Package 'TestFunctions'

May 9, 2017

Type Package

Title Test Functions for Simulation Experiments and Evaluating Optimization and Emulation Algorithms

Version 0.2.0

Author Collin Erickson

Maintainer Collin Erickson <collinberickson@gmail.com>

Description Test functions are often used to test computer code. They are used in optimization to test algorithms and in metamodeling to evaluate model predictions. This package provides test functions that can be used for any purpose. Some functions are taken from <https://www.sfu.ca/~ssurjano>, but their R code is not used.

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, numDeriv, ContourFunctions

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-09 21:25:06 UTC

R topics documented:

`add_linear_terms` *add_linear_terms: Add linear terms to another function. Allows you to easily change an existing function to include linear terms.*

Description

add_linear_terms: Add linear terms to another function. Allows you to easily change an existing function to include linear terms.

Usage

```
add_linear_terms(func, coeffs)
```

Arguments

`func`	Function to add linear terms to
`coeffs`	Linear coefficients, should have same length as function has dimensions

Value

Function with added linear terms

Examples

```
banana(c(.1,.2))
add_linear_terms(banana, coeffs=c(10,1000))(c(.1,.2))
```

`add_noise` *add_noise: Adds noise to any function*

Description

add_noise: Adds noise to any function

Usage

```
add_noise(func, noise = 0, noise_type = "Gauss")
```

Arguments

`func`	Function to add noise to.
`noise`	Standard deviation of Gaussian noise
`noise_type`	Type of noise, only option now is "Gauss" for Gaussian noise.

Value

A function that has noise

Examples

```
tf <- add_noise(function(x)sin(2*x*pi));curve(tf)
tf <- add_noise(function(x)sin(2*x*pi), noise=.1);curve(tf)
```

`add_null_dims` *add_null_dims: Add null dimensions to another function. Allows you to pass in input data with any number of dimensions and it will only keep the first nactive.*

Description

add_null_dims: Add null dimensions to another function. Allows you to pass in input data with any number of dimensions and it will only keep the first nactive.

Usage

```
add_null_dims(func, nactive)
```

Arguments

`func`	Function to add null dimensions to
`nactive`	Number of active dimensions in func

Value

Function that can take any dimensional input

Examples

```
banana(c(.1,.2))
# banana(c(.1,.2,.4,.5,.6,.7,.8)) # gives warning
add_null_dims(banana, nact=2)(c(.1,.2,.4,.5,.6,.7,.8))
```

`add_zoom` *add_zoom: Zoom in on region of another function. Allows you to easily change an existing function so that [0,1]^n refers to a subregion of the original function*

Description

add_zoom: Zoom in on region of another function. Allows you to easily change an existing function so that [0,1]^n refers to a subregion of the original function

Usage

```
add_zoom(func, scale_low, scale_high)
```

Arguments

func	Function to add linear terms to
scale_low	Vector of low end of scale values for each dimension
scale_high	Vector of high end of scale values for each dimension

Value

Function with added linear terms

Examples

```
banana(c(.5,.85))
add_zoom(banana, c(0,.5), c(1,1))(c(.5,.7))
add_zoom(banana, c(.2,.5), c(.8,1))(matrix(c(.5,.7),ncol=2))
ContourFunctions::cf(banana)
ContourFunctions::cf(add_zoom(banana, c(0,.5), c(1,1)))
ContourFunctions::cf(add_zoom(banana, c(.2,.5), c(.8,1)))
```

---

branin *Test function.*

Description

branin: A function. 2 dimensional function.

borehole: A function estimating water flow through a borehole. 8 dimensional function.

franke: A function. 2 dimensional function.

zhou1998: A function. 2 dimensional function.

currin1991: A function. 2 dimensional function.

lim2002: Some function? 2 dimensional function.

banana: A banana shaped function. 2 dimensional function.

gaussian1: A Gaussian function centered at 0.5. Any dimensional function.

sinumoid: A sinusoid added to a sigmoid function. 2 dimensional function.

waterfall: A sinusoid added to a sigmoid function. 2 dimensional function.

sqrtsin: A square root of a sine function. Any dimensional function.

powsin: A sine function raised to a power keeping its original sign. Any dimensional function.

OTL_Circuit: OTL Circuit. 6 dimensional function.

GoldsteinPrice: Goldstein-Price function. Exponential scale, you might want to use Goldstein-PriceLog instead 2 dimensional function.

GoldsteinPriceLog: Goldstein-Price function on a log scale. 2 dimensional function.

ackley: Ackley function. 2 dimensional function.

piston: Piston simulation function. 7 dimensional function.

wingweight: Wing weight function. 10 dimensional function.

welch: Welch et al (1992) function. 20 dimensional function.

robotarm: Robot arm function. 8 dimensional function.

RoosArnold: Roos & Arnold (1963) function. d dimensional function.

Gfunction: G-function d dimensional function.

beale: Beale function 2 dimensional function.

easom: Easom function 2 dimensional function.

griewank: Griewank function n dimensional function.

hump: Hump function 2 dimensional function.

levy: Levy function n dimensional function.

michalewicz: Michalewicz function n dimensional function.

rastrigin: Rastrigin function n dimensional function.

moon_high: Moon (2010) high-dimensional function for screening 20 dimensional function.

linkletter_nosignal: Linkletter (2006) no signal function, just returns zero d dimensional function.

Morris: Morris function 20 dimensional function.

detpep8d: detpep8d function 8 dimensional function.

hartmann: hartmann function 6 dimensional function.

quad_peaks: quad_peaks function 2 dimensional function.

quad_peaks_slant: quad_peaks_slant function 2 dimensional function.

General function for evaluating a test function

Usage

```
branin(x, scale_it = T, scale_low = c(-5, 0), scale_high = c(10, 15),
  noise = 0)

borehole(x, scale_it = T, scale_low = c(0.05, 100, 63070, 990, 63.1, 700,
  1120, 9855), scale_high = c(0.15, 50000, 115600, 1110, 116, 820, 1680,
  12045), noise = 0)

franke(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

zhou1998(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

currin1991(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

lim2002(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

banana(x, scale_it = T, scale_low = c(-20, -10), scale_high = c(20, 5),
```

```
  noise = 0)

gaussian1(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

sinumoid(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

waterfall(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0)

sqrtsin(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0, freq = 2 * pi)

powsin(x, scale_it = F, scale_low = c(0, 0), scale_high = c(1, 1),
  noise = 0, freq = 2 * pi, pow = 0.7)

OTL_Circuit(x, scale_it = T, scale_low = c(50, 25, 0.5, 1.2, 0.25, 50),
  scale_high = c(150, 70, 3, 2.5, 1.2, 300), noise = 0)

GoldsteinPrice(x, scale_it = T, scale_low = c(-2, -2), scale_high = c(2,
  2), noise = 0)

GoldsteinPriceLog(x, scale_it = T, scale_low = c(-2, -2),
  scale_high = c(2, 2), noise = 0)

ackley(x, scale_it = T, scale_low = -32.768, scale_high = 32.768,
  noise = 0, a = 20, b = 0.2, c = 2 * pi)

piston(x, scale_it = T, scale_low = c(30, 0.005, 0.002, 1000, 90000, 290,
  340), scale_high = c(60, 0.02, 0.01, 5000, 110000, 296, 360), noise = 0)

wingweight(x, scale_it = T, scale_low = c(150, 220, 6, -10, 16, 0.5, 0.08,
  2.5, 1700, 0.025), scale_high = c(200, 300, 10, 10, 45, 1, 0.18, 6, 2500,
  0.08), noise = 0)

welch(x, scale_it = T, scale_low = c(-0.5), scale_high = c(0.5),
  noise = 0)

robotarm(x, scale_it = T, scale_low = rep(0, 8), scale_high = c(rep(2 *
  pi, 4), rep(1, 4)), noise = 0)

RoosArnold(x, scale_it = F, scale_low = 0, scale_high = 1, noise = 0)

Gfunction(x, scale_it = F, scale_low = 0, scale_high = 1, noise = 0,
  ...)

beale(x, scale_it = T, scale_low = -4.5, scale_high = 4.5, noise = 0,
```

```
  ...)

easom(x, scale_it = T, scale_low = -4.5, scale_high = 4.5, noise = 0,
  ...)

griewank(x, scale_it = T, scale_low = -600, scale_high = 600, noise = 0,
  ...)

hump(x, scale_it = T, scale_low = -5, scale_high = 5, noise = 0, ...)

levy(x, scale_it = T, scale_low = -10, scale_high = 10, noise = 0, ...)

michalewicz(x, scale_it = T, scale_low = 0, scale_high = pi, noise = 0,
  ...)

rastrigin(x, scale_it = T, scale_low = -5.12, scale_high = 5.12,
  noise = 0, ...)

moon_high(x, scale_it = F, scale_low = 0, scale_high = 1, noise = 0,
  ...)

linkletter_nosignal(x, scale_it = F, scale_low = 0, scale_high = 1,
  noise = 0, ...)

morris(x, scale_it = T, scale_low = 0, scale_high = 1, noise = 0, ...)

detpep8d(x, scale_it = T, scale_low = 0, scale_high = 1, noise = 0, ...)

hartmann(x, scale_it = F, scale_low = 0, scale_high = 1, noise = 0, ...)

quad_peaks(x, scale_it = T, scale_low = 0, scale_high = 1, noise = 0,
  ...)

quad_peaks_slant(x, scale_it = T, scale_low = 0, scale_high = 1,
  noise = 0, ...)

test_func_apply(func, x, scale_it, scale_low, scale_high, noise = 0, ...)
```

Arguments

x	Input value, either a matrix whose rows are points or a vector for a single point. Be careful with 1-D functions.
scale_it	Should the data be scaled from [0, 1]^D to [scale_low, scale_high]? This means the input data is confined to be in [0, 1]^D, but the function isn't.
scale_low	Lower bound for each variable
scale_high	Upper bound for each variable
noise	If white noise should be added, specify the standard deviation for normal noise

freq	Wave frequency for sqrtsin and powsin
pow	Power for powsin
a	A constant for ackley()
b	A constant for ackley()
c	A constant for ackley()
...	Additional parameters for func
func	A function to evaluate

Value

Function values at x

References

http://www.abe.ufl.edu/jjones/ABE_5646/2010/Morris.1991

http://www.tandfonline.com/doi/pdf/10.1198/TECH.2010.09157?needAccess=true

Examples

```
branin(runif(2))
branin(matrix(runif(20), ncol=2))
borehole(runif(8))
borehole(matrix(runif(80), ncol=8))
franke(runif(2))
zhou1998(runif(2))
currin1991(runif(2))
lim2002(runif(2))
banana(runif(2))
x <- y <- seq(0, 1, len=100)
z <- outer(x, y, Vectorize(function(a, b){banana(c(a, b))}))
contour(x, y, z)
gaussian1(runif(2))
sinumoid(runif(2))
x <- y <- seq(0, 1, len=100)
z <- outer(x, y, Vectorize(function(a, b){sinumoid(c(a, b))}))
contour(x, y, z)
waterfall(runif(2))
sqrtsin(runif(1))
curve(sqrtsin(matrix(x,ncol=1)))
powsin(runif(1))#,pow=2)
OTL_Circuit(runif(6))
OTL_Circuit(matrix(runif(60),ncol=6))
GoldsteinPrice(runif(2))
GoldsteinPrice(matrix(runif(60),ncol=2))
GoldsteinPriceLog(runif(2))
GoldsteinPriceLog(matrix(runif(60),ncol=2))
ackley(runif(2))
ackley(matrix(runif(60),ncol=2))
piston(runif(7))
piston(matrix(runif(7*20),ncol=7))
```

```
wingweight(runif(10))
wingweight(matrix(runif(10*20),ncol=10))
welch(runif(20))
welch(matrix(runif(20*20),ncol=20))
robotarm(runif(8))
robotarm(matrix(runif(8*20),ncol=8))
RoosArnold(runif(8))
RoosArnold(matrix(runif(8*20),ncol=8))
Gfunction(runif(8))
Gfunction(matrix(runif(8*20),ncol=8))
beale(runif(2))
beale(matrix(runif(2*20),ncol=2))
easom(runif(2))
easom(matrix(runif(2*20),ncol=2))
griewank(runif(2))
griewank(matrix(runif(2*20),ncol=2))
hump(runif(2))
hump(matrix(runif(2*20),ncol=2))
levy(runif(2))
levy(matrix(runif(2*20),ncol=2))
michalewicz(runif(2))
michalewicz(matrix(runif(2*20),ncol=2))
rastrigin(runif(2))
rastrigin(matrix(runif(2*20),ncol=2))
moon_high(runif(20))
moon_high(matrix(runif(20*20),ncol=20))
linkletter_nosignal(runif(2))
linkletter_nosignal(matrix(runif(2*20),ncol=2))
morris(runif(20))
morris(matrix(runif(20*20),ncol=20))
detpep8d(runif(2))
detpep8d(matrix(runif(2*20),ncol=2))
hartmann(runif(2))
hartmann(matrix(runif(6*20),ncol=6))
quad_peaks(runif(2))
quad_peaks(matrix(runif(2*20),ncol=2))
quad_peaks_slant(runif(2))
quad_peaks_slant(matrix(runif(2*20),ncol=2))
x <- matrix(seq(0,1,length.out=10), ncol=1)
y <- test_func_apply(sin, x, TRUE, 0, 2*pi, .05)
plot(x,y)
curve(sin(2*pi*x), col=2, add=TRUE)
```

`nsin` *Wave functions*

Description

nsin: Block wave

vsin: v wave

Usage

```
nsin(xx)

vsin(xx)
```

Arguments

xx	Input values

Value

nsin evaluated at nsin

Examples

```
curve(nsin(2*pi*x), n = 1000)
curve(nsin(12*pi*x), n = 1000)
curve(vsin(2*pi*x), n = 1000)
curve(vsin(12*pi*x), n = 1000)
```

---

numGrad *Create function calculating the numerical gradient*

Description

Create function calculating the numerical gradient

Usage

```
numGrad(func, ...)
```

Arguments

func	Function to get gradient of.
...	Arguments passed to numDeriv::grad().

Value

A gradient function

Examples

```
numGrad(sin)
```

numHessian *Create function calculating the numerical hessian*

Description

Create function calculating the numerical hessian

Usage

```
numHessian(func, ...)
```

Arguments

func	Function to get hessian of
...	Arguments passed to numDeriv::hessian().

Value

A hessian function

Examples

```
numHessian(sin)
```

RFF *Evaluate an RFF (random wave function) at given input*

Description

Evaluate an RFF (random wave function) at given input

Usage

```
RFF(x, freq, mag, dirr, offset, wave = sin, noise = 0)
```

Arguments

x	Matrix whose rows are points to evaluate or a vector representing a single point. In 1 dimension you must use a matrix for multiple points, not a vector.
freq	Vector of wave frequencies
mag	Vector of wave magnitudes
dirr	Matrix of wave directions
offset	Vector of wave offsets
wave	Type of wave
noise	Standard deviation of random normal noise to add

Value

Output of RFF evaluated at x

Examples

```
curve(RFF(matrix(x,ncol=1),3,1,1,0))
curve(RFF(matrix(x,ncol=1),3,1,1,0, noise=.1), n=1e3, type='p', pch=19)

curve(RFF(matrix(x,ncol=1),c(3,20),c(1,.1),c(1,1),c(0,0)), n=1e3)
```

`RFF_get` *Create a new RFF function*

Description

Create a new RFF function

Usage

```
RFF_get(D = 2, M = 30, wave = sin, noise = 0, seed = NULL)
```

Arguments

`D`	Number of dimensions
`M`	Number of random waves
`wave`	Type of wave
`noise`	Standard deviation of random normal noise to add
`seed`	Seed to set before randomly selecting function

Value

A random wave function

Examples

```
func <- RFF_get(D=1)
curve(func)

f <- RFF_get(D=1, noise=.1)
curve(f(matrix(x,ncol=1)))
for(i in 1:100) curve(f(matrix(x,ncol=1)), add=TRUE, col=sample(2:8,1))
```

standard_test_func *Create a standard test function.*

Description

This makes it easier to create many functions that follow the same template. R CMD check doesn't like the ... if this command is used to create functions in the package, so it is not currently used.

Usage

```
standard_test_func(func, scale_it_ = F, scale_low_ = NULL,
  scale_high_ = NULL, noise_ = 0, ...)
```

Arguments

func	A function that takes a vector representing a single point.
scale_it_	Should the function scale the inputs from [0, 1]^D to [scale_low_, scale_high_] by default? This can be overridden when actually giving the output function points to evaluate.
scale_low_	What is the default lower bound of the data?
scale_high_	What is the default upper bound of the data?
noise_	Should noise be added to the function by default?
...	Parameters passed to func when evaluating points.

Value

A test function created using the standard_test_func template.

Examples

```
.gaussian1 <- function(x, center=.5, s2=.01) {
  exp(-sum((x-center)^2/2/s2))
}
gaussian1 <- standard_test_func(.gaussian1, scale_it=FALSE, scale_low = c(0,0), scale_high = c(1,1))
curve(gaussian1(matrix(x,ncol=1)))
```

TF_ackley *TF_ackley: Ackley function for evaluating a single point.*

Description

TF_ackley: Ackley function for evaluating a single point.

Usage

```
TF_ackley(x, a = 20, b = 0.2, c = 2 * pi)
```

Arguments

x	Input vector at which to evaluate.
a	A constant for ackley()
b	A constant for ackley()
c	A constant for ackley()

Value

Function output evaluated at x.

Examples

```
TF_ackley(c(0, 0)) # minimum of zero, hard to solve
```

TF_beale *TF_beale: Beale function for evaluating a single point.*

Description

TF_beale: Beale function for evaluating a single point.

Usage

```
TF_beale(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_beale(rep(0,2))
TF_beale(rep(1,2))
```

`TF_branin` *Base test function.*

Description

TF_branin: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_borehole: A function taking in a single vector. 8 dimensional function. See corresponding function with "TF_" for more details.

TF_franke: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_zhou1998: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_currin1991: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_lim2002: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_banana: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_gaussian1: A function taking in a single vector. Any dimensional function. See corresponding function with "TF_" for more details.

TF_sinumoid: A function taking in a single vector. 2 dimensional function. See corresponding function with "TF_" for more details.

TF_sqrtsin: A function taking in a single vector. Any dimensional function. See corresponding function with "TF_" for more details.

TF_powsin: A function taking in a single vector. Any dimensional function. See corresponding function with "TF_" for more details.

TF_OTL_Circuit: OTL Circuit function for evaluating a single point

Usage

```
TF_branin(x, a = 1, b = 5.1/(4 * pi^2), cc = 5/pi, r = 6, s = 10,
  tt = 1/(8 * pi))

TF_borehole(x)

TF_franke(x)

TF_zhou1998(x)
```

```
TF_currin1991(x)

TF_lim2002(x)

TF_banana(x)

TF_gaussian1(x, center = 0.5, s2 = 0.01)

TF_sinumoid(x)

TF_sqrtsin(x, freq = 2 * pi)

TF_powsin(x, freq = 2 * pi, pow = 0.7)

TF_OTL_Circuit(x)
```

Arguments

x	Input vector at which to evaluate.
a	Parameter for TF_branin
b	Parameter for TF_branin
cc	Parameter for TF_branin
r	Parameter for TF_branin
s	Parameter for TF_branin
tt	Parameter for TF_branin
center	Where to center the function, a vector.
s2	Variance of the Gaussian.
freq	Wave frequency for TF_sqrtsin and TF_powsin
pow	Power to raise wave to for TF_powsin.

Value

Function output evaluated at x.

Examples

```
TF_branin(runif(2))
TF_borehole(runif(8))
TF_franke(runif(2))
TF_zhou1998(runif(2))
TF_currin1991(runif(2))
TF_lim2002(runif(2))
TF_banana(runif(2))
TF_gaussian1(runif(2))
TF_sinumoid(runif(2))
TF_sqrtsin(runif(2))
```

```
TF_powsin(runif(2))
TF_OTL_Circuit(c(50,25,0.5,1.2,0.25,50))
```

`TF_detpep8d` *TF_detpep8d: detpep8d function for evaluating a single point.*

Description

TF_detpep8d: detpep8d function for evaluating a single point.

Usage

```
TF_detpep8d(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_detpep8d(rep(0,2))
TF_detpep8d(rep(1,2))
```

`TF_easom` *TF_easom: Easom function for evaluating a single point.*

Description

TF_easom: Easom function for evaluating a single point.

Usage

```
TF_easom(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_easom(rep(0,2))
TF_easom(rep(1,2))
```

TF_Gfunction *TF_Gfunction: G-function for evaluating a single point.*

Description

TF_Gfunction: G-function for evaluating a single point.

Usage

```
TF_Gfunction(x, a = (1:length(x) - 1)/2)
```

Arguments

x	Input vector at which to evaluate.
a	Parameter for Gfunction

Value

Function output evaluated at x.

Examples

```
TF_Gfunction(rep(0,8))
TF_Gfunction(rep(1,8))
```

TF_GoldsteinPrice *TF_GoldsteinPrice: Goldstein Price function for evaluating a single point*

Description

TF_GoldsteinPrice: Goldstein Price function for evaluating a single point

Usage

```
TF_GoldsteinPrice(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_GoldsteinPrice(c(0, -1)) # minimum
```

TF_GoldsteinPriceLog *TF_GoldsteinPrice: Goldstein Price function for evaluating a single point on a log scale, normalized to have mean 0 and variance 1.*

Description

TF_GoldsteinPrice: Goldstein Price function for evaluating a single point on a log scale, normalized to have mean 0 and variance 1.

Usage

```
TF_GoldsteinPriceLog(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_GoldsteinPriceLog(c(0, -1)) # minimum
```

TF_griewank *TF_griewank: Griewank function for evaluating a single point.*

Description

TF_griewank: Griewank function for evaluating a single point.

Usage

```
TF_griewank(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_griewank(rep(0,2))
TF_griewank(rep(1,2))
```

TF_hartmann *TF_hartmann: hartmann function for evaluating a single point.*

Description

TF_hartmann: hartmann function for evaluating a single point.

Usage

```
TF_hartmann(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_hartmann(rep(0,6))
TF_hartmann(rep(1,6))
TF_hartmann(c(.20169, .150011, .476874, .275332, .311652, .6573)) # Global minimum of -3.322368
```

TF_hump *TF_hump: Hump function for evaluating a single point.*

Description

TF_hump: Hump function for evaluating a single point.

Usage

```
TF_hump(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_hump(rep(0,2))
TF_hump(rep(1,2))
```

TF_levy *TF_levy: Levy function for evaluating a single point.*

Description

TF_levy: Levy function for evaluating a single point.

Usage

```
TF_levy(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_levy(rep(0,2))
TF_levy(rep(1,2))
```

TF_linkletter_nosignal *TF_linkletter_nosignal: Linkletter (2006) no signal function for evaluating a single point.*

Description

TF_linkletter_nosignal: Linkletter (2006) no signal function for evaluating a single point.

Usage

```
TF_linkletter_nosignal(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_linkletter_nosignal(rep(0,2))
TF_linkletter_nosignal(rep(1,2))
```

TF_michalewicz *TF_michalewicz: Michalewicz function for evaluating a single point.*

Description

TF_michalewicz: Michalewicz function for evaluating a single point.

Usage

```
TF_michalewicz(x, m = 10)
```

Arguments

x	Input vector at which to evaluate.
m	Parameter for the michalewicz function

Value

Function output evaluated at x.

Examples

```
TF_michalewicz(rep(0,2))
TF_michalewicz(rep(1,2))
```

TF_moon_high *TF_moon_high: Moon (2010) high-dimensional function for evaluating a single point.*

Description

TF_moon_high: Moon (2010) high-dimensional function for evaluating a single point.

Usage

```
TF_moon_high(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_moon_high(rep(0,20))
TF_moon_high(rep(1,20))
```

`TF_morris` *TF_morris: morris function for evaluating a single point.*

Description

TF_morris: morris function for evaluating a single point.

Usage

```
TF_morris(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

References

http://www.abe.ufl.edu/jjones/ABE_5646/2010/Morris.1991

Examples

```
TF_morris(rep(0,20))
TF_morris(rep(1,20))
```

`TF_piston` *TF_piston: Piston simulation function for evaluating a single point.*

Description

TF_piston: Piston simulation function for evaluating a single point.

Usage

```
TF_piston(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_piston(c(30,.005,.002,1e3,9e4,290,340)) # minimum of zero, hard to solve
```

TF_quad_peaks *TF_quad_peaks: quad_peaks function for evaluating a single point.*

Description

TF_quad_peaks: quad_peaks function for evaluating a single point.

Usage

```
TF_quad_peaks(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_quad_peaks(rep(0,2))
TF_quad_peaks(rep(1,2))
```

TF_quad_peaks_slant *TF_quad_peaks_slant: quad_peaks_slant function for evaluating a single point.*

Description

TF_quad_peaks_slant: quad_peaks_slant function for evaluating a single point.

Usage

```
TF_quad_peaks_slant(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_quad_peaks_slant(rep(0,2))
TF_quad_peaks_slant(rep(1,2))
```

TF_rastrigin *TF_rastrigin: Rastrigin function for evaluating a single point.*

Description

TF_rastrigin: Rastrigin function for evaluating a single point.

Usage

```
TF_rastrigin(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_rastrigin(rep(0,2))
TF_rastrigin(rep(1,2))
```

TF_robotarm *TF_robotarm: Robot arm function for evaluating a single point.*

Description

TF_robotarm: Robot arm function for evaluating a single point.

Usage

```
TF_robotarm(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_robotarm(rep(0,8))
TF_robotarm(rep(1,8))
```

`TF_RoosArnold` *TF_RoosArnold: Roos & Arnold (1963) function for evaluating a single point.*

Description

TF_RoosArnold: Roos & Arnold (1963) function for evaluating a single point.

Usage

```
TF_RoosArnold(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_RoosArnold(rep(0,8))
TF_RoosArnold(rep(1,8))
```

`TF_welch` *TF_welch: Welch function for evaluating a single point.*

Description

TF_welch: Welch function for evaluating a single point.

Usage

```
TF_welch(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_welch(rep(0,20)) # minimum of zero, hard to solve
```

`TF_wingweight` *TF_wingweight: Wing weight function for evaluating a single point.*

Description

TF_wingweight: Wing weight function for evaluating a single point.

Usage

```
TF_wingweight(x)
```

Arguments

x	Input vector at which to evaluate.

Value

Function output evaluated at x.

Examples

```
TF_wingweight(c(150,220,6,-10,16,.5,.08,2.5,1700,.025)) # minimum of zero, hard to solve
```

Index