

# Package ‘DiceOptim’

March 26, 2012

**Version** 1.3

**Title** Kriging-based optimization for computer experiments

**Date** 2012-03-08

**Author** D. Ginsbourger, V. Picheny, O. Roustant, with contributions by C. Chevalier and T. Wagner

**Maintainer** D. Ginsbourger <david.ginsbourger@stat.unibe.ch>

**Description** Expected Improvement. EGO algorithm. Parallelized versions of EGO: Constant Liars. Noisy Kriging-based Optimization criteria and algorithms, including AEI, AKG, EQI, and EI.plugin.

**Depends** DiceKriging (>= 1.2), rgenoud, MASS, lhs

**License** GPL-3

**URL** <http://dice.emse.fr/>

**Repository** CRAN

**Date/Publication** 2012-03-26 12:45:54

## R topics documented:

DiceOptim-package . . . . .	2
AEI . . . . .	4
AEI.grad . . . . .	6
AEI.grad_optim . . . . .	8
AKG . . . . .	9
AKG.grad . . . . .	11
AKG.grad_optim . . . . .	12
CL.nsteps . . . . .	13
EGO.nsteps . . . . .	16
EI . . . . .	20
EI.grad . . . . .	21
EI.plugin . . . . .	23
EI.plugin.grad . . . . .	26

EQI . . . . .	27
EQI.grad . . . . .	29
EQI.grad_optim . . . . .	31
kriging.quantile . . . . .	32
kriging.quantile.grad . . . . .	34
kriging.quantile.grad_optim . . . . .	36
max_AEI . . . . .	38
max_AKG . . . . .	40
max_EI . . . . .	42
max_EI.plugin . . . . .	45
max_EQI . . . . .	47
max_qEI.CL . . . . .	49
min_quantile . . . . .	51
noisy.optimizer . . . . .	53
qEI . . . . .	57
update_km . . . . .	59

<b>Index</b>	<b>61</b>
--------------	-----------

---

DiceOptim-package	<i>Kriging-based optimization methods for computer experiments</i>
-------------------	--

---

## Description

Sequential and parallel Kriging-based optimization methods relying on expected improvement criteria.

## Details

Package: DiceOptim  
 Type: Package  
 Version: 1.3  
 Date: March 2012  
 License: GPL-3

## Note

This work is a follow-up of DiceOptim 1.0, which was conducted within the frame of the DICE (Deep Inside Computer Experiments) Consortium between ARMINES, Renault, EDF, IRSN, ONERA and TOTAL S.A. (<http://www.dice-consortium.fr/>).

The authors would like to thank Yves Deville for his precious advices in R programming and packaging, as well as the DICE members for useful feedbacks, and especially Yann Richet (IRSN) for numerous discussions concerning the user-friendliness of this package.

Package rgenoud >=5.3.3. is recommended.

Important functions or methods:

EI	One-point noise-free EI criterion
qEI	q-points noise-free EI criterion
EGO.nsteps	Sequential EI Algorithm —model updates including re-estimation of covariance hyperparameters— with a fixed number of iterations (nsteps)
max_EI	One-point noise-free EI maximization. No need to specify any objective function
max_qEI.CL	(sub-)maximization of the q-points EI, based on the Constant Liar heuristic No need to specify any objective function
CL.nsteps	Parallel EI Algorithm —model updates including re-estimation of covariance hyperparameters— with a fixed number of iterations (nsteps)
EQI	One-point noisy EI-like criterion (Expected Quantile Improvement)
AEI	One-point noisy EI-like criterion (Augmented Expected Improvement)
AKG	One-point noisy EI criterion (Approximate Knowledge Gradient)
noisy.optimizer	Sequential EI-like Algorithms for Noisy Kriging-based Optimization

### Author(s)

David Ginsbourger (Departement of Mathematics and Statistics, University of Bern, Switzerland)

Victor Picheny (CERFACS, Toulouse, France)

Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

with contributions by C. Chevalier and T. Wagner

### References

- N.A.C. Cressie (1993), *Statistics for spatial data*, Wiley series in probability and mathematical statistics.
- D. Ginsbourger (2009), *Multiplés metamodeles pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>
- D. Ginsbourger, R. Le Riche, and L. Carraro (2009), chapter "Kriging is well-suited to parallelize optimization", to appear in *Computational Intelligence in Expensive Optimization Problems*, Studies in Evolutionary Learning and Optimization, Springer.
- D.R. Jones (2001), A taxonomy of global optimization methods based on response surfaces, *Journal of Global Optimization*, 21, 345-383.
- D.R. Jones, M. Schonlau, and W.J. Welch (1998), Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, 13, 455-492.
- W.R. Jr. Mebane and J.S. Sekhon (2009), in press, Genetic optimization using derivatives: The rgenoud package for R, *Journal of Statistical Software*.
- J. Mockus (1988), *Bayesian Approach to Global Optimization*. Kluwer academic publishers.
- C.E. Rasmussen and C.K.I. Williams (2006), *Gaussian Processes for Machine Learning*, the MIT Press, <http://www.GaussianProcess.org/gpml>
- B.D. Ripley (1987), *Stochastic Simulation*, Wiley.
- T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.

M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

---

 AEI

---

*Augmented Expected Improvement*


---

### Description

Evaluation of the Augmented Expected Improvement (AEI) criterion, which is a modification of the classical EI criterion for noisy functions. The AEI consists of the regular EI multiplied by a penalization function that accounts for the diminishing payoff of observation replicates. The current minimum `y_min` is chosen as the kriging predictor of the observation with smallest kriging quantile.

### Usage

```
AEI(x, model, new.noise.var=0, y.min=NULL, type = "UK", envir=NULL)
```

### Arguments

<code>x</code>	the input vector at which one wants to evaluate the criterion
<code>model</code>	a Kriging model of "km" class
<code>new.noise.var</code>	the (scalar) noise variance of the future observation.
<code>y.min</code>	The kriging predictor at the current best point (point with smallest kriging quantile). If not provided, this quantity is evaluated.
<code>type</code>	Kriging type: "SK" or "UK"
<code>envir</code>	environment for saving intermediate calculations and reusing them within <code>AEI.grad_optim</code>

### Value

Augmented Expected Improvement

### Author(s)

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

### References

D.R. Jones, M. Schonlau, and W.J. Welch (1998), Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, 13, 455-492.

D. Huang, T.T. Allen, W.I. Notz, and N. Zeng (2006), Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models, *Journal of Global Optimization*, 34, 441-466.

## Examples

```
#####
###   AEI SURFACE ASSOCIATED WITH AN ORDINARY KRIGING MODEL       ###
### OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN ###
#####

#library("lhs")
set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
            covtype="gauss", noise.var=rep(noise.var,1,doe.size),
            lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
func.grid <- rep(0,1,nt)

for (i in 1:nt)
{
  crit.grid[i] <- AEI(x=design.grid[i,], model=model, new.noise.var=noise.var)
  func.grid[i] <- test.function(design.grid[i,])
}

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1", "V2")
pred <- predict.km(model, newdata=design.grid, type="UK")
mk.grid <- pred$m
sk.grid <- pred$sd

# Plot actual function
```

```

z.grid <- matrix(func.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Actual function");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging mean
z.grid <- matrix(mk.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Kriging mean");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging variance
z.grid <- matrix(sk.grid^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Kriging variance");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot AEI criterion
z.grid <- matrix(crit.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("AEI");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

```

---

AEI.grad

*AEI's Gradient*


---

## Description

Gradient of the Augmented Expected Improvement (AEI) criterion.

## Usage

```
AEI.grad(x, model, new.noise.var=0, y.min=NULL, type = "UK")
```

## Arguments

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	the (scalar) noise variance of the new observation.
y.min	The kriging predictor at the current best point (point with smallest kriging quantile). If not provided, this quantity is evaluated.
type	Kriging type: "SK" or "UK"

**Value**

Gradient of the Augmented Expected Improvement

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```

set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size),
           lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
#names(design.grid) <- c("V1", "V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
crit.grad <- matrix(0,nt,2)

for (i in 1:nt)
{
  crit.grid[i] <- AEI(x=design.grid[i,], model=model, new.noise.var=noise.var)
  crit.grad[i,] <- AEI.grad(x=design.grid[i,], model=model, new.noise.var=noise.var)
}

z.grid <- matrix(crit.grid, n.grid, n.grid)
contour(x.grid,y.grid, z.grid, 30)
title("AEI and its gradient")
points(model@X[,1],model@X[,2],pch=17,col="blue")

```

```

for (i in 1:nt)
{
  x <- design.grid[i,]
  arrows(x$Var1,x$Var2, x$Var1+crit.grad[i,1]*.6,x$Var2+crit.grad[i,2]*.6,
length=0.04,code=2,col="orange",lwd=2)
}

```

---

AEI.grad\_optim

*AEI's Gradient*


---

### Description

Gradient of the Augmented Expected Improvement (AEI) criterion.

### Usage

```
AEI.grad_optim(x, model, new.noise.var=0, y.min, type = "UK", envir)
```

### Arguments

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	the (scalar) noise variance of the new observation.
y.min	The kriging predictor at the current best point (point with smallest kriging quantile). If not provided, this quantity is evaluated.
type	Kriging type: "SK" or "UK"
envir	environment for inheriting intermediate calculations from AEI

### Value

Gradient of the Augmented Expected Improvement

### Author(s)

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (IMSV, University of Bern)

---

AKG *Approximate Knowledge Gradient (AKG)*

---

**Description**

Evaluation of the Approximate Knowledge Gradient (AKG) criterion.

**Usage**

```
AKG(x, model, new.noise.var=0, type = "UK", envir=NULL)
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	(scalar) noise variance of the future observation. Default value is 0 (noise-free observation).
type	Kriging type: "SK" or "UK"
envir	environment for saving intermediate calculations and reusing them within AKG.grad_optim

**Value**

Approximate Knowledge Gradient

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
#####
###   AKG SURFACE ASSOCIATED WITH AN ORDINARY KRIGING MODEL   #####
### OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN #####
#####

#library("lhs")
set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2
```

```

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
            covtype="gauss", noise.var=rep(noise.var,1,doe.size),
            lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
func.grid <- rep(0,1,nt)

for (i in 1:nt)
{
  crit.grid[i] <- AKG(x=design.grid[i,], model=model, new.noise.var=noise.var)
  func.grid[i] <- test.function(design.grid[i,])
}

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1", "V2")
pred <- predict.km(model, newdata=design.grid, type="UK")
mk.grid <- pred$m
sk.grid <- pred$sd

# Plot actual function
z.grid <- matrix(func.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Actual function");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging mean
z.grid <- matrix(mk.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Kriging mean");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging variance
z.grid <- matrix(sk.grid^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Kriging variance");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

```

```
# Plot AKG criterion
z.grid <- matrix(crit.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("AKG");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})
```

AKG.grad

*AKG's Gradient***Description**

Gradient of the Approximate Knowledge Gradient (AKG) criterion.

**Usage**

```
AKG.grad(x, model, new.noise.var=0, type = "UK")
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	(scalar) noise variance of the future observation. Default value is 0 (noise-free observation).
type	Kriging type: "SK" or "UK"

**Value**

Gradient of the Approximate Knowledge Gradient

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France)

**Examples**

```
#####
###   AKG SURFACE AND ITS GRADIENT ASSOCIATED WITH AN ORDINARY   ###
###                                     KRIGING MODEL             ###
### OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN ###
#####

set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
```

```

test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
covtype="gauss", noise.var=rep(noise.var,1,doe.size),
lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
#names(design.grid) <- c("V1", "V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
crit.grad <- matrix(0,nt,2)

for (i in 1:nt)
{
crit.grid[i] <- AKG(x=design.grid[i,], model=model, new.noise.var=noise.var)
crit.grad[i,] <- AKG.grad(x=design.grid[i,], model=model, new.noise.var=noise.var)
}

z.grid <- matrix(crit.grid, n.grid, n.grid)
contour(x.grid,y.grid, z.grid, 30)
title("AKG and its gradient")
points(model@X[,1],model@X[,2],pch=17,col="blue")

for (i in 1:nt)
{
x <- design.grid[i,]
arrows(x$Var1,x$Var2, x$Var1+crit.grad[i,1]*.2,x$Var2+crit.grad[i,2]*.2,
length=0.04,code=2,col="orange",lwd=2)
}

```

**Description**

Gradient of the Approximate Knowledge Gradient (AKG) criterion.

**Usage**

```
AKG.grad_optim(x, model, new.noise.var=0, type = "UK", envir)
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	(scalar) noise variance of the future observation. Default value is 0 (noise-free observation).
type	Kriging type: "SK" or "UK"
envir	environment for inheriting intermediate calculations from AKG

**Value**

Gradient of the Approximate Knowledge Gradient

**Author(s)**

Victor Picheny

---

CL.nsteps

*Parallelized version of EGO.nsteps, based on the CL strategy*

---

**Description**

Adaptation of the EI Algorithm for synchronous parallel computing, based on the "Constant Liar" heuristic strategy. Indeed, `max_qEI.CL` is used at every iteration to provide a new q-points design. The Kriging model is updated after each batch of objective function evaluations, including re-estimation of the covariance hyperparameters.

**Usage**

```
CL.nsteps(model, fun, npoints, nsteps, lower, upper, parinit = NULL,
kmcontrol = NULL, control = NULL)
```

**Arguments**

model	an object of class <code>km</code> ,
fun	the objective function to be minimized,
npoints	an integer representing the desired number of parallel computations,
nsteps	an integer representing the desired number of batches,
lower	vector of lower bounds for the variables to be optimized over,
upper	vector of upper bounds for the variables to be optimized over,
parinit	optional vector of initial values for the variables to be optimized over,
control	an optional list of control parameters for optimization. One can control "pop.size" (default : [4+3*log(nb of variables)]), "max.generations" (default :5), "wait.generations" (default :2), "BFGSburnin" (default :0), of the function <code>genoud</code> .
kmcontrol	an optional list representing the control variables for the re-estimation of the kriging model. The items are the same as in <code>km</code> : penalty, optim.method, parinit, control. The default values are those contained in <code>model</code> , typically corresponding to the variables used in <code>km</code> to estimate a kriging model from the initial design points.

**Value**

A list with components:

par	a data frame representing the additional points visited during the algorithm,
value	a data frame representing the response values at the points given in <code>par</code> ,
npoints	an integer representing the number of parallel computations (given in argument),
nsteps	an integer representing the desired number of iterations (given in argument),
lastmodel	an object of class <code>km</code> corresponding to the last kriging model fitted.

**Author(s)**

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, Switzerland)  
Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

**References**

- D. Ginsbourger (2009), *Multiplés metamodeles pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>
- D. Ginsbourger, R. Le Riche, and L. Carraro (2009), chapter "Kriging is well-suited to parallelize optimization", to appear in *Computational Intelligence in Expensive Optimization Problems*, Studies in Evolutionary Learning and Optimization, Springer.

W.R. Jr. Mebane and J.S. Sekhon (2009), in press, Genetic optimization using derivatives: The rgenoud package for R, *Journal of Statistical Software*.

B.D. Ripley (1987), *Stochastic Simulation*, Wiley.

T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.

M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

### See Also

[max\\_qEI.CL](#), [qEI](#), [EI](#), [max\\_EI](#), [EG0.nsteps](#)

### Examples

```
set.seed(123)
# 3 Iterations of the Constant Liar are applied to Branin's function,
# with 8 points in parallel at each iteration

# a 9-points factorial design, and the corresponding response
d <- 2
n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.branin <- apply(design.fact, 1, branin)
response.branin <- data.frame(response.branin)
names(response.branin) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.branin,
covtype="gauss", control=list(pop.size=50,trace=FALSE), parinit=c(0.5, 0.5))

# parallel EGO
library(rgenoud)
nsteps <- 3
npoints <- 8
lower <- rep(0,d)
upper <- rep(1,d)

oEG0paralle11 <- CL.nsteps(model=fitted.model1, fun=branin, npoints=npoints,
nsteps=nsteps, lower, upper, control=list(pop.size=20, BFGSburnin=2))

print(oEG0paralle11)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, branin)
z.grid <- matrix(response.grid, n.grid, n.grid)
```

```

contour(x.grid,y.grid,z.grid,40)
title("3 iterations of Constant Liar with 8 parallel searches")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGOparallel1$par[1:npoints],, pch=19, col="red")
points(oEGOparallel1$par[(npoints+1):(2*npoints)],, pch=19, col="violet")
points(oEGOparallel1$par[(2*npoints+1):(3*npoints)],, pch=19, col="green")
text(oEGOparallel1$par[,1], oEGOparallel1$par[,2],
labels=1:(npoints*nsteps), pos=3)

```

---

EGO.nsteps	<i>Sequential EI maximization and model re-estimation, with a number of iterations fixed in advance by the user</i>
------------	---

---

### Description

Executes *nsteps* iterations of the EGO method to an object of class `km`. At each step, a kriging model is re-estimated (including covariance parameters re-estimation) based on the initial design points plus the points visited during all previous iterations; then a new point is obtained by maximizing the Expected Improvement criterion (EI).

### Usage

```

EGO.nsteps(model, fun, nsteps, lower, upper, parinit=NULL,
control=NULL, kmcontrol=NULL)

```

### Arguments

model	an object of class <code>km</code> ,
fun	the objective function to be minimized,
nsteps	an integer representing the desired number of iterations,
lower	vector of lower bounds for the variables to be optimized over,
upper	vector of upper bounds for the variables to be optimized over,
parinit	optional vector of initial values for the variables to be optimized over,
control	an optional list of control parameters for optimization. One can control "pop.size" (default : [4+3*log(nb of variables)]), "max.generations" (default :5), "wait.generations" (default :2), "BFGSburnin" (default :0), of the function <code>genoud</code> .
kmcontrol	an optional list representing the control variables for the re-estimation of the kriging model. The items are the same as in <code>km</code> : penalty, optim.method, parinit, control. The default values are those contained in <code>model</code> , typically corresponding to the variables used in <code>km</code> to estimate a kriging model from the initial design points.

**Value**

A list with components:

par	a data frame representing the additional points visited during the algorithm,
value	a data frame representing the response values at the points given in par,
npoints	an integer representing the number of parallel computations (=1 here),
nsteps	an integer representing the desired number of iterations (given in argument),
lastmodel	an object of class <a href="#">km</a> corresponding to the last kriging model fitted.

**Note**

Most EGO-like methods (EI algorithms) usually work with Ordinary Kriging (constant trend), by maximization of the expected improvement. Here, the EI maximization is also possible with any linear trend. However, note that the optimization may perform much faster and better when the trend is a constant since it is the only case where the analytical gradient is available.

For more details on `kmcontrol`, see the documentation of [km](#).

**Author(s)**

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, Switzerland)  
Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

**References**

- D.R. Jones, M. Schonlau, and W.J. Welch (1998), Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, 13, 455-492.
- J. Mockus (1988), *Bayesian Approach to Global Optimization*. Kluwer academic publishers.
- T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.
- M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

**See Also**

[EI](#), [max\\_EI](#), [EI.grad](#), [CL.nsteps](#)

**Examples**

```
set.seed(123)
#####
### 10 ITERATIONS OF EGO ON THE BRANIN FUNCTION,   ###
### STARTING FROM A 9-POINTS FACTORIAL DESIGN     ###
#####
# a 9-points factorial design, and the corresponding response
d <- 2
```

```

n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.branin <- apply(design.fact, 1, branin)
response.branin <- data.frame(response.branin)
names(response.branin) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.branin,
covtype="gauss", control=list(pop.size=50,trace=FALSE), parinit=c(0.5, 0.5))

# EGO n steps
library(rgenoud)
nsteps <- 10
lower <- rep(0,d); upper <- rep(1,d)
oEGO <- EGO.nsteps(model=fitted.model1, fun=branin, nsteps=nsteps,
lower, upper, control=list(pop.size=20, BFGSburnin=2))
print(oEGO$par)
print(oEGO$value)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, branin)
z.grid <- matrix(response.grid, n.grid, n.grid)
contour(x.grid, y.grid, z.grid, 40)
title("Branin function")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGO$par, pch=19, col="red")
text(oEGO$par[,1], oEGO$par[,2], labels=1:nsteps, pos=3)

#####
### 20 ITERATIONS OF EGO ON THE GOLDSTEIN-PRICE,      ###
### STARTING FROM A 9-POINTS FACTORIAL DESIGN        ###
#####

# a 9-points factorial design, and the corresponding response
d <- 2
n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.goldsteinPrice <- apply(design.fact, 1, goldsteinPrice)
response.goldsteinPrice <- data.frame(response.goldsteinPrice)
names(response.goldsteinPrice) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.goldsteinPrice,

```

```

covtype="gauss", control=list(pop.size=50, max.generations=50,
wait.generations=5, BFGSburnin=10,trace=FALSE), parinit=c(0.5, 0.5), optim.method="gen")

# EGO n steps
library(rgenoud)
nsteps <- 20
lower <- rep(0,d)
upper <- rep(1,d)
oEGO <- EGO.nsteps(model=fitted.model1, fun=goldsteinPrice, nsteps=nsteps,
lower, upper, control=list(pop.size=20, BFGSburnin=2))
print(oEGO$par)
print(oEGO$value)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, goldsteinPrice)
z.grid <- matrix(response.grid, n.grid, n.grid)
contour(x.grid, y.grid, z.grid, 40)
title("Goldstein-Price Function")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGO$par, pch=19, col="red")
text(oEGO$par[,1], oEGO$par[,2], labels=1:nsteps, pos=3)

#####
###  nsteps ITERATIONS OF EGO ON THE HARTMAN6 FUNCTION,      ###
###  STARTING FROM A 10-POINTS UNIFORM DESIGN                ###
#####

fonction<-hartman6
data(mydata)
a <- mydata
nb<-10
nsteps <- 2
x1<-a[[1]][1:nb];x2<-a[[2]][1:nb];x3<-a[[3]][1:nb]
x4<-a[[4]][1:nb];x5<-a[[5]][1:nb];x6<-a[[6]][1:nb]
design <- data.frame(cbind(x1,x2,x3,x4,x5,x6))
names(design)<-c("x1", "x2", "x3", "x4", "x5", "x6")
n <- nrow(design)

response <- data.frame(q=apply(design,1,fonction))
names(response) <- "y"
fitted.model1 <- km(~1, design=design, response=response, covtype="gauss",
control=list(pop.size=50, max.generations=20, wait.generations=5, BFGSburnin=5,
trace=FALSE), optim.method="gen", parinit=rep(0.8,6))

res.nsteps <- EGO.nsteps(model=fitted.model1, fun=fonction, nsteps=nsteps,
lower=rep(0,6), upper=rep(1,6), parinit=rep(0.5,6), control=list(pop.size=50,
max.generations=20, wait.generations=5, BFGSburnin=5), kmcontrol=NULL)
print(res.nsteps)
plot(res.nsteps$value, type="l")

```

---

EI	<i>Analytical expression of the Expected Improvement criterion (noise-free version)</i>
----	---

---

### Description

Computes the Expected Improvement at current location.

### Usage

```
EI(x, model, type="UK", envir=NULL)
```

### Arguments

x	a vector representing the input for which one wishes to calculate EI.
model	an object of class <code>km</code> .
type	"SK" or "UK" (by default), depending whether uncertainty related to trend estimation has to be taken into account.
envir	an optional environment specifying where to assign intermediate values for future gradient calculations. Default is NULL.

### Value

The expected improvement, defined as

$$EI(x) := E[(\min(Y(X)) - Y(x))^+ | Y(X) = y(X)],$$

where X is the current design of experiments and Y is the random process assumed to have generated the objective function y.

### Author(s)

David Ginsbourger (Departement of Mathematics and Statistics, University of Bern, Switzerland)  
Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

### References

D.R. Jones, M. Schonlau, and W.J. Welch (1998), Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, 13, 455-492.  
J. Mockus (1988), *Bayesian Approach to Global Optimization*. Kluwer academic publishers.  
T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.  
M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

**See Also**

[max\\_EI](#), [EGO.nsteps](#), [qEI](#)

**Examples**

```
set.seed(123)
#####
###   EI SURFACE ASSOCIATED WITH AN ORDINARY KRIGING MODEL   ###
###   OF THE BRANIN FUNCTION KNOWN AT A 9-POINTS FACTORIAL DESIGN   ###
#####

# a 9-points factorial design, and the corresponding response
d <- 2; n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.branin <- apply(design.fact, 1, branin)
response.branin <- data.frame(response.branin)
names(response.branin) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.branin,
covtype="gauss", control=list(pop.size=50,trace=FALSE), parinit=c(0.5, 0.5))

# graphics
n.grid <- 30
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
#response.grid <- apply(design.grid, 1, branin)
EI.grid <- apply(design.grid, 1, EI,fitted.model1)
z.grid <- matrix(EI.grid, n.grid, n.grid)
contour(x.grid,y.grid,z.grid,25)
title("Expected Improvement for the Branin function known at 9 points")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
```

---

EI.grad

*Analytical gradient of the Expected Improvement criterion (noise-free version)*

---

**Description**

Computes the gradient of the Expected Improvement at the current location. Only available for Universal Kriging with constant trend (Ordinary Kriging).

**Usage**

```
EI.grad(x, model, envir)
```

**Arguments**

x	a vector representing the input for which one wishes to calculate <a href="#">EI</a> .
model	an object of class <a href="#">km</a> .
envir	an environment specifying where to get intermediate values calculated in <a href="#">EI</a> .

**Value**

The gradient of the expected improvement criterion with respect to x. Returns 0 at design points (where the gradient does not exist).

**Author(s)**

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, Switzerland)  
Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

**References**

- D. Ginsbourger (2009), *Multiplés métamodèles pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://members.unine.ch/david.ginsbourger/recherche/these.htm>
- J. Mockus (1988), *Bayesian Approach to Global Optimization*. Kluwer academic publishers.
- T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.
- M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

**See Also**

[EI](#)

**Examples**

```
set.seed(123)
# a 9-points factorial design, and the corresponding response
d <- 2; n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.branin <- apply(design.fact, 1, branin)
response.branin <- data.frame(response.branin)
names(response.branin) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.branin,
covtype="gauss", control=list(pop.size=50,trace=FALSE), parinit=c(0.5, 0.5))

# graphics
```

```

n.grid <- 50
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
#response.grid <- apply(design.grid, 1, branin)
EI.grid <- apply(design.grid, 1, EI,fitted.model1)
#EI.grid <- apply(design.grid, 1, EI.plot,fitted.model1, gr=TRUE)

z.grid <- matrix(EI.grid, n.grid, n.grid)

contour(x.grid,y.grid,z.grid,20)
title("Expected Improvement for the Branin function known at 9 points")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")

# graphics
n.gridx <- 15
n.gridy <- 20
x.grid2 <- seq(0,1,length=n.gridx)
y.grid2 <- seq(0,1,length=n.gridy)
design.grid2 <- expand.grid(x.grid2, y.grid2)

EI.envir <- new.env()
environment(EI) <- environment(EI.grad) <- EI.envir

options(warn=-1)
for(i in seq(1, nrow(design.grid2)) )
{
  x <- design.grid2[i,]
  ei <- EI(x, model=fitted.model1, envir=EI.envir)
  eigrad <- EI.grad(x , model=fitted.model1, envir=EI.envir)
  if(!is.null(ei))
  {
    arrows(x$Var1,x$Var2,
           x$Var1 + eigrad[1]*2.2*10e-5, x$Var2 + eigrad[2]*2.2*10e-5,
           length = 0.04, code=2, col="orange", lwd=2)
  }
}

```

---

EI.plugin

*Expected Improvement with plugin*


---

### Description

Evaluation of the Expected Improvement criterion, where the current minimum of the observations (y.min) is replaced by an arbitrary value (plugin). Suitable for noisy frameworks.

### Usage

```
EI.plugin(x, model, plugin=NULL, type="UK", envir=NULL)
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
plugin	Plugin for the minimum of the observations.
type	Kriging type: "SK" or "UK"
envir	an optional environment specifying where to assign intermediate values for future gradient calculations. Default is NULL.

**Value**

The Expected Improvement with plug-in, defined as

$$EI(x) := E[(m - Y(x))^+ | Y(X) + \epsilon = \tilde{y}],$$

where X is the current design of experiments, m is the plugin value for the current minimum (here called ymin.plugin), and Y is the random process assumed to have generated the objective function y.

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
#####
###  EI.plugin SURFACE ASSOCIATED WITH AN ORDINARY KRIGING MODEL  #####
###  OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN  #####
#####

set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
```

```

model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size),
           lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
func.grid <- rep(0,1,nt)

for (i in 1:nt)
{
crit.grid[i] <- EI.plugin(x=design.grid[i,], model=model, plugin=1)
func.grid[i] <- test.function(design.grid[i,])
}

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1","V2")
pred <- predict.km(model, newdata=design.grid, type="UK")
mk.grid <- pred$m
sk.grid <- pred$sd

# Plot actual function
z.grid <- matrix(func.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color =topo.colors,
plot.axes = {title("Actual function");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging mean
z.grid <- matrix(mk.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color =topo.colors,
plot.axes = {title("Kriging mean");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging variance
z.grid <- matrix(sk.grid^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color =topo.colors,
plot.axes = {title("Kriging variance");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot EI.plugin criterion
z.grid <- matrix(crit.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color =topo.colors,
plot.axes = {title("EI.plugin");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

```

---

EI.plugin.grad	<i>Analytical gradient of the Expected Improvement criterion with plugin</i>
----------------	--

---

**Description**

Computes the gradient of the Expected Improvement at the current location. Only available for Universal Kriging with constant trend (Ordinary Kriging).

**Usage**

```
EI.plugin.grad(x, model, plugin, type="UK", envir=NULL)
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
plugin	Plugin for the minimum of the observations.
type	Kriging type: "SK" or "UK"
envir	an optional environment specifying where to assign intermediate values for future gradient calculations. Default is NULL.

**Value**

The gradient of the expected improvement criterion with respect to x. Returns 0 at design points (where the gradient does not exist).

**Author(s)**

David Ginsbourger (University of Bern, Switzerland) Victor Picheny (CERFACS, Toulouse, France)

**See Also**

[EI.plugin](#)

**Examples**

```
set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2
```

```

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size), lower=rep(.1,dim), upper=rep(1,dim))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
crit.grad <- matrix(0,nt,2)

EI.envir <- new.env()

for (i in 1:nt)
{
  crit.grid[i] <- EI.plugin(x=design.grid[i,], model=model, plugin=1, envir=EI.envir)
  crit.grad[i,] <- EI.plugin.grad(x=design.grid[i,], model=model, plugin=1, envir=EI.envir)
}

z.grid <- matrix(crit.grid, n.grid, n.grid)
contour(x.grid,y.grid, z.grid, 30)
title("EI.plugin and its gradient")
points(model@X[,1],model@X[,2],pch=17,col="blue")

for (i in 1:nt)
{
  x <- design.grid[i,]
  arrows(x$Var1,x$Var2, x$Var1+crit.grad[i,1]*.03,x$Var2+crit.grad[i,2]*.03, length=0.04,code=2,col="orange",lwd
}

```

---

EQI

*Expected Quantile Improvement*


---

### Description

Evaluation of the Expected Quantile Improvement (EQI) criterion.

### Usage

EQI(x, model, new.noise.var=0, alpha=0.9, q.min=NULL, type = "UK", envir=NULL)

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	(scalar) noise variance of the future observation. Default value is 0 (noise-free observation).
alpha	Quantile level (default value is 0.9)
q.min	Best kriging quantile. If not provided, this quantity is evaluated.
type	Kriging type: "SK" or "UK"
envir	environment for saving intermediate calculations and reusing them within EQI.grad_optim

**Value**

Expected Quantile Improvement

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
#####
###   EQI SURFACE ASSOCIATED WITH AN ORDINARY KRIGING MODEL   ###
### OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN ###
#####

#library("lhs")
set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
            covtype="gauss", noise.var=rep(noise.var,1,doe.size),
```

```

lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
func.grid <- rep(0,1,nt)

for (i in 1:nt)
{
crit.grid[i] <- EQI(x=design.grid[i,], model=model, new.noise.var=noise.var, alpha=0.9)
func.grid[i] <- test.function(design.grid[i,])
}

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1", "V2")
pred <- predict.km(model, newdata=design.grid, type="UK")
mk.grid <- pred$m
sk.grid <- pred$sd

# Plot actual function
z.grid <- matrix(func.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Actual function");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging mean
z.grid <- matrix(mk.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Kriging mean");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging variance
z.grid <- matrix(sk.grid^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Kriging variance");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot EQI criterion
z.grid <- matrix(crit.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("EQI");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

```

**Description**

(Analytical gradient of the Expected Quantile Improvement (EQI) criterion.

**Usage**

```
EQI.grad(x, model, new.noise.var=0, alpha=0.9, q.min=NULL, type = "UK")
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	(scalar) noise variance of the future observation. Default value is 0 (noise-free observation).
alpha	Quantile level (default value is 0.9)
q.min	Best kriging quantile. If not provided, this quantity is evaluated.
type	Kriging type: "SK" or "UK"

**Value**

Gradient of the Expected Quantile Improvement

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
```

```

      covtype="gauss", noise.var=rep(noise.var,1,doe.size),
      lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
#names(design.grid) <- c("V1", "V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
crit.grad <- matrix(0,nt,2)

for (i in 1:nt)
{
  crit.grid[i] <- EQI(x=design.grid[i,], model=model, new.noise.var=noise.var)
  crit.grad[i,] <- EQI.grad(x=design.grid[i,], model=model, new.noise.var=noise.var)
}

z.grid <- matrix(crit.grid, n.grid, n.grid)
contour(x.grid,y.grid, z.grid, 30)
title("EQI and its gradient")
points(model@X[,1],model@X[,2],pch=17,col="blue")

for (i in 1:nt)
{
  x <- design.grid[i,]
  arrows(x$Var1,x$Var2, x$Var1+crit.grad[i,1]*.2,x$Var2+crit.grad[i,2]*.2,
  length=0.04,code=2,col="orange",lwd=2)
}

```

EQI.grad\_optim

*EQI's Gradient, for usage in max\_EQI***Description**

Gradient of the Expected Quantile Improvement (EQI) criterion.

**Usage**

```
EQI.grad_optim(x, model, new.noise.var=0, alpha=0.9, q.min=NULL, type = "UK", envir)
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
new.noise.var	(scalar) noise variance of the future observation. Default value is 0 (noise-free observation).
alpha	Quantile level (default value is 0.9)

q.min	Best kriging quantile. If not provided, this quantity is evaluated.
type	Kriging type: "SK" or "UK"
envir	environment for inheriting intermediate calculations from EQI

**Value**

Gradient of the Expected Quantile Improvement

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (IMSV, University of Bern, Switzerland)

---

<code>kriging.quantile</code>	<i>Kriging quantile</i>
-------------------------------	-------------------------

---

**Description**

Evaluation of a kriging quantile at a new point. To be used in an optimization loop.

**Usage**

```
kriging.quantile(x, model, alpha=0.1, type = "UK", envir=NULL)
```

**Arguments**

x	the input vector at which one wants to evaluate the criterion
model	a Kriging model of "km" class
alpha	Quantile level (default value is 0.1)
type	Kriging type: "SK" or "UK"
envir	an optional environment specifying where to assign intermediate values for future gradient calculations. Default is NULL.

**Value**

Kriging quantile

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

## Examples

```
#####
###   KRIGING QUANTILE SURFACE                               #####
### OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN #####
#####

#library("lhs")
set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
            covtype="gauss", noise.var=rep(noise.var,1,doe.size),
            lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
func.grid <- rep(0,1,nt)

for (i in 1:nt)
{
  crit.grid[i] <- kriging.quantile(x=design.grid[i,], model=model, alpha=0.1)
  func.grid[i] <- test.function(design.grid[i,])
}

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1", "V2")
pred <- predict.km(model, newdata=design.grid, type="UK")
mk.grid <- pred$m
sk.grid <- pred$sd

# Plot actual function
```

```

z.grid <- matrix(func.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Actual function");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging mean
z.grid <- matrix(mk.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Kriging mean");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot Kriging variance
z.grid <- matrix(sk.grid^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("Kriging variance");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

# Plot kriging.quantile criterion
z.grid <- matrix(crit.grid, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title("kriging.quantile");
points(model@X[,1],model@X[,2],pch=17,col="blue");
axis(1); axis(2)})

```

---

kriging.quantile.grad *Analytical gradient of the Kriging quantile of level alpha*

---

## Description

Computes the gradient of the Kriging quantile of level alpha at the current location. Only available for Universal Kriging with constant trend (Ordinary Kriging).

## Usage

```
kriging.quantile.grad(x, model, alpha=0.1, type="UK")
```

## Arguments

x	a vector representing the input for which one wishes to calculate kriging.quantile.grad.
model	an object of class <a href="#">km</a> .
alpha	A quantile level (between 0 and 1)
type	Kriging type: "SK" or "UK"

## Value

The gradient of the Kriging mean predictor with respect to x.

**Author(s)**

David Ginsbourger (IMSV, University of Berne, Switzerland)

**References**

O. Roustant, D. Ginsbourger, Y. Deville, *DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization*, submitted to J. Stat. Soft., 2010. [http://hal.archives-ouvertes.fr/hal-00495766\\_v2/](http://hal.archives-ouvertes.fr/hal-00495766_v2/)

D. Ginsbourger (2009), *Multiplés metamodels pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>

**See Also**

[EI.grad](#)

**Examples**

```
#####
###   KRIGING QUANTILE SURFACE AND ITS GRADIENT FOR   ###
###   THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN   ###
#####

set.seed(421)

# Set test problem parameters
doe.size <- 12
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
  y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)
}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size),
           lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
#names(design.grid) <- c("V1", "V2")
```

```

nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
crit.grad <- matrix(0,nt,2)

for (i in 1:nt)
{
crit.grid[i] <- kriging.quantile(x=design.grid[i,], model=model, alpha=.1)
crit.grad[i,] <- kriging.quantile.grad(x=design.grid[i,], model=model, alpha=.1)
}

z.grid <- matrix(crit.grid, n.grid, n.grid)
contour(x.grid,y.grid, z.grid, 30)
title("kriging.quantile and its gradient")
points(model@X[,1],model@X[,2],pch=17,col="blue")

for (i in 1:nt)
{
x <- design.grid[i,]
arrows(x$Var1,x$Var2, x$Var1+crit.grad[i,1]*.01,x$Var2+crit.grad[i,2]*.01,
length=0.04,code=2,col="orange",lwd=2)
}

```

---

```
kriging.quantile.grad_optim
```

*Analytical gradient of the Kriging quantile of level alpha*

---

### Description

Computes the gradient of the Kriging quantile of level alpha at the current location. Only available for Universal Kriging with constant trend (Ordinary Kriging).

### Usage

```
kriging.quantile.grad_optim(x, model, alpha=0.1, type="UK", envir)
```

### Arguments

x	a vector representing the input for which one wishes to calculate <code>kriging.quantile.grad_optim</code> .
model	an object of class <code>km</code> .
alpha	A quantile level (between 0 and 1)
type	Kriging type: "SK" or "UK"
envir	an environment specifying where to get intermediate values calculated in <code>kriging.quantile</code> .

### Value

The gradient of the Kriging mean predictor with respect to x.

**Author(s)**

David Ginsbourger (IMSV, University of Bern, Switzerland)

**References**

O. Roustant, D. Ginsbourger, Y. Deville, *DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization*, submitted to J. Stat. Soft., 2010. [http://hal.archives-ouvertes.fr/hal-00495766\\_v2/](http://hal.archives-ouvertes.fr/hal-00495766_v2/)

D. Ginsbourger (2009), *Multiplés metamodels pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>

**See Also**

[EI.grad](#)

**Examples**

```
##### 1-D example with noisy observations #####

fundet <- function (x) {
  return((sin(10 * x)/(1 + x) + 2 * cos(5 * x) * x^3 + 0.841)/1.6)}
theta <- 1/sqrt(30)
p <- 2
x <- c(0,0.15,0.40,0.9)
n <- length(x)
t <- seq(0,1,by=0.01)
covtype="gauss"#"matern5_2"

#####

noise.var <- c(0.1,0.01,0.3,0.1)^2
z <- fundet(x);
y <- z + sqrt(noise.var)*rnorm(length(x))
model <- km(y~1, design=data.frame(x=x), response=
data.frame(y=y), coef.trend=0, covtype=covtype,
coef.cov=theta, coef.var=1, noise.var=noise.var,
control=list(trace=FALSE))
p <- predict.km(model, newdata=t, type="SK")

par(mfrow=c(2,1))
lower <- p$lower95; upper <- p$upper95
plot(t, p$mean, type="l", ylim=c(1.1*min(c(lower,y)) ,
1.5*max(c(upper,y))), xlab="x", ylab="y", col="blue", lwd=1.5)
polygon(c(t,rev(t)), c(lower, rev(upper)), col=gray(0.9),
border = gray(0.9))
lines(t, p$mean, type="l", ylim=c(min(lower) ,max(upper)),
xlab="x", ylab="y", col="blue", lwd=1)
lines(t, lower , col="blue", lty=4, lwd=1.7);
lines(t, upper , col="blue", lty=4, lwd=1.7)
```

```

lines(t,fundet(t),col="black",lwd=2);
points(x, y, pch=8,col="blue", cex=2, lwd=2)
segments(x, y-1.96*sqrt(noise.var), x,
y+1.96*sqrt(noise.var), col= 'red', lwd=2)
segments(x-0.2/n, y-1.96*sqrt(noise.var), x+0.2/n,
y-1.96*sqrt(noise.var), col= 'red', lwd=2)
segments(x-0.2/n, y+1.96*sqrt(noise.var), x+0.2/n,
y+1.96*sqrt(noise.var), col= 'red', lwd=2)

#####

plot(t, p$upper, type="l", xlab="x", ylab="y",col="blue", lwd=1.5)

quantile.envir <- new.env()
environment(kriging.quantile) <- environment(kriging.quantile.grad_optim) <- quantile.envir

q <- matrix("NA", ncol=1, nrow=length(t))
gq <- matrix("NA", ncol=1, nrow=length(t))
for(i in seq(1,length(t))){
q[i] <- kriging.quantile(t[i], model, alpha=0.95, envir=quantile.envir)
gq[i] <- kriging.quantile.grad_optim(t[i], model, alpha=0.95,
envir=quantile.envir)
}

plot(t, gq, type="l", xlab="x", ylab="y",col="blue", lwd=2)
abline(h=0, col="red")

#####

```

---

max_AEI	<i>Maximizer of the Augmented Expected Improvement criterion function</i>
---------	---

---

### Description

Maximization, based on the package rgenoud of the Augmented Expected Improvement (AEI) criterion.

### Usage

```
max_AEI(model, new.noise.var=0, y.min=NULL, type = "UK", lower, upper, parinit=NULL, control=NULL)
```

### Arguments

model	a Kriging model of "km" class
new.noise.var	the (scalar) noise variance of the new observation.
y.min	The kriging mean prediction at the current best point (point with smallest kriging quantile). If not provided, this quantity is evaluated inside the AEI function (may increase computational time).

type	Kriging type: "SK" or "UK"
lower	vector containing the lower bounds of the variables to be optimized over
upper	vector containing the upper bounds of the variables to be optimized over
parinit	optional vector containing the initial values for the variables to be optimized over
control	optional list of control parameters for optimization. One can control "pop.size" (default: $[4+3*\log(\text{nb of variables})]$ ), "max.generations" (5), "wait.generations" (2) and "BFGSburnin" (0) of function "genoud" (see <a href="#">genoud</a> ). Numbers into brackets are the default values

### Value

A list with components:

par	the best set of parameters found.
value	the value AEI at par.

### Author(s)

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

### Examples

```
set.seed(10)

# Set test problem parameters
doe.size <- 10
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {y.tilde[i] <- test.function(doe[i,])
+ sqrt(noise.var)*rnorm(n=1)}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
  covtype="gauss", noise.var=rep(noise.var,1,doe.size),
  lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Optimisation using max_AEI
res <- max_AEI(model, new.noise.var=noise.var, type = "UK",
  lower=c(0,0), upper=c(1,1))
X.genoud <- res$par
```

```

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
names(design.grid) <- c("V1", "V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)

for (i in 1:nt)
{
crit.grid[i] <- AEI(x=design.grid[i,], model=model,
new.noise.var=noise.var)
}

# # 2D plots
z.grid <- matrix(crit.grid, n.grid, n.grid)
tit <- "Green: best point found by optimizer"
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title(tit);points(model@X[,1],model@X[,2],pch=17,col="blue");
points(X.genoud[1],X.genoud[2],pch=17,col="green");
axis(1); axis(2)})

```

---

max\_AKG

*Maximizer of the Expected Quantile Improvement criterion function*


---

## Description

Maximization, based on the package rgenoud of the Expected Quantile Improvement (AKG) criterion.

## Usage

```
max_AKG(model, new.noise.var=0, type = "UK", lower, upper, parinit=NULL, control=NULL)
```

## Arguments

model	a Kriging model of "km" class
new.noise.var	the (scalar) noise variance of an observation. Default value is 0 (noise-free observation).
type	Kriging type: "SK" or "UK"
lower	vector containing the lower bounds of the variables to be optimized over
upper	vector containing the upper bounds of the variables to be optimized over
parinit	optional vector containing the initial values for the variables to be optimized over
control	optional list of control parameters for optimization. One can control "pop.size" (default : $[N=3*2^{\dim}$ for $\dim < 6$ and $N=32*\dim$ otherwise]), "max.generations" (10), "wait.generations" (2) and "BFGSburnin" (0) of function "genoud" (see <a href="#">genoud</a> ). Numbers into brackets are the default values

**Value**

A list with components:

par	the best set of parameters found.
value	the value AKG at par.

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
#####
###   AKG SURFACE AND OPTIMIZATION PERFORMED BY GENOUD           ###
###   FOR AN ORDINARY KRIGING MODEL                             ###
###   OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN #####
#####
set.seed(10)

# Set test problem parameters
doe.size <- 10
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {y.tilde[i] <- test.function(doe[i,])
+ sqrt(noise.var)*rnorm(n=1)}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size),
           lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Optimisation using max_AKG
res <- max_AKG(model, new.noise.var=noise.var, type = "UK",
              lower=c(0,0), upper=c(1,1))
X.genoud <- res$par

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
names(design.grid) <- c("V1","V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)
```

```

for (i in 1:nt)
{
crit.grid[i] <- AKG(x=design.grid[i,], model=model,
new.noise.var=noise.var)
}

# # 2D plots
z.grid <- matrix(crit.grid, n.grid, n.grid)
tit <- "Green: best point found by optimizer"
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title(tit);points(model@X[,1],model@X[,2],pch=17,col="blue");
points(X.genoud[1],X.genoud[2],pch=17,col="green");
axis(1); axis(2)})

```

---

max\_EI

*One-shot maximization of the Expected Improvement criterion (noise-free version)*


---

## Description

Given an object of class `km` and a set of tuning parameters (`lower`, `upper`, `parinit`, and `control`), `max_EI` performs the maximization of the Expected Improvement criterion and delivers the next point to be visited in an EGO-like procedure.

The latter maximization relies on a genetic algorithm using derivatives, `genoud`. This function plays a central role in the package since it is in constant use in the proposed algorithms. It is important to remark that the information needed about the objective function reduces here to the vector of response values embedded in `model` (no call to the objective function or simulator).

## Usage

```
max_EI(model, lower, upper, parinit=NULL, control = NULL)
```

## Arguments

<code>model</code>	an object of class <code>km</code> ,
<code>lower</code>	vector of lower bounds for the variables to be optimized over,
<code>upper</code>	vector of upper bounds for the variables to be optimized over,
<code>parinit</code>	optional vector of initial values for the variables to be optimized over,
<code>control</code>	an optional list of control parameters for optimization. One can control "pop.size" (default : $[4+3*\log(\text{nb of variables})]$ ), "max.generations" (default :5), "wait.generations" (default :2), "BFGSburnin" (default :0), of the function <code>genoud</code> .

**Value**

A list with components:

par	The best set of parameters found.
value	The value of expected improvement at par.

**Note**

Most EGO-like methods (EI algorithms) usually work with Ordinary Kriging (constant trend), by maximization of the expected improvement. Here, the EI maximization is also possible with any linear trend. However, note that the optimization may perform much faster and better when the trend is a constant since it is the only case where the analytical gradient is available.

**Author(s)**

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, Switzerland)  
Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

**References**

D. Ginsbourger (2009), *Multiplés metamodels pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>

D.R. Jones, M. Schonlau, and W.J. Welch (1998), Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, 13, 455-492.

W.R. Jr. Mebane and J.S. Sekhon (2009), in press, Genetic optimization using derivatives: The rgenoud package for R, *Journal of Statistical Software*.

**Examples**

```
set.seed(123)
#####
### "ONE-SHOT" EI-MAXIMIZATION OF THE BRANIN FUNCTION ###
### KNOWN AT A 9-POINTS FACTORIAL DESIGN          #####
#####

# a 9-points factorial design, and the corresponding response
d <- 2
n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact) <- c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact) <- c("x1", "x2")
response.branin <- apply(design.fact, 1, branin)
response.branin <- data.frame(response.branin)
names(response.branin) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.branin,
```

```

covtype="gauss", control=list(pop.size=50,trace=FALSE), parinit=c(0.5, 0.5))

# EGO one step
library(rgenoud)
lower <- rep(0,d)
upper <- rep(1,d)      # domain for Branin function
oEGO <- max_EI(fitted.model1, lower, upper,
control=list(pop.size=20, BFGSburnin=2))
print(oEGO)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, branin)
z.grid <- matrix(response.grid, n.grid, n.grid)
contour(x.grid,y.grid,z.grid,40)
title("Branin Function")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGO$par[1], oEGO$par[2], pch=19, col="red")

#####
### "ONE-SHOT" EI-MAXIMIZATION OF THE CAMELBACK FUNCTION ###
### KNOWN AT A 16-POINTS FACTORIAL DESIGN          #####
#####

# a 16-points factorial design, and the corresponding response
d <- 2
n <- 16
design.fact <- expand.grid(seq(0,1,length=4), seq(0,1,length=4))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact) <- c("x1", "x2")
response.camelback <- apply(design.fact, 1, camelback)
response.camelback <- data.frame(response.camelback)
names(response.camelback) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.camelback,
covtype="gauss", control=list(pop.size=50,trace=FALSE), parinit=c(0.5, 0.5))

# EI maximization
library(rgenoud)
lower <- rep(0,d)
upper <- rep(1,d)
oEGO <- max_EI(fitted.model1, lower, upper,
control=list(pop.size=20, BFGSburnin=2))
print(oEGO)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)

```

```

design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, camelback)
z.grid <- matrix(response.grid, n.grid, n.grid)
contour(x.grid,y.grid,z.grid,40)
title("Camelback Function")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGO$par[1], oEGO$par[2], pch=19, col="red")

#####
### "ONE-SHOT" EI-MAXIMIZATION OF THE GOLDSTEIN-PRICE FUNCTION #####
###      KNOWN AT A 9-POINTS FACTORIAL DESIGN          #####
#####

# a 9-points factorial design, and the corresponding response
d <- 2
n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.goldsteinPrice <- apply(design.fact, 1, goldsteinPrice)
response.goldsteinPrice <- data.frame(response.goldsteinPrice)
names(response.goldsteinPrice) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.goldsteinPrice,
covtype="gauss", control=list(pop.size=50, max.generations=50,
wait.generations=5, BFGSburnin=10, trace=FALSE), parinit=c(0.5, 0.5), optim.method="gen")

# EI maximization
library(rgenoud)
lower <- rep(0,d); upper <- rep(1,d);      # domain for Branin function
oEGO <- max_EI(fitted.model1, lower, upper, control
=list(pop.size=50, max.generations=50, wait.generations=5, BFGSburnin=10))
print(oEGO)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, goldsteinPrice)
z.grid <- matrix(response.grid, n.grid, n.grid)
contour(x.grid,y.grid,z.grid,40)
title("Goldstein-Price Function")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGO$par[1], oEGO$par[2], pch=19, col="red")

```

---

max\_EI.plugin                    *Maximizer of the Expected Improvement criterion function with plugin of the minimum.*

---

### Description

Maximization, based on the package `rgenoud` of the Expected Improvement (EI) criterion with plugin of the minimum.

### Usage

```
max_EI.plugin(model, plugin, type = "UK", lower, upper, parinit=NULL, control=NULL)
```

### Arguments

<code>model</code>	a Kriging model of "km" class
<code>plugin</code>	Plugin for the current minimum of the observations.
<code>type</code>	Kriging type: "SK" or "UK"
<code>lower</code>	vector containing the lower bounds of the variables to be optimized over
<code>upper</code>	vector containing the upper bounds of the variables to be optimized over
<code>parinit</code>	optional vector containing the initial values for the variables to be optimized over
<code>control</code>	optional list of control parameters for optimization. One can control "pop.size" (default: $[4+3*\log(\text{nb of variables})]$ ), "max.generations" (5), "wait.generations" (2) and "BFGSburnin" (0) of function "genoud" (see <a href="#">genoud</a> ). Numbers into brackets are the default values

### Value

A list with components:

<code>par</code>	the best set of parameters found.
<code>value</code>	the value EI at par.

### Author(s)

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

### Examples

```
set.seed(10)

# Set test problem parameters
doe.size <- 10
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
```

```

upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {
y.tilde[i] <- test.function(doe[i,]) + sqrt(noise.var)*rnorm(n=1)}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size),
           lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Optimisation using max_AEI
res <- max_AEI(model, new.noise.var=noise.var, type = "UK",
              lower=c(0,0), upper=c(1,1))
X.genoud <- res$par

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
names(design.grid) <- c("V1","V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)

for (i in 1:nt)
{
  crit.grid[i] <- AEI(x=design.grid[i,], model=model, new.noise.var=noise.var)
}

# # 2D plots
z.grid <- matrix(crit.grid, n.grid, n.grid)
tit <- c("Green: best point found by optimizer")
tit <- paste(tit, collapse="")
# tit <- "; black: best on grid"
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
              plot.axes = {title(tit);points(model@X[,1],model@X[,2],pch=17,col="blue");
              points(X.genoud[1],X.genoud[2],pch=17,col="green");
              axis(1); axis(2)})

```

---

max\_EQI

*Maximizer of the Expected Quantile Improvement criterion function*


---

### Description

Maximization, based on the package rgenoud of the Expected Quantile Improvement (EQI) criterion.

**Usage**

```
max_EQI(model, new.noise.var=0, alpha=0.9, q.min=NULL, type = "UK", lower, upper, parinit=NULL, control)
```

**Arguments**

model	a Kriging model of "km" class
new.noise.var	the (scalar) noise variance of an observation. Default value is 0 (noise-free observation).
alpha	Quantile level (default value is 0.9)
q.min	The current best kriging quantile. If not provided, this quantity is evaluated inside the EQI function (may increase computational time).
type	Kriging type: "SK" or "UK"
lower	vector containing the lower bounds of the variables to be optimized over
upper	vector containing the upper bounds of the variables to be optimized over
parinit	optional vector containing the initial values for the variables to be optimized over
control	optional list of control parameters for optimization. One can control "pop.size" (default : $[N=3*2^{\dim}$ for $\dim < 6$ and $N=32*\dim$ otherwise]), "max.generations" (10), "wait.generations" (2) and "BFGSburnin" (0) of function "genoud" (see <a href="#">genoud</a> ). Numbers into brackets are the default values

**Value**

A list with components:

par	the best set of parameters found.
value	the value EQI at par.

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
set.seed(10)

# Set test problem parameters
doe.size <- 10
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
```

```

y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {y.tilde[i] <- test.function(doe[i,])
+ sqrt(noise.var)*rnorm(n=1)}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
  covtype="gauss", noise.var=rep(noise.var,1,doe.size),
  lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Optimisation using max_EQI
res <- max_EQI(model, new.noise.var=noise.var, type = "UK",
lower=c(0,0), upper=c(1,1))
X.genoud <- res$par

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
names(design.grid) <- c("V1","V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)

for (i in 1:nt)
{
crit.grid[i] <- EQI(x=design.grid[i,], model=model,
new.noise.var=noise.var)
}

# # 2D plots
z.grid <- matrix(crit.grid, n.grid, n.grid)
tit <- "Green: best point found by optimizer"
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = rainbow,
plot.axes = {title(tit);points(model@X[,1],model@X[,2],pch=17,col="blue");
points(X.genoud[1],X.genoud[2],pch=17,col="green");
axis(1); axis(2)})

```

---

max\_qEI.CL

*One-shot pseudo-maximization of qEI using the Constant Liar strategy*


---

## Description

Approached maximization of the q-points Expected Improvement criterion **qEI** using the "constant liar" heuristic. First, the regular Expected Improvement **EI** is maximized. Then, for the next points, the Expected Improvement is maximized again, but with an artificially updated Kriging model. Since the response values corresponding to the last best point obtained are not available, the idea of CL is to replace them by an arbitrary constant value L (a "lie") set by the user.

**Usage**

```
max_qEI.CL(model, npoints, L, lower, upper, parinit=NULL, control=NULL)
```

**Arguments**

model	an object of class <code>km</code> ,
npoints	an integer representing the desired number of iterations,
L	a real number equal to the "constant liar" value. Typically, L is fixed to the smaller observed function value.
lower	vector of lower bounds for the variables to be optimized over,
upper	vector of upper bounds for the variables to be optimized over,
parinit	optional vector of initial values for the variables to be optimized over,
control	an optional list of control parameters for optimization. One can control "pop.size" (default : [4+3*log(nb of variables)]), "max.generations" (default :5), "wait.generations" (default :2), "BFGSburnin" (default :0), of the function <code>genoud</code> .

**Value**

A list with components:

par	A matrix containing the npoints input vectors found.
value	A vector giving the npoints values of artificial responses (all equal to L here).

**Author(s)**

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, Switzerland)  
Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

**References**

- D. Ginsbourger (2009), *Multiplés metamodeles pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>
- D. Ginsbourger, R. Le Riche, and L. Carraro (2009), chapter "Kriging is well-suited to parallelize optimization", to appear in *Computational Intelligence in Expensive Optimization Problems*, Studies in Evolutionary Learning and Optimization, Springer.
- W.R. Jr. Mebane and J.S. Sekhon (2009), in press, Genetic optimization using derivatives: The rgenoud package for R, *Journal of Statistical Software*.
- T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.
- M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

**See Also**

[CL.nsteps](#), [EI](#), [max\\_EI](#)

**Examples**

```

set.seed(123)
# a 9-points factorial design, and the corresponding response
d <- 2
n <- 9
design.fact <- expand.grid(seq(0,1,length=3), seq(0,1,length=3))
names(design.fact)<-c("x1", "x2")
design.fact <- data.frame(design.fact)
names(design.fact)<-c("x1", "x2")
response.branin <- apply(design.fact, 1, branin)
response.branin <- data.frame(response.branin)
names(response.branin) <- "y"

# model identification
fitted.model1 <- km(~1, design=design.fact, response=response.branin,
covtype="gauss", control=list(pop.size=50, trace=FALSE), parinit=c(0.5, 0.5))

# EGO parallel
library(rgenoud)
npoints <- 10
lower <- rep(0,d)
upper <- rep(1,d)
oEGOpallel <- max_qEI.CL(model=fitted.model1, npoints=npoints,
L=min(response.branin), lower, upper,
control=list(pop.size=20, BFGSburnin=2))
print(oEGOpallel)

# graphics
n.grid <- 20
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
response.grid <- apply(design.grid, 1, branin)
z.grid <- matrix(response.grid, n.grid, n.grid)
contour(x.grid,y.grid,z.grid,40)
title("Branin function")
points(design.fact[,1], design.fact[,2], pch=17, col="blue")
points(oEGOpallel$par, pch=19, col="red")
text(oEGOpallel$par[,1], oEGOpallel$par[,2], labels=1:npoints, pos=3)

```

---

min\_quantile

*Minimizer of the Kriging quantile.*


---

**Description**

Minimization, based on the package rgenoud of the kriging quantile.

**Usage**

```
min_quantile(model, alpha=0.1, type = "UK", lower, upper, parinit=NULL, control=NULL)
```

**Arguments**

model	a Kriging model of "km" class
alpha	Quantile level (default value is 0.1)
type	Kriging type: "SK" or "UK"
lower	vector containing the lower bounds of the variables to be optimized over
upper	vector containing the upper bounds of the variables to be optimized over
parinit	optional vector containing the initial values for the variables to be optimized over
control	optional list of control parameters for optimization. One can control "pop.size" (default : [4+3*log(nb of variables)]), "max.generations" (5), "wait.generations" (2) and "BFGSburnin" (0) of function "genoud" (see <a href="#">genoud</a> ). Numbers into brackets are the default values

**Value**

A list with components:

par	the best set of parameters found.
value	the value of the krigign quantile at par.

**Author(s)**

Victor Picheny (CERFACS, Toulouse, France) David Ginsbourger (University of Bern, Switzerland)

**Examples**

```
#####
###   KRIGING QUANTILE SURFACE AND OPTIMIZATION PERFORMED BY GENOUD   ###
###   FOR AN ORDINARY KRIGING MODEL                                   #####
###   OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN #####
#####
set.seed(10)

# Set test problem parameters
doe.size <- 10
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.2

# Generate DOE and response
```

```

doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- rep(0, 1, doe.size)
for (i in 1:doe.size) {y.tilde[i] <- test.function(doe[i,])
+ sqrt(noise.var)*rnorm(n=1)}
y.tilde <- as.numeric(y.tilde)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
  covtype="gauss", noise.var=rep(noise.var,1,doe.size),
  lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Optimisation using max_kriging.quantile
res <- min_quantile(model, alpha=0.1, type = "UK", lower=c(0,0), upper=c(1,1))
X.genoud <- res$par

# Compute actual function and criterion on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
names(design.grid) <- c("V1","V2")
nt <- nrow(design.grid)
crit.grid <- rep(0,1,nt)

for (i in 1:nt)
{
crit.grid[i] <- kriging.quantile(x=design.grid[i,], model=model,alpha=0.1)
}

# # 2D plots
z.grid <- matrix(crit.grid, n.grid, n.grid)
tit <- "Green: best point found by optimizer"
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title(tit);points(model@X[,1],model@X[,2],pch=17,col="blue");
points(X.genoud[1],X.genoud[2],pch=17,col="green");
axis(1); axis(2)})

```

---

noisy.optimizer

*Optimization of homogeneously noisy functions based on Kriging*


---

## Description

Sequential optimization of kriging-based criterion conditional on noisy observations, with model update after each evaluation. Eight criteria are proposed to choose the next observation: random search, sequential parameter optimization (SPO), reinterpolation, Expected Improvement (EI) with plugin, Expected Quantile Improvement (EQI), quantile minimization, Augmented Expected Improvement (AEI) and Approximate Knowledge Gradient (AKG). The criterion optimization is based on the package rgenoud.

**Usage**

```
noisy.optimizer(optim.crit, optim.param=NULL, model, n.ite, noise.var=NULL, funnoise, lower, upper, pa
               noiseReEstimate=FALSE, nugget.LB=1e-5, obs.n.rep=NULL, estim.model=NULL)
```

**Arguments**

optim.crit	String defining the criterion to be optimized at each iteration. Possible values are: "random.search", "SPO", "reinterpolation", "EI.plugin", "EQI", "min.quantile", "AEI", "AKG".
optim.param	List of parameters for the chosen criterion. For "SPO": optim.param\$nrep is a vector of integers representing the number of repetitions for each observation; optim.param\$history.best is a vector of the indices (integers) of the best observations. If not provided, nrep is set to a vector of ones and history.best is set to the index of the current best observation. For "EI.plugin": optim.param\$plugin.type is a string defining which plugin is to be used. Possible values are "ytilde", "quantile" and "other". If "quantile" is chosen, optim.param\$quantile defines the quantile level. If "other" is chosen, optim.param\$plugin directly sets the plugin value. For "EQI": optim.param\$quantile defines the quantile level. If not provided, default value is 0.9. For "min.quantile": optim.param\$quantile defines the quantile level. If not provided, default value is 0.1. For "AEI": optim.param\$quantile defines the quantile level to choose the current best point. If not provided, default value is 0.75.
model	a Kriging model of "km" class
n.ite	Number of iterations
noise.var	Noise variance (scalar). If noiseReEstimate=TRUE, it is an initial guess for the unknown variance (used in optimization).
funnoise	objective (noisy) function
lower	vector containing the lower bounds of the variables to be optimized over
upper	vector containing the upper bounds of the variables to be optimized over
parinit	optional vector of initial values for the variables to be optimized over
control	optional list of control parameters for optimization. One can control "pop.size" (default : [N=3*2^dim for dim<6 and N=32*dim otherwise]), "max.generations" (N), "wait.generations" (2) and "BFGSburnin" (0) of function "genoud" (see <a href="#">genoud</a> ). Numbers into brackets are the default values
CovReEstimate	optional boolean specifying if the covariance parameters should be re-estimated at every iteration (default value = TRUE)
noiseReEstimate	optional boolean specifying if the noise variance should be re-estimated at every iteration (default value = FALSE)
obs.n.rep	optional vector containing the number of repetitions per observation point. Required if "model" has heterogeneous variances and noise is reestimated. In that case noise variances must be of the form $\sigma^2/\text{obs.n.rep}[i]$ .

nugget.LB	optional scalar of minimal value for the estimated noise variance. Default value is 1e-5.
estim.model	optional kriging model of "km" class with homogeneous nugget effect (no noise.var). Required if noise variance is reestimated and the initial "model" has heterogeneous noise variances.

### Value

A list with components:

par	The added observations
value	the added observation values
nsteps	the number of added observations
lastmodel	the current (last) kriging model of "km" class
estim.model	If noiseReEstimate=TRUE, the current (last) kriging model of "km" class for estimating the noise variance.

### Author(s)

Victor Picheny (CERFACS, Toulouse, France)

### Examples

```
#####
### EXAMPLE 1: 5 OPTIMIZATION STEPS USING EQI WITH KNOWN NOISE      ###
### AND KNOWN COVARIANCE PARAMETERS FOR THE BRANIN FUNCTION      ###
#####

set.seed(10)

# Set test problem parameters
doe.size <- 9
dim <- 2
test.function <- get("branin2")
lower <- rep(0,1,dim)
upper <- rep(1,1,dim)
noise.var <- 0.1

# Build noisy simulator
funnoise <- function(x)
{   f.new <- test.function(x) + sqrt(noise.var)*rnorm(n=1)
    return(f.new)}

# Generate DOE and response
doe <- as.data.frame(optimumLHS(n=doe.size, k=dim))
y.tilde <- funnoise(doe)

# Create kriging model
model <- km(y~1, design=doe, response=data.frame(y=y.tilde),
           covtype="gauss", noise.var=rep(noise.var,1,doe.size),
```

```

    lower=rep(.1,dim), upper=rep(1,dim), control=list(trace=FALSE))

# Optimisation with noisy.optimizer
n.ite <- 15
optim.param <- list()
optim.param$quantile <- .9
optim.result <- noisy.optimizer(optim.crit="EQI", optim.param=optim.param, model=model,
n.ite=n.ite, noise.var=noise.var, funnoise=funnoise, lower=lower, upper=upper,
noiseReEstimate=FALSE, CovReEstimate=FALSE)

new.model <- optim.result$model
new.doe <- optim.result$X
new.y <- optim.result$y

##### DRAW RESULTS #####
# Compute actual function on a grid
n.grid <- 21
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
names(design.grid) <- c("V1","V2")
nt <- nrow(design.grid)
func.grid <- rep(0,1,nt)

for (i in 1:nt)
{ func.grid[i] <- test.function(x=design.grid[i,])}

# Compute initial and final kriging on a grid
pred <- predict.km(object=model, newdata=design.grid, type="UK")
mk.grid1 <- pred$m
sk.grid1 <- pred$sd

pred <- predict.km(object=new.model, newdata=design.grid, type="UK")
mk.grid2 <- pred$m
sk.grid2 <- pred$sd

# Plot initial kriging mean
z.grid <- matrix(mk.grid1, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Initial kriging mean");
points(model@X[,1],model@X[,2],pch=17,col="black");
axis(1); axis(2)})

# Plot initial kriging variance
z.grid <- matrix(sk.grid1^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Initial kriging variance");
points(model@X[,1],model@X[,2],pch=17,col="black");
axis(1); axis(2)})

# Plot final kriging mean
z.grid <- matrix(mk.grid2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Final kriging mean");

```

```

points(new.model@X[,1],new.model@X[,2],pch=17,col="black");
axis(1); axis(2))

# Plot final kriging variance
z.grid <- matrix(sk.grid2^2, n.grid, n.grid)
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title("Final kriging variance");
points(new.model@X[,1],new.model@X[,2],pch=17,col="black");
axis(1); axis(2)})

# Plot actual function and observations
z.grid <- matrix(func.grid, n.grid, n.grid)
tit <- "Actual function - Black: initial points; red: added points"
filled.contour(x.grid,y.grid, z.grid, nlevels=50, color = topo.colors,
plot.axes = {title(tit);points(model@X[,1],model@X[,2],pch=17,col="black");
points(new.doe[1,],new.doe[2,],pch=15,col="red");
axis(1); axis(2)})

```

---

qEI	<i>Monte-Carlo estimation of the multipoints Expected Improvement criterion (noise-free version)</i>
-----	--

---

## Description

Computes the Expected Improvement at any vector of new locations.

## Usage

```
qEI(newdata, model, type="UK", MC.samples=10000, return.I=FALSE)
```

## Arguments

newdata	a matrix representing the set of input vectors (columns) where to estimate qEI.
model	an object of class km.
type	"SK" or "UK", distinguishing the cases of known or unknown trend parameters.
MC.samples	integer, size of the Monte-Carlo sample to be used within the estimation of qEI.
return.I	logical, standing for returning improvements.matrix in the list of outputs or not

## Value

The multipoints Expected Improvement, defined as

$$EI(X_{new}) := E[(\min(Y(X)) - \min(Y(X_{new})))^+ | Y(X) = y(X)],$$

where  $X$  is the current design of experiments,  $X_{new}$  is a new candidate design, and  $Y$  is a random process assumed to have generated the objective function  $y$ .

**Author(s)**

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, Switzerland)  
 Olivier Roustant (Ecole Nationale Supérieure des Mines de Saint-Etienne, France).

**References**

D. Ginsbourger (2009), *Multiplés metamodels pour l'approximation et l'optimisation de fonctions numériques multivariées*, Ph.D. thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009. <http://www.ginsbourger.ch/recherche/these.php>

D. Ginsbourger, R. Le Riche, and L. Carraro (2009), chapter "Kriging is well-suited to parallelize optimization", to appear in *Computational Intelligence in Expensive Optimization Problems*, Studies in Evolutionary Learning and Optimization, Springer.

J. Mockus (1988), *Bayesian Approach to Global Optimization*. Kluwer academic publishers.

B.D. Ripley (1987), *Stochastic Simulation*, Wiley.

T.J. Santner, B.J. Williams, and W.J. Notz (2003), *The design and analysis of computer experiments*, Springer.

M. Schonlau (1997), *Computer experiments and global optimization*, Ph.D. thesis, University of Waterloo.

**See Also**

[max\\_qEI.CL](#), [CL.nsteps](#)

**Examples**

```
set.seed(123)
# -----
# A 1D example with known parameters
# -----

x <- c(0, 0.4, 0.6, 0.8, 1)
y <- 10*c(-0.6, 0, -2, 0.5, 0.9)

theta <- 0.1
sigma <- 10
trend <- 5*c(-2,1)
model <- km(~x,design=data.frame(x=x), response=data.frame(y=y),
  coef.trend=trend, covtype="gauss", coef.cov=theta, coef.var=sigma^2, control <- list(trace=FALSE))

## Kriging with gaussian covariance, and linear trend t(x) = -10 + 5x
type="SK"
t <- seq(from=0, to=1, by=0.005)
p <- predict(model, newdata=t, type=type)
plot(t, p$mean, type="l", ylim=c(min(p$lower95),max(p$upper95)),
  xlab="x", ylab="y")
lines(t, p$lower95, col="black", lty=2)
lines(t, p$upper95, col="black", lty=2)
points(x, y, col="red", pch=19)
abline(h=0)
```

```

## A first try of qEI
candidate.design <- seq(0,1,,101)
res <- qEI(newdata=candidate.design, model=model, type=type,
MC.samples=10000, return.I=TRUE)

## One_point EI can easily be estimated on the basis of the matrix of improvements
EI_simples <- colMeans(res$I)
plot(candidate.design,EI_simples, type="l")

## Let us check that the analytical formula of EI and the MC estimate match
EI_analytical <- apply(as.data.frame(candidate.design), 1, EI, model, type=type)
plot(EI_simples, EI_analytical)

# Computation of the 2-points EI grid associated to candidate.design
# (Straightforward computation method: faster not available in this version yet)

two_points_EI <- matrix(0,ncol=length(candidate.design),
nrow=length(candidate.design))

for(i in seq(1,length(candidate.design)) ){
for(j in seq(i,length(candidate.design)) ){
qI <- pmax(res$I[,i],res$I[,j])
two_points_EI[i,j] <- mean(qI)
two_points_EI[j,i] <- two_points_EI[i,j]
}}

# Plotting the 2_points EI as a function of both points,
# based on the latter qEI estimation

contour(two_points_EI)

```

---

update\_km

*Update of a Kriging model when adding new observation*


---

### Description

Update of a Kriging model when adding new observation, with or without covariance parameter re-estimation.

### Usage

```
update_km(model, NewX, NewY, NewX_AllreadyExist=FALSE,
CovReEstimate=TRUE,new.noise.var=NULL,kmcontrol=NULL)
```

### Arguments

model	a Kriging model of "km" class
NewX	a matrix containing the new points of experiments

NewY                    a matrix containing the function values on the points NewX  
NewX\_AllreadyExist    a boolean: do the NewX points already exist in the design of experiment model@X  
                          ?  
CovReEstimate        optional boolean specifying if the covariance parameters should be re-estimated  
                          at every iteration (default value = TRUE)  
new.noise.var        a vector containing the noise variance at each new observations  
kmcontrol            kmcontrol (to develop)

**Value**

Updated Kriging model

**Author(s)**

Clement Chevalier, University of Bern (IMSV) Victor Picheny, Ecole Centrale Paris

# Index

## \*Topic **models**

AEI, [4](#)  
EI, [20](#)  
EI.grad, [21](#)  
EI.plugin.grad, [26](#)  
kriging.quantile.grad, [34](#)  
kriging.quantile.grad\_optim, [36](#)  
qEI, [57](#)

## \*Topic **optimize**

CL.nsteps, [13](#)  
EGO.nsteps, [16](#)  
EI.grad, [21](#)  
EI.plugin.grad, [26](#)  
kriging.quantile.grad, [34](#)  
kriging.quantile.grad\_optim, [36](#)  
max\_EI, [42](#)  
max\_qEI.CL, [49](#)

## \*Topic **package**

DiceOptim-package, [2](#)

AEI, [4](#)  
AEI.grad, [6](#)  
AEI.grad\_optim, [8](#)  
AKG, [9](#)  
AKG.grad, [11](#)  
AKG.grad\_optim, [12](#)

CL.nsteps, [13](#), [17](#), [51](#), [58](#)

DiceOptim (DiceOptim-package), [2](#)  
DiceOptim-package, [2](#)

EGO.nsteps, [15](#), [16](#), [21](#)  
EI, [15–17](#), [20](#), [22](#), [49](#), [51](#)  
EI.grad, [17](#), [21](#), [35](#), [37](#)  
EI.plugin, [23](#), [26](#)  
EI.plugin.grad, [26](#)  
EQI, [27](#)  
EQI.grad, [29](#)  
EQI.grad\_optim, [31](#)

genoud, [14](#), [16](#), [39](#), [40](#), [42](#), [46](#), [48](#), [50](#), [52](#), [54](#)

km, [14](#), [16](#), [17](#), [20](#), [22](#), [34](#), [36](#), [42](#), [50](#)  
kriging.quantile, [32](#), [36](#)  
kriging.quantile.grad, [34](#)  
kriging.quantile.grad\_optim, [36](#)

max\_AEI, [38](#)  
max\_AKG, [40](#)  
max\_EI, [15](#), [17](#), [21](#), [42](#), [51](#)  
max\_EI.plugin, [45](#)  
max\_EQI, [47](#)  
max\_qEI.CL, [13](#), [15](#), [49](#), [58](#)  
min\_quantile, [51](#)

noisy.optimizer, [53](#)

qEI, [15](#), [21](#), [49](#), [57](#)

update\_km, [59](#)