Package 'MuFiMeshGP'

May 7, 2026

Type Package

Title Multi-Fidelity Emulator for Computer Experiments with Tunable Fidelity Levels

Version 0.0.1

Date 2025-08-11

Description Multi-Fidelity emulator for data from computer simulations of the same underlying system but at different input locations and fidelity level, where both the input locations and fidelity level can be continuous. Active Learning can be performed with an implementation of the Integrated Mean Square Prediction Error (IMSPE) criterion developed by Boutelet and Sung (2025, <doi:10.48550/arXiv.2503.23158>).

License LGPL (>= 2)

Encoding UTF-8

Imports lhs, parallel, methods, Rcpp

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

NeedsCompilation yes

Author Romain Boutelet [aut, cre], Chih-Li Sung [aut]

Maintainer Romain Boutelet <boutelet@msu.edu>

Repository CRAN

Date/Publication 2025-09-01 09:50:16 UTC

Contents

cov_gen *Generates the covariance matrix*

Description

Generates the covariance matrix for the Gaussian kernel or Matern kernel

Usage

```
cov_gen(
  x1,
  x2 = NULL,
  t1,
  t2 = NULL,
  phi1sq,
  phi2sq,
  sigma1sq,
  sigma2sq,
  H,
  l = 4,
  covtype,
  iso,
  nugget = sqrt(.Machine$double.eps)
)
```

Arguments

x1, x2	Design input location matrices. x2 is to be used only to create cross-covariance matrix.
t1, t2	Design tunable parameter vectors. t2 is to be used only to create cross-covariance matrix.
phi1sq, phi2sq, sigma1sq, sigma2sq, H, l	hyper-parameters for the covariance function
covtype	kernel function used: "Gaussian" is the only available one at the moment.
iso	If TRUE, then the covariance function is isotropic. If FALSE, the covariance function is anisotropic.
nugget	optional

Value

a matrix of size (nrow(x1), nrow(x2)), or (nrow(x1), nrow(x1)) if x2 and t2 are NULL.

IMSPE_AL *IMSPE optimal design point search*

Description

Search for the best next design point according to the IMSPE criterion given a current MuFiMeshGP model fit.

Usage

```
IMSPE_AL(
  object,
  t.min,
  t.max,
  cost.func,
  cost.new = 0,
  gr = FALSE,
  gr_cost.func = NULL,
  DesCand = NULL,
  Wijs = NULL,
  Hijs = NULL,
 control = list(multi.start.n = 20, maxit = 20, DesStart = NULL, seed = NULL, ncores =
    1)
)
```

Arguments

object	Current MuFiMeshGP model fit.
t.min, t.max	Lower and upper bounds on the fidelity space for the search.
cost.func	Function that maps the tunable parameter t to the corresponding cost running a simulation at that fidelity level. For example, function(t) 1/t^2.
cost.new	(optional) Cost of running a new simulation at a new fidelity level, scalar.
gr	whether the gradient should be used in the optimization of the IMSPE. (Not recommended due to numerical errors)
gr_cost.func	If grad is TRUE, the user needs to specify the gradient of the cost function, as a function.
DesCand	Design candidates to evaluate from.
Wijs, Hijs	(optional) Matrices from previous IMSPE search to obtain faster computation through matrix decomposition.
control	list of arguments udes for the optimization.

Value

a list with:

- x: the optimal input parameter, a vector.
- t: the optimal tuning parameter, a scalar.
- value: the IMSPE reduction at the optimal design location.
- new: whether the optimal tuning parameter defines a new fidelity level.
- id: the index of the optimal design location if DesCand is used.

Examples

```
# Example code

f <- function(x, t){
  x <- c(x)
  return(exp(-1.4*x)*cos(3.5*pi*x)+sin(40*x)/10*t^2)
}

set.seed(1)
X <- matrix(runif(15,0,1), ncol = 1)
tt <- runif(15,0.5,2)

Y <- f(c(X), tt)

fit.mufimeshgp <- MuFiMeshGP(X, tt, Y)

xx <- matrix(seq(0,1,0.01), ncol = 1)
ftrue <- f(xx, 0)

# predict
pred.mufimeshgp <- predict(fit.mufimeshgp, xx, rep(0,101))

mu <- pred.mufimeshgp$mean
s <- pred.mufimeshgp$sd
lower <- mu + qnorm(0.025)*s
upper <- mu + qnorm(0.975)*s

# plot

oldpar <- par(mfrow = c(1,2))
plot(xx, ftrue, "l", ylim = c(-1,1.3), ylab = "y", xlab = "x")
lines(c(xx), mu, col = "blue")
lines(c(xx), lower, col = "blue", lty = 2)
lines(c(xx), upper, col = "blue", lty = 2)
points(c(X), Y, col = "red")

### RMSE ###
print(sqrt(mean((ftrue - mu))^2))

best <- IMSPE_AL(fit.mufimeshgp, 0.5, 2, function(t) return(1 / t^2))
new.Y <- f(best$x, best$t)
```

```
fit.mufimeshgp <- update(fit.mufimeshgp, best$x, best$t, new.Y)

pred.mufimeshgp <- predict(fit.mufimeshgp, xx, rep(0, 101))
mu <- pred.mufimeshgp$mean
s <- pred.mufimeshgp$sd
lower <- mu + qnorm(0.025)*s
upper <- mu + qnorm(0.975)*s

plot(xx, ftrue, "l", ylim = c(-1,1.3), ylab = "y", xlab = "x")
lines(c(xx), mu, col = "blue")
lines(c(xx), lower, col = "blue", lty = 2)
lines(c(xx), upper, col = "blue", lty = 2)
points(c(X), Y, col = "red")
points(c(best$x), new.Y, col = "green")
par(oldpar)

### RMSE ###
print(sqrt(mean((ftrue - mu))^2))
```

`MuFiMeshGP` *Prediction of the MuFiMeshGP emulator for any fidelity level.*

Description

The function computes the posterior mean and standard deviation of the MuFiMeshGP model.

Usage

```
MuFiMeshGP(
  X,
  t,
  Y,
  covtype = "Gaussian",
  trend.type = "OK",
  trend.dim = "input",
  trend.pol = "quadratic",
  interaction = NULL,
  mean.known = NULL,
  H.known = NULL,
  gradient = TRUE,
  init = NULL,
  single_fidelity = FALSE,
  param.bounds = NULL,
  iso = FALSE,
  l = 4,
  nugget = 1e-06,
  ncores = 1
)
```

Arguments

X	matrix of input locations. Each row represents a sample.
t	vector of fidelity levels. Each element is a sample and is connected to the corresponding row in X.
Y	vector of response values.
covtype	covariance kernel type, only 'Gaussian' is available for now, 'Matern5_2' or 'Matern3_2' will be available soon (see cov_gen).
trend.type, trend.dim, trend.pol, interaction	define the mean function form of the Gaussian process. trend.type can be: "SK" in which case mean.known needs to be specified as a scalar; "OK" in which case the constant mean will be evaluated through MLE; "UK" in which case trend.dim specifies whether the trend will be along the input space ("input"), the fidelity space ("fidelity"), or both ("both"). If trend.dim is "input" or "both", the user can use the trend.pol to specify if the trend on the input space alone should be "linear" or "quadratic}. Finally, if \code{trend.dim} is \code{"both"}, then an \code{interaction} term specify the polynomial order (\code{"linear or "quadratic") of the input space trend that is multiplied to the fidelity space trend. See regF_gen for further details.
mean.known	Specifies the mean if "SK" as trend.type, scalar.
H.known	allow the user to specify the value of H as H.known, a scalar in (0,1).
gradient	whether or not the gradient of the log-likelihood shouldbe used in the parameter estimation.
init	Where should the parameter estimation start from, a vector.
single_fidelity	can be used as TRUE to use MuFiMeshGP as a single fidelity Gaussian Process. This will set sigma2sq as 0.
param.bounds	a list with two arguments(lower and upper) describing the bounds used for MLE optimization of phi1sq and phi2sq. Each argument should be a vector of length ncol(X). If NULL the bounds of phi1sq and phi2sq are specified automatically from the design matrix.
iso	whether the covariance function will be isotropic (TRUE or FALSE)
l	rate of convergence of the system (see Details), scalar.
nugget	(optional) for controlling numerical error.
ncores	(optional) number of cores for parallelization.

Details

From the model fitted by MuFiMeshGP or update.MuFiMeshGP the posterior mean and standard deviation are calculated for any input location and fidelity level. For details, see Boutelet and Sung (2025, <arXiv:2503.23158>).

Value

a list which is given the S3 class "MuFiMeshGP"

See Also

MuFiMeshGP for the model.

Examples

```
# Example code

f <- function(x, t){
  x <- c(x)
  return(exp(-1.4*x)*cos(3.5*pi*x)+sin(40*x)/10*t^2)
}

set.seed(1)
X <- matrix(runif(15,0,1), ncol = 1)
tt <- runif(15,0.5,2)

Y <- f(c(X), tt)

fit.mufimeshgp <- MuFiMeshGP(X, tt, Y)

xx <- matrix(seq(0,1,0.01), ncol = 1)
ftrue <- f(xx, 0)

# predict
pred.mufimeshgp <- predict(fit.mufimeshgp, xx, rep(0,101))

mu <- pred.mufimeshgp$mean
s <- pred.mufimeshgp$sd
lower <- mu + qnorm(0.025)*s
upper <- mu + qnorm(0.975)*s

# plot

oldpar <- par(mfrow = c(1,1))
plot(xx, ftrue, "l", ylim = c(-1,1.3), ylab = "y", xlab = "x")
lines(c(xx), mu, col = "blue")
lines(c(xx), lower, col = "blue", lty = 2)
lines(c(xx), upper, col = "blue", lty = 2)
points(c(X), Y, col = "red")
par(oldpar)

### RMSE ###
print(sqrt(mean((ftrue - mu))^2))
```

---

predict.MuFiMeshGP *predict.MuFiMeshGP*

Description

The function computes the posterior mean and standard deviation of the MuFiMeshGP model.

Usage

```
## S3 method for class 'MuFiMeshGP'
predict(object, x, t, ...)
```

Arguments

object	an object of class MuFiMeshGP.
x	matrix of new input locations to predict.
t	vector or new fidelity levels to use for predictions.
...	no other argument.

Details

Prediction of the MuFiMeshGP emulator for any fidelity level.

From the object fitted by MuFiMeshGP or update.MuFiMeshGP the posterior mean and standard deviation are calculated for any input location and fidelity level. For details, see Boutelet and Sung (2025, <arXiv:2503.23158>).

Value

- mean: vector of predictive posterior mean.
- sd: vector of predictive posterior standard deviation.

See Also

MuFiMeshGP for the model

Examples

```
# Example code
f <- function(x, t){
  x <- c(x)
  return(exp(-1.4*x)*cos(3.5*pi*x)+sin(40*x)/10*t^2)
}

set.seed(1)
X <- matrix(runif(15,0,1), ncol = 1)
tt <- runif(15,0.5,2)

Y <- f(c(X), tt)

fit.mufimeshgp <- MuFiMeshGP(X, tt, Y)

xx <- matrix(seq(0,1,0.01), ncol = 1)
ftrue <- f(xx, 0)

# predict
pred.mufimeshgp <- predict(fit.mufimeshgp, xx, rep(0,101))
```

```
mu <- pred.mufimeshgp$mean
s <- pred.mufimeshgp$sd
lower <- mu + qnorm(0.025)*s
upper <- mu + qnorm(0.975)*s

# plot

oldpar <- par(mfrow = c(1,1))
plot(xx, ftrue, "l", ylim = c(-1,1.3), ylab = "y", xlab = "x")
lines(c(xx), mu, col = "blue")
lines(c(xx), lower, col = "blue", lty = 2)
lines(c(xx), upper, col = "blue", lty = 2)
points(c(X), Y, col = "red")
par(oldpar)

### RMSE ###
print(sqrt(mean((ftrue - mu))^2))
```

`regF_gen` *Creates the regression function for the mean*

Description

Creates the regression function for the GP mean.

Usage

```
regF_gen(trend.dim, trend.pol, interaction, l, d)
```

Arguments

`trend.dim`	which dimension should the trend follow: `"input"`, `"fidelity"`, or `"both"`.
`trend.pol`	Which polynomial degree should the input mean trend have: `"linear"` or `"quadratic"`.
`interaction`	polynomial degree of the interaction between input trend and fidelity trend of: `NULL`, `"linear"`, or `"quadratic"`. `"linear"` or `"quadratic"`.
`l`	convergence rate parameter, usually `l = 4`.
`d`	input space dimension

Value

a function

`update.MuFiMeshGP` *update.MuFiMeshGP*

Description

The function updates the current `MuFiMeshGP` model.

Usage

```
## S3 method for class 'MuFiMeshGP'
update(object, x, t, y, param.estim = TRUE, init = NULL, ...)
```

Arguments

`object`	an object of class `MuFiMeshGP`.
`x`	matrix of new input locations.
`t`	new tunable parameter, a scalar.
`y`	observation corresponding to input location `x` and tunable parameter `t`.
`param.estim`	if `TRUE`, the hyper-parameters are estimated by running it through `MuFiMeshGP`. If `FALSE`, the hyper-parameters from `object` are used to update the `MuFiMeshGP` model fit.
`init`	See `MuFiMeshGP`.
`...`	no other argument.

Details

Updates the `MuFiMeshGP` model fit with new observations

From the model fitted by `MuFiMeshGP` or `update.MuFiMeshGP` the posterior mean and standard deviation are calculated for any input location and fidelity level. For details, see Boutelet and Sung (2025, <arXiv:2503.23158>).

Value

a list which is given the S3 class "MuFiMeshGP"

See Also

`MuFiMeshGP` for initializing the model.

Examples

```
# Example code

f <- function(x, t){
  x <- c(x)
  return(exp(-1.4*x)*cos(3.5*pi*x)+sin(40*x)/10*t^2)
}

set.seed(1)
X <- matrix(runif(15,0,1), ncol = 1)
tt <- runif(15,0.5,2)

Y <- f(c(X), tt)

fit.mufimeshgp <- MuFiMeshGP(X, tt, Y)

xx <- matrix(seq(0,1,0.01), ncol = 1)
ftrue <- f(xx, 0)

# predict
pred.mufimeshgp <- predict(fit.mufimeshgp, xx, rep(0,101))

mu <- pred.mufimeshgp$mean
s <- pred.mufimeshgp$sd
lower <- mu + qnorm(0.025)*s
upper <- mu + qnorm(0.975)*s

# plot

oldpar <- par(mfrow = c(1,2))
plot(xx, ftrue, "l", ylim = c(-1,1.3), ylab = "y", xlab = "x")
lines(c(xx), mu, col = "blue")
lines(c(xx), lower, col = "blue", lty = 2)
lines(c(xx), upper, col = "blue", lty = 2)
points(c(X), Y, col = "red")

### RMSE ###
print(sqrt(mean((ftrue - mu))^2))

best <- IMSPE_AL(fit.mufimeshgp, 0.5, 2, function(t) return(1 / t^2))
new.Y <- f(best$x, best$t)
fit.mufimeshgp <- update(fit.mufimeshgp, best$x, best$t, new.Y)

pred.mufimeshgp <- predict(fit.mufimeshgp, xx, rep(0, 101))
mu <- pred.mufimeshgp$mean
s <- pred.mufimeshgp$sd
lower <- mu + qnorm(0.025)*s
upper <- mu + qnorm(0.975)*s

plot(xx, ftrue, "l", ylim = c(-1,1.3), ylab = "y", xlab = "x")
lines(c(xx), mu, col = "blue")
lines(c(xx), lower, col = "blue", lty = 2)
```

```
lines(c(xx), upper, col = "blue", lty = 2)
points(c(X), Y, col = "red")
points(c(best$x), new.Y, col = "green")

par(oldpar)

### RMSE ###
print(sqrt(mean((ftrue - mu))^2))
```

Index