

# Package ‘rSPDE’

May 25, 2023

**Type** Package

**Title** Rational Approximations of Fractional Stochastic Partial  
Differential Equations

**Version** 2.3.1

**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:10.48550/arXiv.2209.04670](https://doi.org/10.48550/arXiv.2209.04670)> for the covariance-based method and in Bolin and Kirchner (2020) <[doi:10.1080/10618600.2019.1665537](https://doi.org/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, numDeriv

**License** GPL (>= 3) | file LICENSE

**URL** <https://davidbolin.github.io/rSPDE/>

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Suggests** R.rsp, rmarkdown, INLA (>= 22.12.14), testthat, rgdal,  
ggplot2, lattice, splancs, optimParallel, inlabru (>= 2.7.0),  
sn, viridis, scoringRules, doParallel, foreach

**Additional\_repositories** <https://inla.r-inla-download.org/R/testing>

**BugReports** <https://github.com/davidbolin/rSPDE/issues>

**VignetteBuilder** R.rsp

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2023-05-25 11:40:02 UTC

**R topics documented:**

bru_get_mapper.inla_rspde . . . . .	3
construct.spde.matern.loglike . . . . .	4
cross_validation . . . . .	7
folded.matern.covariance.1d . . . . .	9
folded.matern.covariance.2d . . . . .	10
fractional.operators . . . . .	12
get.initial.values.rSPDE . . . . .	14
gg_df . . . . .	15
gg_df.rspde_result . . . . .	16
matern.covariance . . . . .	17
matern.operators . . . . .	17
operator.operations . . . . .	22
precision . . . . .	23
precision.inla_rspde . . . . .	24
predict.CBrSPDEobj . . . . .	25
predict.rSPDEobj . . . . .	27
predict.rspde_lme . . . . .	29
rational.order . . . . .	31
rational.order<- . . . . .	31
rational.type . . . . .	32
rational.type<- . . . . .	32
require.nowarnings . . . . .	33
rSPDE . . . . .	34
rSPDE.A1d . . . . .	34
rSPDE.construct.matern.loglike . . . . .	35
rSPDE.fem1d . . . . .	38
rSPDE.loglike . . . . .	39
rspde.make.A . . . . .	40
rspde.make.index . . . . .	42
rspde.matern . . . . .	44
rSPDE.matern.loglike . . . . .	47
rspde.matern.precision . . . . .	49
rspde.matern.precision.integer . . . . .	51
rspde.matern.precision.integer.opt . . . . .	53
rspde.matern.precision.opt . . . . .	54
rspde.mesh.project . . . . .	55
rspde.metric_graph . . . . .	56
rspde.result . . . . .	59
rspde_lme . . . . .	62
simulate.CBrSPDEobj . . . . .	64
simulate.rSPDEobj . . . . .	65
spde.matern.loglike . . . . .	66
spde.matern.operators . . . . .	68
summary.CBrSPDEobj . . . . .	71
summary.rSPDEobj . . . . .	72
summary.rspde_lme . . . . .	73

*bru\_get\_mapper.inla\_rspde* 3

summary.rspde\_result . . . . . 73  
update.CBrSPDEobj . . . . . 75  
update.rSPDEobj . . . . . 77

**Index** 80

---

*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 <i>inla_rspde</i> for which to construct or extract a mapper
...	Arguments passed on to other methods
mapper	A <i>bru_mapper_inla_rspde</i> 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])
coordinates(data_df) <- c("x1", "x2")
rspde_model <- rspde.matern(
  mesh = mesh_2d,
  nu_upper_bound = 2
)

cmp <- y ~ Intercept(1) +
      field(coordinates, model = rspde_model)

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

```

---

```
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  $\epsilon_i$  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,
  user_nu = NULL,
  user_m = NULL,
  log_scale = TRUE,
  return_negative_likelihood = TRUE
)
```

**Arguments**

object	The rational SPDE approximation, computed using <a href="#">matern.operators()</a>
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).
user_nu	If non-null, the shape parameter will be kept fixed in the returned likelihood.
user_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")
))

```

---

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 = 30,
  number_folds = 10,
  n_samples = 1000,
  return_scores_folds = FALSE,
  orientation_results = c("negative", "positive"),
  train_test_indexes = NULL,
  return_train_test = FALSE,
  parallelize_RP = FALSE,
  n_cores_RP = parallel::detectCores() - 1,
  true_CV = FALSE,
  save_settings = FALSE,
  print = TRUE,
  fit_verbose = FALSE
)

```

**Arguments**

models	A fitted model obtained from calling the <code>bru()</code> function or a list of models fitted with the <code>bru()</code> function.
model_names	A vector containing the names of the models to appear in the returned <code>data.frame</code> . If <code>NULL</code> , the names will be of the form <code>Model 1</code> , <code>Model 2</code> , and so on.
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 <code>k-fold</code> for <code>k</code> -fold cross-validation, in which case the parameter <code>k</code> should be provided, <code>loo</code> , for leave-one-out and <code>lpo</code> for leave-percentage-out, in this case, the parameter <code>percentage</code> should be given, and also the <code>number_folds</code> with the number of folds to be done. The default is <code>k-fold</code> .
k	The number of folds to be used in <code>k</code> -fold cross-validation. Will only be used if <code>cv_type</code> is <code>k-fold</code> .
percentage	The percentage (from 1 to 99) of the data to be used to train the model. Will only be used if <code>cv_type</code> is <code>lpo</code> .
number_folds	Number of folds to be done if <code>cv_type</code> is <code>lpo</code> .
n_samples	Number of samples to compute the posterior statistics to be used to compute the scores.
return_scores_folds	If <code>TRUE</code> , 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.
train_test_indexes	A list containing two entries <code>train</code> , which is a list whose elements are vectors of indexes of the training data, and <code>test</code> , 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 <code>return_train_test</code> to <code>TRUE</code> .
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.
parallelize_RP	Logical. Should the computation of CRPS and SCRPS be parallelized?
n_cores_RP	Number of cores to be used if <code>parallelize_rp</code> is <code>TRUE</code> .
true_cv	Should a <code>TRUE</code> cross-validation be performed? If <code>TRUE</code> the models will be fitted on the training dataset. If <code>FALSE</code> , the parameters will be kept fixed at the ones obtained in the result object.
save_settings	Logical. If <code>TRUE</code> , the settings used in the cross-validation will also be returned.
print	Should partial results be printed throughout the computation?
fit_verbos	Should INLA's run during cross-validation be verbose? Default is <code>FALSE</code>

**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 corresponding 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,  $\tau$  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  $m_\beta$  is the integer part of  $\beta$  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 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:

P1	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 <code>t(P1) %*% solve(C, P1)</code> .
type	String indicating the type of approximation.
P1.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 <code>metric_graph</code> object. To be used in case both <code>mesh</code> and <code>mesh.range</code> 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 $\tau$ . Will be used if <code>parameterization = 'spde'</code> .
B.kappa	Matrix with specification of log-linear model for $\kappa$ . Will be used if <code>parameterization = 'spde'</code> .
B.sigma	Matrix with specification of log-linear model for $\sigma$ . Will be used if <code>parameterization = 'matern'</code> .
B.range	Matrix with specification of log-linear model for $\rho$ , which is a range-like parameter (it is exactly the range parameter in the stationary case). Will be used if <code>parameterization = 'matern'</code> .
nu	The smoothness parameter.
parameterization	Which parameterization to use? <code>matern</code> uses range, std. deviation and nu (smoothness). <code>spde</code> uses kappa, tau and nu (smoothness). The default is <code>matern</code> .
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	<i>Data frame for result objects from R-INLA fitted models to be used in ggplot2</i>
-------	--

---

**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.



---

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"),
  fem_mesh_matrices = NULL
)

```

**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 inla 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".
fem_mesh_matrices	A list containing FEM-related matrices. The list should contain elements c0, g1, g2, g3, etc.

## 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  $m_\beta$  is the integer part of  $\beta$  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 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)

```

---

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)

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

Qsqr.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. `Qsqr.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	<i>Get the precision matrix of CBrSPDEobj objects</i>
-----------	---

---

### Description

Function to get the precision matrix of a CBrSPDEobj object

### Usage

```
precision(object, ...)

## S3 method for class 'CBrSPDEobj'
precision(
  object,
  user_nu = NULL,
  user_kappa = NULL,
  user_sigma = NULL,
  user_range = NULL,
  user_tau = NULL,
  user_m = NULL,
  ...
)
```

### Arguments

object	The covariance-based rational SPDE approximation, computed using <a href="#">matern.operators()</a>
...	Currently not used.
user_nu	If non-null, update the shape parameter of the covariance function.
user_kappa	If non-null, update the range parameter of the covariance function.
user_sigma	If non-null, update the standard deviation of the covariance function.
user_range	If non-null, update the range parameter of the covariance function.
user_tau	If non-null, update the parameter tau.
user_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.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 supplied as a vector. For stationary models, we have $\theta = c(\log(\tau), \log(\kappa), \nu)$ . For nonstationary models, we have $\theta = c(\theta_1, \theta_2, \dots, \theta_n, \nu)$ .
...	Currently not used.

**Value**

The precision matrix.

**See Also**

[precision.CBrSPDEobj\(\)](#), [matern.operators\(\)](#)

---

predict.CBrSPDEobj	<i>Prediction of a fractional SPDE using the covariance-based rational SPDE approximation</i>
--------------------	---

---

**Description**

The function is used for computing kriging predictions based on data  $Y_i = u(s_i) + \epsilon_i$ , where  $\epsilon$  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 <code>matern.operators()</code>
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 <code>sampling</code> 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).

**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.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  $\epsilon$  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 <code>fractional.operators()</code> , <code>matern.operators()</code> , or <code>spde.matern.operators()</code> .
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 <code>sampling</code> 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 <code>sampling</code> 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,
  data = NULL,
  loc = NULL,
  mesh = FALSE,
  which_repl = NULL,
  compute_variances = FALSE,
  posterior_samples = FALSE,

```

```

n_samples = 100,
only_latent = FALSE,
edge_number = "edge_number",
distance_on_edge = "distance_on_edge",
normalized = FALSE,
return_as_list = FALSE,
return_original_order = TRUE,
...
)

```

### Arguments

object	The fitted object with the <code>rspde_lme()</code> function
data	A <code>data.frame</code> 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 <code>data.frame</code> , a matrix or a character vector, that contains the names of the columns of the coordinates of the locations. For models using <code>metric_graph</code> objects, please use <code>edge_number</code> and <code>distance_on_edge</code> instead.
mesh	Obtain predictions for mesh nodes? The graph must have a mesh, and either <code>only_latent</code> is set to <code>TRUE</code> or the model does not have covariates.
which_repl	Which replicates to use? If <code>NULL</code> all replicates will be used.
compute_variances	Set to also <code>TRUE</code> to compute the kriging variances.
posterior_samples	If <code>TRUE</code> , posterior samples will be returned.
n_samples	Number of samples to be returned. Will only be used if <code>sampling</code> is <code>TRUE</code> .
only_latent	Should the posterior samples and predictions be only given to the latent model?
edge_number	Name of the variable that contains the edge number, the default is <code>edge_number</code> .
distance_on_edge	Name of the variable that contains the distance on edge, the default is <code>distance_on_edge</code> .
normalized	Are the distances on edges normalized?
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?
...	Not used.

---

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.

---

<code>rational.type</code>	<i>Get type of rational approximation.</i>
----------------------------	--

---

**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.

---

<code>rational.type&lt;-</code>	<i>Changing the type of the rational approximation</i>
---------------------------------	--

---

**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

*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  $\mathbb{R}$ . 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.

rSPDE.A1d

*Observation matrix for finite element discretization on  $\mathbb{R}$* **Description**

A finite element discretization on  $\mathbb{R}$  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  $s = (s_1, \dots, s_m)$  in the domain through  $A_{ij} = \varphi_j(s_i)$ .

**Usage**

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

**Arguments**

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

**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.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  $\epsilon_i$  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,
  user_nu = NULL,
  user_tau = NULL,
```

```

user_kappa = NULL,
user_sigma = NULL,
user_range = NULL,
parameterization = c("spde", "matern"),
user_m = NULL,
log_scale = TRUE,
return_negative_likelihood = TRUE
)

```

### Arguments

<code>object</code>	The rational SPDE approximation, computed using <a href="#">matern.operators()</a>
<code>Y</code>	The observations, either a vector or a matrix where the columns correspond to independent replicates of observations.
<code>A</code>	An observation matrix that links the measurement location to the finite element basis.
<code>sigma.e</code>	If non-null, the standard deviation of the measurement noise will be kept fixed in the returned likelihood.
<code>mu</code>	Expectation vector of the latent field (default = 0).
<code>user_nu</code>	If non-null, the shape parameter will be kept fixed in the returned likelihood.
<code>user_tau</code>	If non-null, the tau parameter will be kept fixed in the returned likelihood. (Replaces sigma)
<code>user_kappa</code>	If non-null, the range parameter will be kept fixed in the returned likelihood.
<code>user_sigma</code>	If non-null, the standard deviation will be kept fixed in the returned likelihood.
<code>user_range</code>	If non-null, the range parameter will be kept fixed in the returned likelihood. (Replaces kappa)
<code>parameterization</code>	If <code>spde</code> , then one will use the parameters <code>tau</code> and <code>kappa</code> . If <code>matern</code> , then one will use the parameters <code>sigma</code> and <code>range</code> .
<code>user_m</code>	If non-null, update the order of the rational approximation, which needs to be a positive integer.
<code>log_scale</code>	Should the parameters be evaluated in log-scale?
<code>return_negative_likelihood</code>	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.  
`C`                      The mass matrix.

**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.loglike	<i>Object-based log-likelihood function for latent Gaussian fractional SPDE model</i>
---------------	---

---

### 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  $\epsilon_i$  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 <a href="#">fractional.operators()</a> , <a href="#">matern.operators()</a> , or <a href="#">spde.matern.operators()</a> .
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"
)

# 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 = 2,
  nu = NULL,
  index = NULL,
  group = NULL,
  repl = 1L,
  n.group = NULL,
  n.repl = NULL
)

```



**Arguments**

mesh	An <code>inla.mesh</code> , an <code>inla.mesh.1d</code> object or a <code>metric_graph</code> 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 <code>inla.spde.make.A</code> . 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 <code>seq_len(nrow(A.loc))</code> .
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 = 2,
  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 <code>inla.mesh</code> , an <code>inla.mesh.1d</code> object or a <code>metric_graph</code> 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 = 4,
  rspde.order = 2,
  nu = NULL,
  B.sigma = matrix(c(0, 1, 0), 1, 3),
  B.range = matrix(c(0, 0, 1), 1, 3),
  parameterization = c("matern", "spde", "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 <code>inla.mesh</code> or an <code>inla.mesh.1d</code> object. Otherwise, should be a list containing elements <code>d</code> , the dimension, <code>C</code> , the mass matrix, and <code>G</code> , the stiffness matrix.
nu.upper.bound	Upper bound for the smoothness parameter.
rspde.order	The order of the covariance-based rational SPDE approach.
nu	If <code>nu</code> is set to a parameter, <code>nu</code> will be kept fixed and will not be estimated. If <code>nu</code> is <code>NULL</code> , it will be estimated.
B.sigma	Matrix with specification of log-linear model for $\sigma$ (for 'matern' parameterization) or for $\sigma^2$ (for 'matern2' parameterization). Will be used if <code>parameterization = 'matern'</code> or <code>parameterization = 'matern2'</code> .
B.range	Matrix with specification of log-linear model for $\rho$ , which is a range-like parameter (it is exactly the range parameter in the stationary case). Will be used if <code>parameterization = 'matern'</code> or <code>parameterization = 'matern2'</code> .
parameterization	Which parameterization to use? <code>matern</code> uses range, std. deviation and <code>nu</code> (smoothness). <code>spde</code> uses <code>kappa</code> , <code>tau</code> and <code>nu</code> (smoothness). <code>matern2</code> uses range-like ( <code>1/kappa</code> ), variance and <code>nu</code> (smoothness). The default is <code>matern</code> .
B.tau	Matrix with specification of log-linear model for $\tau$ . Will be used if <code>parameterization = 'spde'</code> .
B.kappa	Matrix with specification of log-linear model for $\kappa$ . Will be used if <code>parameterization = 'spde'</code> .
start.nu	Starting value for <code>nu</code> .
start.theta	Starting values for the model parameters. In the stationary case, if <code>parameterization='matern'</code> , then <code>theta[1]</code> is the <code>std.dev</code> and <code>theta[2]</code> is the range parameter. If <code>parameterization = 'spde'</code> , then <code>theta[1]</code> is <code>tau</code> and <code>theta[2]</code> is <code>kappa</code> .
prior.nu	a list containing the elements <code>mean</code> and <code>prec</code> for beta distribution, or <code>loglocation</code> and <code>logscale</code> for a truncated lognormal distribution. <code>loglocation</code> stands for the location parameter of the truncated lognormal distribution in the log scale. <code>prec</code> stands for the precision of a beta distribution. <code>logscale</code> 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 <code>theta</code> .
theta.prior.prec	A precision matrix for the prior of <code>theta</code> .
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 <code>kappa</code> to be used for the priors and for the starting values.
prior.tau.mean	Prior <code>tau</code> to be used for the priors and for the starting values.

<code>start.lstd.dev</code>	Starting value for log of std. deviation. Will not be used if <code>start.ltau</code> is non-null. Will be only used in the stationary case and if <code>parameterization = 'matern'</code> .
<code>start.lrange</code>	Starting value for log of range. Will not be used if <code>start.lkappa</code> is non-null. Will be only used in the stationary case and if <code>parameterization = 'matern'</code> .
<code>start.ltau</code>	Starting value for log of tau. Will be only used in the stationary case and if <code>parameterization = 'spde'</code> .
<code>start.lkappa</code>	Starting value for log of kappa. Will be only used in the stationary case and if <code>parameterization = 'spde'</code> .
<code>prior.theta.param</code>	Should the lognormal prior be on theta or on the SPDE parameters (tau and kappa on the stationary case)?
<code>prior.nu.dist</code>	The distribution of the smoothness parameter. The current options are "beta" or "lognormal". The default is "lognormal".
<code>nu.prec.inc</code>	Amount to increase the precision in the beta prior distribution. Check details below.
<code>type.rational.approx</code>	Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".
<code>debug</code>	INLA debug argument
<code>shared_lib</code>	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.
<code>...</code>	Only being used internally.
<code>prior.kappa</code>	a list containing the elements <code>meanlog</code> and <code>sdlog</code> , that is, the mean and standard deviation on the log scale.
<code>prior.tau</code>	a list containing the elements <code>meanlog</code> and <code>sdlog</code> , that is, the mean and standard deviation on the log scale.
<code>prior.range</code>	a list containing the elements <code>meanlog</code> and <code>sdlog</code> , that is, the mean and standard deviation on the log scale. Will not be used if <code>prior.kappa</code> is non-null.
<code>prior.std.dev</code>	a list containing the elements <code>meanlog</code> and <code>sdlog</code> , that is, the mean and standard deviation on the log scale. Will not be used if <code>prior.tau</code> is non-null.

## 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  $\epsilon_i$  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,
  user_nu = NULL,
  user_kappa = NULL,
  user_sigma = NULL,
  user_range = NULL,
  user_tau = NULL,
  user_m = NULL
)
```

## Arguments

object	The rational SPDE approximation, computed using <code>matern.operators()</code>
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).
user_nu	If non-null, update the shape parameter of the covariance function.
user_kappa	If non-null, update the range parameter of the covariance function.
user_sigma	If non-null, update the standard deviation of the covariance function.
user_range	If non-null, update the range parameter of the covariance function.
user_tau	If non-null, update the parameter tau.
user_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, user_kappa = kappa, user_sigma = sigma,
    user_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.mesh.project	<i>Calculate a lattice projection to/from an inla.mesh for rSPDE objects</i>
--------------------	--

---

**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 = 2,
  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 = 2,
  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 = 2, 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 = 2,
  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 <code>metric_graph</code> . 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 $\sigma$ . Will be used if parameterization = 'matern'.
B.range	Matrix with specification of log-linear model for $\rho$ , 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? <code>matern</code> uses range, std. deviation and nu (smoothness). <code>spde</code> uses kappa, tau and nu (smoothness). The default is <code>matern</code> .
B.tau	Matrix with specification of log-linear model for $\tau$ . Will be used if parameterization = 'spde'.
B.kappa	Matrix with specification of log-linear model for $\kappa$ . Will be used if parameterization = 'spde'.
start.nu	Starting value for nu.

<code>start.theta</code>	Starting values for the model parameters. In the stationary case, if <code>parameterization='matern'</code> , then <code>theta[1]</code> is the <code>std.dev</code> and <code>theta[2]</code> is the range parameter. If <code>parameterization = 'spde'</code> , then <code>theta[1]</code> is <code>tau</code> and <code>theta[2]</code> is <code>kappa</code> .
<code>prior.nu</code>	a list containing the elements <code>mean</code> and <code>prec</code> for beta distribution, or <code>loglocation</code> and <code>logscale</code> for a truncated lognormal distribution. <code>loglocation</code> stands for the location parameter of the truncated lognormal distribution in the log scale. <code>prec</code> stands for the precision of a beta distribution. <code>logscale</code> stands for the scale of the truncated lognormal distribution on the log scale. Check details below.
<code>theta.prior.mean</code>	A vector for the mean priors of <code>theta</code> .
<code>theta.prior.prec</code>	A precision matrix for the prior of <code>theta</code> .
<code>prior.std.dev.nominal</code>	Prior std. deviation to be used for the priors and for the starting values.
<code>prior.range.nominal</code>	Prior range to be used for the priors and for the starting values.
<code>prior.kappa.mean</code>	Prior <code>kappa</code> to be used for the priors and for the starting values.
<code>prior.tau.mean</code>	Prior <code>tau</code> to be used for the priors and for the starting values.
<code>start.lstd.dev</code>	Starting value for log of std. deviation. Will not be used if <code>start.ltau</code> is non-null. Will be only used in the stationary case and if <code>parameterization = 'matern'</code> .
<code>start.lrange</code>	Starting value for log of range. Will not be used if <code>start.lkappa</code> is non-null. Will be only used in the stationary case and if <code>parameterization = 'matern'</code> .
<code>start.ltau</code>	Starting value for log of <code>tau</code> . Will be only used in the stationary case and if <code>parameterization = 'spde'</code> .
<code>start.lkappa</code>	Starting value for log of <code>kappa</code> . Will be only used in the stationary case and if <code>parameterization = 'spde'</code> .
<code>prior.theta.param</code>	Should the lognormal prior be on <code>theta</code> or on the SPDE parameters ( <code>tau</code> and <code>kappa</code> on the stationary case)?
<code>prior.nu.dist</code>	The distribution of the smoothness parameter. The current options are "beta" or "lognormal". The default is "lognormal".
<code>nu.prec.inc</code>	Amount to increase the precision in the beta prior distribution. Check details below.
<code>type.rational.approx</code>	Which type of rational approximation should be used? The current types are "chebfun", "brasil" or "chebfunLB".
<code>shared_lib</code>	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.
<code>prior.kappa</code>	a list containing the elements <code>meanlog</code> and <code>sdlog</code> , 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	<i>rSPDE result extraction from INLA estimation results</i>
--------------	---

---

**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)
```

**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?

**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\_lme

*rspDE linear mixed effects models*


---

## Description

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

## Usage

```
rspde_lme(
  formula,
  loc,
  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 <code>metric_graph</code> , 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, <code>loc</code> must be either a <code>matrix</code> or <code>data.frame</code> 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.
<code>data</code>	A <code>data.frame</code> containing the data to be used.
<code>model</code>	Either an object generated by <code>matern.operators()</code> or <code>spde.matern.operators()</code> . If NULL, a simple linear regression will be performed.
<code>repl</code>	Vector indicating the replicate of each observation. If NULL it will assume there is only one replicate.
<code>which_repl</code>	Which replicates to use? If NULL all replicates will be used.
<code>optim_method</code>	The method to be used with <code>optim</code> function.
<code>use_data_from_graph</code>	Logical. Only for models generated from graphs from <code>metric_graph</code> class. In this case, should the data, the locations and the replicates be obtained from the graph object?
<code>starting_values_latent</code>	A vector containing the starting values for the latent model. If the latent model was generated by <code>matern.operators()</code> , then the starting values should be provided as a vector of the form <code>c(tau,kappa)</code> . If the model was generated by <code>spde.matern.operators()</code> , then the starting values should be provided as a vector containing the nonstationary parameters.
<code>start_sigma_e</code>	Starting value for the standard deviation of the measurement error.
<code>start_alpha</code>	Starting value for the smoothness parameter. Will be used if <code>start_nu</code> is not given.
<code>alpha</code>	If NULL, the smoothness parameter will be estimated, otherwise the smoothness parameter will be kept fixed at the provided value. Will be used if <code>nu</code> is not given.
<code>start_nu</code>	Starting value for the smoothness parameter.
<code>nu</code>	If NULL, the smoothness parameter will be estimated, otherwise the smoothness parameter will be kept fixed at the provided value.
<code>nu_upper_bound</code>	A parameter that limits the maximum value that <code>nu</code> can assume.
<code>rspde_order</code>	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.
<code>parallel</code>	logical. Indicating whether to use <code>optimParallel</code> or not.
<code>n_cores</code>	Number of cores to be used if <code>parallel</code> is true.
<code>optim_controls</code>	Additional controls to be passed to <code>optim</code> or <code>optimParallel</code> .
<code>improve_hessian</code>	Should a more precise estimate of the hessian be obtained? Turning on might increase the overall time.
<code>hessian_args</code>	List of controls to be used if <code>improve_hessian</code> is TRUE. The list can contain the arguments to be passed to the <code>method.args</code> argument in the <code>numDeriv::hessian</code> function. See the help of the <code>hessian</code> function in <code>numDeriv</code> 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	<i>Simulation of a fractional SPDE using the covariance-based rational SPDE approximation</i>
---------------------	---

---

**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,
  user_nu = NULL,
  user_kappa = NULL,
  user_sigma = NULL,
  user_range = NULL,
  user_tau = NULL,
  user_theta = NULL,
  user_m = NULL,
  ...
)
```

**Arguments**

object	The covariance-based rational SPDE approximation, computed using <a href="#">matern.operators()</a>
nsim	The number of simulations.
seed	An object specifying if and how the random number generator should be initialized ('seeded').
user_nu	If non-null, update the shape parameter of the covariance function.
user_kappa	If non-null, update the range parameter of the covariance function.
user_sigma	If non-null, update the standard deviation of the covariance function.
user_range	If non-null, update the range parameter of the covariance function.
user_tau	If non-null, update the parameter tau.
user_theta	For non-stationary models. If non-null, update the vector of parameters.
user_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.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 <a href="#">fractional.operators()</a> , <a href="#">matern.operators()</a> , or <a href="#">spde.matern.operators()</a> .
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")
```

---

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,
```

```

    user_nu = NULL,
    user_kappa = NULL,
    user_tau = NULL,
    user_theta = NULL,
    user_m = NULL
  )

```

### Arguments

object	The rational SPDE approximation, computed using <a href="#">spde.matern.operators()</a>
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).
user_nu	If non-null, the shape parameter will be kept fixed in the returned likelihood.
user_kappa	If non-null, updates the range parameter.
user_tau	If non-null, updates the parameter tau.
user_theta	If non-null, updates the parameter theta (that connects tau and kappa to the model matrices in object).
user_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  $\epsilon_i$  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]),
                             user_nu = exp(theta[3]),
                             user_kappa = exp(theta[2]),
                             user_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 $\tau$ . Will be used if parameterization = 'spde'.
B.kappa	Matrix with specification of log-linear model for $\kappa$ . Will be used if parameterization = 'spde'.
B.sigma	Matrix with specification of log-linear model for $\sigma$ . Will be used if parameterization = 'matern'.
B.range	Matrix with specification of log-linear model for $\rho$ , 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  $m_\beta$  is the integer part of  $\beta$  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 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

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  $\nu$ .

### 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))

# compute rational approximation
op <- spde.matern.operators(
  kappa = kappa, tau = tau, nu = nu,
  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 <code>matern.operators()</code> .
...	further arguments passed to or from other methods.
x	an object of class "summary.CBrSPDEobj", usually, a result of a call to <code>summary.CBrSPDEobj()</code> .

**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.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 <a href="#">fractional.operators()</a> , <a href="#">matern.operators()</a> , or <a href="#">spde.matern.operators()</a> .
...	further arguments passed to or from other methods.
x	an object of class "summary.rSPDEobj", usually, a result of a call to <a href="#">summary.rSPDEobj()</a> .



---

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")})
```

---

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,
  user_nu = NULL,
  user_alpha = NULL,
  user_kappa = NULL,
  user_tau = NULL,
  user_sigma = NULL,
  user_range = NULL,
  user_theta = NULL,
  user_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 <a href="#">matern.operators()</a>
user_nu	If non-null, update the shape parameter of the covariance function. Will be used if parameterization is 'matern'.
user_alpha	If non-null, update the fractional SPDE order parameter. Will be used if parameterization is 'spde'.
user_kappa	If non-null, update the parameter kappa of the SPDE. Will be used if parameterization is 'spde'.
user_tau	If non-null, update the parameter tau of the SPDE. Will be used if parameterization is 'spde'.

user_sigma	If non-null, update the standard deviation of the covariance function. Will be used if parameterization is 'matern'.
user_range	If non-null, update the range parameter of the covariance function. Will be used if parameterization is 'matern'.
user_theta	For non-stationary models. If non-null, update the vector of parameters.
user_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 <code>rspde_lme()</code> function and will not provide neither <code>graph</code> nor <code>mesh</code> . Only works for 1d data. Does not work for metric graphs. For metric graphs you should supply the <code>graph</code> using the <code>graph</code> argument.
graph	An optional <code>metric_graph</code> 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 <code>mesh</code> and <code>graph</code> 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, user_kappa = 20)
op_cov

```

---

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,
  user_nu = NULL,
  user_alpha = NULL,
  user_kappa = NULL,
  user_sigma = NULL,
  user_range = NULL,
  user_tau = NULL,
  user_theta = NULL,
  user_m = NULL,
  mesh = NULL,
  loc_mesh = NULL,
  graph = NULL,
  range_mesh = NULL,
  parameterization = NULL,
  ...
)

```

**Arguments**

object	The operator-based rational SPDE approximation, computed using <code>matern.operators()</code> with <code>type="operator"</code>
user_nu	If non-null, update the shape parameter of the covariance function.
user_alpha	If non-null, update the fractional order.
user_kappa	If non-null, update the range parameter of the covariance function.
user_sigma	If non-null, update the standard deviation of the covariance function.
user_range	If non-null, update the range parameter of the covariance function.
user_tau	If non-null, update the parameter tau.
user_theta	If non-null, update the parameter theta, that connects tau and kappa to the model matrices.
user_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 <code>rspde_lme()</code> function and will not provide neither <code>graph</code> nor <code>mesh</code> . Only works for 1d data. Does not work for metric graphs. For metric graphs you should supply the graph using the <code>graph</code> argument.
graph	An optional <code>metric_graph</code> 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 <code>mesh</code> and <code>graph</code> 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, user_kappa = 20)
op
```

# Index

`bru_get_mapper.inla_rspde`, 3

`construct.spde.matern.loglike`, 4

`cross_validation`, 7

`folded.matern.covariance.1d`, 9

`folded.matern.covariance.2d`, 10

`fractional.operators`, 12

`fractional.operators()`, 20, 28, 34, 39, 65, 70, 72

`get.initial.values.rSPDE`, 14

`gg_df`, 15

`gg_df.rspde_result`, 16

`ibm_jacobian.bru_mapper_inla_rspde`  
(`bru_get_mapper.inla_rspde`), 3

`ibm_n.bru_mapper_inla_rspde`  
(`bru_get_mapper.inla_rspde`), 3

`ibm_values.bru_mapper_inla_rspde`  
(`bru_get_mapper.inla_rspde`), 3

`matern.covariance`, 17

`matern.operators`, 17

`matern.operators()`, 5, 13, 20, 23–26, 28, 34, 36, 39, 47, 48, 64, 65, 70–72, 75, 76, 78

`operator.operations`, 22

`operator.operations()`, 13, 20, 70

`Pl.mult` (`operator.operations`), 22

`Pl.solve` (`operator.operations`), 22

`Pr.mult` (`operator.operations`), 22

`Pr.solve` (`operator.operations`), 22

`precision`, 23

`precision.CBrSPDEobj()`, 25

`precision.inla_rspde`, 24

`predict.CBrSPDEobj`, 25

`predict.CBrSPDEobj()`, 5, 36, 48

`predict.rspde_lme`, 29

`predict.rSPDEobj`, 27

`print.CBrSPDEobj` (`summary.CBrSPDEobj`), 71

`print.rSPDEobj` (`summary.rSPDEobj`), 72

`print.summary.CBrSPDEobj`  
(`summary.CBrSPDEobj`), 71

`print.summary.rSPDEobj`  
(`summary.rSPDEobj`), 72

`Q.mult` (`operator.operations`), 22

`Q.solve` (`operator.operations`), 22

`Qsqrt.mult` (`operator.operations`), 22

`Qsqrt.solve` (`operator.operations`), 22

`rational.order`, 31

`rational.order<=`, 31

`rational.type`, 32

`rational.type<=`, 32

`require()`, 33

`require.nowarnings`, 33

`rSPDE`, 34

`rSPDE.A1d`, 34

`rSPDE.A1d()`, 38

`rSPDE.construct.matern.loglike`, 35

`rSPDE.fem1d`, 38

`rSPDE.fem1d()`, 35

`rSPDE.loglike`, 39

`rSPDE.loglike()`, 67

`rspde.make.A`, 40

`rspde.make.index`, 42

`rspde.matern`, 44

`rspde.matern()`, 34

`rSPDE.matern.loglike`, 47

`rspde.matern.precision`, 49

`rspde.matern.precision.integer`, 51

`rspde.matern.precision.integer.opt`, 53

`rspde.matern.precision.opt`, 54

`rspde.mesh.project`, 55

`rspde.mesh.projector`  
(`rspde.mesh.project`), 55



rspde.metric\_graph, 56  
rspde.result, 59  
rspde\_lme, 62

Sigma.mult (operator.operations), 22  
Sigma.solve (operator.operations), 22  
simulate.CBrSPDEobj, 64  
simulate.CBrSPDEobj(), 24, 66, 76  
simulate.rSPDEobj, 65  
simulate.rSPDEobj(), 78  
spde.matern.loglike, 66  
spde.matern.loglike(), 39  
spde.matern.operators, 68  
spde.matern.operators(), 13, 20, 28, 34,  
39, 65, 67, 70, 72  
summary.CBrSPDEobj, 71  
summary.CBrSPDEobj(), 71  
summary.rspde\_lme, 73  
summary.rspde\_result, 73  
summary.rSPDEobj, 72  
summary.rSPDEobj(), 72

update.CBrSPDEobj, 75  
update.rSPDEobj, 77