Package 'DiceOptim'

March 10, 2015

Version 1.5

Title Kriging-Based Optimization for Computer Experiments

Date 2015-03-10

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

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

Description Expected Improvement. EGO algorithm. Multipoint EI and parallelized versions of EGO. Criteria and algorithms for Noisy Kriging-based Optimization , including AEI, AKG, EQI, and EI with plugin.

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

License GPL-2 | GPL-3

URL http://dice.emse.fr/

Repository CRAN

Date/Publication 2015-03-10 18:32:47

NeedsCompilation no

R topics documented:

`DiceOptim-package` *Kriging-based optimization methods for computer experiments*

Description

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

Details

Package:	DiceOptim
Type:	Package
Version:	1.5
Date:	March 2015
License:	GPL-2 | 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 advice in R programming and packaging, as well as DICE and ReDICE 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 Expected Improvement criterion
`qEI`	Analytical expression of the multipoint Expected Improvement (qEI) 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`	Maximization of multipoint expected improvement criterion (qEI)
`qEGO.nsteps`	Batch-sequential EI Algorithm —model updates including re-estimation of covariance
`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 (INRA, Castanet-Tolosan, France)

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

with contributions by C. Chevalier, S. Marmin, and T. Wagner

References

C. Chevalier and D. Ginsbourger (2014) Learning and Intelligent Optimization - 7th International Conference, Lion 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers, chapter Fast computation of the multipoint Expected Improvement with applications in batch selection, pages 59-69, Springer.

N.A.C. Cressie (1993), *Statistics for spatial data*, Wiley series in probability and mathematical statistics.

D. Ginsbourger, R. Le Riche, L. Carraro (2007), A Multipoint Criterion for Deterministic Parallel Global Optimization based on Kriging. The International Conference on Non Convex Programming, 2007.

D. Ginsbourger (2009), *Multiples metamodeles pour l'approximation et l'optimisation de fonctions numeriques multivariables*, Ph.D. thesis, Ecole Nationale Superieure des Mines de Saint-Etienne, 2009. http://www.ginsbourger.ch/recherche/these.php

D. Ginsbourger, R. Le Riche, and L. Carraro (2010), chapter "Kriging is well-suited to parallelize optimization", 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.

V. Picheny and D. Ginsbourger (2013), Noisy kriging-based optimization methods: A unified implementation within the DiceOptim package, *Computational Statistics & Data Analysis*, http://dx.doi.org/10.1016/j.csda.2013.03.018

V. Picheny, D. Ginsbourger, Y. Richet and G. Caplin (2013), Quantile-Based Optimization of Noisy Computer Experiments With Tunable Precision (with discussion and rejoinder), *Technometrics*, http://www.tandfonline.com/doi/abs/10.1080/00401706.2012.707580

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.

O. Roustant, D. Ginsbourger and Yves Deville (2012), DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization, *Journal of Statistical Software*, **51**(1), 1-55, http://www.jstatsoft.org/v51/i01/.

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 disminishing 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

`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 future 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 saving intermediate calculations and reusing them within AEI.grad

Value

Augmented Expected Improvement

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, 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 <- 12 # Change to 21 for a nicer picture
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)
```

```
crit.grid <- apply(design.grid, 1, AEI, model=model, new.noise.var=noise.var)
func.grid <- apply(design.grid, 1, test.function)

# 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

Analytical gradient of the Augmented Expected Improvement (AEI) criterion.

Usage

```
AEI.grad(x, model, new.noise.var=0, y.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	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 (INRA, Castanet-Tolosan, France)

David Ginsbourger (University of Bern, Switzerland)

Examples

```
## Not run:
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 <- 12 # Change to 21 for a nicer picture
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)
```

```
crit.grid <- apply(design.grid, 1, AEI, model=model, new.noise.var=noise.var)
crit.grad <- t(apply(design.grid, 1, AEI.grad, 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")

options(warn=-1)
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)
}

## End(Not run)
```

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

Value

Approximate Knowledge Gradient

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, 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 <- 12 # Change to 21 for a nicer picture
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)

crit.grid <- apply(design.grid, 1, AKG, model=model, new.noise.var=noise.var)
func.grid <- apply(design.grid, 1, test.function)

# 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", 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`	optional: environment for reusing intermediate calculations from AKG

Value

Gradient of the Approximate Knowledge Gradient

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, France)

David Ginsbourger (University of Bern, Switzerland)

Examples

```
################################################################################
###    AKG SURFACE AND ITS GRADIENT ASSOCIATED WITH AN ORDINARY       ####
###                                            KRIGING MODEL          ####
### OF THE BRANIN FUNCTION KNOWN AT A 12-POINT LATIN HYPERCUBE DESIGN ####
################################################################################
## Not run:
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 <- 12 # Change to 21 for a nicer picture
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)

crit.grid <- apply(design.grid, 1, AKG, model=model, new.noise.var=noise.var)
crit.grad <- t(apply(design.grid, 1, AKG.grad, 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")
options(warn=-1)
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)
}

## End(Not run)
```

`EGO.nsteps` *Sequential EI maximization and model re-estimation, with a number of iterations fixed in advance by the user*

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 `km` ,
`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 `genoud`.
`kmcontrol`	an optional list representing the control variables for the re-estimation of the kriging model. The items are the same as in `km` : `penalty`, `optim.method`, `parinit`, `control`. The default values are those contained in `model`, typically corresponding to the variables used in `km` 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 `km` 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 Superieure 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`

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 <- 5 # Was 10, reduced to 5 for speeding up compilation
lower <- rep(0,d)
upper <- rep(1,d)
oEGO <- EGO.nsteps(model=fitted.model1, fun=branin, nsteps=nsteps,
lower=lower, upper=upper, control=list(pop.size=20, BFGSburnin=2))
print(oEGO$par)
print(oEGO$value)

# graphics
n.grid <- 15 # Was 20, reduced to 15 for speeding up compilation
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       ####
##################################################################
## Not run:
# 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="BFGS")

# EGO n steps
library(rgenoud)
nsteps <- 10 # Was 20, reduced to 10 for speeding up compilation
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 <- 15 # Was 20, reduced to 15 for speeding up compilation
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)

## End(Not run)

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

## Not run:
fonction<-hartman6
data(mydata)
a <- mydata
nb<-10
nsteps <- 3 # Maybe be changed to a larger value
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")

## End(Not run)
```

EI *Analytical expression of the Expected Improvement criterion*

Description

Computes the Expected Improvement at current location. The current minimum of the observations can be replaced by an arbitrary value (plugin), which is usefull in particular in noisy frameworks.

Usage

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

Arguments

x	a vector representing the input for which one wishes to calculate EI,
model	an object of class km,
plugin	optional scalar: if provided, it replaces the minimum of the current observations,
type	"SK" or "UK" (by default), depending whether uncertainty related to trend estimation has to be taken into account,
minimization	logical specifying if EI is used in minimiziation or in maximization,
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. If a plugin is specified, it replaces

$$min(Y(X))$$

in the previous formula.

Author(s)

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

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

Victor Picheny (INRA, Castanet-Tolosan, 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 <- 12
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*

Description

Computes the gradient of the Expected Improvement at the current location. The current minimum of the observations can be replaced by an arbitrary value (plugin), which is usefull in particular in noisy frameworks.

Usage

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

Arguments

x	a vector representing the input for which one wishes to calculate EI.
model	an object of class km.
plugin	optional scalar: if provided, it replaces the minimum of the current observations,
type	Kriging type: "SK" or "UK"
minimization	logical specifying if EI is used in minimiziation or in maximization,
envir	an optional environment specifying where to get intermediate values calculated in EI.

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 Superieure des Mines de Saint-Etienne, France).

Victor Picheny (INRA, Castanet-Tolosan, France)

References

D. Ginsbourger (2009), *Multiples metamodeles pour l'approximation et l'optimisation de fonctions numeriques multivariables*, Ph.D. thesis, Ecole Nationale Superieure des Mines de Saint-Etienne, 2009. http://www.ginsbourger.ch/recherche/these.php

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

```
## Not run:
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)
}
}

## End(Not run)
```

EQI *Expected Quantile Improvement*

Description

Evaluation of the Expected Quantile Improvement (EQI) criterion.

Usage

```
EQI(x, model, new.noise.var=0, beta=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).
beta	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

Value

Expected Quantile Improvement

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, 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 <- 12 # Change to 21 for a nicer picture
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)

crit.grid <- apply(design.grid, 1, EQI, model=model, new.noise.var=noise.var, beta=.9)
func.grid <- apply(design.grid, 1, test.function)

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1","V2")
pred <- predict(model, newdata=design.grid, type="UK", checkNames = FALSE)
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)})
```

`EQI.grad` *EQI's Gradient*

Description

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

Usage

```
EQI.grad(x, model, new.noise.var=0, beta=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).
`beta`	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 (INRA, Castanet-Tolosan, France)

David Ginsbourger (University of Bern, Switzerland)

Examples

```
## Not run:
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 <- 12
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)

crit.grid <- apply(design.grid, 1, EQI, model=model,
new.noise.var=noise.var, beta=.9)
crit.grad <- t(apply(design.grid, 1, EQI.grad, model=model,
new.noise.var=noise.var, beta=.9))

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")

options(warn=-1)
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)
```

```
}

## End(Not run)
```

`kriging.quantile` *Kriging quantile*

Description

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

Usage

```
kriging.quantile(x, model, beta=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
`beta`	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 (INRA, Castanet-Tolosan, France)

David Ginsbourger (University of Bern, Switzerland)

Examples

```
##############################################################################
###    KRIGING QUANTILE SURFACE                                            ####
### 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 <- 12 # Change to 21 for a nicer picture
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)

crit.grid <- apply(design.grid, 1, kriging.quantile, model=model, beta=.1)
func.grid <- apply(design.grid, 1, test.function)

# Compute kriging mean and variance on a grid
names(design.grid) <- c("V1","V2")
pred <- predict(model, newdata=design.grid, type="UK", checkNames = FALSE)
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 beta*

Description

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

Usage

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

Arguments

`x`	a vector representing the input for which one wishes to calculate kriging.quantile.grad.
`model`	an object of class `km`.
`beta`	A quantile level (between 0 and 1)
`type`	Kriging type: "SK" or "UK"
`envir`	environment for inheriting intermediate calculations from `"kriging.quantile"`

Value

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

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, France)

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*, J. Stat. Soft., Vol. 51, Issue 1, Oct 2012. http://www.jstatsoft.org/v51/i01/

D. Ginsbourger (2009), *Multiples metamodeles pour l'approximation et l'optimisation de fonctions numeriques multivariables*, Ph.D. thesis, Ecole Nationale Superieure des Mines de Saint-Etienne, 2009. http://www.ginsbourger.ch/recherche/these.php

See Also

EI.grad

Examples

```
## Not run:
##############################################################################
###    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 <- 12 # Change to 21 for a nicer picture
x.grid <- y.grid <- seq(0,1,length=n.grid)
design.grid <- expand.grid(x.grid, y.grid)
nt <- nrow(design.grid)

crit.grid <- apply(design.grid, 1, kriging.quantile, model=model, beta=.1)
crit.grad <- t(apply(design.grid, 1, kriging.quantile.grad, model=model, beta=.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)
```

```
}

## End(Not run)
```

`max_AEI` *Maximizer of the Augmented Expected Improvement criterion function*

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`	optional 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"` (12), `"wait.generations"` (2) and `"BFGSburnin"` (2) of function `"genoud"` (see genoud). 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 (INRA, Castanet-Tolosan, France)

David Ginsbourger (University of Bern, Switzerland)

Examples

```
set.seed(100)

# 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 <- 12 # Change to 21 for a nicer picture
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 <- apply(design.grid, 1, AEI, model=model, new.noise.var=noise.var)

## Not run:
# # 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)})

## End(Not run)
```

---

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" (12), "wait.generations" (2) and "BFGSburnin" (2) of function "genoud" (see genoud). 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

## Not run:
# Compute actual function and criterion on a grid
n.grid <- 12 # Change to 21 for a nicer picture
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 <- apply(design.grid, 1, AKG, 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)})

## End(Not run)
```

`max_EI` *Maximization of the Expected Improvement criterion*

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).

The current minimum of the observations can be replaced by an arbitrary value (plugin), which is usefull in particular in noisy frameworks.

Usage

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

Arguments

`model`	an object of class `km` ,
`plugin`	optional scalar: if provided, it replaces the minimum of the current observations,
`type`	Kriging type: "SK" or "UK"
`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,
`minimization`	logical specifying if EI is used in minimiziation or in maximization,
`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"` (12), `"wait.generations"` (2) and `"BFGSburnin"` (2) of function `"genoud"` (see `genoud`). Numbers into brackets are the default values

Value

A list with components:

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

Author(s)

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

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

Victor Picheny (INRA, Castanet-Tolosan, France)

References

D. Ginsbourger (2009), *Multiples metamodeles pour l'approximation et l'optimisation de fonctions numeriques multivariables*, Ph.D. thesis, Ecole Nationale Superieure 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=lower, upper=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            ####
#############################################################
## Not run:
# 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=lower, upper=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")

## End(Not run)

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

## Not run:
# 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=lower, upper=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")

## End(Not run)
```

`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, beta=0.9, q.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 an observation. Default value is 0 (noise-free observation).
`beta`	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`	optional 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" (12), "wait.generations" (2) and "BFGSburnin" (2) of function "genoud" (see genoud). 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 (INRA, Castanet-Tolosan, 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

## Not run:
# Compute actual function and criterion on a grid
n.grid <- 12 # Change to 21 for a nicer picture
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 <- apply(design.grid, 1, EQI, model=model, new.noise.var=noise.var, beta=.9)

# # 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)})

## End(Not run)
```

`max_qEI` *Maximization of multipoint expected improvement criterion (qEI)*

Description

Maximization of the `qEI` criterion. Two options are available : Constant Liar (CL), and brute force qEI maximization with Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, or GENetic Optimization Using Derivative (genoud) algorithm.

- CL is a heuristic method. First, the regular Expected Improvement EI is maximized (`max_EI`). 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 (default is the minimum of all currently available observations).

- The BFGS algorithm is implemented in the standard function `optim`. Analytical formulae of `qEI` and its gradient `qEI.grad` are used. The `nStarts` starting points are by default sampled with respect to the regular EI (`sampleFromEI`) criterion.

- The "genoud" method calls the function `genoud` using analytical formulae of `qEI` and its gradient `qEI.grad`.

Usage

```
max_qEI(model, npoints, lower, upper, crit = "exact",
minimization = TRUE, optimcontrol = NULL)
```

Arguments

`model`	an object of class `km` ,
`npoints`	an integer representing the desired number of iterations,
`lower`	vector of lower bounds,
`upper`	vector of upper bounds,
`crit`	"exact", "CL" : a string specifying the criterion used. "exact" triggers the maximization of the multipoint expected improvement at each iteration (see `max_qEI`), "CL" applies the Constant Liar heuristic,

minimization	logical specifying if the qEI to be maximized is used in minimiziation or in maximization,
optimcontrol	an optional list of control parameters for optimization. See details.

Details

The parameters of list optimcontrol are :

- optimcontrol$method : "BFGS" (default), "genoud" ; a string specifying the method used to maximize the criterion (irrelevant when crit is "CL" because this method always uses genoud),

- when crit="CL" :

+ optimcontrol$parinit : optional matrix of initial values (must have model@d columns, the number of rows is not constrained),

+ optimcontrol$L : "max", "min", "mean" or a scalar value specifying the liar ; "min" takes model@min, "max" takes model@max, "mean" takes the prediction of the model ; When L is NULL, "min" is taken if minimization==TRUE, else it is "max".

+ The parameters of function genoud. Main parameters are : "pop.size" (default : [N=3*2^model@d for dim<6 and N=32*model@d otherwise]), "max.generations" (default : 12), "wait.generations" (default : 2) and "BFGSburnin" (default : 2).

- when optimcontrol$method = "BFGS" :

+ optimcontrol$nStarts (default : 4),

+ optimcontrol$fastCompute : if TRUE (default), a fast approximation method based on a semi-analytic formula is used, see [Marmin 2014] for details,

+ optimcontrol$samplingFun : a function which sample a batch of starting point (default : sampleFromEI),

+ optimcontrol$parinit : optional 3d-array of initial (or candidate) batches (for all k, parinit[,,k] is a matrix of size npoints*model@d representing one batch). The number of initial batches (length(parinit[1,1,])) is not contrained and does not have to be equal to nStarts. If there is too few initial batches for nStarts, missing batches are drawn with samplingFun (default : NULL),

- when optimcontrol$method = "genoud" :

+ optimcontrol$fastCompute : if TRUE (default), a fast approximation method based on a semi-analytic formula is used, see [Marmin 2014] for details,

+ optimcontrol$parinit : optional matrix of candidate starting points (one row corresponds to one point),

+ The parameters of the genoud function. Main parameters are "pop.size" (default : [50*(model@d)*(npoints)]), "max.generations" (default : 5), "wait.generations" (default : 2), "BFGSburnin" (default : 2).

Value

A list with components:

par	A matrix containing the npoints input vectors found.
value	A value giving the qEI computed in par.

Author(s)

Sebastien Marmin (Departement of Mathematics and Statistics, University of Berne, Switzerland ; Ecole Centrale de Marseille, France ; Institut de Radioprotection et de Surete Nucleaire, France),

Clement Chevalier (Institute of Mathematics, University of Zurich, Switzerland),

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

References

C. Chevalier and D. Ginsbourger (2014) Learning and Intelligent Optimization - 7th International Conference, Lion 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers, chapter Fast computation of the multipoint Expected Improvement with applications in batch selection, pages 59-69, Springer.

D. Ginsbourger, R. Le Riche, L. Carraro (2007), A Multipoint Criterion for Deterministic Parallel Global Optimization based on Kriging. The International Conference on Non Convex Programming, 2007.

D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization (2010), In Lim Meng Hiot, Yew Soon Ong, Yoel Tenne, and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, Adaptation Learning and Optimization, pages 131-162. Springer Berlin Heidelberg.

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

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

See Also

qEI, qEI.grad

Examples

```
## Not run:

set.seed(000)
# 3-points EI maximization.
# 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"
lower <- c(0,0)
upper <- c(1,1)

# number of point in the bacth
```

```
batchSize <- 3

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

# maximization of qEI

# With a multistarted BFGS algorithm
maxBFGS <- max_qEI(model = fitted.model, npoints = batchSize, lower = lower, upper = upper,
crit = "exact",optimcontrol=list(nStarts=3,method = "BFGS"))
# With a genetic algorithme using derivatives
maxGen  <- max_qEI(model = fitted.model, npoints = batchSize, lower = lower, upper = upper,
crit = "exact", optimcontrol=list(nStarts=3,method = "genoud",pop.size=100,max.generations = 15))
# With the constant liar heuristic
maxCL   <- max_qEI(model = fitted.model, npoints = batchSize, lower = lower, upper = upper,
crit = "CL",optimcontrol=list(pop.size=20))

# comparison
print(maxBFGS$value)
print(maxGen$value)
print(maxCL$value)


## End(Not run)
```

`min_quantile` *Minimization of the Kriging quantile.*

Description

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

Usage

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

Arguments

`model`	a Kriging model of "km" class
`beta`	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 : [N=3*2^dim for dim<6 and N=32*dim otherwise]), "`max.generations`" (12), "`wait.generations`" (2) and "`BFGSburnin`" (2) of function "`genoud`" (see genoud). 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 (INRA, Castanet-Tolosan, 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, beta=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 <- 12 # Change to 21 for a nicer picture
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 <- apply(design.grid, 1, kriging.quantile, model=model, beta=.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 homogenously 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, parinit=NULL, control=NULL, CovReEstimate=TRUE,
NoiseReEstimate=FALSE, nugget.LB=1e-5, estim.model=NULL, type="UK")
```

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 "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 genoud). Numbers into brackets are the default values
CovReEstimate	optional boolean specfiying if the covariance parameters should be re-estimated at every iteration (default value = TRUE)
NoiseReEstimate	optional boolean specfiying if the noise variance should be re-estimated at every iteration (default value = FALSE)
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 heterogenenous noise variances.
type	"SK" or "UK" for Kriging with known or estimated trend

Value

A list with components:

model	the current (last) kriging model of "km" class
best.x	The best design found
best.y	The objective function value at best.x
best.index	The index of best.x in the design of experiments
history.x	The added observations
history.y	The added observation values
history.hyperparam	The history of the kriging parameters
estim.model	If noiseReEstimate=TRUE, the current (last) kriging model of "km" class for estimating the noise variance.
history.noise.var	If noiseReEstimate=TRUE, the history of the noise variance estimate.

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, France)

Examples

```
##########################################################################
### EXAMPLE 1: 3 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 can be increased)
n.ite <- 2
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$model
best.x    <- optim.result$best.x
new.doe   <- optim.result$history.x

## Not run:
##### DRAW RESULTS #####
# Compute actual function on a grid
n.grid <- 12
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(object=model, newdata=design.grid, type="UK", checkNames = FALSE)
mk.grid1 <- pred$m
sk.grid1 <- pred$sd

pred <- predict(object=new.model, newdata=design.grid, type="UK", checkNames = FALSE)
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)})

## End(Not run)

##############################################################################
### EXAMPLE 2: 3 OPTIMIZATION STEPS USING EQI WITH UNKNOWN NOISE       ###
### AND UNKNOWN COVARIANCE PARAMETERS FOR THE BRANIN FUNCTION          ###
```

```
##########################################################################
# Same initial model and parameters as for example 1
n.ite <- 2 # May be changed to a larger value
res <- noisy.optimizer(optim.crit="min.quantile",
optim.param=list(type="quantile",quantile=0.01),
model=model, n.ite=n.ite, noise.var=noise.var, funnoise=funnoise,
lower=lower, upper=upper,
control=list(print.level=0),CovReEstimate=TRUE, NoiseReEstimate=TRUE)

# Plot actual function and observations
plot(model@X[,1], model@X[,2], pch=17,xlim=c(0,1),ylim=c(0,1))
points(res$history.x[1,], res$history.x[2,], col="blue")

# Restart: requires the output estim.model of the previous run
# to deal with potential repetitions
res2 <- noisy.optimizer(optim.crit="min.quantile",
optim.param=list(type="quantile",quantile=0.01),
model=res$model, n.ite=n.ite, noise.var=noise.var, funnoise=funnoise,
lower=lower, upper=upper, estim.model=res$estim.model,
control=list(print.level=0),CovReEstimate=TRUE, NoiseReEstimate=TRUE)

# Plot new observations
points(res2$history.x[1,], res2$history.x[2,], col="red")
```

`qEGO.nsteps` *Sequential multipoint Expected improvement (qEI) maximizations and model re-estimation*

Description

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

Usage

```
qEGO.nsteps(fun, model, npoints, nsteps, lower = rep(0, model@d),
upper = rep(1, model@d), crit = "exact",  minimization = TRUE,
optimcontrol = NULL, cov.reestim = TRUE, ...)
```

Arguments

`fun`	the objective function to be optimized,
`model`	an object of class `km` ,
`npoints`	an integer repesenting the desired batchsize,
`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,
crit	"exact", "CL" : a string specifying the criterion used. "exact" triggers the maximization of the multipoint expected improvement at each iteration (see max_qEI), "CL" applies the Constant Liar heuristic,
minimization	logical specifying if we want to minimize or maximize fun,
optimcontrol	an optional list of control parameters for the qEI optimization (see details or max_qEI),
cov.reestim	optional boolean specifying if the kriging hyperparameters should be re-estimated at each iteration,
...	optional arguments for fun.

Details

The parameters of list optimcontrol are :

- optimcontrol$method : "BFGS" (default), "genoud" ; a string specifying the method used to maximize the criterion (irrelevant when crit is "CL" because this method always uses genoud),

- when crit="CL" :

+ optimcontrol$parinit : optional matrix of initial values (must have model@d columns, the number of rows is not constrained),

+ optimcontrol$L : "max", "min", "mean" or a scalar value specifying the liar ; "min" takes model@min, "max" takes model@max, "mean" takes the prediction of the model ; When L is NULL, "min" is taken if minimization==TRUE, else it is "max".

+ The parameters of function genoud. Main parameters are : "pop.size" (default : [N=3*2^model@d for dim<6 and N=32*model@d otherwise]), "max.generations" (default : 12), "wait.generations" (default : 2) and "BFGSburnin" (default : 2).

- when optimcontrol$method = "BFGS" :

+ optimcontrol$nStarts (default : 4),

+ optimcontrol$fastCompute : if TRUE (default), a fast approximation method based on a semi-analytic formula is used, see [Marmin 2014] for details,

+ optimcontrol$samplingFun : a function which sample a batch of starting point (default : sampleFromEI),

+ optimcontrol$parinit : optional 3d-array of initial (or candidate) batches (for all k, parinit[,,k] is a matrix of size npoints*model@d representing one batch). The number of initial batches (length(parinit[1,1,])) is not contrained and does not have to be equal to nStarts. If there is too few initial batches for nStarts, missing batches are drawn with samplingFun (default : NULL),

- when optimcontrol$method = "genoud" :

+ optimcontrol$fastCompute : if TRUE (default), a fast approximation method based on a semi-analytic formula is used, see [Marmin 2014] for details,

+ optimcontrol$parinit : optional matrix of candidate starting points (one row corresponds to one point),

+ The parameters of the genoud function. Main parameters are "pop.size" (default : [50*(model@d)*(npoints)]), "max.generations" (default : 5), "wait.generations" (default : 2), "BFGSburnin" (default : 2).

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,
`nsteps`	an integer representing the desired number of iterations (given in argument),
`lastmodel`	an object of class `km` corresponding to the last kriging model fitted,
`history`	a vector of size `nsteps` representing the current known optimum at each step.

Author(s)

Sebastien Marmin (Departement of Mathematics and Statistics, University of Berne, Switzerland ; Ecole Centrale de Marseille, France ; Institut de Radioprotection et de Surete Nucleaire, France),

Clement Chevalier (Institute of Mathematics, University of Zurich, Switzerland),

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

References

C. Chevalier and D. Ginsbourger (2014) Learning and Intelligent Optimization - 7th International Conference, Lion 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers, chapter Fast computation of the multipoint Expected Improvement with applications in batch selection, pages 59-69, Springer.

D. Ginsbourger, R. Le Riche, L. Carraro (2007), A Multipoint Criterion for Deterministic Parallel Global Optimization based on Kriging. The International Conference on Non Convex Programming, 2007.

D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization (2010), In Lim Meng Hiot, Yew Soon Ong, Yoel Tenne, and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, Adaptation Learning and Optimization, pages 131-162. Springer Berlin Heidelberg.

S. Marmin. Developpements pour l'evaluation et la maximisation du critere d'amelioration esperee multipoint en optimisation globale (2014). Master's thesis, Mines Saint-Etienne (France) and University of Bern (Switzerland).

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

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

See Also

`qEI`, `max_qEI`, `qEI.grad`

Examples

```
## Not run:
set.seed(123)
#####################################################
### 4 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 <- 4 # Was 10, reduced to 4 for speeding up execution
lower <- rep(0,d)
upper <- rep(1,d)
npoints <- 3 # The batchsize
oEGO <- qEGO.nsteps(model = fitted.model1, branin, npoints = npoints, nsteps = nsteps,
crit="exact", lower, upper, optimcontrol = NULL)
print(oEGO$par)
print(oEGO$value)
plot(c(1:nsteps),oEGO$history,xlab='step',ylab='Current known minimum')

# graphics
n.grid <- 15 # Was 20, reduced to 15 for speeding up compilation
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=c(tcrossprod(rep(1,npoints),1:nsteps)), pos=3)

## End(Not run)
```

qEI *Analytical expression of the multipoint expected improvement (qEI) criterion*

Description

Computes the multipoint expected improvement criterion.

Usage

```
qEI(x, model, plugin = NULL, type = "UK", minimization = TRUE, fastCompute = TRUE,
eps = 10^(-5), envir = NULL)
```

Arguments

x	a matrix representing the set of input points (one row corresponds to one point) where to evaluate the qEI criterion,
model	an object of class km,
plugin	optional scalar: if provided, it replaces the minimum of the current observations,
type	"SK" or "UK" (by default), depending whether uncertainty related to trend estimation has to be taken into account,
minimization	logical specifying if EI is used in minimiziation or in maximization,
fastCompute	if TRUE, a fast approximation method based on a semi-analytic formula is used (see [Marmin 2014] for details),
eps	the value of *epsilon* of the fast computation trick. Relevant only if fastComputation is TRUE,
envir	an optional environment specifying where to get intermediate values calculated in qEI.

Value

The multipoint Expected Improvement, defined as

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

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)

Sebastien Marmin (Departement of Mathematics and Statistics, University of Berne, Switzerland ; Ecole Centrale de Marseille, France ; Institut de Radioprotection et de Surete Nucleaire, France),

Clement Chevalier (Institute of Mathematics, University of Zurich, Switzerland),

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

References

C. Chevalier and D. Ginsbourger (2014) Learning and Intelligent Optimization - 7th International Conference, Lion 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers, chapter Fast computation of the multipoint Expected Improvement with applications in batch selection, pages 59-69, Springer.

D. Ginsbourger, R. Le Riche, L. Carraro (2007), A Multipoint Criterion for Deterministic Parallel Global Optimization based on Kriging. The International Conference on Non Convex Programming, 2007.

S. Marmin. Developpements pour l'evaluation et la maximisation du critere d'amelioration esperee multipoint en optimisation globale (2014). Master's thesis, Mines Saint-Etienne (France) and University of Bern (Switzerland).

D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization (2010), In Lim Meng Hiot, Yew Soon Ong, Yoel Tenne, and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, Adaptation Learning and Optimization, pages 131-162. Springer Berlin Heidelberg.

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

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

See Also

EI

Examples

```
## Not run:
set.seed(007)

# Monte-Carlo validation

# a 4-d, 81-points grid design, and the corresponding response
d <- 4;n <- 3^d
design <- do.call(expand.grid,rep(list(seq(0,1,length=3)),d))
names(design) <- paste("x",1:d,sep="")
y <- data.frame(apply(design, 1, hartman4))
names(y) <- "y"

# learning
model <- km(~1, design=design, response=y, control=list(trace=FALSE))

# pick up 10 points sampled from the 1-point expected improvement
q <- 10
X <- sampleFromEI(model,n=q)

# simulation of the minimum of the kriging random vector at X
t1 <- proc.time()
newdata <- as.data.frame(X); colnames(newdata) <- colnames(model@X)
krig  <- predict(object=model, newdata=newdata,type="UK",se.compute=TRUE, cov.compute=TRUE)
mk <- krig$mean
```

```
Sigma.q <- krig$cov
mychol <- chol(Sigma.q)
nsim <- 300000
white.noise <- rnorm(n=nsim*q)
minYsim <- apply(crossprod(mychol,matrix(white.noise,nrow=q)) + mk,2,min)

# simulation of the improvement (minimization)
qImprovement <- (min(model@y)-minYsim)*((min(model@y)-minYsim) > 0)

# empirical expectation of the improvement and confident interval (95
eiMC <- mean(qImprovement)
confInterv <- c(eiMC - 1.96*sd(qImprovement)*1/sqrt(nsim),eiMC + 1.96*sd(qImprovement)*1/sqrt(nsim))

# MC estimation of the qEI
print(eiMC)
t2 <- proc.time()
# qEI with analytical formula
qEI(X,model,fastCompute= FALSE)
t3 <- proc.time()
# qEI with fast computation trick
qEI(X,model)
t4 <- proc.time()
t2-t1  # Time of MC computation
t3-t2  # Time of normal computation
t4-t3  # Time of fast computation

## End(Not run)
```

`qEI.grad` *Gradient of the multipoint expected improvement (qEI) criterion*

Description

Computes an exact or approximate gradient of the multipoint expected improvement criterion

Usage

```
qEI.grad(x, model, plugin = NULL, type = "UK", minimization = TRUE,
fastCompute = TRUE, eps = 10^(-6), envir=NULL)
```

Arguments

`x`	a matrix representing the set of input points (one row corresponds to one point) where to evaluate the gradient,
`model`	an object of class `km`,
`plugin`	optional scalar: if provided, it replaces the minimum of the current observations,
`type`	"SK" or "UK" (by default), depending whether uncertainty related to trend estimation has to be taken into account,

`minimization`	logical specifying if EI is used in minimiziation or in maximization,
`fastCompute`	if TRUE, a fast approximation method based on a semi-analytic formula is used (see [Marmin 2014] for details),
`eps`	the value of *epsilon* of the fast computation trick. Relevant only if `fastComputation` is TRUE,
`envir`	an optional environment specifying where to get intermediate values calculated in `qEI`.

Value

The gradient of the multipoint expected improvement criterion with respect to x. A 0-matrix is returned if the batch of input points contains twice the same point or a point from the design experiment of the km object (the gradient does not exist in these cases).

Author(s)

Sebastien Marmin (Departement of Mathematics and Statistics, University of Berne, Switzerland ; Ecole Centrale de Marseille, France ; Institut de Radioprotection et de Surete Nucleaire, France),

Clement Chevalier (Institute of Mathematics, University of Zurich, Switzerland),

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

References

C. Chevalier and D. Ginsbourger (2014) Learning and Intelligent Optimization - 7th International Conference, Lion 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers, chapter Fast computation of the multipoint Expected Improvement with applications in batch selection, pages 59-69, Springer.

D. Ginsbourger, R. Le Riche, L. Carraro (2007), A Multipoint Criterion for Deterministic Parallel Global Optimization based on Kriging. The International Conference on Non Convex Programming, 2007.

D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization (2010), In Lim Meng Hiot, Yew Soon Ong, Yoel Tenne, and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, Adaptation Learning and Optimization, pages 131-162. Springer Berlin Heidelberg.

S. Marmin. Developpements pour l'evaluation et la maximisation du critere d'amelioration esperee multipoint en optimisation globale (2014). Master's thesis, Mines Saint-Etienne (France) and University of Bern (Switzerland).

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

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

See Also

`qEI`

Examples

```
## Not run:

set.seed(5182)
# Example 1 - validation by comparison to finite difference approximations

# a 10-points optimum LHS design and the corresponding responses
d <- 2;n <- d*5
design <- optimumLHS(n=n,k=d)
names(design)<-c("x1", "x2")
lower <- c(0,0)
upper <- c(1,1)
y <- data.frame(apply(design, 1, branin))
names(y) <- "y"

# learning
model <- km(~1, design=design, response=y)

# pick up 4 points sampled from the simple expected improvement
q <- 4
X <- sampleFromEI(model,n=q)

# compute the gradient at the 4-point batch
grad.analytic <- qEI.grad(X,model)
# numerically compute the gradient
grad.numeric <- matrix(NaN,q,d)
eps <- 10^(-6)
EPS <- matrix(0,q,d)
for (i in 1:q) {
  for (j in 1:d) {
    EPS[i,j] <- eps
   grad.numeric[i,j] <- 1/eps*(qEI(X+EPS,model,fastCompute=FALSE)-qEI(X,model,fastCompute=FALSE))
    EPS[i,j] <- 0
  }
}
print(grad.numeric)
print(grad.analytic)

# graphics: displays the EI criterion, the design points in black,
# the batch points in red and the gradient in blue.
nGrid <- 15
gridAxe1 <- seq(lower[1],upper[1],length=nGrid)
gridAxe2 <- seq(lower[2],upper[2],length=nGrid)
grid <- expand.grid(gridAxe1,gridAxe2)
aa <- apply(grid,1,EI,model=model)
myMat <- matrix(aa,nrow=nGrid)
image(x = gridAxe1, y = gridAxe2, z = myMat,
      col = colorRampPalette(c("darkgray","white"))(5*10),
      ylab = names(design)[1], xlab=names(design)[2],
      main = "qEI-gradient of a batch of 4 points", axes = TRUE,
      zlim = c(min(myMat), max(myMat)))
contour(x = gridAxe1, y = gridAxe2, z = myMat,
```

```
        add = TRUE, nlevels = 10)
points(X[,1],X[,2],pch=19,col='red')
points(model@X[,1],model@X[,2],pch=19)
arrows(X[,1],X[,2],X[,1]+0.012*grad.analytic[,1],X[,2]+0.012*grad.analytic[,2],col='blue')

## End(Not run)
```

`sampleFromEI` *Sampling points according to the expected improvement criterion*

Description

Samples `n` points from a distribution proportional to the expected improvement (EI) computed from a `km` object.

Usage

```
sampleFromEI(model, minimization = TRUE, n = 1, initdistrib = NULL,
lower = rep(0,model@d), upper = rep(1,model@d), T = NULL)
```

Arguments

`model`	an object of class `km`,
`minimization`	logical specifying if EI is used in minimiziation or in maximization,
`n`	number of points to be sampled,
`initdistrib`	matrix of candidate points. If `NULL`, (`model@d*1000`) points are obtained by latin hypercube sampling,
`lower`	vector of lower bounds,
`upper`	vector of upper bounds,
`T`	optional scalar : if provided, it replaces the current minimum (or maximum) of observations.

Value

A `n*model@d` matrix containing the sampled points.

Author(s)

Sebastien Marmin (Departement of Mathematics and Statistics, University of Berne, Switzerland ; Ecole Centrale de Marseille, France ; Institut de Radioprotection et de Surete Nucleaire, France),

Clement Chevalier (Institute of Mathematics, University of Zurich, Switzerland),

David Ginsbourger (Departement of Mathematics and Statistics, University of Berne, 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.

See Also

EI, km, qEI

Examples

```
## Not run:

set.seed(004)

# a 9-points factorial design, and the corresponding responses
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)
lower <- c(0,0)
upper <- c(1,1)
names(response.branin) <- "y"


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

# sample a 30 point batch
batchSize <- 30
x <- sampleFromEI(model = fitted.model, n = batchSize, lower = lower, upper = upper)

# graphics
# displays the EI criterion, the design points in black and the EI-sampled points in red.
nGrid <- 15
gridAxe1 <- seq(lower[1],upper[1],length=nGrid)
gridAxe2 <- seq(lower[2],upper[2],length=nGrid)
grid <- expand.grid(gridAxe1,gridAxe2)
aa <- apply(grid,1,EI,model=fitted.model)
myMat <- matrix(aa,nrow=nGrid)
image(x = gridAxe1, y = gridAxe2, z = myMat,
      col = colorRampPalette(c("darkgray","white"))(5*10),
      ylab = names(design.fact)[1], xlab=names(design.fact)[2],
      main = "Sampling from the expected improvement criterion",
      axes = TRUE, zlim = c(min(myMat), max(myMat)))
contour(x = gridAxe1, y = gridAxe2, z = myMat,
        add = TRUE, nlevels = 10)
points(x[,1],x[,2],pch=19,col='red')
points(fitted.model@X[,1],fitted.model@X[,2],pch=19)

## End(Not run)
```

update_km_noisyEGO *Update of one or two Kriging models when adding new observation*

Description

Update of a noisy Kriging model when adding new observation, with or without covariance parameter re-estimation. When the noise level is unkown, a twin model "estim.model" is also updated.

Usage

```
update_km_noisyEGO(model, x.new, y.new, noise.var=0, type="UK",
add.obs=TRUE, index.in.DOE=NULL, CovReEstimate=TRUE,
NoiseReEstimate=FALSE, estim.model=NULL, nugget.LB=1e-5)
```

Arguments

model	a Kriging model of "km" class
x.new	a matrix containing the new points of experiments
y.new	a matrix containing the function values on the points NewX
noise.var	scalar: noise variance
type	kriging type: "SK" or "UK"
add.obs	boolean: if TRUE, the new point does not exist already in the design of experiment model@X
index.in.DOE	optional integer: if add.obs=TRUE, it specifies the index of the observation in model@X corresponding to x.new
CovReEstimate	optional boolean specfiying if the covariance parameters should be re-estimated (default value = TRUE)
NoiseReEstimate	optional boolean specfiying if the noise variance should be re-estimated (default value = TRUE)
estim.model	optional input of "km" class. Required if NoiseReEstimate=TRUE, in order to deal with repetitions.
nugget.LB	optional scalar: is used to define a lower bound on the noise variance.

Value

A list containing:

model	The updated Kriging model
estim.model	If NoiseReEstimate=TRUE, the updated estim.model
noise.var	If NoiseReEstimate=TRUE, the re-estimated noise variance

Author(s)

Victor Picheny (INRA, Castanet-Tolosan, France)

Index