution in step 2 is carried out with `backsolve` and the option `transpose = TRUE`.

If `tol` is specified a regularization of the form $A_\epsilon = A + \epsilon I$ is conducted. Here, `tol` is the upper bound for the logarithmic condition number of A_ϵ. Then

$$\epsilon = \max \left\{ \frac{\lambda_{\max}(\kappa(A) - e^{\texttt{tol}})}{\kappa(A)(e^{\texttt{tol}} - 1)}, 0 \right\}$$

is chosen as the minimal "nugget" that is added to the diagonal of A to ensure $\log(\kappa(A_\epsilon)) \leq$ `tol`.

Within **gek** this function is used to calculate the block Cholesky decomposition of the correlation matrix with derivatives. Here `K` is the Kriging correlation matrix. `R` is the matrix containing the first partial derivatives and `S` consists of the second partial derivatives of the correlation matrix `K`.

Value

`blockChol` returns a list with the following components:

L	the upper triangular factor of the Cholesky decomposition of `K`.
Q	the solution of the triangular system `t(L) %*% Q == t(R)`.
M	the upper triangular factor of the Cholesky decomposition of the Schur complement `N`.

If `R` or `S` are not specified, `Q` and `M` are set to `NULL`, i.e. only the Cholesky decomposition of `K` is calculated.

The attribute `"eps"` gives the minimum "nugget" that is added to the diagonal.

Warning

As in chol there is no check for symmetry.

Author(s)

Carmen van Meegen

References

Chen, J., Jin, Z., Shi, Q., Qiu, J., and Liu, W. (2013). Block Algorithm and Its Implementation for Cholesky Factorization.

Gustavson, F. G. and Jonsson, I. (2000). Minimal-storage high-performance Cholesky factorization via blocking and recursion. *IBM Journal of Research and Development*, **44**(6):823–850. doi:10.1147/rd.446.0823.

Ranjan, P., Haynes, R. and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data. *Technometrics*, **53**:366–378. doi:10.1198/TECH.2011.09141.

See Also

chol for the Cholesky decomposition.

backsolve for backward substitution.

blockCor for computing a correlation matrix with derivatives.

Examples

```
# Construct correlation matrix
x <- matrix(seq(0, 1, length = 5), ncol = 1)
res <- blockCor(x, theta = 1, covtype = "gaussian", derivatives = TRUE)
A <- cbind(rbind(res$K, res$R), rbind(t(res$R), res$S))

# Cholesky decomposition of correlation matix without derivatives
cholK <- blockChol(res$K)
cholK
cholK$L == chol(res$K)

# Cholesky decomposition of correlation matix with derivatives
cholA <- blockChol(res$K, res$R, res$S)
cholA <- cbind(rbind(cholA$L, matrix(0, ncol(cholA$Q), nrow(cholA$Q))),
rbind(cholA$Q, cholA$M))
cholA
cholA == chol(A)

# Cholesky decomposition of correlation matix with derivatives with regularization
res <- blockCor(x, theta = 2, covtype = "gaussian", derivatives = TRUE)
A <- cbind(rbind(res$K, res$R), rbind(t(res$R), res$S))
try(blockChol(res$K, res$R, res$S))
blockChol(res$K, res$R, res$S, tol = 35)
```

`blockCor` *Correlation Matrix without or with Derivatives*

Description

Calculation of a correlation matrix with or without derivatives according to the specification of a correlation structure.

Usage

```
blockCor(x, ...)

## Default S3 method:
blockCor(x, theta, covtype = c("matern5_2", "matern3_2", "gaussian"),
derivatives = FALSE, ...)
## S3 method for class 'gekm'
blockCor(x, ...)
```

Arguments

`x`	a numeric `matrix` or an object of class `gekm`.
`theta`	`numeric` vector of length `d` for the hyperparameters.
`covtype`	`character` specifying the correlation function to be used. Must be one of `"matern5_2"`, `"matern3_2"` or `"gaussian"`. See 'Details'.
`derivatives`	`logical`, if `TRUE` the first and second partial derivatives of the correlation matrix are calculated, otherwise not.
`...`	further arguments passed to or from other methods.

Details

The correlation matrix with derivatives is defined as a block matrix of the form

$$\begin{pmatrix} K & R^{\mathrm{T}} \\ R & S \end{pmatrix}.$$

Three correlation functions are currently implemented in `blockCor`:

- Matérn 5/2 kernel with `covtype = "matern5_2"`:

$$\phi(x, x'; \theta) = \prod_{k=1}^{d} \left(1 + \frac{\sqrt{5}|x_k - x'_k|}{\theta_k} + \frac{5|x_k - x'_k|^2}{3\theta_k^2}\right) \exp\left(-\frac{\sqrt{5}|x_k - x'_k|}{\theta_k}\right)$$

- Matérn 3/2 kernel with `covtype = "matern3_2"`:

$$\phi(x, x'; \theta) = \prod_{k=1}^{d} \left(1 + \frac{\sqrt{3}|x_k - x'_k|}{\theta_k}\right) \exp\left(-\frac{\sqrt{3}|x_k - x'_k|}{\theta_k}\right)$$

- Gaussian kernel with `covtype = "gaussian"`:

$$\phi(x, x'; \theta) = \prod_{k=1}^{d} \exp\left(-\frac{(x_k - x'_k)^2}{2\theta_k^2}\right)$$

Value

`blockCor` returns a list with the following components:

`K`	The correlation matrix without derivatives.
`R`	If `derivatives = TRUE`, the correlation matrix with first partial derivatives, otherwise `NULL`.
`S`	If `derivatives = TRUE`, the correlation matrix with second partial derivatives, otherwise `NULL`.

The components of the list can be combined in the following form to constructed the complete correlation matrix with derivatives: `cbind(rbind(K, R), rbind(t(R), S))`.

Author(s)

Carmen van Meegen

References

Koehler, J. and Owen, A. (1996). Computer Experiments. In Ghosh, S. and Rao, C. (eds.), *Design and Analysis of Experiments*, volume 13 of *Handbook of Statistics*, pp. 261–308. Elsevier Science. doi:10.1016/S01697161(96)13011X.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press. https://gaussianprocess.org/gpml/.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer-Verlag. doi:10.1007/9781461214946.

See Also

`blockChol` for block Cholesky decomposition.

`tangents` for drawing tangent lines.

Examples

```
# Some examples of correlation matrices:

x <- matrix(1:4, ncol = 1)
blockCor(x, theta = 1)
blockCor(x, theta = 1, covtype = "gaussian")
blockCor(x, theta = 1, covtype = "gaussian", derivatives = TRUE)
blockCor(x, theta = 1, covtype = "matern3_2", derivatives = TRUE)

res <- blockCor(x, theta = 2, covtype = "gaussian", derivatives = TRUE)
cbind(rbind(res$K, res$R), rbind(t(res$R), res$S))
```

```
# Illustration of correlation functions and their derivatives:

x <- seq(0, 1, length = 100)
X <- matrix(x, ncol = 1)
gaussian <- blockCor(X, theta = 0.25, covtype = "gaussian", derivatives = TRUE)
matern5_2 <- blockCor(X, theta = 0.25, covtype = "matern5_2", derivatives = TRUE)
matern3_2 <- blockCor(X, theta = 0.25, covtype = "matern3_2", derivatives = TRUE)

# Correlation functions and first partial derivatives:
index <- c(10, 20, 40, 80)
par(mar = c(5.1, 5.1, 4.1, 2.1))

# Matern 3/2
plot(x, matern3_2$K[1, ], type = "l", xlab = expression(group("|", x - x*minute, "|")),
ylab = expression(phi(x, x*minute, theta == 0.25)), lwd = 2)
tangents(x[index], matern3_2$K[index, 1], matern3_2$R[index, 1],
length = 0.15, lwd = 2, col = 2)
points(x[index], matern3_2$K[index, 1], pch = 16)

# Matern 5/2
lines(x, matern5_2$K[1, ], lwd = 2, col = 3)
tangents(x[index], matern5_2$K[index, 1], matern5_2$R[index, 1],
length = 0.15, lwd = 2, col = 2)
points(x[index], matern5_2$K[index, 1], pch = 16)

# Gaussian
lines(x, gaussian$K[1, ], lwd = 2, col = 4)
tangents(x[index], gaussian$K[index, 1], gaussian$R[index, 1],
length = 0.15, lwd = 2, col = 2)
points(x[index], gaussian$K[index, 1], pch = 16)

legend("topright", lty = 1, lwd = 2, col = c(1, 3, 4), bty = "n",
legend = c("Matern 3/2", "Matern 5/2", "Gaussian"))


# First and second partial derivatives of correlation functions:
index <- c(5, 10, 20, 40, 80)
par(mar = c(5.1, 6.1, 4.1, 2.1))

# Gaussian
plot(x, matern3_2$R[1, ], type = "l", xlab = expression(group("|", x - x*minute, "|")),
ylab = expression(frac(partialdiff * phi(x, x*minute, theta == 0.25),
partialdiff * x * minute)), lwd = 2)
tangents(x[index], matern3_2$R[1, index], matern3_2$S[index, 1],
length = 0.15, lwd = 2, col = 2)
points(x[index], matern3_2$R[1, index], pch = 16)

# Matern 5/2
lines(x, matern5_2$R[1, ], lwd = 2, col = 3)
tangents(x[index], matern5_2$R[1, index], matern5_2$S[index, 1],
length = 0.15, lwd = 2, col = 2)
```

```
points(x[index], matern5_2$R[1, index], pch = 16)

# Matern 3/2
lines(x, gaussian$R[1, ], lwd = 2, col = 4)
tangents(x[index], gaussian$R[1, index], gaussian$S[index, 1],
length = 0.15, lwd = 2, col = 2)
points(x[index], gaussian$R[1, index], pch = 16)

legend("topright", lty = 1, lwd = 2, col = c(1, 3, 4), bty = "n",
legend = c("Matern 3/2", "Matern 5/2", "Gaussian"))
```

`borehole` *Borehole Function*

Description

The borehole function is defined by

$$f_{\text{borehole}}(x) = \frac{2\pi T_u (H_u - H_l)}{\log(r/r_w)\left(1 + \frac{2LT_u}{\log(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)}$$

with $x = (r_w, r, T_u, H_u, T_l, H_l, L, K_w)$.

Usage

```
borehole(x)
boreholeGrad(x)
```

Arguments

`x`	a numeric `vector` of length `8` or a numeric `matrix` with `n` rows and `8` columns.

Details

The borehole function calculates the water flow rate $[\mathrm{m}^3/\mathrm{yr}]$ through a borehole.

Input	Domain	Distribution	Description
r_w	$[0.05, 0.15]$	$\mathcal{N}(0.1, 0.0161812)$	radius of borehole in m
r	$[100, 50\,000]$	$\mathcal{LN}(7.71, 1.0056)$	radius of influence in m
T_u	$[63070, 115600]$	$\mathcal{U}(63070, 115600)$	transmissivity of upper aquifer in m^2/yr
H_u	$[990, 1100]$	$\mathcal{U}(990, 1110)$	potentiometric head of upper aquifer in m
T_l	$[63.1, 116]$	$\mathcal{U}(63.1, 116)$	transmissivity of lower aquifer in m^2/yr
H_l	$[700, 820]$	$\mathcal{U}(700, 820)$	potentiometric head of lower aquifer in m
L	$[1120, 1680]$	$\mathcal{U}(1120, 1680)$	length of borehole in m
K_w	$[9855, 12045]$	$\mathcal{U}(9855, 12045)$	hydraulic conductivity of borehole in m/yr

Note, $\mathcal{N}(\mu, \sigma)$ represents the normal distribution with expected value μ and standard deviation σ and $\mathcal{LN}(\mu, \sigma)$ is the log-normal distribution with mean μ and standard deviation σ of the logarithm. Further, $\mathcal{U}(a, b)$ denotes the continuous uniform distribution over the interval $[a, b]$.

Value

`borehole` returns the function value of borehole function at `x`.

`boreholeGrad` returns the gradient of borehole function at `x`.

Author(s)

Carmen van Meegen

References

Harper, W. V. and Gupta, S. K. (1983). Sensitivity/Uncertainty Analysis of a Borehole Scenario Comparing Latin Hypercube Sampling and Deterministic Sensitivity Approaches. BMI/ONWI-516, Office of Nuclear Waste Isolation, Battelle Memorial Institute, Columbus, OH.

Morris, M., Mitchell, T., and Ylvisaker, D. (1993). Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction. *Technometrics*, **35**(3):243–255. doi:10.1080/00401706.1993.10485320.

See Also

`gekm` for another example.

Examples

```
# List of inputs with their distributions and their respective ranges
inputs <- list("r_w" = list(dist = "norm", mean =  0.1, sd = 0.0161812, min = 0.05, max = 0.15),
"r" = list(dist = "lnorm", meanlog = 7.71, sdlog = 1.0056, min = 100, max = 50000),
"T_u" = list(dist = "unif", min = 63070, max = 115600),
"H_u" = list(dist = "unif", min = 990, max = 1110),
"T_l" = list(dist = "unif", min = 63.1, max = 116),
"H_l" = list(dist = "unif", min = 700, max = 820),
"L" = list(dist = "unif", min = 1120, max = 1680),
# for a more nonlinear, nonadditive function, see Morris et al. (1993)
"K_w" = list(dist = "unif", min = 1500, max = 15000))

# Function for Monte Carlo simulation
samples <- function(x, N = 10^5){
switch(x$dist,
"norm" = rnorm(N, x$mean, x$sd),
"lnorm" = rlnorm(N, x$meanlog, x$sdlog),
"unif" = runif(N, x$min, x$max))
}

# Uncertainty distribution of the water flow rate
set.seed(1)
X <- sapply(inputs, samples)
y <- borehole(X)
hist(y, breaks = 50, xlab = expression(paste("Water flow rate ", group("[", m^3/yr, "]"))),
main = "", freq = FALSE)
```

`branin` *Branin-Hoo Function*

Description

The Branin-Hoo function is defined by

$$f_{\text{branin}}(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

with $x_1 \in [-5, 10]$ and $x_2 \in [0, 15]$.

Usage

```
branin(x)
braninGrad(x)
```

Arguments

`x`	a numeric `vector` of length `2` or a numeric `matrix` with `n` rows and `2` columns.

Details

The gradient of the Branin-Hoo function is

$$\nabla f_{\text{branin}}(x_1, x_2) = \begin{pmatrix} 2\left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)\left(-10.2\frac{x_1}{4\pi^2} + \frac{5}{\pi}\right) - 10\left(1 - \frac{1}{8\pi}\right)\sin(x_1) \\ 2\left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right) \end{pmatrix}.$$

The Branin-Hoo function has three global minima $f_{\text{branin}}(x^\star) = 0.397887$ at $x^\star = (-\pi, 12.275)$, $x^\star = (\pi, 2.275)$ and $x^\star = (9.42478, 2.475)$.

Value

`branin` returns the function value of the Branin-Hoo function at `x`.

`braninGrad` returns the gradient of the Branin-Hoo function at `x`.

Author(s)

Carmen van Meegen

References

Branin, Jr., F. H. (1972). Widely Convergent Method of Finding Multiple Solutions of Simultaneous Nonlinear Equations. *IBM Journal of Research and Development*, **16**(5):504–522.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

Examples

```
# Contour plot of Branin-Hoo function
n.grid <- 50
x1 <- seq(-5, 10, length.out = n.grid)
x2 <- seq(0, 15, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) branin(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Branin-Hoo function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

camel3 *Three-Hump Camel Function*

Description

The three-hump camel function is defined by

$$f_{\text{camel3}}(x_1, x_2) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$$

with $x_1, x_2 \in [-5, 5]$.

Usage

```
camel3(x)
camel3Grad(x)
```

Arguments

x	a numeric vector of length 2 or a numeric matrix with n rows and 2 columns.

Details

The gradient of the three-hump camel function is

$$\nabla f_{\text{camel3}}(x_1, x_2) = \begin{pmatrix} 4x_1 - 4.2x_1^3 + x_1^5 + x_2 \\ x_1 + 2x_2 \end{pmatrix}.$$

The three-hump camel function has one global minimum $f_{\text{camel3}}(x^\star) = 0$ at $x^\star = (0, 0)$.

Value

`camel3` returns the function value of the three-hump camel function at `x`.

`camel3Grad` returns the gradient of the three-hump camel function at `x`.

Author(s)

Carmen van Meegen

References

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150-–194. doi:10.1504/IJMMNO.2013.055204.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

Examples

```
# Contour plot of three-hump camel function
n.grid <- 50
x1 <- x2 <- seq(-2, 2, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) camel3(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of three-hump camel function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

camel6 *Six-Hump Camel Function*

Description

The six-hump camel function is defined by

$$f_{\text{camel6}}(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + (-4 + 4x_2^2)x_2^2$$

with $x_1 \in [-3, 3]$ and $x_2 \in [-2, 2]$.

Usage

```
camel6(x)
camel6Grad(x)
```

Arguments

x	a numeric vector of length 2 or a numeric matrix with n rows and 2 columns.

Details

The gradient of the six-hump camel function is

$$\nabla f_{\text{camel6}}(x_1, x_2) = \begin{pmatrix} 8x_1 - 8.4x_1^3 + 2x_1^5 + x_2 \\ x_1 - 8x_2 + 16x_2^3 \end{pmatrix}.$$

The six-hump camel function has two global minima $f_{\text{camel6}}(x^\star) = -1.031628$ at $x^\star = (0.0898, -0.7126)$ and $x^\star = (-0.0898, 0.7126)$.

Value

`camel6` returns the function value of the six-hump camel function function at `x`.

`camel6Grad` returns the gradient of the six-hump camel function function at `x`.

Author(s)

Carmen van Meegen

References

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150-–194. doi:10.1504/IJMMNO.2013.055204.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

Examples

```
# Contour plot of six-hump camel function
n.grid <- 50
x1 <- seq(-2, 2, length.out = n.grid)
x2 <- seq(-1, 1, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) camel6(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of six-hump camel function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

`cigar` *Bent Cigar Function*

Description

The Bent cigar function is defined by

$$f_{\text{cigar}}(x_1, ..., x_d) = x_1^2 + 10^6 \sum_{k=2}^{d} x_k^2$$

with $x_k \in [-100, 100]$ for $k = 1, ..., d$.

Usage

```
cigar(x)
cigarGrad(x)
```

Arguments

`x`	a numeric vector of length `2` or a numeric matrix with `n` rows and `2` columns.

Details

The gradient of the bent cigar function is

$$\nabla f_{\text{cigar}}(x_1, ..., x_d) = \begin{pmatrix} 2x_1 \\ 20^6 x_2 \\ \vdots \\ 20^6 x_d \end{pmatrix}.$$

The bent cigar function has one global minimum $f_{\text{cigar}}(x^\star) = 0$ at $x^\star = (1, \ldots, 1)$.

Value

`cigar` returns the function value of the bent cigar function at `x`.

`cigarGrad` returns the gradient of the bent cigar function at `x`.

Author(s)

Carmen van Meegen

References

Plevris, V. and Solorzano, G. (2022). A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data*, **7**(4):46. doi:10.3390/data7040046.

See Also

tangents for drawing tangent lines.

Examples

```
# 1-dimensional Cigar function with tangents
curve(cigar(x), from = -5, to = 5, n = 200)
x <- seq(-4.5, 4.5, length = 5)
y <- cigar(x)
dy <- cigarGrad(x)
tangents(x, y, dy, length = 2, lwd = 2, col = "red")
points(x, y, pch = 16)

# Contour plot of Cigar function
n.grid <- 50
x1 <- x2 <- seq(-100, 100, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) cigar(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Cigar function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

derivModelMatrix *Derivatives of Model Matrix*

Description

Determine the derivatives of a model matrix.

Usage

```
derivModelMatrix(object, ...)

## Default S3 method:
derivModelMatrix(object, data, ...)
## S3 method for class 'gekm'
derivModelMatrix(object, ...)
## S3 method for class 'gekm'
model.matrix(object, ...)
```

Arguments

object	an object of an appropriate class. For the default method, a formula defining the regression functions.
data	a data.frame with named columns.
...	further arguments, yet not used.

Details

derivModelMatrix makes use of the function deriv. Accordingly, the calculation of derivatives is only possible for functions that are contained in the derivatives table of deriv.

Note, in contrast to model.matrix, factors are not supported.

Value

The derivatives of the model (or design) matrix.

As in model.matrix there is an attribute "assign".

Author(s)

Carmen van Meegen

See Also

deriv for more details on supported arithmetic operators and functions.

model.matrix for construction of a design (or model) matrix.

Examples

```
## Several examples for the derivatives of a model matrix

dat <- data.frame(x1 = seq(-2, 2, length.out = 5))

model.matrix(~ 1, dat)
derivModelMatrix(~ 1, dat)

model.matrix(~ ., dat)
derivModelMatrix(~ ., dat)

model.matrix(~ . - 1, dat)
derivModelMatrix(~ . - 1, dat)

model.matrix(~ sin(x1) + I(x1^2), dat)
derivModelMatrix(~ sin(x1) + I(x1^2), dat)

dat <- cbind(dat, x2 = seq(1, 5, length.out = 5))

model.matrix(~ 1, dat)
derivModelMatrix(~ 1, dat)
```

```
model.matrix(~ .^2, dat)
derivModelMatrix(~ .^2, dat)

model.matrix(~ log(x2), dat)
derivModelMatrix(~ log(x2), dat)

model.matrix(~ x1:x2, dat)
derivModelMatrix(~ x1:x2, dat)

model.matrix(~ I(x1^2) * I(x2^3), dat)
derivModelMatrix(~ I(x1^2) * I(x2^3), dat)

model.matrix(~ sin(x1) + cos(x2) + atan(x1 * x2), dat)
derivModelMatrix(~ sin(x1) + cos(x2) + atan(x1 * x2), dat)
```

---

gekm *Fitting (Gradient-Enhanced) Kriging Models*

Description

Estimation of a Kriging model with or without derivatives.

Usage

```
gekm(formula, data, deriv, covtype = c("matern5_2", "matern3_2", "gaussian"),
theta = NULL, tol = NULL, optimizer = c("NMKB", "L-BFGS-B"),
lower = NULL, upper = NULL, start = NULL, ncalls = 20, control = NULL,
model = TRUE, x = FALSE, y = FALSE, dx = FALSE, dy = FALSE, ...)

## S3 method for class 'gekm'
print(x, digits = 4L, ...)
```

Arguments

formula	a formula that defines the regression functions. Note that only formula expressions for which the derivations are contained in the derivatives table of deriv are supported in the gradient-enhanced Kriging model. In addition, formulas containing I also work for the gradient-enhanced Kriging model, although this function is not included in the derivatives table of deriv. See derivModelMatrix for some examples of the trend specification.
data	a data.frame with named columns of n training points of dimension d. Note, all variables contained in data are used for the construction of the correlation matrix without and with derivatives.
deriv	an optional data.frame with the derivatives, whose columns should be named like those of data. If not specified, a Kriging model without derivatives is estimated.
covtype	a character to specify the covariance structure to be used. One of matern5_2, matern3_2 or gaussian. Default is matern5_2.

theta	a numeric vector of length d for the hyperparameters (optional). If not given, hyperparameters will be estimated via maximum likelihood.
tol	a tolerance for the conditional number of the correlation matrix, see blockChol for details. Default is NULL, i.e. no regularization is applied.
optimizer	an optional character that characterizes the optimization algorithms to be used for maximum likelihood estimation. See 'Details'.
lower	an optional lower bound for the optimization of the correlation parameters.
upper	an optional upper bound for the optimization of the correlation parameters.
start	an optional vector of inital values for the optimization of the correlation parameters.
ncalls	an optional integer that defines the number of randomly selected initial values for the optimization.
control	a list of control parameters for the optimization routine. See optim or nmkb.
model	logical. Should the model frame be returned? Default is TRUE.
x	logical. Should the model matrix be returned? Default is FALSE.
y	logical. Should the response vector be returned? Default is FALSE.
dx	logical. Should the derivative of the model matrix be returned? Default is FALSE.
dy	logical. Should the derivatives of the response be returned? Default is FALSE.
digits	number of digits to be used for the print method.
...	further arguments, currently not used.

Details

Parameter estimation is performed via maximum likelihood. The optimizer argument can be used to select one of the optimization algorithms "NMBK" or "L-BFGS-B". In the case of the "L-BFGS-B", analytical gradients of the "concentrated" log likelihood are used. For one-dimensional problems, optimize is called and the algorithm selected via optimizer is ignored.

Value

gekm returns an object of class "gekm" whose underlying structure is a list containing at least the following components:

coefficients	the estimated regression coefficients.
sigma	the estimated process standard deviation.
theta	the (estimated) correlation parameters.
covtype	the name of the correlation function.
chol	(the components of) the upper triangular matrix of the Cholesky decomposition of the correlation matrix.
optimizer	the optimization algorithm.
convergence	the convergence code.
message	information from the optimizer.

`logLik`	the value of the negative "concentrated" log likelihood at the estimated parameters.
`derivatives`	`TRUE` if a gradient-enhanced Kriging model was adapted, otherwise `FALSE`.
`data`	the `data.frame` that was specified via the `data` argument.
`deriv`	if `derivatives = TRUE`, the `data.frame` with the derivatives that was specified via the `deriv` argument.
`call`	the matched call.
`terms`	the `terms` object used.
`model`	if requested (the default), the model frame used.
`x`	if requested, the model matrix.
`y`	if requested, the response vector.
`dx`	if requested, the derivatives of the model matrix.
`dx`	if requested, the vector of derivatives of the response.

Author(s)

Carmen van Meegen

References

Cressie, N. A. C. (1993). *Statistics for Spartial Data*. John Wiley & Sons. doi:10.1002/9781119115151.

Koehler, J. and Owen, A. (1996). Computer Experiments. In Ghosh, S. and Rao, C. (eds.), *Design and Analysis of Experiments*, volume 13 of *Handbook of Statistics*, pp. 261–308. Elsevier Science. doi:10.1016/S01697161(96)13011X.

Krige, D. G. (1951). A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, **52**(6):199–139.

Laurent, L., Le Riche, R., Soulier, B., and Boucard, PA. (2019). An Overview of Gradient-Enhanced Metamodels with Applications. *Archives of Computational Methods in Engineering*, **26**(1):61–106. doi:10.1007/s1183101792263.

Martin, J. D. and Simpson, T. W. (2005). Use of Kriging Models to Approximate Deterministic Computer Models. *AIAA Journal*, **43**(4):853–863. doi:10.2514/1.8650.

Morris, M., Mitchell, T., and Ylvisaker, D. (1993). Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction. *Technometrics*, **35**(3):243–255. doi:10.1080/00401706.1993.10485320.

Oakley, J. and O'Hagan, A. (2002). Bayesian Inference for the Uncertainty Distribution of Computer Model Outputs. *Biometrika*, **89**(4):769–784. doi:10.1093/biomet/89.4.769.

O'Hagan, A., Kennedy, M. C., and Oakley, J. E. (1999). Uncertainty Analysis and Other Inference Tools for Complex Computer Codes. In *Bayesian Statistics 6*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A .F. M. Smith, 503–524. Oxford University Press.

O'Hagan, A. (2006). Bayesian Analysis of Computer Code Outputs: A Tutorial. *Reliability Engineering & System Safet*, **91**(10):1290–1300. doi:10.1016/j.ress.2005.11.025.

Park, J.-S. and Beak, J. (2001). Efficient Computation of Maximum Likelihood Estimators in a Spatial Linear Model with Power Exponential Covariogram. *Computers & Geosciences*, **27**(1):1–7. doi:10.1016/S00983004(00)000169.

Ranjan, P., Haynes, R. and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data. *Technometrics*, **53**:366–378. doi:10.1198/TECH.2011.09141.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press. https://gaussianprocess.org/gpml/.

Ripley, B. D. (1981). *Spatial Statistics*. John Wiley & Sons. doi:10.1002/0471725218.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and Analysis of Computer Experiments. *Statistical Science*, **4**(4):409–423. doi:10.1214/ss/1177012413.

Santner, T. J., Williams, B. J., and Notz, W. I. (2018). *The Design and Analysis of Computer Experiments*. 2nd edition. Springer-Verlag.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer-Verlag. doi:10.1007/9781461214946.

See Also

predict.gekm for prediction at new data points based on a model of class "gekm".

simulate.gekm for simulation of process paths conditional on a model of class "gekm".

Examples

```
## 1-dimensional example: Oakley and O'Hagan (2002)

# Define test function and its gradient
f <- function(x) 5 + x + cos(x)
fGrad <- function(x) 1 - sin(x)

# Generate coordinates and calculate slopes
x <- seq(-5, 5, length = 5)
y <- f(x)
dy <- fGrad(x)
dat <- data.frame(x, y)
deri <- data.frame(x = dy)

# Fit Kriging model
km.1d <- gekm(y ~ x, data = dat, covtype = "gaussian", theta = 1)
km.1d

# Fit Gradient-Enhanced Kriging model
gekm.1d <- gekm(y ~ x, data = dat, deriv = deri, covtype = "gaussian", theta = 1)
gekm.1d


## 2-dimensional example: Morris et al. (1993)

# List of inputs with their distributions and their respective ranges
inputs <- list("r_w" = list(dist = "norm", mean =  0.1, sd = 0.0161812, min = 0.05, max = 0.15),
"r" = list(dist = "lnorm", meanlog = 7.71, sdlog = 1.0056, min = 100, max = 50000),
"T_u" = list(dist = "unif", min = 63070, max = 115600),
"H_u" = list(dist = "unif", min = 990, max = 1110),
```

```
"T_l" = list(dist = "unif", min = 63.1, max = 116),
"H_l" = list(dist = "unif", min = 700, max = 820),
"L" = list(dist = "unif", min = 1120, max = 1680),
# for a more nonlinear, nonadditive function, see Morris et al. (1993)
"K_w" = list(dist = "unif", min = 1500, max = 15000))

# Generate design
design <- data.frame("r_w" = c(0, 0.268, 1),
"r" = rep(0, 3),
"T_u" = rep(0, 3),
"H_u" = rep(0, 3),
"T_l" = rep(0, 3),
"H_l" = rep(0, 3),
"L" = rep(0, 3),
"K_w" = c(0, 1, 0.268))

# Function to transform design onto input range
transform <- function(x, data){
for(p in names(data)){
data[ , p] <- (x[[p]]$max - x[[p]]$min) * data[ , p] + x[[p]]$min
}
data
}

# Function to transform derivatives
deriv.transform <- function(x, data){
for(p in colnames(data)){
data[ , p] <- data[ , p] * (x[[p]]$max - x[[p]]$min)
}
data
}

# Generate outcome and derivatives
design.trans <- transform(inputs, design)
design$y <- borehole(design.trans)
deri.trans <- boreholeGrad(design.trans)
deri <- data.frame(deriv.transform(inputs, deri.trans))

# Design and data
cbind(design[ , c("r_w", "K_w", "y")], deri[ , c("r_w", "K_w")])

# Fit gradient-enhanced Kriging model with Gaussian correlation function
mod <- gekm(y ~ 1, data = design[ , c("r_w", "K_w", "y")],
deriv = deri[ , c("r_w", "K_w")], covtype = "gaussian")
mod

## Compare results with Morris et al. (1993):

# Estimated hyperparameters
# in Morris et al. (1993): 0.429 and 0.467
1 / (2 * mod$theta^2)
# Estimated intercept
# in Morris et al. (1993): 69.15
```

```
coef(mod)
# Estimated standard deviation
# in Morris et al. (1993): 135.47
mod$sigma
# Posterior mean and standard deviation at (0.5, 0.5)
# in Morris et al. (1993): 69.4 and 2.7
predict(mod, data.frame("r_w" = 0.5, "K_w" = 0.5))
# Posterior mean and standard deviation at (1, 1)
# in Morris et al. (1993): 230.0 and 19.2
predict(mod, data.frame("r_w" = 1, "K_w" = 1))

## Graphical comparison:

# Generate a 21 x 21 grid for prediction
n_grid <- 21
x <- seq(0, 1, length.out = n_grid)
grid <- expand.grid("r_w" = x, "K_w" = x)
pred <- predict(mod, grid)

# Compute ground truth on (transformed) grid
newdata <- data.frame("r_w" = grid[ , "r_w"],
"r" = 0, "T_u" = 0, "H_u" = 0,
"T_l" = 0, "H_l" = 0, "L" = 0,
"K_w" = grid[ , "K_w"])
newdata <- transform(inputs, newdata)
truth <- borehole(newdata)

# Contour plots of predicted and actual output
par(mfrow = c(1, 2), oma = c(3.5, 3.5, 0, 0.2), mar = c(0, 0, 1.5, 0))
contour(x, x, matrix(pred$fit, nrow = n_grid, ncol = n_grid, byrow = TRUE),
levels = c(seq(10, 50, 10), seq(100, 250, 50)),
main = "Predicted output")
points(design[ , c("K_w", "r_w")], pch = 16)
contour(x, x, matrix(truth, nrow = n_grid, ncol = n_grid, byrow = TRUE),
levels = c(seq(10, 50, 10), seq(100, 250, 50)),
yaxt = "n", main = "Ground truth")
points(design[ , c("K_w", "r_w")], pch = 16)
mtext(side = 1, outer = TRUE, line = 2.5, "Normalized hydraulic conductivity of borehole")
mtext(side = 2, outer = TRUE, line = 2.5, "Normalized radius of borehole")
```

---

griewank *Griewank Function*

Description

Griewank function is defined by

$$f_{\text{griewank}}(x_1, ..., x_d) = \sum_{k=1}^{d} \frac{x_k^2}{4000} - \prod_{k=1}^{d} \cos\left(\frac{x_k}{\sqrt{k}}\right) + 1$$

with $x_k \in [-600, 600]$ for $k = 1, ..., d$.

Usage

```
griewank(x)
griewankGrad(x)
```

Arguments

x	a numeric vector or a numeric matrix with n rows and d columns. If a vector is passed, the 1-dimensional version of the Griewank function is calculated.

Details

The gradient of Griewank function is

$$\nabla f_{\text{griewank}}(x_1, ..., x_d) = \begin{pmatrix} \frac{x_1}{2000} + \frac{1}{\sqrt{1}} \sin\left(\frac{x_1}{\sqrt{1}}\right) \prod_{k=2}^{d} \cos\left(\frac{x_k}{\sqrt{k}}\right) \\ \vdots \\ \frac{x_d}{2000} + \frac{1}{\sqrt{d}} \sin\left(\frac{x_d}{\sqrt{d}}\right) \prod_{k=1}^{d-1} \cos\left(\frac{x_k}{\sqrt{k}}\right) \end{pmatrix}.$$

Griewank function has one global minimum $f_{\text{griewank}}(x^\star) = 0$ at $x^\star = (0, \ldots, 0)$.

Value

griewank returns the function value of Griewank function at x.

griewankGrad returns the gradient of Griewank function at x.

Author(s)

Carmen van Meegen

References

Plevris, V. and Solorzano, G. (2022). A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data*, **7**(4):46. doi:10.3390/data7040046.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

See Also

tangents for drawing tangent lines.

Examples

```
# 1-dimensional Griewank function with tangents
curve(griewank(x), from = -5, to = 5, n = 200)
x <- seq(-5, 5, length = 5)
y <- griewank(x)
dy <- griewankGrad(x)
tangents(x, y, dy, length = 2, lwd = 2, col = "red")
points(x, y, pch = 16)
```

```
# Contour plot of Griewank function
n.grid <- 50
x1 <- x2 <- seq(-5, 5, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) griewank(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Griewank function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

`himmelblau` *Himmelblau's Function*

Description

Himmelblau's function is defined by

$$f_{\text{himmelblau}}(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

with $x_1, x_2 \in [-5, 5]$.

Usage

```
himmelblau(x)
himmelblauGrad(x)
```

Arguments

`x`	a numeric `vector` of length 2 or a numeric `matrix` with `n` rows and 2 columns.

Details

The gradient of Himmelblau's function is

$$\nabla f_{\text{himmelblau}}(x_1, x_2) = \begin{pmatrix} 4x_1(x_1^2 + x_2 - 11) + 2(x_1 + x_2^2 - 7) \\ 2(x_1^2 + x_2 - 11) + 4x_2(x_1 + x_2^2 - 7) \end{pmatrix}.$$

Himmelblau's function has four global minima $f_{\text{himmelblau}}(x^\star) = 0$ at $x^\star = (3, 2)$, $x^\star = (-2.805118, 3.131312)$, $x^\star = (-3.779310, -3.283186)$ and $x^\star = (3.584428, -1.848126)$.

Value

`himmelblau` returns the function value of Himmelblau's function at `x`.

`himmelblauGrad` returns the gradient of Himmelblau's function at `x`.

Author(s)

Carmen van Meegen

References

Himmelblau, D. (1972). Applied Nonlinear Programming. McGraw-Hill. ISBN 0-07-028921-2.

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150–194. doi:10.1504/IJMMNO.2013.055204.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

Examples

```
# Contour plot of Himmelblau's function
n.grid <- 50
x1 <- x2 <- seq(-5, 5, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) himmelblau(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Himmelblau's function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

predict.gekm *Predict Method for (Gradient-Enhanced) Kriging Model Fits*

Description

Predicted values and standard deviations based on a gekm model.

Usage

```
## S3 method for class 'gekm'
predict(object, newdata, ...)
```

Arguments

object	an object of class "gekm".
newdata	a data.frame containing the points where to perform predictions.
...	further arguments, currently not used.

Value

`fit`	predicted mean computed at newdata.
`sd.fit`	predicted standard deviation of predicted mean.

Author(s)

Carmen van Meegen

References

Cressie, N. A. C. (1993). *Statistics for Spartial Data*. John Wiley & Sons. doi:10.1002/9781119115151.

Koehler, J. and Owen, A. (1996). Computer Experiments. In Ghosh, S. and Rao, C. (eds.), *Design and Analysis of Experiments*, volume 13 of *Handbook of Statistics*, pp. 261–308. Elsevier Science. doi:10.1016/S01697161(96)13011X.

Krige, D. G. (1951). A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, **52**(6):199–139.

Laurent, L., Le Riche, R., Soulier, B., and Boucard, PA. (2019). An Overview of Gradient-Enhanced Metamodels with Applications. *Archives of Computational Methods in Engineering*, **26**(1):61–106. doi:10.1007/s1183101792263.

Martin, J. D. and Simpson, T. W. (2005). Use of Kriging Models to Approximate Deterministic Computer Models. *AIAA Journal*, **43**(4):853–863. doi:10.2514/1.8650.

Morris, M., Mitchell, T., and Ylvisaker, D. (1993). Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction. *Technometrics*, **35**(3):243–255. doi:10.1080/00401706.1993.10485320.

Oakley, J. and O'Hagan, A. (2002). Bayesian Inference for the Uncertainty Distribution of Computer Model Outputs. *Biometrika*, **89**(4):769–784. doi:10.1093/biomet/89.4.769.

O'Hagan, A., Kennedy, M. C., and Oakley, J. E. (1999). Uncertainty Analysis and Other Inference Tools for Complex Computer Codes. In *Bayesian Statistics 6*, Ed. J. M. Bernardo, J. O. Berger, A. P. Dawid and A .F. M. Smith, 503–524. Oxford University Press.

O'Hagan, A. (2006). Bayesian Analysis of Computer Code Outputs: A Tutorial. *Reliability Engineering & System Safet*, **91**(10):1290–1300. doi:10.1016/j.ress.2005.11.025.

Park, J.-S. and Beak, J. (2001). Efficient Computation of Maximum Likelihood Estimators in a Spatial Linear Model with Power Exponential Covariogram. *Computers & Geosciences*, **27**(1):1–7. doi:10.1016/S00983004(00)000169.

Ranjan, P., Haynes, R. and Karsten, R. (2011). A Computationally Stable Approach to Gaussian Process Interpolation of Deterministic Computer Simulation Data. *Technometrics*, **53**:366–378. doi:10.1198/TECH.2011.09141.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press. https://gaussianprocess.org/gpml/.

Ripley, B. D. (1981). *Spatial Statistics*. John Wiley & Sons. doi:10.1002/0471725218.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and Analysis of Computer Experiments. *Statistical Science*, **4**(4):409–423. doi:10.1214/ss/1177012413.

Santner, T. J., Williams, B. J., and Notz, W. I. (2018). *The Design and Analysis of Computer Experiments*. 2nd edition. Springer-Verlag.

Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer-Verlag. doi:10.1007/9781461214946.

See Also

gekm for fitting a (gradient-enhanced) Kriging model.

tangents for drawing tangent lines.

Examples

```
## 1-dimensional example: Oakley and O'Hagan (2002)

# Define test function and its gradient
f <- function(x) 5 + x + cos(x)
fGrad <- function(x) 1 - sin(x)

# Generate coordinates and calculate slopes
x <- seq(-5, 5, length = 5)
y <- f(x)
dy <- fGrad(x)
dat <- data.frame(x, y)
deri <- data.frame(x = dy)

# Fit Kriging model
km.1d <- gekm(y ~ x, data = dat, covtype = "gaussian", theta = 1)

# Fit gradient-enhanced Kriging model
gekm.1d <- gekm(y ~ x, data = dat, deriv = deri, covtype = "gaussian", theta = 1)

# Generate new data for prediction
newdat <- data.frame(x = seq(-6, 6, length = 100))

# Prediction for both models
pred.km.1d <- predict(km.1d, newdat)
pred.gekm.1d  <- predict(gekm.1d, newdat)

# Draw curves
curve(f(x), from = -6, to = 6, lwd = 2)
lines(newdat$x, pred.km.1d$fit, type = "l", col = rgb(0.5, 0.5, 0.5), lwd = 2, lty = 2)
lines(newdat$x, pred.gekm.1d$fit, type = "l", col = rgb(0, 0.5, 1), lwd = 2, lty = 2)

# Add pointswise confidence intervals
lower.km.1d <- pred.km.1d$fit - qnorm(0.975) * pred.km.1d$sd
upper.km.1d <- pred.km.1d$fit + qnorm(0.975) * pred.km.1d$sd
lower.gekm.1d <- pred.gekm.1d$fit - qnorm(0.975) * pred.gekm.1d$sd
upper.gekm.1d <- pred.gekm.1d$fit + qnorm(0.975) * pred.gekm.1d$sd

polygon(c(newdat$x, rev(newdat$x)),c(lower.km.1d, rev(upper.km.1d)),
col = rgb(0.5, 0.5, 0.5, alpha = 0.1), border = NA)
polygon(c(newdat$x, rev(newdat$x)),c(lower.gekm.1d, rev(upper.gekm.1d)),
col = rgb(0, 0.5, 1, alpha = 0.1), border = NA)
```

```
# Add tangent lines and observations
tangents(x, y, dy, lwd = 2, col = "red")
points(x, y, pch = 16)

# Add legend
legend("topleft", lty = c(1, 2, 2), col = c("black", rgb(0.5, 0.5, 0.5), rgb(0, 0.5, 1)),
legend = c("Test function", "Kriging", "GEK"), lwd = 2, bty = "n")


## 2-dimensional example: Branin-Hoo function

# Generate a grid for training
n <- 5
x1 <- seq(-5, 10, length = n)
x2 <- seq(0, 15, length = n)
x <- expand.grid(x1 = x1, x2 = x2)
y <- branin(x)
dy <- braninGrad(x)
dat <- data.frame(x, y)
deri <- data.frame(dy)

# Fit Kriging model
km.2d <- gekm(y ~ ., data = dat)
km.2d

# Fit gradient-enhanced Kriging model
gekm.2d <- gekm(y ~ ., data = dat, deriv = deri)
gekm.2d

# Generate new data for prediction
n.grid <- 50
x1.grid <- seq(-5, 10, length = n.grid)
x2.grid <- seq(0, 15, length = n.grid)
newdat <- expand.grid(x1 = x1.grid, x2 = x2.grid)

# Prediction for both models and actual outcome
pred.km.2d <- predict(km.2d, newdat)
pred.gekm.2d  <- predict(gekm.2d, newdat)
truth <- outer(x1.grid, x2.grid, function(x1, x2) branin(cbind(x1, x2)))

# Contour plots of predicted and actual output
par(mfrow = c(1, 3), oma = c(3.5, 3.5, 0, 0.2), mar = c(0, 0, 1.5, 0))
contour(x1.grid, x2.grid, truth, nlevels = 30,
levels = seq(0, 300, 10), main = "Branin-Hoo")
points(x, pch = 16)
contour(x1.grid, x2.grid, matrix(pred.km.2d$fit, nrow = n.grid, ncol = n.grid),
levels = seq(0, 300, 10), main = "Kriging", yaxt = "n")
points(x, pch = 16)
contour(x1.grid, x2.grid, matrix(pred.gekm.2d$fit, nrow = n.grid, ncol = n.grid),
levels = seq(0, 300, 10), main = "GEK", yaxt = "n")
points(x, pch = 16)
mtext(side = 1, outer = TRUE, line = 2.5, expression(x[1]))
mtext(side = 2, outer = TRUE, line = 2, expression(x[2]))
```

```
# Contour plots of predicted variance
par(mfrow = c(1, 2), oma = c(3.5, 3.5, 0, 0.2), mar = c(0, 0, 1.5, 0))
contour(x1.grid, x2.grid, matrix(pred.km.2d$sd.fit^2, nrow = n.grid, ncol = n.grid),
main = "Kriging variance")
points(x, pch = 16)
contour(x1.grid, x2.grid, matrix(pred.gekm.2d$sd.fit^2, nrow = n.grid, ncol = n.grid),
main = "GEK variance", yaxt = "n")
points(x, pch = 16)
mtext(side = 1, outer = TRUE, line = 2.5, expression(x[1]))
mtext(side = 2, outer = TRUE, line = 2, expression(x[2]))
```

`qing` *Qing Function*

Description

Qing function is defined by

$$f_{\text{qing}}(x_1, ..., x_d) = \sum_{k=1}^{d} (x_i^2 - i)^2$$

with $x_k \in [-500, 500]$ for $k = 1, ..., d$.

Usage

```
qing(x)
qingGrad(x)
```

Arguments

`x`	a numeric `vector` or a numeric `matrix` with `n` rows and `d` columns. If a `vector` is passed, the 1-dimensional version of the Rastrigin function is calculated.

Details

The gradient of Qing function is

$$\nabla f_{\text{qing}}(x_1, ..., x_d) = \begin{pmatrix} 4x_1(x_1^2 - 1) \\ \vdots \\ 4x_d(x_d^2 - d) \end{pmatrix}.$$

Qing function has 2^d global minimum $f_{\text{qing}}(x^\star) = 0$ at $x^\star = (\pm\sqrt{1}, \ldots, \pm\sqrt{d})$.

Value

`qing` returns the function value of Qing function at `x`.

`qingGrad` returns the gradient of Qing function at `x`.

Author(s)

Carmen van Meegen

References

Qing, A. (2006). Dynamic Differential Evolution Strategy and Applications in Electromagnetic Inverse Scattering Problems. *IEEE Transactions on Geoscience and Remote Sensing*, **44**(1):116-–125. doi:10.1109/TGRS.2005.859347.

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150-–194. doi:10.1504/IJMMNO.2013.055204.

Examples

```
# 1-dimensional Qing function with tangents
curve(qing(x), from = -1.7, to = 1.7)
x <- seq(-1.5, 1.5, length = 5)
y <- qing(x)
dy <- qingGrad(x)
tangents(x, y, dy, length = 1, lwd = 2, col = "red")
points(x, y, pch = 16)

# Contour plot of Qing function
n.grid <- 50
x1 <- seq(-2, 2, length.out = n.grid)
x2 <- seq(-2, 2, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) qing(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Qing function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

rastrigin *Rastrigin Function*

Description

Rastrigin function is defined by

$$f_{\text{rastrigin}}(x_1, ..., x_d) = 10d + \sum_{k=1}^{d} (x_k^2 - 10\cos(2\pi x_k))$$

with $x_k \in [-5.12, 5.12]$ for $k = 1, ..., d$.

Usage

```
rastrigin(x)
rastriginGrad(x)
```

Arguments

x	a numeric vector or a numeric matrix with n rows and d columns. If a vector is passed, the 1-dimensional version of the Rastrigin function is calculated.

Details

The gradient of Rastrigin function is

$$\nabla f_{\text{rastrigin}}(x_1, ..., x_d) = \begin{pmatrix} 2x_1 + 20\sin(2\pi x_1) \\ \vdots \\ 2x_d + 20\sin(2\pi x_d)) \end{pmatrix}.$$

Rastrigin function has one global minimum $f_{\text{rastrigin}}(x^\star) = 0$ at $x^\star = (0, \ldots, 0)$.

Value

`rastrigin` returns the function value of Rastrigin function at `x`.

`rastriginGrad` returns the gradient of Rastrigin function at `x`.

Author(s)

Carmen van Meegen

References

Plevris, V. and Solorzano, G. (2022). A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data*, **7**(4):46. doi:10.3390/data7040046.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

See Also

`tangents` for drawing tangent lines.

Examples

```
# 1-dimensional Rastrigin function with tangents
curve(rastrigin(x), from = -5, to = 5, n = 200)
x <- seq(-4.5, 4.5, length = 5)
y <- rastrigin(x)
dy <- rastriginGrad(x)
tangents(x, y, dy, length = 2, lwd = 2, col = "red")
points(x, y, pch = 16)
```

```
# Contour plot of Rastrigin function
n.grid <- 100
x1 <- x2 <- seq(-5.12, 5.12, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) rastrigin(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Rastrigin function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

schwefel *Schwefel Function*

Description

The Schwefel function is defined by

$$f_{\text{schwefel}}(x_1, ..., x_d) = 418.9829d - \sum_{k=1}^{d} x_k \sin\left(\sqrt{|x_k|}\right)$$

with $x_k \in [-500, 500]$ for $k = 1, ..., d$.

Usage

```
schwefel(x)
schwefelGrad(x)
```

Arguments

x	a numeric vector of length d or a numeric matrix with n rows and d columns.

Details

The gradient of the Schwefel function is

$$\nabla f_{\text{schwefel}}(x_1, \ldots, x_d) = \begin{pmatrix} -\sin\left(\sqrt{|x_1|}\right) - \frac{x_1^2 \cos\left(\sqrt{|x_1|}\right)}{2|x_1|^{\frac{3}{2}}} \\ \vdots \\ -\sin\left(\sqrt{|x_d|}\right) - \frac{x_d^2 \cos\left(\sqrt{|x_d|}\right)}{2|x_d|^{\frac{3}{2}}} \end{pmatrix}.$$

The Schwefel function has one global minimum $f_{\text{schwefel}}(x^\star) = 0$ at $x^\star = (420.968746, \ldots, 420.968746)$.

Value

`schwefel` returns the function value of the Schwefel function at `x`.

`schwefelGrad` returns the gradient of the Schwefel function at `x`.

Author(s)

Carmen van Meegen

References

Plevris, V. and Solorzano, G. (2022). A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data*, **7**(4):46. doi:10.3390/data7040046.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

See Also

`tangents` for drawing tangent lines.

Examples

```
# 1-dimensional Schwefel function with tangents
curve(schwefel(x), from = -500, to = 500, n = 500)
x <- seq(-450, 450, length = 5)
y <- schwefel(x)
dy <- schwefelGrad(x)
tangents(x, y, dy, length = 200, lwd = 2, col = "red")
points(x, y, pch = 16)

# Contour plot of Schwefel function
n.grid <- 75
x1 <- x2 <- seq(-500, 500, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) schwefel(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Schwefel function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

`short` *Short Column Function*

Description

The short column function is defined by

$$f_{\text{short}}(x) = 1 - \frac{4M}{bh^2Y} - \frac{P^2}{b^2h^2Y^2}$$

with $x = (Y, M, P)$.

Usage

```
short(x, b = 5, h = 15)
shortGrad(x, b = 5, h = 15)
```

Arguments

`x`	a numeric `vector` of length `3` or a numeric `matrix` with `n` rows and `3` columns.
`b`	width of the cross-section in mm of the short column. Default is `5`.
`h`	depth of the cross-section in mm of the short column. Default is `15`.

Details

The short column function describes the limite state function of a short column with uncertain material properties and loads.

Input	Distribution	Mean	Standard deviation	Description
Y	$\mathcal{LN}$	5	0.5	yield stress in MPa
M	$\mathcal{N}$	2000	400	bending moment in MNm
P	$\mathcal{N}$	500	100	axial force in MPa

The bending moment and the axial force are correlated with $\mathrm{Cor}(M, P) = 0.5$. Note, $\mathcal{N}$ represents the normal distribution and $\mathcal{LN}$ is the log-normal distribution.

Value

`short` returns the function value of short column function at `x`.

`shortGrad` returns the gradient of short column function at `x`.

Author(s)

Carmen van Meegen

References

Kuschel, N. and Rackwitz, R. (1997). Two Basic Problems in Reliability-Based Structural Optimization. *Mathematical Methods of Operations Research*, **46**(3):309–333.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

`simulate.gekm` *Simulates Conditional Process Paths*

Description

Simulates process paths conditional on a fitted `gekm` model.

Usage

```
## S3 method for class 'gekm'
simulate(object, nsim = 1, seed = NULL, newdata = NULL, tol = NULL, ...)
```

Arguments

`object`	an object of class `"gekm"`.
`nsim`	number of simulated process paths. Default is `1`.
`seed`	argument is not supported.
`newdata`	a `data.frame` containing the points where the process in `object` should be evaluated.
`tol`	a tolerance for the conditional number of the conditional correlation matrix of `newdata`, see `blockChol` for details. Default is `NULL`, i.e. no regularization is applied.
`...`	further arguments, not used.

Value

`val`	a `matrix` with `nrow(newdata)` rows and `nsim` columns of simulated response values at the points of `newdata`. Each column represents one conditional simulated process path.

Author(s)

Carmen van Meegen

References

Cressie, N. A. C. (1993). *Statistics for Spartial Data*. John Wiley & Sons. doi:10.1002/9781119115151.

Ripley, B. D. (1981). *Spatial Statistics*. John Wiley & Sons. doi:10.1002/0471725218.

See Also

gekm for fitting a (gradient-enhanced) Kriging model.

predict.gekm for prediction at new data points based on a model of class "gekm".

Examples

```
## 1-dimensional example

# Define test function and its gradient from Oakley and O'Hagan (2002)
f <- function(x) 5 + x + cos(x)
fGrad <- function(x) 1 - sin(x)

# Generate coordinates and calculate slopes
x <- seq(-5, 5, length = 5)
y <- f(x)
dy <- fGrad(x)
dat <- data.frame(x, y)
deri <- data.frame(x = dy)

# Fit Kriging model
km.1d <- gekm(y ~ x, data = dat, covtype = "gaussian", theta = 1)

# Fit Gradient-Enhanced Kriging model
gekm.1d <- gekm(y ~ x, data = dat, deriv = deri, covtype = "gaussian", theta = 1)

# Generate new data for prediction and simulation
newdat <- data.frame(x = seq(-6, 6, length = 600))

# Prediction for both models
pred.km.1d <- predict(km.1d, newdat)
pred.gekm.1d  <- predict(gekm.1d, newdat)

# Simulate process path conditional on fitted models
set.seed(1)
n <- 50
sim.km.1d <- simulate(km.1d, nsim = n, newdata = newdat, tol = 35)
sim.gekm.1d <- simulate(gekm.1d, nsim = n, newdata = newdat, tol = 35)

par(mfrow = c(1, 2), oma = c(3.5, 3.5, 0, 0.2), mar = c(0, 0, 1.5, 0))
matplot(newdat$x, sim.km.1d, type = "l", lty = 1, col = 2:8, lwd = 1,
ylim = c(-1, 12), main = "Kriging")
lines(newdat$x, pred.km.1d$fit + qnorm(0.975) * pred.km.1d$sd, lwd = 1.5)
lines(newdat$x, pred.km.1d$fit - qnorm(0.975) * pred.km.1d$sd, lwd = 1.5)
points(x, y, pch = 16, cex = 1)

matplot(newdat$x, sim.gekm.1d, type = "l", lty = 1, col = 2:8,
lwd = 1, ylim = c(-1, 12), main = "GEK", yaxt = "n")
lines(newdat$x, pred.gekm.1d$fit + qnorm(0.975) * pred.gekm.1d$sd, lwd = 1.5)
lines(newdat$x, pred.gekm.1d$fit - qnorm(0.975) * pred.gekm.1d$sd, lwd = 1.5)
points(x, y, pch = 16, cex = 1)

mtext(side = 1, outer = TRUE, line = 2.5, "x")
```

```
mtext(side = 2, outer = TRUE, line = 2.5, "f(x)")
```

---

sphere *Sphere Function*

Description

The sphere function is defined by

$$f_{\text{sphere}}(x_1, ..., x_d) = \sum_{k=1}^{d} x_k^2$$

with $x_k \in [-5.12, 5.12]$ for $k = 1, ..., d$.

Usage

```
sphere(x)
sphereGrad(x)
```

Arguments

x	a numeric vector or a numeric matrix with n rows and d columns. If a vector is passed, the 1-dimensional version of the sphere function is calculated.

Details

The gradient of the sphere function is

$$\nabla f_{\text{sphere}}(x_1, \ldots, x_d) = \begin{pmatrix} 2x_1 \\ \vdots \\ 2x_d \end{pmatrix}.$$

The sphere function has one global minimum $f_{\text{sphere}}(x^\star) = 0$ at $x^\star = (0, \ldots, 0)$.

Value

sphere returns the function value of the sphere function at x.

sphereGrad returns the gradient of the sphere function at x.

Author(s)

Carmen van Meegen

References

Plevris, V. and Solorzano, G. (2022). A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data*, **7**(4):46. doi:10.3390/data7040046.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

See Also

tangents for drawing tangent lines.

Examples

```
# 1-dimensional sphere function with tangents
curve(sphere(x), from = -5, to = 5)
x <- seq(-4.5, 4.5, length = 5)
y <- sphere(x)
dy <- sphereGrad(x)
tangents(x, y, dy, length = 2, lwd = 2, col = "red")
points(x, y, pch = 16)

# Contour plot of sphere function
n.grid <- 50
x1 <- x2 <- seq(-5.12, 5.12, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) sphere(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of sphere function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

---

steel *Steel Column Function*

Description

The steel column function is defined by

$$f_{\text{steel}}(x) = F_S - P\left(\frac{1}{2BD} + \frac{F_0 E_b}{BDH(E_b - P)}\right),$$

with $P = P_1 + P_2 + P_3$, $E_b = \frac{\pi^2 EBDH^2}{2L^2}$ and $x = (F_S, P_1, P_2, P_3, B, D, H, F_0, E)$.

Usage

```
steel(x, L = 7500)
steelGrad(x, L = 7500)
```

Arguments

x	a numeric vector of length 9 or a numeric matrix with n rows and 9 columns.
L	length in mm of the steel column. Default is 7500.

Details

The steel column function describes the limite state function of a steel column with uncertain parameters.

Input	Distribution	Mean	Standard Deviation	Description
F_S	$\mathcal{LN}$	400	35	yield stress in MPa
P_1	$\mathcal{N}$	500000	50000	dead weight load in N
P_2	$\mathcal{G}$	600000	90000	variable load in N
P_3	$\mathcal{G}$	600000	90000	variable load in N
B	$\mathcal{LN}$	b	3	flange breadth in mm
D	$\mathcal{LN}$	t	2	flange thickness in mm
H	$\mathcal{LN}$	h	5	profile height in mm
F_0	$\mathcal{N}$	30	10	initial deflection in mm
E	$\mathcal{W}$	210000	4200	Young's modulus in MPa

Here, $\mathcal{N}$ is the normal distribution and $\mathcal{LN}$ is the log-normal distribution. Further, $\mathcal{G}$ represents the Gumbel distribution and $\mathcal{W}$ denotes the Weibull distribution.

Value

`steel` returns the function value of steel column function at `x`.

`steelGrad` returns the gradient of steel column function at `x`.

Author(s)

Carmen van Meegen

References

Kuschel, N. and Rackwitz, R. (1997). Two Basic Problems in Reliability-Based Structural Optimization. *Mathematical Methods of Operations Research*, **46**(3):309–333.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

`styblinski` *Styblinski-Tang Function*

Description

Styblinski-Tang function is defined by

$$f_{\text{styblinski}}(x_1, ..., x_d) = \frac{1}{2}\sum_{k=1}^{d}(x_k^4 - 16x_k^2 + 5x_k)$$

with $x_k \in [-5, 5]$ for $k = 1, ..., d$.

Usage

```
styblinski(x)
styblinskiGrad(x)
```

Arguments

x	a numeric vector or a numeric matrix with n rows and d columns. If a vector is passed, the 1-dimensional version of the Rastrigin function is calculated.

Details

The gradient of Styblinski-Tang function is

$$\nabla f_{\text{styblinski}}(x_1, ..., x_d) = \begin{pmatrix} 2x_1^3 - 16x_1 + 2.5 \\ \vdots \\ 2x_d^3 - 16x_d + 2.5 \end{pmatrix}.$$

Styblinski-Tang function has one global minimum $f_{\text{styblinski}}(x^\star) = -39.16599d$ at $x^\star = (-2.903534, \ldots, -2.903534)$.

Value

`styblinski` returns the function value of Styblinski-Tang function at `x`.

`styblinskiGrad` returns the gradient of Styblinski-Tang function at `x`.

Author(s)

Carmen van Meegen

References

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150-–194. doi:10.1504/IJMMNO.2013.055204.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

See Also

`tangents` for drawing tangent lines.

Examples

```
# 1-dimensional Styblinski-Tang function with tangents
curve(styblinski(x), from = -5, to = 5)
x <- seq(-4.5, 4.5, length = 5)
y <- styblinski(x)
dy <- styblinskiGrad(x)
tangents(x, y, dy, length = 2, lwd = 2, col = "red")
points(x, y, pch = 16)
```

```
# Contour plot of Styblinski-Tang function
n.grid <- 50
x1 <- seq(-5, 5, length.out = n.grid)
x2 <- seq(-5, 5, length.out = n.grid)
y <- outer(x1, x2, function(x1, x2) styblinski(cbind(x1, x2)))
contour(x1, x2, y, xaxs = "i", yaxs = "i", nlevels = 25, xlab = "x1", ylab = "x2")

# Perspective plot of Styblinski-Tang function
col.pal <- colorRampPalette(c("#00007F", "blue", "#007FFF", "cyan", "#7FFF7F", "yellow",
"#FF7F00", "red", "#7F0000"))
colors <- col.pal(100)
y.facet.center <- (y[-1, -1] + y[-1, -n.grid] + y[-n.grid, -1] + y[-n.grid, -n.grid])/4
y.facet.range <- cut(y.facet.center, 100)
persp(x1, x2, y, phi = 30, theta = -315, expand = 0.75, ticktype = "detailed",
col = colors[y.facet.range])
```

`sulfur` *Sulfur Model Function*

Description

The sulfur function is defined by

$$f_{\text{sulfur}}(x) = -\frac{1}{2} S_0^2 (1 - A_c) T^2 (1 - R_s)^2 \bar{\beta} \Psi_e f_{\Psi_e} \frac{3QYL}{A}$$

with $x = (Q, Y, L, \Psi_e, \bar{\beta}, f_{\Psi_e}, T, 1 - A_c, 1 - R_s)$.

Usage

```
sulfur(x, S_0 = 1366, A = 5.1e+14)
sulfurGrad(x, S_0 = 1366, A = 5.1e+14)
```

Arguments

`x`	a numeric `vector` of length `9` or a numeric `matrix` with `n` rows and `9` columns.
`S_0`	solar constant in $\mathrm{W/m^2}$. Default is `1366`.
`A`	surface area of the earth in $\mathrm{m^2}$. Default is `5.1e+14`.

Details

The sulfur model function calculates the direct radiative forcing by sulfate aerosols $[\mathrm{W/m^2}]$.

Input	Central value	Uncertainty factor	Description
Q	71	1.15	source strength of anthropogenic sulfur in $\mathrm{Tg/yr}$
Y	0.5	1.5	fraction of SO_2 oxidized to $SO_4^=$
L	5.5	1.5	average lifetime of atmospheric $SO_4^=$ in days

Ψ_e	5	1.4	aerosol mass scattering efficiency in m^2/g
$\bar{\beta}$	0.3	1.3	fraction of light scattering into upward hemisphere
f_{Ψ_e}	1.7	1.2	fractional increase in aerosol scattering efficiency due to hygroscopic growth
T	0.76	1.2	atmospheric transmittance above aerosol layer
$1 - A_c$	0.39	1.1	fraction of earth not covered by cloud
$1 - R_s$	0.85	1.1	surface coalbedo

The inputs are all log-normally distributed.

Value

`sulfur` returns the function value of sulfur function at `x`.

`sulfurGrad` returns the gradient of sulfur function at `x`.

Author(s)

Carmen van Meegen

References

Charlson, R. J., Schwartz, S. E., Hales, J. M., Cess, R. D., Coakley, Jr., J. A., Hansen, J. E., and Hoffman, D. J. (1992). Climate Forcing by Anthropogenic Aerosols. *Science*, **255**:423–430. doi:10.1126/science.255.5043.423.

Penner, J. E., Charlson, R. J., Hales, J. M., Laulainen, N. S., Leifer, R., Novakov, T., Ogren, J., Radke, L. F., Schwartz, S. E., and Travis, L. (1994). Quantifying and Minimizing Uncertainty of Climate Forcing by Anthropogenic Aerosols. *Bulletin of the American Meteorological Society*, **75**(3):375–400. doi:10.1175/15200477(1994)075<0375:QAMUOC>2.0.CO;2.

Tatang, M. A., Pan, W., Prinn, R. G., and McRae, G. J. (1997). An Efficient Method for Parametric Uncertainty Analysis of Numerical Geophysical Model. *Journal of Geophysical Research Atmospheres*, **102**(18):21925–21932. doi:10.1029/97JD01654.

`tangents` *Add Tangent Lines to a Plot*

Description

Draw tangent lines to an existing plot.

Usage

```
tangents(x, y, slope, length = 1, ...)
```

Arguments

x, y	coordinate vectors of points x and function values y.
slope	vector of slopes at the points x.
length	desired length of tangent lines, see 'Details'.
...	further graphical parameters to be passed to segments.

Details

The length of the tangent lines is scaled according to the current aspect ratio of the existing plot.

Author(s)

Carmen van Meegen

See Also

segments for drawing line segments between pairs of points.

Examples

```
# Define test function and its gradient from Oakley and O'Hagan (2002)
f <- function(x) 5 + x + cos(x)
fGrad <- function(x) 1 - sin(x)

# Generate coordinates and calculate slopes
x <- seq(-5, 5, length = 5)
y <- f(x)
dy <- fGrad(x)

# Draw curve and tangent lines
curve(f(x), from = -6, to = 6)
tangents(x, y, dy, length = 2, lwd = 2, col = 2:6)
points(x, y, pch = 16)
```

---

testfunctions *Testfunctions in* **gek**

Description

Overview of testfunctions available in **gek**.

2-dimensional testfunctions for optimization

- branin: Branin-Hoo function
- camel3: Three-hump camel function
- camel6: Six-hump camel function
- himmelblau: Himmelblaus's function

Multi-dimensional testfunctions for optimization

- banana: Rosenbrock's Banana function
- cigar: Bent Cigar function
- griewank: Griewank function
- qing: Qing function
- rastrigin: Rastrigin function
- schwefel: Schwefel function
- sphere: Sphere function
- styblinski: Styblinski-Tang function

Testfunctions for uncertainty quantification

- borehole: Borehole function
- steel: Steel column function
- short: Short column function
- sulfur: Sulfur model function

Author(s)

Carmen van Meegen

References

Branin, Jr., F. H. (1972). Widely Convergent Method of Finding Multiple Solutions of Simultaneous Nonlinear Equations. *IBM Journal of Research and Development*, **16**(5):504–522.

Jamil, M. and Yang, X.-S. (2013). A Literature Survey of Benchmark Functions for Global Optimization Problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, **4**(2):150-–194. doi:10.1504/IJMMNO.2013.055204.

Harper, W. V. and Gupta, S. K. (1983). Sensitivity/Uncertainty Analysis of a Borehole Scenario Comparing Latin Hypercube Sampling and Deterministic Sensitivity Approaches. BMI/ONWI-516, Office of Nuclear Waste Isolation, Battelle Memorial Institute, Columbus, OH.

Himmelblau, D. (1972). Applied Nonlinear Programming. McGraw-Hill. ISBN 0-07-028921-2.

Kuschel, N. and Rackwitz, R. (1997). Two Basic Problems in Reliability-Based Structural Optimization. *Mathematical Methods of Operations Research*, **46**(3):309–333.

Morris, M., Mitchell, T., and Ylvisaker, D. (1993). Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction. *Technometrics*, **35**(3):243–255. doi:10.1080/00401706.1993.10485320.

Plevris, V. and Solorzano, G. (2022). A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking. *Data*, **7**(4):46. doi:10.3390/data7040046.

Rosenbrock, H. H. (1960). An Automatic Method for Finding the Greatest or least Value of a Function. *The Computer Journal*, **3**(3):175–184. doi:10.1093/comjnl/3.3.175.

Surjanovic, S. and Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. https://www.sfu.ca/~ssurjano/ (retrieved January 19, 2024).

See Also

banana, bananaGrad, borehole, boreholeGrad, branin, braninGrad, camel3, camel3Grad, camel6, camel6Grad, cigar, cigarGrad, griewank, griewankGrad, himmelblau, himmelblauGrad, qing, qingGrad, rastrigin, rastriginGrad, schwefel, schwefelGrad, short, shortGrad, sphere, sphereGrad, steel, steelGrad, styblinski, styblinskiGrad, sulfur, sulfurGrad

Index