Package ‘rSPDE’

December 2, 2024

Type Package

Title Rational Approximations of Fractional Stochastic Partial Differential Equations

Version 2.4.0

Maintainer David Bolin <davidbolin@gmail.com>

Description Functions that compute rational approximations of fractional elliptic stochastic partial differential equations. The package also contains functions for common statistical usage of these approximations. The main references for rSPDE are Bolin, Simas and Xiong (2023) <doi:10.1080/10618600.2023.2231051> for the covariance-based method and Bolin and Kirchner (2020) <doi:10.1080/10618600.2019.1665537> for the operator-based rational approximation. These can be generated by the citation function in R.

Depends R (>= 3.5.0), Matrix

Imports stats, methods, fmesher (>= 0.2.0), lifecycle, broom

License GPL (>= 3) | file LICENSE

URL https://davidbolin.github.io/rSPDE/

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, INLA (>= 24.12.01), testthat, ggplot2, lattice, splancs, optimParallel, RSpectra, numDeriv, inlabru (>= 2.12.0), sn, viridis, doParallel, foreach, tidyr, dplyr, GeneralizedHyperbolic, gridExtra, MetricGraph, sf

Additional_repositories https://inla.r-inla-download.org/R/testing

BugReports https://github.com/davidbolin/rSPDE/issues

VignetteBuilder knitr

NeedsCompilation no

Author David Bolin [cre, aut],
Alexandre Simas [aut],
Finn Lindgren [ctb]

Repository CRAN

Date/Publication 2024-12-02 15:40:01 UTC

Contents

rSPDE-package *Rational approximations of fractional SPDEs.*

Description

rSPDE is used for approximating fractional elliptic SPDEs

$$L^\beta(\tau u(s)) = W,$$

where L is a differential operator and $\beta > 0$ is a general fractional power.

Details

The approximation is based on a rational approximation of the fractional operator, and allows for computationally efficient inference and simulation.

The main functions for computing rational approximation objects are:

fractional.operators() works for general rational operators

matern.operators() works for random fields with stationary Matern covariance functions

spde.matern.operators() works for random fields with defined as solutions to a possibly non-stationary Matern-type SPDE model.

rspde.matern() R-INLA implementation of the covariance-based rational approximation for random fields with stationary Matern covariance functions

Basic statistical operations such as likelihood evaluations (see [rSPDE.loglike], [rSPDE.matern.loglike]) and kriging predictions (see [predict.rSPDEobj], [predict.CBrSPDEobj]) using the rational approximations are also implemented.

For illustration purposes, the package contains a simple FEM implementation for models on R. For spatial models, the FEM implementation in the R-INLA package is recommended.

For a more detailed introduction to the package, see the rSPDE Vignettes.

Author(s)

Maintainer: David Bolin <davidbolin@gmail.com>

Authors:

- Alexandre Simas <alexandre.impa@gmail.com>

Other contributors:

- Finn Lindgren <finn.lindgren@ed.ac.uk> [contributor]

See Also

Useful links:

- https://davidbolin.github.io/rSPDE/
- Report bugs at https://github.com/davidbolin/rSPDE/issues

augment.rspde_lme *Augment data with information from a* rspde_lme *object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. It includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. It also contains the New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Usage

```
## S3 method for class 'rspde_lme'
augment(
  x,
  newdata = NULL,
  loc = NULL,
  mesh = FALSE,
  which_repl = NULL,
  se_fit = FALSE,
  conf_int = FALSE,
  pred_int = FALSE,
  level = 0.95,
  n_samples = 100,
  ...
)
```

Arguments

x	A rspde_lme object.
newdata	A data.frame or a list containing the covariates, the edge number and the distance on edge for the locations to obtain the prediction. If NULL, the fitted values will be given for the original locations where the model was fitted.
loc	Prediction locations. Can either be a data.frame, a matrix or a character vector, that contains the names of the columns of the coordinates of the locations. For models using metric_graph objects, plase use edge_number and distance_on_edge instead.
mesh	Obtain predictions for mesh nodes? The graph must have a mesh, and either only_latent is set to TRUE or the model does not have covariates.
which_repl	Which replicates to obtain the prediction. If NULL predictions will be obtained for all replicates. Default is NULL.
se_fit	Logical indicating whether or not a .se.fit column should be added to the augmented output. If TRUE, it only returns a non-NA value if type of prediction is 'link'.
conf_int	Logical indicating whether or not confidence intervals for the fitted variable should be built.

pred_int	Logical indicating whether or not prediction intervals for future observations should be built.
level	Level of confidence and prediction intervals if they are constructed.
n_samples	Number of samples when computing prediction intervals.
...	Additional arguments. Expert use only.

Value

A `tidyr::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.fittedlwrconf` Lower bound of the confidence interval, if conf_int = TRUE
- `.fitteduprconf` Upper bound of the confidence interval, if conf_int = TRUE
- `.fittedlwrpred` Lower bound of the prediction interval, if pred_int = TRUE
- `.fitteduprpred` Upper bound of the prediction interval, if pred_int = TRUE
- `.fixed` Prediction of the fixed effects.
- `.random` Prediction of the random effects.
- `.resid` The ordinary residuals, that is, the difference between observed and fitted values.
- `.se_fit` Standard errors of fitted values, if se_fit = TRUE.

See Also

glance.rspde_lme

`bru_get_mapper.inla_rspde` *rSPDE inlabru mapper*

Description

rSPDE inlabru mapper

Usage

```
bru_get_mapper.inla_rspde(model, ...)

ibm_n.bru_mapper_inla_rspde(mapper, ...)

ibm_values.bru_mapper_inla_rspde(mapper, ...)

ibm_jacobian.bru_mapper_inla_rspde(mapper, input, ...)
```

Arguments

model	An `inla_rspde` object for which to construct or extract a mapper
...	Arguments passed on to other methods
mapper	A `bru_mapper_inla_rspde` object
input	The values for which to produce a mapping matrix

Examples

```
 #tryCatch version
tryCatch({
if (requireNamespace("INLA", quietly = TRUE) &&
  requireNamespace("inlabru", quietly = TRUE)) {
  library(INLA)
  library(inlabru)

  set.seed(123)
  m <- 100
  loc_2d_mesh <- matrix(runif(m * 2), m, 2)
  mesh_2d <- inla.mesh.2d(
    loc = loc_2d_mesh,
    cutoff = 0.05,
    max.edge = c(0.1, 0.5)
  )
  sigma <- 1
  range <- 0.2
  nu <- 0.8
  kappa <- sqrt(8 * nu) / range
  op <- matern.operators(
    mesh = mesh_2d, nu = nu,
    range = range, sigma = sigma, m = 2,
    parameterization = "matern"
  )
  u <- simulate(op)
  A <- inla.spde.make.A(
    mesh = mesh_2d,
    loc = loc_2d_mesh
  )
  sigma.e <- 0.1
  y <- A %*% u + rnorm(m) * sigma.e
  y <- as.vector(y)

  data_df <- data.frame(
    y = y, x1 = loc_2d_mesh[, 1],
    x2 = loc_2d_mesh[, 2]
  )
  rspde_model <- rspde.matern(
    mesh = mesh_2d,
    nu_upper_bound = 2
  )

  cmp <- y ~ Intercept(1) +
```

```
    field(cbind(x1,x2), model = rspde_model)


  rspde_fit <- bru(cmp, data = data_df)
  summary(rspde_fit)
}
#stable.tryCatch
}, error = function(e){print("Could not run the example")})
```

`bru_get_mapper.inla_rspde_anisotropic2d` *rSPDE anisotropic inlabru mapper*

Description

rSPDE anisotropic inlabru mapper

Usage

```
bru_get_mapper.inla_rspde_anisotropic2d(model, ...)
```

Arguments

model	An `inla_rspde_anisotropic2d` object for which to construct or extract a mapper
...	Arguments passed on to other methods

`bru_get_mapper.inla_rspde_matern1d` *rSPDE stationary inlabru mapper*

Description

rSPDE stationary inlabru mapper

Usage

```
bru_get_mapper.inla_rspde_matern1d(model, ...)

ibm_n.bru_mapper_inla_rspde_matern1d(mapper, ...)

ibm_values.bru_mapper_inla_rspde_matern1d(mapper, ...)

ibm_jacobian.bru_mapper_inla_rspde_matern1d(mapper, input, ...)
```

Arguments

`model`	An `inla_rspde_matern1d` object for which to construct or extract a mapper
`...`	Arguments passed on to other methods
`mapper`	A `bru_mapper_inla_rspde_matern1d` object
`input`	The values for which to produce a mapping matrix

`bru_get_mapper.inla_rspde_spacetime`

rSPDE space time inlabru mapper

Description

rSPDE space time inlabru mapper

Usage

```
bru_get_mapper.inla_rspde_spacetime(model, ...)
```

Arguments

`model`	An `inla_rspde_spacetime` object for which to construct or extract a mapper
`...`	Arguments passed on to other methods

`construct.spde.matern.loglike`

Constructor of Matern loglikelihood functions for non-stationary models.

Description

This function evaluates the log-likelihood function for observations of a non-stationary Gaussian process defined as the solution to the SPDE

$$(\kappa(s) - \Delta)^{\beta}(\tau(s)u(s)) = W.$$

The observations are assumed to be generated as $Y_i = u(s_i) + \epsilon_i$, where ϵ_i are iid mean-zero Gaussian variables. The latent model is approximated using a rational approximation of the fractional SPDE model.

Usage

```
construct.spde.matern.loglike(
  object,
  Y,
  A,
  sigma.e = NULL,
  mu = 0,
  nu = NULL,
  m = NULL,
  log_scale = TRUE,
  return_negative_likelihood = TRUE
)
```

Arguments

object	The rational SPDE approximation, computed using matern.operators()
Y	The observations, either a vector or a matrix where the columns correspond to independent replicates of observations.
A	An observation matrix that links the measurement location to the finite element basis.
sigma.e	IF non-null, the standard deviation of the measurement noise will be kept fixed in the returned likelihood.
mu	Expectation vector of the latent field (default = 0).
nu	If non-null, the shape parameter will be kept fixed in the returned likelihood.
m	If non-null, update the order of the rational approximation, which needs to be a positive integer.
log_scale	Should the parameters be evaluated in log-scale?
return_negative_likelihood	Return minus the likelihood to turn the maximization into a minimization?

Value

The log-likelihood function. The parameters of the returned function are given in the order theta, nu, sigma.e, whenever they are available.

See Also

matern.operators(), predict.CBrSPDEobj()

Examples

```
# this example illustrates how the function can be used for maximum
# likelihood estimation
# Sample a Gaussian Matern process on R using a rational approximation
set.seed(123)
sigma.e <- 0.1
n.rep <- 10
```

```
n.obs <- 100
n.x <- 51
# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = n.x)
fem <- rSPDE.fem1d(x)
tau <- rep(0.5, n.x)
nu <- 0.8
alpha <- nu + 0.5
kappa <- rep(1, n.x)
# Matern parameterization
# compute rational approximation
op <- spde.matern.operators(
  loc_mesh = x,
  kappa = kappa, tau = tau, alpha = alpha,
  parameterization = "spde", d = 1
)
# Sample the model
u <- simulate(op, n.rep)
# Create some data
obs.loc <- runif(n = n.obs, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
noise <- rnorm(n.obs * n.rep)
dim(noise) <- c(n.obs, n.rep)
Y <- as.matrix(A %*% u + sigma.e * noise)
# define negative likelihood function for optimization using matern.loglike
mlik <- construct.spde.matern.loglike(op, Y, A)
#' #The parameters can now be estimated by minimizing mlik with optim

# Choose some reasonable starting values depending on the size of the domain
theta0 <- log(c(1 / sqrt(var(c(Y))), sqrt(8), 0.9, 0.01))
# run estimation and display the results
theta <- optim(theta0, mlik)
print(data.frame(
  tau = c(tau[1], exp(theta$par[1])), kappa = c(kappa[1], exp(theta$par[2])),
  nu = c(nu, exp(theta$par[3])), sigma.e = c(sigma.e, exp(theta$par[4])),
  row.names = c("Truth", "Estimates")
))


# SPDE parameterization
# compute rational approximation
op <- spde.matern.operators(
  kappa = kappa, tau = tau, alpha = alpha,
  loc_mesh = x, d = 1,
  parameterization = "spde"
)
# Sample the model
u <- simulate(op, n.rep)
# Create some data
obs.loc <- runif(n = n.obs, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
noise <- rnorm(n.obs * n.rep)
dim(noise) <- c(n.obs, n.rep)
```

```
Y <- as.matrix(A %*% u + sigma.e * noise)
# define negative likelihood function for optimization using matern.loglike
mlik <- construct.spde.matern.loglike(op, Y, A)
#' #The parameters can now be estimated by minimizing mlik with optim

# Choose some reasonable starting values depending on the size of the domain
theta0 <- log(c(1 / sqrt(var(c(Y))), sqrt(8), 0.9, 0.01))
# run estimation and display the results
theta <- optim(theta0, mlik)
print(data.frame(
  tau = c(tau[1], exp(theta$par[1])), kappa = c(kappa[1], exp(theta$par[2])),
  nu = c(nu, exp(theta$par[3])), sigma.e = c(sigma.e, exp(theta$par[4])),
  row.names = c("Truth", "Estimates")
))
```

`create_train_test_indices` *Create train and test splits to be used in the* `cross_validation` *function*

Description

Train and test splits

Usage

```
create_train_test_indices(
  data,
  cv_type = c("k-fold", "loo", "lpo"),
  k = 5,
  percentage = 20,
  number_folds = 10
)
```

Arguments

`data`	A `list`, `data.frame`, `SpatialPointsDataFrame` or `metric_graph_data` objects.
`cv_type`	The type of the folding to be carried out. The options are `k-fold` for k-fold cross-validation, in which case the parameter `k` should be provided, `loo`, for leave-one-out and `lpo` for leave-percentage-out, in this case, the parameter `percentage` should be given, and also the `number_folds` with the number of folds to be done. The default is `k-fold`.
`k`	The number of folds to be used in k-fold cross-validation. Will only be used if `cv_type` is `k-fold`.
`percentage`	The percentage (from 1 to 99) of the data to be used to train the model. Will only be used if `cv_type` is `lpo`.
`number_folds`	Number of folds to be done if `cv_type` is `lpo`.

Value

A list with two elements, `train` containing the training indices and `test` containing indices.

---

cross_validation *Perform cross-validation on a list of fitted models.*

Description

Obtain several scores for a list of fitted models according to a folding scheme.

Usage

```
cross_validation(
  models,
  model_names = NULL,
  scores = c("mse", "crps", "scrps", "dss"),
  cv_type = c("k-fold", "loo", "lpo"),
  k = 5,
  percentage = 20,
  number_folds = 10,
  n_samples = 1000,
  return_scores_folds = FALSE,
  orientation_results = c("negative", "positive"),
  include_best = TRUE,
  train_test_indexes = NULL,
  return_train_test = FALSE,
  return_post_samples = FALSE,
  return_true_test_values = FALSE,
  parallelize_RP = FALSE,
  n_cores_RP = parallel::detectCores() - 1,
  true_CV = TRUE,
  save_settings = FALSE,
  print = TRUE,
  fit_verbose = FALSE
)
```

Arguments

`models`	A fitted model obtained from calling the `bru()` function or a list of models fitted with the `bru()` function.
`model_names`	A vector containing the names of the models to appear in the returned `data.frame`. If `NULL`, the names will be of the form `Model 1`, `Model 2`, and so on. By default, it will try to obtain the name from the models list.
`scores`	A vector containing the scores to be computed. The options are "mse", "crps", "scrps" and "dss". By default, all scores are computed.

cv_type — The type of the folding to be carried out. The options are k-fold for k-fold cross-validation, in which case the parameter k should be provided, loo, for leave-one-out and lpo for leave-percentage-out, in this case, the parameter percentage should be given, and also the number_folds with the number of folds to be done. The default is k-fold.

k — The number of folds to be used in k-fold cross-validation. Will only be used if cv_type is k-fold.

percentage — The percentage (from 1 to 99) of the data to be used to train the model. Will only be used if cv_type is lpo.

number_folds — Number of folds to be done if cv_type is lpo.

n_samples — Number of samples to compute the posterior statistics to be used to compute the scores.

return_scores_folds — If TRUE, the scores for each fold will also be returned.

orientation_results — character vector. The options are "negative" and "positive". If "negative", the smaller the scores the better. If "positive", the larger the scores the better.

include_best — Should a row indicating which model was the best for each score be included?

train_test_indexes — A list containing two entries train, which is a list whose elements are vectors of indexes of the training data, and test, which is a list whose elements are vectors of indexes of the test data. Typically this will be returned list obtained by setting the argument return_train_test to TRUE.

return_train_test — Logical. Should the training and test indexes be returned? If 'TRUE' the train and test indexes will the 'train_test' element of the returned list.

return_post_samples — If TRUE the posterior samples will be included in the returned list.

return_true_test_values — If TRUE the true test values will be included in the returned list.

parallelize_RP — Logical. Should the computation of CRPS and SCRPS (and for some cases, DSS) be parallelized?

n_cores_RP — Number of cores to be used if parallelize_rp is TRUE.

true_CV — Should a TRUE cross-validation be performed? If TRUE the models will be fitted on the training dataset. If FALSE, the parameters will be kept fixed at the ones obtained in the result object.

save_settings — Logical. If TRUE, the settings used in the cross-validation will also be returned.

print — Should partial results be printed throughout the computation?

fit_verbose — Should INLA's run during cross-validation be verbose?

Value

A data.frame with the fitted models and the corresponding scores.

`folded.matern.covariance.1d`
The 1d folded Matern covariance function

Description

`folded.matern.covariance.1d` evaluates the 1d folded Matern covariance function over an interval $[0, L]$.

Usage

```
folded.matern.covariance.1d(
  h,
  m,
  kappa,
  nu,
  sigma,
  L = 1,
  N = 10,
  boundary = c("neumann", "dirichlet", "periodic")
)
```

Arguments

`h, m`	Vectors of arguments of the covariance function.
`kappa`	Range parameter.
`nu`	Shape parameter.
`sigma`	Standard deviation.
`L`	The upper bound of the interval $[0, L]$. By default, `L=1`.
`N`	The truncation parameter.
`boundary`	The boundary condition. The possible conditions are `"neumann"` (default), `"dirichlet"` or `"periodic"`.

Details

`folded.matern.covariance.1d` evaluates the 1d folded Matern covariance function over an interval $[0, L]$ under different boundary conditions. For periodic boundary conditions

$$C_{\mathcal{P}}(h, m) = \sum_{k=-\infty}^{\infty} (C(h - m + 2kL),$$

for Neumann boundary conditions

$$C_{\mathcal{N}}(h, m) = \sum_{k=-\infty}^{\infty} (C(h - m + 2kL) + C(h + m + 2kL)),$$

and for Dirichlet boundary conditions:

$$C_{\mathcal{D}}(h,m) = \sum_{k=-\infty}^{\infty} (C(h-m+2kL) - C(h+m+2kL)),$$

where $C(\cdot)$ is the Matern covariance function:

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h).$$

We consider the truncation:

$$C_{\mathcal{P},N}(h,m) = \sum_{k=-N}^{N} C(h-m+2kL), C_{\mathcal{N},N}(h,m) = \sum_{k=-\infty}^{\infty} (C(h-m+2kL)+C(h+m+2kL)),$$

and

$$C_{\mathcal{D},N}(h,m) = \sum_{k=-N}^{N} (C(h-m+2kL) - C(h+m+2kL)).$$

Value

A matrix with the corresponding covariance values.

Examples

```
x <- seq(from = 0, to = 1, length.out = 101)
plot(x, folded.matern.covariance.1d(rep(0.5, length(x)), x,
  kappa = 10, nu = 1 / 5, sigma = 1
),
type = "l", ylab = "C(h)", xlab = "h"
)
```

---

folded.matern.covariance.2d

The 2d folded Matern covariance function

Description

folded.matern.covariance.2d evaluates the 2d folded Matern covariance function over an interval $[0, L] \times [0, L]$.

Usage

```
folded.matern.covariance.2d(
  h,
  m,
  kappa,
  nu,
  sigma,
  L = 1,
  N = 10,
  boundary = c("neumann", "dirichlet", "periodic", "R2")
)
```

Arguments

`h`, `m`	Vectors with two coordinates.
`kappa`	Range parameter.
`nu`	Shape parameter.
`sigma`	Standard deviation.
`L`	The upper bound of the square $[0, L] \times [0, L]$. By default, `L=1`.
`N`	The truncation parameter.
`boundary`	The boundary condition. The possible conditions are `"neumann"` (default), `"dirichlet"`, `"periodic"` or `"R2"`.

Details

`folded.matern.covariance.2d` evaluates the 1d folded Matern covariance function over an interval $[0, L] \times [0, L]$ under different boundary conditions. For periodic boundary conditions

$$C_{\mathcal{P}}((h_1, h_2), (m_1, m_2)) = \sum_{k_2=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} (C(\|(h_1 - m_1 + 2k_1L, h_2 - m_2 + 2k_2L)\|),$$

for Neumann boundary conditions

$$C_{\mathcal{N}}((h_1, h_2), (m_1, m_2)) = \sum_{k_2=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} (C(\|(h_1-m_1+2k_1L, h_2-m_2+2k_2L)\|)+C(\|(h_1-m_1+2k_1L, h_2+m_2+2k_2L)\|$$

and for Dirichlet boundary conditions:

$$C_{\mathcal{D}}((h_1, h_2), (m_1, m_2)) = \sum_{k_2=-\infty}^{\infty} \sum_{k_1=-\infty}^{\infty} (C(\|(h_1-m_1+2k_1L, h_2-m_2+2k_2L)\|)-C(\|(h_1-m_1+2k_1L, h_2+m_2+2k_2L)\|$$

where $C(\cdot)$ is the Matern covariance function:

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa h)^{\nu} K_{\nu}(\kappa h).$$

We consider the truncation for k_1, k_2 from $-N$ to N.

Value

The correspoding covariance.

Examples

```
h <- c(0.5, 0.5)
m <- c(0.5, 0.5)
folded.matern.covariance.2d(h, m, kappa = 10, nu = 1 / 5, sigma = 1)
```

---

fractional.operators *Rational approximations of fractional operators*

Description

`fractional.operators` is used for computing an approximation, which can be used for inference and simulation, of the fractional SPDE

$$L^\beta(\tau u(s)) = W.$$

Here L is a differential operator, $\beta > 0$ is the fractional power, τ is a positive scalar or vector that scales the variance of the solution u, and W is white noise.

Usage

```
fractional.operators(L, beta, C, scale.factor, m = 1, tau = 1)
```

Arguments

`L`	A finite element discretization of the operator L.
`beta`	The positive fractional power.
`C`	The mass matrix of the finite element discretization.
`scale.factor`	A constant c is a lower bound for the the smallest eigenvalue of the non-discretized operator L.
`m`	The order of the rational approximation, which needs to be a positive integer. The default value is 1. Higer values gives a more accurate approximation, which are more computationally expensive to use for inference. Currently, the largest value of m that is implemented is 4.
`tau`	The constant or vector that scales the variance of the solution. The default value is 1.

Details

The approximation is based on a rational approximation of the fractional operator, resulting in an approximate model on the form

$$P_l u(s) = P_r W,$$

where $P_j = p_j(L)$ are non-fractional operators defined in terms of polynomials p_j for $j = l, r$. The order of p_r is given by `m` and the order of p_l is $m + m_\beta$ where m_β is the integer part of β if $\beta > 1$ and $m_\beta = 1$ otherwise.

The discrete approximation can be written as $u = P_r x$ where $x \sim N(0, Q^{-1})$ and $Q = P_l^T C^{-1} P_l$. Note that the matrices P_r and Q may be be ill-conditioned for $m > 1$. In this case, the methods in `operator.operations()` should be used for operations involving the matrices, since these methods are more numerically stable.

Value

`fractional.operators` returns an object of class "rSPDEobj". This object contains the following quantities:

`Pl`	The operator P_l.
`Pr`	The operator P_r.
`C`	The mass lumped mass matrix.
`Ci`	The inverse of `C`.
`m`	The order of the rational approximation.
`beta`	The fractional power.
`type`	String indicating the type of approximation.
`Q`	The matrix `t(Pl) %*% solve(C,Pl)`.
`type`	String indicating the type of approximation.
`Pl.factors`	List with elements that can be used to assemble P_l.
`Pr.factors`	List with elements that can be used to assemble P_r.

See Also

`matern.operators()`, `spde.matern.operators()`, `matern.operators()`

Examples

```
# Compute rational approximation of a Gaussian process with a
# Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
```

```
tau <- sqrt(gamma(nu) / (sigma^2 * kappa^(2 * nu) *
  (4 * pi)^(1 / 2) * gamma(nu + 1 / 2)))
op <- fractional.operators(
  L = fem$G + kappa^2 * fem$C, beta = (nu + 1 / 2) / 2,
  C = fem$C, scale.factor = kappa^2, tau = tau
)

v <- t(rSPDE.A1d(x, 0.5))
c.approx <- Sigma.mult(op, v)

# plot the result and compare with the true Matern covariance
plot(x, matern.covariance(abs(x - 0.5), kappa, nu, sigma),
  type = "l", ylab = "C(h)",
  xlab = "h", main = "Matern covariance and rational approximations"
)
lines(x, c.approx, col = 2)
```

`get.initial.values.rSPDE`

Initial values for log-likelihood optimization in rSPDE models with a latent stationary Gaussian Matern model

---

Description

Auxiliar function to obtain domain-based initial values for log-likelihood optimization in rSPDE models with a latent stationary Gaussian Matern model

Usage

```
get.initial.values.rSPDE(
  mesh = NULL,
  mesh.range = NULL,
  graph.obj = NULL,
  n.spde = 1,
  dim = NULL,
  B.tau = NULL,
  B.kappa = NULL,
  B.sigma = NULL,
  B.range = NULL,
  nu = NULL,
  parameterization = c("matern", "spde"),
  include.nu = TRUE,
  log.scale = TRUE,
  nu.upper.bound = NULL
)
```

Arguments

`mesh`	An in INLA mesh
`mesh.range`	The range of the mesh.
`graph.obj`	A `metric_graph` object. To be used in case both `mesh` and `mesh.range` are `NULL`.
`n.spde`	The number of basis functions in the mesh model.
`dim`	The dimension of the domain.
`B.tau`	Matrix with specification of log-linear model for τ. Will be used if `parameterization = 'spde'`.
`B.kappa`	Matrix with specification of log-linear model for κ. Will be used if `parameterization = 'spde'`.
`B.sigma`	Matrix with specification of log-linear model for σ. Will be used if `parameterization = 'matern'`.
`B.range`	Matrix with specification of log-linear model for ρ, which is a range-like parameter (it is exactly the range parameter in the stationary case). Will be used if `parameterization = 'matern'`.
`nu`	The smoothness parameter.
`parameterization`	Which parameterization to use? `matern` uses range, std. deviation and nu (smoothness). `spde` uses kappa, tau and nu (smoothness). The default is `matern`.
`include.nu`	Should we also provide an initial guess for nu?
`log.scale`	Should the results be provided in log scale?
`nu.upper.bound`	Should an upper bound for nu be considered?

Value

A vector of the form (theta_1,theta_2,theta_3) or where theta_1 is the initial guess for tau, theta_2 is the initial guess for kappa and theta_3 is the initial guess for nu.

---

gg_df — *Data frame for result objects from R-INLA fitted models to be used in ggplot2*

Description

Data frame for result objects from R-INLA fitted models to be used in ggplot2

Usage

```
gg_df(result, ...)
```

Arguments

`result`	a result object for which the data frame is desired
`...`	further arguments passed to or from other methods.

Value

A data frame containing the posterior densities.

`gg_df.rspde_result` *Data frame for rspde_result objects to be used in ggplot2*

Description

Returns a ggplot-friendly data-frame with the marginal posterior densities.

Usage

```
## S3 method for class 'rspde_result'
gg_df(
  result,
  parameter = result$params,
  transform = TRUE,
  restrict_x_axis = NULL,
  restrict_quantiles = NULL,
  ...
)
```

Arguments

`result`	An rspde_result object.
`parameter`	Vector. Which parameters to get the posterior density in the data.frame? The options are `std.dev`, `range`, `tau`, `kappa` and `nu`.
`transform`	Should the posterior density be given in the original scale?
`restrict_x_axis`	Variables to restrict the range of x axis based on quantiles.
`restrict_quantiles`	Named list of quantiles to restrict x axis. It should contain the name of the parameter along with a vector with two elements specifying the lower and upper quantiles. The names should be match the ones in result$params. For example, if we want to restrict nu to the 0.05 and 0.95 quantiles we do `restrict_quantiles = c(0.05, 0.95)`.
`...`	currently not used.

Value

A data frame containing the posterior densities.

glance.rspde_lme *Glance at an* `rspde_lme` *object*

Description

Glance accepts a `rspde_lme` object and returns a `tidyr::tibble()` with exactly one row of model summaries. The summaries are the square root of the estimated variance of the measurement error, residual degrees of freedom, AIC, BIC, log-likelihood, the type of latent model used in the fit and the total number of observations.

Usage

```
## S3 method for class 'rspde_lme'
glance(x, ...)
```

Arguments

`x`	An `rspde_lme` object.
`...`	Currently not used.

Value

A `tidyr::tibble()` with exactly one row and columns:

- `nobs` Number of observations used.
- `sigma` the square root of the estimated residual variance
- `logLik` The log-likelihood of the model.
- `AIC` Akaike's Information Criterion for the model.
- `BIC` Bayesian Information Criterion for the model.
- `deviance` Deviance of the model.
- `df.residual` Residual degrees of freedom.
- `model.type` Type of latent model fitted.

See Also

augment.rspde_lme

graph_data_rspde *Data extraction from metric graphs for 'rSPDE' models*

Description

Extracts data from metric graphs to be used by 'INLA' and 'inlabru'.

Usage

```
graph_data_rspde(
  graph_rspde,
  name = "field",
  repl = NULL,
  repl_col = NULL,
  group = NULL,
  group_col = NULL,
  only_pred = FALSE,
  time = NULL,
  bru = FALSE,
  tibble = FALSE,
  drop_na = FALSE,
  drop_all_na = TRUE
)
```

Arguments

graph_rspde	An inla_metric_graph_spde or inla_rspde_spacetime object built with the rspde.metric_graph() or rspde.spacetime() function.
name	A character string with the base name of the effect.
repl	Which replicates? If there is no replicates, one can set repl to NULL. If one wants all replicates, then one sets to repl to .all.
repl_col	Which "column" of the data contains the replicate variable?
group	Which groups? If there is no groups, one can set group to NULL. If one wants all groups, then one sets to group to .all.
group_col	Which "column" of the data contains the group variable?
only_pred	Should only return the data.frame to the prediction data?
time	Column containing times for space time models. Not needed when using inlabru. Only for INLA implementation of space time model.
bru	Should the data be processed for inlabru?
tibble	Should the data be returned as a tidyr::tibble?
drop_na	Should the rows with at least one NA for one of the columns be removed? DEFAULT is FALSE. This option is turned to FALSE if only_pred is TRUE.
drop_all_na	Should the rows with all variables being NA be removed? DEFAULT is TRUE. This option is turned to FALSE if only_pred is TRUE.

Value

An 'INLA' and 'inlabru' friendly list with the data.

`group_predict` *Perform prediction on a testing set based on a training set*

Description

Compute prediction of a formula-based expression on a testing set based on a training set.

Usage

```
group_predict(
  models,
  model_names = NULL,
  formula = NULL,
  train_indices,
  test_indices,
  n_samples = 1000,
  pseudo_predict = TRUE,
  return_samples = FALSE,
  return_hyper_samples = FALSE,
  n_hyper_samples = 1,
  compute_posterior_means = TRUE,
  print = TRUE,
  fit_verbose = FALSE
)
```

Arguments

`models`	A fitted model obtained from calling the `bru()` function or a list of models fitted with the `bru()` function.
`model_names`	A vector containing the names of the models to appear in the returned `data.frame`. If `NULL`, the names will be of the form `Model 1`, `Model 2`, and so on. By default, it will try to obtain the name from the models list.
`formula`	A formula where the right hand side defines an R expression to evaluate for each generated sample. If `NULL`` , the latent and hyperparameter states are returned as named lis` dict`method in the`inlabru' package.
`train_indices`	A list containing the indices of the observations for the model to be trained, or a numerical vector containing the indices.
`test_indices`	A list containing the indices of the test data, where the prediction will be done, or a numerical vector containing the indices.
`n_samples`	Number of samples to compute the posterior statistics to be used to compute the scores.

`pseudo_predict` If TRUE, the models will NOT be refitted on the training data, and the parameters obtained on the entire dataset will be used. If FALSE, the models will be refitted on the training data.

`return_samples` Should the posterior samples be returned?

`return_hyper_samples`
Should samples for the hyperparameters be obtained?

`n_hyper_samples`
Number of independent samples of the hyper parameters of size n_samples.

`compute_posterior_means`
Should the posterior means be computed from the posterior samples?

`print` Should partial results be printed throughout the computation?

`fit_verbose` Should INLA's run during the prediction be verbose?

Value

A data.frame with the fitted models and the corresponding scores.

---

intrinsic.matern.operators
Covariance-based approximations of intrinsic fields

Description

intrinsic.matern.operators is used for computing a covariance-based rational SPDE approximation of intrinsic fields on R^d defined through the SPDE

$$(-\Delta)^{\beta/2}(\kappa^2 - \Delta)^{\alpha/2}(\tau u) = \mathcal{W}$$

Usage

```
intrinsic.matern.operators(
  kappa,
  tau,
  alpha,
  beta = 1,
  G = NULL,
  C = NULL,
  d = NULL,
  mesh = NULL,
  graph = NULL,
  loc_mesh = NULL,
  m_alpha = 2,
  m_beta = 2,
  compute_higher_order = FALSE,
  return_block_list = FALSE,
  type_rational_approximation = c("chebfun", "brasil", "chebfunLB"),
```

```
  fem_mesh_matrices = NULL,
  scaling = NULL
)
```

Arguments

- `kappa` range parameter
- `tau` precision parameter
- `alpha` Smoothness parameter
- `beta` Smoothness parameter
- `G` The stiffness matrix of a finite element discretization of the domain of interest.
- `C` The mass matrix of a finite element discretization of the domain of interest.
- `d` The dimension of the domain.
- `mesh` An inla mesh.
- `graph` An optional `metric_graph` object. Replaces `d`, `C` and `G`.
- `loc_mesh` locations for the mesh for `d=1`.
- `m_alpha` The order of the rational approximation for the Matérn part, which needs to be a positive integer. The default value is 2.
- `m_beta` The order of the rational approximation for the intrinsic part, which needs to be a positive integer. The default value is 2.
- `compute_higher_order` Logical. Should the higher order finite element matrices be computed?
- `return_block_list` Logical. For `type = "covariance"`, should the block parts of the precision matrix be returned separately as a list?
- `type_rational_approximation` Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".
- `fem_mesh_matrices` A list containing FEM-related matrices. The list should contain elements c0, g1, g2, g3, etc.
- `scaling` second lowest eigenvalue of g1

Details

The covariance operator

$$\tau^{-2}(-\Delta)^{\beta}(\kappa^2 - \Delta)^{\alpha}$$

is approximated based on rational approximations of the two fractional components. The Laplacians are equipped with homogeneous Neumann boundary conditions and a zero-mean constraint is additionally imposed to obtained a non-intrinsic model.

Value

intrinsic.matern.operators returns an object of class "intrinsicCBrSPDEobj". This object is a list containing the following quantities:

- C — The mass lumped mass matrix.
- Ci — The inverse of C.
- GCi — The stiffness matrix G times Ci
- Gk — The stiffness matrix G along with the higher-order FEM-related matrices G2, G3, etc.
- fem_mesh_matrices — A list containing the mass lumped mass matrix, the stiffness matrix and the higher-order FEM-related matrices.
- m_alpha — The order of the rational approximation for the Matérn part.
- m_beta — The order of the rational approximation for the intrinsic part.
- alpha — The fractional power of the Matérn part of the operator.
- beta — The fractional power of the intrinsic part of the operator.
- type — String indicating the type of approximation.
- d — The dimension of the domain.
- A — Matrix that sums the components in the approximation to the mesh nodes.
- kappa — Range parameter of the covariance function
- tau — Scale parameter of the covariance function.
- type — String indicating the type of approximation.

Examples

```
if (requireNamespace("RSpectra", quietly = TRUE)) {
  x <- seq(from = 0, to = 10, length.out = 201)
  beta <- 1
  alpha <- 1
  kappa <- 1
  op <- intrinsic.matern.operators(
    kappa = kappa, tau = 1, alpha = alpha,
    beta = beta, loc_mesh = x, d = 1
  )
  # Compute and plot the variogram of the model
  Sigma <- op$A[,-1] %*% solve(op$Q[-1,-1], t(op$A[,-1]))
  One <- rep(1, times = ncol(Sigma))
  D <- diag(Sigma)
  Gamma <- 0.5 * (One %*% t(D) + D %*% t(One) - 2 * Sigma)
  k <- 100
  plot(x, Gamma[k, ], type = "l")
  lines(x,
    variogram.intrinsic.spde(x[k], x, kappa, alpha, beta, L = 10, d = 1),
    col = 2, lty = 2
  )
}
```

matern.covariance *The Matern covariance function*

Description

`matern.covariance` evaluates the Matern covariance function

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h).$$

Usage

```
matern.covariance(h, kappa, nu, sigma)
```

Arguments

`h`	Distances to evaluate the covariance function at.
`kappa`	Range parameter.
`nu`	Shape parameter.
`sigma`	Standard deviation.

Value

A vector with the values C(h).

Examples

```
x <- seq(from = 0, to = 1, length.out = 101)
plot(x, matern.covariance(abs(x - 0.5), kappa = 10, nu = 1 / 5, sigma = 1),
  type = "l", ylab = "C(h)", xlab = "h"
)
```

matern.operators *Rational approximations of stationary Gaussian Matern random fields*

Description

`matern.operators` is used for computing a rational SPDE approximation of a stationary Gaussian random fields on R^d with a Matern covariance function

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h)$$

Usage

```
matern.operators(
  kappa = NULL,
  tau = NULL,
  alpha = NULL,
  sigma = NULL,
  range = NULL,
  nu = NULL,
  G = NULL,
  C = NULL,
  d = NULL,
  mesh = NULL,
  graph = NULL,
  range_mesh = NULL,
  loc_mesh = NULL,
  m = 1,
  type = c("covariance", "operator"),
  parameterization = c("spde", "matern"),
  compute_higher_order = FALSE,
  return_block_list = FALSE,
  type_rational_approximation = c("chebfun", "brasil", "chebfunLB"),
  compute_logdet = FALSE
)
```

Arguments

`kappa`	Parameter kappa of the SPDE representation. If `NULL`, the range parameter will be used. If the range is also `NULL`, a starting value based on the mesh will be supplied.
`tau`	Parameter tau of the SPDE representation. If both sigma and tau are `NULL`, a starting value based on the mesh will be supplied.
`alpha`	Parameter alpha of the SPDE representation. If `alpha` is `NULL`, a starting value will be supplied.
`sigma`	Standard deviation of the covariance function. Used if `parameterization` is `matern`. If `NULL`, tau will be used. If tau is also `NULL`, a starting value based on the mesh will be supplied.
`range`	Range parameter of the covariance function. Used if `parameterization` is `matern`. If range is `NULL`, a starting value based on the mesh will be supplied.
`nu`	Shape parameter of the covariance function. Used if `parameterization` is `matern`. If `NULL`, a starting value will be supplied.
`G`	The stiffness matrix of a finite element discretization of the domain of interest. Does not need to be given if either `mesh` or `graph` is supplied.
`C`	The mass matrix of a finite element discretization of the domain of interest. Does not need to be given if either `mesh` or `graph` is supplied.
`d`	The dimension of the domain. Does not need to be given if either `mesh` or `graph` is provided.

`mesh` An optional fmesher mesh. Replaces `d`, `C` and `G`.

`graph` An optional `metric_graph` object. Replaces `d`, `C` and `G`.

`range_mesh` The range of the mesh. Will be used to provide starting values for the parameters. Will be used if `mesh` and `graph` are `NULL`, and if one of the parameters (kappa or tau for spde parameterization, or sigma or range for matern parameterization) are not provided.

`loc_mesh` The mesh locations used to construct the matrices C and G. This option should be provided if one wants to use the `rspde_lme()` function and will not provide neither graph nor mesh. Only works for 1d data. Does not work for metric graphs. For metric graphs you should supply the graph using the `graph` argument.

`m` The order of the rational approximation, which needs to be a positive integer. The default value is 1.

`type` The type of the rational approximation. The options are "covariance" and "operator". The default is "covariance".

`parameterization` Which parameterization to use? `matern` uses range, std. deviation and nu (smoothness). `spde` uses kappa, tau and alpha. The default is `spde`.

`compute_higher_order` Logical. Should the higher order finite element matrices be computed?

`return_block_list` Logical. For `type = "covariance"`, should the block parts of the precision matrix be returned separately as a list?

`type_rational_approximation` Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

`compute_logdet` Should log determinants be computed while building the model? (For covariance-based models)

Details

If `type` is "covariance", we use the covariance-based rational approximation of the fractional operator. In the SPDE approach, we model u as the solution of the following SPDE:

$$L^{\alpha/2}(\tau u) = \mathcal{W},$$

where $L = -\Delta + \kappa^2 I$ and $\mathcal{W}$ is the standard Gaussian white noise. The covariance operator of u is given by $L^{-\alpha}$. Now, let L_h be a finite-element approximation of L. We can use a rational approximation of order m on $L_h^{-\alpha}$ to obtain the following approximation:

$$L_{h,m}^{-\alpha} = L_h^{-m_\alpha} p(L_h^{-1}) q(L_h^{-1})^{-1},$$

where $m_\alpha = \lfloor \alpha \rfloor$, p and q are polynomials arising from such rational approximation. From this approximation we construct an approximate precision matrix for u.

If `type` is "operator", the approximation is based on a rational approximation of the fractional operator $(\kappa^2 - \Delta)^\beta$, where $\beta = (\nu + d/2)/2$. This results in an approximate model of the form

$$P_l u(s) = P_r W,$$

where $P_j = p_j(L)$ are non-fractional operators defined in terms of polynomials p_j for $j = l, r$. The order of p_r is given by m and the order of p_l is $m + m_\beta$ where m_β is the integer part of β if $\beta > 1$ and $m_\beta = 1$ otherwise.

The discrete approximation can be written as $u = P_r x$ where $x \sim N(0, Q^{-1})$ and $Q = P_l^T C^{-1} P_l$. Note that the matrices P_r and Q may be be ill-conditioned for $m > 1$. In this case, the methods in `operator.operations()` should be used for operations involving the matrices, since these methods are more numerically stable.

Value

If `type` is "covariance", then `matern.operators` returns an object of class "CBrSPDEobj". This object is a list containing the following quantities:

- `C` The mass lumped mass matrix.
- `Ci` The inverse of `C`.
- `GCi` The stiffness matrix G times `Ci`
- `Gk` The stiffness matrix G along with the higher-order FEM-related matrices G2, G3, etc.
- `fem_mesh_matrices` A list containing the mass lumped mass matrix, the stiffness matrix and the higher-order FEM-related matrices.
- `m` The order of the rational approximation.
- `alpha` The fractional power of the precision operator.
- `type` String indicating the type of approximation.
- `d` The dimension of the domain.
- `nu` Shape parameter of the covariance function.
- `kappa` Range parameter of the covariance function
- `tau` Scale parameter of the covariance function.
- `sigma` Standard deviation of the covariance function.
- `type` String indicating the type of approximation.

If `type` is "operator", then `matern.operators` returns an object of class "rSPDEobj". This object contains the quantities listed in the output of `fractional.operators()`, the `G` matrix, the dimension of the domain, as well as the parameters of the covariance function.

See Also

`fractional.operators()`, `spde.matern.operators()`, `matern.operators()`

Examples

```
# Compute the covariance-based rational approximation of a
# Gaussian process with a Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8
```

```
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
nobs <- 101
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
  parameterization = "matern"
)

v <- t(rSPDE.A1d(x, 0.5))
# Compute the precision matrix
Q <- op_cov$Q
# A matrix here is the identity matrix
A <- Diagonal(nobs)
# We need to concatenate 3 A's since we are doing a covariance-based rational
# approximation of order 2
Abar <- cbind(A, A, A)
w <- rbind(v, v, v)
# The approximate covariance function:
c_cov.approx <- (Abar) %*% solve(Q, w)
c.true <- folded.matern.covariance.1d(
  rep(0.5, length(x)),
  abs(x), kappa, nu, sigma
)

# plot the result and compare with the true Matern covariance
plot(x, c.true,
  type = "l", ylab = "C(h)",
  xlab = "h", main = "Matern covariance and rational approximations"
)
lines(x, c_cov.approx, col = 2)


# Compute the operator-based rational approximation of a Gaussian
# process with a Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
op <- matern.operators(
  range = range, sigma = sigma, nu = nu,
  loc_mesh = x, d = 1,
```

```
  type = "operator",
  parameterization = "matern"
)

v <- t(rSPDE.A1d(x, 0.5))
c.approx <- Sigma.mult(op, v)
c.true <- folded.matern.covariance.1d(
  rep(0.5, length(x)),
  abs(x), kappa, nu, sigma
)

# plot the result and compare with the true Matern covariance
plot(x, c.true,
  type = "l", ylab = "C(h)",
  xlab = "h", main = "Matern covariance and rational approximation"
)
lines(x, c.approx, col = 2)
```

`matern.rational` *Rational approximation of the Matern fields on intervals and metric graphs*

Description

The function is used for computing an approximation, which can be used for inference and simulation, of the fractional SPDE

$$(\kappa^2 - \Delta)^{\alpha/2}(\tau u(s)) = W$$

on intervals or metric graphs. Here W is Gaussian white noise, κ controls the range, $\alpha = \nu + 1/2$ with $\nu > 0$ controls the smoothness and τ is related to the marginal variances through

$$\sigma^2 = \frac{\Gamma(\nu)}{\tau^2\Gamma(\alpha)2\sqrt{\pi}\kappa^{2\nu}}.$$

Usage

```
matern.rational(
  graph = NULL,
  loc = NULL,
  bc = c("free", "Neumann", "Dirichlet"),
  kappa = NULL,
  range = NULL,
  nu = NULL,
  sigma = NULL,
  tau = NULL,
  alpha = NULL,
  m = 2,
  parameterization = c("matern", "spde"),
  type_rational_approximation = "brasil",
  type_interp = "spline"
)
```

Arguments

- `graph` Metric graph object. The default is NULL, which means that a stationary Matern model on the line is created.
- `loc` Locations where to evaluate the model.
- `bc` Specifies the boundary conditions. The default is "free" which gives stationary Matern models on intervals. Other options are "Neumann" or "Dirichlet".
- `kappa` Range parameter
- `range` practical correlation range
- `nu` Smoothness parameter
- `sigma` Standard deviation
- `tau` Precision parameter
- `alpha` Smoothness parameter
- `m` The order of the approximation
- `parameterization` Which parameterization to use? `matern` uses range, std. deviation and nu (smoothness). `spde` uses kappa, tau and alpha. The default is `matern`.
- `type_rational_approximation` Method used to compute the coefficients of the rational approximation.
- `type_interp` Interpolation method for the rational coefficients.

Value

A model object for the the approximation

Examples

```
s <- seq(from = 0, to = 1, length.out = 101)
kappa <- 20
sigma <- 2
nu <- 0.8
r <- sqrt(8*nu)/kappa #range parameter
op_cov <- matern.rational(loc = s, nu = nu, range = r, sigma = sigma, m = 2,
parameterization = "matern")
cov.true <- matern.covariance(abs(s-s[1]), kappa = kappa, sigma = sigma, nu = nu)
cov.approx <- op_cov$covariance(ind = 1)

plot(s, cov.true)
lines(s, cov.approx, col = 2)
```

matern.rational.cov *Rational approximation of the Matern covariance*

Description

Computes a rational approximation of the Matern covariance function on intervals.

Usage

```
matern.rational.cov(
  h,
  order,
  kappa,
  nu,
  sigma,
  type_rational = "brasil",
  type_interp = "linear"
)
```

Arguments

h	Distances to compute the covariance for
order	The order of the approximation
kappa	Range parameter
nu	Smoothness parameter
sigma	Standard deviation
type_rational	Method used to compute the coefficients of the rational approximation.
type_interp	Interpolation method for the rational coefficients.

Value

The covariance matrix of the approximation

Examples

```
h <- seq(from = 0, to = 1, length.out = 100)
cov.true <- matern.covariance(h, kappa = 10, sigma = 1, nu = 0.8)
cov.approx <- matern.rational.cov(h, kappa = 10, sigma = 1, nu = 0.8, order = 2)

plot(h, cov.true)
lines(h, cov.approx, col = 2)
```

matern2d.operators *Rational approximations of stationary anisotropic Gaussian Matern random fields*

Description

`matern2d.operators` is used for computing a rational SPDE approximation of a stationary Gaussian random fields on R^d with a Matern covariance function

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\sqrt{h^T H^{-1} h})^\nu K_\nu(\sqrt{h^T H^{-1} h})$$

, based on a SPDE representation of the form

$$(I - \nabla \cdot (H\nabla))^{(\nu+1)/2} u = c\sigma W$$

, where $c>0$ is a constant. The matrix H is defined as

$$\begin{bmatrix} h_x^2 & h_x h_y h_{xy} \\ h_x h_y h_{xy} & h_y^2 \end{bmatrix}$$

Usage

```
matern2d.operators(
  hx = NULL,
  hy = NULL,
  hxy = NULL,
  nu = NULL,
  sigma = NULL,
  mesh = NULL,
  fem = NULL,
  m = 1,
  type_rational_approximation = c("chebfun", "brasil", "chebfunLB"),
  return_fem_matrices = FALSE
)
```

Arguments

`hx`	Parameter in the H matrix.
`hy`	Parameter in the H matrix.
`hxy`	Parameter in the H matrix.
`nu`	Smoothness parameter.
`sigma`	standard deviation parameter.
`mesh`	An `fmesher` mesh.
`fem`	Optional precomputed FEM matrices.
`m`	The order of the rational approximation, which needs to be a positive integer. The default value is 1.

`type_rational_approximation`
: Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

`return_fem_matrices`
: Should the FEM matrices be returned?

Value

An object of type `CBrSPDEobj2d`

See Also

`fractional.operators()`, `spde.matern.operators()`, `matern.operators()`

Examples

```
library(fmesher)
n_loc <- 2000
loc_2d_mesh <- matrix(runif(n_loc * 2), n_loc, 2)
mesh_2d <- fm_mesh_2d(loc = loc_2d_mesh, cutoff = 0.03, max.edge = c(0.1, 0.5))
op <- matern2d.operators(mesh = mesh_2d)
```

`operator.operations` *Operations with the Pr and Pl operators*

Description

Functions for multiplying and solving with the P_r and P_l operators as well as the latent precision matrix $Q = P_l C^{-1} P_l$ and covariance matrix $\Sigma = P_r Q^{-1} P_r^T$. These operations are done without first assembling P_r, P_l in order to avoid numerical problems caused by ill-conditioned matrices.

Usage

```
Pr.mult(obj, v, transpose = FALSE)

Pr.solve(obj, v, transpose = FALSE)

Pl.mult(obj, v, transpose = FALSE)

Pl.solve(obj, v, transpose = FALSE)

Q.mult(obj, v)

Q.solve(obj, v)

Qsqrt.mult(obj, v, transpose = FALSE)

Qsqrt.solve(obj, v, transpose = FALSE)
```

```
Sigma.mult(obj, v)

Sigma.solve(obj, v)
```

Arguments

`obj`	rSPDE object
`v`	vector to apply the operation to
`transpose`	set to TRUE if the operation should be performed with the transposed object

Details

`Pl.mult`, `Pr.mult`, and `Q.mult` multiplies the vector with the respective object. Changing `mult` to `solve` in the function names multiplies the vector with the inverse of the object. `Qsqrt.mult` and `Qsqrt.solve` performs the operations with the square-root type object $Q_r = C^{-1/2}P_l$ defined so that $Q = Q_r^T Q_r$.

Value

A vector with the values of the operation

`precision` *Get the precision matrix of CBrSPDEobj objects*

Description

Function to get the precision matrix of a CBrSPDEobj object

Usage

```
precision(object, ...)

## S3 method for class 'CBrSPDEobj'
precision(
  object,
  nu = NULL,
  kappa = NULL,
  sigma = NULL,
  range = NULL,
  tau = NULL,
  m = NULL,
  ...
)
```

Arguments

object	The covariance-based rational SPDE approximation, computed using `matern.operators()`
...	Currently not used.
nu	If non-null, update the shape parameter of the covariance function.
kappa	If non-null, update the range parameter of the covariance function.
sigma	If non-null, update the standard deviation of the covariance function.
range	If non-null, update the range parameter of the covariance function.
tau	If non-null, update the parameter tau.
m	If non-null, update the order of the rational approximation, which needs to be a positive integer.

Value

The precision matrix.

See Also

`simulate.CBrSPDEobj()`, `matern.operators()`

Examples

```
# Compute the covariance-based rational approximation of a
# Gaussian process with a Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8
range <- 0.2

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
tau <- sqrt(gamma(nu) / (sigma^2 * kappa^(2 * nu) *
  (4 * pi)^(1 / 2) * gamma(nu + 1 / 2)))
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
  parameterization = "matern"
)

# Get the precision matrix:
prec_matrix <- precision(op_cov)
```

`precision.CBrSPDEobj2d` *Get the precision matrix of CBrSPDEobj2d objects*

Description

Function to get the precision matrix of a CBrSPDEobj2d object

Usage

```
## S3 method for class 'CBrSPDEobj2d'
precision(
  object,
  nu = NULL,
  hx = NULL,
  hy = NULL,
  hxy = NULL,
  sigma = NULL,
  m = NULL,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `matern2d.operators()`
`nu`	If non-null, update the shape parameter of the covariance function.
`hx`	If non-null, update the hx parameter.
`hy`	If non-null, update the hy parameter.
`hxy`	If non-null, update the hxy parameter.
`sigma`	If non-null, update the standard deviation of the covariance function.
`m`	If non-null, update the order of the rational approximation, which needs to be a positive integer.
`...`	Currently not used.

Value

The precision matrix.

See Also

`simulate.CBrSPDEobj2d()`, `matern2d.operators()`

Examples

```
library(fmesher)
n_loc <- 2000
loc_2d_mesh <- matrix(runif(n_loc * 2), n_loc, 2)
mesh_2d <- fm_mesh_2d(loc = loc_2d_mesh, cutoff = 0.03, max.edge = c(0.1, 0.5))
op <- matern2d.operators(mesh = mesh_2d)
Q <- precision(op)
```

---

precision.inla_rspde *Get the precision matrix of* inla_rspde *objects*

Description

Function to get the precision matrix of an inla_rspde object created with the rspde.matern() function.

Usage

```
## S3 method for class 'inla_rspde'
precision(object, theta = NULL, ...)
```

Arguments

object	The inla_rspde object obtained with the rspde.matern() function.
theta	If null, the starting values for theta will be used. Otherwise, it must be suplied as a vector. For stationary models, we have theta = c(log(tau), log(kappa), nu). For nonstationary models, we have theta = c(theta_1, theta_2, ..., theta_n, nu).
...	Currently not used.

Value

The precision matrix.

See Also

precision.CBrSPDEobj(), matern.operators()

precision.rSPDEobj1d   *Get the precision matrix of rSPDEobj1d objects*

Description

Function to get the precision matrix of a rSPDEobj1d object

Usage

```
## S3 method for class 'rSPDEobj1d'
precision(
  object,
  loc = NULL,
  nu = NULL,
  kappa = NULL,
  sigma = NULL,
  range = NULL,
  tau = NULL,
  m = NULL,
  ordering = c("field", "location"),
  ldl = FALSE,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `matern.rational()`
`loc`	If non-null, update the locations where to evaluate the model.
`nu`	If non-null, update the shape parameter of the covariance function.
`kappa`	If non-null, update the range parameter of the covariance function.
`sigma`	If non-null, update the standard deviation of the covariance function.
`range`	If non-null, update the range parameter of the covariance function.
`tau`	If non-null, update the parameter tau.
`m`	If non-null, update the order of the rational approximation, which needs to be a positive integer.
`ordering`	Return the matrices ordered by field or by location?
`ldl`	Directly build the LDL factorization of the precision matrix?
`...`	Currently not used.

Value

A list containing the precision matrix `Q` of the process and its derivatives if they exist, and a matrix `A` that extracts the elements corresponding to the process. If `ldl=TRUE`, the LDL factorization is returned instead of `Q`. If the locations are not ordered, the precision matrix is given for the ordered locations, but the `A` matrix returns to the original order.

See Also

simulate.rSPDEobj1d(), matern.rational()

Examples

```
# Compute the covariance-based rational approximation of a
# Gaussian process with a Matern covariance function on R
sigma <- 1
nu <- 0.8
range <- 0.2

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)

op_cov <- matern.rational(
  loc = x, nu = nu,
  range = range, sigma = sigma, m = 2,
  parameterization = "matern"
)

# Get the precision matrix:
prec_matrix <- precision(op_cov)
```

---

precision.spacetimeobj — *Get the precision matrix of spacetimeobj objects*

Description

Function to get the precision matrix of a spacetimeobj object

Usage

```
## S3 method for class 'spacetimeobj'
precision(object, kappa = NULL, sigma = NULL, gamma = NULL, rho = NULL, ...)
```

Arguments

object	The model object computed using spacetime.operators()
kappa	If non-null, update the range parameter of the covariance function.
sigma	If non-null, update the standard deviation of the covariance function.
gamma	If non-null, update the temporal range parameter of the covariance function.
rho	If non-null, update the drift parameter of the covariance function.
...	Currently not used.

Value

The precision matrix.

See Also

simulate.spacetimeobj(), spacetime.operators()

Examples

```
s <- seq(from = 0, to = 20, length.out = 101)
t <- seq(from = 0, to = 20, length.out = 31)

op_cov <- spacetime.operators(space_loc = s, time_loc = t,
                              kappa = 5, sigma = 10, alpha = 1,
                              beta = 2, rho = 1, gamma = 0.05)
prec_matrix <- precision(op_cov)
```

---

predict.CBrSPDEobj — *Prediction of a fractional SPDE using the covariance-based rational SPDE approximation*

Description

The function is used for computing kriging predictions based on data $Y_i = u(s_i) + \epsilon_i$, where ϵ is mean-zero Gaussian measurement noise and $u(s)$ is defined by a fractional SPDE $(\kappa^2 I - \Delta)^{\alpha/2}(\tau u(s)) = W$, where W is Gaussian white noise and $\alpha = \nu + d/2$, where d is the dimension of the domain.

Usage

```
## S3 method for class 'CBrSPDEobj'
predict(
  object,
  A,
  Aprd,
  Y,
  sigma.e,
  mu = 0,
  compute.variances = FALSE,
  posterior_samples = FALSE,
  n_samples = 100,
  only_latent = FALSE,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `matern.operators()`
`A`	A matrix linking the measurement locations to the basis of the FEM approximation of the latent model.
`Aprd`	A matrix linking the prediction locations to the basis of the FEM approximation of the latent model.
`Y`	A vector with the observed data, can also be a matrix where the columns are observations of independent replicates of u.
`sigma.e`	The standard deviation of the Gaussian measurement noise. Put to zero if the model does not have measurement noise.
`mu`	Expectation vector of the latent field (default = 0).
`compute.variances`	Set to also TRUE to compute the kriging variances.
`posterior_samples`	If `TRUE`, posterior samples will be returned.
`n_samples`	Number of samples to be returned. Will only be used if `sampling` is `TRUE`.
`only_latent`	Should the posterior samples be only given to the laten model?
`...`	further arguments passed to or from other methods.

Value

A list with elements

`mean`	The kriging predictor (the posterior mean of u|Y).
`variance`	The posterior variances (if computed).

Examples

```
set.seed(123)
# Sample a Gaussian Matern process on R using a rational approximation
kappa <- 10
sigma <- 1
nu <- 0.8
sigma.e <- 0.3
range <- 0.2

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

tau <- sqrt(gamma(nu) / (sigma^2 * kappa^(2 * nu) *
  (4 * pi)^(1 / 2) * gamma(nu + 1 / 2)))

# Compute the covariance-based rational approximation
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
```

```
  parameterization = "matern"
)

# Sample the model
u <- simulate(op_cov)

# Create some data
obs.loc <- runif(n = 10, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
Y <- as.vector(A %*% u + sigma.e * rnorm(10))

# compute kriging predictions at the FEM grid
A.krig <- rSPDE.A1d(x, x)
u.krig <- predict(op_cov,
  A = A, Aprd = A.krig, Y = Y, sigma.e = sigma.e,
  compute.variances = TRUE
)

plot(obs.loc, Y,
  ylab = "u(x)", xlab = "x", main = "Data and prediction",
  ylim = c(
    min(u.krig$mean - 2 * sqrt(u.krig$variance)),
    max(u.krig$mean + 2 * sqrt(u.krig$variance))
  )
)
lines(x, u.krig$mean)
lines(x, u.krig$mean + 2 * sqrt(u.krig$variance), col = 2)
lines(x, u.krig$mean - 2 * sqrt(u.krig$variance), col = 2)
```

---

predict.CBrSPDEobj2d *Prediction of an anisotropic Whittle-Matern field*

Description

The function is used for computing kriging predictions based on data $Y_i = u(s_i) + \epsilon_i$, where ϵ is mean-zero Gaussian measurement noise and $u(s)$ is defined by a SPDE as described in matern2d.operators().

Usage

```
## S3 method for class 'CBrSPDEobj2d'
predict(
  object,
  A,
  Aprd,
  Y,
  sigma.e,
  mu = 0,
  compute.variances = FALSE,
  posterior_samples = FALSE,
```

```
  n_samples = 100,
  only_latent = FALSE,
  ...
)
```

Arguments

object — The covariance-based rational SPDE approximation, computed using `matern2d.operators()`

A — A matrix linking the measurement locations to the basis of the FEM approximation of the latent model.

Aprd — A matrix linking the prediction locations to the basis of the FEM approximation of the latent model.

Y — A vector with the observed data, can also be a matrix where the columns are observations of independent replicates of u.

sigma.e — The standard deviation of the Gaussian measurement noise. Put to zero if the model does not have measurement noise.

mu — Expectation vector of the latent field (default = 0).

compute.variances — Set to also TRUE to compute the kriging variances.

posterior_samples — If `TRUE`, posterior samples will be returned.

n_samples — Number of samples to be returned. Will only be used if `sampling` is `TRUE`.

only_latent — Should the posterior samples be only given to the laten model?

... — further arguments passed to or from other methods.

Value

A list with elements

mean — The kriging predictor (the posterior mean of u|Y).

variance — The posterior variances (if computed).

Examples

```
library(fmesher)
n_loc <- 2000
loc_2d_mesh <- matrix(runif(n_loc * 2), n_loc, 2)
mesh_2d <- fm_mesh_2d(loc = loc_2d_mesh, cutoff = 0.01, max.edge = c(0.1, 0.5))
op <- matern2d.operators(hx = 0.08, hy = 0.08, hxy = 0.5, nu = 0.5,
sigma = 1, mesh = mesh_2d)
u <- simulate(op)
n.obs <- 2000
obs.loc <- cbind(runif(n.obs),runif(n.obs))
A <- fm_basis(mesh_2d,obs.loc)
sigma.e <- 0.1
Y <- as.vector(A%*%u + sigma.e*rnorm(n.obs))
A <- op$make_A(obs.loc)
```

```
proj <- fm_evaluator(mesh_2d, dims = c(100, 100),
            xlim = c(0,1), ylim = c(0,1))
Aprd <- op$make_A(proj$lattice$loc)
u.krig <- predict(op, A = A, Aprd = Aprd, Y = Y, sigma.e = sigma.e)
```

---

predict.inla_rspde_matern1d
Predict method for 'inlabru' stationary Matern 1d models

Description

Auxiliar function to obtain predictions of the stationary Matern 1d models using 'inlabru'.

Usage

```
## S3 method for class 'inla_rspde_matern1d'
predict(
  object,
  cmp,
  bru_fit,
  newdata = NULL,
  formula = NULL,
  n.samples = 100,
  seed = 0L,
  probs = c(0.025, 0.5, 0.975),
  return_original_order = TRUE,
  num.threads = NULL,
  include = NULL,
  exclude = NULL,
  drop = FALSE,
  tolerance = 1e-04,
  ...
)
```

Arguments

`object`	An `inla_rspde_matern1d` object built with the `rspde.matern1d()` function.
`cmp`	The 'inlabru' component used to fit the model.
`bru_fit`	A fitted model using 'inlabru' or 'INLA'.
`newdata`	A data.frame of covariates needed for the prediction.
`formula`	A formula where the right hand side defines an R expression to evaluate for each generated sample. If NULL, the latent and hyperparameter states are returned as named list elements. See Details for more information.
`n.samples`	Integer setting the number of samples to draw in order to calculate the posterior statistics. The default is rather low but provides a quick approximate result.
`seed`	Random number generator seed passed on to `inla.posterior.sample()`

probs	A numeric vector of probabilities with values in the standard unit interval to be passed to stats::quantile
return_original_order	Should the predictions be returned in the original order?
num.threads	Specification of desired number of threads for parallel computations. Default NULL, leaves it up to 'INLA'. When seed != 0, overridden to "1:1"
include	Character vector of component labels that are needed by the predictor expression; Default: NULL (include all components that are not explicitly excluded)
exclude	Character vector of component labels that are not used by the predictor expression. The exclusion list is applied to the list as determined by the include parameter; Default: NULL (do not remove any components from the inclusion list)
drop	logical; If keep=FALSE, data is a SpatialDataFrame, and the prediciton summary has the same number of rows as data, then the output is a SpatialDataFrame object. Default FALSE.
tolerance	Tolerance for merging locations.
...	Additional arguments passed on to `inla.posterior.sample()`.

Value

A list with predictions.

`predict.rSPDEobj` *Prediction of a fractional SPDE using a rational SPDE approximation*

Description

The function is used for computing kriging predictions based on data $Y_i = u(s_i) + \epsilon_i$, where ϵ is mean-zero Gaussian measurement noise and $u(s)$ is defined by a fractional SPDE $L^\beta u(s) = W$, where W is Gaussian white noise.

Usage

```
## S3 method for class 'rSPDEobj'
predict(
  object,
  A,
  Aprd,
  Y,
  sigma.e,
  compute.variances = FALSE,
  posterior_samples = FALSE,
  n_samples = 100,
  only_latent = FALSE,
  ...
)
```

Arguments

object
: The rational SPDE approximation, computed using `fractional.operators()`, `matern.operators()`, or `spde.matern.operators()`.

A
: A matrix linking the measurement locations to the basis of the FEM approximation of the latent model.

Aprd
: A matrix linking the prediction locations to the basis of the FEM approximation of the latent model.

Y
: A vector with the observed data, can also be a matrix where the columns are observations of independent replicates of u.

sigma.e
: The standard deviation of the Gaussian measurement noise. Put to zero if the model does not have measurement noise.

compute.variances
: Set to also TRUE to compute the kriging variances.

posterior_samples
: If `TRUE`, posterior samples will be returned.

n_samples
: Number of samples to be returned. Will only be used if `sampling` is `TRUE`.

only_latent
: Should the posterior samples be only given to the latent model?

...
: further arguments passed to or from other methods.

Value

A list with elements

mean
: The kriging predictor (the posterior mean of u|Y).

variance
: The posterior variances (if computed).

samples
: A matrix containing the samples if `sampling` is `TRUE`.

Examples

```
# Sample a Gaussian Matern process on R using a rational approximation
kappa <- 10
sigma <- 1
nu <- 0.8
sigma.e <- 0.3
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation
op <- matern.operators(
  range = range, sigma = sigma,
  nu = nu, loc_mesh = x, d = 1,
  parameterization = "matern"
)
```

```
# Sample the model
u <- simulate(op)

# Create some data
obs.loc <- runif(n = 10, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
Y <- as.vector(A %*% u + sigma.e * rnorm(10))

# compute kriging predictions at the FEM grid
A.krig <- rSPDE.A1d(x, x)
u.krig <- predict(op,
  A = A, Aprd = A.krig, Y = Y, sigma.e = sigma.e,
  compute.variances = TRUE
)

plot(obs.loc, Y,
  ylab = "u(x)", xlab = "x", main = "Data and prediction",
  ylim = c(
    min(u.krig$mean - 2 * sqrt(u.krig$variance)),
    max(u.krig$mean + 2 * sqrt(u.krig$variance))
  )
)
lines(x, u.krig$mean)
lines(x, u.krig$mean + 2 * sqrt(u.krig$variance), col = 2)
lines(x, u.krig$mean - 2 * sqrt(u.krig$variance), col = 2)
```

---

predict.rspde_lme   *Prediction of a mixed effects regression model on a metric graph.*

Description

Prediction of a mixed effects regression model on a metric graph.

Usage

```
## S3 method for class 'rspde_lme'
predict(
  object,
  newdata = NULL,
  loc = NULL,
  time = NULL,
  mesh = FALSE,
  which_repl = NULL,
  compute_variances = FALSE,
  posterior_samples = FALSE,
  n_samples = 100,
  sample_latent = FALSE,
  return_as_list = FALSE,
  return_original_order = TRUE,
```

```
  ...,
  data = deprecated()
)
```

Arguments

object
: The fitted object with the `rspde_lme()` function

newdata
: A `data.frame` or a `list` containing the covariates, the edge number and the distance on edge for the locations to obtain the prediction.

loc
: Prediction locations. Can either be a `data.frame`, a `matrix` or a character vector, that contains the names of the columns of the coordinates of the locations. For models using `metric_graph` objects, plase use `edge_number` and `distance_on_edge` instead.

time
: Prediction times for spatio-temporal models.

mesh
: Obtain predictions for mesh nodes? The graph must have a mesh, and either `only_latent` is set to TRUE or the model does not have covariates.

which_repl
: Which replicates to use? If `NULL` all replicates will be used.

compute_variances
: Set to also TRUE to compute the kriging variances.

posterior_samples
: If `TRUE`, posterior samples will be returned.

n_samples
: Number of samples to be returned. Will only be used if `sampling` is `TRUE`.

sample_latent
: Do posterior samples only for the random effects?

return_as_list
: Should the means of the predictions and the posterior samples be returned as a list, with each replicate being an element?

return_original_order
: Should the results be return in the original (input) order or in the order inside the graph?

...
: Additional arguments. Expert use only.

data
: **[Deprecated]** Use `newdata` instead.

Value

A list with elements `mean`, which contains the means of the predictions, `fe_mean`, which is the prediction for the fixed effects, `re_mean`, which is the prediction for the random effects, `variance` (if `compute_variance` is `TRUE`), which contains the variances of the predictions, `samples` (if `posterior_samples` is `TRUE`), which contains the posterior samples.

predict.spacetimeobj *Prediction of a space-time SPDE*

Description

The function is used for computing kriging predictions based on data $Y_i = u(s_i, t_i) + \epsilon_i$, where ϵ is mean-zero Gaussian measurement noise and $u(s, t)$ is defined by a spatio-temporal SPDE as described in `spacetime.operators()`.

Usage

```
## S3 method for class 'spacetimeobj'
predict(
  object,
  A,
  Aprd,
  Y,
  sigma.e,
  mu = 0,
  compute.variances = FALSE,
  posterior_samples = FALSE,
  n_samples = 100,
  only_latent = FALSE,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `spacetime.operators()`
`A`	A matrix linking the measurement locations to the basis of the FEM approximation of the latent model.
`Aprd`	A matrix linking the prediction locations to the basis of the FEM approximation of the latent model.
`Y`	A vector with the observed data, can also be a matrix where the columns are observations of independent replicates of u.
`sigma.e`	The standard deviation of the Gaussian measurement noise. Put to zero if the model does not have measurement noise.
`mu`	Expectation vector of the latent field (default = 0).
`compute.variances`	Set to also TRUE to compute the kriging variances.
`posterior_samples`	If `TRUE`, posterior samples will be returned.
`n_samples`	Number of samples to be returned. Will only be used if `sampling` is `TRUE`.
`only_latent`	Should the posterior samples be only given to the laten model?
`...`	further arguments passed to or from other methods.

Value

A list with elements

`mean`	The kriging predictor (the posterior mean of u|Y).
`variance`	The posterior variances (if computed).

Examples

```
s <- seq(from = 0, to = 20, length.out = 101)
t <- seq(from = 0, to = 20, length.out = 31)

op_cov <- spacetime.operators(space_loc = s, time_loc = t,
                             kappa = 5, sigma = 10, alpha = 1,
                             beta = 2, rho = 1, gamma = 0.05)
# generate data
sigma.e <- 0.01
n.obs <- 500
obs.loc <- data.frame(x = max(s)*runif(n.obs),
                     t = max(t)*runif(n.obs))
A <- rSPDE.Ast(space_loc = s, time_loc = t, obs.s = obs.loc$x, obs.t = obs.loc$t)
Aprd <- Diagonal(dim(A)[2])
x <- simulate(op_cov, nsim = 1)
Y <- A%*%x + sigma.e*rnorm(n.obs)
u.krig <- predict(op_cov, A, Aprd, Y, sigma.e)
```

`rational.order` *Get the order of rational approximation.*

Description

Get the order of rational approximation.

Usage

```
rational.order(object)
```

Arguments

`object`	A `CBrSPDEobj` object or an `inla_rspde` object.

Value

The order of rational approximation.

`rational.order<-` *Changing the order of the rational approximation*

Description

Changing the order of the rational approximation

Usage

```
rational.order(x) <- value
```

Arguments

`x`	A `CBrSPDE` or an `rpsde.inla` object
`value`	The order of rational approximation.

Value

An object of the same class with the new order of rational approximation.

`rational.type` *Get type of rational approximation.*

Description

Get type of rational approximation.

Usage

```
rational.type(object)
```

Arguments

`object`	A `CBrSPDEobj` object or an `inla_rspde` object.

Value

The type of rational approximation.

rational.type<- *Changing the type of the rational approximation*

Description

Changing the type of the rational approximation

Usage

```
rational.type(x) <- value
```

Arguments

x	A CBrSPDE or an rpsde.inla object
value	The type of rational approximation. The current options are "chebfun", "brasil" and "chebfunLB"

Value

An object of the same class with the new rational approximation.

require.nowarnings *Warnings free loading of add-on packages*

Description

Turn off all warnings for require(), to allow clean completion of examples that require unavailable Suggested packages.

Usage

```
require.nowarnings(package, lib.loc = NULL, character.only = FALSE)
```

Arguments

package	The name of a package, given as a character string.
lib.loc	a character vector describing the location of R library trees to search through, or NULL. The default value of NULL corresponds to all libraries currently known to .libPaths(). Non-existent library trees are silently ignored.
character.only	a logical indicating whether package can be assumed to be a character string.

Details

require(package) acts the same as require(package, quietly = TRUE) but with warnings turned off. In particular, no warning or error is given if the package is unavailable. Most cases should use requireNamespace(package, quietly = TRUE) instead, which doesn't produce warnings.

Value

require.nowarnings returns (invisibly) TRUE if it succeeds, otherwise FALSE

See Also

require()

Examples

```
## This should produce no output:
if (require.nowarnings(nonexistent)) {
  message("Package loaded successfully")
}
```

---

rSPDE.A1d *Observation matrix for finite element discretization on R*

Description

A finite element discretization on R can be written as $u(s) = \sum_i^n u_i \varphi_i(s)$ where $\varphi_i(s)$ is a piecewise linear "hat function" centered at location x_i. This function computes an $m \times n$ matrix A that links the basis function in the expansion to specified locations $s = (s_1, \ldots, s_m)$ in the domain through $A_i j = \varphi_j(s_i)$.

Usage

```
rSPDE.A1d(x, loc)
```

Arguments

x	The locations of the nodes in the FEM discretization.
loc	The locations $(s_1, \ldots, s_m)$

Value

The sparse matrix A.

Author(s)

David Bolin <davidbolin@gmail.com>

See Also

rSPDE.fem1d()

Examples

```
# create mass and stiffness matrices for a FEM discretization on [0,1]
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# create the observation matrix for some locations in the domain
obs.loc <- runif(n = 10, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
```

`rspde.anistropic2d` *Rational approximations of stationary anisotropic Gaussian Matern random fields*

Description

`rspde.anistropic2d` computes a Finite Element Method (FEM) approximation of a Gaussian random field defined as the solution to the stochastic partial differential equation (SPDE):

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\sqrt{h^T H^{-1} h})^\nu K_\nu(\sqrt{h^T H^{-1} h})$$

, based on a SPDE representation of the form

$$(I - \nabla \cdot (H\nabla))^{(\nu+1)/2} u = c\sigma W$$

, where $c>0$ is a constant. The matrix H is defined as

$$\begin{bmatrix} h_x^2 & h_x h_y h_{xy} \\ h_x h_y h_{xy} & h_y^2 \end{bmatrix}$$

Usage

```
rspde.anistropic2d(
  mesh,
  nu = NULL,
  nu.upper.bound = 2,
  rspde.order = 1,
  prior.hx = NULL,
  prior.hy = NULL,
  prior.hxy = NULL,
  prior.sigma = NULL,
  prior.precision = NULL,
  prior.nu = NULL,
  prior.nu.dist = "lognormal",
  nu.prec.inc = 0.01,
  type.rational.approx = "chebfun",
  shared_lib = "detect",
  debug = FALSE,
  ...
)
```

Arguments

`mesh`	Spatial mesh for the FEM approximation.
`nu`	If nu is set to a parameter, nu will be kept fixed and will not be estimated. If nu is `NULL`, it will be estimated.
`nu.upper.bound`	Upper bound for the smoothness parameter ν. If `NULL`, it will be set to 2.
`rspde.order`	The order of the covariance-based rational SPDE approach. The default order is 1.
`prior.hx`	A list specifying the prior for the parameter h_x in the matrix H. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars. The precision refers to the prior on $\log(h_x)$. If `NULL`, default values will be used. The `mean` value is also used as starting value for hx.
`prior.hy`	A list specifying the prior for the parameter h_y in the matrix H. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars. The precision refers to the prior on $\log(h_x)$. If `NULL`, default values will be used. The `mean` value is also used as starting value for hy.
`prior.hxy`	A list specifying the prior for the parameter h_x in the matrix H. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars. The precision refers to the prior on $\log((h_{xy}+1)/(1-h_{xy}))$. If `NULL`, default values will be used. The `mean` value is also used as starting value for hxy.
`prior.sigma`	A list specifying the prior for the variance parameter σ. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars. The precision refers to the prior on $\log(\sigma)$. If `NULL`, default values will be used. The `mean` value is also used as starting value for sigma.
`prior.precision`	A precision matrix for $\log(h_x), \log(h_y), \log((h_{xy}+1)/(1-h_{xy})), \log(\sigma)$. This matrix replaces the precision element from `prior.kappa`, `prior.sigma`, `prior.gamma`, and `prior.rho` respectively. For dimension 1 `prior.precision` must be a 4x4 matrix. For dimension 2, ρ is a vector of length 2, so in this case `prior.precision` must be a 5x5 matrix. If `NULL`, a diagonal precision matrix with default values will be used.
`prior.nu`	a list containing the elements `mean` and `prec` for beta distribution, or `loglocation` and `logscale` for a truncated lognormal distribution. `loglocation` stands for the location parameter of the truncated lognormal distribution in the log scale. `prec` stands for the precision of a beta distribution. `logscale` stands for the scale of the truncated lognormal distribution on the log scale. Check details below.
`prior.nu.dist`	The distribution of the smoothness parameter. The current options are "beta" or "lognormal". The default is "lognormal".
`nu.prec.inc`	Amount to increase the precision in the beta prior distribution. Check details below.
`type.rational.approx`	Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

shared_lib	String specifying which shared library to use for the Cgeneric implementation. Options are "detect", "INLA", or "rSPDE". You may also specify the direct path to a .so (or .dll) file.
debug	Logical value indicating whether to enable INLA debug mode.
...	Additional arguments passed internally for configuration purposes.

Value

An object of class `inla_rspde_spacetime` representing the FEM approximation of the space-time Gaussian random field.

`rSPDE.Ast` *Observation matrix for space-time models*

Description

Observation matrix for space-time models

Usage

```
rSPDE.Ast(
  mesh_space = NULL,
  space_loc = NULL,
  mesh_time = NULL,
  time_loc = NULL,
  graph = NULL,
  obs.s = NULL,
  obs.t = NULL
)
```

Arguments

mesh_space	mesh object for models on 1d or 2d domains
space_loc	mesh locations for models on 1d domains
mesh_time	mesh object for time discretization
time_loc	mesh locations for time discretization
graph	MetricGraph object for models on metric graphs
obs.s	spatial locations of observations
obs.t	time points for observations

Value

Observation matrix linking observation locations to mesh nodes

Examples

```
s <- seq(from = 0, to = 20, length.out = 11)
t <- seq(from = 0, to = 20, length.out = 5)
n.obs <- 10
obs.loc <- data.frame(x = max(s)*runif(n.obs),
t = max(t)*runif(n.obs))
A <- rSPDE.Ast(space_loc = s,time_loc = t,
               obs.s = obs.loc$x, obs.t = obs.loc$t)
```

---

rSPDE.construct.matern.loglike

Constructor of Matern loglikelihood functions.

Description

This function returns a log-likelihood function for a Gaussian process with a Matern covariance function, that is observed under Gaussian measurement noise: $Y_i = u(s_i) + \epsilon_i$, where ϵ_i are iid mean-zero Gaussian variables. The latent model is approximated using the a rational approximation of the fractional SPDE model corresponding to the Gaussian process.

Usage

```
rSPDE.construct.matern.loglike(
  object,
  Y,
  A,
  sigma.e = NULL,
  mu = 0,
  nu = NULL,
  tau = NULL,
  kappa = NULL,
  sigma = NULL,
  range = NULL,
  parameterization = c("spde", "matern"),
  m = NULL,
  log_scale = TRUE,
  return_negative_likelihood = TRUE
)
```

Arguments

object	The rational SPDE approximation, computed using `matern.operators()`
Y	The observations, either a vector or a matrix where the columns correspond to independent replicates of observations.
A	An observation matrix that links the measurement location to the finite element basis.

sigma.e	IF non-null, the standard deviation of the measurement noise will be kept fixed in the returned likelihood.
mu	Expectation vector of the latent field (default = 0).
nu	If non-null, the shape parameter will be kept fixed in the returned likelihood.
tau	If non-null, the tau parameter will be kept fixed in the returned likelihood. (Replaces sigma)
kappa	If non-null, the range parameter will be kept fixed in the returned likelihood.
sigma	If non-null, the standard deviation will be kept fixed in the returned likelihood.
range	If non-null, the range parameter will be kept fixed in the returned likelihood. (Replaces kappa)
parameterization	If spde, then one will use the parameters tau and kappa. If matern, then one will use the parameters sigma and range.
m	If non-null, update the order of the rational approximation, which needs to be a positive integer.
log_scale	Should the parameters be evaluated in log-scale?
return_negative_likelihood	Return minus the likelihood to turn the maximization into a minimization?

Value

The log-likelihood function. The parameters of the returned function are given in the order sigma, kappa, nu, sigma.e, whenever they are available.

See Also

matern.operators(), predict.CBrSPDEobj()

Examples

```
# this example illustrates how the function can be used for maximum
# likelihood estimation

set.seed(123)
# Sample a Gaussian Matern process on R using a rational approximation
nu <- 0.8
sigma <- 1
sigma.e <- 0.1
n.rep <- 10
n.obs <- 200
n.x <- 51
range <- 0.2
# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = n.x)
# Compute the covariance-based rational approximation
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
```

```
  parameterization = "matern"
)
# Sample the model
u <- simulate(op_cov, n.rep)
# Create some data
obs.loc <- runif(n = n.obs, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
noise <- rnorm(n.obs * n.rep)
dim(noise) <- c(n.obs, n.rep)
Y <- as.matrix(A %*% u + sigma.e * noise)

# Define the negative likelihood function for optimization
# using CBrSPDE.matern.loglike
# Matern parameterization
loglike <- rSPDE.construct.matern.loglike(op_cov, Y, A, parameterization = "matern")

# The parameters can now be estimated by minimizing mlik with optim

# Choose some reasonable starting values depending on the size of the domain
theta0 <- c(
  get.initial.values.rSPDE(mesh.range = 1, dim = 1),
  log(0.1 * sd(as.vector(Y)))
)
# run estimation and display the results
theta <- optim(theta0, loglike,
  method = "L-BFGS-B"
)
print(data.frame(
  sigma = c(sigma, exp(theta$par[1])), range = c(range, exp(theta$par[2])),
  nu = c(nu, exp(theta$par[3])), sigma.e = c(sigma.e, exp(theta$par[4])),
  row.names = c("Truth", "Estimates")
))
```

`rSPDE.fem1d` *Finite element calculations for problems on R*

Description

This function computes mass and stiffness matrices for a FEM approximation on R, assuming Neumann boundary conditions. These matrices are needed when discretizing the operators in rational approximations.

Usage

```
rSPDE.fem1d(x)
```

Arguments

x	Locations of the nodes in the FEM approximation.

Value

The function returns a list with the following elements

`G`	The stiffness matrix with elements $(\nabla \phi_i, \nabla \phi_j)$.
`C`	The mass matrix with elements (ϕ_i, ϕ_j).
`Cd`	Mass lumped mass matrix.
`B`	Matrix with elements $(\nabla \phi_i, \phi_j)$.

Author(s)

David Bolin `<davidbolin@gmail.com>`

See Also

`rSPDE.A1d()`

Examples

```
# create mass and stiffness matrices for a FEM discretization on [0,1]
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)
```

`rSPDE.fem2d` *Finite element calculations for problems in 2D*

Description

This function computes mass and stiffness matrices for a mesh in 2D, assuming Neumann boundary conditions.

Usage

```
rSPDE.fem2d(FV, P)
```

Arguments

`FV`	Matrix where each row defines a triangle
`P`	Locations of the nodes in the mesh.

Value

The function returns a list with the following elements

- `G` The stiffness matrix with elements $(\nabla\phi_i, \nabla\phi_j)$.
- `C` The mass matrix with elements (ϕ_i, ϕ_j).
- `Cd` The mass lumped matrix with diagonal elements $(\phi_i, 1)$.
- `Hxx` Matrix with elements $(\partial_x\phi_i, \partial_x\phi_j)$.
- `Hyy` Matrix with elements $(\partial_y\phi_i, \partial_y\phi_j)$.
- `Hxy` Matrix with elements $(\partial_x\phi_i, \partial_y\phi_j)$.
- `Hyx` Matrix with elements $(\partial_y\phi_i, \partial_x\phi_j)$.
- `Bx` Matrix with elements $(\partial_x\phi_i, \phi_j)$.
- `By` Matrix with elements $(\partial_y\phi_i, \phi_j)$.

Author(s)

David Bolin <davidbolin@gmail.com>

See Also

`rSPDE.fem1d()`

Examples

```
P <- rbind(c(0, 0), c(1, 0), c(1, 1), c(0, 1))
FV <- rbind(c(1, 2, 3), c(2, 3, 4))
fem <- rSPDE.fem2d(FV, P)
```

`rSPDE.loglike` *Object-based log-likelihood function for latent Gaussian fractional SPDE model*

Description

This function evaluates the log-likelihood function for a fractional SPDE model $L^\beta u(s) = W$ that is observed under Gaussian measurement noise: $Y_i = u(s_i) + \epsilon_i$, where ϵ_i are iid mean-zero Gaussian variables and $x(s) = \mu(s) + u(s)$, where $\mu(s)$ is the expectation vector of the latent field.

Usage

```
rSPDE.loglike(obj, Y, A, sigma.e, mu = 0)
```

Arguments

obj	The rational SPDE approximation, computed using fractional.operators(), matern.operators(), or spde.matern.operators().
Y	The observations, either a vector or a matrix where the columns correspond to independent replicates of observations.
A	An observation matrix that links the measurement location to the finite element basis.
sigma.e	The standard deviation of the measurement noise.
mu	Expectation vector of the latent field (default = 0).

Value

The log-likelihood value.

Note

This example below shows how the function can be used to evaluate the likelihood of a latent Matern model.

See Also

spde.matern.loglike()

Examples

```
# Sample a Gaussian Matern process on R using a rational approximation
kappa <- 10
sigma <- 1
nu <- 0.8
sigma.e <- 0.3
range <- 0.2

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation
op <- matern.operators(
  range = range, sigma = sigma, nu = nu,
  loc_mesh = x, d = 1,
  type = "operator", parameterization = "matern"
)

# Sample the model
u <- simulate(op)

# Create some data
obs.loc <- runif(n = 10, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
Y <- as.vector(A %*% u + sigma.e * rnorm(10))
```

```
# compute log-likelihood of the data
lik1 <- rSPDE.loglike(op, Y, A, sigma.e)
cat(lik1)
```

---

rspde.make.A *Observation/prediction matrices for rSPDE models.*

Description

Constructs observation/prediction weight matrices for rSPDE models based on `inla.mesh` or `inla.mesh.1d` objects.

Usage

```
rspde.make.A(
  mesh = NULL,
  loc = NULL,
  A = NULL,
  dim = NULL,
  rspde.order = 1,
  nu = NULL,
  index = NULL,
  group = NULL,
  repl = 1L,
  n.group = NULL,
  n.repl = NULL
)
```

Arguments

`mesh`	An `inla.mesh`, an `inla.mesh.1d` object or a `metric_graph` object.
`loc`	Locations, needed if an INLA mesh is provided
`A`	The A matrix from the standard SPDE approach, such as the matrix returned by `inla.spde.make.A`. Should only be provided if `mesh` is not provided.
`dim`	the dimension. Should only be provided if an `mesh` is not provided.
`rspde.order`	The order of the covariance-based rational SPDE approach.
`nu`	If `NULL`, then the model will assume that nu will be estimated. If nu is fixed, you should provide the value of nu.
`index`	For each observation/prediction value, an index into loc. Default is seq_len(nrow(A.loc)).
`group`	For each observation/prediction value, an index into the group model.
`repl`	For each observation/prediction value, the replicate index.
`n.group`	The size of the group model.
`n.repl`	The total number of replicates.

Value

The A matrix for rSPDE models.

Examples

```
 #tryCatch version
tryCatch({
if (requireNamespace("INLA", quietly = TRUE)) {
  library(INLA)

  set.seed(123)
  loc <- matrix(runif(100 * 2) * 100, 100, 2)
  mesh <- inla.mesh.2d(
    loc = loc,
    cutoff = 50,
    max.edge = c(50, 500)
  )
  A <- rspde.make.A(mesh, loc = loc, rspde.order = 3)
}
#stable.tryCatch
}, error = function(e){print("Could not run the example")})
```

---

rspde.make.index *rSPDE model index vector generation*

Description

Generates a list of named index vectors for an rSPDE model.

Usage

```
rspde.make.index(
  name,
  n.spde = NULL,
  n.group = 1,
  n.repl = 1,
  mesh = NULL,
  rspde.order = 1,
  nu = NULL,
  dim = NULL
)
```

Arguments

`name`	A character string with the base name of the effect.
`n.spde`	The number of basis functions in the mesh model.
`n.group`	The size of the group model.

n.repl	The total number of replicates.
mesh	An inla.mesh, an inla.mesh.1d object or a metric_graph object.
rspde.order	The order of the rational approximation
nu	If NULL, then the model will assume that nu will be estimated. If nu is fixed, you should provide the value of nu.
dim	the dimension of the domain. Should only be provided if mesh is not provided.

Value

A list of named index vectors.

name	Indices into the vector of latent variables
name.group	'group' indices
name.repl	Indices for replicates

Examples

```
 #tryCatch version
tryCatch({
if (requireNamespace("INLA", quietly = TRUE)) {
  library(INLA)

  set.seed(123)

  m <- 100
  loc_2d_mesh <- matrix(runif(m * 2), m, 2)
  mesh_2d <- inla.mesh.2d(
    loc = loc_2d_mesh,
    cutoff = 0.05,
    max.edge = c(0.1, 0.5)
  )
  sigma <- 1
  range <- 0.2
  nu <- 0.8
  kappa <- sqrt(8 * nu) / range
  op <- matern.operators(
    mesh = mesh_2d, nu = nu,
    range = range, sigma = sigma, m = 2,
    parameterization = "matern"
  )
  u <- simulate(op)
  A <- inla.spde.make.A(
    mesh = mesh_2d,
    loc = loc_2d_mesh
  )
  sigma.e <- 0.1
  y <- A %*% u + rnorm(m) * sigma.e
  Abar <- rspde.make.A(mesh = mesh_2d, loc = loc_2d_mesh)
  mesh.index <- rspde.make.index(name = "field", mesh = mesh_2d)
  st.dat <- inla.stack(
```

```
    data = list(y = as.vector(y)),
    A = Abar,
    effects = mesh.index
  )
  rspde_model <- rspde.matern(
    mesh = mesh_2d,
    nu.upper.bound = 2
  )
  f <- y ~ -1 + f(field, model = rspde_model)
  rspde_fit <- inla(f,
    data = inla.stack.data(st.dat),
    family = "gaussian",
    control.predictor =
      list(A = inla.stack.A(st.dat))
  )
  result <- rspde.result(rspde_fit, "field", rspde_model)
  summary(result)
}
#stable.tryCatch
}, error = function(e){print("Could not run the example")})
```

`rspde.matern` *Matern rSPDE model object for INLA*

Description

Creates an INLA object for a stationary Matern model with general smoothness parameter.

Usage

```
rspde.matern(
  mesh,
  nu.upper.bound = NULL,
  rspde.order = 1,
  nu = NULL,
  B.sigma = matrix(c(0, 1, 0), 1, 3),
  B.range = matrix(c(0, 0, 1), 1, 3),
  parameterization = c("spde", "matern", "matern2"),
  B.tau = matrix(c(0, 1, 0), 1, 3),
  B.kappa = matrix(c(0, 0, 1), 1, 3),
  start.nu = NULL,
  start.theta = NULL,
  prior.nu = NULL,
  theta.prior.mean = NULL,
  theta.prior.prec = 0.1,
  prior.std.dev.nominal = 1,
  prior.range.nominal = NULL,
  prior.kappa.mean = NULL,
```

```
  prior.tau.mean = NULL,
  start.lstd.dev = NULL,
  start.lrange = NULL,
  start.ltau = NULL,
  start.lkappa = NULL,
  prior.theta.param = c("theta", "spde"),
  prior.nu.dist = c("beta", "lognormal"),
  nu.prec.inc = 1,
  type.rational.approx = c("chebfun", "brasil", "chebfunLB"),
  debug = FALSE,
  shared_lib = "detect",
  ...
)
```

Arguments

- `mesh` The mesh to build the model. It can be an `inla.mesh` or an `inla.mesh.1d` object. Otherwise, should be a list containing elements d, the dimension, C, the mass matrix, and G, the stiffness matrix.
- `nu.upper.bound` Upper bound for the smoothness parameter. If `NULL`, it will be set to 2.
- `rspde.order` The order of the covariance-based rational SPDE approach. The default order is 1.
- `nu` If nu is set to a parameter, nu will be kept fixed and will not be estimated. If nu is `NULL`, it will be estimated.
- `B.sigma` Matrix with specification of log-linear model for σ (for 'matern' parameterization) or for σ^2 (for 'matern2' parameterization). Will be used if `parameterization = 'matern'` or `parameterization = 'matern2'`.
- `B.range` Matrix with specification of log-linear model for ρ, which is a range-like parameter (it is exactly the range parameter in the stationary case). Will be used if `parameterization = 'matern'` or `parameterization = 'matern2'`.
- `parameterization` Which parameterization to use? `matern` uses range, std. deviation and nu (smoothness). `spde` uses kappa, tau and nu (smoothness). `matern2` uses range-like (1/kappa), variance and nu (smoothness). The default is `spde`.
- `B.tau` Matrix with specification of log-linear model for τ. Will be used if `parameterization = 'spde'`.
- `B.kappa` Matrix with specification of log-linear model for κ. Will be used if `parameterization = 'spde'`.
- `start.nu` Starting value for nu.
- `start.theta` Starting values for the model parameters. In the stationary case, if `parameterization='matern'`, then `theta[1]` is the std.dev and `theta[2]` is the range parameter. If `parameterization = 'spde'`, then `theta[1]` is `tau` and `theta[2]` is `kappa`.
- `prior.nu` a list containing the elements `mean` and `prec` for beta distribution, or `loglocation` and `logscale` for a truncated lognormal distribution. `loglocation` stands for the location parameter of the truncated lognormal distribution in the log scale. `prec` stands for the precision of a beta distribution. `logscale` stands for the

scale of the truncated lognormal distribution on the log scale. Check details below.

`theta.prior.mean`
: A vector for the mean priors of `theta`.

`theta.prior.prec`
: A precision matrix for the prior of `theta`.

`prior.std.dev.nominal`
: Prior std. deviation to be used for the priors and for the starting values.

`prior.range.nominal`
: Prior range to be used for the priors and for the starting values.

`prior.kappa.mean`
: Prior kappa to be used for the priors and for the starting values.

`prior.tau.mean`
: Prior tau to be used for the priors and for the starting values.

`start.lstd.dev`
: Starting value for log of std. deviation. Will not be used if start.ltau is non-null. Will be only used in the stationary case and if `parameterization = 'matern'`.

`start.lrange`
: Starting value for log of range. Will not be used if start.lkappa is non-null. Will be only used in the stationary case and if `parameterization = 'matern'`.

`start.ltau`
: Starting value for log of tau. Will be only used in the stationary case and if `parameterization = 'spde'`.

`start.lkappa`
: Starting value for log of kappa. Will be only used in the stationary case and if `parameterization = 'spde'`.

`prior.theta.param`
: Should the lognormal prior be on `theta` or on the SPDE parameters (`tau` and `kappa` on the stationary case)?

`prior.nu.dist`
: The distribution of the smoothness parameter. The current options are "beta" or "lognormal". The default is "lognormal".

`nu.prec.inc`
: Amount to increase the precision in the beta prior distribution. Check details below.

`type.rational.approx`
: Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

`debug`
: INLA debug argument

`shared_lib`
: Which shared lib to use for the cgeneric implementation? If "detect", it will check if the shared lib exists locally, in which case it will use it. Otherwise it will use INLA's shared library. If "INLA", it will use the shared lib from INLA's installation. If 'rSPDE', then it will use the local installation (does not work if your installation is from CRAN). Otherwise, you can directly supply the path of the .so (or .dll) file.

`...`
: Only being used internally.

`prior.kappa`
: a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale.

`prior.tau`
: a list containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale.

prior.range
: a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale. Will not be used if prior.kappa is non-null.

prior.std.dev
: a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale. Will not be used if prior.tau is non-null.

Value

An INLA model.

`rspde.matern.intrinsic` *Intrinsic Matern rSPDE model object for INLA*

Description

Creates an INLA object for a stationary intrinsic Matern model. Currently, alpha is fixed to 2 and beta is fixed to 1.

Usage

```
rspde.intrinsic.matern(
  mesh,
  alpha = 2,
  mean.correction = FALSE,
  prior.lkappa.mean = NULL,
  prior.ltau.mean = 1,
  prior.lkappa.prec = 0.1,
  prior.ltau.prec = 0.1,
  start.ltau = NULL,
  start.lkappa = NULL,
  true.scaling = TRUE,
  diagonal = 0,
  debug = FALSE,
  shared_lib = "detect",
  ...
)
```

Arguments

mesh
: The mesh to build the model. It can be an `inla.mesh` or an `inla.mesh.1d` object. Otherwise, should be a list containing elements d, the dimension, C, the mass matrix, and G, the stiffness matrix.

alpha
: Smoothness parameter, need to be 1 or 2.

mean.correction
: Add mean correction for extreme value models?

prior.lkappa.mean
: Prior on log kappa to be used for the priors and for the starting values.

`prior.ltau.mean`
: Prior on log tau to be used for the priors and for the starting values.

`prior.lkappa.prec`
: Precision to be used on the prior on log kappa to be used for the priors and for the starting values.

`prior.ltau.prec`
: Precision to be used on the prior on log tau to be used for the priors and for the starting values.

`start.ltau`
: Starting value for log of tau.

`start.lkappa`
: Starting value for log of kappa.

`true.scaling`
: Compute the true normalizing constant manually? Default `TRUE`. The alternative is to set this to `FALSE` and set the `diagonal` argument to some small positive value. In the latter case, the model is approximated by a non-intrinsic model with a precision matrix that has the `diagonal` value added to the diagonal.

`diagonal`
: Value of diagonal correction for INLA stability. Default 0.

`debug`
: INLA debug argument

`shared_lib`
: Which shared lib to use for the cgeneric implementation? If "detect", it will check if the shared lib exists locally, in which case it will use it. Otherwise it will use INLA's shared library. If "INLA", it will use the shared lib from INLA's installation. If 'rSPDE', then it will use the local installation (does not work if your installation is from CRAN). Otherwise, you can directly supply the path of the .so (or .dll) file.

`...`
: Only being used internally.

Value

An INLA model.

`rSPDE.matern.loglike` *Object-based log-likelihood function for latent Gaussian fractional SPDE model using the rational approximations*

Description

This function evaluates the log-likelihood function for a Gaussian process with a Matern covariance function, that is observed under Gaussian measurement noise: $Y_i = u(s_i) + \epsilon_i$, where ϵ_i are iid mean-zero Gaussian variables. The latent model is approximated using the a rational approximation of the fractional SPDE model corresponding to the Gaussian process.

Usage

```
rSPDE.matern.loglike(
  object,
  Y,
  A,
```

```
  sigma.e,
  mu = 0,
  nu = NULL,
  kappa = NULL,
  sigma = NULL,
  range = NULL,
  tau = NULL,
  m = NULL
)
```

Arguments

object	The rational SPDE approximation, computed using matern.operators()
Y	The observations, either a vector or a matrix where the columns correspond to independent replicates of observations.
A	An observation matrix that links the measurement location to the finite element basis.
sigma.e	The standard deviation of the measurement noise.
mu	Expectation vector of the latent field (default = 0).
nu	If non-null, update the shape parameter of the covariance function.
kappa	If non-null, update the range parameter of the covariance function.
sigma	If non-null, update the standard deviation of the covariance function.
range	If non-null, update the range parameter of the covariance function.
tau	If non-null, update the parameter tau.
m	If non-null, update the order of the rational approximation, which needs to be a positive integer.

Value

The log-likelihood value.

See Also

matern.operators(), predict.CBrSPDEobj()

Examples

```
# this example illustrates how the function can be used for maximum likelihood estimation

set.seed(123)
# Sample a Gaussian Matern process on R using a rational approximation
nu <- 0.8
kappa <- 5
sigma <- 1
sigma.e <- 0.1
n.rep <- 10
n.obs <- 100
```

```
n.x <- 51
range <- 0.2

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = n.x)
fem <- rSPDE.fem1d(x)

tau <- sqrt(gamma(nu) / (sigma^2 * kappa^(2 * nu) *
  (4 * pi)^(1 / 2) * gamma(nu + 1 / 2)))

# Compute the covariance-based rational approximation
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
  parameterization = "matern"
)

# Sample the model
u <- simulate(op_cov, n.rep)

# Create some data
obs.loc <- runif(n = n.obs, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
noise <- rnorm(n.obs * n.rep)
dim(noise) <- c(n.obs, n.rep)
Y <- as.matrix(A %*% u + sigma.e * noise)

# Define the negative likelihood function for optimization
# using CBrSPDE.matern.loglike

# Notice that we are also using sigma instead of tau, so it can be compared
# to matern.loglike()
mlik_cov <- function(theta, Y, A, op_cov) {
  kappa <- exp(theta[1])
  sigma <- exp(theta[2])
  nu <- exp(theta[3])
  return(-rSPDE.matern.loglike(
    object = op_cov, Y = Y,
    A = A, kappa = kappa, sigma = sigma,
    nu = nu, sigma.e = exp(theta[4])
  ))
}

# The parameters can now be estimated by minimizing mlik with optim

# Choose some reasonable starting values depending on the size of the domain
theta0 <- log(c(sqrt(8), 1 / sqrt(var(c(Y))), 0.9, 0.01))

# run estimation and display the results
theta <- optim(theta0, mlik_cov,
  Y = Y, A = A, op_cov = op_cov,
  method = "L-BFGS-B"
)
```

```
print(data.frame(
  range = c(range, exp(theta$par[1])), sigma = c(sigma, exp(theta$par[2])),
  nu = c(nu, exp(theta$par[3])), sigma.e = c(sigma.e, exp(theta$par[4])),
  row.names = c("Truth", "Estimates")
))
```

---

rspde.matern.precision

Precision matrix of the covariance-based rational approximation of stationary Gaussian Matern random fields

---

Description

rspde.matern.precision is used for computing the precision matrix of the covariance-based rational SPDE approximation of a stationary Gaussian random fields on R^d with a Matern covariance function

$$C(h) = \frac{\sigma^2}{2^(\nu - 1)\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h)$$

Usage

```
rspde.matern.precision(
  kappa,
  nu,
  tau = NULL,
  sigma = NULL,
  rspde.order,
  dim,
  fem_mesh_matrices,
  only_fractional = FALSE,
  return_block_list = FALSE,
  type_rational_approx = "chebfun"
)
```

Arguments

kappa	Range parameter of the covariance function.
nu	Shape parameter of the covariance function.
tau	Scale parameter of the covariance function. If sigma is not provided, tau should be provided.
sigma	Standard deviation of the covariance function. If tau is not provided, sigma should be provided.
rspde.order	The order of the rational approximation

`dim` The dimension of the domain

`fem_mesh_matrices`
A list containing the FEM-related matrices. The list should contain elements c0, g1, g2, g3, etc.

`only_fractional`
Logical. Should only the fractional-order part of the precision matrix be returned?

`return_block_list`
Logical. For `type = "covariance"`, should the block parts of the precision matrix be returned separately as a list?

`type_rational_approx`
Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

Value

The precision matrix

Examples

```
set.seed(123)
nobs <- 101
x <- seq(from = 0, to = 1, length.out = nobs)
fem <- rSPDE.fem1d(x)
kappa <- 40
sigma <- 1
d <- 1
nu <- 2.6
tau <- sqrt(gamma(nu) / (kappa^(2 * nu) * (4 * pi)^(d / 2) *
  gamma(nu + d / 2)))
range <- sqrt(8 * nu) / kappa
op_cov <- matern.operators(
  loc_mesh = x, nu = nu, range = range, sigma = sigma,
  d = 1, m = 2, compute_higher_order = TRUE,
  parameterization = "matern"
)
v <- t(rSPDE.A1d(x, 0.5))
c.true <- matern.covariance(abs(x - 0.5), kappa, nu, sigma)
Q <- rspde.matern.precision(
  kappa = kappa, nu = nu, tau = tau, rspde.order = 2, d = 1,
  fem_mesh_matrices = op_cov$fem_mesh_matrices
)
A <- Diagonal(nobs)
Abar <- cbind(A, A, A)
w <- rbind(v, v, v)
c.approx_cov <- (Abar) %*% solve(Q, w)

# plot the result and compare with the true Matern covariance
plot(x, matern.covariance(abs(x - 0.5), kappa, nu, sigma),
  type = "l", ylab = "C(h)",
  xlab = "h", main = "Matern covariance and rational approximations"
```

```
)
lines(x, c.approx_cov, col = 2)
```

---

rspde.matern.precision.integer *Precision matrix of stationary Gaussian Matern random fields with integer covariance exponent*

Description

rspde.matern.precision.integer.opt is used for computing the precision matrix of stationary Gaussian random fields on R^d with a Matern covariance function

$$C(h) = \frac{\sigma^2}{2^(\nu - 1)\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h)$$

, where $\alpha = \nu + d/2$ is a natural number.

Usage

```
rspde.matern.precision.integer(
  kappa,
  nu,
  tau = NULL,
  sigma = NULL,
  dim,
  fem_mesh_matrices
)
```

Arguments

kappa	Range parameter of the covariance function.
nu	Shape parameter of the covariance function.
tau	Scale parameter of the covariance function.
sigma	Standard deviation of the covariance function. If tau is not provided, sigma should be provided.
dim	The dimension of the domain
fem_mesh_matrices	A list containing the FEM-related matrices. The list should contain elements c0, g1, g2, g3, etc.

Value

The precision matrix

Examples

```
set.seed(123)
nobs <- 101
x <- seq(from = 0, to = 1, length.out = nobs)
fem <- rSPDE.fem1d(x)
kappa <- 40
sigma <- 1
d <- 1
nu <- 0.5
tau <- sqrt(gamma(nu) / (kappa^(2 * nu) *
  (4 * pi)^(d / 2) * gamma(nu + d / 2)))
range <- sqrt(8 * nu) / kappa
op_cov <- matern.operators(
  loc_mesh = x, nu = nu, range = range, sigma = sigma,
  d = 1, m = 2, parameterization = "matern"
)
v <- t(rSPDE.A1d(x, 0.5))
c.true <- matern.covariance(abs(x - 0.5), kappa, nu, sigma)
Q <- rspde.matern.precision.integer(
  kappa = kappa, nu = nu, tau = tau, d = 1,
  fem_mesh_matrices = op_cov$fem_mesh_matrices
)
A <- Diagonal(nobs)
c.approx_cov <- A %*% solve(Q, v)

# plot the result and compare with the true Matern covariance
plot(x, matern.covariance(abs(x - 0.5), kappa, nu, sigma),
  type = "l", ylab = "C(h)",
  xlab = "h", main = "Matern covariance and rational approximations"
)
lines(x, c.approx_cov, col = 2)
```

`rspde.matern.precision.integer.opt`
Optimized precision matrix of stationary Gaussian Matern random fields with integer covariance exponent

---

Description

`rspde.matern.precision.integer.opt` is used for computing the optimized version of the precision matrix of stationary Gaussian random fields on R^d with a Matern covariance function

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h),$$

where $\alpha = \nu + d/2$ is a natural number.

Usage

```
rspde.matern.precision.integer.opt(
  kappa,
  nu,
  tau,
  d,
  fem_matrices,
  graph = NULL
)
```

Arguments

kappa	Range parameter of the covariance function.
nu	Shape parameter of the covariance function.
tau	Scale parameter of the covariance function.
d	The dimension of the domain
fem_matrices	A list containing the FEM-related matrices. The list should contain elements C, G, G_2, G_3, etc.
graph	The sparsity graph of the matrices. If NULL, only a vector of the elements will be returned, if non-NULL, a sparse matrix will be returned.

Value

The precision matrix

---

rspde.matern.precision.opt

Optimized precision matrix of the covariance-based rational approximation

---

Description

`rspde.matern.precision` is used for computing the optimized version of the precision matrix of the covariance-based rational SPDE approximation of a stationary Gaussian random fields on R^d with a Matern covariance function

$$C(h) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\kappa h)^\nu K_\nu(\kappa h).$$

Usage

```
rspde.matern.precision.opt(
  kappa,
  nu,
  tau,
  rspde.order,
```

```
  dim,
  fem_matrices,
  graph = NULL,
  sharp,
  type_rational_approx
)
```

Arguments

kappa	Range parameter of the covariance function.
nu	Shape parameter of the covariance function.
tau	Scale parameter of the covariance function.
rspde.order	The order of the rational approximation
dim	The dimension of the domain
fem_matrices	A list containing the FEM-related matrices. The list should contain elements C, G, G_2, G_3, etc.
graph	The sparsity graph of the matrices. If NULL, only a vector of the elements will be returned, if non-NULL, a sparse matrix will be returned.
sharp	The sparsity graph should have the correct sparsity (costs more to perform a sparsity analysis) or an upper bound for the sparsity?
type_rational_approx	Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

Value

The precision matrix

---

rspde.matern1d *Matern rSPDE model object for INLA*

Description

Creates an INLA object for a stationary Matern model with general smoothness parameter.

Usage

```
rspde.matern1d(
  loc,
  nu.upper.bound = NULL,
  rspde.order = 1,
  nu = NULL,
  parameterization = c("spde", "matern", "matern2"),
  start.nu = NULL,
  start.theta = NULL,
```

```
  prior.nu = NULL,
  theta.prior.mean = NULL,
  theta.prior.prec = 0.1,
  prior.std.dev.nominal = 1,
  prior.range.nominal = NULL,
  prior.kappa.mean = NULL,
  prior.tau.mean = NULL,
  start.lstd.dev = NULL,
  start.lrange = NULL,
  start.ltau = NULL,
  start.lkappa = NULL,
  prior.theta.param = c("theta", "spde"),
  prior.nu.dist = c("beta", "lognormal"),
  nu.prec.inc = 1,
  type.rational.approx = c("chebfun", "brasil", "chebfunLB"),
  debug = FALSE,
  shared_lib = "detect",
  ...
)
```

Arguments

`loc` A vector of spatial locations.

`nu.upper.bound` Upper bound for the smoothness parameter. If `NULL`, it will be set to 2.

`rspde.order` The order of the covariance-based rational SPDE approach. The default order is 1.

`nu` If nu is set to a parameter, nu will be kept fixed and will not be estimated. If nu is `NULL`, it will be estimated.

`parameterization`
Which parameterization to use? `matern` uses range, std. deviation and nu (smoothness). `spde` uses kappa, tau and nu (smoothness). `matern2` uses range-like (1/kappa), variance and nu (smoothness). The default is `spde`.

`start.nu` Starting value for nu.

`start.theta` Starting values for the model parameters. In the stationary case, if `parameterization='matern'`, then `theta[1]` is the std.dev and `theta[2]` is the range parameter. If `parameterization = 'spde'`, then `theta[1]` is `tau` and `theta[2]` is `kappa`.

`prior.nu` a list containing the elements `mean` and `prec` for beta distribution, or `loglocation` and `logscale` for a truncated lognormal distribution. `loglocation` stands for the location parameter of the truncated lognormal distribution in the log scale. `prec` stands for the precision of a beta distribution. `logscale` stands for the scale of the truncated lognormal distribution on the log scale. Check details below.

`theta.prior.mean`
A vector for the mean priors of `theta`.

`theta.prior.prec`
A precision matrix for the prior of `theta`.

prior.std.dev.nominal
: Prior std. deviation to be used for the priors and for the starting values.

prior.range.nominal
: Prior range to be used for the priors and for the starting values.

prior.kappa.mean
: Prior kappa to be used for the priors and for the starting values.

prior.tau.mean
: Prior tau to be used for the priors and for the starting values.

start.lstd.dev
: Starting value for log of std. deviation. Will not be used if start.ltau is non-null. Will be only used in the stationary case and if `parameterization = 'matern'`.

start.lrange
: Starting value for log of range. Will not be used if start.lkappa is non-null. Will be only used in the stationary case and if `parameterization = 'matern'`.

start.ltau
: Starting value for log of tau. Will be only used in the stationary case and if `parameterization = 'spde'`.

start.lkappa
: Starting value for log of kappa. Will be only used in the stationary case and if `parameterization = 'spde'`.

prior.theta.param
: Should the lognormal prior be on `theta` or on the SPDE parameters (`tau` and `kappa` on the stationary case)?

prior.nu.dist
: The distribution of the smoothness parameter. The current options are "beta" or "lognormal". The default is "lognormal".

nu.prec.inc
: Amount to increase the precision in the beta prior distribution.

type.rational.approx
: Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

debug
: INLA debug argument

shared_lib
: Which shared lib to use for the cgeneric implementation? If "detect", it will check if the shared lib exists locally, in which case it will use it. Otherwise it will use INLA's shared library. If "INLA", it will use the shared lib from INLA's installation. If 'rSPDE', then it will use the local installation (does not work if your installation is from CRAN). Otherwise, you can directly supply the path of the .so (or .dll) file.

...
: Only being used internally.

prior.kappa
: a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale.

prior.tau
: a list containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale.

prior.range
: a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale. Will not be used if prior.kappa is non-null.

prior.std.dev
: a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale. Will not be used if prior.tau is non-null.

Value

An INLA model.

rspde.mesh.project *Calculate a lattice projection to/from an* `inla.mesh` *for rSPDE objects*

Description

Calculate a lattice projection to/from an `inla.mesh` for rSPDE objects

Usage

```
rspde.mesh.project(...)

rspde.mesh.projector(
  mesh,
  nu = NULL,
  rspde.order = 1,
  loc = NULL,
  lattice = NULL,
  xlim = NULL,
  ylim = NULL,
  dims = c(100, 100),
  projection = NULL,
  ...
)

## S3 method for class 'inla.mesh'
rspde.mesh.project(
  mesh,
  loc = NULL,
  field = NULL,
  rspde.order = 1,
  nu = NULL,
  ...
)

## S3 method for class 'rspde.mesh.projector'
rspde.mesh.project(projector, field, ...)

## S3 method for class 'inla.mesh.1d'
rspde.mesh.project(mesh, loc, field = NULL, rspde.order = 1, nu = NULL, ...)
```

Arguments

`...`	Additional parameters.
`mesh`	An `inla.mesh` or `inla.mesh.1d` object.
`nu`	The smoothness parameter. If `NULL`, it will be assumed that nu was estimated.
`rspde.order`	The order of the rational approximation.

`loc`	Projection locations. Can be a matrix or a SpatialPoints or a SpatialPoints-DataFrame object.
`lattice`	An `inla.mesh.lattice` object.
`xlim`	X-axis limits for a lattice. For R2 meshes, defaults to covering the domain.
`ylim`	Y-axis limits for a lattice. For R2 meshes, defaults to covering the domain.
`dims`	Lattice dimensions.
`projection`	One of c("default", "longlat", "longsinlat", "mollweide").
`field`	Basis function weights, one per mesh basis function, describing the function to be evaluated at the projection locations.
`projector`	A `rspde.mesh.projector` object.

Details

This function is built upon the inla.mesh.project and inla.mesh.projector functions from INLA.

Value

A list with projection information for rspde.mesh.project. For rspde.mesh.projector(mesh, ...), a rspde.mesh.projector object. For rspde.mesh.project(projector, field, ...), a field projected from the mesh onto the locations given by the projector object.

`rspde.metric_graph` *Matern rSPDE model object for metric graphs in INLA*

Description

Creates an INLA object for a stationary Matern model on a metric graph with general smoothness parameter.

Usage

```
rspde.metric_graph(
  graph_obj,
  h = NULL,
  nu.upper.bound = 2,
  rspde.order = 1,
  nu = NULL,
  debug = FALSE,
  B.sigma = matrix(c(0, 1, 0), 1, 3),
  B.range = matrix(c(0, 0, 1), 1, 3),
  parameterization = c("matern", "spde"),
  B.tau = matrix(c(0, 1, 0), 1, 3),
  B.kappa = matrix(c(0, 0, 1), 1, 3),
  start.nu = NULL,
  start.theta = NULL,
```

```
  prior.nu = NULL,
  theta.prior.mean = NULL,
  theta.prior.prec = 0.1,
  prior.std.dev.nominal = 1,
  prior.range.nominal = NULL,
  prior.kappa.mean = NULL,
  prior.tau.mean = NULL,
  start.lstd.dev = NULL,
  start.lrange = NULL,
  start.ltau = NULL,
  start.lkappa = NULL,
  prior.theta.param = c("theta", "spde"),
  prior.nu.dist = c("lognormal", "beta"),
  nu.prec.inc = 1,
  type.rational.approx = c("chebfun", "brasil", "chebfunLB"),
  shared_lib = "INLA"
)
```

Arguments

graph_obj	The graph object to build the model. Needs to be of class metric_graph. It should have a built mesh. If the mesh is not built, one will be built using h=0.01 as default.
h	The width of the mesh in case the mesh was not built.
nu.upper.bound	Upper bound for the smoothness parameter.
rspde.order	The order of the covariance-based rational SPDE approach.
nu	If nu is set to a parameter, nu will be kept fixed and will not be estimated. If nu is NULL, it will be estimated.
debug	INLA debug argument
B.sigma	Matrix with specification of log-linear model for σ. Will be used if parameterization = 'matern'.
B.range	Matrix with specification of log-linear model for ρ, which is a range-like parameter (it is exactly the range parameter in the stationary case). Will be used if parameterization = 'matern'.
parameterization	Which parameterization to use? matern uses range, std. deviation and nu (smoothness). spde uses kappa, tau and nu (smoothness). The default is matern.
B.tau	Matrix with specification of log-linear model for τ. Will be used if parameterization = 'spde'.
B.kappa	Matrix with specification of log-linear model for κ. Will be used if parameterization = 'spde'.
start.nu	Starting value for nu.
start.theta	Starting values for the model parameters. In the stationary case, if parameterization='matern', then theta[1] is the std.dev and theta[2] is the range parameter. If parameterization = 'spde', then theta[1] is tau and theta[2] is kappa.

- `prior.nu` a list containing the elements `mean` and `prec` for beta distribution, or `loglocation` and `logscale` for a truncated lognormal distribution. `loglocation` stands for the location parameter of the truncated lognormal distribution in the log scale. `prec` stands for the precision of a beta distribution. `logscale` stands for the scale of the truncated lognormal distribution on the log scale. Check details below.
- `theta.prior.mean` A vector for the mean priors of `theta`.
- `theta.prior.prec` A precision matrix for the prior of `theta`.
- `prior.std.dev.nominal` Prior std. deviation to be used for the priors and for the starting values.
- `prior.range.nominal` Prior range to be used for the priors and for the starting values.
- `prior.kappa.mean` Prior kappa to be used for the priors and for the starting values.
- `prior.tau.mean` Prior tau to be used for the priors and for the starting values.
- `start.lstd.dev` Starting value for log of std. deviation. Will not be used if start.ltau is non-null. Will be only used in the stationary case and if `parameterization = 'matern'`.
- `start.lrange` Starting value for log of range. Will not be used if start.lkappa is non-null. Will be only used in the stationary case and if `parameterization = 'matern'`.
- `start.ltau` Starting value for log of tau. Will be only used in the stationary case and if `parameterization = 'spde'`.
- `start.lkappa` Starting value for log of kappa. Will be only used in the stationary case and if `parameterization = 'spde'`.
- `prior.theta.param` Should the lognormal prior be on `theta` or on the SPDE parameters (`tau` and `kappa` on the stationary case)?
- `prior.nu.dist` The distribution of the smoothness parameter. The current options are "beta" or "lognormal". The default is "beta".
- `nu.prec.inc` Amount to increase the precision in the beta prior distribution. Check details below.
- `type.rational.approx` Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".
- `shared_lib` Which shared lib to use for the cgeneric implementation? If "INLA", it will use the shared lib from INLA's installation. If 'rSPDE', then it will use the local installation (does not work if your installation is from CRAN). Otherwise, you can directly supply the path of the .so (or .dll) file.
- `prior.kappa` a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale.
- `prior.tau` a list containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale.
- `prior.range` a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale. Will not be used if prior.kappa is non-null.

prior.std.dev a `list` containing the elements `meanlog` and `sdlog`, that is, the mean and standard deviation on the log scale. Will not be used if prior.tau is non-null.

Value

An INLA model.

`rspde.result` *rSPDE result extraction from INLA estimation results*

Description

Extract field and parameter values and distributions for an rspde effect from an inla result object.

Usage

```
rspde.result(
  inla,
  name,
  rspde,
  compute.summary = TRUE,
  parameterization = "detect",
  n_samples = 5000,
  n_density = 1024
)
```

Arguments

`inla`	An `inla` object obtained from a call to `inla()`.
`name`	A character string with the name of the rSPDE effect in the inla formula.
`rspde`	The `inla_rspde` object used for the effect in the inla formula.
`compute.summary`	Should the summary be computed?
`parameterization`	If 'detect', the parameterization from the model will be used. Otherwise, the options are 'spde', 'matern' and 'matern2'.
`n_samples`	The number of samples to be used if parameterization is different from the one used to fit the model.
`n_density`	The number of equally spaced points to estimate the density.

Value

If the model was fitted with `matern` parameterization (the default), it returns a list containing:

`marginals.range`
: Marginal densities for the range parameter

`marginals.log.range`
: Marginal densities for log(range)

`marginals.std.dev`
: Marginal densities for std. deviation

`marginals.log.std.dev`
: Marginal densities for log(std. deviation)

`marginals.values`
: Marginal densities for the field values

`summary.log.range`
: Summary statistics for log(range)

`summary.log.std.dev`
: Summary statistics for log(std. deviation)

`summary.values`
: Summary statistics for the field values

If `compute.summary` is `TRUE`, then the list will also contain

`summary.kappa`
: Summary statistics for kappa

`summary.tau`
: Summary statistics for tau

If the model was fitted with the `spde` parameterization, it returns a list containing:

`marginals.kappa`
: Marginal densities for kappa

`marginals.log.kappa`
: Marginal densities for log(kappa)

`marginals.log.tau`
: Marginal densities for log(tau)

`marginals.tau`
: Marginal densities for tau

`marginals.values`
: Marginal densities for the field values

`summary.log.kappa`
: Summary statistics for log(kappa)

`summary.log.tau`
: Summary statistics for log(tau)

`summary.values`
: Summary statistics for the field values

If `compute.summary` is `TRUE`, then the list will also contain

`summary.kappa`
: Summary statistics for kappa

`summary.tau`
: Summary statistics for tau

For both cases, if nu was estimated, then the list will also contain marginals.nu Marginal densities for nu

If nu was estimated and a beta prior was used, then the list will also contain

marginals.logit.nu
Marginal densities for logit(nu)

summary.logit.nu
Marginal densities for logit(nu)

If nu was estimated and a truncated lognormal prior was used, then the list will also contain

marginals.log.nu
Marginal densities for log(nu)

summary.log.nu Marginal densities for log(nu)

If nu was estimated and compute.summary is TRUE, then the list will also contain

summary.nu Summary statistics for nu

Examples

```
 #tryCatch version
tryCatch({
if (requireNamespace("INLA", quietly = TRUE)) {
  library(INLA)

  set.seed(123)

  m <- 100
  loc_2d_mesh <- matrix(runif(m * 2), m, 2)
  mesh_2d <- inla.mesh.2d(
    loc = loc_2d_mesh,
    cutoff = 0.05,
    max.edge = c(0.1, 0.5)
  )
  sigma <- 1
  range <- 0.2
  nu <- 0.8
  kappa <- sqrt(8 * nu) / range
  op <- matern.operators(
    mesh = mesh_2d, nu = nu,
    range = range, sigma = sigma, m = 2,
    parameterization = "matern"
  )
  u <- simulate(op)
  A <- inla.spde.make.A(
    mesh = mesh_2d,
    loc = loc_2d_mesh
  )
  sigma.e <- 0.1
  y <- A %*% u + rnorm(m) * sigma.e
  Abar <- rspde.make.A(mesh = mesh_2d, loc = loc_2d_mesh)
  mesh.index <- rspde.make.index(name = "field", mesh = mesh_2d)
```

```
  st.dat <- inla.stack(
    data = list(y = as.vector(y)),
    A = Abar,
    effects = mesh.index
  )
  rspde_model <- rspde.matern(
    mesh = mesh_2d,
    nu.upper.bound = 2
  )
  f <- y ~ -1 + f(field, model = rspde_model)
  rspde_fit <- inla(f,
    data = inla.stack.data(st.dat),
    family = "gaussian",
    control.predictor =
      list(A = inla.stack.A(st.dat))
  )
  result <- rspde.result(rspde_fit, "field", rspde_model)
  summary(result)
}
#stable.tryCatch
}, error = function(e){print("Could not run the example")})
```

`rspde.spacetime` *Space-Time Random Fields via SPDE Approximation*

Description

`rspde.spacetime` computes a Finite Element Method (FEM) approximation of a Gaussian random field defined as the solution to the stochastic partial differential equation (SPDE):

$$du + \gamma(\kappa^2 + \kappa^{d/2}\rho \cdot \nabla - \Delta)^\alpha u = \sigma dW_C$$

where C is a Whittle-Matérn covariance operator with smoothness parameter β and range parameter κ. This function is designed to handle space-time random fields using either 1D spatial models or higher-dimensional FEM-based approaches.

Usage

```
rspde.spacetime(
  mesh_space = NULL,
  mesh_time = NULL,
  space_loc = NULL,
  time_loc = NULL,
  drift = TRUE,
  alpha,
  beta,
  prior.kappa = NULL,
  prior.sigma = NULL,
```

```
  prior.rho = NULL,
  prior.gamma = NULL,
  prior.precision = NULL,
  bounded_rho = TRUE,
  shared_lib = "detect",
  debug = FALSE,
  ...
)
```

Arguments

mesh_space
: Spatial mesh for the FEM approximation, or a `metric_graph` object for handling models on metric graphs.

mesh_time
: Temporal mesh for the FEM approximation.

space_loc
: A vector of spatial locations for mesh nodes in 1D spatial models. This should be provided when `mesh_space` is not specified.

time_loc
: A vector of temporal locations for mesh nodes. This should be provided when `mesh_time` is not specified.

drift
: Logical value indicating whether the drift term should be included. If `FALSE`, the drift coefficient ρ is set to zero.

alpha
: Integer smoothness parameter α.

beta
: Integer smoothness parameter β.

prior.kappa
: A list specifying the prior for the range parameter κ. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars (numeric vectors of length 1). The precision refers to the prior on $\log(\kappa)$. If `NULL`, default values will be used. The `mean` value is also used as starting value for kappa.

prior.sigma
: A list specifying the prior for the variance parameter σ. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars. The precision refers to the prior on $\log(\sigma)$. If `NULL`, default values will be used. The `mean` value is also used as starting value for sigma.

prior.rho
: A list specifying the prior for the drift coefficient ρ. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars if dimension is one, and numeric vectors of length 2 if dimension is 2. The precision applies directly to ρ without log transformation. If `NULL`, default values will be used. Will not be used if `drift = FALSE`. The `mean` value is also used as starting value for rho.

prior.gamma
: A list specifying the prior for the weight γ in the SPDE operator. This list may contain two elements: `mean` and/or `precision`, both of which must be numeric scalars. The precision refers to the prior on $\log(\gamma)$. If `NULL`, default values will be used. The `mean` value is also used as starting value for gamma.

prior.precision
: A precision matrix for $\log(\kappa), \log(\sigma), \log(\gamma), \rho$. This matrix replaces the precision element from `prior.kappa`, `prior.sigma`, `prior.gamma`, and `prior.rho` respectively. For dimension 1 `prior.precision` must be a 4x4 matrix. For dimension 2, ρ is a vector of length 2, so in this case `prior.precision` must be

	a 5x5 matrix. If NULL, a diagonal precision matrix with default values will be used.
bounded_rho	Logical. Should rho be bounded to ensure the existence, uniqueness, and well-posedness of the solution? Defaults to TRUE. Note that this bounding is not a strict condition; there may exist values of rho beyond the upper bound that still satisfy these properties. When bounded_rho = TRUE, the rspde_lme models enforce bounded rho for consistency. If the estimated value of rho approaches the upper bound too closely, we recommend refitting the model with bounded_rho = FALSE. However, this should be done with caution, as it may lead to instability in some cases, though it can also result in a better model fit. The actual bound used for rho can be accessed from the bound_rho element of the returned object.
shared_lib	String specifying which shared library to use for the Cgeneric implementation. Options are "detect", "INLA", or "rSPDE". You may also specify the direct path to a .so (or .dll) file.
debug	Logical value indicating whether to enable INLA debug mode.
...	Additional arguments passed internally for configuration purposes.

Value

An object of class inla_rspde_spacetime representing the FEM approximation of the space-time Gaussian random field.

---

rspde_lme *rSPDE linear mixed effects models*

Description

Fitting linear mixed effects model with latent Whittle-Matern models.

Usage

```
rspde_lme(
  formula,
  loc,
  loc_time = NULL,
  data,
  model = NULL,
  repl = NULL,
  which_repl = NULL,
  optim_method = "L-BFGS-B",
  use_data_from_graph = TRUE,
  starting_values_latent = NULL,
  start_sigma_e = NULL,
  start_alpha = NULL,
  alpha = NULL,
```

```
  start_nu = NULL,
  nu = NULL,
  nu_upper_bound = 4,
  rspde_order = NULL,
  parallel = FALSE,
  n_cores = parallel::detectCores() - 1,
  optim_controls = list(),
  improve_hessian = FALSE,
  hessian_args = list()
)
```

Arguments

`formula` Formula object describing the relation between the response variables and the fixed effects. If the response variable is a matrix, each column of the matrix will be treated as a replicate.

`loc` A vector with the names of the columns in `data` that contain the observation locations, or a `matrix` or a `data.frame` containing the observation locations. If the model is of class `metric_graph`, the locations must be either a `matrix` or a `data.frame` with two columns, or a character vector with the names of the two columns. The first column being the number of the edge, and the second column being the normalized position on the edge. If the model is a 2d model, `loc` must be either a `matrix` or `data.frame` with two columns or a character vector with the name of the two columns that contain the location, the first entry corresponding to the `x` entry and the second corresponding to the `y` entry.

`loc_time` For spatio-temporal models, the name of the column in `data` that is the time variable, or a `matrix` or `vector` containing the observation time points.

`data` A `data.frame` containing the data to be used.

`model` Object generated by `matern.operators()`, `spde.matern.operators()` or `spacetime.operators()`. If `NULL`, simple linear regression will be performed.

`repl` Vector indicating the replicate of each observation. If `NULL` it will assume there is only one replicate. If the model is generated from graphs from `metric_graph` class and `use_data_from_graph` is `TRUE`, `repl` needs to be the name of the column inside the metric graph data that contains the replicate. If `NULL` it will assume there is only one replicate.

`which_repl` Which replicates to use? If `NULL` all replicates will be used.

`optim_method` The method to be used with `optim` function.

`use_data_from_graph`
Logical. Only for models generated from graphs from `metric_graph` class. In this case, should the data, the locations and the replicates be obtained from the graph object?

`starting_values_latent`
A vector containing the starting values for the latent model. If the latent model is generated by `matern.operators()`, then the vector should be on the form c(tau,kappa). If the model is generated by `spde.matern.operators()`, the vector should contain the nonstationary parameters. If the model is generated by `spacetime.operators()`, the vector should be on the form c(kappa,sigma,gamma,rho).

start_sigma_e	Starting value for the standard deviation of the measurement error.
start_alpha	Starting value for the smoothness parameter of spatial models. Will be used if `start_nu` is not given. Not used for spatio-temporal models.
alpha	If `NULL`, the smoothness parameter will be estimated for spatial models, otherwise it is kept fixed at the provided value. Will be used if `nu` is not given. Not used for spatio-temporal models. returned as component of the returned value.
start_nu	Starting value for the smoothness parameter of spatial models. Not used for spatio-temporal models.
nu	If `NULL`, the smoothness parameter will be estimated for spatial models, otherwise the smoothness parameter will be kept fixed at the provided value. Not used for spatio-temporal models.
nu_upper_bound	A parameter that limits the maximum value that nu can assume. Not used for spatio-temporal models.
rspde_order	The order of the rational approximation to be used while fitting the model. If not given, the order from the model object will be used. Not used for spatio-temporal models.
parallel	logical. Indicating whether to use optimParallel or not.
n_cores	Number of cores to be used if parallel is true.
optim_controls	Additional controls to be passed to `optim` or `optimParallel`.
improve_hessian	Should a more precise estimate of the hessian be obtained? Turning on might increase the overall time.
hessian_args	List of controls to be used if `improve_hessian` is `TRUE`. The list can contain the arguments to be passed to the `method.args` argument in the `numDeriv::hessian` function. See the help of the `hessian` function in `numDeriv` package for details. Observe that it only accepts the "Richardson" method for now, the method "complex" is not supported.

Value

A list containing the fitted model.

`simulate.CBrSPDEobj` *Simulation of a fractional SPDE using the covariance-based rational SPDE approximation*

Description

The function samples a Gaussian random field based using the covariance-based rational SPDE approximation.

Usage

```
## S3 method for class 'CBrSPDEobj'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  nu = NULL,
  kappa = NULL,
  sigma = NULL,
  range = NULL,
  tau = NULL,
  theta = NULL,
  m = NULL,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `matern.operators()`
`nsim`	The number of simulations.
`seed`	An object specifying if and how the random number generator should be initialized ('seeded').
`nu`	If non-null, update the shape parameter of the covariance function.
`kappa`	If non-null, update the range parameter of the covariance function.
`sigma`	If non-null, update the standard deviation of the covariance function.
`range`	If non-null, update the range parameter of the covariance function.
`tau`	If non-null, update the parameter tau.
`theta`	For non-stationary models. If non-null, update the vector of parameters.
`m`	If non-null, update the order of the rational approximation, which needs to be a positive integer.
`...`	Currently not used.

Value

A matrix with the `n` samples as columns.

Examples

```
# Sample a Gaussian Matern process on R using a rational approximation
kappa <- 10
sigma <- 1
nu <- 0.8
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
```

```
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
tau <- sqrt(gamma(nu) / (sigma^2 * kappa^(2 * nu) *
  (4 * pi)^(1 / 2) * gamma(nu + 1 / 2)))
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
  parameterization = "matern"
)

# Sample the model and plot the result
Y <- simulate(op_cov)
plot(x, Y, type = "l", ylab = "u(x)", xlab = "x")
```

---

simulate.CBrSPDEobj2d *Simulation of a fractional SPDE using the covariance-based rational SPDE approximation*

Description

The function samples a Gaussian random field based using the covariance-based rational SPDE approximation.

Usage

```
## S3 method for class 'CBrSPDEobj2d'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  nu = NULL,
  hx = NULL,
  hy = NULL,
  hxy = NULL,
  sigma = NULL,
  m = NULL,
  ...
)
```

Arguments

object	The covariance-based rational SPDE approximation, computed using `matern2d.operators()`
nsim	The number of simulations.
seed	An object specifying if and how the random number generator should be initialized (‘seeded’).
nu	If non-null, update the shape parameter of the covariance function.

hx	If non-null, update the hx parameter.
hy	If non-null, update the hy parameter.
hxy	If non-null, update the hxy parameter.
sigma	If non-null, update the standard deviation of the covariance function.
m	If non-null, update the order of the rational approximation, which needs to be a positive integer.
...	Currently not used.

Value

A matrix with the n samples as columns.

Examples

```
library(fmesher)
n_loc <- 2000
loc_2d_mesh <- matrix(runif(n_loc * 2), n_loc, 2)
mesh_2d <- fm_mesh_2d(loc = loc_2d_mesh, cutoff = 0.03, max.edge = c(0.1, 0.5))
op <- matern2d.operators(mesh = mesh_2d, sigma = 1, nu = 1, hx = 0.1, hy = 0.1, hxy = 0)
u <- simulate(op)
```

---

simulate.intrinsicCBrSPDEobj

Simulation of a fractional intrinsic SPDE using the covariance-based rational SPDE approximation

---

Description

The function samples a Gaussian random field based using the covariance-based rational SPDE approximation.

Usage

```
## S3 method for class 'intrinsicCBrSPDEobj'
simulate(object, nsim = 1, seed = NULL, integral.constraint = TRUE, ...)
```

Arguments

object	The covariance-based rational SPDE approximation, computed using intrinsic.matern.operators()
nsim	The number of simulations.
seed	An object specifying if and how the random number generator should be initialized ('seeded').
integral.constraint	Should the contraint on the integral be done?
...	Currently not used.

Value

A matrix with the `nsim` samples as columns.

`simulate.rSPDEobj` *Simulation of a fractional SPDE using a rational SPDE approximation*

Description

The function samples a Gaussian random field based on a pre-computed rational SPDE approximation.

Usage

```
## S3 method for class 'rSPDEobj'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`object`	The rational SPDE approximation, computed using `fractional.operators()`, `matern.operators()`, or `spde.matern.operators()`.
`nsim`	The number of simulations.
`seed`	an object specifying if and how the random number generator should be initialized ('seeded').
`...`	Currently not used.

Value

A matrix with the `n` samples as columns.

See Also

`simulate.CBrSPDEobj()`

Examples

```
# Sample a Gaussian Matern process on R using a rational approximation
kappa <- 10
sigma <- 1
nu <- 0.8
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation
op <- matern.operators(
```

```
  range = range, sigma = sigma,
  nu = nu, loc_mesh = x, d = 1,
  parameterization = "matern"
)

# Sample the model and plot the result
Y <- simulate(op)
plot(x, Y, type = "l", ylab = "u(x)", xlab = "x")
```

`simulate.rSPDEobj1d` *Simulation of a Matern field using a rational SPDE approximation*

Description

The function samples a Gaussian random field based on a pre-computed rational SPDE approximation.

Usage

```
## S3 method for class 'rSPDEobj1d'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`object`	The rational SPDE approximation, computed using `matern.rational()`.
`nsim`	The number of simulations.
`seed`	an object specifying if and how the random number generator should be initialized ('seeded').
`...`	Currently not used.

Value

A matrix with the `n` samples as columns.

See Also

`matern.rational()`

Examples

```
# Sample a Gaussian Matern process on R using a rational approximation
range <- 0.2
sigma <- 1
nu <- 0.8

# compute rational approximation
x <- seq(from = 0, to = 1, length.out = 100)
```

```
op <- matern.rational(
  range = range, sigma = sigma,
  nu = nu, loc = x
)

# Sample the model and plot the result
Y <- simulate(op)
plot(x, Y, type = "l", ylab = "u(x)", xlab = "x")
```

---

simulate.spacetimeobj *Simulation of space-time models*

Description

Simulation of space-time models

Usage

```
## S3 method for class 'spacetimeobj'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  kappa = NULL,
  sigma = NULL,
  gamma = NULL,
  rho = NULL,
  ...
)
```

Arguments

object	Space-time object created by spacetime.operators()
nsim	The number of simulations.
seed	an object specifying if and how the random number generator should be initialized (‘seeded’).
kappa	kappa parameter if it should be updated
sigma	sigma parameter if it should be updated
gamma	gamma parameter if it should be updated
rho	rho parameter if it should be updated
...	Currently not used.

Value

A matrix with the simulations as columns.

Examples

```
s <- seq(from = 0, to = 20, length.out = 101)
t <- seq(from = 0, to = 20, length.out = 31)

op_cov <- spacetime.operators(space_loc = s, time_loc = t,
                              kappa = 5, sigma = 10, alpha = 1,
                              beta = 2, rho = 1, gamma = 0.05)
x <- simulate(op_cov, nsim = 1)
image(matrix(x, nrow = length(s), ncol = length(t)))
```

`spacetime.operators` *Space-time random fields*

Description

`spacetime.operators` is used for computing a FEM approximation of a Gaussian random field defined as a solution to the SPDE

$$du + \gamma(\kappa^2 + \kappa^{d/2}\rho \cdot \nabla - \Delta)^\alpha u = \sigma dW_C.$$

where C is a Whittle-Matern covariance operator with smoothness parameter β and range parameter κ

Usage

```
spacetime.operators(
  mesh_space = NULL,
  mesh_time = NULL,
  space_loc = NULL,
  time_loc = NULL,
  graph = NULL,
  kappa = NULL,
  sigma = NULL,
  gamma = NULL,
  rho = NULL,
  alpha = NULL,
  beta = NULL,
  bounded_rho = TRUE
)
```

Arguments

`mesh_space`	Spatial mesh for FEM approximation
`mesh_time`	Temporal mesh for FEM approximation
`space_loc`	Locations of mesh nodes for spatial mesh for 1d models.
`time_loc`	Locations of temporal mesh nodes.

graph	An optional `metric_graph` object. Replaces `mesh` for models on metric graphs.
kappa	Positive spatial range parameter
sigma	Positive variance parameter
gamma	Temporal range parameter.
rho	Drift parameter. Real number on metric graphs and one-dimensional spatial domains, a vector with two number on 2d domains.
alpha	Integer smoothness parameter alpha.
beta	Integer smoothness parameter beta.
bounded_rho	Logical. Specifies whether `rho` should be bounded to ensure the existence, uniqueness, and well-posedness of the solution. Defaults to `TRUE`. Note that this bounding is not a strict condition; there may exist values of rho beyond the upper bound that still satisfy these properties. When `bounded_rho = TRUE`, the `rspde_lme` models enforce bounded `rho` for consistency. If the estimated value of `rho` approaches the upper bound too closely, we recommend refitting the model with `bounded_rho = FALSE`. However, this should be done with caution, as it may lead to instability in some cases, though it can also result in a better model fit. The actual bound used for `rho` can be accessed from the `bound_rho` element of the returned object.

Value

An object of type spacetimeobj.

Examples

```
s <- seq(from = 0, to = 20, length.out = 101)
t <- seq(from = 0, to = 20, length.out = 31)

op_cov <- spacetime.operators(space_loc = s, time_loc = t,
                              kappa = 5, sigma = 10, alpha = 1,
                              beta = 2, rho = 1, gamma = 0.05)
Q <- op_cov$Q
v <- rep(0,dim(Q)[1])
v[1565] <- 1
Sigma <- solve(Q,v)

image(matrix(Sigma, nrow=length(s), ncol = length(t)))
```

---

spde.make.A — *Observation/prediction matrices for rSPDE models with integer smoothness.*

Description

Constructs observation/prediction weight matrices for rSPDE models with integer smoothness based on `inla.mesh` or `inla.mesh.1d` objects.

Usage

```
spde.make.A(
  mesh = NULL,
  loc = NULL,
  A = NULL,
  index = NULL,
  group = NULL,
  repl = 1L,
  n.group = NULL,
  n.repl = NULL
)
```

Arguments

`mesh`	An `inla.mesh`, an `inla.mesh.1d` object or a `metric_graph` object.
`loc`	Locations, needed if an INLA mesh is provided
`A`	The A matrix from the standard SPDE approach, such as the matrix returned by `inla.spde.make.A`. Should only be provided if `mesh` is not provided.
`index`	For each observation/prediction value, an index into loc. Default is seq_len(nrow(A.loc)).
`group`	For each observation/prediction value, an index into the group model.
`repl`	For each observation/prediction value, the replicate index.
`n.group`	The size of the group model.
`n.repl`	The total number of replicates.

Value

The A matrix for rSPDE models.

Examples

```
 #tryCatch version
tryCatch({
if (requireNamespace("fmesher", quietly = TRUE)) {
  library(fmesher)

  set.seed(123)
  loc <- matrix(runif(100 * 2) * 100, 100, 2)
  mesh <- fm_mesh_2d(
    loc = loc,
    cutoff = 50,
    max.edge = c(50, 500)
  )
  A <- spde.make.A(mesh, loc = loc)
}
#stable.tryCatch
}, error = function(e){print("Could not run the example")})
```

spde.matern.loglike *Parameter-based log-likelihood for a latent Gaussian Matern SPDE model using a rational SPDE approximation*

Description

This function evaluates the log-likelihood function for observations of a Gaussian process defined as the solution to the SPDE

$$(\kappa(s) - \Delta)^{\beta}(\tau(s)u(s)) = W.$$

Usage

```
spde.matern.loglike(
  object,
  Y,
  A,
  sigma.e,
  mu = 0,
  nu = NULL,
  kappa = NULL,
  tau = NULL,
  theta = NULL,
  m = NULL
)
```

Arguments

object	The rational SPDE approximation, computed using spde.matern.operators()
Y	The observations, either a vector or a matrix where the columns correspond to independent replicates of observations.
A	An observation matrix that links the measurement location to the finite element basis.
sigma.e	IF non-null, the standard deviation of the measurement noise will be kept fixed in the returned likelihood.
mu	Expectation vector of the latent field (default = 0).
nu	If non-null, the shape parameter will be kept fixed in the returned likelihood.
kappa	If non-null, updates the range parameter.
tau	If non-null, updates the parameter tau.
theta	If non-null, updates the parameter theta (that connects tau and kappa to the model matrices in object).
m	If non-null, update the order of the rational approximation, which needs to be a positive integer.

Details

The observations are assumed to be generated as $Y_i = u(s_i) + \epsilon_i$, where ϵ_i are iid mean-zero Gaussian variables. The latent model is approximated using a rational approximation of the fractional SPDE model.

Value

The log-likelihood value.

See Also

rSPDE.loglike().

Examples

```
# this example illustrates how the function can be used for maximum
# likelihood estimation
# Sample a Gaussian Matern process on R using a rational approximation
sigma.e <- 0.1
n.rep <- 10
n.obs <- 100
n.x <- 51
# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = n.x)
fem <- rSPDE.fem1d(x)
tau <- rep(0.5, n.x)
nu <- 0.8
alpha <- nu + 1 / 2
kappa <- rep(1, n.x)
# compute rational approximation
op <- spde.matern.operators(
  kappa = kappa, tau = tau, alpha = alpha,
  parameterization = "spde", d = 1,
  loc_mesh = x
)
# Sample the model
u <- simulate(op, n.rep)
# Create some data
obs.loc <- runif(n = n.obs, min = 0, max = 1)
A <- rSPDE.A1d(x, obs.loc)
noise <- rnorm(n.obs * n.rep)
dim(noise) <- c(n.obs, n.rep)
Y <- as.matrix(A %*% u + sigma.e * noise)
# define negative likelihood function for optimization using matern.loglike
mlik <- function(theta) {
  return(-spde.matern.loglike(op, Y, A,
    sigma.e = exp(theta[4]),
    nu = exp(theta[3]),
    kappa = exp(theta[2]),
    tau = exp(theta[1])
  ))
}
```

```
#' #The parameters can now be estimated by minimizing mlik with optim

# Choose some reasonable starting values depending on the size of the domain
theta0 <- log(c(1 / sqrt(var(c(Y))), sqrt(8), 0.9, 0.01))
# run estimation and display the results
theta <- optim(theta0, mlik)
print(data.frame(
  tau = c(tau[1], exp(theta$par[1])), kappa = c(kappa[1], exp(theta$par[2])),
  nu = c(nu, exp(theta$par[3])), sigma.e = c(sigma.e, exp(theta$par[4])),
  row.names = c("Truth", "Estimates")
))
```

`spde.matern.operators` *Rational approximations of non-stationary Gaussian SPDE Matern random fields*

Description

`spde.matern.operators` is used for computing a rational SPDE approximation of a Gaussian random fields on R^d defined as a solution to the SPDE

$$(\kappa(s) - \Delta)^\beta (\tau(s) u(s)) = W.$$

Usage

```
spde.matern.operators(
  kappa = NULL,
  tau = NULL,
  theta = NULL,
  B.tau = matrix(c(0, 1, 0), 1, 3),
  B.kappa = matrix(c(0, 0, 1), 1, 3),
  B.sigma = matrix(c(0, 1, 0), 1, 3),
  B.range = matrix(c(0, 0, 1), 1, 3),
  alpha = NULL,
  nu = NULL,
  parameterization = c("spde", "matern"),
  G = NULL,
  C = NULL,
  d = NULL,
  graph = NULL,
  mesh = NULL,
  range_mesh = NULL,
  loc_mesh = NULL,
  m = 1,
  type = c("covariance", "operator"),
  type_rational_approximation = c("chebfun", "brasil", "chebfunLB")
)
```

Arguments

- `kappa` Vector with the, possibly spatially varying, range parameter evaluated at the locations of the mesh used for the finite element discretization of the SPDE.
- `tau` Vector with the, possibly spatially varying, precision parameter evaluated at the locations of the mesh used for the finite element discretization of the SPDE.
- `theta` Theta parameter that connects B.tau and B.kappa to tau and kappa through a log-linear regression, in case the parameterization is `spde`, and that connects B.sigma and B.range to tau and kappa in case the parameterization is `matern`.
- `B.tau` Matrix with specification of log-linear model for τ. Will be used if `parameterization = 'spde'`.
- `B.kappa` Matrix with specification of log-linear model for κ. Will be used if `parameterization = 'spde'`.
- `B.sigma` Matrix with specification of log-linear model for σ. Will be used if `parameterization = 'matern'`.
- `B.range` Matrix with specification of log-linear model for ρ, which is a range-like parameter (it is exactly the range parameter in the stationary case). Will be used if `parameterization = 'matern'`.
- `alpha` smoothness parameter. Will be used if the parameterization is 'spde'.
- `nu` Shape parameter of the covariance function. Will be used if the parameterization is 'matern'.
- `parameterization` Which parameterization to use? `matern` uses range, std. deviation and nu (smoothness). `spde` uses kappa, tau and nu (smoothness). The default is `matern`.
- `G` The stiffness matrix of a finite element discretization of the domain of interest.
- `C` The mass matrix of a finite element discretization of the domain of interest.
- `d` The dimension of the domain. Does not need to be given if `mesh` is used.
- `graph` An optional `metric_graph` object. Replaces `d`, `C` and `G`.
- `mesh` An optional inla mesh. `d`, `C` and `G` must be given if `mesh` is not given.
- `range_mesh` The range of the mesh. Will be used to provide starting values for the parameters. Will be used if `mesh` and `graph` are `NULL`, and if one of the parameters (kappa or tau for spde parameterization, or sigma or range for matern parameterization) are not provided.
- `loc_mesh` The mesh locations used to construct the matrices C and G. This option should be provided if one wants to use the `rspde_lme()` function and will not provide neither graph nor mesh. Only works for 1d data. Does not work for metric graphs. For metric graphs you should supply the graph using the `graph` argument.
- `m` The order of the rational approximation, which needs to be a positive integer. The default value is 1.
- `type` The type of the rational approximation. The options are "covariance" and "operator". The default is "covariance".
- `type_rational_approximation` Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

Details

The approximation is based on a rational approximation of the fractional operator $(\kappa(s)^2 - \Delta)^\beta$, where $\beta = (\nu + d/2)/2$. This results in an approximate model on the form

$$P_l u(s) = P_r W,$$

where $P_j = p_j(L)$ are non-fractional operators defined in terms of polynomials p_j for $j = l, r$. The order of p_r is given by `m` and the order of p_l is $m + m_\beta$ where m_β is the integer part of β if $\beta > 1$ and $m_\beta = 1$ otherwise.

The discrete approximation can be written as $u = P_r x$ where $x \sim N(0, Q^{-1})$ and $Q = P_l^T C^{-1} P_l$. Note that the matrices P_r and Q may be be ill-conditioned for $m > 1$. In this case, the metehods in `operator.operations()` should be used for operations involving the matrices, since these methods are more numerically stable.

Value

`spde.matern.operators` returns an object of class "rSPDEobj. This object contains the quantities listed in the output of `fractional.operators()` as well as the smoothness parameter ν.

See Also

`fractional.operators()`, `spde.matern.operators()`, `matern.operators()`

Examples

```
# Sample non-stationary Matern field on R
tau <- 1
nu <- 0.8

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# define a non-stationary range parameter
kappa <- seq(from = 2, to = 20, length.out = length(x))
alpha <- nu + 1 / 2
# compute rational approximation
op <- spde.matern.operators(
  kappa = kappa, tau = tau, alpha = alpha,
  G = fem$G, C = fem$C, d = 1
)

# sample the field
u <- simulate(op)

# plot the sample
plot(x, u, type = "l", ylab = "u(s)", xlab = "s")
```

summary.CBrSPDEobj *Summarise CBrSPDE objects*

Description

Summary method for class "CBrSPDEobj"

Usage

```
## S3 method for class 'CBrSPDEobj'
summary(object, ...)

## S3 method for class 'summary.CBrSPDEobj'
print(x, ...)

## S3 method for class 'CBrSPDEobj'
print(x, ...)
```

Arguments

`object`	an object of class "CBrSPDEobj", usually, a result of a call to `matern.operators()`.
`...`	further arguments passed to or from other methods.
`x`	an object of class "summary.CBrSPDEobj", usually, a result of a call to `summary.CBrSPDEobj()`.

Examples

```
# Compute the covariance-based rational approximation of a
# Gaussian process with a Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
tau <- sqrt(gamma(nu) / (sigma^2 * kappa^(2 * nu) *
  (4 * pi)^(1 / 2) * gamma(nu + 1 / 2)))
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
  parameterization = "matern"
)

op_cov
```

summary.CBrSPDEobj2d *Summarise CBrSPDEobj2d objects*

Description

Summary method for class "CBrSPDEobj2d"

Usage

```
## S3 method for class 'CBrSPDEobj2d'
summary(object, ...)

## S3 method for class 'summary.CBrSPDEobj2d'
print(x, ...)

## S3 method for class 'CBrSPDEobj2d'
print(x, ...)
```

Arguments

`object`	an object of class "CBrSPDEobj2d", usually, a result of a call to `matern2d.operators()`.
`...`	further arguments passed to or from other methods.
`x`	an object of class "summary.CBrSPDEobj2d", usually, a result of a call to `summary.CBrSPDEobj2d()`.

Examples

```
library(fmesher)
n_loc <- 2000
loc_2d_mesh <- matrix(runif(n_loc * 2), n_loc, 2)
mesh_2d <- fm_mesh_2d(loc = loc_2d_mesh, cutoff = 0.03, max.edge = c(0.1, 0.5))
op <- matern2d.operators(mesh = mesh_2d)
op
```

summary.rSPDEobj *Summarise rSPDE objects*

Description

Summary method for class "rSPDEobj"

Usage

```
## S3 method for class 'rSPDEobj'
summary(object, ...)

## S3 method for class 'summary.rSPDEobj'
print(x, ...)

## S3 method for class 'rSPDEobj'
print(x, ...)
```

Arguments

`object`	an object of class "rSPDEobj", usually, a result of a call to `fractional.operators()`, `matern.operators()`, or `spde.matern.operators()`.
`...`	further arguments passed to or from other methods.
`x`	an object of class "summary.rSPDEobj", usually, a result of a call to `summary.rSPDEobj()`.

`summary.rSPDEobj1d` *Summarise rSPDE objects without FEM*

Description

Summary method for class "rSPDEobj1d"

Usage

```
## S3 method for class 'rSPDEobj1d'
summary(object, ...)

## S3 method for class 'summary.rSPDEobj1d'
print(x, ...)

## S3 method for class 'rSPDEobj1d'
print(x, ...)
```

Arguments

`object`	an object of class "rSPDEobj1d", usually, a result of a call to `matern.rational()`.
`...`	further arguments passed to or from other methods.
`x`	an object of class "summary.rSPDEobj1d", usually, a result of a call to `summary.rSPDEobj1d()`.

summary.rspde_lme — *Summary Method for* `rspde_lme` *Objects.*

Description

Function providing a summary of results related to mixed effects regression models with Whittle-Matern latent models.

Usage

```
## S3 method for class 'rspde_lme'
summary(object, all_times = FALSE, ...)
```

Arguments

`object`	an object of class "rspde_lme" containing results from the fitted model.
`all_times`	Show all computed times.
`...`	not used.

Value

An object of class `summary_rspde_lme` containing several informations of a *rspde_lme* object.

summary.rspde_result — *Summary for posteriors of field parameters for an* `inla_rspde` *model from a* `rspde_result` *object*

Description

Summary for posteriors of rSPDE field parameters in their original scales.

Usage

```
## S3 method for class 'rspde_result'
summary(object, digits = 6, ...)
```

Arguments

`object`	A `rspde_result` object.
`digits`	integer, used for number formatting with signif()
`...`	Currently not used.

Value

Returns a `data.frame` containing the summary.

Examples

```
 #tryCatch version
tryCatch({
if (requireNamespace("INLA", quietly = TRUE)) {
  library(INLA)

  set.seed(123)

  m <- 100
  loc_2d_mesh <- matrix(runif(m * 2), m, 2)
  mesh_2d <- inla.mesh.2d(
    loc = loc_2d_mesh,
    cutoff = 0.05,
    max.edge = c(0.1, 0.5)
  )
  sigma <- 1
  range <- 0.2
  nu <- 0.8
  kappa <- sqrt(8 * nu) / range
  op <- matern.operators(
    mesh = mesh_2d, nu = nu,
    range = range, sigma = sigma, m = 2,
    parameterization = "matern"
  )
  u <- simulate(op)
  A <- inla.spde.make.A(
    mesh = mesh_2d,
    loc = loc_2d_mesh
  )
  sigma.e <- 0.1
  y <- A %*% u + rnorm(m) * sigma.e
  Abar <- rspde.make.A(mesh = mesh_2d, loc = loc_2d_mesh)
  mesh.index <- rspde.make.index(name = "field", mesh = mesh_2d)
  st.dat <- inla.stack(
    data = list(y = as.vector(y)),
    A = Abar,
    effects = mesh.index
  )
  rspde_model <- rspde.matern(
    mesh = mesh_2d,
    nu.upper.bound = 2
  )
  f <- y ~ -1 + f(field, model = rspde_model)
  rspde_fit <- inla(f,
    data = inla.stack.data(st.dat),
    family = "gaussian",
    control.predictor =
      list(A = inla.stack.A(st.dat))
  )
  result <- rspde.result(rspde_fit, "field", rspde_model)
  summary(result)
}
```

```
#stable.tryCatch
}, error = function(e){print("Could not run the example")})
```

summary.spacetimeobj *Summarise spacetime objects*

Description

Summary method for class "spacetimeobj"

Usage

```
## S3 method for class 'spacetimeobj'
summary(object, ...)

## S3 method for class 'summary.spacetimeobj'
print(x, ...)

## S3 method for class 'spacetimeobj'
print(x, ...)
```

Arguments

object	an object of class "spacetimeobj", usually, a result of a call to `spacetime.operators()`.
...	further arguments passed to or from other methods.
x	an object of class "summary.spacetimeobj", usually, a result of a call to `summary.spacetimeobj()`.

transform_parameters_anisotropic *Transform Anisotropic SPDE Model Parameters to Original Scale*

Description

This function takes a vector of transformed parameters and applies the appropriate transformations to return them in the original scale for use in anisotropic SPDE models.

Usage

```
transform_parameters_anisotropic(theta, nu_upper_bound = NULL)
```

Arguments

`theta` A numeric vector of length 4 or 5, containing the transformed parameters in this order:

- **lhx** The logarithmic representation of hx.
- **lhy** The logarithmic representation of hy.
- **logit_hxy** The logit-transformed representation of hxy.
- **lsigma** The logarithmic representation of sigma.
- **lnu (optional)** The logarithmic representation of nu. If not provided, nu is not returned.

`nu_upper_bound` (optional) A numeric value representing the upper bound for the smoothness parameter nu. This is only used, and must be provided, if `lnu` is provided.

Value

A named list with the parameters in the original scale:

- **hx** The original scale for hx (exponential of lhx).
- **hy** The original scale for hy (exponential of lhy).
- **hxy** The original scale for hxy (inverse logit transformation of logit_hxy).
- **sigma** The original scale for sigma (exponential of lsigma).
- **nu (optional)** The original scale for nu (using the forward_nu transformation). Only included if `lnu` is provided.

Examples

```
# With lnu
theta <- c(log(0.1), log(0.2), log((0.3 + 1) / (1 - 0.3)), log(0.5), log(1))
nu_upper_bound <- 2
transform_parameters_anisotropic(theta, nu_upper_bound)

# Without lnu
theta <- c(log(0.1), log(0.2), log((0.3 + 1) / (1 - 0.3)), log(0.5))
transform_parameters_anisotropic(theta)
```

`transform_parameters_spacetime`
Transform Spacetime SPDE Model Parameters to Original Scale

Description

This function takes a vector of transformed parameters and applies the appropriate transformations to return them in the original scale for use in spacetime SPDE models.

Usage

```
transform_parameters_spacetime(theta, st_model)
```

Arguments

`theta` A numeric vector containing the transformed parameters in this order:

- **lkappa** The logarithmic representation of kappa.
- **lsigma** The logarithmic representation of sigma.
- **lgamma** The logarithmic representation of gamma.
- **logit_rho (optional)** The logit-transformed representation of rho, if drift = 1.
- **logit_rho2 (optional)** The logit-transformed representation of rho2, if drift = 1 and d = 2.

`st_model` A list containing the spacetime model parameters:

- **d** The dimension (e.g., 1 or 2).
- **bound** The bound for rho and rho2.
- **is_bounded** A logical value indicating if rho and rho2 are bounded.
- **drift** A logical value indicating if drift is included in the model.

Value

A named list with the parameters in the original scale:

- **kappa** The original scale for kappa (exponential of lkappa).
- **sigma** The original scale for sigma (exponential of lsigma).
- **gamma** The original scale for gamma (exponential of lgamma).
- **rho (optional)** The original scale for rho.
- **rho2 (optional)** The original scale for rho2, if d = 2.

---

update.CBrSPDEobj *Update parameters of CBrSPDEobj objects*

Description

Function to change the parameters of a CBrSPDEobj object

Usage

```
## S3 method for class 'CBrSPDEobj'
update(
  object,
  nu = NULL,
  alpha = NULL,
  kappa = NULL,
  tau = NULL,
  sigma = NULL,
  range = NULL,
  theta = NULL,
  m = NULL,
```

```
  mesh = NULL,
  loc_mesh = NULL,
  graph = NULL,
  range_mesh = NULL,
  compute_higher_order = object$higher_order,
  parameterization = NULL,
  type_rational_approximation = object$type_rational_approximation,
  return_block_list = object$return_block_list,
  ...
)
```

Arguments

- `object` The covariance-based rational SPDE approximation, computed using `matern.operators()`
- `nu` If non-null, update the shape parameter of the covariance function. Will be used if parameterization is 'matern'.
- `alpha` If non-null, update the fractional SPDE order parameter. Will be used if parameterization is 'spde'.
- `kappa` If non-null, update the parameter kappa of the SPDE. Will be used if parameterization is 'spde'.
- `tau` If non-null, update the parameter tau of the SPDE. Will be used if parameterization is 'spde'.
- `sigma` If non-null, update the standard deviation of the covariance function. Will be used if parameterization is 'matern'.
- `range` If non-null, update the range parameter of the covariance function. Will be used if parameterization is 'matern'.
- `theta` For non-stationary models. If non-null, update the vector of parameters.
- `m` If non-null, update the order of the rational approximation, which needs to be a positive integer.
- `mesh` An optional inla mesh. Replaces `d`, `C` and `G`.
- `loc_mesh` The mesh locations used to construct the matrices C and G. This option should be provided if one wants to use the `rspde_lme()` function and will not provide neither graph nor mesh. Only works for 1d data. Does not work for metric graphs. For metric graphs you should supply the graph using the `graph` argument.
- `graph` An optional `metric_graph` object. Replaces `d`, `C` and `G`.
- `range_mesh` The range of the mesh. Will be used to provide starting values for the parameters. Will be used if `mesh` and `graph` are `NULL`, and if one of the parameters (kappa or tau for spde parameterization, or sigma or range for matern parameterization) are not provided.
- `compute_higher_order` Logical. Should the higher order finite element matrices be computed?
- `parameterization` If non-null, update the parameterization. Only works for stationary models.

`type_rational_approximation`
: Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

`return_block_list`
: Logical. For `type = "covariance"`, should the block parts of the precision matrix be returned separately as a list?

`...`
: Currently not used.

Value

It returns an object of class "CBrSPDEobj. This object contains the same quantities listed in the output of `matern.operators()`.

See Also

`simulate.CBrSPDEobj()`, `matern.operators()`

Examples

```
# Compute the covariance-based rational approximation of a
# Gaussian process with a Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
op_cov <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2,
  parameterization = "matern"
)
op_cov

# Update the range parameter of the model:
op_cov <- update(op_cov, kappa = 20)
op_cov
```

---

update.CBrSPDEobj2d *Update parameters of CBrSPDEobj2d objects*

Description

Function to change the parameters of a CBrSPDEobj object

Usage

```
## S3 method for class 'CBrSPDEobj2d'
update(
  object,
  hx = NULL,
  hy = NULL,
  hxy = NULL,
  sigma = NULL,
  nu = NULL,
  m = NULL,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `matern2d.operators()`
`hx`	If non-null, update the hx parameter.
`hy`	If non-null, update the hy parameter.
`hxy`	If non-null, update the hxy parameter.
`sigma`	If non-null, update the standard deviation of the covariance function.
`nu`	If non-null, update the shape parameter of the covariance function. Will be used if parameterization is 'matern'.
`m`	If non-null, update the order of the rational approximation, which needs to be a positive integer.
`...`	Currently not used.

Value

It returns an object of class "CBrSPDEobj2d.

See Also

`simulate.CBrSPDEobj2d()`, `matern2d.operators()`

Examples

```
library(fmesher)
n_loc <- 2000
loc_2d_mesh <- matrix(runif(n_loc * 2), n_loc, 2)
mesh_2d <- fm_mesh_2d(loc = loc_2d_mesh, cutoff = 0.03, max.edge = c(0.1, 0.5))
op <- matern2d.operators(mesh = mesh_2d)
op <- update(op, nu = 0.5)
```

`update.rSPDEobj` *Update parameters of rSPDEobj objects*

Description

Function to change the parameters of a rSPDEobj object

Usage

```
## S3 method for class 'rSPDEobj'
update(
  object,
  nu = NULL,
  alpha = NULL,
  kappa = NULL,
  sigma = NULL,
  range = NULL,
  tau = NULL,
  theta = NULL,
  m = NULL,
  mesh = NULL,
  loc_mesh = NULL,
  graph = NULL,
  range_mesh = NULL,
  parameterization = NULL,
  ...
)
```

Arguments

`object`	The operator-based rational SPDE approximation, computed using `matern.operators()` with `type="operator"`
`nu`	If non-null, update the shape parameter of the covariance function.
`alpha`	If non-null, update the fractional order.
`kappa`	If non-null, update the range parameter of the covariance function.
`sigma`	If non-null, update the standard deviation of the covariance function.
`range`	If non-null, update the range parameter of the covariance function.
`tau`	If non-null, update the parameter tau.
`theta`	If non-null, update the parameter theta, that connects tau and kappa to the model matrices.
`m`	If non-null, update the order of the rational approximation, which needs to be a positive integer.
`mesh`	An optional inla mesh. Replaces `d`, `C` and `G`.

`loc_mesh` The mesh locations used to construct the matrices C and G. This option should be provided if one wants to use the `rspde_lme()` function and will not provide neither graph nor mesh. Only works for 1d data. Does not work for metric graphs. For metric graphs you should supply the graph using the `graph` argument.

`graph` An optional `metric_graph` object. Replaces `d`, `C` and `G`.

`range_mesh` The range of the mesh. Will be used to provide starting values for the parameters. Will be used if `mesh` and `graph` are `NULL`, and if one of the parameters (kappa or tau for spde parameterization, or sigma or range for matern parameterization) are not provided.

`parameterization`
If non-null, update the parameterization. Only works for stationary models.

`...` Currently not used.

Value

It returns an object of class "rSPDEobj. This object contains the same quantities listed in the output of `matern.operators()`.

See Also

`simulate.rSPDEobj()`, `matern.operators()`

Examples

```
# Compute the operator-based rational approximation of a
# Gaussian process with a Matern covariance function on R
kappa <- 10
sigma <- 1
nu <- 0.8
range <- sqrt(8 * nu) / kappa

# create mass and stiffness matrices for a FEM discretization
x <- seq(from = 0, to = 1, length.out = 101)
fem <- rSPDE.fem1d(x)

# compute rational approximation of covariance function at 0.5
op <- matern.operators(
  loc_mesh = x, nu = nu,
  range = range, sigma = sigma, d = 1, m = 2, type = "operator",
  parameterization = "matern"
)
op

# Update the range parameter of the model:
op <- update(op, kappa = 20)
op
```

update.rSPDEobj1d *Update parameters of rSPDEobj1d objects*

Description

Function to change the parameters of a rSPDEobj1d object

Usage

```
## S3 method for class 'rSPDEobj1d'
update(
  object,
  nu = NULL,
  alpha = NULL,
  kappa = NULL,
  tau = NULL,
  sigma = NULL,
  range = NULL,
  theta = NULL,
  m = NULL,
  loc = NULL,
  graph = NULL,
  parameterization = NULL,
  type_rational_approximation = object$type_rational_approximation,
  ...
)
```

Arguments

`object`	The covariance-based rational SPDE approximation, computed using `matern.rational()`
`nu`	If non-null, update the shape parameter of the covariance function. Will be used if parameterization is 'matern'.
`alpha`	If non-null, update the fractional SPDE order parameter. Will be used if parameterization is 'spde'.
`kappa`	If non-null, update the parameter kappa of the SPDE. Will be used if parameterization is 'spde'.
`tau`	If non-null, update the parameter tau of the SPDE. Will be used if parameterization is 'spde'.
`sigma`	If non-null, update the standard deviation of the covariance function. Will be used if parameterization is 'matern'.
`range`	If non-null, update the range parameter of the covariance function. Will be used if parameterization is 'matern'.
`theta`	For non-stationary models. If non-null, update the vector of parameters.
`m`	If non-null, update the order of the rational approximation, which needs to be a positive integer.

loc The locations of interest for evaluating the model.

graph An optional metric_graph object.

parameterization If non-null, update the parameterization.

type_rational_approximation Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".

... Currently not used.

Value

It returns an object of class "rSPDEobj1d". This object contains the same quantities listed in the output of matern.rational().

See Also

simulate.rSPDEobj1d(), matern.rational()

Examples

```
s <- seq(from = 0, to = 1, length.out = 101)
kappa <- 20
sigma <- 2
nu <- 0.8
r <- sqrt(8*nu)/kappa #range parameter
op_cov <- matern.rational(loc = s, nu = nu, range = r, sigma = sigma, m = 2,
parameterization = "matern")
cov1 <- op_cov$covariance(ind = 1)
op_cov <- update(op_cov, range = 0.2)
cov2 <- op_cov$covariance(ind = 1)
plot(s, cov1, type = "l")
lines(s, cov2, col = 2)
```

---

update.spacetimeobj *Update parameters of spacetimeobj objects*

Description

Function to change the parameters of a spacetimeobj object

Usage

```
## S3 method for class 'spacetimeobj'
update(object, kappa = NULL, sigma = NULL, gamma = NULL, rho = NULL, ...)
```

Arguments

`object`	Space-time object created by `spacetime.operators()`
`kappa`	kappa value to be updated.
`sigma`	sigma value to be updated.
`gamma`	gamma value to be updated.
`rho`	rho value to be updated.
`...`	currently not used.

Value

An object of type spacetimeobj with updated parameters.

Examples

```
s <- seq(from = 0, to = 20, length.out = 101)
t <- seq(from = 0, to = 20, length.out = 31)

op_cov <- spacetime.operators(space_loc = s, time_loc = t,
                              kappa = 5, sigma = 10, alpha = 1,
                              beta = 2, rho = 1, gamma = 0.05)
op_cov <- update(op_cov, kappa = 4,
                              sigma = 2, gamma = 0.1)
```

---

variogram.intrinsic.spde

Variogram of intrinsic SPDE model

Description

Variogram $\gamma(s_0, s)$ of intrinsic SPDE model

$$(-\Delta)^{\beta/2}(\kappa^2 - \Delta)^{\alpha/2}(\tau u) = \mathcal{W}$$

with Neumann boundary conditions and a mean-zero constraint on a square $[0, L]^d$ for $d = 1$ or $d = 2$.

Usage

```
variogram.intrinsic.spde(
  s0 = NULL,
  s = NULL,
  kappa = NULL,
  alpha = NULL,
  beta = NULL,
  tau = 1,
  L = NULL,
  N = 100,
  d = NULL
)
```

Arguments

`s0`	The location where the variogram should be evaluated, either a double for 1d or a vector for 2d
`s`	A vector (in 1d) or matrix (in 2d) with all locations where the variogram is computed
`kappa`	Range parameter.
`alpha`	Smoothness parameter.
`beta`	Smoothness parameter.
`tau`	Precision parameter.
`L`	The side length of the square domain.
`N`	The number of terms in the Karhunen-Loeve expansion.
`d`	The dimension (1 or 2).

Details

The variogram is computed based on a Karhunen-Loeve expansion of the covariance function.

See Also

`intrinsic.matern.operators()`

Examples

```
if (requireNamespace("RSpectra", quietly = TRUE)) {
  x <- seq(from = 0, to = 10, length.out = 201)
  beta <- 1
  alpha <- 1
  kappa <- 1
  op <- intrinsic.matern.operators(
    kappa = kappa, tau = 1, alpha = alpha,
    beta = beta, loc_mesh = x, d = 1
  )
  # Compute and plot the variogram of the model
  Sigma <- op$A[,-1] %*% solve(op$Q[-1,-1], t(op$A[,-1]))
  One <- rep(1, times = ncol(Sigma))
  D <- diag(Sigma)
  Gamma <- 0.5 * (One %*% t(D) + D %*% t(One) - 2 * Sigma)
  k <- 100
  plot(x, Gamma[k, ], type = "l")
  lines(x,
    variogram.intrinsic.spde(x[k], x, kappa, alpha, beta, L = 10, d = 1),
    col = 2, lty = 2
  )
}
```

Index