Package: ARCokrig (via r-universe)

October 23, 2026

Type Package

Title Autoregressive Cokriging Models for Multifidelity Codes

Version 0.1.3

Date 2026-02-25

Maintainer Pulong Ma <mpulong@gmail.com>

Description For emulating multifidelity computer models. The major methods include univariate autoregressive cokriging and multivariate autoregressive cokriging. The autoregressive cokriging methods are implemented for both hierarchically nested design and non-nested design. For hierarchically nested design, the model parameters are estimated via standard optimization algorithms; For non-nested design, the model parameters are estimated via Monte Carlo expectation-maximization (MCEM) algorithms. In both cases, the priors are chosen such that the posterior distributions are proper. Notice that the uniform priors on range parameters in the correlation function lead to improper posteriors. This should be avoided when Bayesian analysis is adopted. The development of objective priors for autoregressive cokriging models can be found in Pulong Ma (2020) <DOI:10.1137/19M1289893>. The development of the multivariate autoregressive cokriging models with possibly non-nested design can be found in Pulong Ma, Georgios Karagiannis, Bledar A Konomi, Taylor G Asher, Gabriel R Toro, and Andrew T Cox (2022) <DOI:10.1111/rssc.12558>.

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/pulongma/ARCokrig/issues

Depends R (>= 3.5.0)

Imports Rcpp, mvtnorm (>= 1.0-10), stats, methods, ggplot2

LinkingTo Rcpp, RcppArmadillo, RcppEigen

RoxygenNote 6.1.1

NeedsCompilation yes

Author Pulong Ma [aut, cre]

Repository https://cran.r-universe.dev

Date/Publication 2026-03-01 22:30:02 UTC

RemoteUrl https://github.com/cran/ARCokrig

RemoteRef HEAD

RemoteSha 9e465d934b0e79fd860239171c28ed2be9cba398

Contents

ARCokrig 2
cokm 5
cokm-class 7
cokm.condsim 8
cokm.fit 9
cokm.param 11
cokm.predict 11
CRPS 13
mvcokm 13
mvcokm-class 15
mvcokm.condsim 16
mvcokm.fit 17
mvcokm.param 17
mvcokm.predict 18

Index **19**

`ARCokrig` *Fit the AR-Cokriging model and make predictions*

Description

This is a simple and high-level funciton to fit autoregressive cokriging models to multifidelity computer model outputs.

Usage

```
ARCokrig(
  formula = list(~1, ~1),
  output,
  input,
  cov.model = "matern_5_2",
  nugget.est = FALSE,
  input.new,
  prior = list(),
```

```
  opt = list(),
  NestDesign = TRUE,
  tuning = list(),
  info = list()
)
```

Arguments

formula	a list of s elements, each of which contains the formula to specify fixed basis functions or regressors.
output	a list of s elements, each of which contains a matrix of computer model outputs.
input	a list of s elements, each of which contains a matrix of inputs.
cov.model	a string indicating the type of covariance function in AR-cokriging models. Current covariance functions include **exp** product form of exponential covariance functions. **matern_3_2** product form of Matern covariance functions with smoothness parameter 3/2. **matern_5_2** product form of Matern covariance functions with smoothness parameter 5/2. **Gaussian** product form of Gaussian covariance functions. **powexp** product form of power-exponential covariance functions with roughness parameter fixed at 1.9.
nugget.est	a logical value indicating whether nugget parameter is included or not. Default value is FALSE.
input.new	a matrix including new inputs for making prediction
prior	a list of arguments to setup the prior distributions **name** the name of the prior. Current implementation includes JR, Reference, Jeffreys, Ind_Jeffreys **hyperparam** hyperparameters in the priors. For jointly robust (JR) prior, three parameters are included: a refers to the polynomial penalty to avoid singular correlation matrix with a default value 0.2; b refers to the exponenetial penalty to avoid diagonal correlation matrix with a default value 1; nugget.UB is the upper bound of the nugget variance with default value 1, which indicates that the nugget variance has support (0, 1).
opt	a list of arguments to setup the optim routine.
NestDesign	a logical value indicating whether the experimental design is hierarchically nested within each level of the code.
tuning	a list of arguments to control the MCEM algorithm for non-nested design. It includes the arguments **maxit** the maximum number of MCEM iterations. **tol** a tolerance to stop the MCEM algorithm. If the parameter difference between any two consecutive MCEM algorithm is less than this tolerance, the MCEM algorithm is stopped. **n.sample** the number of Monte Carlo samples in the MCEM algorithm.

verbose a logical value to show the MCEM iterations if it is true.

`info` a list that contains

iter number of iterations used in the MCEM algorithm

eps parameter difference after the MCEM algorithm stops

Value

The main call inside `ARCokrig` consists of `cokm`, `cokm.fit`, and `cokm.predict`. Thus, the function returns the `cokm` object and predictions over new inputs.

Author(s)

Pulong Ma <mpulong@gmail.com>

References

- Ma, P. (2019). "Objective Bayesian Analysis of a Cokriging Model for Hierarchical Multifidelity Codes." arXiv:1910.10225. https://arxiv.org/abs/1910.10225.
- Ma, P., Karagiannis, G., Konomi, B., Asher, T., Toro, G., and Cox, A. (2019) "Multifidelity Computer Model Emulation with High-Dimensional Output: An Application to Storm Surge." arXiv:1909.01836. https://arxiv.org/abs/1909.01836.

See Also

`cokm`, `cokm.param`, `cokm.fit`, `cokm.predict`

Examples

```
#################################################################
#################################################################
############ Example
Funcc = function(x){
  return(0.5*(6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)
}

Funcf = function(x){
  z1 = Funcc(x)
  z2 = 2*z1-20*x+20 + sin(10*cos(5*x))
  return(z2)
}

#########################################################################
###### Nested design
#########################################################################
Dc <- seq(-1,1,0.1)
indDf <- c(1, 3, 6, 8, 10, 13, 17, 21)
zc <- Funcc(Dc)
Df <- Dc[indDf]
zf <- Funcf(Df)
```

```
input.new = as.matrix(seq(-1,1,length.out=200))


## fit and predict with the AR-Cokriging model

out = ARCokrig(formula=list(~1,~1+x1), output=list(c(zc), c(zf)),
               input=list(as.matrix(Dc), as.matrix(Df)),
               cov.model="matern_5_2",
               input.new=input.new)

## plot results

library(ggplot2)
cokrig = out$cokrig
df.l1 = data.frame(x=c(Dc), y=c(zc))
df.l2 = data.frame(x=c(Df), y=c(zf))
CI.lower = cokrig$lower95[[2]]
CI.upper = cokrig$upper95[[2]]
df.CI = data.frame(x=c(input.new),lower=CI.lower, upper=CI.upper)
df.pred = data.frame(x=c(input.new), y=cokrig$mu[[2]])

g = ggplot(data.frame(x=c(-1,1)), aes(x)) +
 stat_function(fun=Funcc, geom="line", aes(colour="level 1"), n=500) +
 stat_function(fun=Funcf, geom="line", aes(colour="level 2"), n=500)

g = g + geom_point(data=df.l1, mapping=aes(x=x, y=y), shape=16, size=2, color="black") +
 geom_point(data=df.l2, mapping=aes(x=x, y=y), shape=17, size=2, color="black")

g = g + geom_line(data=df.pred, aes(x=x, y=y, colour="cokriging"), inherit.aes=FALSE) +
 geom_ribbon(data=df.CI, mapping=aes(x=x,ymin=lower, ymax=upper), fill="gray40",
             alpha=0.3, inherit.aes=FALSE)
g = g + scale_colour_manual(name=NULL, values=c("red","blue", "turquoise3"),
                            breaks=c("cokriging","level 1", "level 2"))

g = g + ggtitle("A Two-Level Example") +
 theme(plot.title=element_text(size=14),
       axis.title.x=element_text(size=14),
       axis.text.x=element_text(size=14),
       axis.title.y=element_text(size=14),
       axis.text.y=element_text(size=14),
       legend.text = element_text(size=12),
       legend.direction = "horizontal",
       legend.position = c(0.6, 0.1)) + xlab("") + ylab("")
print(g)
```

---

cokm *Construct the cokm object*

Description

This function constructs the cokm object in autogressive cokriging models

Usage

```
cokm(
  formula = list(~1, ~1),
  output,
  input,
  cov.model = "matern_5_2",
  nugget.est = FALSE,
  prior = list(),
  opt = list(),
  NestDesign = TRUE,
  tuning = list(),
  info = list()
)
```

Arguments

formula	a list of s elements, each of which contains the formula to specify fixed basis functions or regressors.
output	a list of s elements, each of which contains a matrix of computer model outputs.
input	a list of s elements, each of which contains a matrix of inputs.
cov.model	a string indicating the type of covariance function in AR-cokriging models. Current covariance functions include **exp** product form of exponential covariance functions. **matern_3_2** product form of Matern covariance functions with smoothness parameter 3/2. **matern_5_2** product form of Matern covariance functions with smoothness parameter 5/2. **Gaussian** product form of Gaussian covariance functions. **powexp** product form of power-exponential covariance functions with roughness parameter fixed at 1.9.
nugget.est	a logical value indicating whether the nugget is included or not. Default value is FALSE.
prior	a list of arguments to setup the prior distributions with the reference prior as default. **name** the name of the prior. Current implementation includes JR, Reference, Jeffreys, Ind_Jeffreys **hyperparam** hyperparameters in the priors. For jointly robust (JR) prior, three parameters are included: a refers to the polynomial penalty to avoid singular correlation matrix with a default value 0.2; b refers to the exponenetial penalty to avoid diagonal correlation matrix with a default value 1; nugget.UB is the upper bound of the nugget variance with default value 1, which indicates that the nugget variance has support (0, 1).

`opt` a list of arguments to setup the `optim` routine.

`NestDesign` a logical value indicating whether the experimental design is hierarchically nested within each level of the code.

`tuning` a list of arguments to control the MCEM algorithm for non-nested design. It includes the arguments

maxit the maximum number of MCEM iterations.

tol a tolerance to stop the MCEM algorithm. If the parameter difference between any two consecutive MCEM algorithm is less than this tolerance, the MCEM algorithm is stopped.

n.sample the number of Monte Carlo samples in the MCEM algorithm.

verbose a logical value to show the MCEM iterations if it is true.

`info` a list that contains

iter number of iterations used in the MCEM algorithm

eps parameter difference after the MCEM algorithm stops

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

`ARCokrig`, `cokm.fit`, `cokm.predict`

`cokm-class` *cokm Class*

Description

This is an S4 class definition for `cokm` in the `ARCokrig` package

Slots

`output` a list of s elements, each of which contains a matrix of computer model outputs.

`input` a list of s elements, each of which contains a matrix of inputs.

`param` a list of s elements, each of which contains a vector of initial values for correlation parameters (and nugget variance parameters if nugget terms are included in AR-cokriging models).

`cov.model` a string indicating the type of covariance function in AR-cokriging models. Current covariance functions include

exp product form of exponential covariance functions.

matern_3_2 product form of Matern covariance functions with smoothness parameter 3/2.

matern_5_2 product form of Matern covariance functions with smoothness parameter 5/2.

Gaussian product form of Gaussian covariance functions.

powexp product form of power-exponential covariance functions with roughness parameter fixed at 1.9.

`nugget.est` a logical value indicating whether nugget parameter is included or not. Default value is `FALSE`.

`prior` a list of arguments to setup the prior distributions with the reference prior as default

name the name of the prior. Current implementation includes `JR`, `Reference`, `Jeffreys`, `Ind_Jeffreys`

hyperparam hyperparameters in the priors. For jointly robust (JR) prior, three parameters are included: a refers to the polynomial penalty to avoid singular correlation matrix with a default value 0.2; b refers to the exponenetial penalty to avoid diagonal correlation matrix with a default value 1; nugget.UB is the upper bound of the nugget variance with default value 1, which indicates that the nugget variance has support (0, 1).

`opt` a list of arguments to setup the `optim` routine.

`NestDesign` a logical value indicating whether the experimental design is hierarchically nested within each level of the code.

`tuning` a list of arguments to control the MCEM algorithm for non-nested design. It includes the arguments

maxit the maximum number of MCEM iterations.

tol a tolerance to stop the MCEM algorithm. If the parameter difference between any two consecutive MCEM algorithm is less than this tolerance, the MCEM algorithm is stopped.

n.sample the number of Monte Carlo samples in the MCEM algorithm.

verbose a logical value to show the MCEM iterations if it is true.

`info` a list that contains

iter number of iterations used in the MCEM algorithm

eps parameter difference after the MCEM algorithm stops.

Author(s)

Pulong Ma <mpulong@gmail.com>

`cokm.condsim` *Conditional simulation at new inputs in the autoregressive cokriging model*

Description

This function simulate from predictive distributions in autogressive cokriging models

Usage

```
cokm.condsim(obj, input.new, nsample = 30)
```

Arguments

`obj`	a `cokm` object construted via the function `cokm` in this package
`input.new`	a matrix including new inputs for making prediction
`nsample`	a numerical value indicating the number of samples

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

cokm, cokm.fit, cokm.predict, ARCokrig

Examples

```
Funcc = function(x){
  return(0.5*(6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)
}

Funcf = function(x){
  z1 = Funcc(x)
  z2 = 2*z1-20*x+20 + sin(10*cos(5*x))
  return(z2)
}

#########################################################################
###### Nested design
#########################################################################
Dc <- seq(-1,1,0.1)
indDf <- c(1, 3, 6, 8, 10, 13, 17, 21)
zc <- Funcc(Dc)
Df <- Dc[indDf]
zf <- Funcf(Df)

input.new = as.matrix(seq(-1,1,length.out=200))


## create the cokm object
prior = list(name="Reference")
obj = cokm(formula=list(~1,~1+x1), output=list(c(zc), c(zf)),
             input=list(as.matrix(Dc), as.matrix(Df)),
             prior=prior, cov.model="matern_5_2")

## update model parameters in the cokm object

obj = cokm.fit(obj)


cokrige = cokm.condsim(obj, input.new, nsample=30)
```

---

cokm.fit *fit the autoregressive cokriging model*

Description

This function estimates parameters in autogressive cokriging models

Usage

```
cokm.fit(obj)
```

Arguments

obj	a cokm object construted via the function cokm in this package

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

cokm, cokm.param, cokm.predict, ARCokrig

Examples

```
Funcc = function(x){
  return(0.5*(6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)
}

Funcf = function(x){
  z1 = Funcc(x)
  z2 = 2*z1-20*x+20 + sin(10*cos(5*x))
  return(z2)
}

#########################################################################
###### Nested design
#########################################################################
Dc <- seq(-1,1,0.1)
indDf <- c(1, 3, 6, 8, 10, 13, 17, 21)
zc <- Funcc(Dc)
Df <- Dc[indDf]
zf <- Funcf(Df)

input.new = as.matrix(seq(-1,1,length.out=200))


## create the cokm object
prior = list(name="JR")
obj = cokm(formula=list(~1,~1+x1), output=list(c(zc), c(zf)),
             input=list(as.matrix(Dc), as.matrix(Df)),
             prior=prior, cov.model="matern_5_2")

## update model parameters in the cokm object

obj = cokm.fit(obj)
```

cokm.param *Get model parameters in the autoregressive cokriging model*

Description

This function compute estimates for regression and variance parameters given the correlation parameters are known. It is used to show all model parameters in one place.

Usage

```
cokm.param(obj)
```

Arguments

obj	a cokm object construted via the function cokm in this package

Value

a list of model parameters including regression coefficients β, scale discrepancy γ, variance parameters σ^2, and correlation parameters ϕ in covariance functions. If nugget parameters are included in the model, then nugget parameters are shown in ϕ.

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

cokm, cokm.fit, cokm.condsim, ARCokrig

cokm.predict *Prediction at new inputs in the autoregressive cokriging model*

Description

This function makes prediction in autogressive cokriging models. If a nested design is used, the predictive mean and predictive variance are computed exactly; otherwise, Monte Carlo simulation from the predictive distribution is used to approximate the predictive mean and predictive variance.

Usage

```
cokm.predict(obj, input.new)
```

Arguments

obj	a cokm object construted via the function cokm in this package
input.new	a matrix including new inputs for making prediction

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

cokm, cokm.fit, cokm.condsim, ARCokrig

Examples

```
Funcc = function(x){
  return(0.5*(6*x-2)^2*sin(12*x-4)+10*(x-0.5)-5)
}

Funcf = function(x){
  z1 = Funcc(x)
  z2 = 2*z1-20*x+20 + sin(10*cos(5*x))
  return(z2)
}

#########################################################################
###### Nested design
#########################################################################
Dc <- seq(-1,1,0.1)
indDf <- c(1, 3, 6, 8, 10, 13, 17, 21)
zc <- Funcc(Dc)
Df <- Dc[indDf]
zf <- Funcf(Df)

input.new = as.matrix(seq(-1,1,length.out=200))


## create the cokm object
prior = list(name="Reference")
obj = cokm(formula=list(~1,~1+x1), output=list(c(zc), c(zf)),
              input=list(as.matrix(Dc), as.matrix(Df)),
              prior=prior, cov.model="matern_5_2")

## update model parameters in the cokm object

obj = cokm.fit(obj)

cokrige = cokm.predict(obj, input.new)
```

CRPS *Compute continous rank probability score for normal distributions*

Description

This function compute the continous rank probability score for normal distributions. It is mainly used to evaluate the validility of predictive distributions.

Usage

```
CRPS(x, mu, sig)
```

Arguments

x	a vector of true values (held-out data)
mu	a vector of predictive means
sig	a vector of predictive standard deviations

Author(s)

Pulong Ma <mpulong@gmail.com>

mvcokm *Construct the mvcokm object*

Description

This function constructs the mvcokm object in autogressive cokriging models for multivariate outputs. The model is known as the parallel partial (PP) cokriging emulator.

Usage

```
mvcokm(
  formula = list(~1, ~1),
  output,
  input,
  cov.model = "matern_5_2",
  nugget.est = FALSE,
  prior = list(),
  opt = list(),
  NestDesign = TRUE,
  tuning = list(),
  info = list()
)
```

Arguments

`formula`	a list of s elements, each of which contains the formula to specify fixed basis functions or regressors.
`output`	a list of s elements, each of which contains a matrix of computer model outputs.
`input`	a list of s elements, each of which contains a matrix of inputs.
`cov.model`	a string indicating the type of covariance function in the PP cokriging models. Current covariance functions include **exp** product form of exponential covariance functions. **matern_3_2** product form of Matern covariance functions with smoothness parameter 3/2. **matern_5_2** product form of Matern covariance functions with smoothness parameter 5/2. **Gaussian** product form of Gaussian covariance functions. **powexp** product form of power-exponential covariance functions with roughness parameter fixed at 1.9.
`nugget.est`	a logical value indicating whether the nugget is included or not. Default value is `FALSE`.
`prior`	a list of arguments to setup the prior distributions with the jointly robust prior as default **name** the name of the prior. Current implementation includes `JR`, `Reference`, `Jeffreys`, `Ind_Jeffreys` **hyperparam** hyperparameters in the priors. For jointly robust (JR) prior, three parameters are included: a refers to the polynomial penalty to avoid singular correlation matrix with a default value 0.2; b refers to the exponenetial penalty to avoid diagonal correlation matrix with a default value 1; nugget.UB is the upper bound of the nugget variance with default value 1, which indicates that the nugget variance has support (0, 1).
`opt`	a list of arguments to setup the `optim` routine.
`NestDesign`	a logical value indicating whether the experimental design is hierarchically nested within each level of the code.
`tuning`	a list of arguments to control the MCEM algorithm for non-nested design. It includes the arguments **maxit** the maximum number of MCEM iterations. **tol** a tolerance to stop the MCEM algorithm. If the parameter difference between any two consecutive MCEM algorithm is less than this tolerance, the MCEM algorithm is stopped. **n.sample** the number of Monte Carlo samples in the MCEM algorithm. **verbose** a logical value to show the MCEM iterations if it is true.
`info`	a list that contains **iter** number of iterations used in the MCEM algorithm **eps** parameter difference after the MCEM algorithm stops

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

ARCokrig, mvcokm.fit, mvcokm.predict, mvcokm.condsim

---

mvcokm-class *mvcokm Class*

Description

This is an S4 class definition for mvcokm in the ARCokrig package

Slots

output a list of s elements, each of which contains a matrix of computer model outputs.

input a list of s elements, each of which contains a matrix of inputs.

param a list of s elements, each of which contains a vector of initial values for correlation parameters (and nugget variance parameters if nugget terms are included in AR-cokriging models).

cov.model a string indicating the type of covariance function in AR-cokriging models. Current covariance functions include

exp product form of exponential covariance functions.

matern_3_2 product form of Matern covariance functions with smoothness parameter 3/2.

matern_5_2 product form of Matern covariance functions with smoothness parameter 5/2.

Gaussian product form of Gaussian covariance functions.

powexp product form of power-exponential covariance functions with roughness parameter fixed at 1.9.

aniso_exp anisotropic form of exponential covariance function.

aniso_matern_3_2 anisotropic form of Matern covariance functions with smoothness parameter 3/2.

aniso_matern_5_2 anisotropic form of Matern covariance functions with smoothness parameter 5/2.

nugget.est a logical value indicating whether the nugget is included or not. Default value is FALSE.

prior a list of arguments to setup the prior distributions with the jointly robust prior as default

name the name of the prior. Current implementation includes JR, Reference, Jeffreys, Ind_Jeffreys

hyperparam hyperparameters in the priors. For jointly robust (JR) prior, three parameters are included: a refers to the polynomial penalty to avoid singular correlation matrix with a default value 0.2; b refers to the exponenetial penalty to avoid diagonal correlation matrix with a default value 1; nugget.UB is the upper bound of the nugget variance with default value 1, which indicates that the nugget variance has support (0, 1).

`opt` a list of arguments to setup the `optim` routine.

`NestDesign` a logical value indicating whether the experimental design is hierarchically nested within each level of the code.

`tuning` a list of arguments to control the MCEM algorithm for non-nested design. It includes the arguments

maxit the maximum number of MCEM iterations.

tol a tolerance to stop the MCEM algorithm. If the parameter difference between any two consecutive MCEM algorithm is less than this tolerance, the MCEM algorithm is stopped.

n.sample the number of Monte Carlo samples in the MCEM algorithm.

`info` a list that contains

iter number of iterations used in the MCEM algorithm

eps parameter difference after the MCEM algorithm stops

Author(s)

Pulong Ma <mpulong@gmail.com>

`mvcokm.condsim` *Conditional simulation at new inputs in autoregressive cokriging models for multivarite output*

Description

This function makes prediction based on conditional simulation in autogressive cokriging models for multivariate output

Usage

```
mvcokm.condsim(obj, input.new, nsample = 30)
```

Arguments

`obj`	a `mvcokm` object construted via the function `mvcokm` in this package
`input.new`	a matrix including new inputs for making prediction
`nsample`	a numerical value indicating the number of samples

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

`mvcokm`, `mvcokm.fit`, `mvcokm.predict`, `ARCokrig`

mvcokm.fit — *fit the autoregressive cokriging model for multivariate output*

Description

This function estimates parameters in the parallel partial cokriging model

Usage

```
mvcokm.fit(obj)
```

Arguments

obj	a mvcokm object construted via the function mvcokm in this package

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

mvcokm, mvcokm.predict, mvcokm.condsim, ARCokrig

mvcokm.param — *Get model parameters in autoregressive cokriging models for multivarite output*

Description

This function computes estimates for regression and variance parameters given the correlation parameters are known. It is used to show all model parameters in one place.

Usage

```
mvcokm.param(obj)
```

Arguments

obj	a mvcokm object construted via the function mvcokm in this package

Value

a list of model parameters including regression coefficients β, scale discrepancy γ, variance parameters σ^2, and correlation parameters ϕ in covariance functions. If nugget parameters are included in the model, then nugget parameters are shown in ϕ.

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

mvcokm, mvcokm.fit, mvcokm.predict, ARCokrig

---

mvcokm.predict *Prediction at new inputs in autoregressive cokriging models for multivarite output*

Description

This function makes prediction in the parallel partial cokriging model. If a nested design is used, the predictive mean and predictive variance are computed exactly; otherwise, Monte Carlo simulation from the predictive distribution is used to approximate the predictive mean and predictive variance.

Usage

```
mvcokm.predict(obj, input.new)
```

Arguments

obj	a mvcokm object construted via the function mvcokm in this package
input.new	a matrix including new inputs for making prediction

Author(s)

Pulong Ma <mpulong@gmail.com>

See Also

mvcokm, mvcokm.fit, mvcokm.condsim, ARCokrig

Index

∗ **AR-Cokriging**
    cokm-class, 7
    mvcokm-class, 15
∗ **Computer-Experiments**
    cokm-class, 7
    mvcokm-class, 15
∗ **Objective-Bayes**
    cokm-class, 7
    mvcokm-class, 15
∗ **Uncertainty-Quantification**
    cokm-class, 7
    mvcokm-class, 15

ARCokrig, 2, *4*, *7*, *9–12*, *15–18*

cokm, *4*, 5, *7–12*
cokm-class, 7
cokm.condsim, 8, *11, 12*
cokm.fit, *4*, *7*, 9, *9*, *11, 12*
cokm.param, *4*, *10*, 11
cokm.predict, *4*, *7*, *9, 10*, 11
CRPS, 13

mvcokm, 13, *15–18*
mvcokm-class, 15
mvcokm.condsim, *15*, 16, *17, 18*
mvcokm.fit, *15, 16*, 17, *18*
mvcokm.param, 17
mvcokm.predict, *15–17*, 18, *18*

optim, *3*, *7, 8*, *14*, *16*