Package ‘pGPx’

August 20, 2018

Type Package

Title Pseudo-Realizations for Gaussian Process Excursions

Version 0.1.1

Date 2018-08-20

Author Dario Azzimonti [aut, cre, cph] (<https://orcid.org/0000-0001-5080-3061>), Julien Bect [ctb], Pedro Felzenszwalb [ctb, cph] (Original dt code, http://cs.brown.edu/people/pfelzens/dt/index.html)

Maintainer Dario Azzimonti <dario.azzimonti@gmail.com>

Description Computes pseudo-realizations from the posterior distribution of a Gaussian Process (GP) with the method described in Azzimonti et al. (2016) <doi:10.1137/141000749>. The realizations are obtained from simulations of the field at few well chosen points that minimize the expected distance in measure between the true excursion set of the field and the approximate one. Also implements a R interface for (the main function of) Distance Transform of sampled Functions (<http://cs.brown.edu/people/pfelzens/dt/index.html>).

URL https://epubs.siam.org/doi/abs/10.1137/141000749

License GPL-3

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.13), DiceKriging, pbivnorm, KrigInv, rgenoud, randtoolbox, pracma, grDevices

Suggests anMC, DiceDesign

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-08-20 20:40:03 UTC

R topics documented:

`computeVolumes` *Compute Excursion Volume Distribution*

Description

Compute the volume of excursion for each realization, includes a bias.correction for the mean. If the input is the actual GP values, compute also the random sets.

Usage

```
computeVolumes(rand.set, threshold, nsim, n.int.points, bias.corr = F,
  model = NULL, bias.corr.points = NULL)
```

Arguments

`rand.set`	a matrix of size `n.int.pointsxnsim` containing the excursion set realizations stored as long vectors. For example the excursion set obtained from the result of `simulate_and_interpolate`.
`threshold`	threshold value
`nsim`	number of simulations of the excursion set
`n.int.points`	total length of the excursion set discretization. The size of the image is `sqrt(n.int.points)`.
`bias.corr`	a boolean, if `TRUE` it computes the bias correction for the volume distribution.
`model`	the km model for computing the bias correction. If `NULL` the bias correction is not computed.
`bias.corr.points`	a matrix with d columns with the input points where to compute the bias correction. If `NULL` it is initialized as the first `n.int.points` of the Sobol' sequence.

Value

A vector of size nsim containing the excursion volumes for each realization.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
### Simulate and interpolate for a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=50,
                                                                   dimension = 2,
                                                                  seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))
# Get simulation points
# Here they are not optimized, you can use optim_dist_measure to find optimized points
simu_points <- DiceDesign::maximinSA_LHS(DiceDesign::lhsDesign(n=100,
                                                                dimension = d,
                                                                seed=1)$design)$design
# obtain nsims posterior realization at simu_points
nsims <- 30
nn_data<-expand.grid(seq(0,1,,50),seq(0,1,,50))
nn_data<-data.frame(nn_data)
colnames(nn_data)<-colnames(kmModel@X)
approx.simu <- simulate_and_interpolate(object=kmModel, nsim = nsims, simupoints = simu_points,
                                        interpolatepoints = as.matrix(nn_data),
                                        nugget.sim = 0, type = "UK")
exVol<- computeVolumes(rand.set = approx.simu,threshold = -10,
                         nsim = nsims,n.int.points = 50^2,bias.corr=TRUE, model=kmModel)
```

```
hist(exVol, main="Excursion Volume")
```

---

compute_contourLength *Compute contour lenghts*

Description

Computes the contour lengths for the excursion sets in gpRealizations

Usage

```
compute_contourLength(gpRealizations, threshold, nRealizations, verb = 1)
```

Arguments

gpRealizations	a matrix of size nRealizationsximageSize^2 containing the GP realizations stored as long vectors. For example the object returned by simulate_and_interpolate.
threshold	threshold value
nRealizations	number of simulations of the excursion set
verb	an integer to choose the level of verbosity

Value

A vector of size nRealizations containing the countour lines lenghts.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
### Simulate and interpolate for a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
```

```
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=50,
                                                                   dimension = 2,
                                                                  seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))
# Get simulation points
# Here they are not optimized, you can use optim_dist_measure to find optimized points
simu_points <- DiceDesign::maximinSA_LHS(DiceDesign::lhsDesign(n=100,
                                                                dimension = d,
                                                                seed=1)$design)$design


# obtain nsims posterior realization at simu_points
nsims <- 1
nn_data<-expand.grid(seq(0,1,,50),seq(0,1,,50))
nn_data<-data.frame(nn_data)
colnames(nn_data)<-colnames(kmModel@X)
approx.simu <- simulate_and_interpolate(object=kmModel, nsim = nsims, simupoints = simu_points,
                                        interpolatepoints = as.matrix(nn_data),
                                        nugget.sim = 0, type = "UK")
cLLs<- compute_contourLength(gpRealizations = approx.simu,threshold = -10,
                             nRealizations = nsims,verb = 1)
```

`dtt_fast` *Rcpp implementation of Felzenszwalb distance transfom*

Description

Rcpp wrapper for the distance transform algorithm described in Felzenszwalb and Huttenlocher (2012)

Usage

```
dtt_fast(x)
```

Arguments

x	matrix of booleans of size nxm representing a (binary) image

Value

A matrix of size nxm containing the distance transform result. Note that this function does not perform any checks on x.

Author(s)

Pedro Felzenszwalb for the header files dt.h and misc.h that do the work, Dario Azzimonti and Julien Bect for the wrapper.

References

Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance Transforms of Sampled Functions. Theory of Computing, 8(19):415-428.

Examples

```
# Create an image with a square
nc = 256
nr = 256
xx = matrix(FALSE,ncol=nc,nrow=nr)
xx[(nr/16):(nr/16*15-1),nc/16]<-rep(TRUE,nr/16*14)
xx[(nr/16):(nr/16*15-1),nc/16*15]<-rep(TRUE,nr/16*14)
xx[nr/16,(nc/16):(nc/16*15-1)]<-rep(TRUE,nc/16*14)
xx[nr/16*15,(nc/16):(nc/16*15-1)]<-rep(TRUE,nc/16*14)
# Compute Distance transform
zz<- dtt_fast(xx)

# Plot the results
image(xx,col=grey.colors(20), main="Original image")
image(zz,col=grey.colors(20), main="Distance transform")
```

DTV *Compute Distance Transform Variability*

Description

Compute the expected L^2 distance between the average distance transform and the set realizations. If the input is the actual values of the gaussian process, compute also the random sets.

Usage

```
DTV(rand.set, threshold, nsim, n.int.points)
```

Arguments

rand.set	a matrix of size n.int.pointsxnsim containing the excursion set realizations stored as long vectors. For example the excursion set obtained from the result of simulate_and_interpolate.
threshold	threshold value
nsim	number of simulations of the excursion set
n.int.points	total length of the excursion set discretization. The size of the image is sqrt(n.int.points).

Value

A list containing

- `variance`: Value of the distance transform variability. The integral of `dvar` over the spatial domain.
- `dbar`: empirical distance average transform $1/N \sum_{i=1}^{N} d(x, \Gamma_i)$, a matrix of size `n.int.points` x `n.int.points`
- `dvar`: empirical variance of distance transform $1/N \sum_{i=1}^{N} (d(x, \Gamma_i) - dbar)^2$, a matrix of size `n.int.points` x `n.int.points`
- `alldt`: distance transforms for all realizations, a matrix of size `n.int.points` x `nsim`
- `naTot`: Total number of infinite distance transform values. These are returned in realizations where there is no excursion.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance Transforms of Sampled Functions. Theory of Computing, 8(19):415-428.

Examples

```
### Simulate and interpolate for a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=50,
                                                                  dimension = 2,
                                                                 seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))
# Get simulation points
# Here they are not optimized, you can use optim_dist_measure to find optimized points
```

```
simu_points <- DiceDesign::maximinSA_LHS(DiceDesign::lhsDesign(n=100,
                                                               dimension = d,
                                                               seed=1)$design)$design

# obtain nsims posterior realization at simu_points
nsims <- 30
nn_data<-expand.grid(seq(0,1,,50),seq(0,1,,50))
nn_data<-data.frame(nn_data)
colnames(nn_data)<-colnames(kmModel@X)
approx.simu <- simulate_and_interpolate(object=kmModel, nsim = nsims, simupoints = simu_points,
                                        interpolatepoints = as.matrix(nn_data),
                                        nugget.sim = 0, type = "UK")
Dvar<- DTV(rand.set = approx.simu,threshold = -10,
                             nsim = nsims,n.int.points = 50^2)

image(matrix(Dvar$dbar,ncol=50),col=grey.colors(20),main="average distance transform")
image(matrix(Dvar$dvar,ncol=50),col=grey.colors(20),main="variance of distance transform")
points(design,pch=17)
```

`edm_crit` *Distance in measure criterion*

Description

Computes the distance in measure criterion.

Usage

```
edm_crit(x, integration.points, integration.weights = NULL,
  intpoints.oldmean, intpoints.oldsd, precalc.data, model, threshold,
  batchsize, alpha, current.crit)
```

Arguments

- `x` vector of dimension d representing the ith point where to compute the criterion
- `integration.points` p*d matrix of points for numerical integration in the X space.
- `integration.weights` Vector of size p corresponding to the weights of these integration points.
- `intpoints.oldmean` Vector of size p corresponding to the kriging mean at the integration points.
- `intpoints.oldsd` Vector of size p corresponding to the kriging standard deviation at the integration points.
- `precalc.data` list result of precomputeUpdateData with `model` and `x`.
- `model` km model

threshold	threshold selected for excursion set
batchsize	number of simulation points
alpha	value of Vorob'ev threshold
current.crit	Current value of the criterion

Value

the value of the expected distance in measure criterion at x

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

`edm_crit2` *Distance in measure criterion*

Description

Computes the distance in measure criterion. To be used in optimization routines.

Usage

```
edm_crit2(x, other.points, integration.points,
  integration.weights = NULL, intpoints.oldmean, intpoints.oldsd,
  precalc.data, model, threshold, batchsize, alpha, current.crit)
```

Arguments

x	vector of dimension d representing the point where to compute the criterion
other.points	Vector giving the other batchsize-1 points at which one wants to evaluate the criterion
integration.points	p*d matrix of points for numerical integration in the X space.
integration.weights	Vector of size p corresponding to the weights of these integration points.
intpoints.oldmean	Vector of size p corresponding to the kriging mean at the integration points.
intpoints.oldsd	Vector of size p corresponding to the kriging standard deviation at the integration points.
precalc.data	list result of precomputeUpdateData with model and x.

`model`	km model
`threshold`	threshold selected for excursion set
`batchsize`	number of simulation points
`alpha`	value of Vorob'ev threshold
`current.crit`	Current value of the criterion

Value

the value of the expected distance in measure criterion at x,`other.points`.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

`expDistMeasure` *Compute expected distance in measure of approximate excursion set*

Description

Computes expected distance in measure between the excursion set of the approximated process and the true excursion set.

Usage

```
expDistMeasure(simupoints, model, threshold, batchsize,
  integration.param = NULL)
```

Arguments

`simupoints`	a numeric array of size `batchsize*d` containing the simulation points.
`model`	a km model
`threshold`	threshold value
`batchsize`	number of simulations points
`integration.param`	a list containing parameters for the integration of the criterion A, see max_sur_parallel for more details.

Value

A positive value indicating the expected distance in measure.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
### Compute optimal simulation points in a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=20,
                                                                  dimension = 2,
                                                                 seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))


threshold <- -10

# Obtain simulation point sampling from maximin LHS design
batchsize <- 50
set.seed(1)
mmLHS_simu_points <-  DiceDesign::maximinSA_LHS(DiceDesign::lhsDesign(n=batchsize,
                                                                     dimension = d,
                                                                  seed=1)$design)$design


# Compute expected distance in measure for approximation obtain from random simulation points
EDM_mmLHS <- rep(NA,batchsize)
integcontrol <- list(distrib="sobol",n.points=1000)
integration.param <- KrigInv::integration_design(integcontrol,d=d,
                                        lower=c(0,0),upper=c(1,1),
                                        model=kmModel,T=threshold)
integration.param$alpha <- 0.5
for(i in seq(1,batchsize)){
```

```
EDM_mmLHS[i]<-expDistMeasure( mmLHS_simu_points[1:i,],model = kmModel,
                              threshold = threshold,batchsize = i,
                              integration.param = integration.param  )
}

plot(EDM_mmLHS,type='l',main="Expected distance in measure",xlab="batchsize")


## Not run:
# Get optimized simulation points with algorithm B
simu_points <- optim_dist_measure(model=kmModel,threshold = threshold,
                                  lower = c(0,0),upper = c(1,1),
                                  batchsize = batchsize,algorithm = "B")
# plot the criterion value
plot(1:batchsize,simu_points$value,type='l',main="Criterion value")

# Compute expected distance in measure for approximation obtained from optimized simulation points
EDM_optB <- rep(NA,batchsize)
for(i in seq(1,batchsize)){
  EDM_optB[i]<-expDistMeasure( simu_points$par[1:i,],model = kmModel,threshold = threshold,
                                 batchsize = i,integration.param = integration.param  )
}
plot(EDM_mmLHS,type='l',main="Expected distance in measure",
     xlab="batchsize",ylab="EDM",
     ylim=range(EDM_mmLHS,EDM_optB))
lines(EDM_optB,col=2,lty=2)
legend("topright",c("Maximin LHS","B"),lty=c(1,2),col=c(1,2))

# Get optimized simulation points with algorithm A
simu_pointsA <- optim_dist_measure(model=kmModel,threshold = threshold,
                                   lower = c(0,0),upper = c(1,1),
                                   batchsize = batchsize,algorithm = "A")
# plot the criterion value
plot(1:batchsize,simu_pointsA$value,type='l',main="Criterion value")

# Compute expected distance in measure for approximation obtained from optimized simulation points
EDM_optA <- rep(NA,batchsize)
for(i in seq(1,batchsize)){
  EDM_optA[i]<-expDistMeasure( simu_pointsA$par[1:i,],model = kmModel,threshold = threshold,
                                 batchsize = i,integration.param = integration.param  )
}
plot(EDM_mmLHS,type='l',main="Expected distance in measure",
     xlab="batchsize",ylab="EDM",
     ylim=range(EDM_mmLHS,EDM_optB,EDM_optA))
lines(EDM_optB,col=2,lty=2)
lines(EDM_optA,col=3,lty=3)
legend("topright",c("Maximin LHS","A","B"),lty=c(1,3,2),col=c(1,3,2))


## End(Not run)
```

grad_kweights — *Gradient of the weights for interpolating simulations*

Description

Returns a list with the gradients of the posterior mean and the gradient of the (ordinary) kriging weights for simulations points.

Usage

```
grad_kweights(object, simu_points, krig_points, T.mat = NULL,
  F.mat = NULL)
```

Arguments

`object`	km object
`simu_points`	simulations points, locations where the field was simulated.
`krig_points`	one point where the interpolation is computed.
`T.mat`	a matrix (n+p)x(n+p) representing the Choleski factorization of the covariance matrix for the initial design and simulation points.
`F.mat`	a matrix (n+p)x(fdim) representing the evaluation of the model matrix at the initial design and simulation points.

Value

A list containing the gradients of posterior mean and kriging weights for simulation points.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
#######################################################################
### Compute the weights and gradient on a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
```

```
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=50,
                                                                  dimension = 2,
                                                                 seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))
# Get simulation points
# Here they are not optimized, you can use optim_dist_measure to find optimized points
set.seed(1)
simu_points <- matrix(runif(100*d),ncol=d)
# obtain nsims posterior realization at simu_points
nsims <- 1
set.seed(2)
some.simu <- DiceKriging::simulate(object=kmModel,nsim=nsims,newdata=simu_points,nugget.sim=1e-6,
                         cond=TRUE,checkNames = FALSE)
nn_data<-expand.grid(seq(0,1,,50),seq(0,1,,50))
nn_data<-data.frame(nn_data)
colnames(nn_data)<-colnames(kmModel@X)
obj<-krig_weight_GPsimu(object = kmModel,simu_points = simu_points,krig_points = as.matrix(nn_data))


## Plot the approximate process realization and the gradient vector field
k_scale<-5e-4
image(matrix(obj$krig.mean.init+crossprod(obj$Lambda.end,some.simu[1,]),ncol=50),
      col=grey.colors(20))
contour(matrix(obj$krig.mean.init+crossprod(obj$Lambda.end,some.simu[1,]),ncol=50),
        nlevels = 20,add=TRUE)

for(c_ii in c(1,seq(10,2500,by = 64))){
   pp<-t(as.matrix(nn_data)[c_ii,])
  obj_deriv <- grad_kweights(object = kmModel,simu_points = simu_points,krig_points = pp)
   S_der<-obj_deriv$krig.mean.init + crossprod(obj_deriv$Lambda.end,some.simu[1,])
   points(x = pp[1],y = pp[2],pch=16)
   arrows(x0=pp[1],y0=pp[2],x1 = pp[1]+k_scale*S_der[1,1],y1=pp[2]+k_scale*S_der[2,1])
}
```

`integrand_edm_crit` *Integrand of the distance in measure criterion*

Description

Computes the integrand of the distance in measure criterion.

Usage

```
integrand_edm_crit(x, E, model, Thresh, batchsize, alpha, predE,
  predx = NULL, precalc.data = NULL)
```

Arguments

x	vector of dimension d representing the ith point where to compute the criterion
E	matrix of dimension $d * (i-1)$ containing the previously optimized simulation points
model	km model
Thresh	threshold selected for excursion set
batchsize	number of simulation points
alpha	value of Vorob'ev threshold
predE	list containing the posterior mean and standard deviation at E
predx	list containing the posterior mean and standard deviation at x
precalc.data	list result of precomputeUpdateData with `model` and `x`.

Value

the value of the integrand at x

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

---

krig_weight_GPsimu *Weights for interpolating simulations*

Description

Returns a list with the posterior mean and the kriging weights for simulations points.

Usage

```
krig_weight_GPsimu(object, simu_points, krig_points, T.mat = NULL,
  F.mat = NULL)
```

Arguments

`object`	km object.
`simu_points`	simulations points, locations where the field was simulated.
`krig_points`	points where the interpolation is computed.
`T.mat`	a matrix (n+p)x(n+p) representing the Choleski factorization of the covariance matrix for the initial design and simulation points.
`F.mat`	a matrix (n+p)x(fdim) representing the evaluation of the model matrix at the initial design and simulation points.

Value

A list containing the posterior mean and the (ordinary) kriging weights for simulation points.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
#######################################################################
### Compute the weights for approximating process on a 1d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
## Create kriging model from GP realization
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=20,
                                                                  dimension = 1,
                                                                  seed=42)$design)$design
colnames(design)<-c("x1")
gp0 <- DiceKriging::km (formula = ~1, design = design,
                        response = rep (x = 0, times = nrow (design)),
                        covtype = "matern3_2", coef.trend = 0,
                        coef.var = 1, coef.cov = 0.2)
set.seed(1)
observations <- t (DiceKriging::simulate (object = gp0, newdata = design, cond = FALSE))

# Fit OK km model
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))
```

```
# Get simulation points
# Here they are not optimized, you can use optim_dist_measure to find optimized points
set.seed(2)
simu_points <- matrix(runif(20),ncol=1)
# obtain nsims posterior realization at simu_points
nsims <- 10
set.seed(3)
some.simu <- DiceKriging::simulate(object=kmModel,nsim=nsims,newdata=simu_points,nugget.sim=1e-6,
                         cond=TRUE,checkNames = FALSE)
grid<-seq(0,1,,100)
nn_data<-data.frame(grid)
colnames(nn_data)<-colnames(kmModel@X)
pred_nn<-DiceKriging::predict.km(object = kmModel,newdata = nn_data,type = "UK")
obj <- krig_weight_GPsimu(object=kmModel,simu_points=simu_points,krig_points=grid)


# Plot the posterior mean and some approximate process realizations
result <- matrix(nrow=nsims,ncol=length(grid))

plot(nn_data$x1,pred_nn$mean,type='l')
for(i in 1:nsims){
   some.simu.i <- matrix(some.simu[i,],ncol=1)
   result[i,] <- obj$krig.mean.init + crossprod(obj$Lambda.end,some.simu.i)
   points(simu_points,some.simu.i)
   lines(grid,result[i,],col=3)
}
```

`max_distance_measure` *Minimize the distance in measure criterion*

Description

Optimizes the distance in measure criterion.

Usage

```
max_distance_measure(lower, upper, optimcontrol = NULL, batchsize,
  integration.param, T, model)
```

Arguments

`lower`	a d dimensional vector containing the lower bounds for the optimization
`upper`	a d dimensional vector containing the upper bounds for the optimization
`optimcontrol`	the parameters for the optimization, see max_sur_parallel for more details.
`batchsize`	number of simulations points to find
`integration.param`	the parameters for the integration of the criterion, see max_sur_parallel for more details.

`T`	threshold value
`model`	a km model

Value

A list containing

- `par` a matrix `batchsize*d` containing the optimal points
- `value` if `optimcontrol$optim.option!=1` and `optimcontrol$method=="genoud"` (default options) a vector of length `batchsize` containing the optimum at each step otherwise the value of the criterion at the optimum.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

`max_integrand_edm` *Maximize the integrand distance in measure criterion*

Description

Optimizes the integrand of the distance in measure criterion.

Usage

```
max_integrand_edm(lower, upper, batchsize, alpha = 0.5, Thresh, model,
  verb = 1)
```

Arguments

`lower`	a d dimensional vector containing the lower bounds for the optimization
`upper`	a d dimensional vector containing the upper bounds for the optimization
`batchsize`	number of simulations points to find
`alpha`	value of Vorob'ev threshold
`Thresh`	threshold value
`model`	a km model
`verb`	an integer to choose the level of verbosity

Value

A list containing

- `par` a matrix `batchsize*d` containing the optimal points
- `value` a vector of length `batchsize` with the value of the criterion after each optimization
- `fcount` count of the number of criterion evaluations

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

`optim_dist_measure` *Choose simulation points*

Description

Selects `batchsize` locations where to simulate the field by minimizing the distance in measure criterion or by maximizing the integrand of the distance in measure criterion. Currently it is only a wrapper for the functions `max_distance_measure` and `max_integrand_edm`.

Usage

```
optim_dist_measure(model, threshold, lower, upper, batchsize,
  algorithm = "B", alpha = 0.5, verb = 1, optimcontrol = NULL,
  integration.param = NULL)
```

Arguments

`model`	a km model
`threshold`	threshold value
`lower`	a d dimensional vector containing the lower bounds for the optimization
`upper`	a d dimensional vector containing the upper bounds for the optimization
`batchsize`	number of simulations points to find
`algorithm`	type of algorithm used to obtain simulation points: • `"A"` minimize the full integral criterion; • `"B"` maximize the integrand of the criterion.
`alpha`	value of Vorob'ev threshold
`verb`	an integer to choose the level of verbosity
`optimcontrol`	a list containing optional parameters for the optimization, see max_sur_parallel for more details.
`integration.param`	a list containing parameters for the integration of the criterion A, see max_sur_parallel for more details.

Value

A list containing

- `par` a matrix `batchsize*d` containing the optimal points
- `value` a vector of length `batchsize` with the values of the criterion at each step

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
### Compute optimal simulation points in a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=20,
                                                                  dimension = 2,
                                                                 seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))

# Run optim_dist_measure, algorithm B to obtain one simulation point
# NOTE: the approximating process resulting from 1 simulation point
# is very rough and it should not be used, see below for a more principled example.
simu_pointsB <- optim_dist_measure(model=kmModel,threshold = -10,
                                  lower = c(0,0),upper = c(1,1),
                                  batchsize = 1,algorithm = "B")

## Not run:
# Get 75 simulation points with algorithm A
batchsize <- 50
simu_pointsA <- optim_dist_measure(model=kmModel,threshold = -10,
                                  lower = c(0,0),upper = c(1,1),
```

```
                                  batchsize = batchsize,algorithm = "A")

# Get 75 simulation points with algorithm B
batchsize <- 75
simu_pointsB <- optim_dist_measure(model=kmModel,threshold = -10,
                                  lower = c(0,0),upper = c(1,1),
                                  batchsize = batchsize,algorithm = "B")
# plot the criterion value
critValA <-c(simu_pointsA$value,rep(NA,25))
par(mar = c(5,5,2,5))
plot(1:batchsize,critValA,type='l',main="Criterion value",ylab="Algorithm A",xlab="batchsize")
par(new=T)
plot(1:batchsize,simu_pointsB$value, axes=F, xlab=NA, ylab=NA,col=2,lty=2,type='l')
axis(side = 4)
mtext(side = 4, line = 3, 'Algorithm B')
legend("topright",c("Algorithm A","Algorithm B"),lty=c(1,2),col=c(1,2))
par(mar= c(5, 4, 4, 2) + 0.1)

# obtain nsims posterior realization at simu_points
nsims <- 1
nn_data<-expand.grid(seq(0,1,,50),seq(0,1,,50))
nn_data<-data.frame(nn_data)
colnames(nn_data)<-colnames(kmModel@X)
approx.simu <- simulate_and_interpolate(object=kmModel, nsim = 1, simupoints = simu_pointsA$par,
                                        interpolatepoints = as.matrix(nn_data),
                                        nugget.sim = 0, type = "UK")

## Plot the approximate process realization
image(matrix(approx.simu[1,],ncol=50),
      col=grey.colors(20))
contour(matrix(approx.simu[1,],ncol=50),
        nlevels = 20,add=TRUE)
points(simu_pointsA$par,pch=17)
points(simu_pointsB$par,pch=1,col=2)


## End(Not run)
```

---

pGPx *Pseudo-realizations for GP eXcursion*

Description

Generates posterior pseudo-realizations of Gaussian processes for excursion set estimation. The package provides posterior pseudo-realizations over large designs by simulating the field at few well chosen points and interpolating the result. The points are chosen minimizing the (posterior) expected distance in measure between the approximate excursion set and the full excursion set. The main functions in the package are:

Approximation:
- `optim_dist_measure`: Given a km objects computes the optimal simulation points e_1, ... , e_m according to algorithm A or B.

- `krig_weight_GPsimu`: Given the simulations points and the interpolation points computes the kriging weights for the approximate process $\tilde{Z}$ at the interpolation points.
- `grad_kweights`: Given the simulations points and the interpolation points returns the gradient of kriging weights with respect to the interpolation points.
- `expDistMeasure`: computes the expected distance in measure between the excursion set of the approximated process and the true excursion set.

Simulation:
- `simulate_and_interpolate`: Generates `nsims` approximate posterior field realizations at `interpolatepoints` given the optimized simulation points.

Applications:
- *Contour length*: the function `compute_contourLength` computes the excursion set contour length for each GP realization.
- *Distance transform*: the function `dtt_fast` computes the distance transform of a binary image (Felzenszwalb and Huttenlocher, 2012) and the function `DTV` computes the distance transfom variability.
- *Volumes*: the function `computeVolumes` computes the excursion volumes for each GP realization. It also applies a bias correction for approximate realizations.

Details

Package: pGPx
Type: Package
Version: 0.1.1
Date: 2018-08-20

Note

This work was supported in part by the Swiss National Science Foundation, grant numbers 146354, 167199 and the Hasler Foundation, grant number 16065. The author wishes to thank David Ginsbourger, Clément Chevalier and Julien Bect for the fruitful discussions.

Author(s)

Dario Azzimonti (IDSIA, Switzerland, dario.azzimonti@gmail.com) with contributions from Julien Bect (CentraleSupélec, France). The function `dtt_fast` is a wrapper for C++ code by Pedro Felzenszwalb.

References

Azzimonti, D., Bect, J., Chevalier, C., and Ginsbourger, D. (2016a). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Azzimonti, D. and Ginsbourger, D. (2017). Estimating orthant probabilities of high dimensional Gaussian vectors with an application to set estimation. Journal of Computational and Graphical Statistics.

Bolin, D. and Lindgren, F. (2015). Excursion and contour uncertainty regions for latent Gaussian models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 77(1):85–106.

Chevalier, C. (2013). Fast uncertainty reduction strategies relying on Gaussian process models. PhD thesis, University of Bern.

Chevalier, C., Bect, J., Ginsbourger, D., Vazquez, E., Picheny, V., and Richet, Y. (2014). Fast kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. Technometrics, 56(4):455–465.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance Transforms of Sampled Functions. Theory of Computing, 8(19):415-428.

`simulate_and_interpolate`

Simulate and interpolate

Description

Generates `nsims` approximate posterior field realizations at `interpolatepoints`. The approximate realizations are computed by simulating the field only at `simupoints` simulation points.

Usage

```
simulate_and_interpolate(object, nsim = 1, simupoints = NULL,
  interpolatepoints = NULL, nugget.sim = 0, type = "UK")
```

Arguments

`object`	km object
`nsim`	numbero of simulations
`simupoints`	simulations points, locations where the field was simulated
`interpolatepoints`	points where posterior simulations are approximated
`nugget.sim`	nugget to be added to simulations for numerical stability
`type`	type of kriging model used for approximation (default Universal Kriging)

Value

A matrix `nsim*interpolatepoints` containing the approximate realizations.

References

Azzimonti D. F., Bect J., Chevalier C. and Ginsbourger D. (2016). Quantifying uncertainties on excursion sets under a Gaussian random field prior. SIAM/ASA Journal on Uncertainty Quantification, 4(1):850–874.

Azzimonti, D. (2016). Contributions to Bayesian set estimation relying on random field priors. PhD thesis, University of Bern.

Examples

```
### Simulate and interpolate for a 2d example
if (!requireNamespace("DiceKriging", quietly = TRUE)) {
stop("DiceKriging needed for this example to work. Please install it.",
     call. = FALSE)
}
if (!requireNamespace("DiceDesign", quietly = TRUE)) {
stop("DiceDesign needed for this example to work. Please install it.",
     call. = FALSE)
}
# Define the function
g=function(x){
  return(-DiceKriging::branin(x))
}
d=2
# Fit OK km model
design<-DiceDesign::maximinESE_LHS(design = DiceDesign::lhsDesign(n=50,
                                                                  dimension = 2,
                                                                 seed=42)$design)$design
colnames(design)<-c("x1","x2")
observations<-apply(X = design,MARGIN = 1,FUN = g)
kmModel<-DiceKriging::km(formula = ~1,design = design,response = observations,
                         covtype = "matern3_2",control=list(trace=FALSE))
# Get simulation points
# Here they are not optimized, you can use optim_dist_measure to find optimized points
simu_points <- DiceDesign::maximinSA_LHS(DiceDesign::lhsDesign(n=100,
                                                               dimension = d,
                                                               seed=1)$design)$design
# obtain nsims posterior realization at simu_points
nsims <- 1
nn_data<-expand.grid(seq(0,1,,50),seq(0,1,,50))
nn_data<-data.frame(nn_data)
colnames(nn_data)<-colnames(kmModel@X)
approx.simu <- simulate_and_interpolate(object=kmModel, nsim = 1, simupoints = simu_points,
                                        interpolatepoints = as.matrix(nn_data),
                                        nugget.sim = 0, type = "UK")

## Plot the approximate process realization
image(matrix(approx.simu[1,],ncol=50),
      col=grey.colors(20))
contour(matrix(approx.simu[1,],ncol=50),
        nlevels = 20,add=TRUE)
```

Index